Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence

Interactive Value Iteration for Markov Decision Processes with Unknown Rewards

Paul Weng
UPMC, LIP6
France
paul.weng@lip6.fr

Bruno Zanuttini
University of Caen
France
bruno.zanuttini@unicaen.fr

Abstract

To tackle the potentially hard task of defining the reward function in a Markov Decision Process, we propose a new approach, based on Value Iteration, which interweaves the elicitation and optimization phases. We assume that rewards whose numeric values are unknown can only be ordered, and that a tutor is present to help comparing sequences of rewards. We first show how the set of possible reward functions for a given preference relation can be represented as a polytope. Then our algorithm, called Interactive Value Iteration, searches for an optimal policy while refining its knowledge about the possible reward functions, by querying a tutor when necessary. We prove that the number of queries needed before finding an optimal policy is upper-bounded by a polynomial in the size of the problem, and we present experimental results which demonstrate that our approach is efficient in practice.

1 Introduction

Sequential decision-making is the general process of deciding what action to take in some state, so as to optimise a criterion in the long term. In the setting of Markov Decision Processes (MDPs), actions have stochastic effects on the environment, yielding numeric rewards and optimal decisions maximize an expected sum of rewards. If an agent has to make decisions in this setting, the reward function has to be specified by the user (or more generally by the environment). The user may have to do so when writing a model, as in the context of decision support, or when giving the agent some feedback about its actions, as in the setting or reinforcement learning. It is well-known however that the optimal policy for an MDP may be very sensitive to the values actually chosen for the reward function, and that eliciting the correct values (in that they indeed represent the user's preferences) is a very difficult problem. We consider here a variant of the framework, namely *ordinal reward MDPs* [Weng, 2011], in which rewards are taken from a given, ordinal scale, which is cognitively much simpler for the user to specify. The specification of optimal policies is then induced by a preference relation on sequences (or multisets) of such rewards.

As an example, consider an autonomous vacuum cleaner which is taught what to do by a user. In the real-valued setting, the user needs to specify a real value, say 50, for cleaning the living-room, one for cleaning the kitchen (say 30), and one for cleaning the bedroom (say 20). From these values it results that cleaning the living-room and the bedroom yields a reward of 70, and hence is a better trajectory to follow than cleaning twice the kitchen, which is worth 60. However, assigning a 40 reward to the kitchen would change this preference. Contrastingly, in an ordinal setting, only the relative order between rewards (which is the same in both cases above) is initially specified, for instance: cleaning the living-room is better than cleaning the kitchen, which is better than cleaning the bedroom. Since this is in general not enough to determine an optimal policy, this qualitative scale is completed with preferences on sequences or on multisets of rewards, for instance: cleaning the bathroom twice and the bedroom once is better than cleaning the kitchen and the bedroom.

In this paper, we address the problem of computing an optimal policy for an MDP with such ordinal information about the reward function. We assume that the only information initially available is the ordinal reward for each state-action pair. So as to gather enough information, we assume that there is a tutor (*e.g.*, the user, or any kind of oracle) who may answer requests for comparing two sequences of rewards. The objective is to compute an optimal policy while asking as few queries as possible.

Our main contribution is an interactive version of the well-known *value iteration* scheme, in which the tutor is questioned only when necessary. We show that this algorithm computes an (approximate) optimal policy using only a finite number of queries, which is moreover polynomial in the size of the MDP. This could not be taken for granted since in general, an infinite number of queries may be needed for eliciting an arbitrary utility function. Finally, we report on experiments which show that the number of queries issued in pratice is actually quite small.

2 Related Work

The problem of computing optimal policies for MDPs with rewards not strictly defined by real values has been mainly addressed in the setting of *robust optimisation* [Regan and Boutilier, 2011b]. In a series of papers [Regan and Boutilier, 2009a; 2009b; 2010; 2011a; 2011b], the authors consider *im-*

precise reward MDPs (IRMDPs), for which only a set of candidate reward functions is known. They show how to compute policies which optimize minmax regret with respect to all candidate functions, and discuss how this criterion can be used to generate informative queries to ask the user about the actual reward function. Iteratively issuing such queries is shown to allow convergence to the optimal policy for this function [Regan and Boutilier, 2011b]. Nevertheless, our work departs from these on several aspects:

- we address the more direct problem of computing an optimal policy, without trying to compute robust policies of any kind before having acquired enough information; this weaker requirement allows our strategy to ask fewer queries in the end,
- tackling this more direct problem makes the task computationally easier, in particular, we avoid the NP-complete problem of computing the optimal policy under the criterion of minimax regret [Xu and Mannor, 2009],
- we do not assume that the user has an actual reward function in mind, but rather that she has a preference over multisets of (ordinal) rewards; doing so we require less cognitive effort from the user, and in particular we do not use *bound queries* of the form "is the utility for taking action a in state s greater than b?" (for some $b \in \mathbb{N}$).

In the present paper we use results from [Weng, 2011], in particular an axiomatization of preferences over trajectories for ordinal rewards. Algorithms for computing optimal policies of ordinal reward MDPs are also given there, but assuming that the whole preference relation (on multiset of rewards) is given as an input.

The problem of computing optimal decisions with respect to imprecise utility functions has also been addressed in the setting of one-stage (non-sequential) decision making, precisely for recommendation tasks [Viappiani and Boutilier, 2011]. In rough terms, it has been shown that the optimal action (in this case, a set of recommendations) is also a (myopically) maximally informative query to ask the user. However, the approach does not translate to the sequential setting which we consider here.

Finally, related problems have been addressed in the setting of reinforcement learning (RL). In *inverse RL* [Ng and Russell, 2000], the agent observes a policy and its goal is to estimate a reward function according to which this policy is optimal. Hence the reward function is somehow implicitely and qualitatively specified *via* an optimal policy. More recently, and closer to our setting, a formalization of *preference-based RL* has been proposed [Fürnkranz *et al.*, 2012]. There it is assumed that the user (or the environment) can compare trajectories in a qualitative fashion, and the agent builds on such feedbacks for learning a near-optimal policy. The key difference between this work and ours are that

- we consider an offline setting, i.e., the queries the agent can issue are not constrained by the process dynamics,
- the preference relation which we use on multisets of rewards is less general than the one used there.

Both points make our setting more specific, but again they allow us to design a more efficient strategy.

3 Background

3.1 Markov Decision Process

A *Markov Decision Process* (MDP) [Puterman, 1994] is defined by $\mathcal{M} = (S, A, p, r, \gamma)$ where S is a finite set of states, A is a finite set of actions, $p\colon S \times A \to \mathcal{P}(S)$ is a transition function with $\mathcal{P}(S)$ being the set of probability distributions over states, $r\colon S \times A \to \mathbb{R}$ is a reward function and $\gamma \in [0, 1[$ is a discount factor.

A (stationary, deterministic) *policy* $\pi : S \to A$ associates an action to each state. Such a policy is valuated by a *value function* $v^\pi : S \to \mathrm{R}$ defined as follows:

$$v^\pi(s) = r(s, \pi(s)) + \gamma \sum_{s' \in S} p(s, \pi(s), s') v^\pi(s') \qquad (1)$$

Then a preference relation is defined over policies as follows:

$$\pi \gtrsim \pi' \Leftrightarrow \forall s, v^\pi(s) \geq v^{\pi'}(s)$$

A solution to an MDP is a policy, called *optimal policy*, that ranks the highest with respect to $\gtrsim$. Such a policy can be found by solving *Bellman equations*.

$$v^*(s) = \max_{a \in A} r(s, a) + \gamma \sum_{s' \in S} p(s, a, s') v^*(s') \qquad (2)$$

As can be seen, the preference relation $\gtrsim$ over policies is directly induced by the reward function r.

3.2 Ordinal Reward MDP

In this work, we do not assume that numerical rewards are given, however, we assume that the rewards can be ordered. We now present how such a setting can be handled.

As introduced in [Weng, 2011], an *Ordinal Reward MDP* (ORMDP) is an MDP $(S, A, p, \hat{r}, \gamma)$ where the reward function $\hat{r}\colon S \times A \to E$ takes its values in a qualitative, totally ordered scale $E = \{r_1 < r_2 \ldots < r_k\}$.

As shown in [Weng, 2011], one can reformulate an ORMDP as a Vector Reward MDP (VMDP): $(S, A, p, \overline{r}, \gamma)$ where $\overline{r}(s, a)$ is the vector in $\mathbb{R}^k$ whose i-th component is 1 for $\hat{r}(s, a) = r_i$, and 0 on the other components. Like in standard MDPs, in such a VMDP we can define the value function $\overline{v}^\pi$ of a policy π by:

$$\overline{v}^\pi(s) = \overline{r}(s, \pi(s)) + \gamma \sum_{s' \in S} p(s, \pi(s), s') \overline{v}^\pi(s') \qquad (3)$$

where sums and products over vectors are componentwise.

This equation amounts to counting the number of ordinal rewards obtained by applying a policy. Therefore, a value function in a state can be interpreted as a multiset or bag of elements of E.

Now comparing policies boils down to comparing vectors. In the following, the preference relation over vectors will be denoted $\gtrsim$ and vectors will be denoted $N = (N_1, \ldots, N_k)$.

If E were a numerical scale, we would have:

Proposition 1 $v^\pi(s) = \sum_{i=1}^k \overline{v}_i^\pi(s) r_i$.

A natural dominance relation $\succsim_D$ can be defined between any two vectors $N, N' \in \mathbb{R}^k$:

$$N \succsim_D N' \Leftrightarrow \forall i = 1, \ldots, k, \sum_{j=1}^{i} N_j \geq \sum_{j=1}^{i} N'_j \qquad (4)$$

This relation has a natural interpretation. It states that for any reward r_i, the number of rewards better than r_i is higher in N than in N'. This dominance is the first-order stochastic dominance [Shaked and Shanthikumar, 1994] expressed in our settting. It can also be viewed as Pareto dominance over transformed vectors $L(N) = \big(N_1, N_1 + N_2, \ldots, \sum_{j=1}^{k} N_j\big)$. Although $\succsim_D$ is a partial relation and may not be very discriminating, we will use it for eliminating vectors corresponding to policies that are dominated for every reward function.

It is shown in [Weng, 2011] that under natural axioms about the preference relation $\succsim$ over vectors, there always exists a numerical function $\rho : E \to \mathbb{R}$ representing $\succsim$: $\forall N, N', N \succsim N' \iff \sum_{i=1}^{k} N_i \rho(r_i) \geq \sum_{i=1}^{k} N'_i \rho(r_i)$. Hence in the following we will assume that the optimal policies for the ORMDP to be solved are defined by such a function ρ. Precisely, to simplify the presentation we will see the target ORMDP simply as an MDP $\mathcal{M} = (S, A, p, r, \gamma)$ with $r = \rho \circ \hat{r}$. Nevertheless, keep in mind that the user does *not* know the function ρ (or r), and is only able to compare vectors and rewards in a qualitative manner.

4 Admissible Reward Functions

In this section, we present the structure of the set of all reward functions representing the same preference relation over policies, in a given MDP $\mathcal{M} = (S, A, p, r, \gamma)$, as that induced by r. This result will be exploited in our algorithm.

We first introduce two definitions. A reward function $f : S \times A \to \mathrm{R}$ is called *admissible* for $\mathcal{M}$ if the preference relation over policies induced by f coincides with that by r. A reward function f is called *weakly admissible* for $\mathcal{M}$ if optimal policies for f coincides with those for r. We present below the structure of the set of (weakly) admissible reward functions for a given MDP.

For a given MDP, $\mathcal{R}$ denotes the set of all admissible reward functions. One can show that $\mathcal{R}$ is a polyhedral cone [Boyd and Vandenberghe, 2004].

Proposition 2 *For any two r and r' in $\mathcal{R}$, for any two positive reals α, β, the reward function defined by $\alpha r + \beta r'$ is also in $\mathcal{R}$ (i.e., $\mathcal{R}$ is a cone). More specifically, $\mathcal{R}$ is a polyhedral cone generated by a finite number of reward functions.*

Proof Denote V_r^π the value function of a policy π with respect to a reward function r. Let r and r' both represent the preference relation $\succsim$ over policies and let π, π' be two policies such that $\pi \succsim \pi'$. Equation 1 for π yields in vectorial form $\alpha V_r^\pi + \beta V_{r'}^\pi = \alpha r + \beta r' + \gamma T^\pi(\alpha V_r^\pi + \beta V_{r'}^\pi)$, showing that $V_{\alpha r + \beta r'}^\pi = \alpha V_r^\pi + \beta V_{r'}^\pi$ is the value function of π for reward function $\alpha r + \beta r'$. As $V_r^\pi \geq V_r^{\pi'}$ and $V_{r'}^\pi \geq V_{r'}^{\pi'}$, a positive linear combination of them yields $V_{\alpha r + \beta r'}^\pi \geq V_{\alpha r + \beta r'}^{\pi'}$.

For the second part of the proposition, write all the inequalities (with r as the unknown) induced by $\succsim$. $\mathcal{R}$ is defined as an intersection of half-spaces, showing that $\mathcal{R}$ is polyhedral. As the number of inequalities is finite, $\mathcal{R}$ is generated by a finite number of points. □

For a given MDP, $\mathcal{R}^*$ denotes the set of all weakly admissible reward functions. Similarly to Prop. 2, one gets:

Proposition 3 *$\mathcal{R}^*$ is a polyhedral cone generated by a finite number of reward functions. Moreover, $\mathcal{R} \subseteq \mathcal{R}^*$.*

As shown in [Weng, 2011], positive affine transformations of rewards do not change preferences over policies. Using this observation, one can set $r_1 = 0$ and $r_k = 1$ without loss of generality. In the remaining of the paper, we will enforce these two constraints. Then the last two propositions imply that the set of (weakly) admissible reward functions respecting these two constraints is a polytope.

While elicitation procedures would try to recover $\mathcal{R}$, we focus on $\mathcal{R}^*$, as we are only interested in determining optimal policies. Doing so we are able to gather enough information for determining an optimal policy while asking far less queries than would be needed for recovering $\mathcal{R}$.

5 Interactive Value Iteration

We first present the general idea of our algorithm, called *Interactive Value Iteration*. Section 5.1 explains the algorithm in details. Section 5.2 provides an analysis and a discussion.

In order to find an (approximate) optimal policy for an ORMDP with an initially unknown preference relation over vectors, we propose a variant of value iteration where the agent may ask a tutor which of two reward sequences it prefers. An example query is "$r_1 + r_3 \geq 2r_2$?", meaning "is receiving r_1 and r_3 at least as good as receiving twice r_2?" As the tutor answers queries, the set of candidate, weakly admissible reward functions shrinks. Specifically, given the representation of $\succsim$ and E by a reward function r, a query is interpreted as an inequality of the form $\sum_{i=1}^{k} \alpha_i r_i \geq 0$. The answer to such a query indicates whether $\sum_{i=1}^{k} \alpha_i r_i \geq 0$ or $\sum_{i=1}^{k} (-\alpha_i) r_i \geq 0$ holds according to the relation $\succsim$.

At the beginning of the process, the agent only knows the order over rewards, e.g. $r_1 < r_2 < \ldots < r_k$. As underlined above, one can always assume that $r_1 = 0$ and $r_k = 1$. This initial knowledge can be represented as a polytope $\mathcal{K}^{(0)}$ defined by the following inequalities $\forall i = 1, \ldots, k-1, r_{i+1} - r_i > 0$. To avoid strict inequalities, we assume that we know a positive lower bound η on the difference between two consecutive rewards, i.e., $0 < \eta \leq \min_{i=1,\ldots,k-1} \big(r_{i+1} - r_i\big)$. The polytope $\mathcal{K}^{(0)}$ can be seen as a set of inequalities. Let r be the column vector $(r_1, \ldots, r_k)^T \in \mathbb{R}^k$ and $\mathbf{1}_{k-1}$ be the column vector $(1, \ldots, 1)^T \in \mathbb{R}^{k-1}$. Let $K^{(0)}$ be a $(k-1) \times k$ real matrix with $\forall i = 1, \ldots, k-1$, $K^{(0)}_{i,i} = -1$, $K^{(0)}_{i,i+1} = 1$ and null everywhere else, i.e.,

$$K^{(0)} = \begin{bmatrix} -1 & 1 & & & \\ & -1 & 1 & & \\ & & & \ldots & \\ & & & -1 & 1 \end{bmatrix}$$

Then the polytope $\mathcal{K}^{(0)}$ is defined by:

$$K^{(0)} \times r \geq \eta \, \mathbf{1}_{k-1} \qquad (5)$$

Algorithm 1 Interactive Value Iteration$(S, A, p, \hat{r}, \gamma, E, \epsilon)$

$t \leftarrow 0$
compute $\overline{r}$ from $\hat{r}$
$\mathcal{K} \leftarrow$ Init(E)
$\forall s \in S, \overline{v}_0(s) \leftarrow (0, \ldots, 0)$
repeat
  $t \leftarrow t + 1$
  for $s \in S$ **do**
    $best \leftarrow (0, \ldots, 0)$
    for $a \in A$ **do**
      $\overline{v} \leftarrow \overline{r}(s, a) + \gamma \sum_{s'} p(s, a, s') \overline{v}_{t-1}(s')$
      $(best, \mathcal{K}) \leftarrow$ getBest$(best, \overline{v}, \mathcal{K})$
    $\overline{v}_t(s) \leftarrow best$
until $\|\overline{v}_t - \overline{v}_{t-1}\| < \epsilon$
return $\overline{v}_t$

Assume the current knowledge about weakly admissible reward functions is represented by $\mathcal{K}^{(t-1)}$. When an answer is given to a query, yielding an inequality $\sum_{i=1}^{k} \alpha_i r_i \geq 0$, the new knowledge is obtained by intersecting the half-space defined by that inequality and the current polytope $\mathcal{K}^{(t-1)}$ represented by matrix $K^{(t-1)}$. This can simply be represented by adding the row vector $(\alpha_1, \ldots, \alpha_n)$ to matrix $K^{(t-1)}$. Hence the updated knowledge is represented by the following inequalities, $\mathcal{C}(K^{(t-1)})$:

$$K^{(t)} \times r = \begin{bmatrix} K^{(t-1)} \\ \alpha_1 \ldots \alpha_n \end{bmatrix} \times r \geq \eta \begin{bmatrix} \mathbf{1}_{k-1} \\ \mathbf{0}_{t-k+1} \end{bmatrix} \quad (6)$$

where $\mathbf{0}_{t-k+1}$ is the column vector $(0, \ldots, 0) \in \mathbb{R}^{t-k+1}$.

5.1 IVI Algorithm

Using the previous ideas, we propose Algorithm 1, which we call *Interactive Value Iteration* (IVI)[1]. It uses five functions: $Init$, $getBest$ (Algo. 2), $ParetoDominates$, $KDominates$ and $query$ (Algo. 3), that we explain now.

Function $Init(E)$ returns the set of inequalities (5), defining polytope $\mathcal{K}^{(0)}$. Function $ParetoDominates(\overline{v}, \overline{v}')$ returns true if $\overline{v}$ dominates $\overline{v}'$ with respect to $\succsim_D$ as defined in (4). Function $KDominates(\overline{v}, \overline{v}', \mathcal{K})$ returns true if the current knowledge, represented by $\mathcal{K}$, about weakly admissible reward functions implies that $\overline{v}$ is preferred to $\overline{v}'$. This can be checked by solving the following linear program:

$$\begin{cases} y = \min . \quad (\overline{v} - \overline{v}').r \\ \text{s.t.} \quad \mathcal{C}(\mathcal{K}) \end{cases} \quad (7)$$

A nonnegative optimal value for the objective function implies that for any possible reward function, $\overline{v}$ is preferred to $\overline{v}'$. Function $getBest(\overline{v}, \overline{v}', \mathcal{K})$ returns the best of the two vectors $\overline{v}, \overline{v}'$. It first tries to compare them according to $\succsim_D$ (by calling $ParetoDominates$), which is the least computationally costly. If it cannot, it tries using knowledge $\mathcal{K}$ (by calling $KDominates$) for deciding which vector is the best.

[1]For simplicity, IVI is stated as returning an (approximate) optimal vector value function. It could be slightly adapted to return the associated (approximate) optimal policy.

Algorithm 2 getBest$(\overline{v}, \overline{v}', \mathcal{K})$

if ParetoDominates$(\overline{v}, \overline{v}')$ **then**
  return $\langle \overline{v}, \mathcal{K} \rangle$
if ParetoDominates$(\overline{v}', \overline{v})$ **then**
  return $\langle \overline{v}', \mathcal{K} \rangle$
if KDominates$(\overline{v}, \overline{v}', \mathcal{K})$ **then**
  return $\langle \overline{v}, \mathcal{K} \rangle$
if KDominates$(\overline{v}', \overline{v}, \mathcal{K})$ **then**
  return $\langle \overline{v}', \mathcal{K} \rangle$
$\langle best, \mathcal{K} \rangle \leftarrow$ query$(\overline{v}, \overline{v}', \mathcal{K})$
return $\langle best, \mathcal{K} \rangle$

Algorithm 3 query$(\overline{v}, \overline{v}', \mathcal{K})$

build query q for "$\overline{v} \succsim \overline{v}'$"
if *answer to q from tutor is yes* ***then***
  return $\langle \overline{v}, \mathcal{K} \cup \{(\overline{v} - \overline{v}').r \geq 0\} \rangle$
else
  return $\langle \overline{v}', \mathcal{K} \cup \{(\overline{v}' - \overline{v}).r \geq 0\} \rangle$

Finally, it resorts to a query if it cannot decide with its current information. Function $query(\overline{v}, \overline{v}', \mathcal{K})$ queries the tutor to know whether $\overline{v}$ is preferred to $\overline{v}'$ or not. The information obtained from the answer is added to $\mathcal{K}$.

5.2 Analysis of IVI

Numbers of queries The query step is a preference elicitation phase, which is well-studied in decision theory and considered a hard problem. It is known that in the general case, one may need an infinite number of queries for finding an adequate preference representation [Krantz *et al.*, 1971]. However, in the ORMDP setting, we can prove that a finite number of queries is sufficient for IVI to find an optimal solution; in fact, even a polynomial number (in the number of states n, the number of actions m, and $\frac{1}{1-\gamma}$).

Standard value iteration normally yields an ϵ-optimal policy. When ϵ is set low enough (depending on γ and the number of bits needed to represent the numeric values of the problem), the ϵ-optimal policy is guaranteed to be optimal. It can then be shown that an optimal policy can be computed in a polynomial (in n, m and $\frac{1}{1-\gamma}$) number of steps [Littman *et al.*, 1995]. Let $T_{\max}(n, m, \gamma)$ be such a polynomial upperbound on the number of iterations of value iteration for an MDP $(S, A, p, \hat{r})$. Based on this result, we can prove that IVI only needs at most a polynomial number of queries for finding an optimal policy.

Theorem 1 *For an ORMDP $\mathcal{M} = (S, A, p, \hat{r}, \gamma)$, the number of queries needed for IVI to converge is upperbounded by $n(m-1)T_{\max}(n, m, \gamma)$.*

Proof We just need to check that for one iteration of IVI, the number of queries is upperbounded by $n(m-1)$. For a given state s, in the worst case, $ParetoDominates$ and $KDominates$ do not remove any vector and $m-1$ queries must be asked for finding the best vector/action in s. □

In the context of preference elicitation, such a bound is in fact not so good. One does not want the tutor to answer so

many queries. However, the obtained bound is attained in the worst case, when the current knowledge $\mathcal{K}$ never helps. In practice, this should not happen. This is indeed suggested by our experiments, which show that very few queries are indeed needed in general. Besides, IVI could be extended so that not every query is asked. In that case, $\overline{v}_t(s)$ would store a set of nondominated vectors and IVI could be modified in a similar fashion as value iteration for multiobjective MDPs [White, 1982]. An interesting research direction is to find a compromise between the number of queries asked and the size of the $\overline{v}_t(s)$'s. We leave this extension for future work.

Complexity of Queries Query q in Algorithm 3 can be build such that it is presented in the simplest form possible. Assume we were to compare $\overline{v} = (\overline{v}_1, \ldots, \overline{v}_k)$ and $\overline{v}' = (\overline{v}'_1, \ldots, \overline{v}'_k)$. As they are discounted occurence numbers, they are all positive. Let c be the componentwise $\min$ of the two vectors, i.e., $\forall i = 1, \ldots, k, c_i = \min(\overline{v}_i, \overline{v}'_i)$. Then, comparing $\overline{v}$ and $\overline{v}'$ is equivalent to comparing $(\overline{v} - c)$ and $(\overline{v}' - c)$, which is a simpler query as k zeroes have been introduced. Assuming that all parameters in the MDP are rational, each component of $\overline{v}$ and $\overline{v}'$ can be written as an irreducible fraction. Let D be the least common multiple of the denominators of those fractions. The query may be simplified even more by comparing the integer vectors $D(\overline{v} - c)$ and $D(\overline{v}' - c)$ (if the components are not too large).

Besides, when a query q requires to compare too many rewards at the same time, such as "do you prefer getting twice r_4, once r_3 and twice r_1, or once r_4, twice r_3 and twice r_2?", it would be interesting to investigate the possibility of breaking down q into several simpler queries, with less rewards to compare. In the example, we may ask for instance first whether the user prefers $2r_4 + r_3 + 2r_1$ to $r_4 + 4r_3$. In case the answer is affirmative, we already know the answer to the initial query (because of $r_3 > r_2$), and otherwise we can refine the query. We leave this for a future deeper investigation.

Unknown Order over Rewards Our approach can easily be adapted to the case where even the order over rewards is initially only partially known, or not known at all. In the latter case, the inequalities defining $\mathcal{K}^{(0)}$ would be $\forall i = 1, \ldots, k$, $0 \leq r_i \leq 1$. In the former case, starting from the previous inequalities, one would only need to add all inequalities encoding the known partial order. In IVI, queries would be issued for comparing two rewards r_i, r_j that cannot be ordered yet, only if necessary. Note that if such a query is asked, a special treatment is needed. For instance, an answer stating $r_i > r_j$ translates to $r_i - r_j \geq \eta$ instead of $r_i - r_j \geq 0$.

Finally, the approach could also be extended to the case where the number of ordinal rewards are not known in advance. We leave this extension for future work.

Size of $\mathcal{K}$ We finish this section on a side note. Adding an inequality to $\mathcal{K}$ in Function $query$ may render some previous inequalities redundant. One can test whether an inequality $\alpha^T \times r \geq d$ is redundant for a system of inequalities $K \times r \geq b$ by solving the following linear program:

$$\min . \alpha^T \times r \quad \text{s.t.} K \times r \geq b \tag{8}$$

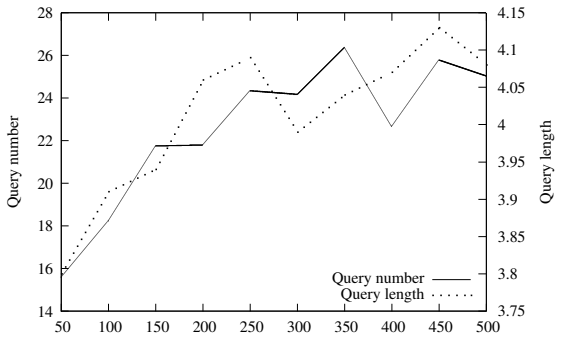

Figure 1: Results as a function of n with $m = 5$, $k = 10$.

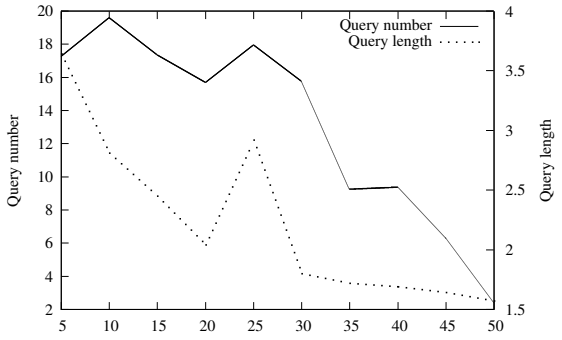

Figure 2: Results as a function of m with $n = 50$, $k = 10$.

The inequality is redundant if and only if the resulting optimal value is not lower than d. Each previous inequality of $\mathcal{K}$ may be tested for redundancy so as to keep the size of the representation of the agent's knowledge minimal. This however comes at the cost of solving $|\mathcal{K}|$ LPs each time an inequality is added, where $|\mathcal{K}|$ is the number of inequalities in $\mathcal{K}$.

6 Experimental Results

We tested our approach on three different domains: random MDPs, Autonomic Computing and Coach domains. Algorithm IVI was coded in R using Gurobi 5.0 as an LP solver and all experiments were run on a PC (Intel Core 2 CPU 2.66Ghz) with 4GB of RAM. The discount factor γ was set to 0.95 and the Bellman error ϵ was set to $1e-3$. The numeric results are averaged over 20 runs.

Random MDPs We first tried IVI on random MDPs with no special structure. A random MDP is characterized by parameters n, m, k denoting the number of states, actions, and steps in the reward scale. Each pair (s, a) is assumed to have $\log_2(n)$ possible successors. Besides, we made some experiments on "grid world" MDPs, but we do not present the results obtained as they are similar to those with random MDPs. We computed two measures to assess our method: the number of queries before convergence of IVI and the query length, which is the number of different ordinal rewards in a multiset intervening in a query. We provide those two measures as a function of the number of states (Fig. 1), the number of actions (Fig. 2) and the size of the ordinal scale E (Fig. 3).

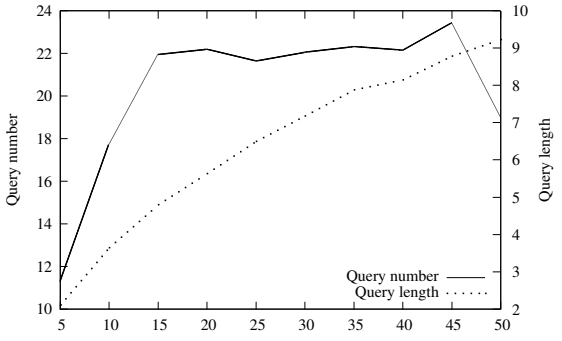

Figure 3: Results as a function of k with $n = 50$, $m = 5$

The figures clearly show that though our theoretical bound on the number of queries is large, in practice this number remains very small and scales nicely with respect to all three parameters. For instance, Figure 1 shows that the mean number is less than 30 for 500 states, 5 actions and 10 steps on the reward scale. Importantly also, the size of queries, which to some extent measures the cognitive effort required from the user, remains acceptable, and again scales nicely with the size of the scale (Figure 3). For instance, with 15 steps, which already allows for a very fine-grained specification of preferences, the mean length is 5, meaning that queries look like "Would you prefer getting $3r_1 + 2r_3 + r_{10}$ or $5r_4 + r_9$?". Anyway, in future work we will consider means of further reducing the complexity of queries, as discussed in Section 5.

The next two problems are structured MDPs. However, we translated them to "flat" MDPs and did not exploit the structure (see the conclusion).

Autonomic Computing We tried our method on the *Autonomic Computing* domain [Boutilier *et al.*, 2003] in order to compare our approach with that proposed in [Regan and Boutilier, 2009b]. We briefly recall this domain (see [Regan and Boutilier, 2009b] for more details). In this problem, we are given $k\kappa$ application server elements on which N available resource units are to be assigned. A feasible allocation is an integral vector $\mathbf{n} = (n_1, \ldots, n_\kappa) \in [0, N]^\kappa$ with $\sum_{i=1}^{k} n_i \leq N$. The clients' demand is modeled as an integral vector $\mathbf{d} = (d_1, \ldots, d_\kappa)$ representing κ levels of demands in $\{1, \ldots, D\}$. A state of the MDP is a vector $(\mathbf{n}, \mathbf{d})$ defining the current allocation and clients' demand. An action is a new allocation $\mathbf{m} = (m_1, \ldots, m_k)$. Rewards are defined by $r(\mathbf{n}, \mathbf{d}, \mathbf{m}) = u(\mathbf{n}, \mathbf{d}) - c(\mathbf{n}, \mathbf{d}, \mathbf{m})$ where $u(\mathbf{n}, \mathbf{d})$ is a sum of non-decreasing utility functions $u(n_i, d_i)$ and $c(\mathbf{n}, \mathbf{d}, \mathbf{m})$ is a sum of the costs for removing a resource unit from a server. An action deterministically sets the next allocation, while uncertainty about demands is stochastic and exogenous.

[Regan and Boutilier, 2009b] tried their method on instances where $k = 2$, $N = 3$ and $D = 3$, defining MDPs with 90 states and 10 actions. Their method needs 200 queries before minmax regret vanishes. On similar instances, IVI asks about 100 queries on average, with length less than 12.

Coach We also tackled the *Coach* domain [Boger *et al.*, 2006] so as to compare IVI to the method proposed in [Regan and Boutilier, 2011b], to which we refer the reader for a more detailed presentation. In this problem, the agent provides assistance to a person with dementia accomplishing a daily-life activity (e.g. handwashing) which is decomposed into $T = \{0, 1, \ldots, l\}$ steps. An action $a \in A = \{0, 1, \ldots, m\}$ in the MDP describes the level of intrusiveness of the prompt the agent provides, i.e., 0 represents no prompt, $m - 1$ represents the most intrusive prompt and m means a caregiver has been called. A state in the MDP is described as (t, d, f) where $t \in T$ is the current step the person is at, $d \in D = \{0, 1, \ldots, 4, 5+\}$ is the delay (time stayed in the current step) and $f \in A$ is the previous prompt. At each step $t < l$, the probability that the person succeeds and moves to the next step $t + 1$ is increasing with a, while it is decreasing with d. Rewards are defined by $r(t, d, f, a) = r_{goal}(t) + r_{progress}(d) + r_{delay}(d) + r_{prompt}(a)$ where $r_{goal}(t)$ is a large reward if the final step is reached and 0 otherwise, $r_{progress}(d)$ is a small reward when making progress (*i.e.*, $d = 0$) and 0 otherwise, $r_{delay}(d)$ is an increasing cost function and $r_{prompt}(a)$ is an increasing cost function whose values are greater than those of r_{delay}.

We ran IVI on instances of the same size ($l = 14$ and $m = 6$) as in [Regan and Boutilier, 2011b]. Our approach requires on average 46.5 queries before IVI converges and the queries have an average length of less than 6. This is slightly better than the method in [Regan and Boutilier, 2011b] which needs around 60 queries for finding an approximate solution.

7 Conclusion

We have presented an approach for computing optimal policies for an MDP in which rewards are specified on an ordinal scale, and a user may be queried about her qualitative preferences among multisets of such rewards. Our approach interleaves a Value Iteration scheme and queries, which are asked only if needed. We have shown that the number of queries needed to identify the optimal policy is finite, despite this is not the case in general for eliciting an arbitrary utility function. Besides, we have shown that this number is polynomial in the size of the MDP. This bound is rather loose, but our experiments show that few queries are needed in practice.

Our results together with other approaches [Weng, 2011; Regan and Boutilier, 2011b; Fürnkranz *et al.*, 2012] show the feasibility of decision-making with only qualitative information (or feedback) about the rewards. As a short-term future work, we plan to extend our approach to other algorithms, especially in the family of policy iteration and Q-learning. We also leave a more precise analysis of the number of queries needed for future work. Finally, we plan to extend our approach to structured MDPs [Boutilier *et al.*, 1999]. We believe that it will naturally take advantage of the structure.

Our ultimate goal is to propose an approach to decision-making, especially in a reinforcement learning setting, in which the feedbacks required from the user are as natural and cognitively simple as possible. The present work takes a first step in that direction by using purely qualitative feedbacks.

Acknowledgments

Funded by the French National Research Agency under grant ANR-10-BLAN-0215.

References

[Boger *et al.*, 2006] J. Boger, J. Hoey, P. Poupart, C. Boutilier, G. Fernie, and A. 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.

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

[Boutilier *et al.*, 2003] C. Boutilier, R. Das, J.O. Kephart, G. Tesauro, and W.E. Walsh. Cooperative negotiation in autonomic systems using incremental utility elicitation. In *UAI*, 2003.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge university press, 2004.

[Fürnkranz *et al.*, 2012] J. Fürnkranz, E. Hüllermeier, W. Cheng, and S.H. Park. Preference-based reinforcement learning: A formal framework and a policy iteration algorithm. *Machine Learning*, 89(1):123–156, 2012.

[Krantz *et al.*, 1971] D.H. Krantz, R.D. Luce, P. Suppes, and A. Tversky. *Foundations of measurement*, volume Additive and Polynomial Representations. Academic Press, 1971.

[Littman *et al.*, 1995] M. L. Littman, T. L. Dean, and L. P. Kaelbling. On the complexity of solving markov decision problems. In *UAI*, pages 394–402, 1995.

[Ng and Russell, 2000] A.Y. Ng and S. Russell. Algorithms for inverse reinforcement learning. In *ICML*, 2000.

[Puterman, 1994] M.L. Puterman. *Markov decision processes: discrete stochastic dynamic programming*. Wiley, 1994.

[Regan and Boutilier, 2009a] K. Regan and C. Boutilier. Online feature elicitation in interactive optimization. In *ICML*, 2009.

[Regan and Boutilier, 2009b] K. Regan and C. Boutilier. Regret based reward elicitation for Markov decision processes. In *UAI*, pages 444–451, 2009.

[Regan and Boutilier, 2010] K. Regan and C. Boutilier. Robust policy computation in reward-uncertain MDPs using nondominated policies. In *AAAI*, 2010.

[Regan and Boutilier, 2011a] K. Regan and C. Boutilier. Eliciting additive reward functions for Markov decision processes. In *IJCAI*, 2011.

[Regan and Boutilier, 2011b] K. Regan and C. Boutilier. Robust online optimization of reward-uncertain MDPs. In *IJCAI*, 2011.

[Shaked and Shanthikumar, 1994] M. Shaked and J.G. Shanthikumar. *Stochastic Orders and Their Applications*. Academic press, 1994.

[Viappiani and Boutilier, 2011] P. Viappiani and C. Boutilier. Recommendation sets and choice queries: There is no exploration/exploitation tradeoff! In *AAAI*, 2011.

[Weng, 2011] P. Weng. Markov decision processes with ordinal rewards: Reference point-based preferences. In *ICAPS*, volume 21, pages 282–289, 2011.

[White, 1982] D.J. White. Multi-objective infinite-horizon discounted Markov decision processes. *J. Math. Anal. Appls.*, 89:639–647, 1982.

[Xu and Mannor, 2009] H. Xu and S. Mannor. Parametric regret in uncertain Markov decision processes. In *IEEE Conference on Decision and Control*, 2009.