A Geometric Approach to Find Nondominated Policies to Imprecise Reward MDPs [1]

Author(s):

Valdinei Freire da Silva

Anna Helena Reali Costa

[1]This work was supported by Fapesp Project LogProb, grant 2008/03995-5, São Paulo, Brazil.

A Geometric Approach to Find Nondominated Policies to Imprecise Reward MDPs[⋆]

Valdinei Freire da Silva and Anna Helena Reali Costa

Universidade de São Paulo, São Paulo, Brazil
valdinei.freire@gmail.com, anna.reali@poli.usp.br

Abstract. Markov Decision Processes (MDPs) provide a mathematical framework for modelling decision-making of agents acting in stochastic environments, in which transitions probabilities model the environment dynamics and a reward function evaluates the agent's behaviour. Lately, however, special attention has been brought to the difficulty of modelling precisely the reward function, which has motivated research on MDP with imprecisely specified reward. Some of these works exploit the use of nondominated policies, which are optimal policies for some instantiation of the imprecise reward function. An algorithm that calculates nondominated policies is πWitness, and nondominated policies are used to take decision under the minimax regret evaluation. An interesting matter would be defining a small subset of nondominated policies so that the minimax regret can be calculated faster, but accurately. We modified πWitness to do so. We also present the πHull algorithm to calculate nondominated policies adopting a geometric approach. Under the assumption that reward functions are linearly defined on a set of features, we show empirically that πHull can be faster than our modified version of πWitness.

Keywords: Imprecise Reward MDP, Minimax Regret, Preference Elicitation.

1 Introduction

Markov Decision Processes (MDPs) can be seen as a core to sequential decision problems with nondeterminism [2]. In an MDP transitions among states are seen as markovian and evaluation is done through a reward function. Many decision problems can be modelled by an MDP with imprecise knowledge. This imprecision can be stated as partial observability regarding states [7], intervals of probability transitions [12] or a set of potential reward functions [13].

Scenarios where reward functions are imprecise are quite common in a preference elicitation process [4,6]. Preference elicitation algorithms guide a process of sequential queries to a user so as to elicit his/her preference based on his/her answer. Even if the process is guide to improve the knowledge about the user's preference, after a finite sequential of queries an imprecise representation must be used [3,9]. User's preference may be model for example in a reward function [10].

[⋆] This work was conducted under project LogProb (FAPESP proc. 2008/03995-5). Valdinei F. Silva thanks FAPESP (proc. 09/14650-1) and Anna H. R. Costa thanks CNPq (proc. 305512/2008-0).

In this paper we tackle the problem of Imprecise Reward MDPs (IRMDP). If a decision must be taken in an IRMDP, optimal action must be properly defined. The minimax regret approach considers relatively the worst case decision, providing a balanced decision. First, when evaluating a decision, it is considered an adversary that chooses the reward function that minimises the value of the decision. The regret step compares the actual chosen decision within the best adversarial option for each feasible reward function and the decision with less regret in the worst case is considered.

Since optimisation must go through reward functions and adversarial policies, it cannot be solved by one linear program. Efficient solutions consider an iterative process in which decisions are chosen using a linear programming and adversarial choices are made through a mixed integer programming [13,10] or linear programs based on nondominated policies [11]. In the latter, the πWitness algorithm was used to generate nondominated policies. Nondominated policies are optimal policies for some instantiation of the imprecise reward function.

Even if the set of nondominated policies is much smaller when compared to the set of deterministic policies, the cardinality of the set to be considered is still a burden to deal with. Although πWitness is able to calculate the set of nondominated policies, nothing is said about choosing efficiently a small subset of nondominated policies. Using a small subset helps to choose properly and fast the best minimax regret decision.

We propose the πHull algorithm to calculate an efficient small subset of nondominated policies. In order to compare it with πWitness, we also modify πWitness to generate a small subset of nondominated policies.

The paper is organised as follows. Section 2 introduces theory and notation used and section 3 describes our modified version of πWitness. Section 4 presents our main contribution, the πHull algorithm. Finally, experiments are given in section 5 and section 6 presents our conclusions.

2 Theoretic Background

In this section we summarise some significant theoretic background regarding MDPs and Imprecise Reward MDPs.

2.1 Markov Decision Process

Markov Decision Process (MDP) is a common formulation regarding optimal decisions. An MDP presents two main features [2]: *(i)* an underlying dynamic system, and *(ii)* an evaluation function that is additive in time. An MDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, P_a(\cdot), \gamma, \beta, r(\cdot) \rangle$, where:

- $\mathcal{A}$ is a finite set of possible actions a;
- $\mathcal{S}$ is a finite set of process states s;
- $P_a : \mathcal{S} \times \mathcal{S} \to [0,1]$ models a stationary discrete-time stochastic process on state s_t such that $P_a(i,j) = P(s_{t+1} = j | s_t = i, a_t = a)$;
- $\gamma \in [0,1)$ is a discount factor;
- $\beta : \mathcal{S} \to [0,1]$ is an initial probability distribution;
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function; and

– the system dynamics is such that $s_0 \in \mathcal{S}$ is drawn from distribution $\beta(s)$ and if the process is in the state i at time t and action a is chosen, then: the next state j is chosen according to the transition probabilities $P_a(i,j)$ and a payoff with expectation $r(i,a)$ is incurred.

A solution for an MDP consists in a policy $\pi : \mathcal{S} \times \mathcal{A} \to [0,1]$, i.e., at any time t, $\pi(s,a)$ indicates the probability of executing action $a_t = a$ after observing state $s_t = s$. A policy π is evaluated according to its expected accumulated discounted reward, i.e., $V(\pi) = E_{s_0 \sim \beta, a_t \sim \pi}[\sum_{t=0}^{\infty} \gamma^t r_t]$.

A policy π induces a discounted occupancy frequency $f^\pi(s,a)$ for each pair (s,a), or in vector notation $\mathbf{f}^\pi$, i.e., the accumulated expected occurrences of each pair discounted in time. Let $\mathcal{F}$ be the set of valid occupancy frequencies for a given MDP, then for any $\mathbf{f} \in \mathcal{F}$ it is valid:

$$[\mathbb{1} - \gamma \mathbb{P}]^\top \mathbf{f} = \boldsymbol{\beta},$$

where $\mathbb{P}$ is a $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|$ matrix indicating $P_a(s,s')$, $\mathbb{1}$ is a $|\mathcal{S}||\mathcal{A}| \times |\mathcal{S}|$ matrix with one in self-transitions, i.e., $\mathbb{1}((s,a),s) = 1$ for all $a \in \mathcal{A}$, and $\boldsymbol{\beta}$ is a $|\mathcal{S}|$ vector indicating $\beta(s)$.

Consider the reward function with a vector notation, i.e., $\mathbf{r}$. In this case, the value of a policy π is given by $V(\pi) = \mathbf{f}^\pi \cdot \mathbf{r}$. An optimal occupancy frequency $\mathbf{f}^*$ can be found by solving:

$$\begin{aligned} &\min_{\mathbf{f}} \mathbf{f} \cdot \mathbf{r} \\ \text{subject to: } &[\mathbb{1} - \gamma \mathbb{P}]^\top \cdot \mathbf{f} - \boldsymbol{\beta} = 0 \cdot \\ &\mathbf{f} \geq 0 \end{aligned} \qquad (1)$$

Given an optimal occupancy frequency $\mathbf{f}^*$ the optimal policy can be promptly defined. For any $s \in \mathcal{S}$ and $a \in \mathcal{A}$ an optimal policy π^* is defined by:

$$\pi^*(s,a) = \begin{cases} \dfrac{\mathbf{f}(s,a)}{\sum_{a' \in \mathcal{A}} \mathbf{f}(s,a)} & , \text{ if } \sum_{a' \in \mathcal{A}} \mathbf{f}(s,a) > 0 \\ \dfrac{1}{|\mathcal{A}|} & , \text{ if } \sum_{a' \in \mathcal{A}} \mathbf{f}(s,a) = 0 \end{cases} . \qquad (2)$$

2.2 Imprecise Reward MDP

An imprecise reward MDP (IRMDP) consists in an MDP in which the reward function is not precisely defined [13,10,11]. This can occur due to a preference elicitation process, lack of knowledge regarding to evaluation of policies or reward functions comprising the preferences of a group of people. An IRMDP is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, P_a(\cdot), \gamma, \beta, \mathcal{R} \rangle$, where $\mathcal{R}$ is a set of feasible reward functions. We consider that a reward function is imprecisely determined by $n_{\mathcal{R}}$ strict linear constraints. Given a $|S||A| \times n_{\mathcal{R}}$ matrix $\mathbf{A}$ and a $1 \times n_{\mathcal{R}}$ matrix $\mathbf{b}$, the set of feasible reward functions is defined by $\mathcal{R} = \{\mathbf{r} | \mathbf{A}\mathbf{r} \leq \mathbf{b}\}$.

In an IRMDP, it is also necessary to define how to evaluate decisions. The minimax regret evaluation makes a trade-off between the best and the worst cases. Consider a feasible occupancy frequency $\mathbf{f} \in \mathcal{F}$. One can calculate the regret $\texttt{Regret}(\mathbf{f}, \mathbf{r})$ of taking such the occupancy frequency $\mathbf{f}$ relative to a reward function $\mathbf{r}$ as the difference between $\mathbf{f}$ and the optimal occupancy frequency under $\mathbf{r}$, i.e.,

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

Since any reward function $\mathbf{r}$ can be chosen from $\mathcal{R}$, the maximum regret $\texttt{MR}(\mathbf{f},\mathcal{R})$ evaluates the occupancy frequency $\mathbf{f}$, i.e.,

$$\texttt{MR}(\mathbf{f},\mathcal{R}) = \max_{\mathbf{r}\in\mathcal{R}} \texttt{Regret}(\mathbf{f},\mathbf{r}).$$

Then, the best policy should minimise the maximum regret criterium:

$$\texttt{MMR}(\mathcal{R}) = \min_{\mathbf{f}\in\mathcal{F}} \texttt{MR}(\mathbf{f},\mathcal{R}).$$

In order to calculate $\texttt{MMR}(\mathcal{R})$ efficiently, some works use nondominated policies, i.e., policies that are optimal for some feasible reward functions [11]. Formally, a policy $\mathbf{f}$ is nondominated with respect to $\mathcal{R}$ iff

$$\exists\mathbf{r}\in\mathcal{R} \text{ such that } \mathbf{f}\cdot\mathbf{r} \geq \mathbf{f}'\cdot\mathbf{r},\ \forall\mathbf{f}'\neq\mathbf{f}\in\mathcal{F}. \tag{3}$$

2.3 Reward Functions Based on Features

Reward functions are usually defined on a small set of k features, where $k \ll |\mathcal{S}||\mathcal{A}|$. Features represent semantic events of interest, such as obtaining or consuming a resource.

Consider a feature function that maps each pair (s,a) to a vector of k observed features, i.e., $\boldsymbol{\phi}:\mathcal{S}\times\mathcal{A}\rightarrow\mathbb{R}^k$. Then, the reward function $r(s,a)$ is considered to be a linear combination of such observed features:

$$r(s,a) = \mathbf{w}\cdot\boldsymbol{\phi}(s,a),$$

where $\mathbf{w}$ is a weight vector.

The occupancy frequency vector $\mathbf{f}$ can be reduced to an expected feature vector $\boldsymbol{\mu} = \mathbf{f}^\top\boldsymbol{\phi}$, where $\boldsymbol{\phi}$ is considered to be the matrix $|\mathcal{S}||\mathcal{A}|\times k$ with rows denoting $\boldsymbol{\phi}(s,a)$. Note that we can talk interchangeably about a policy π, its occupancy frequency vector $\mathbf{f}$, or its expected feature vector $\boldsymbol{\mu}$. A stationary policy π is uniquely defined by an occupancy frequency vector $\mathbf{f}$, and the evaluation of a policy π depends only on its expected feature vector $\boldsymbol{\mu}$, i.e., $V(\pi) = \mathbf{w}\cdot\boldsymbol{\mu}$.

Although it makes easy the definition of reward functions, the use of attributes still requires the assignment of scalar values, which is not an easy task. Unless the user is familiar with some concepts of decision theory, it is not a natural task understanding this assignment of precise numerical values. However, defining a weight vector imprecisely is much easier than defining a reward function in the full state-action space.

3 The πWitness Algorithm

Regan and Boutilier [11] present the The πWitness algorithm for identifying nondominated policies. We describe below a modified version of it in order to choose properly a small subset of nondominated policies.

3.1 Witness Reward Functions

Given any occupancy frequency $\mathbf{f} \in \mathcal{F}$, define its corresponding policy $\pi_{\mathbf{f}}$ (see equation 2). Let $\mathbf{f}[s]$ be the occupancy frequency obtained by executing policy $\pi_{\mathbf{f}}$ with determministic initial state $s_0 = s$, i.e., the frequency occupancy of policy $\pi_{\mathbf{f}}$ with an initial state distribution $\beta'(s) = 1$.

Let $\mathbf{f}^{s:a}$ be the occupancy frequency in the case that if s is the initial state then action a is executed and policy $\pi_{\mathbf{f}}$ is followed thereafter, i.e.,

$$\mathbf{f}^{s:a} = \beta(s) \left[e^{s:a} + \gamma \sum_{s' \in \mathcal{S}} \mathbf{f}[s'] \mathbf{P}_a(s, s') \right] + \sum_{s' \neq s} \mathbf{f}[s'] \beta(s')$$

where $e^{s:a}$ is an $|S||A|$ vector with 1 in position (s, a) and zeros elsewhere[1].

The occupancy frequency $\mathbf{f}^{s:a}$ can be used to find nondominated policies in two steps: *(i)* choose arbitrarily $\mathbf{r}_{init}$ and find an optimal occupancy frequency $\mathbf{f}_{\mathbf{r}_{init}}$ with respect to $\mathbf{r}_{init}$, keeping each optimal occupancy frequency in a set Γ; *(ii)* for each occupancy frequency $\mathbf{f} \in \Gamma$, *(iia)* find $s \in \mathcal{S}$, $a \in \mathcal{A}$ and $\mathbf{r} \in \mathcal{R}$ such that:

$$\mathbf{f}^{s:a} \cdot \mathbf{r} > \mathbf{f}' \cdot \mathbf{r} \text{ for all } \mathbf{f}' \in \Gamma, \tag{4}$$

(iib) calculate the respective optimal occupancy frequency $\mathbf{f}_{\mathbf{r}}$, and add it into Γ. The algorithm stops when no reward function can be found such that equation 4 is true. The reward function in equation 4 is a witness that there exists at least another nondominated policy to be defined.

3.2 Efficient Small Subset of Policies

Despite of the set of nondominated policies being much smaller than the set of all deterministic policies, it can still be very large, making costly the calculation of the minimax regret. It is interesting to find a small set of policies that approximates efficiently the set of nondominated policies. By efficient we mean that a occupancy frequency f_Γ chosen within a small subset Γ is as better as the exact minimax regret decision, i.e.,

[1] We changed the original formula [11]:

$$\mathbf{f}^{s:a} = \beta(s) \left[e^{s:a} + \gamma \sum_{s' \in \mathcal{S}} \mathbf{f}[s'] \mathbf{P}_a(s, s') \right] + (1 - \beta(s))\mathbf{f}.$$

Note that such equation implicitly considers a new β' distribution, defined by

$$\beta'(x) = \begin{cases} \beta(x) + \beta(x)(1 - \beta(s)) & \text{, if } x = s \\ \beta(x)(1 - \beta(s)) & \text{, otherwise} \end{cases}.$$

In this case the occupancy frequency $\mathbf{f}^{a:s}$ has the meaning of executing action a when starting in state s with probability

$$\beta(s) + \pi_{\mathbf{f}}(s, a)(1 - \beta(s)).$$

$\texttt{MR}(f_\Gamma, \mathcal{R}) - \texttt{MMR}(\mathcal{R}) \simeq 0$. Consider a witness $\mathbf{r}^w$ and its respective optimal occupancy frequency $\mathbf{f}^w$. The difference

$$\Delta(\mathbf{f}^w, \Gamma) = \max_{\mathbf{r}\in\mathcal{R}} \min_{\mathbf{f}'\in\Gamma} \left[\mathbf{f}^w \cdot \mathbf{r} - \mathbf{f}' \cdot \mathbf{r}\right]$$

can be used to define the gain when adding $\mathbf{f}^w$ to the set Γ. If a small subset of nondominated policies is desired, $\Delta(\mathbf{f}^w, \Gamma)$ may indicate a priority on which policies are added to Γ. Instead of adding to Γ every occupancy frequency $\mathbf{f}^w$ related to nondominated policies, it is necessary to choose carefully among witnesses $\mathbf{f}^w$, and to add only the witness that maximizes $\Delta(\mathbf{f}^w, \Gamma)$.

3.3 The πWitnessBound Algorithm

Table 1 summarises the πWitnessBound algorithm, our modified version of πWitness. It chooses N_Γ nondominated policies. The $\texttt{findBest}(\mathbf{r})$ function solves an MDP with reward function $\mathbf{r}$ (see equation 1). Instead of finding all feasible occupancy frequency vectors in $\mathcal{F}$, the $\texttt{findWitnessReward}(\mathbf{f}^{s:a}, \Gamma))$ tries to find a witness $\mathbf{r}^w$ to $\mathbf{f}^{s:a}$ which guarantees equation 3 within the set Γ.

Algorithm 1. The πWitnessBound algorithm

```
Input: IRMDP, $N_\Gamma$
$\mathbf{r} \leftarrow$ some arbitrary $\mathbf{r} \in \mathcal{R}$
$\mathbf{f} \leftarrow \texttt{findBest}(\mathbf{r})$
$\Gamma \leftarrow \emptyset$
$\Gamma_{\texttt{agenda}} \leftarrow \{\mathbf{f}\}$
while $|\Gamma| < N_\Gamma$ do
    $\mathbf{f} \leftarrow$ best item in $\Gamma_{\texttt{agenda}}$ regarding to $\Delta(\mathbf{f}, \Gamma)$
    add $\mathbf{f}$ to $\Gamma$
    foreach $s, a$ do
        $\mathbf{r}^w \leftarrow \texttt{findWitnessReward}(\mathbf{f}^{s:a}, \Gamma_{\texttt{agenda}})$
        while witness found do
            $\mathbf{f}^{best} \leftarrow \texttt{findBest}(\mathbf{r}^w)$
            add $\mathbf{f}^{best}$ to $\Gamma_{\texttt{agenda}}$
            $\mathbf{r}^w \leftarrow \texttt{findWitnessReward}(\mathbf{f}^{s:a}, \Gamma_{\texttt{agenda}})$
Output: $\Gamma$
```

It is worth to notice that each iteration of πWitness takes at least $|S||A|$ calls to $\texttt{findWitnessReward}(\cdot)$, and if it succeeds $\texttt{findBest}(\cdot)$ is also called. The number of policies in agenda can also increase fast, increasing the burden of calls to $\texttt{findWitnessReward}(\cdot)$. In the next section we consider the hypothesis of the reward function being defined with a small set of features, and we take this into account to define a new algorithm with better run-time performance.

4 A Geometric Approach to Find Nondominated Policies

The problem of finding nondominated policies is similar to the problem of finding the convex hull of a set of points. Here the set of points are occupancy frequency vectors. We consider reward functions defined in terms of features, thus we can work in a space of reduced dimensions. Our algorithm is similar to the Quickhull algorithm [1], but the set of points is not known *a priori*.

4.1 Space of Feature Vector

Even with no information about the reward functions, but considering that they are described by k features, we can analyse the corresponding IRMDP in the feature vector space. The advantage of such analysis is that a conventional metric space can be considered. This is possible because the expected vector of features regarding to a policy accumulates all the necessary knowledge about transitions in an MDP.

In this section we show through two theorems that if we take the set of all feasible expected feature vectors $M = \{\boldsymbol{\mu}^{\pi} | \pi \in \Pi\}$ and define its convex hull $\mathcal{M} = \mathrm{co}(M)$[2], then the vertices $\mathcal{V}$ of the polytope $\mathcal{M}$ represents the expected feature vector of special deterministic policies. Such special policies are the nondominated policies under imprecise reward functions where the set $\mathcal{R}$ is free of constraints.

Theorem 1. *Let Π be the set of stochastic policies and let $M = \{\boldsymbol{\mu}^{\pi} | \pi \in \Pi\}$ be the set of all expected feature vectors defined by Π. The convex hull of M determines a polytope $\mathcal{M} = \mathrm{co}(M)$, where $\mathrm{co}(\cdot)$ stands for the convex hull operator. Let $\mathcal{V}$ be the set of vertices of the polytope $\mathcal{M}$, then for any vertex $\boldsymbol{\mu} \in \mathcal{V}$ there exists a weight vector $\mathbf{w}_{\boldsymbol{\mu}}$ such that:*

$$\mathbf{w}_{\boldsymbol{\mu}} \cdot \boldsymbol{\mu} > \mathbf{w}_{\boldsymbol{\mu}} \cdot \boldsymbol{\mu}' \textit{ for any } \boldsymbol{\mu}' \neq \boldsymbol{\mu} \in M.$$

Proof. If $\boldsymbol{\mu}$ is a vertex of the polytope $\mathcal{M}$, there exists a hyperplane H such that $H \cap \mathcal{M} = \{\boldsymbol{\mu}\}$. The hyperplane H divides the feature vector space in two half-spaces $\mathcal{X}_{H,1}$ and $\mathcal{X}_{H,2}$. Let us define the set $M' = M - \{\boldsymbol{\mu}\}$. Because the set resulting from the intersection of H and $\mathcal{M}$ has cardinality one, all feature vectors in M' are in the same half-space, i.e., either $M' \subset \mathcal{X}_{H,1}$ or $M' \subset \mathcal{X}_{H,2}$.

Take any vector $\mathbf{w}$ orthogonal to H. Since $\boldsymbol{\mu} \in H$, for any $\boldsymbol{\mu}', \boldsymbol{\mu}'' \in M'$ we have $|\mathbf{w} \cdot \boldsymbol{\mu}' - \mathbf{w} \cdot \boldsymbol{\mu}| > 0$ and $\mathrm{sign}(\mathbf{w} \cdot \boldsymbol{\mu}' - \mathbf{w} \cdot \boldsymbol{\mu}) = \mathrm{sign}(\mathbf{w} \cdot \boldsymbol{\mu}'' - \mathbf{w} \cdot \boldsymbol{\mu})$. Take any $\boldsymbol{\mu}' \in M'$ and define:

$$\mathbf{w}_{\boldsymbol{\mu}} = \begin{cases} \mathbf{w} & , \text{ if } \mathbf{w} \cdot \boldsymbol{\mu}' - \mathbf{w} \cdot \boldsymbol{\mu} < 0 \\ -\mathbf{w} & , \text{ if } \mathbf{w} \cdot \boldsymbol{\mu}' - \mathbf{w} \cdot \boldsymbol{\mu} > 0 \end{cases} .$$

In this case $\mathbf{w}_{\boldsymbol{\mu}} \cdot \boldsymbol{\mu} > \mathbf{w}_{\boldsymbol{\mu}} \cdot \boldsymbol{\mu}'$ for any $\boldsymbol{\mu}' \in M'$. □

Theorem 2. *Let Π, M, $\mathcal{M}$ and $\mathcal{V}$ be defined as in the previous theorem. Let Γ be the set of nondominated policies of an IRMDP where $\mathcal{R} = \{r(s,a) | \mathbf{w} \in [-1,1]^k$ and $r(s,a) = \mathbf{w} \cdot \boldsymbol{\phi}(s,a)\}$. Let $M_{\Gamma} = \{\boldsymbol{\mu}_{\pi} | \pi \in \Gamma\}$, then $\mathcal{V} = M_{\Gamma}$.*

[2] The operator $\mathrm{co}(\cdot)$ stands for the convex hull operator.

Proof. In theorem 1 we prove that $\boldsymbol{\mu} \in \mathcal{V} \Rightarrow \boldsymbol{\mu} \in M_\Gamma$. Now we prove the reverse. Consider the set $D = M_\Gamma - \mathcal{V}$, we have $D \subset M - \mathcal{V}$ and we want to show that D must be empty.

Suppose D is not empty and consider a feature vector $\boldsymbol{\mu}' \in D$. Suppose that exists $\mathbf{w}_{\boldsymbol{\mu}'}$ such that:

$$\mathbf{w}_{\boldsymbol{\mu}'} \cdot \boldsymbol{\mu}' - \mathbf{w}_{\boldsymbol{\mu}'} \cdot \boldsymbol{\mu} > 0 \text{ for any } \boldsymbol{\mu} \neq \boldsymbol{\mu}' \in \text{M}. \tag{5}$$

In this case $\mathbf{w}_{\boldsymbol{\mu}'}$ and $\boldsymbol{\mu}'$ define uniquely a hyperplane $H' = \{\boldsymbol{\mu} | \mathbf{w}_{\boldsymbol{\mu}'} \cdot \boldsymbol{\mu}' - \mathbf{w}_{\boldsymbol{\mu}'} \cdot \boldsymbol{\mu} = 0\}$.

Because $\boldsymbol{\mu}'$ is not a vertex of $\mathcal{M}$, the half-spaces $\mathcal{X}_{H',1}$ and $\mathcal{X}_{H',2}$ defined by H' are such that $\exists \boldsymbol{\mu} \in M \cap \mathcal{X}_{H',1}$ and $\exists \boldsymbol{\mu} \in M \cap \mathcal{X}_{H,2}$. Therefore equation 5 is not true for any $\mathbf{w}_{\boldsymbol{\mu}'}$ and $D = \emptyset$. □

4.2 Finding Nondominated Policies

Theorem 2 shows that finding the set of nondominated policies is the same as finding the set of vertices of the convex hull of feasible expected feature vector.

Given a set of vectors, the Quickhull algorithm finds the convex hull of such set. At any iteration the Quickhull algorithm maintains a current polytope. For a chosen facet of the current polytope, the Quickhull algorithm analyses vectors above the facet by calculating the distance between each vector and the facet. The farthest vector is included in the current polytope by creating new facets based on the facets that such vector can see. The algorithms iterates until there exist vectors outside the current polytope [1].

Just as in the Quickhull algorithm, we consider an initial polytope $\widehat{\mathcal{M}}$, and for each facet we search for the farthest vector not in $\widehat{\mathcal{M}}$. However, if we consider the set of all policies Π and their respective expected feature vectors M, the number of vectors to be analysed is too big. Instead, we find the farthest vector of a facet by choosing wisely a weight vector $\mathbf{w}$ and solving an MDP with the reward function modelled by the weight vector $\mathbf{w}$. We show how to choose such weight vector in the next theorem.

Consider an initial polytope $\widehat{\mathcal{M}}$ whose vertices $\widehat{\mathcal{V}}$ are nondominated policies with hypervolume greater than 0. Note that the number of vertices must be at least $k+1$. We start with $\widehat{\mathcal{V}}$ and add new vertices until polytopes $\widehat{\mathcal{M}} = \text{co}(\mathcal{V})$ and $\mathcal{M} = \text{co}(\{\boldsymbol{\mu}_\pi | \pi \in \Pi\})$ are the same. The idea is to test whether each facet of the polytope $\widehat{\mathcal{M}}$ is also a facet of $\mathcal{M}$. If it is not the case, look for a new vertex $\boldsymbol{\mu}$ and add $\boldsymbol{\mu}$ to $\mathcal{V}$. We can do this thanks to the following theorem.

Theorem 3. *Let $\mathcal{H}$ be the smallest set of hyperplanes which constrain the polytope $\mathcal{M}$. Let $\widehat{\mathcal{H}}$ be the smallest set of hyperplanes that bound a polytope $\widehat{\mathcal{M}} \subset \mathcal{M}$. Take a hyperplane $H \in \widehat{\mathcal{H}}$. $H \in \mathcal{H}$ iff there is no policy π such that:*

$$\mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}_\pi > \mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}_H,$$

where $\boldsymbol{\mu}_H$ is any feature vector in H and $\mathbf{w}_{H,\widehat{\mathcal{M}}}$ is such that: $\mathbf{w}_{H,\widehat{\mathcal{M}}}$ and H are orthogonal, and for any $\boldsymbol{\mu} \in \widehat{\mathcal{M}}$ we have $\mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}_H \geq \mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}$.

Proof. Consider that there exists a policy π such that $\mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}_\pi > \mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}_H$. Because of the definition of $\mathbf{w}_{H,\widehat{\mathcal{M}}}$, $\boldsymbol{\mu}_\pi$ is beyond[3] hyperplane H in the direction $\mathbf{w}_{H,\widehat{\mathcal{M}}}$.

[3] A vector $\boldsymbol{\mu}$ is beyond a hyperplane H with respect to the direction $\mathbf{w}$, if for any vector $\mathbf{x} \in H$ it is true that $\langle \mathbf{w}, \boldsymbol{\mu} \rangle > \langle \mathbf{w}, \mathbf{x} \rangle$.

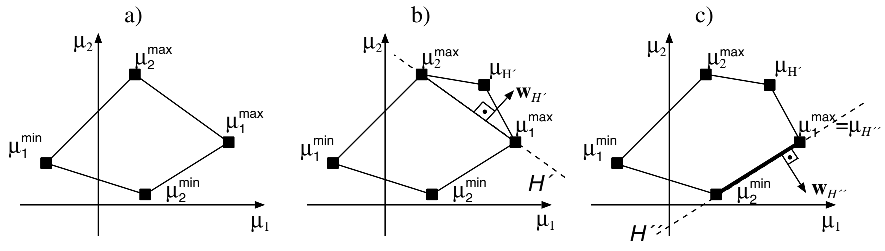

Fig. 1. Constructing the set of feasible expected feature vectors in two dimensions. a) The initial polytope (polygon) $\widehat{\mathcal{M}}$ has vertices that minimise and maximise each feature separately. b) Example of a hyperplane (edge) of the polytope $\widehat{\mathcal{M}}$ which is not in the polytope $\mathcal{M}$. c) Example of a hyperplane (edge) of the polytope $\widehat{\mathcal{M}}$ which is in the polytope $\mathcal{M}$.

Therefore H constrains the feature vector $\boldsymbol{\mu}_\pi$ and $\boldsymbol{\mu}_\pi \notin \widehat{\mathcal{H}}$. However, $\boldsymbol{\mu}_\pi \in \mathcal{M}$ because $\boldsymbol{\mu}_\pi$ is a feasible expected feature vector. Therefore $H \notin \mathcal{H}$.

Now suppose $H \notin \mathcal{H}$. Because $\widehat{\mathcal{M}} \subset \mathcal{M}$ and constraints in $\widehat{\mathcal{H}}$ are tighter than constraints in $\mathcal{H}$, there exists a feasible expected feature vector beyond H. Therefore there exists a policy π such that $\mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}_\pi > \mathbf{w}_{H,\widehat{\mathcal{M}}} \cdot \boldsymbol{\mu}_H$. □

Theorem 3 suggests a rule to improve the set $\widehat{\mathcal{V}}$ so that it becomes the set $\mathcal{V}$ desired:

– Initially a set of vertices $\widehat{\mathcal{V}}$ is considered and the polytope $\widehat{\mathcal{M}} = \mathrm{co}(\widehat{\mathcal{V}})$ is computed (figure 1a).
– For each hyperplane H' which constrains the polytope $\widehat{\mathcal{M}}$ and the vector $\mathbf{w}_{H'}$ orthogonal to H' with direction outwards regarding to the polytope $\widehat{\mathcal{M}}$ (figures 1b and 1c), calculate an optimal policy $\pi^*_{\mathbf{w}_{H'}}$ for $\mathbf{w}_{H'}$ and its corresponding expected feature vector $\boldsymbol{\mu}^{\pi^*_{\mathbf{w}_{H'}}}$.
– If $\boldsymbol{\mu}^{\pi^*_{\mathbf{w}_{H'}}}$ is beyond H', then $\boldsymbol{\mu}^{\pi^*_{\mathbf{w}_{H'}}}$ can be added to $\widehat{\mathcal{V}}$ (figure 1b).
– Otherwise the hyperplane H' constrains the set $\mathcal{M}$ (figure 1c).
– If all the hyperplanes that constrain $\widehat{\mathcal{M}}$ also constrain the set $\mathcal{M}$, then $\widehat{\mathcal{M}} = \mathcal{M}$.
– The end of the process is guaranteed since the cardinality of the set of nondominated policies is finite.

4.3 Normal Vectors and Reward Constraints

In previous section we give some directions on how to find nondominated policies when the set of reward functions are unconstrained. In fact, we consider reward functions to be constrained only to a description based on features. In order to find the farthest feature vector, we considered an orthogonal vector to each facet. However, although an orthogonal vector can be easily calculated, it may not be a feasible weight vector. In this section, we find a *potential* farthest feature vector in three steps: *(i)* by using the orthogonal vector $\mathbf{w}^{ort}$, find the farthest potential feature vector $\boldsymbol{\mu}^{far}$, *(ii)* verify if there exists a witness $\mathbf{w}^{wit}$ to $\boldsymbol{\mu}^{far}$, and *(iii)* find a feasible expected feature vector by maximising the witness $\mathbf{w}^{wit}$.

In the first step, instead of solving an MDP and finding a feasible expected feature vector, which requires optimisation within $|S|$ constraints, we work with potential feature vectors, i.e., vectors in the space under relaxed constraints. By doing so, we can approximate such a solution to a linear optimisation within k constraints in the feature space. First, in the feature space, the lower and upper bounds in each axis can be found previously. We solve MDPs for weight vectors in every possible direction, obtaining respectively upper and lower scalars bounds μ_i^{top} and μ_i^{bottom} for every feature i. Second, when looking for vectors beyond a facet, only constraints applied to such a facet should be considered. Then, given a facet constructed from expected feature vectors $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$ with corresponding weight vectors $\mathbf{w}_1, \ldots, \mathbf{w}_k$ and the orthogonal vector $\mathbf{w}^{ort}$, the farthest feature vector $\boldsymbol{\mu}^{far}$ is obtained from:

$$\begin{array}{rl} \max\limits_{\boldsymbol{\mu}^{far}} & \mathbf{w}^{ort} \cdot \boldsymbol{\mu}^{far} \\ \text{subject to:} & \mathbf{w}_i \cdot \boldsymbol{\mu}^{far} < \mathbf{w}_i \cdot \boldsymbol{\mu}_i, \text{ for } i = 1, \ldots, k \\ & \mu_i^{bottom} \leq \mu_i^{far} \leq \mu_i^{top}, \text{ for } i = 1, \ldots, k \end{array} . \tag{6}$$

The second step verifies if there exists a witness that maximises $\boldsymbol{\mu}^{far}$ compared to $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$. Note that $\boldsymbol{\mu}^{far}$ may not be a feasible expected weight vector ($\mathbf{f}^{s,a}$ is actually feasible in πWitnessBound). But $\boldsymbol{\mu}^{far}$ indicates an upper limit regarding to the distance of the farthest feasible feature vector.

The third step solves an MDP and finds a new expected feature vector to be added to $\widehat{\mathcal{V}}$. This step is the most expensive and we will explore the second step to avoid running into it unnecessarily. If a limited number of nondominated policies is required, not all the policies will be added to $\widehat{\mathcal{V}}$. We can save run-time if we conduct the third step only when necessary, adopting the second step as an advice.

4.4 Initial Polytope

Previously we considered a given initial polytope with hypervolume greater than 0. In this section we will discuss how to obtain that. Two necessary assumptions regarding to the existence of a polytope are: *(i)* all features are relevant, i.e. none can be directly described by the others and, *(ii)* for each feature, different expected occurrences can be obtained with the MDP.

First, if we consider an initial set of weight vector $\mathcal{W} = \{\mathbf{w}_1\}$ and its corresponding set of optimal expected feature vector $\mathcal{V} = \{\boldsymbol{\mu}_1\}$, the idea is to find the farthest feasible feature vector, regarding to the MDP dynamics and the constrained weight vector as previously done. However, there exist more than one orthogonal vector. In order to overcome such problem, we use the following linear program:

$$\begin{array}{rl} \min\limits_{\mathbf{w}^{ort,-}} & \mathbf{w}^{ort,-} \cdot \sum\limits_{\mathbf{w} \in \mathcal{W}} \frac{\mathbf{w}}{|\mathcal{W}|} \\ \text{subject to:} & \mathbf{w}^{ort,-} \cdot (\boldsymbol{\mu}_i - \boldsymbol{\mu}_j) = 0, \forall \boldsymbol{\mu}_i \neq \boldsymbol{\mu}_j \in \mathcal{V} \end{array} . \tag{7}$$

Note that $\mathbf{w}^{ort,-}$ must be orthogonal to any ridge formed by feature vectors in $\mathcal{V}$, but at the same time it tries to be opposite to the weight vectors already maximised. A version $\mathbf{w}^{ort,+}$ in the average direction of $\mathcal{W}$ is also obtained.

We use two directions when looking for the farthest feature vectors because it is not possible to know which direction a ridge faces. By using equation 6 we look for the farthest feature vectors $\boldsymbol{\mu}^{far,+}$ and $\boldsymbol{\mu}^{far,-}$. Witnesses $\mathbf{w}^{+}$ and $\mathbf{w}^{-}$ and optimal expected feature vectors $\boldsymbol{\mu}^{+}$ and $\boldsymbol{\mu}^{-}$ are found for both of them. Then, the farthest feature vector of both are added to $\widehat{\mathcal{V}}$. Here, the distance is measured in the directions $\mathbf{w}^{+}$ and $\mathbf{w}^{-}$ relatively to the set $\widehat{\mathcal{V}}$.

This process goes on until $|\mathcal{M}| = k + 1$, when it is possible to construct a polytope. Table 2 presents the the initHull algorithm.

Algorithm 2. The initHull algorithm

```
Input: IRMDP, $k$, $\phi(\cdot)$
choose $\mathbf{w}$ such that $r(s,a) = \mathbf{w} \cdot \phi(s,a) \in \mathcal{R}$
make $\widehat{\mathcal{W}} = \{\mathbf{w}\}$
find the best expected feature vector $\boldsymbol{\mu}$ to $\mathbf{w}$
make $\widehat{\mathcal{V}} = \{\boldsymbol{\mu}\}$
while $|\mathcal{W}| < k + 1$ do
    calculate $\mathbf{w}^{ort,+}$ and $\mathbf{w}^{ort,-}$ (equation 7)
    calculate $\boldsymbol{\mu}^{far,+}$ and $\boldsymbol{\mu}^{far,-}$
    calculate witnesses $\mathbf{w}^{+}$ and $\mathbf{w}^{-}$ (equation 3)
    calculate the best policies $\boldsymbol{\mu}^{+}$ and $\boldsymbol{\mu}^{-}$ (equation 1)
    choose between $\boldsymbol{\mu}^{+}$ and $\boldsymbol{\mu}^{-}$ the farthest feature vector $\boldsymbol{\mu}$
    get the respective weight vector $\mathbf{w}$
    update $\widehat{\mathcal{W}} \leftarrow \widehat{\mathcal{W}} \cup \{\mathbf{w}\}$
    update $\widehat{\mathcal{V}} \leftarrow \widehat{\mathcal{V}} \cup \{\boldsymbol{\mu}\}$
Output: $\mathcal{W}, \widehat{\mathcal{V}}$
```

4.5 The πHull Algorithm

Finally, before presenting the πHull algorithm, we will discuss some performance issues. The analysis of each facet of a polytope consists of three steps: *(i)* find the outward orthogonal direction and the farthest potential feature vector, *(ii)* find a witness and *(iii)* find an optimal policy. The first step is done with a linear program with k constraints and upper and lower bounds, whereas the third step consists in a linear program with $|S|$ constraints. Differently from the πWitness algorithm where witnesses must be compared against all policies in $\Gamma_{\texttt{agenda}}$, the second step should only look at policies in the facet.

Although there is a gain here when using `findWitnessReward`$(\cdot)$, the number of facets in the polytope $\widehat{\mathcal{M}}$ increases exponentially. If a few nondominated policies are required, this gain may overcome the exponential increase. Therefore, it is interesting to provide a solution which does not increase exponentially, even if it leads to some loss in the quality of the subset of nondominated policies.

The algorithm πHull keeps three sets of facets: $\mathcal{H}_{null}$, $\mathcal{H}_{wit}$ and $\mathcal{H}_{best}$. H_{null} keeps facets which have not gone through any step of analyses. $\mathcal{H}_{wit}$ keeps facets which have gone through the first step and they are ordered by the distance between the facet and the potential feature vector. Finally, $\mathcal{H}_{best}$ keeps facets which have passed the three steps and they are ordered by the distance to the farthest expected feature vector.

In each iteration the πHull algorithm processes randomly N_{null} facets from set $\mathcal{H}_{null}$, then it processes in the given order N_{wit} facets from set $\mathcal{H}_{wit}$. Finally, it chooses from $\mathcal{H}_{best}$ the best facet, i.e., the one with the farthest corresponding expected feature vector and adds it to $\widehat{\mathcal{V}}$. Table 3 summarises the πHull algorithm.

Algorithm 3. The πHull algorithm

```
Input: IRMDP, $N_\Gamma$, $N_{null}$, $N_{wit}$, $k$, $\phi(\cdot)$
$\widehat{\mathcal{W}}, \widehat{\mathcal{V}} \leftarrow$ initHull(IRMDP, $k$, $\phi(\cdot)$)
$\mathcal{H}_{null} = co(\widehat{\mathcal{V}})$
$\mathcal{H}_{wit} = \emptyset$
$\mathcal{H}_{best} = \emptyset$
while $|\widehat{\mathcal{V}}| < N_\Gamma$ do
    for $N_{null}$ facets $H$ in $\mathcal{H}_{null}$ do
        $\mathcal{H}_{null} \leftarrow \mathcal{H}_{null}/\{H\}$
        $\mathbf{w}^{ort}$ ←vector orthogonal to $H$
        $\boldsymbol{\mu}^{far}$ ←farthest potential feature vector
        $\alpha$ ←distance between $\boldsymbol{\mu}^{far}$ and $H$
        associate $\alpha$, $\boldsymbol{\mu}^{far}$ and $\mathbf{w}^{ort}$ with $H$
        $\mathcal{H}_{wit} \leftarrow \mathcal{H}_{wit} \cup \{H\}$
    for first $N_{wit}$ facets $H$ in $\mathcal{H}_{wit}$ do
        $\mathcal{H}_{wit} \leftarrow \mathcal{H}_{wit}/\{H\}$
        $\mathbf{w}^{wit} \leftarrow$ findWitnessReward($\boldsymbol{\mu}^{far}, H$)
        $\boldsymbol{\mu}_{best} \leftarrow$ findBest($\mathbf{w}^{wit}$)
        $\alpha$ ←distance between $\boldsymbol{\mu}_{best}$ and $H$
        associate $\alpha$, $\boldsymbol{\mu}^{best}$ and $\mathbf{w}^{wit}$ with $H$
        $\mathcal{H}_{best} \leftarrow \mathcal{H}_{best} \cup \{H\}$
    $\boldsymbol{\mu}^{new}$, $\mathbf{w}^{wit}$ ←first facet in $\mathcal{H}_{best}$
    $\widehat{\mathcal{V}} \leftarrow \widehat{\mathcal{V}} \cup \{\boldsymbol{\mu}^{new}\}$
    $\widehat{\mathcal{W}} \leftarrow \widehat{\mathcal{W}} \cup \{\mathbf{w}^{wit}\}$
    $\mathcal{H}_{null} = co(\widehat{\mathcal{V}})$
Output: $\widehat{\mathcal{V}}$
```

4.6 Complexity Analysis

We argued that a description of reward functions with features is much more compact than a description of reward functions in the full state-action space. In the πHull algorithm we take advantage of such description to introduce a new algorithm to calculate nondominated policies. On the other hand, our algorithm depends on the number of facets of a convex hull.

The number of facets of a convex hull is known to grow exponentially with the space dimension k, but only linear with the number of vertices $|\mathcal{V}|$ [5]. In the πHull algorithm, the number of vertices grows every iteration, which means that a small time is spent in first iterations. However, for $k = 5$ the linear factor is around 32, while for $k = 10$ the linear factor is around 10^6.

It is clear that our adoption of $\mathcal{H}_{null}$ and $\mathcal{H}_{wit}$ is necessary to calculate nondominated policies even with $k = 10$. On the other hand, the distance of the farthest vector of a facet will be smaller and smaller as new facets are added to $\widehat{\mathcal{V}}$. Then, in the first iterations, when the number of facets is small, the farthest vector of all facets will be calculated and kept in $\mathcal{H}_{best}$. As the number of iterations grows and the farthest vector cannot be promptly calculated for all facets, the facets saved in $\mathcal{H}_{best}$ can be good candidates to be added in $\widehat{\mathcal{V}}$, if not the best ones.

Another interesting point in favour of the πHull algorithm is the relationship with MDP solvers. While πWitnessBound relies on small changes in known nondominated policies, the πHull algorithm relies on the expected feature vector of known nondominated policies. For instance, if a continuous state and continuous action IRMDP is used, how to iterate over states and actions? What would be the occupancy frequency in this case?

The πHull algorithm allows any MDP solver to be used. For instance, if the MDP solver finds approximated optimal policies or expected feature vector, the πHull algorithm would not be affected in the first iterations, where the distance of farthest feature vectors is big.

5 Experiments

We performed experiments on synthetic IRMDPs. Apart from the number of features and the number of constraints, all IRMDPs are randomly drawn from the same distribution. We have $|\mathcal{S}| = 50$, $|\mathcal{A}| = 5$, $\gamma = 0.9$ and β is a uniform distribution. Every state can transit only to 3 other states drawn randomly and such transition function is also drawn randomly. $\phi(\cdot)$ is defined in such a way that for any feature i, we have $\sum_{s,a} \phi_i(s,a) = 10$.

We constructed two groups of 50 IRMDPs. A group with $\mathcal{R}$ defined on 5 features and 3 linear constraints, and another group with $\mathcal{R}$ defined on 10 features and 5 linear constraints.

The first experiment compares πWitnessBound, πHull without time limits ($N_{null} = \infty$ and $N_{wit} = \infty$), and πHull with time limits ($N_{null} = 50$ and $N_{wit} = 10$). This experiment was run on the first group of IRMDPs, where $k = 5$. Figure 2 shows the results comparing the run-time spent in each iteration and the error regarding to the recommended decision and its maximum regret, i.e., in each interaction f^* is chosen to be the occupancy frequency that minimises the maximum regret regarding to the current set of nondominated policies and effectiveness is measured by the error $\epsilon = \texttt{MR}(f^*, \mathcal{R}) - \texttt{MMR}(\mathcal{R})$[4].

The second experiment compares πWitnessBound and πHull with time limits ($N_{null} = 100$ and $N_{wit} = 20$). This experiment was run on the second group of IRMDPs where $k = 10$. Figure 3 shows the results.

In both groups the effectiveness of the subset of nondominated policies was similar for the πWitnessBound algorithm and the πHull algorithm. In both cases, a few policies are enough to make decision under minimax regret criterium. The time spent per iteration in πHull with time limited is at least four times lesser when compared to

[4] We estimate $\texttt{MMR}(\mathcal{R})$ considering the union of policies found at the end of the experiment within all of the algorithms: πWitness and πHull.

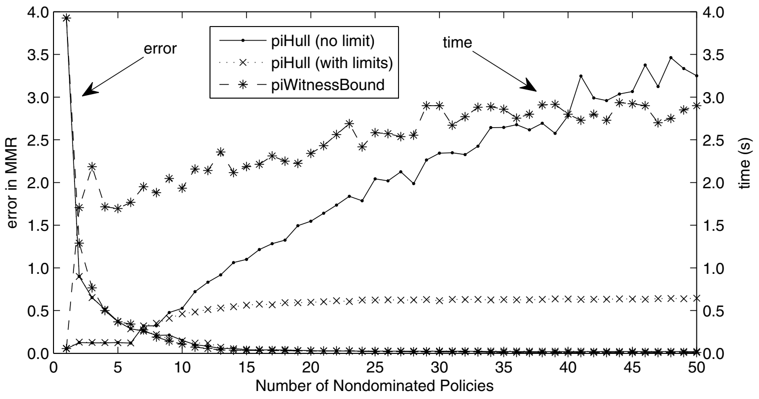

Fig. 2. Results of experiment within 5 features

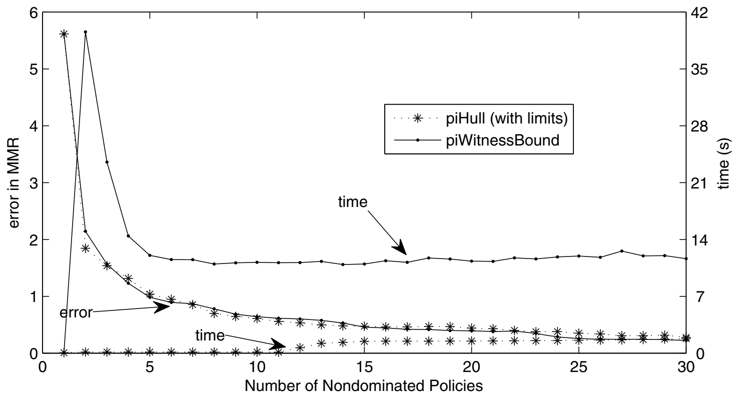

Fig. 3. Results of experiment within 10 features

πWitnessBound. In the case that πHull is used without time limit, the time spent per iteration increases with iterations, but in the early iterations it is still smaller than that found by πWitnessBound.

Our experiments were made with small MDPs (50 states and 5 actions). The run-time of each algorithm depends on the technique used to solve an MDP. However, as πWitnessBound does not take advantage of the reward description based on features, it depends also on how big the sets of state and actions are. On the other hand, πHull depends only the number of features used. In conclusion, the higher the cardinality of the sets of states and actions, the greater the advantage of πHull over πWitnessBound.

Although πWitnessBound is not clearly dependent on the number of features, we can see in the second experiment a run-time four times that of first experiment. When the number of features grows, the number of witnesses also increases, which requires a larger number of MDPs to be solved.

6 Conclusion

We presented two algorithms: πWitnessBound and πHull. The first is a slight modification to πWitness, while the second is a completely new algorithm. Both are effective when they define a small subset of nondominated policies to be used for calculating the minimax regret criterium. πHull shows a better run-time performance in our experiments, mainly due to the large difference in the number of features ($k = 5$ and $k = 10$) and number of states ($S = 50$), since πWitnessBound depends on the second.

Although πHull shows a better run-time performance and similar effectiveness, we have not presented a formal proof that πHull always has a better performance. Future works should seek for three formal results related to the πHull algorithm. First, to prove that our analysis of facet reaches the farthest feature vector when the constraints are considered. Second, to establish a formal relation between the number of nondominated policies and the error of calculating $\mathtt{MMR}(\mathcal{R})$. Third, to set the speed with which πHull reaches a good approximation of the set $\widehat{\mathcal{V}}$, which is very important given the exponential growth in the number of facets. We must also examine how the parameters N_{null} and N_{wit} affect this calculation.

Besides the effectiveness and better run-time performance of πHull confronted with πWitnessBound, there is also qualitative characteristics. Clearly the πHull algorithm cannot be used if a feature description is not at hand. However, a reward function is hardly defined on the state-action space. πHull would present problem if the number of features to be used is too big.

The best advantage of πHull is regarding to the MDP solver to be used. In real problems an MDP solver would take advantage of the problems structure, like factored MDPs [14], or would approximate solutions in order to make it feasible [8]. πWitnessBound must be adapted somehow to work with such solvers.

It is worth to notice that nondominated policies can be a good indicator for a preference elicitation process. They give us a hint about policies to be confronted. For instance, the small set of nondominated policies can be used when enumerative analysis must be done.

References

1. Barber, C.B., Dobkin, D.P., Huhdanpaa, H.: The quickhull algorithm for convex hulls. ACM Trans. Math. Softw. 22, 469–483 (1996), http://doi.acm.org/10.1145/235815.235821
2. Bertsekas, D.P.: Dynamic Programming - Deterministic and Stochastic Models. Prentice-Hall, Englewood Cliffs (1987)
3. Boutilier, C., Patrascu, R., Poupart, P., Schuurmans, D.: Constraint-based optimization and utility elicitation using the minimax decision criterion. Artificial Intelligence 170(8), 686–713 (2006)
4. Braziunas, D., Boutilier, C.: Elicitation of factored utilities. AI Magazine 29(4), 79–92 (2008)
5. Buchta, C., Muller, J., Tichy, R.F.: Stochastical approximation of convex bodies. Mathematische Annalen 271, 225–235 (1985), http://dx.doi.org/10.1007/BF01455988, doi:10.1007/BF01455988

6. Chajewska, U., Koller, D., Parr, R.: Making rational decisions using adaptive utility elicitation. In: Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence, pp. 363–369. AAAI Press / The MIT Press, Austin, Texas (2000)
7. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence 101(1-2), 99–134 (1998)
8. Munos, R., Moore, A.: Variable resolution discretization in optimal control. Machine Learning 49(2/3), 291–323 (2002)
9. Patrascu, R., Boutilier, C., Das, R., Kephart, J.O., Tesauro, G., Walsh, W.E.: New approaches to optimization and utility elicitation in autonomic computing. In: Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, pp. 140–145. AAAI Press / The MIT Press, Pittsburgh, Pennsylvania, USA (2005)
10. Regan, K., Boutilier, C.: Regret-based reward elicitation for markov decision processes. In: UAI 2009: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence, pp. 444–451. AUAI Press, Arlington (2009)
11. Regan, K., Boutilier, C.: Robust policy computation in reward-uncertain mdps using nondominated policies. In: Fox, M., Poole, D. (eds.) AAAI, AAAI Press, Menlo Park (2010)
12. White III, C.C., Eldeib, H.K.: Markov decision processes with imprecise transition probabilities. Operations Research 42(4), 739–749 (1994)
13. Xu, H., Mannor, S.: Parametric regret in uncertain markov decision processes. In: 48th IEEE Conference on Decision and Control, CDC 2009 (2009)
14. Guestrin, C., Koller, D., Parr, R., Venkataraman, S.: Efficient solution algorithms for factored MDPs. Journal of Artificial Intelligence Research 19, 399–468 (2003)