Robust Online Optimization of Reward-Uncertain MDPs

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

Abstract

Imprecise-reward Markov decision processes (IR-MDPs) are MDPs in which the reward function is only partially specified (e.g., by some elicitation process). Recent work using minimax regret to solve IRMDPs has shown, despite their theoretical intractability, how the set of policies that are *nondominated* w.r.t. reward uncertainty can be exploited to accelerate regret computation. However, the number of nondominated policies is generally so large as to undermine this leverage. In this paper, we show how the quality of the approximation can be improved online by pruning/adding nondominated policies during reward elicitation, while maintaining computational tractability. Drawing insights from the POMDP literature, we also develop a new anytime algorithm for constructing the set of nondominated policies with provable (anytime) error bounds. These bounds can be exploited to great effect in our online approximation scheme.

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 rewards. This specification remains a key challenge: while dynamics can be learned from data, 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 are imprecisely specified (e.g., computing robust policies, in the *maximin* sense, given transition probability uncertainty [2; 9; 12]).

Recently, techniques for computing robust solutions for *imprecise reward MDPs (IRMDPs)* have been proposed [7; 15; 19]. 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 with respect to 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).

Fortunately, a fully specified reward function is often not needed to make optimal (or near-optimal) decisions [15]. IRMDPs are defined as MDPs in which the reward function lies in some set $\mathcal{R}$ (e.g., reflecting imprecise bounds on reward parameters). In this paper, we address the problem of fast, online computation of robust solutions for IRMDPs. We use *minimax regret* as our robustness criterion [15; 16; 19]. While solving IRMDPs using this measure is NP-hard [19], several techniques have been developed that allow the solution of small IRMDPs. Of particular note are methods that exploit the set Γ of *nondominated policies*, i.e., those policies that are optimal for some element of $\mathcal{R}$ [16; 19]. Unfortunately, these methods scale directly with the number of dominated policies; and Γ is often too large to admit good computational performance. A subset $\tilde{\Gamma}$ of the nondominated set can be used as an approximation, and, if specific bounds on the approximation quality of that set are known, error bounds on the minimax solution can be derived [16]; but methods for producing a suitable approximate set with the requisite bounds are lacking.

We develop an approach for approximating minimax regret during elicitation by adjusting the subset $\tilde{\Gamma}$ online. It allows us to make explicit tradeoffs between the quality of the approximation and the efficiency of minimax regret computation. In online elicitation of rewards, the feasible reward set $\mathcal{R}$ shrinks as users respond to queries about their preferences. This means that some undominated policies in $\tilde{\Gamma}$ become dominated and can be ignored, thus improving online computational efficiency. This, in turn, permits further nondominated policies to be added to $\tilde{\Gamma}$, allowing for improvement in decision quality. To support online elicitation and computation, we also develop a new algorithm for constructing the set $\tilde{\Gamma}$ in an anytime fashion that provides an upper bound on minimax regret. This algorithm is based on insights from Cheng's [6] linear support method for POMDPs.

We first review relevant background on MDPs, IRMDPs and minimax regret. We then discuss how nondominated policies exploited online during reward elicitation, and develop the *nondominated/region vertex (NRV)* algorithm for

generating nondominated policies for IRMDPs. We evaluate our methods on random MDPs and a "real-world" MDP for cognitive assistance.

2 Imprecise Reward 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 states (given action a taken in state s), discount factor γ, initial state distribution β, and reward function $r : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$. A *policy* π maps states to actions, and its value is given by $V^\pi = \sum_{s_1 \in \mathcal{S}} \beta(s_1) \mathbb{E}\left[\sum_{i=1}^{\infty} \gamma^{i-1} r(s_i, \pi(s_i)) \mid \pi\right]$ (where expectation is taken over the state sequence induced by π). We seek an *optimal policy* π^* s.t. $V^* = V^{\pi^*} \geq V^\pi, \forall \pi$.

A policy π induces *occupancy frequencies* $f^\pi(s,a)$, giving the total discounted probability of being in state s and taking action a. Given f^π, 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 use the following vector notation: $\mathbf{r}$ is an $|\mathcal{S}||\mathcal{A}|$ 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. Define matrix $\mathbf{E}$ to be identical to $\mathbf{P}$ with 1 subtracted from each self-transition probability $P_{sa}(s)$. The set of valid occupancy probabilities for a fixed MDP is given by $\mathcal{F} \equiv \{\mathbf{f} \mid \gamma \mathbf{E}^\top \mathbf{f} + \beta = 0\}$ [14]. We can express the optimal value function in terms of occupancy frequencies: $V^* = \text{argmax}_{\mathbf{f} \in \mathcal{F}} \mathbf{f} \cdot \mathbf{r}$.

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 [15; 16]. We can model this using an *imprecise reward MDP (IRMDP)* $\langle \mathcal{S}, \mathcal{A}, \{P_{sa}\}, \gamma, \beta, \mathcal{R} \rangle$, 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 $\{r \mid \mathbf{A}\mathbf{r} \leq \mathbf{b}\}$ 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 [16]; or policy observation (as in inverse reinforcement learning [11]).

To compute robust policies w.r.t. $\mathcal{R}$ we use the *minimax regret criterion* [5; 18], recently adopted for IRMDPs [16; 19]. Let $\mathbf{f}$ be an occupancy frequency (induced by some policy) and $\mathbf{r}$ a reward function. Define $R(\mathbf{f}, \mathbf{r}) = \max_{\mathbf{g} \in \mathcal{F}} \mathbf{g} \cdot \mathbf{r} - \mathbf{f} \cdot \mathbf{r}$ to be the *regret* of policy $\mathbf{f}$ w.r.t. $\mathbf{r}$. Regret measures the difference in value between $\mathbf{f}$ and the optimal policy given $\mathbf{r}$. Define *pairwise max regret* to be $PMR(\mathbf{f}, \mathbf{g}, \mathcal{R}) = \max_{\mathbf{r} \in \mathcal{R}} \mathbf{g} \cdot \mathbf{r} - \mathbf{f} \cdot \mathbf{r}$; i.e., the maximum value difference between $\mathbf{g}$ and $\mathbf{f}$ over all possible rewards. Define:

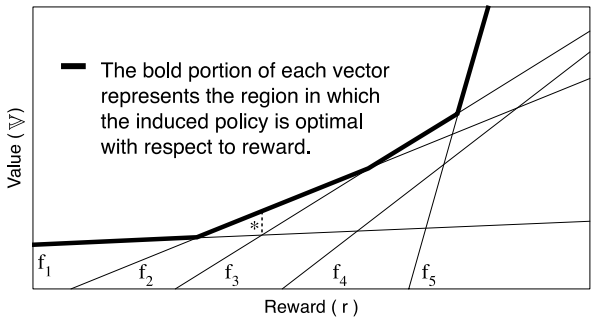

Figure 1: Illustration of value as a linear function of reward.

$$MR(\mathbf{f}, \mathcal{R}) = \max_{\mathbf{r} \in \mathcal{R}} R(\mathbf{f}, \mathbf{r}) = \max_{\mathbf{g} \in \mathcal{F}} PMR(\mathbf{f}, \mathbf{g}, \mathcal{R}) \quad (1)$$

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

$MR(\mathbf{f}, \mathcal{R})$ is the *max regret* of a policy $\mathbf{f}$ with respect to the feasible reward set $\mathcal{R}$. This is simply the worst case loss over all possible realizations of reward. $MMR(\mathcal{R})$ is the *minimax regret* of a feasible reward set $\mathcal{R}$, and the occupancy frequency $\mathbf{f}$ 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 $\mathbf{f}$ to minimize loss relative to the optimal policy and an adversary chooses the reward $\mathbf{r}$ to maximize this loss given $\mathbf{f}$. The minimax regret criterion compares favorably to the *maximin* robustness measure also used in the robust MDP literature [9; 10; 12]. 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 much more effective driver of elicitation than maximin [15].

Several recent approaches to computing minimax regret rely on the concept of policies being *nondominated* w.r.t. reward polytope $\mathcal{R}$ [16; 19]. Formally, we say $\mathbf{f}$ is nondominated w.r.t. $\mathcal{R}$ iff

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

Let $\Gamma_{\mathcal{R}}$ denote the set of all nondominated policies w.r.t. to $\mathcal{R}$; we omit the subscript when $\mathcal{R}$ is clear from context.

We define $\mathbb{V}(\mathbf{r}) = \max_{\mathbf{f} \in \mathcal{F}} \mathbf{f} \cdot \mathbf{r}$ to be the optimal value obtainable when $\mathbf{r} \in \mathcal{R}$ is the true reward. Since policy value is linear in $\mathbf{r}$, $\mathbb{V}$ is *piecewise linear and convex (PWLC)*. Since dominated policies $\mathbf{f}$ cannot contribute to this value, we can define $\mathbb{V}_\Gamma(\mathbf{r}) = \max_{\mathbf{f} \in \Gamma} \mathbf{f} \cdot \mathbf{r}$, and immediately see that $\mathbb{V} = \mathbb{V}_\Gamma$. Fig. 1 illustrates this for a simplified 1-D reward, with nondominated policy set $\Gamma = \{\mathbf{f}_1, \mathbf{f}_2, \mathbf{f}_3, \mathbf{f}_5\}$ ($\mathbf{f}_4$ is dominated, i.e., optimal for no point in reward space).

Given a subset $\tilde{\Gamma} \subseteq \Gamma$ of nondominated policies, the function $\mathbb{V}_{\tilde{\Gamma}}$ approximates the optimal value function. The $\mathbb{V}$-

error of this approximation at point $\mathbf{r}$ is:

$$\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathbf{r}) = \mathbb{V}_{\Gamma}(\mathbf{r}) - \mathbb{V}_{\tilde{\Gamma}}(\mathbf{r}). \quad (3)$$

Define the *(global)* $\mathbb{V}$*-error* of the approximation as whole:

$$\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R}) = \max_{\mathbf{r}\in\mathcal{R}} \varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathbf{r}) = \max_{\mathbf{r}\in\mathcal{R}} \mathbb{V}_{\Gamma}(\mathbf{r}) - \mathbb{V}_{\tilde{\Gamma}}(\mathbf{r}). \quad (4)$$

In Fig. 1, an asterisk marks the point defining (global) $\mathbb{V}$-error of the approximate set $\tilde{\Gamma} = \{\mathbf{f}_1, \mathbf{f}_3, \mathbf{f}_5\}$, which omits $\mathbf{f}_2$ from the nondominated set Γ (the marked, dashed line indicates the magnitude of this error).

While computing minimax regret for IRMDPs is NP-hard [19], several techniques have been proposed for its computation [19; 15; 16]. For example, one can exploit the observation that the adversarial policy (or "witness") $\mathbf{g}$ in Eq. 2 must be nondominated [19; 16]; defining

$$MMR(\Gamma, \mathcal{R}) = \min_{\mathbf{f}\in\mathcal{F}} \max_{\mathbf{r}\in\mathcal{R}} \max_{\mathbf{g}\in\Gamma} \mathbf{g}\cdot\mathbf{r} - \mathbf{f}\cdot\mathbf{r} \quad (5)$$

we immediately obtain that $MMR(\mathcal{R}) = MMR(\Gamma, \mathcal{R})$. In [16] 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):

$$\begin{aligned} \underset{\mathbf{f},\delta}{\text{minimize}} \quad & \delta \\ \text{subj. to:} \quad & \delta \geq \mathbf{r}_i \cdot \mathbf{g}_i - \mathbf{r}_i \cdot \mathbf{f} \quad \forall \ \langle \mathbf{g}_i, \mathbf{r}_i \rangle \in \text{GEN} \\ & \gamma \mathbf{E}^{\top}\mathbf{f} + \beta = 0 \end{aligned}$$

where GEN is a set of constraints corresponding to a *subset* of possible adversarial choices of $\mathbf{r}$ (and the corresponding optimal policies w.r.t. $\mathbf{r}$). 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 $\mathbf{f}$ to a relaxed problem with subset GEN, we find the most violated constraint, i.e., the $\langle \mathbf{r}, \mathbf{g} \rangle$-pair that maximizes the regret of $\mathbf{f}$. If no violated constraint exists, then $\mathbf{f}$ is the minimax optimal policy.

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

$$\underset{\mathbf{r}}{\text{maximize}} \quad \mathbf{g}\cdot\mathbf{r} - \mathbf{f}\cdot\mathbf{r} \qquad \text{subject to:} \quad \mathbf{A}\mathbf{r} \leq \mathbf{b}$$

The $\mathbf{g}$ 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. A sufficiently small set $\tilde{\Gamma} \subset \Gamma$ can be substituted to efficiently approximate minimax regret. If we can bound the $\mathbb{V}$-error $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$, then the error in minimax regret using this approximate set is bounded by $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$; i.e., $MMR(\Gamma, \mathcal{R}) - MMR(\tilde{\Gamma}, \mathcal{R}) \leq \varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$ [16]. Furthermore, the true max regret of the approximately optimal policy $\tilde{\mathbf{f}}$ induced by $\tilde{\Gamma}$ is bounded: $MR(\tilde{\mathbf{f}}, \mathcal{R}) - MMR(\Gamma, \mathcal{R}) \leq 2\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$. Existing approaches to generating nondominated policies do not admit a bound on $\mathbb{V}$-error [16]; to address this, we develop an anytime algorithm in Sec. 4 that generates nondominated policies to directly minimize $\mathbb{V}$-error (with suitable bounds).

Algorithm 1: Online Adjustment during Elicitation

Initialization:
mdp $\leftarrow$ the parameters of our MDP model
$\mathcal{R}_0 \leftarrow$ initial reward polytope
$\tilde{\Gamma}_0 \leftarrow$ initial nondominated policies (computed offline)
$\tau \leftarrow$ acceptable level of regret
Elicitation:
foreach *step* $t \geq 1$ **do**
  $mmr, \mathbf{f}, \mathbf{g}, \mathbf{r} \leftarrow$ ComputeMMR(mdp,$\mathcal{R}_{t-1}, \tilde{\Gamma}_{t-1}$)
  response $\leftarrow$ SelectAndAskQuery($\mathbf{f}, \mathbf{g}, \mathbf{r}$,mdp)
  $\mathcal{R}_t \leftarrow$ Refine(response, $\mathcal{R}_{t-1}$)
  $\tilde{\Gamma}_t \leftarrow$ Prune($\mathcal{R}_t$,$\tilde{\Gamma}_{t-1}$) $\cup$ Add($\mathcal{R}_t, \tilde{\Gamma}_{t-1}$)
  if $mmr < \tau$ **then**
    terminate and return minimax optimal policy $\mathbf{f}$

3 Online Optimization

We now describe how one can best exploit an approximate set $\tilde{\Gamma}$ of nondominated policies, adjusting this set online to compute increasingly accurate approximations of minimax regret during reward elicitation. During elicitation, the feasible reward set $\mathcal{R}$ shrinks as more information is gleaned about the actual reward (e.g., as users respond to queries or behavior is observed). If $\mathcal{R}' \subset \mathcal{R}$ is the *refinement* of $\mathcal{R}$ given by this additional information, then $\Gamma_{\mathcal{R}'} \subseteq \Gamma_{\mathcal{R}}$, i.e., 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 Γ is tightly tied to its size, pruning away newly dominated policies can offer tremendous speed up in minimax regret computation.

Pruning of $\Gamma_{\mathcal{R}'}$—or its approximation $\tilde{\Gamma}$—can be realized as follows: for each $\mathbf{f} \in \tilde{\Gamma}$, we solve a small LP to find a reward point at which $\mathbf{f}$ is nondominated:

$$\begin{aligned} \underset{\mathbf{r},\delta}{\text{maximize}} \quad & \delta \\ \text{subj. to:} \quad & \delta \leq \mathbf{f}\cdot\mathbf{r} - \mathbf{f}'\cdot\mathbf{r}, \qquad \forall \ \mathbf{f}' \in \tilde{\Gamma} \setminus \mathbf{f} \quad (6) \\ & \mathbf{r} \in \mathcal{R}' \end{aligned}$$

If the objective δ is negative, then $\mathbf{f}$ is dominated and can be pruned from $\tilde{\Gamma}$. 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 to $\tilde{\Gamma}$, while maintaining the same online computational overhead by keeping the size of $\tilde{\Gamma}$ roughly constant through the effective use of pruning. Adding new policies is a simple matter of running further iterations of a nondominated policy generation algorithm with suitable anytime behavior: we develop just such an algorithm in the next section.[1]

These considerations lead to the following online optimization algorithm. Offline (prior to elicitation), we compute an initial approximate set $\tilde{\Gamma}_0$ of nondominated policies given the prior feasible reward set $\mathcal{R}_0$. The size of $\tilde{\Gamma}_0$ is determined by the demands of efficient online minimax regret

[1]The entire set Γ could also be computed offline, and policies selectively added as elicitation proceeds.

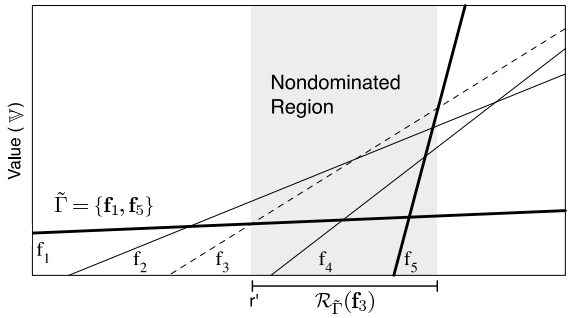

Figure 2: Illustration of a nondominated region.

computation. At iteration t, minimax regret is computed for $\mathcal{R}_{t-1}$ using $\tilde{\Gamma}_{t-1}$ (note that most elicitation schemes require the minimax optimal solution to generate new queries or decide when to terminate the elicitation process in any case [5; 15]). The new set $\mathcal{R}_t$ is formed (incorporating constraints given by the query response), and the set $\tilde{\Gamma}_t$ is constructed by: (a) pruning policies in $\tilde{\Gamma}_{t-1}$ that are dominated relative to $\mathcal{R}_t$; and (b) adding new nondominated policies to $\tilde{\Gamma}_{t-1}$ to improve approximation quality. Alg. 1 outlines how pruning and addition can be integrated into an elicitation algorithm.

Many variants of this scheme exist. The pruning and addition of policies need not be done in real time, but can take place in some parallel background process. Minimax regret w.r.t. $\mathcal{R}_t$ can be computed using a "lagging" set $\tilde{\Gamma}_{t-k}$ without detriment: error would be determined by the error of the lagging approximate set. And update of $\tilde{\Gamma}$ can take place asynchronously: whenever a set of update operations has been "completed" relative to any $\mathcal{R}_{t-k}$, it can be used at stage t.

4 Nondominated Region Vertex Algorithm

The ability to add the most "relevant" new policies to the approximate set $\tilde{\Gamma}$ in our online procedure allows error in minimax regret to be reduced significantly while maintaining good online computational performance. We now describe a principled anytime algorithm for generating an approximate set $\tilde{\Gamma}$ which directly minimizes value function error. Our algorithm is an adaptation of Cheng's [6] classic *linear support method* for POMDPs: rather than computing nondominated α-vectors over belief states, we compute policies that are nondominated w.r.t. uncertain reward. We note that unlike the heuristic technique for generating approximation sets $\tilde{\Gamma}$ proposed in [16], our algorithm comes with theoretical bounds on error, without which we could not offer the user a guarantee on maximum regret during elicitation.

Given an approximate set $\tilde{\Gamma}$, let $\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f})$ be the *nondominated region* of policy $\mathbf{f}$ w.r.t. $\tilde{\Gamma}$:

$$\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f}) \equiv \{r \in \mathcal{R} \mid \mathbf{f} \cdot \mathbf{r} \geq \mathbf{f}' \cdot \mathbf{r}, \forall\, \mathbf{f}' \in \tilde{\Gamma}\},$$

i.e., that region of $\mathcal{R}$ for which $\mathbf{f}$ is the best policy in $\tilde{\Gamma}$. Fig. 2 illustrates the nondominated region $\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f}_3)$ of policy $\mathbf{f}_3$ with respect to the approximate set $\tilde{\Gamma} = \{\mathbf{f}_1, \mathbf{f}_5\}$ (depicted in bold). The nondominated region $\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f})$ for any $\mathbf{f} \in \tilde{\Gamma}$ is a bounded, convex polytope. Furthermore, the error function $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathbf{r})$ is convex over any such region (since error is defined as the difference between $\mathbb{V}$, which is PWLC, and $\mathbb{V}_{\tilde{\Gamma}}$, which is linear over $\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f})$). Hence the maximum of $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathbf{r})$ over the region $\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f})$ must lie at a vertex of $\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f})$. For example, in Fig. 2 the maximum over $\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f}_3)$ is found at the vertex labelled $\mathbf{r}'$. As a consequence, the maximum error must lie at the vertex of the nondominated region of *some* $\mathbf{f} \in \tilde{\Gamma}$:

Lemma 1. *The reward that maximizes $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$ is found at the vertex of the nondominated region $\mathcal{R}_{\Gamma}(\mathbf{f})$ for some $\mathbf{f} \in \tilde{\Gamma}$.*

Algorithm 2: Nondominated Region Vertex algorithm

Let δ be allowable error, and $\mathbf{r}_0$ some vertex of $\mathcal{R}$
$\tilde{\Gamma} \leftarrow \emptyset$ *subset of nondominated policies*
$E \leftarrow \{\mathbf{r}_0\}$ *vertices of the nondominated regions*
$\varepsilon_{\mathbb{V}}(\mathbf{r}_0) \leftarrow \infty$ *(arbitrary) initial error*
$E' \leftarrow \emptyset$ *vertices with error below threshold δ*
while $E - E' \neq \emptyset$ **do**
1 $\quad \mathbf{r}' \leftarrow \text{argmax}_{\mathbf{r} \in E - E'} \varepsilon_{\mathbb{V}}(\mathbf{r})$
2 $\quad \mathbf{f}_{\mathbf{r}'} \leftarrow \text{argmax}_{\mathbf{f} \in \mathcal{F}} \mathbf{f} \cdot \mathbf{r}'$
$\quad \tilde{\Gamma} \leftarrow \tilde{\Gamma} \cup \{\mathbf{f}_{r'}\}$
$\quad E \leftarrow E \cup \textit{Vertices}(\mathcal{R}_{\tilde{\Gamma}}(\mathbf{f}_{r'}))$
$\quad E' \leftarrow E' \cup \{\mathbf{r}'\}$
$\quad$ **foreach** $\mathbf{r} \in E - E'$ **do**
3 $\quad\quad \varepsilon_{\mathbb{V}}(\mathbf{r}) \leftarrow \mathbb{V}(\mathbf{r}) - \mathbb{V}_{\tilde{\Gamma}}(\mathbf{r})$
$\quad\quad$ **if** $\varepsilon_{\mathbb{V}}(\mathbf{r}) \leq \delta$ **then**
$\quad\quad\quad E' \leftarrow E' \cup \{\mathbf{r}\}$

The *nondominated region/vertex (NRV)* algorithm exploits this fact by computing error only at vertices of such regions, and adding (optimal) policies to $\tilde{\Gamma}$ only for those vertices with maximal error. The algorithm (Alg. 2) begins with an initial nondominated policy $\mathbf{f}$ (optimal for some arbitrary $\mathbf{r} \in \mathcal{R}$) in $\tilde{\Gamma}$. It adds a policy by: (a) computing $E_{\tilde{\Gamma}}$, the set of vertices of the nondominated regions of $\tilde{\Gamma}$; (b) computing the optimal policy $\mathbf{f}_{\mathbf{r}}$ for each $\mathbf{r} \in E_{\tilde{\Gamma}}$; and (c) selecting the policy that offers the greatest improvement, i.e., such that the error $\mathbf{r}\mathbf{f}_{\mathbf{r}} - \max_{\mathbf{g} \in \tilde{\Gamma}} \mathbf{r}\mathbf{g}$ is maximal. The selected policy is added to $\tilde{\Gamma}$ and the process repeated until the maximum error at any vertex falls below an acceptable threshold (or some other termination criterion is met). We can show:

Theorem 1. *The NRV algorithm with error threshold δ outputs a set $\tilde{\Gamma}$ satisfying $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R}) \leq \delta$.*

As in [6], many efficiencies are exploited that enhance the high-level description in Alg. 2. For example, caching is used to eliminate duplication in computing max error (see line 1), finding the optimal policy (line 2) and computing error at each new vertex (line 3).[2] Secondary information generated by NRV can also be leveraged. By storing the vertex $\mathbf{r}$ at which each policy $\mathbf{f} \in \tilde{\Gamma}$ was found to be optimal, we can

[2]Further implementation details can be found in forthcoming technical report. We use the LRS backward search algorithm for vertex enumeration [1] and CPLEX 11.1.1 to solve LPs.

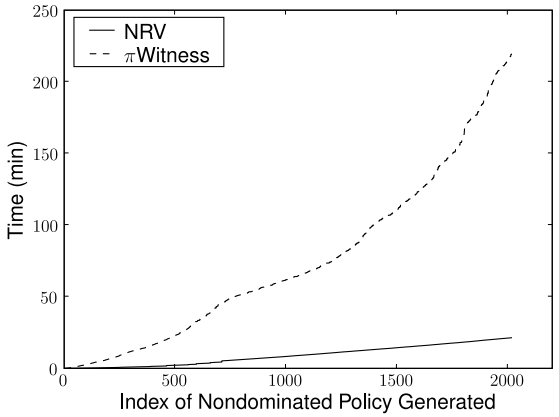

Figure 3: Time (mins.) to generate nondominated policies (20 instances) for both NRV and πWitness: $|\mathcal{R}|$=6.

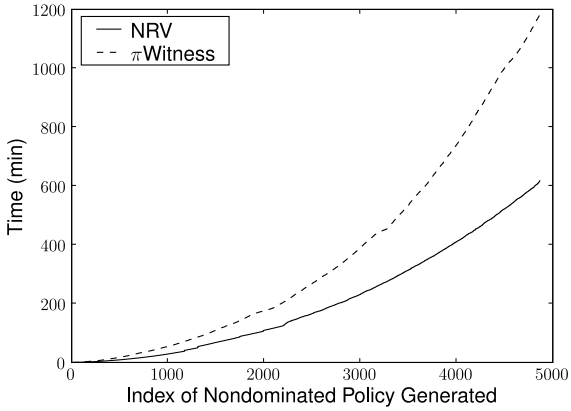

Figure 4: Time (mins.) to generate nondominated policies (20 instances) for both NRV and πWitness: $|\mathcal{R}|$=8.

quickly determine whether $\mathbf{f}$ remains nondominated by testing whether $\mathbf{r}$ remains feasible. If $\mathbf{r} \in \mathcal{R}'$ (i.e., satisfies the new constraints that refine $\mathcal{R}$), then $\mathbf{f}$ remains nondominated. If $\mathbf{r} \notin \mathcal{R}'$, then we resort to LP (6).

Related Work A connection exists between our approach to identifying nondominated policies and work on solving a restricted class of decentralized MDPs (DEC-MDPs) that feature transition and observation independence [3]. In this setting agents can be viewed as having nearly independent "local MDPs" that are coupled only by a joint reward function. The goal in this work is to cooperatively maximize cumulative reward. Several recent approaches for solving such DEC-MDPs involve identifying the set of policies for each agent that are nondominated with respect to the space of potential policies chosen by the other agents [3; 13]. With some adaptation, our NRV algorithm could be used to solve transition independent DEC-MDPs. Likewise the *successive approximation* algorithm for DEC-MDPs [13]—which iteratively finds policies that are nondominated with respect to another agent's policies and admits an anytime error bound—could potentially be adapted to generating policies that are nondominated with respect to reward.

5 Empirical Evaluation

We test the NRV algorithm on small MDPs and compare it to πWitness, an existing method for generating nondominated policies [16]. We test both methods on small, randomly generated IRMDPs with *factored, additive reward functions*. A state $s = \langle x_1, x_2, \ldots, x_7 \rangle$ 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 $\mathcal{R}$ 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 for each reward dimension, and run each algorithm to completion, generating *all* nondominated policies.

Figs. 3 and 4 show the average runtime of each algorithm. While NRV is more efficient than πWitness on small MDPs, the performance gap narrows as reward dimensionality increases. The most striking advantage of NRV is the availability of an error bound $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$ at each iteration. Fig. 5(a) shows that the error $\varepsilon_{\mathbb{V}}$ drops quickly with each added nondominated policy (note the log scale). For example, $\varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$ is reduced to well under 1.0% of its initial value (i.e., when $\tilde{\Gamma} = \emptyset$) after only 500 policies, and to nearly 0.1% after 2000 policies. A small set $\tilde{\Gamma}$ of nondominated policies can be used to quickly approximate minimax regret as discussed above. We note that $\tilde{\Gamma}$ can usually be computed only once (offline), prior to elicitation. Minimax regret, conversely, needs to be computed *repeatedly* and *online*, since it is integral to many elicitation schemes. Thus offline computation of a small $\tilde{\Gamma}$ with small error can greatly enhance online performance. For this reason, the extensive offline computation required for computing nondominated sets is not necessarily problematic. However, we can do even more with NRV when allowing approximation in an online setting.

We now examine how NRV, used in conjunction with constraint generation for fast, approximate solution of minimax regret, works in the context of our online optimization scheme. Recall our online model provides the ability to improve the quality of an approximate solution during elicitation, while maintaining tractability. We demonstrate this potential 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.[3] 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; 15]. Let the *gap* for any (s, a)-pair be

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

[3] More complex queries and methods for MDP reward elicitation are addressed in [17].

CS queries the point (s, a) with the largest *weighted gap* $\mathbf{f}(s, a)\Delta(s, a)$, with weight given by occupancy frequency $\mathbf{f}(s, a)$ in the current solution $\mathbf{f}$ 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 is used to refine $\mathcal{R}$. Elicitation terminates once max regret τ reaches an acceptable level. When a nondominated set $\tilde{\Gamma}$ is used to approximate the minimax optimal policy, we simply require that $MMR(\tilde{\Gamma}, \mathcal{R}) + \varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R}) \leq \tau$ (w.r.t. the current reward polytope $\mathcal{R}$). Online optimization improves solution quality and decreases $\varepsilon_{\mathbb{V}}$, thus admitting faster termination of elicitation.

We examine our online method in a realistic elicitation scenario using a variant of the *COACH* MDP, which models a system that provides cognitive assistance for persons with dementia to enable them to complete activities of daily living [4]. Very roughly, the goal 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 ℓ 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 $\mathcal{S} = \langle T, D, F \rangle$ 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 at the current 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 $r(\langle t, d, f \rangle, a) = r_{goal}(t) + r_{progress}(d = 0) + r_{delay}(d) + r_{prompt}(a)$, where: $r_{goal}(t)$ is a large positive reward when $t = \ell$ for completing the task and is zero when $t < \ell$; $r_{progress}(d = 0)$ is a (small) positive reward for progressing to step t (indicated by d being reset to zero); $r_{delay}(d)$ is a small negative reward for delay in completing a step; and $r_{prompt}(a)$ is the negative cost associated with prompting the person. The precise values of the rewards are not known but must be elicited from a caregiver. We set $\ell = 14$, $k = 6$ and create an IRMDP by setting initial reward bounds to construct $\mathcal{R}$ in a manner similar to the random IRMDPs discussed in the previous section. The resulting IRMDP has size $|\mathcal{S}||\mathcal{A}|$=3012 and reward dimensionality $|\mathcal{R}|$=12. We use the NRV algorithm generate $\tilde{\Gamma}$ with the following criterion in mind. We wish to allow for interactive response times during elicitation, so we choose the size of $\tilde{\Gamma}$ so that $MMR(\tilde{\Gamma}, \mathcal{R})$ takes no more than one second to compute. This results in $\tilde{\Gamma}$ containing less than 5% of all nondominated policies. We further assume that during elicitation there are ten seconds available while waiting for a user response to perform online optimization (pruning and addition) of $\tilde{\Gamma}$. During elicitation we compute minimax regret using: a static set $\tilde{\Gamma}$ with error $\varepsilon_{\mathbb{V}}$; and a dynamic set $\tilde{\Gamma}'$ with decreasing error $\varepsilon'_{\mathbb{V}}$, optimized online by pruning and adding policies during the ten-second period provided by response latency.

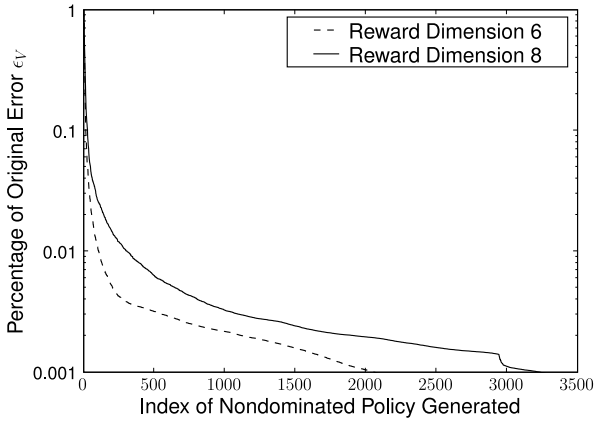

Figure 5: Error $\varepsilon_{\mathbb{V}}$ as function of NRV policies generated (20 random IRMDPs, log scale).

Fig. 6 shows the upper bounds $MMR(\tilde{\Gamma}, \mathcal{R}) + \varepsilon_{\mathbb{V}}(\tilde{\Gamma}, \mathcal{R})$ and $MMR(\tilde{\Gamma}', \mathcal{R}) + \varepsilon_{\mathbb{V}}(\tilde{\Gamma}', \mathcal{R})$ on max regret produced by the static set and online-optimized set, respectively; it is shown as the percentage of the initial upper bound on minimax regret prior to the start of elicitation. We see that online optimization of the nondominated set provides a tremendous benefit in terms of elicitation. First, without online adjustment of $\tilde{\Gamma}$, it is impossible to find the optimal policy: indeed, the static set stalls after roughly 40 queries with minimax regret that is still roughly 18% of the initial regret level. Online online optimization of $\tilde{\Gamma}$ allows discovery of the optimal policy with approximately 60 queries (this is about 5 simple bound queries per reward parameter). Just as importantly, if an approximately optimal policy is desired, the online-optimized $\tilde{\Gamma}$ reduces minimax regret to 20% of its initial levels with only about 12 queries on average, while the static approach requires almost 35 queries. In this example, a small $\tilde{\Gamma}$ with less than 5% of all nondominated policies enables effective reward elicitation, quickly reducing the approximation error to zero if $\tilde{\Gamma}$ is optimized online. This demonstrates the power of our online approach. A very small set of nondominated policies is needed for fast online computation; but a static set of the required size does not admit an approximation of suitable quality. Pruning newly dominated policies during elicitation and adding new policies using NRV allows one to maintain online feasibility while reducing provable max regret to zero, while supporting effective elicitation (both in terms of num-

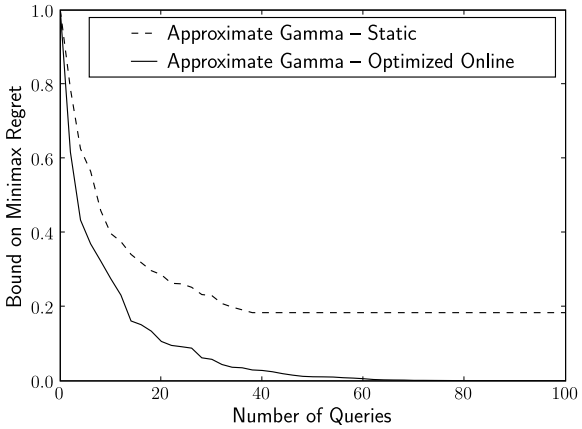

Figure 6: Upper Bound on Minimax Regret (as proportion of initial regret) during Elicitation for the COACH model.

ber of queries and interactive response time).

6 Discussion and Conclusion

We have presented a method for computing approximate, robust solutions to *imprecise-reward* MDPs (IRMDPs) in the context of online reward elicitation. The NRV algorithm generates approximate sets of nondominated policies with provable error bounds, which can be leveraged to efficiently approximate minimax regret using existing constraint generation methods. We also showed how online optimization of the nondominated set, as reward knowledge is refined, allows regret to quickly decrease to zero with only a small fraction of all nondominated policies. Our empirical results demonstrate the value of our online approach. Taken together these results remove a significant computational barrier to online reward elicitation for MDPs.

Future research includes developing more general queries, suited to the sequential nature of MDPs, while remaining intuitive and cognitively tractable to users; for example, the use of additive reward models to support elicitation has recently been explored [17]. Other possibilities include policy or trajectory comparisons, which ask a user to assess their relative preference for two state-action sequences (or distributions over these). Since trajectories often contain extraneous information that is irrelevant to reward, trajectory "summaries" which count reward-bearing events could be used. Other potential queries remain to be explored, providing a richer vocabulary for eliciting MDP reward functions, and easing their adoption by practitioners.

Acknowledgements: Thanks to the reviewers for their helpful suggestions. This research was supported by NSERC.

References

[1] David Avis. lrs: A revised implementation of the reverse search vertex enumeration algorithm. In *Polytopes–Combinatorics and Computation*, pages 177–198. Birkhauser-Verlag, 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] Ralphen Becker, Shlomo Zilberstein, Victor R. Lesser, and Claudia V. Goldman. Solving transition independent decentralized Markov decision processes. *Journal of Artificial Intelligence Research*, 22:423–455, 2004.

[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. *Artifical 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, New York, 1986.

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

[10] 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.

[11] 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.

[12] Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Operations Research*, 53(1):780–798, 2005.

[13] Marek Petrik and Shlomo Zilberstein. A bilinear programming approach for multiagent planning. *Journal of Artificial Intelligence Research*, 35(1):235–274, 2009.

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

[15] Kevin Regan and Craig Boutilier. Regret-based reward elicitation for Markov decision processes. In *Proceedings of the Twenty-fifth Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 454–451, Montreal, 2009.

[16] Kevin Regan and Craig Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *Proceedings of the Twenty-fourth AAAI Conference on Artificial Intelligence (AAAI-10)*, pages 1127–1133, Atlanta, 2010.

[17] Kevin Regan and Craig Boutilier. Eliciting additive reward functions for Markov decision processes. In *Proceedings of the Twenty-second International Joint Conference on Artificial Intelligence (IJCAI-11)*, Barcelona, 2009. To appear.

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

[19] 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.