Regret-Based Reward Elicitation for Markov Decision Processes

Author: Kevin REGAN

Supervisor: Dr. Craig BOUTILIER

August 30, 2010
1 Introduction / Overview

Computational approaches to planning under uncertainty allow optimal policies to be found in complex decision making domains. The Markov decision process (MDP) has proven to be a useful model for such domains; yet it often requires the specification of a large set of model parameters. Model dynamics quantifying the stochastic effects of actions can be learned from the environment. A reward function quantifying which states and actions are desirable must also be specified; however, this reward function reflects the subjective preferences of the user, and as such is not directly observable. The specification of reward can be problematic for a number of reasons. Translating qualitative preferences about which states/actions and “good” and “bad” into a precise numerical reward is a cognitively (or computationally) demanding task. For many real-world MDPs this process can be time consuming since this precise numerical reward must be specified for many states and actions. Furthermore, when preferences vary among individuals, with each new user the reward specification process must be undertaken anew.

An analog to reward specification can be found in the task of determining utility functions in single step decision support scenarios. There is an important trade-off between the improvement in decision making made possible through eliciting utility information and the burden of eliciting information. This burden can be cognitive (requiring a user to answer questions), monetary (paying individuals to analyze business outcomes), computational (solving optimization problems or running simulations), or otherwise. It has been recognized that the burden imposed by elicitation is not always justified by the improvement in decision quality it offers [8], motivating methods for making decisions with partial preference information. The minimax regret criterion [32] has been advocated for robust decision making with incompletely or imprecisely specified utility functions since it guarantees worst-case bounds on the quality of the decision [8, 31]. Techniques for elicitation using minimax regret have proven effective in allowing optimal or near-optimal decisions (or policies) to be found without full utility specification [8] and provide inspiration for approaches to specifying reward functions.

My research goal is to develop a minimax regret-based framework for the incremental elicitation of reward functions for MDPs that is cognitively and computationally effective.

Figure 1 shows the components of the framework and diagrams the steps undertaken during a round of elicitation. In Step A, given an MDP with imprecise reward, a robust policy is computed along with a measure of the quality of the robust policy. If the user is unsatisfied with the quality of the policy, then Step B selects a query to ask the user. The selection of a query involves specifying both the type of query (e.g., asking about the reward for a specific state and action, or eliciting feedback about an entire policy), as well as the parameters of the query (e.g., selecting the specific state and action, or policy). Step C applies the information from the query response to further refine the reward function. The updated imprecise reward specification allows for an improved policy to be computed. The process can then iterate until a desired level of quality is reached. Related work has tackled the robust policy computation required for Step A using a range of robustness criteria [14, 22, 36], however this work fails to integrate the robust policy computation into a framework for interactively specifying the reward function. Our work is the first to propose the use of the minimax regret criterion for computing robust policies [27] and to show how it can be used to guide the interactive elicitation of reward.

The design of this framework involves a number of considerations. The first consideration is the efficient computation of minimax regret optimal policies. In most cases, the aim is to directly involve the user in a real-time interaction. The amount of time a user is willing to wait after a query response imposes a strict upper limit on computation time. This creates a challenge due to the complexity of the minimax regret computation which requires simultaneously optimizing with respect to the imprecise reward and the
space of optimal MDP policies. Finding the optimal policy for a fixed reward function is a non-trivial procedure that involves solving a linear program (LP) (or iterative computing the solution through dynamic programming) [26]. To compute minimax regret we must essentially assess the value of the optimal policy for all potential reward functions. The time required by straightforward approaches to this computation [27, 28, 36] quickly exceeds the amount of time [28] that even the most patient user is willing to wait between a query response and the opportunity to continue the elicitation. This has motivated approaches that involve extensive offline computation in order to increase the efficiency of the required online computation [29, 30, 36]. Also interesting are methods to approximate minimax regret, especially when paired with a reasonable error bound establishing an upper bound on minimax regret for the user [30]. In the absence of an exact measure of minimax regret, the upper bound continues to constitute a valid guarantee on the worst case regret a user may experience, allowing the user to end the elicitation when the regret bound reaches a satisfactory level.

The second major consideration is the effective selection of queries in terms of both the type of query and its parameters. We are interested in the effectiveness of a selected query along two distinct dimensions. First, a query must be conceptually tractable, allowing the user to easily understand and reason about the question posed by the query. Second, a query must engender responses that quickly lead to better policies with respect to regret.

Selecting effective queries is known to be a challenging problem for utility elicitation in the single-step decision making setting [8]. The elicitation of reward for MDPs is complicated by the sequential nature of the underlying decision problem, motivating the development of novel types of queries. However, each type of query must strike a compromise between the ease with which a user can reason about the query and the efficiency with which the query response can reduce regret. For instance, queries about entire policies have the potential to quickly reduce regret, however they require a user to reason about many actions in conjunction, where each action is conditional on a different context (i.e., the state of the MDP).

Even queries that simply refer to the reward for a specific state or action introduce the danger of conflating reward, which captures immediate utility, and value, which captures accumulated utility over time.
Given an established query type, at each step of elicitation we wish to find a setting of query parameters to quickly reduce regret. As we will see in Section 4.2, optimally selecting these parameters is computationally intensive (since it essentially involves solving minimax regret for each setting of the query parameters). However, there are several myopic and heuristic approaches which hold promise for effectively eliciting reward [8, 28, 31, 34, 35]. One potential method gleans information from the current minimax regret optimal policy [28]; here the regret computation is used not only to provide the current best policy along with a regret guarantee, it is also used to guide the selection of the next query.

This document examines each component of the framework outlining ongoing and completed work and describes future directions. Section 3 discusses minimax regret computation, Section 4 discusses several aspects of elicitation, and Section 5 discusses several domains suitable for reward elicitation. We first review some background on MDPs, setting out notation and giving a precise definition of minimax regret. Further details of the completed research can be found in my research papers both published [28, 29] and a tech report with more recent results [30]. These papers have been included as appendices.

2 Background

MDPs We assume an infinite horizon MDP \( \langle S, A, \{P_{sa}\}, \gamma, \beta, r \rangle \), with finite state set \( S \), finite action set \( A \), transition distributions \( P_{sa}(\cdot) \) over next states (given action \( a \) taken in state \( s \)), discount factor \( \gamma \), initial state distribution \( \beta \), and reward function \( r : S \times A \to \mathbb{R} \). A policy \( \pi \) maps states to actions. The value (denoting the expected total reward) of executing a policy is:

\[
V^\pi = \sum_{s_1 \in S} \beta(s_1) \mathbb{E} \left[ \sum_{i=1}^{\infty} \gamma^{i-1} r(s_i, \pi(s_i)) \right],
\]

Here the expectation is over the sequence of states induced by the policy \( \pi \). Solving an MDP finds the optimal policy \( \pi^* \), where \( V^* = V^{\pi^*} \geq V^\pi, \forall \pi \).

A policy \( \pi \) induces occupancy frequencies \( f^\pi(s, a) \) which reflect the total discounted probability of being in state \( s \) and taking action \( a \). Given such frequencies \( f^\pi \), the policy can be recovered via \( \pi(s, a) = f^\pi(s, a) / \sum_{a'} f^\pi(s, a') \). Due to this direct correspondence, we often treat occupancy frequencies and policies interchangeably in our discussion. The MDP value function can be expressed in terms of occupancy frequencies as follows:

\[
V^\pi = \sum_{s \in S} \sum_{a \in A} f^\pi(s, a) r(s, a)
\]

The set \( \mathcal{F} \) of valid occupancy probabilities for a fixed MDP can be characterized with the following constraints [26]:

\[
\sum_{a \in A} f(s_0, a) - \gamma \sum_{s \in S} \sum_{a \in A} \Pr(s_0|s, a) f(s, a) = \beta(s_0) \quad \forall \ s_0 \in S
\]

For ease of exposition we make use of the following vector notation: \( \mathbf{r} \) is an \(|S||A|\) vector with entries \( r(s, a) \); \( \mathbf{f} \) is an \(|S||A|\) vector with entries \( f(s, a) \); and \( \mathbf{P} \) is an \(|S||A| \times |S| \) transition matrix where entry \( P_{sa,s'} \) is the probability \( P_{sa}(s') \) of transitioning to \( s' \) given current state \( s \) and action \( a \). We can thus express the MDP value function in terms of occupancy frequencies as: \( V^\pi = \mathbf{f} \cdot \mathbf{r} \).
Bellman [1] established the following recursive expression of the optimal value function $V^*$:

$$V^*(s) = \max_{a \in A} r(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V^*(s') \quad \forall s \in S$$ (1)

This relationship is the basis of two dynamic programming approaches to computing the optimal policy. The first, value iteration [1], estimates the value of the optimal action for each state and iteratively improves this estimate using Eq. (1). This estimate will linearly converge to the optimal value function, and the optimal policy can be recovered from the actions that maximize Eq. (1) for each state. The second approach, policy iteration [19], finds the policy directly. It comprises two steps: the policy evaluation step in which the value of the current policy is computed, and the policy improvement step which finds the value of each action being executed initially and then following the current policy thereafter. The current policy is updated with the actions that offer the greatest improvement in value, and the process is repeated.

The optimal value function can also be found using a single linear program that directly solves the system of linear equations that encode the value function [26]. The dual formulation of this linear program will find the occupancy frequencies corresponding to the optimal policy [26]. It should be noted that this dual formulation is the basis for many of the linear programming formulations in Section 3. Both of these formulations allow for the optimization to be performed by one of the many fast LP solvers currently available.

**IRMDPs** An imprecise reward MDP (IRMDP) is defined by the tuple $\langle S, A, \{P_{sa}\}, \gamma, \beta, R \rangle$, which replaces the reward $r$ by a set of feasible reward functions $R$ [22, 36]. The set $R$ naturally arises in many scenarios (e.g., observations of user behaviour, partial elicitation of preferences, or information from domain experts). We assume that $R$ is a bounded, convex polytope defined by a set of linear constraints $\{ r \mid A r \leq b \}$ and use $|R|$ to denote the number of constraints. Section 4 will describe how information about the reward function (collected in the example scenarios) naturally induce these linear constraints.

**Minimax Regret** We define $R(f, r)$ to be the regret of policy $f$ w.r.t. reward function $r$. Regret measures the difference in value between $f$ and the optimal policy given $r$. Define $PMR(f, g, R)$ to be pairwise max regret, the maximal difference in value between $g$ and $f$ over all possible rewards. Formally:

$$R(f, r) = \max_{g \in F} g \cdot r - f \cdot r$$ (Regret)

$$PMR(f, g, R) = \max_{r \in R} g \cdot r - f \cdot r$$ (Pairwise Max Regret)

$$MR(f, R) = \max_{r \in R} R(f, r) = \max_{g \in F} PMR(f, g, R)$$ (Max Regret)

$$MMR(R) = \min_{f \in F} MR(f, R) = \min_{f \in F} \max_{r \in R} \max_{g \in F} g \cdot r - f \cdot r$$ (Minimax Regret) (2)

$MR(f, R)$ is the max regret of a policy $f$ w.r.t. to the feasible reward set $R$. It captures the worst case loss over all possible realizations of reward. $MMR(R)$ is the minimax regret of a feasible reward set $R$, and the occupancy frequency $f$ that minimizes max regret (and the corresponding policy) is the minimax optimal policy. This definition can be interpreted as a game in which a decision maker chooses policy $f$ to minimize the loss relative to the optimal policy, and an adversary chooses the reward $r$ to maximize this loss given $f$. Related work by McMahan, Gordon and Blum [22] on computing robust policies for IRMDPs uses the maximin criterion, which computes policies that maximize worst-case value (often referred to as security...
level). The maximin criterion leads to conservative policies by optimizing against the worst-case instantiation of \( r \). Delage and Mannor [14] suggest a related criterion for robust policies in the presence of a prior over reward, using a percentile criterion, which can be less pessimistic than maximin. However, the percentile approach is neither fully Bayesian, nor does it offer a bound on performance. In comparison, the minimax regret criterion measures performance by assessing the policy \( \pi \) \textit{ex post}, making comparisons only w.r.t. specific reward realizations. Thus, a policy \( \pi \) is penalized on reward \( r \) only if there exists a \( \pi' \) that has higher value w.r.t. \( r \) itself. Apart from producing robust policies using an intuitively appealing criterion, minimax regret is also an effective driver of elicitation [28]. Unlike maximin (or the percentile criterion), regret provides guidance as to the maximal improvement in value should we obtain further information about reward. More details of related robustness criteria and their comparison to minimax regret can be found in Appendix A and Appendix B.

**Nondominated Policies** Several recent approaches to computing minimax regret rely on the concept of policies being \textit{nondominated} w.r.t. reward polytope \( \mathcal{R} \) [30, 36]. Formally, we say \( f \) is nondominated w.r.t. \( \mathcal{R} \) iff

\[
\exists r \in \mathcal{R} \quad \text{s.t.} \quad f \cdot r \geq f' \cdot r \quad \forall f' \in \mathcal{F}.
\]

Thus, a nondominated policy will be optimal for some \( r \in \mathcal{R} \). The full set of nondominated policies (for a fixed \( \mathcal{R} \)) can be leveraged to compute minimax regret. Let \( \Gamma_{\mathcal{R}} \) denote this full set of nondominated policies; we omit the subscript when \( \mathcal{R} \) is fixed and clear from context.

We define \( \mathcal{V}(r) = \max_{f \in \mathcal{F}} f \cdot r \) to be the optimal value obtainable when \( r \in \mathcal{R} \) is the true reward. My work [29, 30] shows that, since policy value is linear in \( r \), \( \mathcal{V} \) is \textit{piecewise linear and convex (PWLC)}, much like the belief-state value function for Partially Observable MDPs (POMDPs) [13, 21]. Since dominated policies \( f \) cannot contribute to this value, we can define \( \mathcal{V}_{\Gamma}(r) = \max_{f \in \Gamma} f \cdot r \), and immediately see that \( \mathcal{V} = \mathcal{V}_{\Gamma} \) [30]. In Section 3.1 we discuss reasons for approximating \( \mathcal{V} \) using a subset \( \tilde{\Gamma} \subseteq \Gamma \) of nondominated policies. We define the \( \mathcal{V} \)-error in this approximation to be [30]:

\[
\epsilon_{\mathcal{V}}(\tilde{\Gamma}, \mathcal{R}) = \max_{r \in \mathcal{R}} \mathcal{V}_{\Gamma}(r) - \mathcal{V}(r) \quad (3)
\]

Figure 1 in Appendix B gives a visual representation for \( \mathcal{V} \) and the discussion proceeding the figure offers some intuition as to its structure.

### 3 Computing Minimax Regret

The efficient computation of minimax regret optimal policies for realistic-sized MDPs is an important component of our reward elicitation framework. In order to allow for real-time user interaction, at each step of elicitation the minimax regret, given the current set of feasible reward functions, must be recomputed in a matter of seconds. However, it has been shown that finding a minimax regret optimal policy for an imprecise reward MDP is NP-Hard [36]. The general complexity of the problem has motivated our development of a number of strategies that yield effective computation in different scenarios. For instance, we have developed techniques to shift much complexity involved with finding minimax regret to an offline pre-computation step that can be carried out before interactive elicitation begins. We have also investigated methods to approximate minimax regret. Recall that a reasonable upper bound on regret can be substituted for exact regret during elicitation, since it also represents a guarantee of the worst case regret for the user.
3.1 Completed and Ongoing Work

Exact Minimax Regret We developed the first exact approach to computing minimax regret for IRMDPs [27]. The following discussion gives a brief sketch of the computational aspects of the work; full details can be found in Appendix A. The minimax regret optimization given by Eq. (2) can be reformulated as the following LP using Benders decomposition [2] and constraint generation:

\[
\min_{\delta, f} \delta \quad \text{subject to:} \quad g_i \cdot r_i - f \cdot r_i \quad \forall \langle g_i, r_i \rangle \in GEN
\]  

(4)

where GEN is a set of constraints corresponding to a subset of possible adversarial choices of r (and the corresponding policies). Given the solution f, we find the most violated constraint, i.e., the r, g-pair that maximizes the regret of f. If no violated constraint exists, then f is the minimax optimal policy. Otherwise we add the r, g-pair to the set GEN of constraints and iterate (solving LP (4)). Empirical assessment shows that the constraint generation procedure converges quickly for small MDPs.

Generating the most violated constraint involves the simultaneous maximization of reward and adversary policy that, using the most straightforward formulation, involves a quadratic non-convex optimization. Our work formulated the optimization as a mixed integer linear program (MIP) with \(O(|SA|)\) variables and \(O(|SA|)\) constraints [28]. This method has proven effective for small MDPs of 10 states or less, however the MIP has proven to be a bottleneck rendering the approach intractable for larger MDPs [28].

Approximate Minimax Regret To relieve the computational burden of computing exact minimax regret we explored several approximation strategies. Full details can be found in Appendix A. One approach uses an alternating optimization model that computes an adversarial policy (for a fixed reward) and an adversarial reward (for a fixed policy). This reduces the quadratic optimization for finding violated constraints to a sequence of LPs. Another approach uses a linear relaxation of the MIP used for constraint generation. We empirically found the error produced by this linear relaxation to be low (5% - 10%) [28]. However, neither approximation strategy is amenable to analytically bounding the potential error. In our reward elicitation framework minimax regret is used as a termination criterion, offering the user a guarantee as to maximum worst-case regret. These approximation strategies cannot support this guarantee, since they offer no upper bound on minimax regret. Subsequent discussion will outline alternate approximation schemes which leverage the precomputation of nondominated policies. One such scheme (using the linear support algorithm) achieves the desired upper bound on regret.

MMR Using Nondominated Policies We developed an approach to computing minimax regret that leverages the set \(\Gamma\) of nondominated policies [30], the full details of which can be found in Appendix B. The approach uses the constraint generation framework, and uses the following observation to more efficiently compute max regret (finding the most violated constraint): the adversarial policy used to maximize regret of f must lie in \(\Gamma\), since an adversary can only maximize regret by choosing some \(r \in R\) and an optimal policy \(f^*\) for r. Rather than solving a MIP we solve an LP, finding the optimal reward r for each policy \(g_i \in \Gamma\).

\[
\max_{r \in R} g_i \cdot r - f \cdot r
\]

The \(g_i\) with largest objective value determines the most violated constraint. We performed an empirical evaluation using small random MDPs and found our approach to be more effective than our MIP formulation and a similar approach proposed by Xu & Mannor [36]. For MDPs which have a small set of nondominated policies, this approach allows minimax regret to be computed quickly enough to support real-time user
interaction. However, the generation of the set of nondominated policies presents its own computational challenges, since we must potentially examine all feasible policies.

**Computing Nondominated Policies Using πWitness** The connection between the MDP value function w.r.t. \( r \) and the POMDP value function w.r.t. belief state, inspired our adaptation of a well known POMDP algorithm, Witness [21], to iteratively compute the set of nondominated policies. The full details of the approach can be found in our paper [30], attached as Appendix B.

A *local adjustment* \( f^{s,a} \) is the set of occupancy frequencies induced by starting in state \( s \) and taking action \( a \) rather than the action \( \pi(s) \) prescribed by the policy corresponding to \( f \) and following \( \pi \) thereafter. Given a partial set \( \tilde{\Gamma} \subset \Gamma \) our algorithm, \( \pi \)Witness, examines each \( f \in \tilde{\Gamma} \) and checks whether there is exists a local adjustment \( f^{s,a} \) and a witness reward \( r \) such that \( f^{s,a} \) offers improvement at \( r \). Here, the *improvement* is defined as: \( f^{s,a} \cdot r - \max_{f' \in \tilde{\Gamma}} [f' \cdot r] \), the difference in value between the local adjustment and the optimal policy \( f' \in \tilde{\Gamma} \) at the witness \( r \). If there is an improvement, then we have found a witness \( r \), and we compute the optimal policy for the witness reward function and add it to \( \tilde{\Gamma} \). Our work gives a theorem [30] showing that if there is no improvement offered by the local adjustments to all \( f \in \tilde{\Gamma} \), then \( \tilde{\Gamma} \) contains all nondominated policies.

We also show that the runtime of \( \pi \)Witness is polynomial in the number of nondominated policies generated. One implication is that for any class of MDPs with a polynomial number of nondominated policies, minimax regret computation is itself polynomial [30]. We conducted experiments using randomly generated MDPs with factored reward functions. These experiments indicate that for a fixed reward dimension the number of nondominated policies does not grow appreciably as we scale up the dimension of the state-space. However, as we increase the reward dimension, the number of nondominated policies (and the runtime of \( \pi \)Witness) grows quickly.

Recall that the set of nondominated policies can be generated offline; the runtime of the generation algorithm is less important than the resulting number of nondominated policies, which heavily impacts the online minimax regret computation. The \( \pi \)Witness algorithm can be halted at any time to produce a smaller partial set \( \tilde{\Gamma} \). Let \( MMR(\Gamma) \) denote the true minimax regret when adversarial policy choice is unrestricted, and \( MMR(\tilde{\Gamma}) \) denote the approximation when adversarial choice is restricted to \( \tilde{\Gamma} \). Then \( MMR(\tilde{\Gamma}) \) offers a lower bound on true minimax regret. Our work has shown that the error in minimax regret [30] can be bounded using the error in value as follows:

\[
\epsilon_{MMR}(\tilde{\Gamma}) = MMR(\Gamma) - MMR(\tilde{\Gamma}) \leq \epsilon_V(\tilde{\Gamma})
\]

However, the \( \pi \)Witness algorithm offers no theoretical guarantee on \( \epsilon_V \) and consequently offers no guarantee on the error in minimax regret.

**Computing Nondominated Policies Using Linear Support** To address the shortcomings of the \( \pi \)Witness algorithm, we developed a new method to incrementally generate the set of nondominated policies that naturally produces a bound on the minimax regret error given the partial set \( \tilde{\Gamma} \) at each step. In past work we observed that the error in minimax regret induced by \( \tilde{\Gamma} \) is bounded by the error in the value function \( \epsilon_V(\tilde{\Gamma}, R) \) [30]. The focus of the algorithm is to generate policies that directly reduce the error \( \epsilon_V(\tilde{\Gamma}, R) \). For this purpose we adapted the *linear support algorithm* for POMDPs [13]. Full details can be found Appendix C.

Given an approximate set \( \tilde{\Gamma} \), define the *nondominated region* of policy \( f \) w.r.t. \( \tilde{\Gamma} \), as follows:

\[
R_{\tilde{\Gamma}}(f) = \{ r \in R \mid f \cdot r \geq f' \cdot r, \forall f' \in \tilde{\Gamma} \}
\]
This corresponds to a region of $\mathcal{R}$ for which $f$ is the best policy in $\hat{\Gamma}$. The nondominated region $\mathcal{R}_{\hat{\Gamma}}(f)$ for each $f \in \hat{\Gamma}$ is bounded and convex; and the error function $\epsilon_V(\hat{\Gamma}, \mathcal{R})$ is convex over any such region (since error is defined as the difference between $V$, which is PWLC, and $\hat{V}_{\Gamma}$, which is linear over $\mathcal{R}_{\hat{\Gamma}}(f)$). Hence the maximum of $\epsilon_V(\hat{\Gamma}, \mathcal{R})$ over $\mathcal{R}_{\hat{\Gamma}}(f)$ must lie at a vertex of $\mathcal{R}_{\hat{\Gamma}}(f)$. As a consequence, the maximum error must lie at the vertex of the nondominated region of some $f \in \hat{\Gamma}$.

The linear support algorithm exploits this fact by computing error only at vertices of such regions, and adding (optimal) policies to $\hat{\Gamma}$ only for vertices with maximal error. The algorithm begins with an initial nondominated policy $f$—an optimal policy for some arbitrary $r \in \mathcal{R}$—in $\hat{\Gamma}$. It determines the next policy to add by: (a) computing $E_{\Gamma}$, the set of vertices of the nondominated regions of $\hat{\Gamma}$; (b) computing the optimal policy $f_r$ for each $r \in E_{\Gamma}$; and (c) selecting the policy that offers the greatest improvement. The selected policy is added to $\hat{\Gamma}$ and the process repeated until the maximum error at any vertex falls below some acceptable threshold (or some other termination criterion is met). This variant of the linear support algorithm can be improved upon using a number of optimizations; details can be found in Appendix C.

We undertook experiments to assess the performance of the linear support algorithm using randomly generated MDPs and the COACH assistive technologies domain (described in Section 5.1). The error $\epsilon_V$ was found to drop quickly as new policies are found; while the algorithm relies on vertex enumeration which is exponential in $|\mathcal{R}|$ in the worst case, empirically its runtime is competitive with $\pi$Witness for small MDPs.

**Online Adjustment of the Approximate Set of Nondominated Policies** The results above show that linear support can be used effectively to find the full set of nondominated policies for a specific feasible reward set $\mathcal{R}$; more importantly, its anytime properties allow relatively small sets of nondominated policies to be constructed that are very good approximations w.r.t. $V$-error. While linear support can be run offline to construct $\hat{\Gamma}$, its ability to produce small sets $\hat{\Gamma}$ with small bounded error allows for fast online computation of policies that are approximately minimax optimal. This is significant because small sets $\hat{\Gamma}$ enable efficient approximate minimax regret computation allowing for interactive reward elicitation.

We can further improve both the efficiency and quality of the approximation by adjusting the set $\hat{\Gamma}$ during elicitation [29] (details can be found in Appendix C). The improvement is realized by exploiting the fact that the feasible reward set $\mathcal{R}$ shrinks in size as more information is gleaned about the actual reward (e.g., as users respond to queries or behavior is observed). If $\mathcal{R}' \subseteq \mathcal{R}$ is a refinement of $\mathcal{R}$, then it must have fewer nondominated policies, i.e., $\Gamma_{\mathcal{R}'} \subseteq \Gamma_{\mathcal{R}}$. This means that policies that were nondominated w.r.t. $\mathcal{R}$ may become dominated when the feasible reward set is reduced to $\mathcal{R}'$. Since the computational performance of constraint generation using the nondominated set is tightly tied to its size, pruning away newly dominated policies can offer tremendous speed up in minimax regret computation. We developed a method for the pruning of $\Gamma_{\mathcal{R}'}$—or its approximation $\hat{\Gamma}$—using a simple LP test (analogous to the domination test for POMDPs proposed by Monahan [12]). Specifically, for each $f \in \hat{\Gamma}$, an LP is solved to find a reward point at which $f$ is nondominated [29].

While pruning can speed up online computation, it can also be used to “create space” to add new nondominated policies to the approximate set $\hat{\Gamma}$. Thus we can improve the quality of the approximation by adding new policies, while maintaining the same online computational overhead by using pruning to keep the size of $\hat{\Gamma}$ roughly constant.

We implemented these techniques and evaluated them using the COACH assistive technologies domain [29]. Results indicate that online adjustment of the set of nondominated policies has significant benefits. Elicitation that began with a partial set $\hat{\Gamma}$ with high error, quickly saw the error reduced to zero during elicitation through pruning and adding to $\hat{\Gamma}$. 
**Summary** Ongoing and completed work has developed a number of approaches to computing minimax regret. For small MDPs, the MIP formulation can be used to compute regret exactly. For larger MDPs that admit a small number of nondominated policies, the set $\Gamma$ can be computed offline (using $\pi$Witness or the linear support algorithm), and this set can be leveraged to efficiently compute minimax regret during elicitation. The linear support algorithm can be used in an anytime fashion to terminate when the size $\tilde{\Gamma}$ grows too large to allow for efficient online computation. The set $\tilde{\Gamma}$ can be used to approximate minimax regret with bounded error, and this set can be adjusted during elicitation to improve both efficiency and the quality of the approximation.

### 3.2 Future Directions

While our completed work has developed a wide variety of approaches to computing minimax regret, there remain many directions for research which involve assessment, extension and enhancement of these approaches.

**Characterizing the Set of Nondominated Policies** It would be useful to have a simple characterization of an MDP that allowed us to understand the properties that lead to small or easy-to-approximate sets of nondominated policies. A crisp characterization would allow us to quickly determine which MDPs would be suitable for our computational methods that use nondominated policies. Some heuristic approaches can identify dominance relations in the model reducing the number of states or actions that can possibly constitute a nondominated policy. For instance, the eliminating the choice of an action $a$ for a single state reduces the potential deterministic policies by $|A||S|-1$, since the set of policies that involve $a$ include policies selecting from $|A|$ actions in the other $|S| - 1$ states. This leaves $(|A| - 1)|A||S|-1$ potentially nondominated policies, which constitutes a weak upper bound. Furthermore, the search for dominance relations can require significant computation. Toward the use of more immediate structure (whose discovery does not require additional computation), some initial investigations have verified the correlation between the dimension of the reward polytope and the number of nondominated policies [30]. However, whether we can develop a straightforward bound given the immediate structure of the MDP remains an open question.

**Factored MDPs** In practice, reward functions typically have significant structure. Rather than relying on a flat state space, each state $X$ can be factored into a set of state variables $x = \langle x_1, x_2, \ldots, x_n \rangle$ [7]. Ongoing work has modified the computational framework from Section 3.1 to incorporate additive reward functions defined on only a subset of state variables. Preliminary investigation has indicated that the number of nondominated policies is influenced largely by the dimensionality of the reward function and less by conventional measures of MDP size, $|S|$ and $|A|$. Intuitively, this is so because high dimensional reward allows variability across the state-action space, admitting different optimal policies depending on the realization of reward. An important future direction is to adapt our computational approach to further incorporate structure in the transition dynamics. While most approaches to factored MDPs use dynamic programming approaches [18] which are not amenable to computing minimax regret, there are approaches to computing approximate value functions for factored MDPs using linear programming [17] that may be useful in our context. Supporting minimax regret computation for fully factored MDPs is an important step toward a general reward elicitation framework for two reasons. First is the fact that many existing MDP models are already expressed in a fully factored form because it more naturally models many real world domains; second, the independence assumptions captured by a fully factored model can allow for more refined queries that are better able to focus on the relevant aspects of the MDP. However, to realize these benefits
will require reimplementing most of the algorithms discussed in Section 3.1 and introducing considerable additional algorithmic complexity.

**Use of Prior Information for Approximating $\Gamma$** In some settings we may be able to establish a prior over the user’s reward function (using information from domain experts or collaborative filtering techniques). For techniques that leverage a small approximate set of nondominated policies, the prior can help guide the selection of the approximate set $\tilde{\Gamma}$. One direction for future research is adaptations of point-based POMDP algorithms [33] which find $\alpha$-vectors (analogous to nondominated policies) which exactly represent the value function for a small number of belief points (analogous to reward points) that form a good representation given the distribution over belief. The addition of prior information promises to allow for generating extremely small sets $\tilde{\Gamma}$ with low expected error, however sets computed with respect to this probabilistic information offer no guarantee on worst case error.

**Assessment** The further assessment of our tools for computing minimax regret remains an important research direction. Future work in this direction will perform an in-depth empirical analysis on the performance of $\pi$Witness and the linear support algorithm with respect to the characteristics of the IRMDP, addressing questions of how the policy generation algorithms scale w.r.t. the structure of the reward polytope, the sparseness of the transition function distribution, and the starting state distribution. Another important aspect of assessment is quantifying the performance of policy generation using larger set of real-world IRMDP application domains (some potential domains are described in Section 5).

**4 Elicitation**

Given a tractable approach to minimax regret computation, the remaining components of the reward elicitation framework are the specification of the query structure and its parameterization at each step of the elicitation. For each query we are interested in: its cognitive tractability (i.e., the ease with which the user can form a response), the computational complexity of selecting its parameters, and its effectiveness with respect to how quickly it reduces minimax regret.

We hypothesize that there is a fundamental tension between the conceptual simplicity of a query and the effectiveness of the query in reward elicitation. Queries which are constrained so as to be quickly understood lack the freedom of more general complex queries to elicit the “right” information that maximally reduces minimax regret. While the conceptual simplicity of potential query types can be superficially evaluated by simply putting oneself in the shoes of a user, an in-depth assessment can be carried out by studying groups of real users engaged in reward elicitation for one of the application domains discussed in Section 5. The effectiveness of many query types remains an open question. One goal of our ongoing research is to situate potential query types on a spectrum capturing their conceptual tractability and effectiveness, and to identify query types that strike a good balance.

**4.1 Completed and Ongoing Work**

**Simple Elicitation Strategies** The first query type we investigated was the simple bound query of form “Is $r(s,a) \geq b$?” where the scalar $b$ lies between the upper and lower bound on $r(s,a)$. In the context of the COACH assistive technologies domain, an example of such a query is the following question, “Is the reward for giving a minimally intrusive prompt given the current state of the task greater than 5?” While this appears to require a direct, quantitative assessment of value/reward by the user, it can be recast
as a *standard gamble* [16], a device used in decision analysis to reduce this to preference query over two outcomes (one of which is stochastic). A standard gamble assumes that we have identified the state-action pair with the highest reward \( r(s, a)^T \), and lowest reward \( r(s, a)^\perp \). The user is asked if they would prefer the certain reward \( r(s, a) \) to a lottery where the user receives reward \( r(s, a)^T \) with probability \( b \) and \( r(s, a)^\perp \) with probability \( (1 - b) \). Decision theory holds that these two forms of query imply the constraints on the utility (reward) function [16]. Compared to reward equivalence queries [14], which elicit a direct assessment of \( r(s, a) \), bound queries require only a yes-no response and are less cognitively demanding. A response tightens either the upper or lower bound on \( r(s, a) \).

This initial work set aside the challenge of optimally selecting the parameters (i.e., the state-action pairs) of each bound query and focused on some simple myopic heuristics that have been used with some success for regret-based utility elicitation for single-shot decision problems [8]. These heuristics share the property of being easy to compute. The first selection heuristic is called *halve largest gap (HLG)*, which selects the point \((s, a)\) with the largest gap between its upper and lower bound. Formally, we define the gap \( \Delta(s, a) \) and largest gap as follows:

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

\[
\arg\max_{a^* \in A, s^* \in S} \Delta(s^*, a^*)
\]

The second selection heuristic is the *current solution (CS)* strategy, and uses the visitation frequencies from the minimax optimal solution \( f \) or the adversarial witness \( g \) to weight each gap. Intuitively, if a query involves a reward parameter that influences the value of neither \( f \) nor \( g \), minimax regret will not be reduced, and visitation frequencies quantify the degree of influence. Formally CS selects the point:

\[
\arg\max_{a^* \in A, s^* \in S} \max\{ f(s^*, a^*) \Delta(s^*, a^*), g(s^*, a^*) \Delta(s^*, a^*) \}
\]

Given the selected \((s^*, a^*)\), the bound \( b \) in the query is set to the midpoint of the interval for \( r(s^*, a^*) \). Thus either response will reduce the interval by half.

**Assessment** We analyzed the effectiveness of the preceding simple elicitation strategies on randomly generated IRMDPs [28] (see Appendix A). For a given random IRMDP we engaged in reward elicitation with a simulated user first using the *current solution* strategy to select query parameters. The reward elicitation was then repeated using the *halve largest gap* strategy. At each step we took the robust policy computed and recorded 1) the minimax regret of that policy and 2) the “true regret” of the policy (w.r.t. to a simulated user’s “true” reward function). Our initial assessment found that while both strategies performed well, quickly reducing both measures of regret, the *current solution* strategy was able to reduce regret to zero in less than half the number of queries. In this setting less than 2 queries per parameter were required to find a provably optimal policy [28]. Let \( \chi \) measure the sum of the length of the reward intervals. At the end of elicitation, the *halve largest gap* strategy reduces \( \chi \) to 15.6% of its original value, while the *current solution* strategy only reduces \( \chi \) to 67.8% of its original value. The *current solution* strategy is effectively eliminating regret while leaving a large amount of uncertainty [28].

### 4.2 Future Research Directions

While the results of the evaluation of simple reward bound queries proved promising, there are a number of ways in which they can be improved upon. The reward bound query type, while simple, is also restrictive,
forcing the elicitation to focus on a reward for a single state and action in isolation. Geometrically, the response from these queries impose constraints on the reward polytope $R$ that are necessarily axis aligned. More complex queries yield more general linear constraints on $R$, allowing more freedom in the refinement of $R$, potentially allowing more effective minimax regret reduction. Our initial work has only scratched the surface of the space of potential query selection heuristics. There are more query selection heuristics from the single-step utility elicitation that can be adapted to reward elicitation [8, 20, 34] and there is the potential for sequential queries which better characterize the sequential nature of the underlying decision problem. Along with improvements in query type and query selection, our discussion of future work will also focus on possible extensions of our reward elicitation framework.

**Sequential Queries** The sequential nature of the MDP motivates novel query modes that do not exist in one-shot settings and encapsulate information about a sequence of states or actions in the MDP. These include the queries about policies, full (or partial) state-action trajectories, and occupancy frequencies. It is unreasonable to expect a user to directly assess any of these queries in isolation. For instance, asking a user to specify a numerical value for a policy or occupancy frequency requires implicitly assessing the reward at every state. However, comparisons between policies, trajectories or occupancy frequencies may be more manageable.

One starting point is to ask users to compare full policies (i.e. “Is $\pi$ preferred to $\pi'$?”). The response to such a query induces an inequality with respect to the value functions of the policies (for instance, an answer of “yes” implies $V^\pi > V^{\pi'}$), which in turn imposes a linear constraint on the reward function. However, full policies require simultaneously reasoning about many “what-if” scenarios of action and outcome along with their odds, which, for most users, is conceptually taxing and error prone. A simpler alternative is to design queries about the trajectories sampled by executing two policies for $n$ time steps (i.e. “Is the potential trajectory $s_1, a_1, s_2, a_2, \ldots, s_n$ preferred to $s'_1, a'_1, s'_2, a'_2, \ldots, s'_n$?”). The query response also yields a linear constraint on the reward function. The answer of “yes” implies $\sum_t \gamma^{t-1} r(s_t, a_t) > \sum_t \gamma^{t-1} r(s'_t, a'_t)$. The trajectory query removes the need to reason about the odds or conditional actions reducing the conceptual burden on the user. But, the assessment of long state-action sequences can still be difficult for a user. How to best select the length and composition of each trajectory remains an open question.

Another potential form query involves looking at occupancy frequencies, which represent relatively how much time is spent in state $s$ taking action $a$. We can design queries using these frequencies that compare portions of the policy. For instance, “Would you prefer $(s, a)$ to be visited $f(s, a)$ of the time and for $(s', a')$ to be visited $f(s', a')$ of the time, or would you prefer that $(s, a)$ be visit $g(s, a)$ of time and for $(s', a')$ to be visited $g(s', a')$ of the time?”. Given a factored reward model, this allows for a user to directly specify trade-offs pertaining to portions of the policy. In the COACH domain (described in Section 5.1) a query can distinguish trade-offs between the success of the patient completing the task and the amount of intervention used by the prompting system. An example query is:

All things being equal, which scenario involving step 4 of the task do you prefer: a) 50% of the time there is a delay of 1 minute for which a light prompt level is used and 10% of the time there is a delay of greater than 5 minutes for which a caregiver is called, or b) 70% of the time there is a delay of 1 minute while for which a medium prompt is used and 2% percent of the time there is a delay of more than 5 minutes for which a caregiver is called?

Generally, this form of query takes the following form: “All things being equal, do you prefer the following states and actions to be visited with the frequencies $f$: $f(s_0, a_0), f(s_1, a_1), \ldots, f(s_n, a_n)$ or the frequencies $g$: $g(s_0, a_0), g(s_1, a_1), \ldots, g(s_n, a_n)$”. A user preferring $f$ induces the following linear constraint on reward: $\sum_n f(s_i, a_i)r(s_i, a_i) \geq \sum_n g(s_i, a_i)r(s_i, a_i)$. 12
Unlike reward bound queries, the constraints induced by sequential queries are not restricted to be axis aligned, potentially giving more power to the elicitation process. However, with this increased freedom comes increased cognitive burden and an increase in the complexity of the parameter selection. The following table outlines the dimension of the parameter space for these queries:

| Query Type                        | Parameter Dimension |
|-----------------------------------|---------------------|
| Reward Bound                      | $|S||A|$             |
| Policy Comparison                 | $|A||^S|$            |
| Trajectory Comparison             | $(|A||^S|)^n$        |
| Occupancy Frequency Comparison    | $(|S||A|)^k$         |

The policy comparison potentially selects among $|A|^{|S|}$ possible policies, a trajectory comparison of a sequence $n$ steps long selects among $n$ state and action pairs, and an occupancy frequency comparison examining $k$ states and actions for each state and action selects a real valued occupancy frequency to present to the user.

**Optimal Query Selection** While optimization over the parameter space for many query types imposes a significant computational burden, for some simple settings it may be possible to exactly compute the myopically optimal query. Formally, given a query parameterization $\phi$ and query response $\rho$, let $R^\rho_\phi$, be the reward polytope that results from imposing the constraint implied by $\rho$. Single-step preference elicitation settings have adopted the expected value of information as a measure of the quality of a query [5, 12]. In our setting we define this as follows:

$$EVOI(\phi, R) = \sum_{\rho} \Pr(\rho|\phi) MMR(R^\rho_\phi) - MMR(R)$$

In the absence of a prior over responses, we can either assume a uniform distribution or we can extend our robust paradigm defining the minimum value of information ($EVOI$) w.r.t. worst case response

$$EVOI(\phi, R) = \min_{\rho} \left[ MMR(R^\rho_\phi) - MMR(R) \right]$$

The optimal myopic query then selects the parameters $\phi$ which maximize the expected or minimum value of information. This requires potentially solving minimax regret for each potential parameterization and query response. In single shot settings techniques have been developed to find the myopically optimal query for selecting the most preferred item from a set [35] (and by extension, simple comparison queries, which ask for the most preferred item from a two item set). An open question is whether these techniques can be adapted to our setting in which the analog to an item is a full MDP policy. Another open question is whether the set $\Gamma$ can be leveraged to mitigate the complexity of finding myopically optimal queries. Furthermore, a small subset $\tilde{\Gamma}$ may serve to quickly compute a reasonable approximation. We can explore the option of using a crude (but direct) approximation to the optimal query parameterization and contrast this with the “exact” heuristic techniques like current solution.

It is worth noting that we can further define an optimal sequence of queries that maximizes the expected value of information over the entire sequence. However, the additional computational complexity incurred by modeling and optimizing over the entire sequence of queries places this approach outside the scope of immediate future research plans.
Heuristic Query Selection  To allow for query selection to function within a real-time interactive reward elicitation framework, it will likely be necessary to resort to myopic heuristic query selection approaches.

Of potential value are volumetric methods that aim to directly reduce the volume of the reward polytope. One technique developed by Iyengar [20] examines the hyperplane formed by the linear constraint induced by each query. The distance of the hyperplane from the analytic center of the polytope is used to prune the potential queries. The heuristic then looks at the volume of the partitions resulting from each hyperplane and attempts to find the query/hyperplane that will make these volumes as close to equal as possible.

Volumetric methods have been proposed that go beyond binary comparisons, asking a user to choose the most preferred of a set of $n$ outcomes. An approach by Tobia [34] selects a query by attempting to find the set of outcomes that equally partitions the polytope. This problem is approximately solved by first finding the analytic center of the polytope and then using an ellipsoid to approximate the volume of the polytope. One benefit of these volumetric methods is that they yield bounds on the number of queries required to reduce the volume to some level $\epsilon$. However the end goal is to improve the quality of policy with respect to minimax regret and a large reduction in the volume of the reward polytope may not lead to a large reduction in minimax regret.

The main thrust of future work will be to flesh out each potential query type: developing effective query selection methods and assessing their impact on elicitation in different application domains. Novel application domains have the further potential to inspire novel query types.

Indifference  We currently impose no boundary on the precision with which a user may be expected to answer, earlier experiments have indicated that elicitation tends to focus on a small set of “high impact” reward points repeatedly querying to slice the interval of feasible reward ever smaller [28]. It is reasonable to assume that a user may have a limit to the precision with which they can answer these repeated queries. This motivates allowing users to answer queries with “I don’t know”. The implications of this response vary with query type. For bound queries we can assume that an “I don’t know” response indicates that the “true reward” value is sufficiently close to the bound to render a decision difficult. Such a response constrains $r(s, a)$ to be within a small constant $\delta$ of the bound $b$. For comparison queries, the response indicates that the user is indifferent between the two options. For instance, indifference when comparing two policies, $\pi$ and $\pi'$, imposes the constraint: $|V^\pi - V^{\pi'}| \leq \delta$. Incorporating indifference requires minor changes in implementation and results in query response options which are more natural for a user and potentially increase the effectiveness of elicitation.

Passive information  In some settings we may be able to passively observe behaviour before or in conjunction with the elicitation process. The field of inverse reinforcement learning (IRL) [23] addresses a similar problem recovering a reward function that induces the demonstrated behavior. Given an observed policy $\pi$, we can apply what are termed the IRL constraints [23] defining the set of reward functions that make $\pi$ optimal:

$$\left(P_\pi - P_a\right)(I - \gamma P_\pi)^{-1}r \succeq 0$$

The work on inverse reinforcement learning aims to then choose the “best” single reward function where the measure of best varies from maximizing entropy (with respect to a distribution over policies) [23] to maximizing margin (maximizing the cost of making deviations from the observed policy) [37]. In our setting, given an observed optimal policy $\pi$ we can directly incorporate the constraints on reward as a starting point for elicitation. Given a partial sequence of optimal actions in the form of trajectories
Given \( s_0, a_0, s_1, a_1, \ldots, s_n, a_n \), we cannot directly employ the IRL constraints, however we can formulate constraints on policies that agree with the observed trajectory:

\[
f(s_i, a_i) \geq f(s_i, a'_i) \quad \forall \ i, a'_i \neq a_i
\]

While these constraints on policy do not directly reduce imprecision over the space of feasible reward functions, reducing the space of feasible policies impacts minimax regret.

It may be possible to focus future observed behaviour on some subset of state or action space, or to request the demonstration of a sequence of actions given a starting state. The occupancy frequencies corresponding to the minimax optimal policy may serve to guide the parameterization of these interactions, however this remains an open question.

### Eliciting Subjective Reward Structure

The number of reward bearing states in real-world MDPs can be extremely small compared to the size of the state space. While a domain expert may be able to specify which state-action pairs should be the focus of elicitation, it can be desirable to leave the specification of structure of the reward function up to the user during elicitation. Recent work conducted by Boutilier, Regan & Viapanni [9, 10, 11] has examined a similar scenario in the single-step decision making setting. They enable the user to specify groupings of individual features of the decision space to form composite features that are used during elicitation. For instance, elicitation can determine that a user has specific preferences about “safe” cars that are heavy, have high ground clearance, side air-bags and anti-lock brakes.

A significant component of this future work is the development of a variant of the optimization performed to compute minimax regret with respect to both uncertainty over utility and uncertainty with respect to the definition of composite features. A precomputed set of nondominated policies may continue to serve as leverage to improve the tractability of the minimax regret computation, however an additional wrinkle is that the set policies must be nondominated with respect to both reward uncertainty and the uncertainty over the definition of reward structure.

### Mixing Query Types

The elicitation of both reward structure and the value of reward motivates the study of reward elicitation with multiple query types. A simple model can attempt to select among both query types and their parameterization with the sole aim of maximally reducing regret. Of course, not all query types impose the same cognitive burden on the user. How to best model and integrate the conceptual cost of various query types remains an open question.

### Assessment

In the assessment of the components of the reward elicitation framework, many interesting questions remain. For instance, heuristics using the current solution have proven effective in quickly reducing minimax regret: will the heuristic remain effective given an approximate current solution? How can we best trade-off computational efficiency and elicitation effectiveness? How do heuristic methods compare with optimal query selection? Is using an exact heuristic like current solution preferable to direct approximation the optimal query?

Future work will thoroughly examine the configurations of query type, query selection, and robust policy computation with a focus on how quickly regret is reduced during elicitation. Immediate future work will focus on investigating sequential query types, evaluating their effectiveness first in conjunction with the already implemented halve largest gap and current solution query selection heuristics. The next step is to develop novel query selection heuristics that are tailored to these query types, taking inspiration from volumetric approaches. In the case that myopically optimal query selection proves tractable, we can incorporate...
this to further assess the effectiveness of the new query types. The immediate mode of assessment will involve simulations to examine the computational implications of these research questions. Time permitting, a secondary assessment will be carried out to investigate the cognitive and conceptual effectiveness of our developed query types with a user study.

Another important aspect of assessment is quantifying the effectiveness of elicitation in a number of different application domains. The next section describes some ongoing and future domains that are well suited for reward elicitation.

5 Applications

While our reward elicitation framework can theoretically be applied to any MDP, there are some domains for which the overhead of conducting elicitation is more reasonable. The following is a short list of desirata for potential application domains:

1. Multi-dimensional Reward Domains in which preferences of the user involve a direct mapping of a reward to another measurable scalar quantity such as money or time likely do not require the overhead of reward elicitation. An example of such a domain might be solving a stochastic scheduling problem in which the only relevant measure is the time taken to complete a set of tasks. Instead, desirable domains have many competing aspects to the reward function requiring trade-offs to be made. An example of such a trade-off is in the COACH domain where the autonomy of the individual being prompted is traded-off with the likelihood that the individual will complete the task.

2. Repeatability Desirable domains have reward functions that must be elicited for different users in different settings. One example is an MDP modeling resource allocation in a hospital setting (discussed in Section 5.2). The elicitation establishes trade-offs between competing objectives and can be repeated to reflect the different trade-offs preferred by different hospitals.

3. Natural MDP Structure Ideally the problem domain comfortably fits the following MDP modeling assumptions. Uncertainty is endogenous (i.e. actions influence uncertain outcomes). Outcomes are fully observable. The state and action space is finite. Finally, the transition function is specifiable and accurate (since we do not model the transition uncertainty).

5.1 Ongoing Work

Autonomic Computing The task involves allocating computing or storage resources to application servers as their client demands change over time [6]. We have explored one concrete setting with $k$ application server elements and $N$ units of resource available to be assigned to the servers. There are $D$ demand levels at which each server can operate, reflecting client demands. A demand state $d = \langle d_1 \ldots d_k \rangle$ specifies the current demand for each server. A state of the MDP comprises the current resource allocation and the current demand state: $s = \langle n, d \rangle$. Actions are new allocations $m = \langle m_1 \ldots m_k \rangle$ of the $N$ resources to the $k$ servers. Reward $\tau(n, d, m) = u(n, d) - c(n, d, m)$ decomposes as follows. Utility $u(n, d)$ is the sum of server utilities $u_i(n_i, d_i)$. The structure of the MDP is simplified since uncertainty in demand is exogenous and the action in the current state uniquely determines the allocation in the next state. Reward specification in this context is inherently distributed and quite difficult: the local utility function $u_i$ for server $i$ has no convenient closed form. Generally, server $i$ can respond only to queries about the utility it gets from a specific resource allocation level, and this requires intensive optimization and simulation on the part of
the server [6]; hence minimizing the number of such queries is critical. Further details and the results of a partial assessment of reward elicitation in this domain can be found in Appendix A. While the specification of utility for the autonomic provision of resources constitutes an interesting and valuable domain, it falls short of fully meeting our reward elicitation desirata. The underlying decision problem is not a perfect fit for an MDP since actions have no stochastic effects (all uncertainty is exogenous). Also, the absence of a human user in the process obviates the aspects of the reward elicitation framework that are sensitive to cognitive limitations.

Assistive Technologies  This domain includes systems that provide cognitive assistance for persons with dementia, enabling them to complete the common activities of daily living. An example is the COACH project [3, 4], whose goal is to guide a person through a small task (e.g., hand-washing) by providing verbal or visual cues, while allowing the individual to maintain as much independence as possible.

In ongoing work we have created a more concrete instantiation of the problem modeling a general task of $\ell$ steps. The system can issue prompts at increasing levels of intrusiveness, or can call a caregiver (e.g., therapist or family member) to assist the person in task completion. This results in action space $A = \{0, 1, \ldots, k\}$ where $0$ indicates no prompt was issued, level $k - 1$ indicates the strongest most intrusive prompt and level $k$ indicates that the caregiver was called. The state is defined by three variables $S = \langle t, d, f \rangle$ where $t$ models the number of tasks steps successfully completed by the person, $d$ models the delay (time taken during the current step); and $f$ tracks whether a prompt at a specific level was attempted on the current task step and failed to immediately get the person to the next step. The dynamics express the following intuitions. The no-prompt action will cause a “progress” transition to the next step (setting delay and failed-prompt to zero), or a “stall” transition (same step with delay increased by one). The probability of reaching the next step with action $a = n$ is higher than $a = n - 1$ since more intrusive prompts have a better chance of facilitating progress; however, progress probability decreases as delay increases. Reaching the next step after prompting is less likely if a prompt has already failed at the current step. The factored reward function is defined as

$$r(\langle t, d, f \rangle, a) = r_{\text{goal}}(t) + r_{\text{progress}}(d = 0) + r_{\text{delay}}(d) + r_{\text{prompt}}(a)$$

where $r_{\text{goal}}(t)$ is the reward for completing the task, $r_{\text{progress}}(d = 0)$ is the reward for progressing to the next step, $r_{\text{delay}}(d)$ is the (negative) reward for delay in completing a step; and $r_{\text{prompt}}(a)$ is the (negative) reward associated with prompting the person (calling the caregiver is generally very costly). The precise values of the rewards are not known and must be elicited from a caregiver. Further details, along with an evaluation of our approaches using nondominated policies during reward elicitation in this domain, can be found in Appendix C.

The COACH domain was investigated as a potential reward elicitation domain primarily because previous work using the model encountered great difficulty when specifying the reward function [3, 4]. The reward specification process originally involved hand-picking a reward function, demonstrating the resulting policies for caregivers to critique, and using these critiques to hand-tune the reward function. This process was repeated many times before reasonable policies were found. Our ongoing work with this model will move towards creating queries that domain experts such as care-givers find intuitive allowing for more effective specification of the reward function through elicitation.

5.2 Future Directions

Consumer Decision Support  Consumer decision support is a popular domain for single-step preference elicitation. For example the work of Pu & Faltings describes eliciting a user’s preferences over potential
vacation plans [25]. Decision support is sensible in this domain due to the many options for both hotel accommodation and flight itineraries. The decision problem becomes sequential when we incorporate dynamic pricing and availability. Now the challenge is not simply recommending a hotel, but advising when to make the purchase. Recent work has quantified the dynamics of airline pricing, producing predictive models of when the price of flights will go up and down [15] and demonstrating that significant savings can be realized by delaying a purchase into the future. The states of the MDP will capture current pricing and availability of hotels and flights, their relevant features, what has been spent, and what is confirmed. The actions comprise the potential recommendations for purchase of hotel, and travel options.

This is a suitable domain for reward elicitation due to the complex multi-dimensional reward function that captures the trade-offs between hotel and flight features, pricing and risk attitudes towards availability. Furthermore, there is a clear necessity for reward to be specified for each individual user, removing the need for the time-consuming hand-tuning of reward by domain experts.

**Organizational Planning** There are sequential planning and logistics problems faced by organizations that require specifying trade-offs between many competing objectives. For instance, the problem of the scheduling patients with respect to hospital resources was recently looked at by Patrick & Puterman [24]. They developed a simple model for scheduling computed tomography (CT) scans that minimized unused capacity subject to overtime constraints. Rather than specifying overtime as a constraint a-priori, we can add flexibility by encoding potential preferences for overtime using uncertain reward. We can enrich the model to include other relevant trade-offs such the maximal wait times incurred by different priorities of patients. Reward elicitation in this setting would allow for interactive specification of these trade-offs at each hospital.

### 6 Research Plan

My most immediate research goals involve the further assessment of the techniques leveraging nondominated policies and the development of sequential query types. Specifically, the methods for pruning and adding nondominated policies during elicitation show promise, but further investigation is required (both theoretical and empirical) to determine the size of the initial approximate set $\tilde{\Gamma}$ of nondominated policies and to form suitable guidelines for how much pruning/adding should occur during elicitation.

I intend to implement the necessary code for selecting occupancy frequency queries and for imposing the resulting linear constraints on the reward polytope during elicitation. I will evaluate these new queries using the existing HLG and CS query selection heuristics and I will spend some time exploring novel heuristics tailored to sequential queries. The necessary implementation for allowing users to respond with indifference is not overly complicated and can be carried out next. In conjunction with this work I will continue to flesh out potential applications, consulting with domain experts in the assistive technology and organizational planning settings. This consultation will help to construct MDP models that better reflect these settings, and perhaps more importantly it should offer feedback on the suitability of query types and conceptual tractability of the elicitation in general. One hope is that the structure and challenges of specific settings may lead to novel query types and selection heuristics.

The next step is to explore the optimal selection of query parameters, specifically investigating the computational leverage that may be brought to bear by the set of nondominated policies. While not central to the research plan, broadening our model to allow for elicitation of reward structure would be an interesting capstone to these research plans.
References

[1] Richard E. Bellman. Dynamic Programming. Princeton University Press, Princeton, 1957. 4
[2] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik, 4:238–252, 1962. 6
[3] Jennifer Boger, Pascal Poupart, Jesse Hoey, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05), pages 1293–1299, Edinburgh, 2005. 17
[4] Jennifer Boger, Pascal Poupart, Jesse Hoey, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. A planning system based on Markov decision processes to guide people with dementia through activities of daily living. IEEE Transactions on Information Technology in Biomedicine, 10(2):323–333, 2006. 17
[5] Craig Boutilier. A pomdp formulation of preference elicitation problems. Proceedings of the Eighteenth National Conference on Artificial Intelligence, pages 239–246, Jan 2002. 13
[6] Craig Boutilier, Rajarshi Das, Jeffrey O. Kephart, Gerald Tesauro, and William E. Walsh. Cooperative negotiation in autonomic systems using incremental utility elicitation. In Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-03), pages 89–97, Acapulco, 2003. 16, 17
[7] Craig Boutilier, Thomas Dean, and Steve Hanks. Decision theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research, 11:1–94, 1999. 9
[8] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. Artificial Intelligence, 170(8–9):686–713, 2006. 1, 2, 3, 11, 12
[9] Craig Boutilier, Kevin Regan, and Paolo Viappiani. Online feature elicitation in interactive optimization. Proceedings of the 26th Annual International Conference on Machine Learning, pages 73—80, Jan 2009. 15
[10] Craig Boutilier, Kevin Regan, and Paolo Viappiani. Preference elicitation with subjective features. Proceedings of the third ACM conference on Recommender systems, pages 341—344, Jan 2009. 15
[11] Craig Boutilier, Kevin Regan, and Paolo Viappiani. Simultaneous elicitation of preference features and utility. Twenty-Fourth AAAI Conference on Artificial Intelligence, Jan 2010. 15
[12] Darius Braziunas and Craig Boutilier. Minimax regret-based elicitation of generalized additive utilities. In Proceedings of the Twenty-third Conference on Uncertainty in Artificial Intelligence (UAI-07), pages 25–32, Vancouver, 2007. 13
[13] Hsien-Te Cheng. Algorithms for Partially Observable Markov Decision Processes. PhD thesis, University of British Columbia, Vancouver, 1988. 5, 7
[14] Erick Delage and Shie Mannor. Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML-07), pages 225–232, Corvallis, OR, 2007. 1, 5, 11
[15] Oren Etzioni, Craig Knoblock, Rattapoom Tuchinda, and Alexander Yates. To buy or not to buy: mining airfare data to minimize ticket purchase price. International Conference on Knowledge Discovery and Data Mining, pages 119—128, Jan 2003. 18
[16] Simon French. Decision theory: an introduction to the mathematics of rationality. Halsted Press, 1986. 11
[17] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. Journal of Artificial Intelligence Research, 19(10):399 – 468, Jan 2003. 9
[18] Jesse Hoey, Robert St-Aubin, Alan Hu, and Craig Boutilier. SPUDD: Stochastic planning using decision diagrams. In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI-99), pages 279–288, Stockholm, 1999. 9
[19] Ronald A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, 1960. 4

[20] Vijay S. Iyengar, Jon Lee, and Murray Campbell. Q-Eval: Evaluating multiple attribute items using queries. In *Proceedings of the Third ACM Conference on Electronic Commerce*, pages 144–153, Tampa, FL, 2001. 12, 14

[21] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998. 5, 7

[22] Brendan McMahan, Geoffrey Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. *Proceedings of the Twentieth International Conference on Machine Learning*, page 536, 2003. 1, 4

[23] Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, pages 663–670, Stanford, CA, 2000. 14

[24] Jonathan Patrick and Martin Puterman. Improving resource utilization for diagnostic services through flexible inpatient scheduling: A method for improving resource utilization. *Journal of the Operational Research Society*, Jan 2006. 18

[25] Pearl Pu, Boi Faltings, and Marc Torrens. User-involved preference elicitation. In *IJCAI-03 Workshop on Configuration*, Acapulco, 2003. 18

[26] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994. 2, 3, 4

[27] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. *Neural Information Processing Systems - Workshop on Model Uncertainty and Risk in RL*, 2008. 1, 2, 6

[28] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. *The 25th Conference on Uncertainty in Artificial Intelligence*, 2009. 2, 3, 5, 6, 11, 14

[29] Kevin Regan and Craig Boutilier. Robust online optimization of reward-uncertain MDPs using linear support. *In submission to the Twenty-Fourth Annual Conference on Neural Information Processing Systems*, 2010. 2, 3, 5, 8

[30] Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. *Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010. 2, 3, 5, 6, 7, 9

[31] Ahti Salo and Raimo P. Hämäläinen. Preference ratios in multiattribute evaluation (PRIME)–elicitation and decision procedures under incomplete information. *IEEE Trans. on Systems, Man and Cybernetics*, 31(6):533–545, 2001. 1, 3

[32] Leonard Savage. The foundations of statistics. *Wiley, New York*, 1972. 1

[33] Matthijs T. J. Spaan and Nikos Vlassis. A point-based POMDP algorithm for robot planning. In *IEEE International Conference on Robotics and Automation*, New Orleans, 2004. to appear. 10

[34] Olivier Toubia, John Hauser, and Duncan Simester. Polyhedral methods for adaptive choice-based conjoint analysis. *Journal of Marketing Research*, 41:116–131, 2004. 3, 12, 14

[35] Paolo Viappiani and Craig Boutilier. Regret-based optimal recommendation sets in conversational recommender systems. *Proceedings of the third ACM conference on Recommender systems*, pages 101—109, 2009. 3, 13

[36] Huan Xu and Shie Mannor. Parametric regret in uncertain Markov decision processes. In *48th IEEE Conference on Decision and Control*, pages 3606–3613, Shanghai, 2009. 1, 2, 4, 5, 6

[37] Brian Ziebart, Andrew Maas, Andrew Bagnell, and Anind Dey. Maximum entropy inverse reinforcement learning. *Proceedings of the National Conference on Artificial Intelligence*, pages 1433 – 1438, Jan 2008. 14
A Conference Paper

Regret-based Reward Elicitation for Markov Decision Processes
Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, 2009.

This paper includes the MIP formulation for exact minimax regret computation. It discusses alternating approximation and a linear relaxation of the MIP formulation. The paper empirically investigates computational efficiency of the methods and the approximation quality. A small instantiation of an autonomic computing setting is used to assess reward elicitation using simple bound queries along with myopic selection heuristics.
Regret-based Reward Elicitation for Markov Decision Processes

Kevin Regan
Department of Computer Science
University of Toronto
Toronto, ON, CANADA
kmregan@cs.toronto.edu

Craig Boutilier
Department of Computer Science
University of Toronto
Toronto, ON, CANADA
cebly@cs.toronto.edu

Abstract

The specification of a Markov decision process (MDP) can be difficult. Reward function specification is especially 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 and aims to minimize the degree of precision with which a reward function must be specified while still allowing optimal or near-optimal policies to be produced. We first discuss how robust policies can be computed for MDPs given only partial reward information using the minimax regret criterion. We then demonstrate how regret can be reduced by efficiently eliciting reward information using bound queries, using regret-reduction as a means for choosing suitable queries. Empirical results demonstrate that regret-based reward elicitation offers an effective way to produce near-optimal policies without resorting to the precise specification of the entire reward function.

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 can be difficult, e.g., cognitively demanding, computationally costly, or time consuming. For this reason, much work has been devoted to learning the dynamics of stochastic systems from transition data, both in offline [11] and online (i.e., reinforcement learning) settings [19]. While model dynamics are often relatively stable in many application domains, MDP reward functions are much more variable, reflecting the preferences and goals of specific users in that domain. This makes reward function specification more difficult: they can’t generally be specified a priori, but must be elicited or otherwise assessed for individual users. Even online RL methods require the specification of a user’s reward function in some form: unlike state transitions, it is impossible to directly observe a reward function except in very specific settings with simple, objectively definable, observable performance criteria. The “observability” of reward is a convenient fiction often assumed in the RL literature.

Reward specification is difficult for three reasons. First, it requires the translation of user preferences—which states and actions are “good” and “bad”—into precise numerical rewards. As has been well-recognized in decision analysis, people find it extremely difficult to quantify their strength of preferences precisely using utility functions (and, by extension, reward functions) [10]. Second, the requirement to assess rewards and costs for all states and actions imposes an additional burden (one that can be somewhat alleviated by the use of multiattribute models in factored MDPs [5]). Finally, the elicitation problem in MDPs is further exacerbated by the potential confabulation 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.

In this paper, we tackle the problem of reward elicitation in MDPs by treating it as a preference elicitation problem. Recent research in preference elicitation for non-sequential decision problems exploits the fact that optimal or near-optimal decisions can often be made with relatively imprecise specification of a utility function [6, 8]. Interactive elicitation and optimization techniques take advantage of feasibility restrictions on actions or outcomes to focus their elicitation efforts on only the most relevant aspects of a utility function. We adopt a similar perspective in the MDP setting, demonstrating that optimal and near-optimal policies can be often found with limited reward information. For instance, reward bounds in conjunction with MDP dynamics can render certain regions of state space provably dominated by others (w.r.t. value).

We make two main contributions that allow effective elicitation of reward functions. First, we develop a novel robust optimization technique for solving MDPs with imprecisely specified rewards. Specifically, we adopt the minimax regret decision criterion [6, 18] and develop a formulation for MDPs: intuitively, this determines a policy that has minimum regret, or loss w.r.t. the optimal policy, over all possible reward function realizations consistent with the curr-
rent partial reward specification. Unlike other work on robust optimization for imprecisely specified MDPs, which focuses on the maximin decision criterion [1, 13, 14, 16], minimax regret determines superior policies in the presence of reward function uncertainty. We describe an exact computational technique for minimax regret and suggest several approximations. Second, we develop a simple elicitation procedure that exploits the information provided by the minimax-regret solution to guide the querying process. In this work, we focus on simple schemes that refine the upper minimax-regret solution to guide the querying process. In particular, we see below, we often explicitly wish to leave parts of a reward function unelicited (or otherwise unassessed). Formally we assume that \( r \in \mathcal{R} \), where the feasible reward set \( \mathcal{R} \) reflects current knowledge of the reward. These could reflect: prior bounds specified by a user or domain expert; constraints that emerge from an elicitation process (as discussed below); or constraints that arise from observations of user behavior (as in inverse RL [15]). In all of these situations, we are unlikely to have full reward information. Thus we require a criterion by which to compare policies in an imprecise-reward MDP.

We adopt the minimax regret criterion, originally suggested (though not endorsed) by Savage [18], and applied with some success in non-sequential decision problems [6, 7]. Let \( \mathcal{R} \) be the set of feasible reward functions. Minimax regret can be defined in three stages:

\[
R(f, r) = \max_{g \in \mathcal{F}} r \cdot g - r \cdot f 
\]

(4)

\[
MR(\mathcal{R}) = \max_{r \in \mathcal{R}} R(f, r) 
\]

(5)

\[
MMR(\mathcal{R}) = \min_{f \in \mathcal{F}} MR(f, \mathcal{R}) 
\]

(6)

\( R(f, r) \) is the regret of policy \( f \) (as represented by its visitation frequencies) relative to reward function \( r \): it is simply the loss or difference in value between \( f \) and the optimal policy under \( r \). \( MR(\mathcal{R}) \) is the maximum regret of \( f \) w.r.t. feasible reward set \( \mathcal{R} \). Should we chose a policy with visitation frequencies \( f \), \( MR(\mathcal{R}) \) represents the worst-case loss over all possible realizations of the reward function; i.e., the regret incurred in the presence of an adversary who chooses the \( r \) from \( \mathcal{R} \) to maximize our loss. Finally, in the presence of such an adversary, we wish to minimize this max regret: \( MMR(\mathcal{R}) \) is the minimax regret of feasible reward set \( \mathcal{R} \). This can be viewed as a game between a decision maker choosing \( f \) who wants to minimize loss relative to the optimal policy, and an adversary who chooses a reward to maximize this loss given the decision maker’s
choice of policy. Any \( f^\ast \) that minimizes max regret is a minimax optimal policy, while the \( r \) that maximizes its regret is the witness or adversarial reward function, and the optimal policy \( g \) for \( r \) is the witness or adversarial policy.

Minimax regret has a variety of desirable properties relative to other robust decision criteria [6]. Compared to Bayesian methods that compute expected value using a prior over \( \mathcal{R} \) [3, 8], minimax regret provides worst-case bounds on loss. Specifically, let \( f \) be the minimax regret optimal visitation frequencies and let \( \delta \) be the max regret achieved by \( f \); then, given any instantiation of \( r \), no policy will outperform \( f \) by more than \( \delta \) w.r.t. expected value. Minimax optimal decisions can often be computed more effectively than decisions that maximize expected value w.r.t. some prior. Finally, it has been shown to be a very effective criterion for driving elicitation in one-shot problems [6, 7].

### 2.3 Robust Optimization for Imprecise MDPs

Most work on robust optimization for imprecisely specified MDPs adopts the maximin criterion, producing policies with maximum security level or worst-case value [1, 13, 14, 16]. Restricting attention to imprecise rewards, the maximin value is given by:

\[
\text{MMN}(\mathcal{R}) = \max_{f \in \mathcal{F}} \min_{r \in \mathcal{R}} r \cdot f
\]

Most models are defined for uncertainty in any MDP parameters, but algorithmic work has focused on uncertainty in the transition function, and the of eliciting information about transition functions or rewards is left unaddressed. Robust policies can be computed for uncertain transition functions using the maximin criterion by decomposing the problem across time-steps and using dynamic programming and an efficient suboptimization to find the worst case transition function [1, 13, 16]. McMahan, Gordon, and Blum [14] develop a linear programming approach to efficiently compute the maximin value of an MDP (we empirically compare this approach to ours below). Delage and Mannor [9] address the problem of uncertainty over reward functions (and transition functions) in the presence of prior information, using a percentile criterion, which can be somewhat less pessimistic than maximin. They also contribute a method for eliciting rewards using sampling to approximate the expected value of information of noisy information about a point in reward space. The percentile approach is neither fully Bayesian nor does it offer a bound on performance. Zhang and Parkes ([20]) also adopt maximin in a model that assumes an inverse reinforcement learning setting for policy teaching. The approach is essentially a form of reward elicitation which the queries are changes to a student’s reward, and information is gained by observing change in the student’s behavior.

Generally, the maximin criterion leads to conservative policies by optimizing against the worst possible instantiation of \( r \) (as we will see below). Minimax regret offers a more intuitive measure of performance by assessing the policy \textit{ex post} and making comparisons only w.r.t. specific reward realizations. Thus, policy \( \pi \) is penalized on reward \( r \) only if there exists a \( \pi' \) that has higher value w.r.t. \( r \) itself.

### 3 Minimax Regret Computation

As discussed above, maximin is amenable to dynamic programming since it can be decomposed over decision stages. This decomposition does not appear tenable for minimax regret since it grants the adversary too much power by allowing rewards to be set independently at each stage (though see our discussion of future work below). Following the formulations for non-sequential problems developed in [6, 7], we instead formulate the optimization using a series of linear (LPs) and mixed integer programs (MIPs) that enforce a consistent choice of reward across time.

Assume feasible reward set \( \mathcal{R} \) is represented by a convex polytope \( \mathcal{C}_r \leq d \), which we assume to be bounded. The constraints on \( r \) arise as discussed above (prior bounds, elicitation, or behavioral observation). Minimax regret can then be expressed as following minimax program:

\[
\begin{align*}
\min_{f} \quad & \max_{r} \min_{g} r \cdot f \\
\text{subject to:} \quad & \gamma E^r f + \alpha = 0 \\
& \gamma E^g (\alpha = 0) \\
& Cr \leq d
\end{align*}
\]

This is equivalent to a minimization:

\[
\begin{align*}
\min\limits_{f, \delta} & \delta \\
\text{subject to:} \quad & r \cdot g - r \cdot f \leq \delta \quad \forall g \in \mathcal{F}, r \in \mathcal{R} \\
& \gamma E f + \alpha = 0
\end{align*}
\]

This corresponds to the standard dual LP formulation of an MDP with the addition of adversarial policy constraints. The infinite number of constraints can be reduced: first we need only retain as potentially active those constraints for vertices of polytope \( \mathcal{R} \); and for any \( r \in \mathcal{R} \), we only require the constraint corresponding to its optimal policy \( g^r \). However, vertex enumeration is not feasible; so we apply Benders’ decomposition [2] to iteratively generate constraints.

At each iteration, two optimizations are solved. The master problem solves a relaxation of program (8) using only a small subset of the constraints, corresponding to a subset Gen of all \( (g, r) \) pairs; we call these generated constraints. Initially, this set is arbitrary (e.g., empty). Intuitively, in the game against the adversary, this restricts the adversary to choosing witnesses (i.e., \( (g, r) \) pairs) from Gen.

Let \( f \) be the solution to the current master problem and \( MMR^r(\mathcal{R}) \) its objective value (i.e., minimax regret in the presence of the restricted adversary). The subproblem generates the maximally violated constraint relative to \( f \). In other words, we compute \( MR(f, \mathcal{R}) \); its solution determines the witness points \( (g, r) \) by removing restrictions
on the adversary. If $MR(f, R) = MMR'(R)$ then the constraint for $(g, r)$ is satisfied at the current solution, and indeed all unexpressed constraints must be satisfied as well. The process then terminates with minimax optimal solution $f$. Otherwise, $MR(f, R) > MMR'(R)$, implying that the constraint for $(g, r)$ is violated in the current relaxation (indeed, it is the maximally violated such constraint). So it is added to $Gen$ and the process repeats.

Computation of $MR(f, R)$ is realized by the following MIP, using value and $Q$-functions:\footnote{Specifying max regret in terms of visitation frequencies (i.e., the standard dual MDP formulation) gives rise to a non-convex quadratic program. Regret maximization does not lend itself to a natural, linear primal formulation.}

$$\begin{align*}
\text{maximize} & \quad \alpha \cdot V - r \cdot f \\
\text{subject to:} & \quad Q_a = r_a + \gamma P_a V \quad \forall a \in A \\
& \quad V \geq Q_a \quad \forall a \in A \quad (10) \\
& \quad V \leq (1 - I_a)M_a + Q_a \quad \forall a \in A \quad (11) \\
& \quad Cr \leq d \\
& \quad \sum_a I_a = 1 \quad (12) \\
& \quad I_a(s) \in \{0, 1\} \quad \forall a, s \quad (13) \\
& \quad M_a = M^T - M^a \\
\end{align*}$$

Here $I$ represents the adversary’s policy, with $I_a(s)$ denoting the probability of action $a$ being taken at state $s$ (constraints (12) and (13) restrict it to be deterministic). Constraints (10) and (11) ensure that the optimal value $V(s, a)$ for a single action $a$. We ensure a tight $M_a^+$ by setting $M^T$ to be the optimal value function $V^+$ of the optimal policy with respect to the best setting of each individual reward point and $M^a_+$ to be the $Q$-value $Q_a$ of the optimal policy with respect to the worst point-wise setting of rewards (the resulting rewards need not be feasible).

The subproblem does not directly produce a witness pair $(g_i, r_i)$ for the master constraint set; instead it provides $r_i$ and $V_i$. However, we do not need access to $g_i$ directly; the constraint can be posted using the reward function $r_i$ and the value $\alpha \cdot V_i$, since $\alpha \cdot V_i = r_i \cdot g_i$ (and $g_i$ is required to determine this adversarial value in the posted constraint).

In practice we have found that the iterative constraint generation converges quickly, with relatively few constraints required to determine minimax regret (see Sec. 5). However, the computational cost per iteration can be quite high. This is due exclusively to the subproblem optimization, which requires the solution of a MIP with a large number of integer variables, one per state-action pair. The master problem optimization, by contrast, is extremely effective (since it is basically a standard MDP linear program). This suggests examination of approximations to the subproblem, i.e., the computation of max regret $MR(f, R)$. This is also motivated by our focus on reward elicitation. We wish to use minimax regret to drive query selection: our aim is not to compute minimax regret for its own sake, but to determine which state-action pairs should be queried, i.e., which have the potential to reduce minimax regret. The visitation frequencies used by our heuristics need not correspond to exact minimax optimal policy.

We have explored several promising alternatives, including an alternating optimization model that computes an adversarial policy (for a fixed reward) and an adversarial reward (for a fixed policy). This reduces the quadratic optimization for max regret to a sequence of LPs. An simpler approximation is explored here (which performs as well in practice): we solve the LP relaxation of the MIP by removing the integrality constraints (13) on the binary policy indicators. The value function $V$ resulting from this relaxation does not accurately reflect the (now stochastic) adversarial policy: $V$ may include a fraction of the big-M term due to constraint (10). However, the reward function $r$ selected remains in the feasible set, and, empirically, the optimal value function for $r$ yields a solution to the subproblem that is close to optimal.\footnote{Finding the optimal value function for $r$ requires solving a standard MDP LP.} Since the reward is valid choice, this solution is guaranteed to be a lower bound on the solution to the subproblem. When this approximate subproblem solution is used in constraint generation, convergence is no longer guaranteed; however, the solution to the master problem represents a valid lower bound on minimax regret.

## 4 Reward Elicitation

Reward elicitation and assessment can proceed in a variety of ways. Many different query forms can be adopted for user interaction. Similarly, observed user behavior can be used to induce constraints on the reward function under assumptions of user “optimality” \footnote{We allow reward queries about any state-action pair, in contrast to online RL formalisms, in which information can be gleaned only about the reward (and dynamics) at the current state. As such, we face no exploration-exploitation tradeoff.}. In this work, we focus on simple bound queries, though our strategies can be adapted to more general query types. We discuss some of these below.\footnote{While this appears to require a direct, quantitative assessment of value/reward by the use, it can be recast as a standard gamble \cite{10}, a device used in decision analysis to reduce this to preference query over two outcomes (one of which is stochastic). For simplicity, we express it in this bound form. Unlike reward queries \cite{9}, which require a direct assessment of $r(s, a)$, bound queries require only a yes-no response and are less cognitively demanding. A response tightens either the upper or lower frequencies used by our heuristics need not correspond to exact minimax optimal policy. The visitation frequencies used by our heuristics need not correspond to exact minimax optimal policy.}

We assume that $R$ is given by upper and lower bounds on $r(s, a)$ for each 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 appears to require a direct, quantitative assessment of value/reward by the user, it can be recast as a standard gamble \cite{10}, a device used in decision analysis to reduce this to preference query over two outcomes (one of which is stochastic). For simplicity, we express it in this bound form. Unlike reward queries \cite{9}, which require a direct assessment of $r(s, a)$, bound queries require only a yes-no response and are less cognitively demanding. A response tightens either the upper or lower
Bound queries offer a natural starting point for the investigation of reward elicitation. Of course, many alternative query modes can be used, with the sequential nature of the MDP setting opening up choices that don’t exist in one-shot settings. These include the direct comparison of policies; comparison of (full or partial) state-action trajectories or distributions over trajectories; and comparisons of outcomes in factored reward models. Trajectory comparisons can be facilitated by using counts of relevant (or reward-bearing) events as dictated by a factored reward model for example. These query forms should prove useful and even more cognitively penetrable. However, the principles and heuristics espoused below can be adapted to these settings.

There are many ways to select the point \((s,a)\) at which to ask a bound query. We explore some simple myopic heuristic criteria that are very easy to compute, are based on criteria suggested in [6]. The first selection heuristic is called halve largest gap (HLG), which selects the point \((s,a)\) with the largest gap between its upper and lower bound. Formally, we define the gap \(\Delta(s,a)\) and largest gap by:

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

\[
\arg\max_{a^* \in A, s^* \in S} \Delta(s^*, a^*)
\]

The second selection heuristic is the current solution (CS) strategy, and uses the visitation frequencies from the minimax optimal solution \(g\) or the adversarial witness \(g\) to weight each gap. Intuitively, if a query involves a reward parameter that influences the value of neither \(g\) nor \(g\), minimax regret will not be reduced, and visitation frequencies quantify the degree of influence. Formally CS selects the point:

\[
\arg\max_{a^* \in A, s^* \in S} \max\{f(s^*, a^*)\Delta(s^*, a^*), g(s^*, a^*)\Delta(s^*, a^*)\}.
\]

Given the selected \((s^*, a^*)\), bound \(b\) in the query is set to the midpoint of the interval for \(r(s^*, a^*)\). Thus either response will reduce the interval by half. It is easy to apply CS to the maximin criterion as well, using the visitation frequencies associated with the maximin policy.

5 Experiments

We assess the general performance of our approach using a set of randomly generated MDPs and specific MDPs arising in an autonomic computing setting. We assess scalability of our procedures, as well as the effectiveness of minimax regret as a driver of elicitation.

We first consider randomly generated MDPs. We impose structure on the MDP by creating a semi-sparse transition function: for each \((s,a)\)-pair, \([\log \alpha]\) reachable states are drawn uniformly and a Gaussian is used to generate transition probabilities. We use a uniform initial state distribution \(\alpha\) and discount factor \(\gamma = 0.95\). The true reward is drawn uniformly from a fixed interval and uncertainty w.r.t. this true (but unknown) reward is created by bounding each \((s,a)\)-pair independently with bounds drawn randomly: thus the set of feasible rewards forms a hyper-rectangle.

5.1 Computational Efficiency

To measure the performance of minimax regret computation, we first examine the constraint generation procedure. Fig. 1 plots the regret gap between the master problem value and subproblem value at each iteration versus the time (in ms.) to reach that iteration. Results are shown for 20 randomly generated MDPs with ten states and five actions. Fig. 2 shows how minimax regret computation time increases with the size of the MDP (5 actions, varying number of states). Constraint generation using the MIP formulation scales super-linearly, hence computing minimax regret exactly is only feasible for small MDPs using this formulation; by comparison the linear relaxation is far more efficient. On the other hand, minimax regret computation has very favorable anytime behavior, as exhibited in Fig. 1. During constraint generation, the regret gap shrinks very quickly early on. If exact minimax regret is not needed, this property allows for fast approximation.

5.2 Approximation Error

To evaluate the linear relaxation scheme for max regret, we generated random MDPs, varying the number of states. Fig. 3 shows average relative error over 100 runs. The approximation performs well and, encouragingly, error does not increase with the size of the MDP. We also evaluate its impact on minimax regret when used to generate violated constraints. Fig. 3 also shows relative error for minimax regret to be small, well under 10% on average.

\(^{4}\)Indifference (e.g., “I’m not sure”) can also be handled by constraining bounds to be within \(\varepsilon\) of the query point.

\(^{5}\)Cplex 11 is used for all MIPs and LPs, and all code run on a PowerEdge 2950 server with dual quad-core Intel E5355 CPUs.

\(^{6}\)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 impli-
5.3 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 [14]. The computation time for maximin is significantly less than that of minimax regret—this is expected since maximin requires only the solution of a pair of linear programs.

We use both maximin and minimax regret to compute policies at each step of preference elicitation, and paired each with the current solution (CS) and halve largest gap (HLG) query strategies, giving four elicitation procedures: MMR-HLG (policies are computed using regret, queries generated by HLG); MMR-CS (regret policies, CS queries); MM-HLG (maximin policies, HLG queries); and MM-CS (maximin policies, CS queries). We assess each procedure by measuring the quality of the policies produced after each query, using the following metrics: (a) its maximin value given the current (remaining) reward uncertainty; (b) its max regret given the current (remaining) reward uncertainty; and (c) its true regret (i.e., loss w.r.t. the optimal policy for the true reward function r, where r is used to generate query responses). Minimax regret 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 practice; but it gives an indication of how good the resulting policies actually are (as opposed to a worst-case bound).

Fig. 4 show the results of the comparison on each measure. MMR-CS performs extremely well on all measures. Somewhat surprisingly, it outperforms MM-CS and MM-HLG w.r.t. maximin value (except at the very early stages). Even though the maximin procedures are optimizing maximin value, MMR-CS asks much more informative queries, allowing for a larger reduction in reward uncertainty at the most relevant state-action pairs. This ability of MMR-CS to identify the highest impact reward points becomes clearer still when we examine how much reduction there is in reward intervals over the course of elicitation. Let χ measure the sum of the length of the reward intervals. At the end of elicitation, MMR-HLG reduces 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. Fig. 5 illustrates this using a histogram of the number of queries asked by MMR-CS about each of the 1000 possible state-action pairs.7 We 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” pairs.

Fig. 4(b) 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 (Fig. 4(c)). With the CS heuristic, a near-optimal policy is found after fewer than 50 queries (less than one 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 preference elicitation, HLG does not require that minimax regret actually be computed.

720 MDPs with 10 states, 5 actions each.
Minimax regret is only necessary to assess when to stop the elicitation process (i.e., to determine if minimax regret has dropped to an acceptable level). One possible modification to reduce the time between queries is to only compute minimax regret after every $k$ queries. Of course, the HLG strategy will lead to a slower reduction in true regret and minimax regret as shown in Figs. 4(b) and 4(c).

To further evaluate our approach we elicit the reward function for an autonomic computing scenario [4] in which we must allocate computing or storage resources to application servers as their client demands change over time. We assume $k$ application server elements and $N$ units of resource available to be assigned to the servers (plus a “zero resource”). An allocation $n = (n_1 \ldots n_k)$ must satisfy $\sum n_i < N$. There are $D$ demand levels at which each server can operate, reflecting client demands. A demand state $d = (d_1 \ldots d_k)$ specifies the current demand for each server. A state of the MDP comprises the current resource allocation and the current demand state: $s = (n, d)$. Actions are new allocations $m = (m_1 \ldots m_k)$ of the $N$ resources to the $k$ servers. Reward $r(n, d, m) = u(n, d) − c(n, d, m)$ decomposes as follows. Utility $u(n, d)$ is the sum of server utilities $u_i(n_i, d_i)$. The MDP is initially specified with strict uncertainty over the utilities $u_i$; however, we assume that each utility function $u_i$ is monotonic non-decreasing in demand and resource level. The cost $c(n, d, m)$ is the sum of the costs of taking away one unit of resource from each server at any stage. Uncertainty in demand is exogenous and the action in the current state uniquely determines the allocation in the next state. Thus the transition function is composed of $k$ Markov chains $\Pr(d'_i \mid d_i), i \leq k$. Reward specification in this context is inherently distributed and quite difficult: the local utility function $u_i$ for server $i$ has no convenient closed form. Server $i$ can respond only to queries about the utility it gets from a specific resource allocation level, and this requires intensive optimization and simulation on the part of the server [4]; hence minimizing the number of such queries is critical.

We constructed a small instance of the autonomic computing scenario with 2 servers, 3 demand levels and 3 (indivisible) units of resource. The combined state space of both servers includes $3^2$ demand levels and 10 possible allocations of resources leading to 90 states and 10 actions. We modeled the uncertainty over rewards using a hyperrectangle as with the random MDPs. We compared elicitation approaches as above, this time using the linear relaxation to compute minimax regret (each minimax computation takes under 3s.). Fig. 6 shows that MMR-CS again outperforms the maximin criterion on each measure. Minimax regret and true regret fall to almost zero after 200 queries. Recall that the autonomic MDP had 900 state-action pairs—the additional problem structure results in fewer than 0.25 queries being asked for each state-action pair. In fact, on average MMR-CS only asks about 106.5 distinct state-action pairs, only examining 12% of the reward space. By comparison, the queries chosen by the MM-CS strategy cover just over 68% of the reward space. As with random MDPs, minimax regret quickly reduces regret because it focuses queries on the “high impact” state-action pairs.

Overall, our regret-based approach is quite appealing from the perspective of reward elicitation. While the regret computation is more computationally intensive than other criteria, it provides arguably much more natural decisions in the face of reward uncertainty. More importantly, from the perspective of elicitation, it is much more attractive than minimax w.r.t. the number of queries required to produce high-quality policies. As long as interaction time (time between queries) remains reasonable, reducing user burden (or other computational costs required to answer queries) is our primary goal.

6 Conclusions & Future Work

We have developed an approach to reward elicitation in MDPs that eases the burden of reward function elicitation. Minimax regret not only offers robust policies in the face of reward uncertainty, but we’ve shown it also allows one to focus elicitation attention on the most important aspects of the reward function. While the computational costs are significant, it is an extremely effective driver of elicitation,
thus reducing the (more important) cognitive or computational cost of reward determination. Furthermore, it lends itself to anytime approximation.

The somewhat preliminary nature of this work leaves many interesting directions for future research. Perhaps most interesting is the development of more informative and intuitive queries that capture the sequential nature of the elicitation problem. Direct comparison of policies allows one to distinguish value from reward, but are cognitively demanding. Trajectory comparison similar distinguishes value, but may contain irrelevant detail. However, trajectory summaries (e.g., counts of relevant reward bearing events) may be more perspicuous, and could be generated to reflect expected “event counts” given a policy. Other forms of queries should also prove valuable, but all exploit the basic idea embodied by minimax regret and the current solution heuristic. Another direction for improving elicitation is to incorporate implicit information in a manner similar to policy teaching [20]. Inverse RL [15] can be also used to translate observed behavior into constraints on reward. Some Bayesian models [6, 8] allow noisy query responses and adding this to our regret model is another important direction. Two approaches include: approximate indifference constraints and regret-based sensitivity analysis. The efficiency of the minimax regret computation remains an important research topic. We are exploring the use of dynamic programming to generate linear representations of the best policies over all regions of reward space (much like POMDPs) which can greatly assist max regret computation. We are also exploring techniques that exploit factored MDP structure using LP approaches [12].

References

[1] J. Bagnell, A. Ng, and J. Schneider. Solving uncertain Markov decision problems. Tech Report, Jan 2001.
[2] J. Benders. Partitioning procedures for solving mixed-variables programming problems. Numerische Math., 1962.
[3] C. Boutilier. A POMDP formulation of preference elicitation problems. AAAI-02, pp.239–246, Edmonton, 2002.
[4] C. Boutilier, R. Das, J. O. Kephart, G. Tesauro, and W. E. Walsh. Cooperative negotiation in autonomic systems using incremental utility elicitation UAI-03, pp.89–97, Acapulco, 2003.
[5] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. J. Artif. Intell. Res., 11:1–94, 1999.
[6] C. Boutilier, R. Patrascu, P. Poupart, D. Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. Artificial Intelligence, 170:686–713, 2006.
[7] C. Boutilier, T. Sandholm, and R. Shields. Eliciting bid taker non-price preferences in (combinatorial) auctions. AAAI-04, pp.204–211, San Jose, CA, 2004.
[8] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. AAAI-00, pp.363–369, Austin, TX, 2000.
[9] E. Delage and S. Mannor. Percentile optimization in uncertain markov decision processes with application to efficient exploration. ICML-07, pp.225–232, Corvalis, OR, 2007.
[10] S. French. Decision Theory: An Introduction to the Mathematics of Rationality. Halsted Press, 1986.
[11] N. Friedman, K. Murphy, and S. Russell. Learning the structure of dynamic probabilistic networks. UAI-98, pp.139–147, Madison, WI, 1998.
[12] C. Guestrin, D. Koller, R. Parr, and S. Venkataraman. Efficient solution algorithms for factored mdps. J. Artif. Intell. Res. 19:399-468, 2003.
[13] G. Iyengar. Robust dynamic programming. Mathematics of Operations Research, 30(2):1-21, Jan 2005.
[14] H. McMahan, G. Gordon, and A. Blum. Planning in the presence of cost functions controlled by an adversary. ICML-03, pp.536–543, Washington, DC, 2003.
[15] A. Ng and S. Russell. Algorithms for inverse reinforcement learning. ICML-00, pp.663–670, Stanford, CA, 2000.
[16] A. Nilim and L. El Ghaoui. Robustness in Markov decision problems with uncertain transition matrices. NIPS-03, Vancouver, 2003.
[17] M. Puterman. Markov Decision Processes: Discrete Stochastic Dynamic Programming. Wiley, New York, 1994.
[18] L. Savage. The Foundations of Statistics. Wiley, 1954.
[19] R. Sutton and A. Barto. Reinforcement Learning: An Introduction. MIT Press, Cambridge, MA, 1998.
[20] H. Zhang and D. Parkes. Value-based policy teaching with active indirect elicitation. AAAI-08, pp.208–214, 2008.
This paper is the first foray into computing minimax regret using nondominated policies. A few different methods for computing minimax regret using nondominated policies are described and empirically evaluated. The main contribution of the paper is the $\pi$Witness algorithm for generating the set of nondominated policies. The performance of $\pi$Witness is evaluated using randomly generated MDPs. A theoretical bound on the error imposed by an approximate subset $\tilde{\Gamma}$ of nondominated policies is developed and some heuristics improvements to $\pi$Witness to allow for anytime generation of $\tilde{\Gamma}$ are discussed.
Robust Policy Computation in Reward-uncertain MDPs using Nondominated Policies

Kevin Regan
University of Toronto
Toronto, Ontario, Canada, M5S 3G4
kmregan@cs.toronto.edu

Craig Boutilier
University of Toronto
Toronto, Ontario, Canada, M5S 3G4
cebly@cs.toronto.edu

Abstract
The precise specification of reward functions for Markov decision processes (MDPs) is often extremely difficult, motivating research into both reward elicitation and the robust solution of MDPs with imprecisely specified reward (IRMDPs). We develop new techniques for the robust optimization of IRMDPs, using the minimax regret decision criterion, that exploit the set of nondominated policies, i.e., policies that are optimal for some instantiation of the imprecise reward function. Drawing parallels to POMDP value functions, we devise a Witness-style algorithm for identifying nondominated policies. We also examine several new algorithms for computing minimax regret using the nondominated set, and examine both practically and theoretically the impact of approximating this set. Our results suggest that a small subset of the nondominated set can greatly speed up computation, yet yield very tight approximations to minimax regret.

Introduction
Markov decision processes (MDPs) have proven their value as a formal model for decision-theoretic planning. However, the specification of MDP parameters, whether transition probabilities or rewards, remains a key bottleneck. Recent work has focused on the robust solution of MDPs with imprecisely specified parameters. For instance, if a transition model is learned from observational data, there will generally be some uncertainty associated with its parameters, and a robust solution will offer some guarantees on policy quality even in the face of such uncertainty (Iyengar 2005; Nílím and Ghaoui 2005).

Much research has focused on solving imprecise MDPs using the maximin criterion, emphasizing transition model uncertainty. But recent work deals with the robust solution of MDPs whose rewards are incompletely specified (Delage and Mannor 2007; McMahan, Gordon, and Blum 2003; Regan and Boutilier 2009; Xu and Mannor 2009). This is the problem we consider. Since reward functions must often be tailored to the preferences of specific users, some form of preference elicitation is required (Regan and Boutilier 2009); and to reduce user burden we may wish to solve an MDP before the entire reward function is known. Rather than maximin, minimax regret has been proposed as a suitable criterion for such MDPs with imprecise rewards (IRMDPs) (Regan and Boutilier 2009; Xu and Mannor 2009), providing robust solutions and serving as an effective means of generating elicitation queries (Regan and Boutilier 2009). However, computing the regret-optimal policy in IRMDPs is theoretically complex (Xu and Mannor 2009) and practically difficult (Regan and Boutilier 2009).

In this work, we develop techniques for solving IRMDPs that exploit the existence of nondominated policies. Informally, if \( \mathcal{R} \) is a set of possible reward functions, we say a policy \( \pi \) is nondominated if there is an \( r \in \mathcal{R} \) for which \( \pi \) is optimal. The set of nondominated policies can be exploited to render minimax regret optimization far more efficient. We offer three main contributions. First, we describe a new algorithm for the minimax solution of an IRMDP that uses the set \( \Gamma \) of nondominated policies to great computational effect. Second, we develop an exact algorithm for computing \( \Gamma \) by drawing parallels with partially observable MDPs (POMDPs), specifically, the piecewise linear and convex nature of optimal value over \( \mathcal{R} \). Indeed, we suggest several approaches based on this connection to POMDPs. We also show how to exploit the low-dimensionality of reward space in factored reward models to render the complexity of our algorithm largely independent of state and action space size. Third, we provide a method for generating approximately nondominated sets. While \( \Gamma \) can be extremely large, in practice, very close approximations of small size can be found. We also show how such an approximate set impacts minimax regret computation, bounding the error theoretically, and investigating it empirically.

Background
We begin with relevant background on IRMDPs.

Markov Decision Processes
We restrict our focus to infinite horizon, finite state and action MDPs \( \langle S, A, \{P_{sa}\}, \gamma, \beta, r \rangle \), with states \( S \), actions \( A \), transition model \( P_{sa}(\cdot) \), non-negative reward function \( r(\cdot, \cdot) \), discount factor \( \gamma < 1 \), and initial state distribution \( \beta(\cdot) \). We use vector notation for convenience with: \( r \) an \( |S| \times |A| \) matrix with entries \( r(s,a) \), and \( P \) an \( |S||A| \times |S| \) transition matrix. Their restrictions to action \( a \) are denoted...
\(r_a\) and \(P_a\), respectively; and matrix \(E\) is identical to \(P\) with 1 subtracted from each self-transition probability \(P_{sa}(s)\).

Our aim is to find an optimal policy \(\pi\) that maximizes the sum of expected discounted rewards. Ignoring the initial state distribution \(\beta\), value function \(V^\pi: S \rightarrow \mathbb{R}\) for deterministic policy \(\pi\) satisfies:

\[
V^\pi = r_\pi + \gamma P_\pi V^\pi
\]

(1)

where restrictions to \(\pi\) are defined in the usual way. Given initial distribution \(\beta\), \(\pi\) has expected value \(\beta V^\pi\). It also induces occupancy frequencies \(f^\pi\), where \(f^\pi(s, a)\) is the total discounted probability of being in state \(s\) and taking action \(a\). \(\pi\) can be recovered from \(f^\pi\) via \(\pi(s, a) = f^\pi(s, a)/\sum_a f^\pi(s, a')\) (for deterministic \(\pi\), \(f^\pi = 0\) for all \(a \neq \pi(s)\)). Let \(\mathcal{F}\) be the set of valid occupancy frequencies w.r.t. a fixed MDP, i.e., those satisfying (Puterman 1994):

\[
\gamma E^T f + \beta = 0.
\]

(2)

We write \(f^\pi[s]\) to denote the occupancy frequencies induced by \(\pi\) when starting in state \(s\) (i.e., ignoring \(\beta\)). In what follows, we use frequencies and policies interchangeably since each uniquely determines the other. An optimal policy \(\pi^*\) satisfies \(V^{\pi^*} \geq V^\pi\) (pointwise) for all \(\pi\). For any positive \(\beta > 0\), maximizing expected value \(\beta V^\pi\) requires that \(\pi\) be optimal in this strong sense.

**Imprecise Reward MDPs**

In many settings, a reward function can be hard to obtain, requiring difficult human judgements of preference and tradeoffs (e.g., in domains such as cognitive assistive technologies (Boger et al. 2006), or expensive computation (see, e.g., value computation as a function of resource availability in autonomic computing (Boutilier et al. 2003; Regan and Boutilier 2009)). We define an imprecise reward MDP (IRMDP) \((S, A, \{P_{sa}\}, \gamma, \beta, R)\) by replacing reward \(r\) by a set of feasible reward functions \(\mathcal{R}\). The set \(\mathcal{R}\) naturally arises from observations of user behaviour, partial elicitation of preferences, or information from domain experts, which typically place linear constraints on reward. We assume that \(\mathcal{R}\) is a bounded, convex polytope defined by linear constraint set \(\{r \mid Ar \leq b\}\) and use \(|\mathcal{R}|\) to denote the number of constraints. In the preference elicitation model that motivates this work, these constraints arise from user responses to queries about the reward function (Regan and Boutilier 2009).

Given an IRMDP, we desire a policy that is robust to the imprecision in reward. Most robust optimization for imprecise MDPs adopts the maximin criterion, producing policies with maximum security level or worst-case value (Bagnell, Ng, and Schneider 2003; Iyengar 2005; McMahan, Gordon, and Blum 2003; Nilim and Ghaoui 2005). With imprecise reward, maximin value is:

\[
MMN(\mathcal{R}) = \max_{f \in \mathcal{F}} \min_{r \in \mathcal{R}} r \cdot f
\]

(3)

Maximin policies can be computed given an uncertain transition function by dynamic programming and efficient suboptimization to find worst case transition functions (Bagnell, Ng, and Schneider 2003; Iyengar 2005; Nilim and Ghaoui 2005). However, these models cannot be extended to imprecise rewards. Maximin policies for IRMDPs can be determined using linear programming with constraint generation (McMahan, Gordon, and Blum 2003).

The maximin criterion leads to conservative policies by optimizing against the worst instantiation of \(r\). Instead, we adopt the minimax regret criterion (Savage 1954) applied recently to IRMDPs (Regan and Boutilier 2009; Xu and Mannor 2009). Let \(f, g\) be policies (i.e., their occupancy frequencies), \(r\) a reward function, and define:

\[
R(f, r) = \max_{g \in \mathcal{F}} r \cdot g - r \cdot f
\]

(4)

\[
PMR(f, g, \mathcal{R}) = \max_{r \in \mathcal{R}} r \cdot g - r \cdot f
\]

(5)

\[
MR(f, \mathcal{R}) = \max_{r \in \mathcal{R}} R(f, r) = \max_{g \in \mathcal{F}} PMR(f, g, \mathcal{R})
\]

(6)

\[
MMR(\mathcal{R}) = \min_{f \in \mathcal{F}} \max_{r \in \mathcal{R}} MR(f, \mathcal{R})
\]

(7)

\[
R(f, r)\text{ is the regret or loss of policy } f \text{ relative to } r, \text{ i.e., the difference in value between } f \text{ and the optimal policy under } r. \text{ MR}(f, \mathcal{R}) \text{ is the maximum regret of } f \text{ w.r.t. feasible reward set } \mathcal{R}. \text{ Should we chose a policy } f, \text{ MR}(f, \mathcal{R}) \text{ represents the worst-case loss over possible realizations of reward; i.e., the regret incurred in the presence of an adversary who chooses } r \text{ to maximize loss. Equivalently, it can be viewed as the adversary choosing a policy with greatest pairwise max regret PMR}(f, g, \mathcal{R}), \text{ defined as the maximal difference in value between policies } f \text{ and } g \text{ under possible reward realizations. In the presence of such an adversary, we wish to minimize this max regret: MMR}(\mathcal{R}) \text{ is the minimax regret of feasible reward set } \mathcal{R}. \text{ This can be seen as a game between a decision maker (DM) choosing } f \text{ to minimize loss relative to the optimal policy, and an adversary selecting } r \text{ to maximize this loss given the DM’s choice. Any } f^* \text{ that minimizes max regret is a minimax optimal policy, while the } r \text{ that maximizes regret of } f^* \text{ is the adversarial reward, and the optimal policy } g \text{ for } r \text{ is the adversarial policy. Minimax regret measures performance by assessing the policy ex post and makes comparisons only w.r.t. specific reward realizations. Thus, policy } \pi \text{ is penalized on reward } r \text{ only if there exists a } \pi' \text{ that has higher value w.r.t. } r \text{ itself.}

Apart from producing robust policies using an intuitively appealing criterion, minimax regret is also an effective driver of reward elicitation. Unlike maximin, regret provides guidance as to maximal possible improvement in value should we obtain further information about the reward. Regan and Boutilier (2009) develop an elicitation strategy in which a user is queried about relevant reward data based on the current minimax regret solution. It is empirically shown to reduce regret very quickly and give rise to provably optimal policies for the underlying MDP with very little reward information.

**Using Nondominated Policies**

While minimax regret is a natural robustness criterion and effectively guides elicitation, it is computationally complex. Computing the regret optimal policy for an IRMDP is NP-hard (Xu and Mannor 2009), and empirical studies using a
mixed-integer program (MIP) model with constraint generation (Regan and Boutilier 2009) show poor scaling (we discuss this further below). Hence further development of practical algorithms is needed.

We focus on the use of nondominated policies to ease the burden of minimax regret computation in IRMDPs. In what follows, assume a fixed IRMDP with feasible reward set $\mathcal{R}$. We say policy $f$ is nondominated w.r.t. $\mathcal{R}$ iff

$$\exists r \in \mathcal{R} \text{ s.t. } f \cdot r \geq f' \cdot r, \quad \forall f' \in \mathcal{F}$$

In other words, a nondominated policy is optimal for some feasible reward. Let $\Gamma(\mathcal{R})$ denote the set of nondominated policies w.r.t. $\mathcal{R}$; since $\mathcal{R}$ is fixed, we write $\Gamma$ for simplicity.

**Observation 1.** For any IRMDP and policy $f$, $\text{argmax}_g \text{PMR}(f,g,\mathcal{R}) \in \Gamma$.

Thus the adversarial policy used to maximize regret of $f$ must lie in $\Gamma$, since an adversary can only maximize regret by choosing some $r \in \mathcal{R}$ and an optimal policy $f^*_r$ for $r$. If the set of nondominated policies is relatively small, and can be identified easily, then we can exploit this fact.

Define $V(r) = \max_{f \in \mathcal{F}} f \cdot r$ to be the optimal value obtainable when $r \in \mathcal{R}$ is the true reward. Since policy value is linear in $r$, $V$ is piecewise linear and convex (PWLC), much like the belief-state value function in POMDPs (Cheng 1988; Kaelbling, Littman, and Cassandra 1998), a fact we exploit below. Fig. 1 illustrates this for a simplified 1-D reward, with nondominated policy set $\Gamma = \{f_1, f_2, f_3, f_5\}$ ($f_4$ is dominated, i.e., optimal for no reward).

Xu and Mannor (2009) propose a method that exploits nondominated policies, computing minimax regret using the following linear program (LP), which “enumerates” $\Gamma$ and has $O(|\mathcal{R}| |\Gamma|)$ variables:

$$\text{minimize}_{z,e,\delta} \quad \delta$$

subject to:

$$\sum_{i=1}^{t} c_i = 1$$

$$e \geq 0$$

$$\delta \geq b^\top z(i)$$

$$A^\top z(i) + \Gamma e = f_i$$

$$z(i) \geq 0$$

(9)

This LP, LP-ND1, reduces the representation of the DM’s policy choice using Eq. (2) rather than a convex combination of nondominated policies:

$$\text{maximize} \quad (f^\top - c^\top \Gamma^\top) r$$

subject to: $Ar \leq b$ (10)

Here $\mathcal{R}$ is defined by inequalities $Ar \leq b$, and $\Gamma$ is a matrix whose columns are elements of $\Gamma$. The variables $c$ encode a randomized policy with support set $\Gamma$, which they show must be minimax optimal. For each potential adversarial policy $f_i \in \Gamma$, equations Eq. (10) encode the dual of

We refer to this approach as LP-ND1. (Xu and Mannor (2009) provide no computational results for this formulation.)

We can modify LP-ND1 to obtain the following LP (encoding the DM’s policy choice using Eq. (2) rather than a convex combination of nondominated policies):

$$\text{minimize}_{z,e,\delta} \quad \delta$$

subject to:

$$\gamma E^\top f + \beta = 0$$

$$\delta \geq b^\top z(i)$$

$$A^\top z(i) + f = f_i$$

$$z(i) \geq 0$$

$$i = 1,2,\ldots,t$$

(11)

This LP, LP-ND2, reduces the representation of the DM’s policy from $O(|\Gamma|)$ to $O(|S||A|)$ variables. Empirically, we find that usually $|\Gamma| \gg |S||A|$ (see below).

Rather than solving a single, large LP, we can use the constraint generation approach of Regan and Boutilier (2009), solving a series of LPs:

$$\min_{f,\delta} \quad \delta$$

subject to:

$$\delta \geq r_i \cdot g_i - r_i \cdot f \quad \forall \langle g_i, r_i \rangle \in \text{GEN}$$

$$\gamma Ef + f = 0$$

(12)

Here GEN is a subset of generated constraints corresponding to a subset of possible adversarial choices of policies and rewards. If GEN contains all vertices $r$ of polytope $\mathcal{R}$ and corresponding optimal policies $g^*_r$, this LP solves minimax regret exactly. However, most constraints will not be active so iterative generation is used: given a solution $f$ to the relaxed problem with only a subset of constraints, we wish to find the most violated constraint, i.e., the pair $r, g^*_r$ that maximizes regret of $f$. If no violated constraints exist, then solution $f$ is optimal. In (Regan and Boutilier 2009), violated constraints are computed by solving a MIP (the major computational bottleneck). However, we can instead exploit Obs. 1 and solve, for each $g \in \Gamma$, a small LP to determine which reward gives $g$ maximal advantage over the current relaxed solution $f$:

$$\text{maximize}_{r} \quad g \cdot r - f \cdot r$$

subject to: $Ar \leq b$

The $g$ with largest objective value determines the maximally violated constraint. Thus we replace the MIP for violated constraints in (Regan and Boutilier 2009) with a set of smaller LPs, and denote this approach by ICG-ND.

We compare these three approaches to minimax regret computation using nondominated policies, as well as the
Generating Nondominated Policies

While the effectiveness of ICG-ND in exploiting the nondominated set $\Gamma$ seems evident, the question remains: how to identify $\Gamma$? The PWLC nature of the function $V(r)$ is analogous to the situation in POMDPs, where policy value is linear in belief state. For this reason, we adapt a well-known POMDP algorithm Witness (Kaelbling, Littman, and Cassandra 1998) to iteratively construct the set of nondominated policies. As discussed below, other POMDP methods can be adapted to this problem as well.

The $\pi$Witness Algorithm

Let $\mathbf{f}$ be the occupancy frequencies for policy $\pi$. Suppose, when starting at state $s$ we take action $a$ rather than $\pi(s)$ as prescribed by $\pi$, but follow $\pi$ thereafter. The occupancy frequencies induced by this local adjustment to $\pi$ are given by:

$$
\mathbf{f}^{s,a} = \beta(s)(\mathbf{e}^{s,a} + \gamma \sum_{s'} \Pr(s'|s,a)\mathbf{f}(s')) + (1 - \beta(s))\mathbf{f}
$$

where $\mathbf{e}^{s,a}$ is an $S \times A$ vector with a 1 in position $s,a$ and zeroes elsewhere. It follows from standard policy improvement theorems (Puterman 1994) that if $\mathbf{f}$ is not optimal for reward $r$, there then must be a local adjustment $s,a$ such that $\mathbf{f}^{s,a} \cdot r > \mathbf{f} \cdot r$.1 This gives rise to a key fact:

**Theorem 1.** Let $\Gamma' \subseteq \Gamma$ be a (strictly) partial set of nondominated policies. Then there is an $\mathbf{f} \in \Gamma'$, an $(s,a)$, and an $\mathbf{r} \in \mathcal{R}$ such that $\mathbf{f}^{s,a} \cdot \mathbf{r} > \mathbf{f} \cdot \mathbf{r}$, $\forall \mathbf{f} \in \Gamma'$.

This theorem is analogous to the witness theorem for POMDPs (Kaelbling, Littman, and Cassandra 1998) and suggests a Witness-style algorithm for computing $\Gamma$. Our $\pi$Witness algorithm begins with a partial set $\Gamma$ consisting of a single nondominated policy optimal for an arbitrary $\mathbf{r} \in \mathcal{R}$. At each iteration, for all $\mathbf{f} \in \Gamma$, it checks whether there is a local adjustment $(s,a)$ and a witness reward $\mathbf{r}$ s.t. $\mathbf{f}^{s,a} \cdot \mathbf{r} > \mathbf{f} \cdot \mathbf{r}$ for all $\mathbf{f}^* \in \Gamma$ (i.e., whether $\mathbf{f}^{s,a}$ offers an improvement at $\mathbf{r}$). If there is an improvement, we add the optimal policy $\mathbf{f}^*_r$ for that $r$ to $\Gamma$. If no improvement exists for any $\mathbf{f}$, then by Thm. 1, $\Gamma$ is complete. The algorithm

---

1We assume $\beta$ is strictly positive for ease of exposition. Our definitions are easily modified if $\beta(s) = 0$ for some $s$. 
Algorithm 1: The $\pi$Witness algorithm

\[
\begin{align*}
    r & \leftarrow \text{some arbitrary } r \in R \\
    f & \leftarrow \text{findBest}(r) \\
    \Gamma & \leftarrow \{ f \} \\
    \text{agenda} & \leftarrow \{ f \} \\
    \text{while agenda is not empty do} \\
    & \quad f \leftarrow \text{next item in agenda} \\
    & \quad \text{foreach } s, a \text{ do} \\
    & \quad \quad r^{w} \leftarrow \text{findWitnessReward}(f^{s,a}, \Gamma) \\
    & \quad \quad \text{while witness found do} \\
    & \quad \quad \quad f_{\text{best}} \leftarrow \text{findBest}(r^{w}) \\
    & \quad \quad \quad \text{add } f_{\text{best}} \text{ to } \Gamma \\
    & \quad \quad \quad \text{add } f_{\text{best}} \text{ to agenda} \\
    & \quad \quad r^{w} \leftarrow \text{findWitnessReward}(f^{s,a}, \Gamma) \\
    \end{align*}
\]

is sketched in Alg. 1. The agenda holds the policies for which we have not yet explored all local adjustments. \text{findWitnessReward} tries to find an $r$ for which $f^{s,a}$ has higher value than any $f' \in \Gamma$ by solving the LP:

\[
\begin{align*}
    \text{maximize} & \quad \delta \\
    \text{subject to:} & \quad \delta \leq f^{s,a} \cdot r - f' \cdot r \quad \forall f' \in \Gamma \\
    & \quad A r \leq b
\end{align*}
\]

There may be multiple witnesses for a single adjustment, thus \text{findWitnessReward} is called until no more witnesses are found. \text{findBest} finds the optimal policy given $r$. The order in which the agenda is processed can have an impact on anytime behavior, a fact we explore in the next section.

We can see that the runtime of the $\pi$Witness algorithm is polynomial in inputs $|S|$, $|A|$, $|R|$ (interpreted as the number of constraints defining the polytope), and output $|\Gamma|$, assuming bounded precision in the input representation. When a witness $r^{w}$ is found, it testifies to a nondominated $f$ which is added to $\Gamma$ and the agenda. Thus, the number of policies added to the agenda is exactly $|\Gamma|$. The subroutine \text{findWitnessReward} is called at most $|S||A|$ times for each $f \in \Gamma$ to test local adjustments for witness points (total of $|\Gamma||S||A|$ calls). \text{findWitnessReward} requires solving an LP with $|S||A| + 1$ variables and no more than $|\Gamma| + |R|$ constraints, thus the LP encoding has polynomial size (hence solvable in polynomial time). \text{findBest} is called only when a witness is found, i.e., exactly $|\Gamma|$ times. It requires solving an MDP, which is polynomial in the size of its specification (Puterman 1994). Thus $\pi$Witness is polynomial. This also means that for any class of MDPs with a polynomial number of nondominated policies, minimax regret computation is itself polynomial.

Empirical Results

The number of nondominated policies is influenced largely by the dimensionality of the reward function and less so by conventional measures of MDP size, $|S|$ and $|A|$. Intuitively, this is so because a high dimensional $r$ allows variability across the state-action space, admitting different optimal policies depending on the realization of reward. When reward is completely unrestricted (i.e., the $r(s, a)$ are “independent”), we saw above that even small MDPs can admit a huge number of nondominated policies. However, in practice, reward functions typically have significant structure. \text{Factored MDPs} (Boutilier, Dean, and Hanks 1999) have large state and action spaces defined over sets of state variables; and typically reward depends only on a small fraction of these, often in an additive way. In our empirical investigation of $\pi$Witness, we exploit this fact, exploring how its performance varies with reward dimension.

We first test $\pi$Witness on MDPs of varying sizes, but with reward of small fixed dimension. States are defined by 2–6 binary variables (yielding $|S| = 4 \ldots 64$), and a factored additive reward function on two attributes: $r(s) = r_1(x_1) + r(x_2)$. The transition model and feasible reward set $R$ is generated randomly as above, with random reward intervals generated for the parameters of each factor rather than for each $(s, a)$-pair. Table 1 shows the number of nondominated policies discovered (with mean ($\mu$) and standard deviation ($\sigma$) over 20 runs), and demonstrates that $\Gamma$ does not grow appreciably with $|S|$, as expected with 2-D reward. The running time of $\pi$Witness is similar, growing slightly greater than linearly in $|S|$. We also examine MDPs of fixed size (6 attributes, $|S| = 64$), varying the dimensionality of the reward function from 2–8 by varying the number of additive reward attributes from 1–4. Results (20 instances of each dimension) are shown Table 2. While $\Gamma$ is very small for dimensions 2 and 4, it grows dramatically with reward dimensionality, as does the running time of $\pi$Witness. This demonstrates the strong impact of the size of the output set $\Gamma$ on the running time of $\pi$Witness.

Approximating the Nondominated Set

The complexity of both $\pi$Witness and our procedure ICGND are influenced heavily by the size of $\Gamma$; and while the

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
State & Number of Vectors & $\pi$Witness Runtime (secs) \\
\hline
Size & $\mu$ & $\sigma$ & $\mu$ & $\sigma$ \\
\hline
4 & 3.463 & 2.231 & 0.064 & 0.045 \\
8 & 3.772 & 3.189 & 0.145 & 0.144 \\
16 & 7.157 & 5.743 & 0.433 & 0.329 \\
32 & 7.953 & 6.997 & 1.228 & 1.062 \\
64 & 11.251 & 9.349 & 4.883 & 3.981 \\
\hline
\end{tabular}
\caption{Varying Number of States}
\end{table}

\begin{table}[h]
\centering
\begin{tabular}{|c|c|c|c|c|}
\hline
Reward Dim. & Number of Vectors & $\pi$Witness Runtime (secs) \\
\hline
& $\mu$ & $\sigma$ & $\mu$ & $\sigma$ \\
\hline
2 & 2.050 & 0.887 & 1.093 & 0.634 \\
4 & 10.20 & 10.05 & 4.554 & 4.483 \\
6 & 759.6 & 707.4 & 1178 & 1660 \\
8 & 6116 & 5514 & 80642 & 77635 \\
\hline
\end{tabular}
\caption{Varying Dimension of Reward Space}
\end{table}
number of nondominated policies scales reasonably well with MDP size, it grows quickly with reward dimensionality. This motivates investigation of methods that use only a subset of the nondominated policies that reasonably approximates Γ, or specifically, the PWLC function $V_{\Gamma}(-)$ induced by Γ. We first explore theoretical guarantees on minimax regret when ICG-ND (or any other method that exploits $\pi$Witness) is run using a (hopefully, well-chosen) subset of dominated policies (marked with a *) shows the error introduced by using the subset of the nondominated policies that reasonably approximates $\Gamma$. This error is illustrated in Fig. 1, where the dashed line is the approximation when adversarial policy choice is unrestricted and $\pi$Witness offers a lower bound on $V_{\Gamma}$. Define the error in $V_{\Gamma}$ to be maximum difference between the approximate and exact value functions:

$$\epsilon(\tilde{\Gamma}) = \max_{r \in R} V_{\tilde{\Gamma}}(r) - V_{\Gamma}(r)$$

This error is illustrated in Fig. 1, where the dashed line (marked with a *) shows the error introduced by using the subset of dominated policies $\{f_1, f_2, f_3\}$ (removing $f_2$). The error in $V_{\Gamma}$ can be used to derive a bound on error in computed minimax regret. Let $MMR(\Gamma)$ denote true minimax regret when adversarial policy choice is unrestricted and $MMR(\tilde{\Gamma})$ denote the approximation when adversarial choice is restricted to $\tilde{\Gamma}$. $MMR(\tilde{\Gamma})$ offers a lower bound on true MMR; and the difference $\epsilon_{MMR}(\tilde{\Gamma})$ can be bounded, as can the difference between the true max regret of the approximately optimal policy so constructed:

**Theorem 2.**

$$\epsilon_{MMR}(\tilde{\Gamma}) = MMR(\Gamma) - MMR(\tilde{\Gamma}) \leq \epsilon(\tilde{\Gamma}); \text{ and}$$

$$MR(\tilde{\Gamma}, R) - MMR(\tilde{\Gamma}) \leq 2\epsilon(\tilde{\Gamma}).$$

Thus, should we generate a set of nondominated policies $\tilde{\Gamma}$ that $\epsilon$-approximates $\Gamma$, any algorithm (including ICG-ND) that uses nondominated sets will produce a policy that is within a factor of $2\epsilon(\Gamma)$ of minimizing max regret.

This suggests that careful enumeration of nondominated policies can provide tremendous computational leverage. By adding policies to $\Gamma$ that “contribute” the most to error reduction, we may be able to construct a partial set $\tilde{\Gamma}$ of small size, but that closely approximates $\Gamma$. Indeed, as we discuss below, the agenda in $\pi$Witness can be managed to help accomplish just this. We note that a variety of algorithms for POMDPs attempt to build up partial sets of $f$-vectors to approximate a value function (e.g., (Cheng 1988)) and we are currently investigating the adaptation of such methods to nondominated policy enumeration as well.

**$\pi$Witness Anytime Performance**

We can construct a small approximating set $\tilde{\Gamma}$ using $\pi$Witness by exploiting its anytime properties and careful management of the agenda. Intuitively, we want to add policies to $\tilde{\Gamma}$ that hold the greatest “promise” for reducing error $\epsilon(\tilde{\Gamma})$. We measure this as follows. Let $\tilde{\Gamma}_n$ be the $n$th nondominated set produced by $\pi$Witness, constructed by adding optimal policy $f^*_n$ for the $n$th witness point $r_n$. When $f^*_n$ is added to the agenda, it offers improvement to the current approximation:

$$\Delta(f^*_n) = V_{\Gamma_n}(r_n) - V_{\Gamma_{n-1}}(r_n).$$

We process the agenda in priority queue fashion, using $\Delta(f)$ as the priority measure for any policy $f$ remaining on the agenda. Thus, we examine adjustments to policies that provided greater increase in value when added to $\tilde{\Gamma}$ before considering adjustments to policies that provided lesser value.

Informal experiments show that using a priority queue reduced the error $\epsilon(\tilde{\Gamma})$ much more quickly than using standard stack or queue approaches. Hence we investigate the anytime performance of $\pi$Witness with a priority queue on random MDPs with 128 and 256 states (30 runs of each). The reward dimension is fixed to 6 (3 additive factors) and the number of actions to 5. We first compute the exact minimax regret for the MDP, then run $\pi$Witness. When the $n$th nondominated policy is found, we compute an approximation of

---

![Figure 4](image-url) Relative minimax regret error and cumulative $\pi$Witness runtime vs. number of nondominated policies.

![Figure 5](image-url) $\pi$Witness computation time (hrs.) vs. number of nondominated policies.
minimax regret using the algorithm ICG-ND with approximate nondominated set $\Gamma_n$. We measure the relative error in minimax regret: $\epsilon_{\text{MMR}}(\hat{\Gamma}) / \text{MMR}$.

Fig. 4 shows the relative error as nondominated policies are added using the priority queue implementation. The runtime of ICG-ND algorithm for computing minimax regret is also shown. With 256 (resp., 128) states, relative error drops below 0.02 after just 500 (resp., 300) policies have been added to $\Gamma$. Minimax regret computation using ICG-ND grows linearly with the number of nondominated policies added to $|\Gamma|$, but stays well below 1 second: at the 0.02 error point, solution of 256-state (resp., 128-state) MDPs averages under 0.4 seconds (resp., 0.2 seconds). Given our goal of using minimax regret to drive preference elicitation, these results suggest that using a small set of nondominated policies and the ICG-ND algorithm will admit real-time interaction with users. Critically, while $\pi$Witness is much more computationally intensive, it can be run offline, once, to precompute nondominated policies (or a small approximate set) before engaging in online elicitation with users. Fig. 5 shows the cumulative runtime of $\pi$Witness as it adds policies to $\Gamma$. With 256 states, the first 500 policies (error level 0.02) are generated in under 2 hours on average (128 states, under 1 hour). In both cases, runtime $\pi$Witness is only slightly super-linear in the number of policies.

**Conclusion**

We presented a new class of techniques for solving IR-MDPs that exploit nondominated policies. We described new algorithms for computing robust policies using minimax regret that leverage the set $\Gamma$ of nondominated policies, and developed the $\pi$Witness algorithm, an exact method for computing $\Gamma$ in polynomial time. We showed how low-dimensional factored reward allows $\pi$Witness to scale to large state spaces, and examined the impact of approximate nondominated sets, showing that small sets can yield good, quickly computable approximations to minimax regret.

Some important directions remain. We are investigating methods to compute tight bounds on minimax regret error while generating nondominated policies, drawing on algorithms from the POMDP literature (e.g., Cheng’s (1988) linear support algorithm). An algorithm for generating nondominated policies that yields a bound at each step, allows termination when a suitable degree of approximation is reached. We are exploring the integration with preference elicitation as well. A user provides reward information (e.g., by responding to queries), to reduce reward imprecision and improve policy quality. Since this constrains the feasible reward space, fewer nondominated policies result; thus as elicitation proceeds, the set of nondominated policies can be pruned allowing more effective computation. Finally, we are interested in the formal relationship between the number of nondominated policies and reward dimensionality.

**References**

Bagnell, A.; Ng, A.; and Schneider, J. 2003. Solving uncertain Markov decision problems. Technical Report CMU-RI-TR-01-25, Carnegie Mellon University, Pittsburgh.

Boger, J.; Poupart, P.; Hoey, J.; Boutilier, C.; Fernie, G.; and Mihailidis, A. 2006. A planning system based on Markov decision processes to guide people with dementia through activities of daily living. *IEEE Transactions on Information Technology in Biomedicine* 10(2):323–333.

Boutilier, C.; Das, R.; Kephart, J. O.; Tesouro, G.; and Walsh, W. E. 2003. Cooperative negotiation in autonomic systems using incremental utility elicitation. In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence (UAI-03)*, 89–97.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Brazier, D., and Boutilier, C. 2005. Local utility elicitation in GAI models. In *Proceedings of the Twenty-first Conference on Uncertainty in Artificial Intelligence (UAI-05)*, 42–49.

Cheng, H.-T. 1988. *Algorithms for Partially Observable Markov Decision Processes*. Ph.D. Dissertation, University of British Columbia, Vancouver.

Delage, E., and Mannor, S. 2007. Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML-07)*, 225–232.

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

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101(1-2):99–134.

McMahan, B.; Gordon, G.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, 536–543.

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

Puterman, M. 1994. *Markov decision processes: Discrete stochastic dynamic programming*. Wiley, New York.

Regan, K., and Boutilier, C. 2009. Regret-based reward elicitation for Markov decision processes. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI-09)*.

Savage, L. J. 1954. *The Foundations of Statistics*. New York: Wiley.

Xu, H., and Mannor, S. 2009. Parametric regret in uncertain Markov decision processes. In *48th IEEE Conference on Decision and Control*, 3606–3613.
C Tech Report

Robust policy computation in reward-uncertain Mdp's using nondominated policies.
June, 2010.

This paper develops a novel anytime algorithm for constructing the set of nondominated policies, with provable (anytime) error bounds on minimax regret. The paper show the quality of the approximation can be improved online by pruning/adding nondominated policies during reward elicitation, while maintaining computational tractability. The performance of these techniques is assessed using a small variant of the COACH assistive technologies domain.
Robust Online Optimization of Reward-uncertain MDPs using Linear Support

Kevin Regan
Department of Computer Science
University of Toronto
Toronto, ON
kmregan@cs.toronto.edu

Craig Boutilier
Department of Computer Science
University of Toronto
Toronto, ON
cebly@cs.toronto.edu

Abstract

Imprecise-reward Markov decision processes (IRMDPs) can be used to model MDPs in which the reward function is only partially specified (e.g., by some elicitation process). Recent work on robust policy computation using minimax regret shows IRMDPs to be intractable, but has demonstrated how the set of policies that are nondominated with respect to reward uncertainty can be leveraged to more efficiently compute minimax regret. However, the number of nondominated policies is generally so large as to undermine this leverage. Drawing insights from the POMDP literature, we develop a new anytime algorithm for constructing the set of nondominated policies, with provable (anytime) error bounds on minimax regret. Furthermore, we show how the quality of the approximation can be improved online by pruning/adding nondominated policies during reward elicitation, while maintaining computational tractability.

1 Introduction

The use of Markov decision processes (MDPs) to model decision problems under uncertainty requires the specification of a large number of model parameters to capture both system dynamics and state/action rewards. This specification remains a key challenge: often system dynamics can be learned, though residual uncertainty in estimated parameters often remains; and reward specification typically requires sophisticated human judgement to assess relevant tradeoffs. For this reason, considerable attention has been paid to finding robust solutions to MDPs whose parameters that are imprecisely specified. For instance, algorithms have been developed that find policies that are robust (in the maximin sense) to transition probability uncertainty, offering guarantees on solution quality in the face of such uncertainty [2, 9, 14].

A line of recent work has developed techniques for computing robust solutions for imprecise reward MDPs (IRMMDPs) [7, 16, 20]. The specification of rewards can be especially problematic, since reward functions cannot generally be learned from experience (except for the most simple objectives involving observable metrics). Reward assessment requires the translation of general user preferences, and tradeoffs of the relative desirability of states and actions, into precise quantities—an extremely difficult task, as is well-documented in the decision theory literature [8]. Furthermore, this time-consuming process may need to be repeated for different users (with different preferences). Encouragingly, a fully specified reward function is often not necessary to make optimal (or near-optimal) decisions [16]. IRMDPs allow one to describe an MDP in which the reward function can be any in a specified set \( \mathcal{R} \) (e.g., reflecting imprecise bounds on reward parameters).

In this paper, we address the problem of fast, online, robust solutions of IRMDPs. We adopt minimax regret as our robustness criterion [16, 17, 20], but solving IRMDPs using this measure is NP-hard [20]. Despite this, several techniques have been developed that allow the solution of small IRMDPs.
Of particular note are methods that exploit the set \( \Gamma \) of nondominated policies, i.e., those policies that are optimal for some element of \( \mathcal{R} \) [17, 20]. Unfortunately, these methods scale directly with the number of dominated policies used; and \( \Gamma \) is often too large to admit good computational performance. A subset \( \tilde{\Gamma} \) of the nondominated set can be used as an approximation by these techniques and, if specific bounds on the approximate set are known, error bounds on the minimax solution can be derived [17]; but methods for producing a suitable approximate set are lacking.

In the paper, we develop a new algorithm for constructing approximations to the set of nondominated policies \( \tilde{\Gamma} \). Drawing on parallels between IRMDPs and POMDPs, we adapt Cheng's [6] classic linear support algorithm for POMDPs to IRMDPs. This algorithms allows us to generate \( \tilde{\Gamma} \) in an anytime fashion, and make explicit tradeoffs between the quality of the approximation (hence error in minimax regret) and the number of nondominated policies in the approximating set \( \tilde{\Gamma} \) (hence solution time). In general, we need to restrict the number of nondominated policies to admit efficient solution. However, in an online elicitation setting, the set of feasible rewards \( \mathcal{R} \) shrinks as users respond to queries about their preferences. This means that some undominated policies in \( \tilde{\Gamma} \) may become dominated and can thus be eliminated from consideration. This is turn permits additional nondominated policies to be added to \( \tilde{\Gamma} \), allowing improvement, as elicitation proceeds, in both our approximation of \( \tilde{\Gamma} \) and in decision quality. We develop a model for this form of online, robust optimization of IRMDPs, and demonstrate its value empirically.

We first review relevant background on MDPs, IRMDPs and minimax regret. We then develop the linear support algorithm for IRMDPs. Next we discuss how the set of nondominated policies can be optimized online during reward elicitation and we evaluate these techniques on random MDPs and on a “real-world” MDP for cognitive assistance.

2 Imprecise Reward MDPs

MDPs We assume an infinite horizon MDP \( \langle \mathcal{S}, \mathcal{A}, \{ P_{sa} \}, \gamma, \beta, r \rangle \), with finite state set \( \mathcal{S} \), finite action set \( \mathcal{A} \), transition distributions \( P_{sa}(\cdot) \) over next states (given action \( a \) taken in state \( s \)), discount factor \( \gamma \), initial state distribution \( \beta \), and reward function \( r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R} \). A policy \( \pi \) maps states to actions, and its value is given by \( V^\pi = \sum_{s \in \mathcal{S}} \beta(s) \mathbb{E} \left[ \sum_{i=1}^{\infty} \gamma^{i-1} r(s_i, \pi(s_i)) \mid \pi \right] \). Here the expectation is over the sequence of states induced by the policy \( \pi \). Our objective is to find an optimal policy \( \pi^* \), where \( V^* = V^{\pi^*} \geq V^\pi, \forall \pi \).

A policy \( \pi \) induces occupancy frequencies \( f^\pi(s, a) \) which reflect the total discounted probability of being in state \( s \) and taking action \( a \). Given such frequencies \( f^\pi \), the policy can be recovered via \( \pi(s, a) = f^\pi(s, a) / \sum_a f^\pi(s, a) \). Because of this direct correspondence, we treat occupancy frequencies and policies interchangeably in what follows. For ease of exposition we make use of the following vector notation: \( \mathbf{r} \) is an \( |\mathcal{S}||\mathcal{A}| \) vector with entries \( r(s, a) \); \( \mathbf{f} \) is an \( |\mathcal{S}|\) vector with entries \( f(s, a) \); and \( \mathbf{P} \) is an \( |\mathcal{S}|\times|\mathcal{S}| \) transition matrix. Define matrix \( \mathbf{E} \) to be identical to \( \mathbf{P} \) with 1 subtracted from each self-transition probability \( P_{ss} \). The set of valid occupancy probabilities for a fixed MDP is given by \( \mathcal{F} = \{ \mathbf{f} \mid \mathbf{E} \mathbf{f} + \beta = 0 \} \) [15]. We can express the optimal value function in terms of occupancy frequencies: \( V^* = \arg\max_{\mathbf{f} \in \mathcal{F}} \mathbf{f} \cdot \mathbf{r} \).

Imprecise-reward MDPs Specification of reward functions often requires sophisticated human judgements and tradeoffs to be made regarding the precise value of specific states and actions. To minimize the burden of reward elicitation, it is often desirable to elicit only partial reward information [16, 17]. We can model this using an imprecise reward MDP (IRMMDP) \( \langle \mathcal{S}, \mathcal{A}, \{ P_{sa} \}, \gamma, \beta, \mathcal{R} \rangle \), in which the reward function \( r \) is replaced by feasible reward set \( \mathcal{R} \). Intuitively, any reward function \( r \in \mathcal{R} \) might represent the user’s preferences, with \( \mathcal{R} \) determined using some form of preference assessment or elicitation. We restrict attention in what follows to the case where \( \mathcal{R} \) is a bounded convex polytope defined by linear constraint set \( \{ r \mid A r \leq b \} \) and denote the number of constraints by \( |\mathcal{R}| \). Such linear constraints arise naturally in reward assessment: a priori bounds on plausible reward values from a domain expert; user responses to elicitation queries comparing reward, policies or trajectories [17]; or policy observation (as in inverse reinforcement learning [13]).

To compute robust policies w.r.t. \( \mathcal{R} \) we use the minimax regret criterion [5, 18] which has been recently adopted for IRMDPs [17, 20]. Let \( \mathbf{f} \) be an occupancy frequency (induced by some policy) and \( \mathbf{r} \) a reward function. We define \( R(f, r) = \max_{\mathbf{g} \in \mathcal{F}} [\mathbf{g} \cdot \mathbf{r} - \mathbf{f} \cdot \mathbf{r}] \) to be the regret of policy \( \mathbf{f} \).
w.r.t. reward function \( r \). Regret measures the difference in value between \( f \) and the optimal policy given \( r \). Define pairwise max regret to be \( \text{PMR}(f, g, R) = \max_{\text{f} \in \mathcal{F}} (g \cdot r - f \cdot r) \); i.e., the maximal difference in value between \( g \) and \( f \) over all possible rewards. Define:

\[
\begin{align*}
MR(f, R) &= \max_{r \in R} R(f, r) = \max_{g \in \mathcal{F}} \text{PMR}(f, g, R) \\
\text{MMR}(R) &= \min_{f \in \mathcal{F}} MR(f, R) = \min_{f \in \mathcal{F}} \max_{r \in R} \max_{g \in \mathcal{F}} g \cdot r - f \cdot r
\end{align*}
\]

\( MR(f, R) \) is the max regret of a policy \( f \) w.r.t. to the feasible reward set \( R \). It captures the worst case loss over all possible realizations of reward. \( \text{MMR}(R) \) is the minimax regret of a feasible reward set \( R \), and the occupancy frequency \( f \) that minimizes max regret (and the corresponding policy) is the minimax optimal policy. This definition can be interpreted as a game in which a decision maker chooses policy \( f \) to minimize the loss relative to the optimal policy and an adversary chooses the reward \( r \) to maximize this loss given \( f \). The minimax regret criterion compares favorably to the maximin robustness measure also used in the robust MDP literature [9, 11, 14]. While maximin value is more computationally tractable, it leads to conservative policies since the policy is being optimized against the worst case realization of reward. Minimax regret offers a more intuitive measure of performance by assessing the loss of a policy (relative to the optimal policy) given an instantiation of reward, and is empirically a more effective driver of elicitation than maximin [16].

Several recent approaches to computing minimax regret rely on the concept of policies being nondominated w.r.t. reward polytope \( R \) [17, 20]. Formally, we say \( f \) is nondominated w.r.t. \( R \) iff

\[
\exists r \in R \text{ s.t. } f \cdot r \geq f' \cdot r \quad \forall f' \in \mathcal{F}.
\]

Let \( \Gamma_R \) denote the set of all nondominated policies w.r.t. to \( R \); we omit the subscript when \( R \) is fixed and clear from context.

Following [17] (and assuming a fixed \( R \)), we define \( \mathcal{V}(r) = \max_{f \in \mathcal{F}} f \cdot r \) to be the optimal value obtainable when \( r \in R \) is the true reward. Since policy value is linear in \( r \), \( \mathcal{V} \) is piecewise linear and convex (PWLC), much like the belief-state value function in POMDPs [6, 10]. Since dominated policies \( f \) cannot contribute to this value, we can define \( \mathcal{V}_\Gamma(r) = \max_{f \in \mathcal{F}} f \cdot r \), and immediately see that \( \mathcal{V} = \mathcal{V}_\Gamma \). Fig. 1 illustrates this for a simplified 1-D reward, with nondominated policy set \( \Gamma = \{f_1, f_2, f_3, f_5\} \) (\( f_4 \) is dominated, i.e., optimal for no point in reward space). As we discuss below, there are reasons we might want to approximate \( \mathcal{V} \) using a subset \( \hat{\Gamma} \subseteq \Gamma \) of nondominated policies. We define the \( \mathcal{V} \)-error in this approximation to be [17]:

\[
\epsilon_\mathcal{V}(\hat{\Gamma}, R) = \max_{r \in R} \mathcal{V}_\Gamma(r) - \mathcal{V}_{\hat{\Gamma}}(r)
\]

As an example, Fig. 1 shows the \( \mathcal{V} \)-error induced by the approximate set \( \hat{\Gamma} = \{f_1, f_3, f_5\} \) (omitting \( f_2 \)) as a dashed line marked by an asterisk.

**Computing Minimax Regret** While computing minimax regret for IRMDPs is NP-hard [20], several techniques have been proposed for its computation [16, 17, 20]. For example, one can exploit the observation that the adversarial policy (or “witness”) \( g \) in Eq. 2 must be nondominated [17, 20]; defining

\[
\text{MMR}(\Gamma) = \min_{f \in \mathcal{F}} \max_{r \in R} \max_{g \in \Gamma} g \cdot r - f \cdot r
\]
we immediately obtain that $\text{MMR}(\mathcal{R}) = \text{MMR}(\Gamma)$. In [17] this observation is exploited in a simple constraint generation procedure for minimax regret and is shown to outperform other algorithms on small, random MDPs. The method solves a series of linear programs (LPs):

$$\begin{align*}
\text{minimize} & \quad \delta \\
\text{subject to:} & \quad \delta \geq r_i \cdot g_i - r_i \cdot f \\
& \quad \forall (g_i, r_i) \in \text{GEN} \tag{5}
\end{align*}$$

where GEN is a set of constraints corresponding to a subset of possible adversarial choices of $r$ (and the corresponding policies). Were GEN to contain all vertices of polytope $\mathcal{R}$, this LP would capture minimax regret exactly. However, since most of these constraints will not be binding at the optimal solution, constraint generation is used. Given the solution $f$ to a relaxed problem with subset GEN, we find the most violated constraint, i.e., the $r, g$-pair that maximizes the regret of $f$. If no violated constraint exists, then $f$ is the minimax optimal policy.

Since the adversarial policy must lie in the nondominated set $\Gamma$, we can find the most violated constraint by solving a small LP for each $g \in \Gamma$:

$$\begin{align*}
\text{maximize} & \quad g \cdot r - f \cdot r \\
\text{subject to:} & \quad Ar \leq b \tag{6}
\end{align*}$$

The $g$ with the largest objective value determines the maximally violated constraint. This approach is very efficient if the set of nondominated policies is small. However, this is not often the case. In the next section we discuss the generation of nondominated policies and how one can exploit an approximation of the set $\Gamma$.

### 3 The Linear Support Algorithm

While the constraint generation process described above can work very effectively if the nondominated set $\Gamma$ is small, generally $\Gamma$ can be quite large, even for small MDPs. As a result, generating constraints by solving LPs for each $g \in \Gamma$ often makes constraint generation impractical. As shown in [17] however, approximations of the set $\Gamma$ can be used in the constraint generation procedure. If $\tilde{\Gamma} \subseteq \Gamma$ is a subset of nondominated policies with $\forall$-error $\epsilon_V(\tilde{\Gamma}, \mathcal{R})$, then the error in minimax regret using this approximate set is bounded by $\epsilon_V(\tilde{\Gamma}, \mathcal{R})$; i.e., $\text{MMR}(\Gamma) - \text{MMR}(\tilde{\Gamma}) \leq \epsilon_V(\tilde{\Gamma}, \mathcal{R})$. Furthermore, the true max regret of the approximate minimax optimal policy $\tilde{f}$ induced by $\tilde{\Gamma}$ is close to the max regret of the optimal policy: $\text{MR}(\tilde{f}) - \text{MMR}(\tilde{\Gamma}) \leq 2\epsilon_V(\tilde{\Gamma}, \mathcal{R})$.

This motivates the development of algorithms that can produce small, high quality approximations $\tilde{\Gamma}$ whose error can be bounded; small sets allow optimization (e.g., the constraint generation procedure) to run quickly, while if $\tilde{\Gamma}$ closely approximates $\Gamma$ (i.e., has small error), the resulting policy will indeed be close to minimax optimal. Regan & Boutilier [17] provide an algorithm for generating $\Gamma$ based on the Witness algorithm for POMDPs [10]. They also suggest heuristic, anytime modifications of the algorithm for computing partial sets $\tilde{\Gamma}$ with small error. However, the heuristics do not allow theoretical bounds on error to be derived.

We develop a new algorithm for the generation of nondominated policies that remedies this deficiency. The linear support algorithm for constructing the nondominated policy set for an IRMDP is a direct adaptation of Cheng’s [6] classic linear support method for POMDPs. The intuitions are straightforward. Given an approximate set $\tilde{\Gamma}$, let $\mathcal{R}_f(\tilde{\Gamma}) \equiv \{ r \in \mathcal{R} \mid f \cdot r \geq f' \cdot r, \forall f' \in \tilde{\Gamma} \}$ be the nondominated region of policy $f$ w.r.t. $\tilde{\Gamma}$, i.e., that region of $\mathcal{R}$ for which $f$ is the best policy in $\tilde{\Gamma}$. The nondominated region $\mathcal{R}_f(\tilde{\Gamma})$ is convex over any such region (since error is defined as the difference between $V$, which is PWLC, and $V_f$, which is linear over $\mathcal{R}_f(\tilde{\Gamma}))$. Hence the maximum of $\epsilon_V(\tilde{\Gamma}, \mathcal{R})$ over $\mathcal{R}_f(\tilde{\Gamma})$ must lie at a vertex of $\mathcal{R}_f(\tilde{\Gamma})$. As a consequence, the maximum error must lie at the vertex of the nondominated region of some $f \in \tilde{\Gamma}$.

**Lemma 1.** $\epsilon_V(\tilde{\Gamma}, \mathcal{R})$ is maximal at a vertex of the nondominated region $\mathcal{R}_f(\tilde{\Gamma})$ for some $f \in \tilde{\Gamma}$.

The linear support algorithm exploits this fact by computing error only at vertices of such regions, and adding (optimal) policies to $\tilde{\Gamma}$ only for vertices with maximal error. The algorithm begins with an initial nondominated policy $f$—an optimal policy for some arbitrary $r \in \mathcal{R}$—in $\tilde{\Gamma}$. It determines
Algorithm 1: Policy Linear Support

Let $\delta$ be allowable error, and $r_0$ some vertex of $R$

$\tilde{\Gamma} \leftarrow \emptyset$ subset of nondominated policies

$E \leftarrow \{r_0\}$ vertices of the nondominated regions

$E' \leftarrow \emptyset$ vertices with error below threshold $\delta$

$\epsilon_V(r_0) \leftarrow \infty$ Initial error

while $E - E' \neq \emptyset$ do

1. $r' \leftarrow \arg\max_{r \in E - E'} \epsilon_V(r)$
2. $f_{r'} \leftarrow \arg\max_{E \in \tilde{\Gamma}} f \cdot r'$
3. $E \leftarrow E \cup \{r'\}$
4. $E' \leftarrow E' \cup \{r'\}$

foreach $r \in E - E'$ do

3. if $\epsilon_V(r) \leq \delta$ then

   $E' \leftarrow E' \cup \{r\}$

the next policy to add by: (a) computing $E_{\tilde{\Gamma}}$, the set of vertices of the nondominated regions of $\tilde{\Gamma}$; (b) computing the optimal policy $f_{r'}$ for each $r \in E_{\tilde{\Gamma}}$; and (c) selecting the policy that offers the greatest improvement, i.e., such that the error $r_{r'} - \max_{r \in \tilde{\Gamma}} r g$ is maximal. The selected policy is added to $\tilde{\Gamma}$ and the process repeated until the maximum error at any vertex falls below some acceptable threshold (or some other termination criterion is met). The algorithm is sketched in Alg. 1. It is not hard to see that:

Theorem 1. Linear support with error threshold $\delta$ outputs a set $\tilde{\Gamma}$ s.t. $\epsilon_V(\tilde{\Gamma}, R) \leq \delta$.

As with Cheng’s algorithm, a number of efficiencies are incorporated into our adaptation of linear support to the IRMDP setting. For example, caching is used to prevent duplication of work while computing the max error (see lines 1–1), finding the optimal policy (lines 1–2) and computing the error at each new vertex (lines 1–3). We use an implementation of the LRS backward search algorithm for vertex enumeration [1] and CPLEX 11.1.1 for solving LPs.

Evaluation. We test linear support on small MDPs and compare it to $\pi$ Witness, an existing method for generating nondominated policies [17]. We test both methods on small, randomly generated IRMDPs with factored, additive reward functions. A state $s = \{x_1, x_2, \ldots, x_7\}$ is composed of 7 binary variables, yielding $|S| = 128$. We use two different reward functions: the first $r(s) = r_1(x_1) + r_2(x_2) + r_3(x_3)$ with dimension 6; and the second $r(s) = r_1(x_1) + r_2(x_2) + r_3(x_3) + r_4(x_4)$ with dimension 8. For each MDP we generate a transition model where each $(s, a)$-pair has $\log_2 |S|$ random, nonzero transition probabilities. The imprecise reward polytope $R$ is generated as follows: 1) for each $(s, a)$ we select an underlying “true” reward $r(s, a)$ uniformly from a predefined range; 2) we generate the uncertain interval of random size (normally distributed); and 3) we randomly (uniformly) place the interval “around” the true reward $r(s, a)$. We generate 20 MDPs of each reward dimension, and run each algorithm to completion, generating all nondominated policies.

Fig. 2 shows the average runtime of each algorithm. While linear support is more efficient than $\pi$ Witness, the performance gap narrows as reward dimensionality increases. The most striking advantage of linear support is the availability of a bound on error $\epsilon_V(\tilde{\Gamma}, R)$ at each iteration. Fig. 3(a) shows that the error $\epsilon_V$ drops quickly with each new nondominated policy added (note the log scale). For example, $\epsilon_V(\tilde{\Gamma}, R)$ is reduced to under 1.0% of its initial value (i.e., when $\tilde{\Gamma} = \emptyset$) after only 500 policies are added, and to nearly 0.1% after only 2000 policies. A small set $\tilde{\Gamma}$ of nondominated policies can be used to quickly approximate minimax regret as discussed above (and examined in detail in the next section). We note that $\tilde{\Gamma}$ can usually be computed only once (offline), prior to elicitation. Minimax regret, conversely, usually needs to be computed repeatedly, and online, since it is an integral component of many elicitation schemes (as discussed below).

---

1Further implementation details can be found in forthcoming technical report.
4 Online Optimization

The results above show that linear support can be used effectively to find the full set of nondominated policies for a specific feasible reward set $R$; more importantly, its anytime properties allow relatively small sets of nondominated policies to be constructed that are very good approximations w.r.t. $V$-error. While linear support can be run offline to construct $\tilde{\Gamma}$, its ability to produce small sets $\tilde{\Gamma}$ with small error, allows for the fast computation of policies that are approximately minimax optimal. This is significant because IRMDPs are especially useful in the context where the reward function is being incrementally elicited [16]. In such a setting, fast computation of minimax regret is vital, since it must take place online to support the querying process. In addition, the feasible reward set $R$ shrinks as more information is gleaned about the actual reward (e.g., as users respond to queries or behavior is observed), a fact we can also exploit.

If $R' \subseteq R$ is a refinement of $R$, then it must have fewer nondominated policies, i.e., $\Gamma_{R'} \subseteq \Gamma_R$. This means that policies that were nondominated w.r.t. $R$ may become dominated when the feasible reward set is reduced to $R'$. Since the computational performance of constraint generation using the nondominated set is tightly tied to its size, pruning away newly dominated policies can offer tremendous speed up in minimax regret computation.

Pruning of $\Gamma_R$—or its approximation $\tilde{\Gamma}$—can be realized using a simple LP test (analogous to the domination test for POMDPs proposed by Monahan [12]. Specifically, for each $f \in \tilde{\Gamma}$, we solve the following LP to find a reward point at which $f$ is nondominated:

$$\begin{align*}
\text{maximize} & \quad \delta \\
\text{subject to:} & \quad \delta \leq f \cdot r - f' \cdot r \\
& \quad \forall \quad f' \in \tilde{\Gamma} \setminus f \\
& \quad r \in R'
\end{align*} \quad (7)$$

If the objective is negative, then $f$ is dominated and can be pruned from $\tilde{\Gamma}$.\(^3\)

While pruning can speed up online computation, it can also be used to “create space” to add new nondominated policies to the approximate set $\tilde{\Gamma}$. Thus we can improve the quality of the approximation by adding new policies, while maintaining the same online computational overhead by using pruning to keep the size of $\tilde{\Gamma}$ roughly constant. Adding new policies is a simple matter of continuing further iterations of the linear support algorithm, since it has suitable anytime behavior.\(^4\)

This leads to the following online optimization algorithm. Offline (prior to elicitation), we can compute an initial approximate set $\Gamma_0$ for the prior feasible set $R_0$ of nondominated policies using

\(^2\)We outline one specific elicitation process below.

\(^3\)We can also quickly determine whether $f \in \tilde{\Gamma}$ remains nondominated by testing if the vertex $r$ at which it was found to be optimal by linear support remains feasible. If $r \in R'$ (i.e., if it satisfies the new constraint(s) that refine $R$), then $f$ remains nondominated. If $r \not\in R'$, then we resort to LP (7).

\(^4\)The entire set $\Gamma$ could also be computed offline, and policies selectively added as elicitation proceeds.
linear support. The size of $\tilde{\Gamma}_t$ is determined by the demands for efficient online minimax regret computation. At iteration $t$, the minimax regret optimization is solved for $\mathcal{R}_{t-1}$ using $\tilde{\Gamma}_{t-1}$: most elicitation schemes require the minimax optimal solution to generate new queries or decide when to terminate the elicitation process [5, 16]. The new set $\mathcal{R}_t$ is formed (incorporating constraints induced by the query response), and the set $\tilde{\Gamma}_t$ is constructed by pruning policies in $\tilde{\Gamma}_{t-1}$ that are dominated relative to $\mathcal{R}_t$ and using linear support to add new policies to $\tilde{\Gamma}_t$.

**Evaluation** The linear support algorithm, in conjunction with constraint generation, permits the fast, approximate solution of minimax regret; and our online optimization scheme offers the potential to improve the quality of this approximate solution during elicitation, while maintaining tractability. We now examine how this potential is realized in a specific elicitation setting.

Reward elicitation can proceed using a variety of queries. In our tests, we use *bound queries* of the form “Is $r(s, a) \geq b$?” for simplicity. The choice of query (i.e., which $(s, a)$-pair to ask about and the parameter $b$) is dictated by the current solution (CS) heuristic [5, 16, 19]: let the gap for any $(s, a)$-pair be $\Delta(s, a) = \max_{r \in \mathcal{R}} r(s, a) - \min_{r \in \mathcal{R}} r(s, a)$; then CS queries the point $(s, a)$ with the largest weighted gap $f(s, a)\Delta(s, a)$, with weight given by occupancy frequency $f(s, a)$ in the current solution $f$ to the minimax regret problem. The bound $b$ is selected to be the midpoint of the gap $\Delta(s, a)$. A (yes or no) response to the bound query imposes a linear constraint on $\mathcal{R}$.

Each step of elicitation involves the following: 1) Minimax regret is computed w.r.t. the current reward polytope $\mathcal{R}$; 2) the minimax regret solution is used to select a query using the CS heuristic; 3) the query response imposes an additional constraint refining $\mathcal{R}$. Elicitation terminates once a satisfactory level of max regret $\varepsilon$ has been reached. When a nondominated set $\Gamma$ is used to approximate the minimax optimal, we simply require that $\text{MMR}(\tilde{\Gamma}, \mathcal{R}) + \epsilon_Y(\tilde{\Gamma}, \mathcal{R}) \leq \varepsilon$ (w.r.t. the current reward polytope $\mathcal{R}$). Online optimization improves solution quality and decreases $\epsilon_Y$, allowing elicitation to terminate sooner by reaching this termination threshold earlier.

We examine the online approach using “real-world” elicitation scenario using a variant of the COACH MDP model, which models a system that provides cognitive assistance for persons with dementia to enable them to complete common activities of daily living [3, 4]. Very roughly, the goal in this model is to guide a person through a common task (e.g., hand-washing) by providing verbal or visual cues, while allowing the individual to maintain as much independence as possible. We assume a general task of $\ell$ steps. The system can issue prompts at increasing levels of intrusiveness, or can call a caregiver (e.g., therapist or family member) to assist the person in task completion. This results in action space $\mathcal{A} = \{0, 1, \ldots, k\}$ where 0 indicates no prompt was issued, level $k - 1$
indicates the strongest most intrusive prompt and level $k$ indicates that the caregiver was called. The state is defined by three variables $S = \langle T, D, F \rangle$ where $T = \{0, 1, \ldots, \ell\}$ is the number of tasks steps successfully completed by the person, $D = \{0, 1, 2, 3, 4, 5+\}$ is the delay (time taken during the the current step); and $F = \{0, 1, \ldots, k\}$ tracks whether a prompt at a specific level was attempted on the current task step and failed to immediately get the person to the next step. The dynamics express the following intuitions. The no-prompt action will cause a “progress” transition to the next step (setting delay and failed-prompt to zero), or a “stall” transition (same step with delay increased by one). The probability of reaching the next step with action $a = n$ is higher than $a = n - 1$ since more intrusive prompts have a better chance of facilitating progress; however, progress probability decreases as delay increases. Reaching the next step after prompting is less likely if a prompt has already failed at the current step.

The reward function is $r(t, d, f, a) = r_{\text{goal}}(t) + r_{\text{progress}}(d = 0) + r_{\text{delay}}(d) + r_{\text{prompt}}(a)$, where: $r_{\text{goal}}(t)$ is a large positive reward when $t = k$ for completing the task and is zero when $t < k$; $r_{\text{progress}}(d = 0)$ is a (small) positive reward for progressing to step $t$ (indicated by $d$ being reset to zero); $r_{\text{delay}}(d)$ is a small negative reward for delay in completing a step; and $r_{\text{prompt}}(a)$ is the negative cost associated with prompting the person (calling the caregiver is generally very costly).

We set $\ell = 14$, $k = 6$ and create an IRMDP by setting initial reward bounds for $R$ in a manner similar to that in the previous section. The resulting IRMDP has size $|S||A|=3012$ and reward dimensionality $|R|=12$. We use linear support to generate $\hat{\Gamma}$ with the following criterion in mind. We wish to allow for interactive response times during elicitation, so we choose the size of $\hat{\Gamma}$ so that $MMR(\hat{\Gamma})$ takes no more than one second to compute. This results in $\hat{\Gamma}$ containing less than 5% of all nondominated policies. We further assume that during elicitation there are 10 seconds available while waiting for a user response to perform online optimization (pruning and addition) of $\hat{\Gamma}$. During elicitation we compute minimax regret using: a static set $\hat{\Gamma}$ with error $\epsilon_V$; and a dynamic set $\hat{\Gamma}'$ with decreasing error $\epsilon'_V$, optimized online by pruning and adding policies during the 10 seconds created by response latency. Fig. 3(b) shows the upper bounds $MMR(\hat{\Gamma}) + \epsilon_V$ and $MMR(\hat{\Gamma}') + \epsilon'_V$ on the max regret of the recommended policies as elicitation proceeds (as a fraction of the initial upper bound). Clearly, online refinement of $\hat{\Gamma}$ offers significant benefits: for example, if the user is satisfied with a max regret improvement of $\epsilon=0.25$ of initial regret, online optimization of $\hat{\Gamma}$ allows termination after 16 fewer queries than a static $\hat{\Gamma}$. Indeed, online refinement allows max regret to reach zero (after about 60 queries), something that is not possible with the static set ($\epsilon_V$ alone remains greater than 0.15). Thus we see that a small $\hat{\Gamma}$ with less than 5% of all nondominated policies enables very effective reward elicitation if we allow online refinement.

5 Discussion and Conclusion

We have presented a method for fast computation of approximate, robust solutions to imprecise-reward MDPs. Using insights from POMDPs to generate approximate sets of nondominated policies with provable error bounds with an adaptation of the linear support algorithm, we showed how this set can be leveraged to efficiently approximate minimax regret using existing constraint generation techniques. We further described how the set of nondominated policies can be optimized online as imprecise reward is refined during elicitation (by pruning newly dominated policies and inserting additional nondominated policies). On a small, “real-world” elicitation scenario, we showed that this refinement process allows approximation error to quickly decrease to zero using a fraction of the full set of nondominated policies.

Our framework removes a significant computational impediment to online reward elicitation for MDPs. There are a number of directions in which this work can be extended of course. Developing further computational enhancements to improve both online regret computation using nondominated policies and offline construction of nondominated sets remains of interest. Directly integrating these techniques into the solution of factored (graphical) MDP representations is also important. Finally, we have investigated only rudimentary elicitation queries in this work: the development of new queries and strategies—expanding the vocabulary for eliciting MDP reward functions—and the incorporation of other sources of information (such as observed behavior [13]) for reward assessment are important next steps.
References

[1] David Avis. A revised implementation of the reverse search vertex enumeration algorithm. In G. Kalai and G. Ziegler, editors, *Polytopes—Combinatorics and Computation*, pages 177–198. Birkhauser-Verlag, Basel, 2000.

[2] Andrew Bagnell, Andrew Ng, and Jeff Schneider. Solving uncertain Markov decision problems. Technical Report CMU-RI-TR-01-25, Carnegie Mellon University, Pittsburgh, 2003.

[3] Jennifer Boger, Pascal Poupart, Jesse Hoey, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-05)*, pages 1293–1299, Edinburgh, 2005.

[4] Jennifer Boger, Pascal Poupart, Jesse Hoey, Craig Boutilier, Geoff Fernie, and Alex Mihailidis. A planning system based on Markov decision processes to guide people with dementia through activities of daily living. *IEEE Transactions on Information Technology in Biomedicine*, 10(2):323–333, 2006.

[5] Craig Boutilier, Relu Patrascu, Pascal Poupart, and Dale Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artificial Intelligence*, 170(8–9):686–713, 2006.

[6] Hsien-Te Cheng. *Algorithms for Partially Observable Markov Decision Processes*. PhD thesis, University of British Columbia, Vancouver, 1988.

[7] Erick Delage and Shie Mannor. Percentile optimization in uncertain Markov decision processes with application to efficient exploration. In *Proceedings of the Twenty-fourth International Conference on Machine Learning (ICML-07)*, pages 225–232, Corvallis, OR, 2007.

[8] Simon French. *Decision Theory*. Halsted Press, New York, 1986.

[9] G. Iyengar. Robust dynamic programming. *Mathematics of Operations Research*, 30(2):1–21, 2005.

[10] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2):99–134, 1998.

[11] Brendan McMahan, Geoffrey Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML-03)*, pages 536–543, Washington, DC, 2003.

[12] George E. Monahan. A survey of partially observable Markov decision processes: Theory, models and algorithms. *Management Science*, 28:1–16, 1982.

[13] Andrew Ng and Stuart Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML-00)*, pages 663–670, Stanford, CA, 2000.

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

[15] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, New York, 1994.

[16] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. In *25th Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 444–451, Montreal, 2009.

[17] Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain mdps using nondominated policies. In *Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, Montreal, 2010. to appear.

[18] Leonard J. Savage. *The Foundations of Statistics*. Wiley, New York, 1954.

[19] Paolo Viappiani and Craig Boutilier. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the Third ACM conference on Recommender Systems (RecSys-09)*, pages 101–109, New York, 2009.

[20] Huan Xu and Shie Mannor. Parametric regret in uncertain Markov decision processes. In *48th IEEE Conference on Decision and Control*, pages 3606–3613, Shanghai, 2009.