Eliciting Additive Reward Functions for Markov Decision Processes

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

Abstract

Specifying the reward function of a Markov decision process (MDP) can be demanding, requiring human assessment of the precise quality of, and tradeoffs among, various states and actions. However, reward functions often possess considerable structure which can be leveraged to streamline their specification. We develop new, decision-theoretically sound heuristics for eliciting rewards for factored MDPs whose reward functions exhibit *additive independence*. Since we can often find good policies without complete reward specification, we also develop new (exact and approximate) algorithms for robust optimization of *imprecise-reward MDPs* with such additive reward. Our methods are evaluated in two domains: autonomic computing and assistive technology.

1 Introduction

Markov decision processes (MDPs) require the specification of a large set of model parameters. While model dynamics can be learned by observation of the environment, a reward function usually reflects the subjective preferences of some user; hence, reward specification can be difficult since it requires mapping human preferences over states and actions into numbers. Eliciting *partial* reward specifications can ease this burden. Recently, models have been proposed for *MDPs with imprecise reward (IRMDPs)*, along with methods for computing policies that are robust w.r.t. reward imprecision using minimax regret [9; 10; 12]. Moreover, these methods can be used to guide the elicitation of additional reward information, thereby improving policy quality by reducing reward uncertainty (and lowering max regret) [9]. However, existing models assume a "flat" reward function and do not leverage the multiattribute structure often present is realistic preferences. While significant work exists on modeling and eliciting multiattribute *utility functions* with structure (e.g., additive independence) in one-shot decision scenarios, insights gained from this work have found little application to MDPs.

In this paper, we develop elicitation methods for MDPs that exploit additive independence in multiattribute reward functions. Unlike "flat" utility models, which must grapple with rewards over state spaces of size exponential in the number of attributes, we assume a natural additive decomposition of reward that supports elicitation using much simpler, more natural, and many fewer user queries. Our queries and elicitation techniques draw on the theoretically sound foundations of multiattribute utility theory [5; 6].

We provide two main contributions. First, we develop techniques for computing minimax regret for IRMDPs with additive reward. Our exact algorithm is derived from the approach of [9] for flat reward models, but addresses complications due to parameter interactions in the multiattribute representation. We also propose several approximations for regret computation. Then, after discussing various query types, we develop a simple heuristic elicitation method for IRMDPs. We show that by adapting the *current solution* heuristic to IRMDPs—a method used extensively for utility elicitation in the minimax regret framework [3; 4]—we can effectively solve MDPs optimally or near-optimally with very few queries about reward parameters.

Sec. 2 provides background on MDPs, multiattribute utility, and the robust solution of IRMDPs. Sec. 3 describes reward elicitation heuristics and the types of queries used for eliciting multiattribute reward. In Sec. 4 we outline algorithms for exact and approximate minimax regret computation with additive IRMDPs, and in Sec. 5 we demonstrate the effectiveness of our reward elicitation scheme in two concrete domains, assistive technology and autonomic computing.

2 Background

We focus on infinite horizon MDPs $\langle \mathbf{X}, \mathcal{A}, \{P_{\mathbf{x}a}\}, \gamma, \beta, r\rangle$, with finite state set $\mathbf{X}$, finite action set $\mathcal{A}$, transition distributions $P_{\mathbf{x}a}(\cdot)$, discount factor γ, initial state distribution β, and reward function $r(\mathbf{X}, \mathcal{A})$. Given a (deterministic) policy $\pi : \mathbf{X} \rightarrow \mathcal{A}$, the policy *value function* satisfies:

$$V^{\pi}(\mathbf{x}) = r(\mathbf{x}, \pi(\mathbf{x})) + \gamma \sum_{\mathbf{x}'} P_{\mathbf{x}\pi(\mathbf{x})}(\mathbf{x}') V^{\pi}(\mathbf{x}'). \quad (1)$$

An *optimal policy* π^* maximizes Eq. 1, giving value V^*.

A policy π induces an *occupancy function* f^{π}, where $f^{\pi}(\mathbf{x}, a)$ is the total discounted probability of being in state $\mathbf{x}$ and taking action a, given π. Let $\mathcal{F}$ be the set of feasible occupancy functions (i.e., induced by some policy). Policy π can be recovered from f^{π} and vice-versa [8]; we use the two representations interchangeably, referring to an occupancy function f as if it were its corresponding policy π. Hence, the optimal value function V^* satisfies $\sum_{\mathbf{x}} \beta(\mathbf{x}) V^*(\mathbf{x}) = \sum_{\mathbf{x},a} f^*(\mathbf{x}, a) r(\mathbf{x}, a)$, where $f^* = f^{\pi^*}$ is the optimal occupancy. Let $\mathbf{f}$ be a $|\mathbf{X}| \times |\mathcal{A}|$ matrix with entries $f(\mathbf{x}, a)$, and $\mathbf{r}$ a

$|\mathbf{X}|\times|\mathcal{A}|$ matrix with entries $r(\mathbf{x}, a)$; then $\sum_{\mathbf{x}} \beta(\mathbf{x})V^*(\mathbf{x}) = \mathbf{f}:\mathbf{r}$ (where : is the Frobenius inner product).

We assume that our MDP is *factored* [2], with states defined by a set of *attributes*: $\mathbf{X} = X_1 \times \ldots \times X_n$, where each X_i is an attribute with finite domain. We write $\mathbf{x}[i]$ to denote the value x_i of the ith attribute of $\mathbf{x}$, denote by $\mathbf{x}_{-i}$ the restriction of $\mathbf{x}$ to the attributes excluding X_i, and write $(x_i, \mathbf{x}_{-i})$ to indicate the outcome obtained by conjoining the two. In addition, we assume a user's reward function is *additive independent* [5; 6] over the attribute space, conditional on the choice of action.[1] More precisely, the reward function can be written as, for any state $\mathbf{x} = (x_1, \ldots, x_n)$ and $a \in A$,

$$r(\mathbf{x}, a) = \sum_i r_i(x_i, a) = \sum_i \lambda_i \upsilon_i(x_i, a)$$

where the r_i are *subreward functions* for each attribute, which are themselves expressed using *local utility functions* υ_i and *scaling constants* λ_i s.t. $r_i(x_i, a) = \lambda_i \upsilon_i(x_i, a)$.[2] A reward function is additively decomposable in this way exactly when the user is indifferent between any two distributions (or "lotteries") p, p' over states $\mathbf{x}$ (given a fixed action a) iff p and p' have the same marginals over attributes X_i. Elicitation of additive reward can exploit this structure as discussed below. We use the following notation: $\boldsymbol{\lambda}$ is an n-vector with entries λ_i; and $\boldsymbol{\upsilon}$ an $n \times |\mathbf{X}| \times |\mathcal{A}|$ tensor with entries $\upsilon_i(\mathbf{x}[i], a)$. Thus $\mathbf{r} = \boldsymbol{\lambda} \diamond \boldsymbol{\upsilon} = \sum_i \lambda_i \boldsymbol{\upsilon}_i$ (where $\diamond$ indicates contracted tensor product). We write $\boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{f}$ below to mean $(\boldsymbol{\lambda} \diamond \boldsymbol{\upsilon}):\mathbf{f}$.

An *imprecise reward MDP (IRMDP)* is an MDP with a set $\mathcal{R}$ of feasible reward functions, reflecting uncertainty about $\mathbf{r}$ [9; 12]. In the sequel, $\mathcal{R}$ is represented by a set of constraints on $(\boldsymbol{\lambda}, \boldsymbol{\upsilon})$. These arise naturally in preference assessment (e.g., behavioral observation or elicitation). Since policy value cannot usually be proven without knowing $\mathbf{r}$, we use *max regret* to measure policy quality, and *minimax regret* as our optimization criterion [9; 12]. Define:

$$R(\mathbf{f}, \boldsymbol{\lambda}, \boldsymbol{\upsilon}) = \max_{\mathbf{g}\in\mathcal{F}} \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{g} - \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{f}$$

$$PMR(\mathbf{f}, \mathbf{g}, \mathcal{R}) = \max_{\mathbf{r}\in\mathcal{R}} \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{g} - \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{f}$$

$$MR(\mathbf{f}, \mathcal{R}) = \max_{\boldsymbol{\lambda},\boldsymbol{\upsilon}\in\mathcal{R}} R(\mathbf{f}, \boldsymbol{\lambda}, \boldsymbol{\upsilon}) = \max_{\mathbf{g}\in\mathcal{F}} \max_{\boldsymbol{\lambda},\boldsymbol{\upsilon}\in\mathcal{R}} \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{g} - \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{f} \quad (2)$$

$$MMR(\mathcal{R}) = \min_{\mathbf{f}\in\mathcal{F}} MR(\mathbf{f}, \mathcal{R}) = \min_{\mathbf{f}\in\mathcal{F}} \max_{\mathbf{g}\in\mathcal{F}} \max_{\boldsymbol{\lambda},\boldsymbol{\upsilon}\in\mathcal{R}} \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{g} - \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{f}$$

$R(\mathbf{f}, \boldsymbol{\lambda}, \boldsymbol{\upsilon})$ is the *regret* of policy $\mathbf{f}$ w.r.t. the optimal policy given reward parameters $(\boldsymbol{\lambda}, \boldsymbol{\upsilon})$. $MR(\mathbf{f}, \mathcal{R})$ is the maximum regret of $\mathbf{f}$ under any possible realization of reward (consider an adversary choosing $\mathbf{r}$ from $\mathcal{R}$). $PMR(\mathbf{f}, \mathbf{g}, \mathcal{R})$ is the pairwise max regret of adopting policy $\mathbf{f}$ rather than $\mathbf{g}$. The *minimax optimal policy* $\mathbf{f}^*$ is that with minimum max regret $MR(\mathbf{r}^*, \mathcal{R}) = MMR(\mathcal{R})$. Minimax regret offers *robustness* in the presence of reward uncertainty, and can effectively guide elicitation in one-shot settings [3; 4].

[1] Factored MDPs can also have factored actions, and admit compact specification of dynamics and reward, and efficient solution techniques [2]. Here we exploit only the factored nature of rewards, since our focus is on reward elicitation.

[2] Our notation assumes an explicit dependence on A for ease of exposition, but preferences for some/all attributes will be independent of A in many MDPs. And if action costs are independent of state, we can simply treat A as another attribute.

3 Reward Elicitation

We extend "flat" regret-based reward elicitation for MDPs [9] by incorporating regret-based approaches to *multiattribute* decision problems [3; 4]. Elicitation of a reward function is greatly facilitated by additive structure: we describe some of the natural queries supported by the additive model in this section. We also develop a heuristic query selection strategy guided by minimax regret. Our basic elicitation framework is as follows: given the current constraints defining $\mathcal{R}$, we compute the minimax optimal policy $\mathbf{f}^*$. If $MR(\mathbf{f}^*, \mathcal{R})$ is below some threshold τ, we terminate elicitation with this policy, which is τ-optimal; otherwise, we refine $\mathcal{R}$ by asking the user a reward query, guided by the current solution.

3.1 Query Types

The additive structure of the reward function admits both *local queries* involving only single attributes X_i and *global queries* that involve full state instantiations $\mathbf{x}$. Additive independence allows one to determine the local utility functions υ_i for each X_i independently, and requires global queries only to fix scaling constants λ_i, which calibrate strength of preference across attributes [6; 5].

We first consider local queries. For any fixed action a, let $x^\top_{i,a}$ denote the most preferred value of X_i given a, and $x^\bot_{i,a}$ the least preferred. *Local anchoring* is the process of asking a user to identify these values from the domain of X_i; this is straightforward for a user. With these values in hand, we set $\upsilon_i(x^\top_{i,a}, a) = 1$ and $\upsilon_i(x^\bot_{i,a}, a) = 0$ for convenience. We can then determine $\upsilon_i(x_i, a)$ for any x_i using a *standard gamble query*: the user is asked to identify the probability p with which they would be indifferent between x_i and a *gamble* $\langle x^\top_{i,a}, p, x^\bot_{i,a}\rangle$ (which gives $x^\top_{i,a}$ with probability p, and $x^\bot_{i,a}$ with $1-p$). This determines $\upsilon_i(x_i, a) = p$.

Standard gambles, unfortunately, impose a severe cognitive burden on users because of the precision required, and one which is generally unnecessary: we can often find good or optimal policies with only loose bounds on the $\upsilon_i(x_i, a)$ terms. Instead we use *local bound queries*, which given some *fixed* p, ask a user whether she *prefers* x_i to the gamble $\langle x^\top_{i,a}, p, x^\bot_{i,a}\rangle$. A positive response means $\upsilon_i(x_i, a) > p$, and a negative response that $\upsilon_i(x_i, a) \leq p$.[3] All local utilities υ_i can be *partially elicited* using only local bound queries, using yes/no queries which obviate the need for users to explicitly quantify the strength of their preferences.

Scaling parameters λ_i calibrate strength of preference across attributes; thus, they require global queries, but only of a specific form. *Global anchoring queries* ask the user to specify her most preferred and least preferred state-action pairs $(\mathbf{x}, a)^\top$ and $(\mathbf{x}, a)^\bot$, respectively, and are again easy to assess. We fix a *reference outcome* $\mathbf{x}^0_a$ for $a \in A$.[4] This can be any salient outcome, but for concreteness, we set $\mathbf{x}^0_a[i] = x^\bot_{i,a}$, which yields:

$$r((x^\top_{i,a}, \mathbf{x}^0_{-i}), a) = r_i(x^\top_{i,a}, a) + \sum_{j\neq i} r_j(x^0_j, a) = r_i(x^\top_{i,a}, a) \quad (3)$$

[3] Approximate indifference ("not sure") can also be handled.

[4] The same state can be chosen for each a; and if state rewards are independent of actions, then dependence on a is not needed.

The reward $r((x^{\top}_{i,a}, \mathbf{x}^0_{-i}), a)$ in Eq. 3 can be elicited with a standard gamble asking for the probability p for which the user is indifferent between $((x^{\top}_{i,a}, \mathbf{x}^0_{-i}), a)$ and $\langle (\mathbf{x}, a)^{\top}, p, (\mathbf{x}, a)^{\perp} \rangle$. Since $\upsilon_i(x^{\top}_{i,a}, a) = 1$ and $r_i(x^{\top}_{i,a}, a) = \lambda_i \upsilon_i(x^{\top}_{i,a}, a)$, Eq. 3 implies that λ_i is exactly the reward elicited for outcome $r((x^{\top}_{i,a}, \mathbf{x}^0_{-i}), a)$. As with local queries, rather than asking precise numerical queries, we use *global bound queries* which fix a probability p and ask which of $((x^{\top}_{i,a}, \mathbf{x}^0_{-i}), a)$ and $\langle (\mathbf{x}, a)^{\top}, p, (\mathbf{x}, a)^{\perp} \rangle$ is preferred. Notice global bound queries constrain each λ_i independently. The absence of constraints linking components of $\boldsymbol{\lambda}$ can be leveraged in minimax regret computation (see Sec. 4).

To summarize, we elicit reward parameters by first asking the user to specify $x^{\top}_{i,a}$ and $x^{\perp}_{i,a}$ (local anchoring) for each X_i, a, and $(\mathbf{x}, a)^{\top}$ and $(\mathbf{x}, a)^{\perp}$ (global anchoring). We then use local bound queries to constrain local utilities $\boldsymbol{\upsilon}$ and global bound queries to constrain scaling factors $\boldsymbol{\lambda}$. We now turn to query selection: which queries to ask when.

3.2 Query Selection using Minimax Regret

At each round of elicitation we select a query ϕ, whose response ρ refines our knowledge of $\mathcal{R}$, and ideally reduces minimax regret. To select suitable queries, we adapt the *current solution* (CS) heuristic [3; 9], which uses the solution to the minimax optimization for query selection. Let $(\mathbf{f}^c, \mathbf{g}^c, \boldsymbol{\lambda}^c, \boldsymbol{\upsilon}^c)$ be the solution given current reward polytope $\mathcal{R}$: $\mathbf{f}^c$ is the minimax optimal policy; $\boldsymbol{\lambda}^c, \boldsymbol{\upsilon}^c$ are the reward parameters that maximize regret of $\mathbf{f}^c$; and $\mathbf{g}^c$ is adversarial policy (i.e., optimal for $\boldsymbol{\lambda}^c, \boldsymbol{\upsilon}^c$). CS computes a score $S(\phi)$, for each potential query ϕ, that measures its potential to reduce the max regret of $\mathbf{f}^c$. We develop scoring functions for each query type below. The CS heuristic asks the query with the highest score.[5] We use *marginal occupancy probabilities*, $f_i(x_i, a) = \sum_{\mathbf{x}_{-i}} f((x_i, \mathbf{x}_{-i}), a)$, in defining scores.

Local Bound A local bound query requires selection of an attribute-value, action pair (x_i, a) and local utility bound p. Define the *local utility gap* of (x_i, a) to be:

$$\upsilon\text{-}gap(x_i, a) = \uparrow \upsilon_i(x_i, a) - \downarrow \upsilon_i(x_i, a),$$

where $\uparrow \upsilon_i(x_i, a)$ and $\downarrow \upsilon_i(x_i, a)$ are the maximum and minimum values $\upsilon_i(x_i, a)$ can take in $\mathcal{R}$, when fixing $\upsilon^c_j(x_j, a)$ for all $j \neq i$. We fix the bound p in the local bound query for (x_i, a) at the midpoint of this gap (hence any response narrows the gap by half). To score (x_i, a), notice that

$$PMR(\mathbf{f}, \mathbf{g}, \mathcal{R}) = \max_{\boldsymbol{\lambda}, \boldsymbol{\upsilon}} \sum_i \lambda_i \sum_{x_i} \sum_a gf_{\Delta}(x_i, a) \upsilon_i(x_i, a), \quad (4)$$

where $gf_{\Delta}(x_i, a) = g_i(x_i, a) - f_i(x_i, a)$. Thus, the impact on PMR of tightening the bound on $\upsilon_i(x_i, a)$ is mediated by the difference in policy marginals $g_i(x_i, a) - f_i(x_i, a)$ and the scaling constant λ_i. Any response gives a new constraint on $\upsilon_i(x_i, a)$ that changes PMR by at most

$$S_{\upsilon}(x_i, a) = \lambda^c_i \cdot gf^c_{\Delta}(x_i, a) \cdot \upsilon\text{-}gap(x_i, a)/2.$$

This is the score of a local bound query (x_i, a) (assuming that we fix the query bound p to be the gap midpoint).

[5] The score can be combined with other factors relevant to query selection, e.g., the cognitive effort required to answer a query.

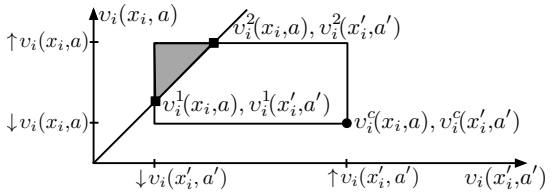

Figure 1: A bisection of the bounding rectangle induced by a comparison query.

Global Bound A global bound query requires selection of an attribute i and a bound p. Similar to local bound queries, define the *scaling gap* λ-*gap*$(i) = \uparrow \lambda_i - \downarrow \lambda_i$, where $\uparrow \lambda_i$ and $\downarrow \lambda_i$ are the max/min values that λ_i can take in $\mathcal{R}$ given other reward parameters fixed by $\boldsymbol{\lambda}^c, \boldsymbol{\upsilon}^c$. We always query the midpoint of this gap, giving a score much like local bounds:

$$S_{\lambda}(i) = \sum_{x_i \in X_i} \sum_{a \in \mathcal{A}} gf^c_{\Delta}(x_i, a). \upsilon^c_i(x_i, a) \lambda\text{-}gap(i)/2$$

Local Comparison *Local comparison queries* ask a user to compare two distinct values of an attribute: is (x_i, a) preferred to (x'_i, a'). We do not use such queries in our experiments (our local domains have a natural preference order), but for completeness, we briefly describe how to score such queries using a method proposed in [4]. Given the query above, we project a bounding hyper-rectangle onto the plane of the two parameters $\upsilon_i(x_i, a), \upsilon_i(x'_i, a')$. The constraint imposed by a query response divides the 2D-rectangle along a 45-degree line (Fig. 1). With respect to PMR, values $\upsilon^c_i(x_i, a)$ and $\upsilon^c_i(x'_i, a')$ may remain feasible, or else are pushed to lie at intersection of the diagonal with the bounding rectangle. Let $\upsilon^1_i(x_i, a), \upsilon^1_i(x'_i, a')$ and $\upsilon^2_i(x_i, a), \upsilon^2_i(x'_i, a')$ be the coordinates of these two points. Let $\upsilon_{\Delta}(x_i, a) = max(|\upsilon^c_i(x_i, a) - \upsilon^1_i(x_i, a)|, |\upsilon^c_i(x_i, a) - \upsilon^2_i(x_i, a)|)$ be the greatest change in value that $\upsilon_i(x_i, a)$ can realize given the query response. We score potential reduction in PMR as:

$$S_l(x_i, a, x'_i, a') = \lambda^c_i \Big[gf^c_{\Delta}(x_i, a) \upsilon_{\Delta}(x_i, a) + gf^c_{\Delta}(x'_i, a') \upsilon_{\Delta}(x'_i, a') \Big].$$

4 Computing Minimax Regret

To compute minimax regret we adapt the constraint generation approach of [10], solving a series of LPs:

$$\begin{aligned} \min_{\mathbf{f}, \delta} \quad & \delta \\ \text{subject to:} \quad & \delta \geq \boldsymbol{\lambda}_i \boldsymbol{\upsilon}_i \mathbf{g}_i - \boldsymbol{\lambda}_i \boldsymbol{\upsilon}_i \mathbf{f}, \quad \forall \langle \mathbf{g}_i, \boldsymbol{\lambda}_i, \boldsymbol{\upsilon}_i \rangle \in \text{GEN} \\ & \mathbf{f} \in \mathcal{F} \end{aligned}$$

Here GEN is a set of constraints corresponding to a subset of possible adversarial choices of policies and rewards. If GEN contains all vertices $\mathbf{r} = (\boldsymbol{\lambda}, \boldsymbol{\upsilon})$ of polytope $\mathcal{R}$ and the corresponding optimal policies $\mathbf{g}^*_{\mathbf{r}}$, this LP computes minimax regret exactly. However, most constraints will not be active at optimality, so iterative constraint generation is used: given solution $\mathbf{f}$ to the relaxed problem with a subset GEN of constraints, we find the most violated constraint, i.e., the $\boldsymbol{\lambda}, \boldsymbol{\upsilon}, \mathbf{g}$ that maximizes regret of $\mathbf{f}$. If no violated constraints exist,

then $\mathbf{f}$ is optimal. Violated constraints are computed by solving $MR(\mathbf{f}, \mathcal{R})$, using a mixed integer program (MIP):

$$\begin{aligned} \underset{\mathbf{Q},\mathbf{V},\mathbf{I},\boldsymbol{\lambda},\boldsymbol{\upsilon}}{\text{maximize}} \quad & \boldsymbol{\beta}\mathbf{V} - \boldsymbol{\lambda}\boldsymbol{\upsilon}\mathbf{f} && (5) \\ \text{subject to:} \quad & \mathbf{Q}_a = \boldsymbol{\lambda}\boldsymbol{\upsilon}_a + \gamma \mathbf{P}_a \cdot \mathbf{V} && \forall\, a \in \mathcal{A} \\ & \mathbf{V} \leq (\mathbf{1} - \mathbf{I_a})\mathbf{M} + \mathbf{Q_a} && \forall\, a \in A \\ & \sum_a \mathbf{I}_a = \mathbf{1} \end{aligned}$$

Here $\mathbf{I}_a$ is an $|S|$-vector of indicator variables denoting whether (adversarial) policy $\mathbf{g}$ takes action a; and $\mathbf{M}$ is a vector of sufficiently large constants (these can be tightened in several ways). This formulation contains the quadratic terms $\boldsymbol{\lambda}\boldsymbol{\upsilon}$. However, when no constraints link scaling constants $\boldsymbol{\lambda}$ (e.g., as with global bound queries), an optimal solution must set λ_i to its maximum $\lambda_i^{\top}$ or minimum $\lambda_i^{\perp}$. In such a case, we can linearize the MIP with the following reformulation. We introduce indicators $\mathbf{J}$, where $J_i = 1$ means $\lambda_i = \lambda_i^{\top}$ and $J_i = 0$ means $\lambda_i = \lambda_i^{\perp}$. Then in the optimal solution we have: $\boldsymbol{\lambda}\boldsymbol{\upsilon} = \mathbf{J}\boldsymbol{\lambda}^{\top}\boldsymbol{\upsilon} - (1 - \mathbf{J})\boldsymbol{\lambda}^{\perp}\boldsymbol{\upsilon}$. The quadratic term $\mathbf{J}\boldsymbol{\upsilon}$ is linearized using a "big-M" formulation in which we introduce a continuous variable $\mathbf{Z}$ which replaces $\mathbf{J}\boldsymbol{\upsilon}$, with constraints $\mathbf{Z} \leq \boldsymbol{\upsilon} + (\mathbf{1} - \mathbf{J})\mathbf{M}'$ and $\mathbf{Z} \leq \mathbf{J}\mathbf{M}'$ (here $\mathbf{M}'$ is a sufficiently large constant). The result is a (linear) MIP with continuous variables $\mathbf{Q}, \mathbf{V}, \mathbf{Z}, \boldsymbol{\lambda}, \boldsymbol{\upsilon}$ and binary variables $\mathbf{I}, \mathbf{J}$.

While constraint generation with this MIP solves minimax regret exactly, exact solutions are not needed to effectively guide query selection [3]: approximations often suffice. Furthermore, the linearization of the quadratic MIP for $MR(\mathbf{f}, \mathcal{R})$ only works when independent upper bounds are available on the λ_i. To this end, we implemented an alternating optimization model that independently optimizes the components of Eq. 2: we repeatedly compute, in turn, an optimal (adversarial) policy $\mathbf{g}$ (fixing $\boldsymbol{\lambda}, \boldsymbol{\upsilon}$), scaling factors (fixing $\mathbf{g}, \boldsymbol{\upsilon}$), and local utility functions $\boldsymbol{\upsilon}$ (fixing $\mathbf{g}, \boldsymbol{\lambda}$). This reduces the optimization for max regret to a sequence of LPs that locally explore solution space, avoiding the potentially expensive solution of the linear MIP; and it is applicable even when the MIP cannot be linearized. Alternating optimization is guaranteed to converge and must return a feasible solution (w.r.t. Eq. 2), and quality can be further improved using random restarts. Finally, when used in constraint generation, it provides a lower bound on minimax regret.

We investigated the performance of exact and approximate (10 random restarts) minimax regret computation on a small set of random MDPs with 2–4 binary attributes. Results in the table below (avg. 30 runs) show that while runtime of the exact formulation grows quickly with reward dimension, approximation runtime and error remain low. This suggests that alternating approximation will provide an effective substitute for the MIP w.r.t. guiding elicitation.

$\mathcal{R}$ dim.		Runtime (secs)		MMR		
$\lvert\boldsymbol{\lambda}\rvert$	$\lvert\boldsymbol{\upsilon}\rvert$	MIP	Alt.	MIP	Alt.	Error
2	4	0.51	0.40	90.66	88.79	1.87
3	6	5.65	0.77	107.94	107.45	0.49
4	8	1744.20	1.09	149.52	149.19	0.33

The alternating model gives only a lower bound on MMR. Upper bounds are also valuable, since they offer a guarantee on max regret at any point during elicitation (e.g., this is useful for termination). To quickly generate upper bounds, we linearize Eq. 2 using the reformulation-linearization technique (RLT) of Sherali [11].[6] We assessed its effectiveness by computing an upper bound on MMR with the MDPs above, with results in the table below. The linearization is, naturally, much more efficient than the MIP, but produces weak upper bounds, that on average are an order of magnitude larger than the exact solution. However, we will see in Sec. 5 that linearization often gives better approximations when: (i) the MDP is structured; and (ii) $\mathcal{R}$ is sufficiently constrained.

$\mathcal{R}$ dim.		Runtime (secs)		MMR		
$\lvert\boldsymbol{\lambda}\rvert$	$\lvert\boldsymbol{\upsilon}\rvert$	MIP	Lin.	MIP	Lin.	Error
2	4	0.51	0.18	90.66	1019.28	928.61
3	6	5.65	0.31	107.94	1564.87	1456.93
4	8	1744.20	0.63	149.52	2236.93	2087.41

5 Experiments

We now describe two domains that are naturally modeled using MDPs with additive reward, and explore the effectiveness of our approach to reward elicitation.

5.1 Assistive Technology

We present a simplified model of the COACH system [1], whose general goal is to guide a patient with dementia through a task with ℓ steps such as hand-washing using verbal or visual cues, while minimizing intrusion. Prompts can be issued at increasing levels of intrusiveness until (at the highest level k) a caregiver is called to assist the person in task completion. This results in action space $\mathcal{A} = \{0, 1, \ldots, k\}$. The state is defined by three variables $\mathcal{S} = \langle T, D, F \rangle$; $T \equiv \{0, 1, \ldots, \ell\}$ is the number of tasks steps successfully completed by the person, $D \equiv \{0, 1, 2, 3, 4, 5+\}$ is the *delay* (time taken during the current step), and $F \equiv \{0, 1, \ldots, k\text{-}1\}$ tracks whether a prompt at a specific level was attempted on the current task step, but 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: $r(t, d, f, a) = r_g(t) + r_d(d) + r_p(a)$, where: $r_g(t)$ is a positive "task completion" reward (non-zero if $t = \ell$ task, zero otherwise); $r_d(d)$ is a negative "delay" penalty; and $r_p(a)$ is a negative "prompting" penalty associated with prompting the person. Typically, $r_p(a = k)$ is a very large negative cost for calling the caregiver (relative to other costs); and the subreward functions $r_d(d)$ and $r_p(a)$ are both assumed to be monotonic (in delay and prompting level, respectively). Each subreward function is the product of a scaling constant and local utility function:

[6] Details of the upper bound formulation are omitted for the sake of brevity and can be found in a forthcoming technical report.

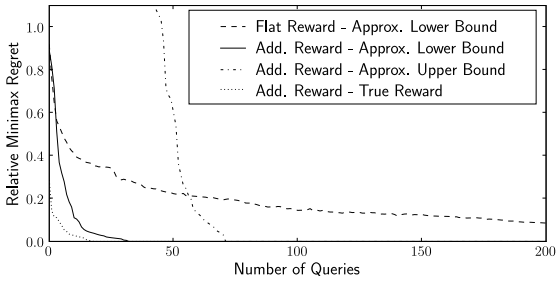

Figure 2: Preference Elicitation in COACH Domain (30 runs)

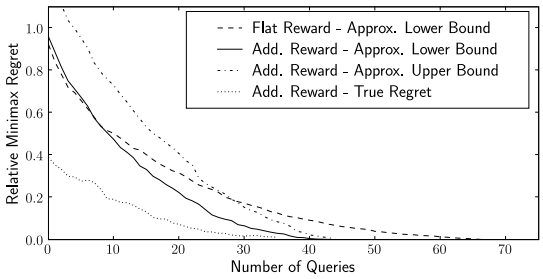

Figure 3: Preference Elicitation in Autonomic Domain (30 runs)

$r_i = \lambda_i v_i$. Attribute F does not occur in the reward function, so requires no elicitation.

Local Anchoring Local anchoring requires no elicitation from the user, since the best/worst values for each attribute T, D, A are given by assumption: penalties for D and A increase monotonically in level, while $t = \ell$ is known to be the preferred value of T (while $v_g(t) = 0$ for all $t \neq \ell$). Since we specify local utility functions on a $[0, 1]$ scale, we use negative scaling constants to ensure that the subreward functions for "penalties" are negative.

Global Anchoring Our domain also permits global anchoring without input from the user. The most preferred state-action pair occurs when the task has been completed with no delay and no prompting: $(\mathbf{x}, a)^\top \equiv (t = \ell, d = 0, a = 0)$. The least preferred pair is when no progress is made beyond the first step despite the most intrusive prompts: $(\mathbf{x}, a)^\perp \equiv (t = 0, d = 5+, a = k)$. We set $r(\mathbf{x}_\top, a_\top) = 1$ and $r(\mathbf{x}_\perp, a_\perp) = -1$. In this case $r(\mathbf{x}^\perp, a^\perp) \neq 0$, and we set reference state $(\mathbf{x}^0, a^0) \equiv (t{=}0, d{=}0, a{=}0\ )$.

Local Bound Local bound queries in this domain are reasonably natural, especially for professional caregivers for whom our potential elicitation application is designed. For instance, consider a query involving v_d, with this structure: $d = 3 \succeq \langle d{=}0, b, d{=}5+\rangle$? For bound $b = 0.6$, we might pose this as: "Would you prefer a certain delay of 3 or a situation in which there is a 60% chance of delay 0 and a 40% chance of delay 5+? (all else being equal)". Local bound queries involving v_p are specified similarly. No local elicitation of v_g is needed: only $v_g(\ell)$ has positive reward.

Global Bound We calibrate the scaling constants $\boldsymbol{\lambda}$ using global bound queries. For instance, consider a query involving λ_d of the form: $(d^\perp, \mathbf{x}^0_{-d}, a^0) \succeq \langle (\mathbf{x}, a)^\top, b, (\mathbf{x}, a)^\perp \rangle$. For $b = 0.2$ this query could be expressed as: "Would you prefer the situation in which at step one there is no delay, and a prompt of level k was issued (calling the caregiver); or would you prefer the following: 20% of the time the goal is reached with no delay and no prompting, but 80% of the time progress is not made beyond the first step and the maximal delay occurs despite the most intrusive prompting?" Given our anchoring assumptions, a positive response constrains $\lambda_d \leq 1 - 2b$.[7] We can bound λ_p in a similar fashion. Note that $\lambda_g = 1$ is implied by our other settings.

Since our local utility functions are monotonic, local comparison queries yield no useful information. The sole challenge of elicitation in this domain is assessing the *strength* of local preference over the domains of attributes D and A, and calibrating the relative importance (via scaling factors) of the local utility functions for each of D, T and A.

Assessment We assessed the effectiveness of elicitation on an instance of this domain with $\ell = 10$ steps and $k = 4$ prompt levels. We simulate elicitation using a reward function reflecting the actual use of the COACH system [1] to generate user responses. The reward function has 11 (unknown) parameters and the size of the state-action space is $|\mathbf{X}||\mathcal{A}| = 960$. This is large enough that exact minimax regret computation is not fast enough to support real-time interaction with a user. Instead we use the alternating approximation to compute minimax regret; this provides us with a lower bound and max regret. We used RLT to compute an upper bound as well. We use the current solution strategy to select local and global bound queries. To determine the advantage of explicitly modeling the additive reward structure in the COACH domain, we also performed elicitation on the same MDP using a flat reward model in the reward function has $|\mathbf{X}||\mathcal{A}| = 960$ dimensions. In this case we use the query selection strategy and the approximate minimax algorithm (which again gives a lower bound on MMR) proposed in [9]. This can be viewed as the flat model variant of our additive approach. Results—shown in Fig. 2—are averaged over 30 runs with randomized initial $\mathcal{R}$.

Comparing lower bounds shows a clear advantage for additive elicitation. But even comparing the *upper* bound for the additive approach with the *lower* bound for the flat model, we see the additive model yields significant leverage, provably reducing minimax regret to zero after 70 queries (the lower bound conjectures MMR=0 after roughly 28 queries). By comparison, after 200 queries 8% of the original regret remains in the *lower bound* produced by the flat reward model. Furthermore, the "flat" queries involve comparisons not over individual attribute values, but of full state-action instantiations, placing additional cognitive burden on the user. *True regret* is also given, showing the actual regret (loss) of the minimax optimal policy in the additive model w.r.t. to the true reward function (of course, the elicitation algorithm does not have access to this). We see that true regret is significantly less than the lower bound on minimax regret. This suggests that (much earlier!) termination using the lower bound on max regret would be beneficial in practice.

5.2 Autonomic Computing

The autonomic computing task [7] involves allocating computing or storage resources to servers as computing demands from clients change over time. We have *K application server*

[7] This constraint is different than if all v_i were positive, but follows directly from $r(d^\perp, \mathbf{x}^0_{-d}, a^0) \geq r(\langle (\mathbf{x}, a)^\top, b, (\mathbf{x}, a)^\perp \rangle)$.

elements $e_1 \dots e_k$, and N *units of resource* which may be assigned to the server elements (plus a "zero resource"). An allocation is specified by $\mathbf{n} = \langle n_1 \dots n_k \rangle$ where $\sum_i^K n_i < N$. Finally there are D *demand levels* at which each server element can operate. A full specification of demand levels is denoted $\mathbf{d} = \langle d_1 \dots d_k \rangle$.

A state is given by the current allocation of resources and current demand levels for each server: $\mathbf{x} = (\mathbf{n}, \mathbf{d})$. Actions are allocations $\mathbf{m} = \langle m_1 \dots m_k \rangle$ of up to N units of resource to the K application servers. 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 i Markov chains $\Pr(d_i' \mid d_i)$, one for the demand at each server element. The reward $r(\mathbf{n}, \mathbf{d}, \mathbf{m}) = u(\mathbf{n}, \mathbf{d}) - c(\mathbf{n}, \mathbf{d}, \mathbf{m})$ is composed of a positive utility $u(\mathbf{n}, \mathbf{d})$ and the negative cost $c(\mathbf{n}, \mathbf{d}, \mathbf{m})$. The cost $c(\mathbf{n}, \mathbf{d}, \mathbf{m})$ is the sum of the costs of taking away one unit of resource from each server element at each time step. We assume that the cost term is known. The utility term $u(\mathbf{n}, \mathbf{d})$ can be factored into local utility functions $v_i(n_i, d_i)$ for each server i. In this setting, utility functions are defined with respect to a common unit (potential revenue), so there is no need for calibration: $\boldsymbol{\lambda} = \mathbf{1}$.

Reward elicitation proceeds by having a server bound its local utility for a given demand and resource level. An example query is: "Given n units of resource and a demand level of d, are the potential earnings generated greater or equal to b". An answer of "yes" imposes the following constraint on the local utility function: $v_i(n_i, d_i) \geq b$.[8]

Assessment We examined the effectiveness of our approach to elicitation of additive reward in this domain using a small MDP with $K = 2, N = 3, D = 3$, yielding a reward function of dimension 24. We used the CS heuristic to select queries and approximated minimax regret using alternating optimization (and RLT to produce an upper bound). As with COACH, we compare to elicitation using a flat version of the reward model in the same MDP (with 90 states/reward parameters) using the same methodology as above. Results are shown in Fig. 3. We see once again that elicitation proceeds quickly and that taking advantage of the additive independence in the reward model yields considerable leverage. The upper bound on max regret generated by the additive model proves that an optimal solution is found more quickly (using 24 fewer queries) than even the lower bound on max regret using the flat model (which cannot guarantee optimality). True regret is, again, less than the lower bound on max regret.

6 Conclusions and Future Work

We have developed an approach to reward elicitation for MDPs whose reward functions exhibit additive independence. As in one-shot multiattribute decision problems, this structure admits more cognitively manageable, yet decision-theoretically sound, queries involving single attributes, and requires very few (and very stylized) global queries. By adopting the minimax regret framework, we further reduce elicitation effort by using simple comparison and bound queries, and allowing elicitation to terminate with approximately optimal solutions. By adapting algorithms for minimax regret computation and regret-based heuristics to query selection, we have demonstrated more effective elicitation performance than can be achieved with unstructured reward.

A number of interesting directions remain. Though our approximations address the computational difficulty of minimax regret to an extent, further research is needed (e.g., by using a precomputed set of *nondominated policies* [10; 12], incorporating solution techniques for factored MDPs, etc.). Our additive assumption, while somewhat restrictive, can be relaxed to allow *generalized additive independence (GAI)* [5]—techniques for elicitation and computing minimax regret in GAI models [4] should be readily adaptable to MDPs. More interesting is the investigation of addition types of queries that exploit the sequential nature of the decision problem and may more quickly rule out certain policies as optimal (e.g., querying preferences over trajectories or attribute occurrence frequencies induced by a policy).

Acknowledgments Thanks to the reviewers for their helpful comments. This research was supported by NSERC.

References

[1] J. Boger, P. Poupart, J. Hoey, C. Boutilier, G. Fernie, and A. Mihailidis. A decision-theoretic approach to task assistance for persons with dementia. *IJCAI-05*, pp.1293–1299, Edinburgh, 2005.

[2] C. Boutilier, T. Dean, and S. Hanks. Decision theoretic planning: Structural assumptions and computational leverage. *JAIR*, 11:1–94, 1999.

[3] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Art. Intel.*, 170:686–713, 2006.

[4] D. Braziunas and C. Boutilier. Minimax regret-based elicitation of generalized additive utilities. *UAI-07*, pp.25–32, Vancouver, 2007.

[5] P. C. Fishburn. Interdependence and additivity in multivariate, unidimensional expected utility theory. *Int. Econ. Rev.*, 8:335–342, 1967.

[6] R. L. Keeney and H. Raiffa. *Decisions with Multiple Objectives: Preferences and Value Trade-offs*. Wiley, NY, 1976.

[7] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer*, 36:41–52, 2003.

[8] M. L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley, NY, 1994.

[9] K. Regan and C. Boutilier. Regret-based reward elicitation for Markov decision processes. *UAI-09*, pp.454–451, Montreal, 2009.

[10] K. Regan and C. Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. *AAAI-10*, pp.1127–1133, Atlanta, 2010.

[11] H. Sherali and A. Alameddine. A new reformulation-linearization technique for bilinear programming problems. *J. Global Optim.*, 2:379–410, 1992.

[12] H. Xu and S. Mannor. Parametric regret in uncertain Markov decision processes. *CDC-09*, pp.3606–3613, Shanghai, 2009.

[8] Generally, server utility has no closed form: utility for a *specific* allocation is bounded using a combination of simulation and numerical optimization; hence elicitation is in fact costly.