Approximate Regret Based Elicitation in Markov Decision Process

Pegah Alizadeh
Institut Galilée,
Université Paris-Nord
99, avenue Jean-Baptiste Clément
93430 Villetaneuse
Email: pegah.alizadeh@lipn.univ-paris13.fr

Yann Chevaleyre
Institut Galilée,
Université Paris-Nord
99, avenue Jean-Baptiste Clément
93430 Villetaneuse
Email: chevaleyre@lipn.univ-paris13.fr

Jean-Daniel Zucker
IRD, UMI 209, UMMISCO,
IRD France Nord,F-93143, Bondy, France.
MSI, IFI, 42 Ta Quang Buu street,
Hai Ba Trung District, Hanoi, Vietnam
Email: jean-daniel.zucker@ird.fr

***Abstract*—Consider a decision support system (DSS) designed to find optimal strategies in stochastic environments, on behalf of a user. To perform this computation, the DSS will need a precise model of the environment. Of course, when the environment can be modeled as a *Markov decision process (MDP)* with numerical rewards (or numerical penalties), the DSS can compute the optimal strategy in polynomial time. But in many real-world cases, rewards are unknown. To compensate this missing information, the DSS may query the user for its preferences among some alternative policies. Based on the user's answers, the DSS can step-by-step compute the user's preferred policy. In this work, we describe a computational method based on *minimax regret* to find optimal policy when rewards are unknown. Then we present types of queries on feasible set of rewards by using preference elicitation approaches. When user answer these queries based on her preferences, we will have more information about rewards which will result in more desirable policies.** [1]

I. Introduction

Suppose Bob wants to determine the shortest route from his house to his work place, using a computer equipped with a GPS. If the computer has access to a map, this problem is easily solved with a shortest-path computation. Assume now that the computer has also access to some statistics about traffic jams: for each road, it knows what is the probability of being slowed down. With this new information, the environment may now be modeled as a stochastic Markov Decision Process (MDP), and Bob's driving strategy can still be efficiently computed. In the other hand, suppose the long term effectiveness of some decisions in some situations are the same in the environment, for instance choosing to turn to the right or continuing straight ahead by being in a junction like T both will result in the same final strategy. Then, instead of defining a new effect for every state and decision, a limited set of possible effects (rewards) can be defined in the MDP. Thus, a wide set of rewards will diminish to a set of possible rewards with few members. Moreover, assume Bob is a very demanding individual which is not satisfied with the *shortest* route, because he has more complex preferences. For example, he might prefer not to drive on highways. He might want to pick up his friend who is waiting for him in the middle of the way. He might also want to avoid driving through large cities or on winding roads. In Bob's mind, his preferences are ordinal rather than cardinal: he cannot evaluate routes with a single number (the expected arrival time), but given two routes A and B, Bob can naturally tell which one he prefers. To find the most satisfying route for Bob, the computer will have to ask Bob many queries in order to elicit his preferences. These queries might be of the form "do you prefer route A to route B ?". Based on these queries, the computer will build a compact model of Bob's preferences, and it will then compute Bob's prefered routes.

Recently a special model of MDPs *with imprecise reward (IRMDPs)* have been proposed [4], [9]. They compute policies which are robust with respect to the reward uncertainty by using minimax regret [3]–[7], [10]. Existing models study various structures on reward functions like "flat" reward function [4], [10] or "additive" reward functions [7]. We use another framework namely *ordinal reward MDPs* [8] in which rewards are taken from a given scale, but in this work orders on rewards is not given.

In this paper, we compute the optimal policy for an ordinal reward MDP. Assuming that very few information about the rewards is that they are between zero and one. To elicit more precise rewards and compute the optimal policy, we ask queries on scale of rewards. There is a user who answers these queries regarding her preferences among trajectories in the Markov Decision process. Our goal is finding optimal policies by asking a few number of informative queries on rewards from the user.

To achieve this goal, we provide two main contributions. First, we develop a computational method (inspired by Ragan and Boutilier work [5]) based on minimax regret for IRMDPs with a special structure on unknown rewards. This structure is also inspired by Weng and Zanuttini work [9]. Second, we develop a heuristic elicitation method for IRMDPs who asks comparison questions among rewards by using the idea of Halve largest Gap [1]. Through these two contributions we adapt our knowledge about reward parameters and will be able to find optimal or near-optimal policies effectively.

We proceed by first reviewing some related definitions to MDPs and explaining minimax regret computation methods for IRMDPs in Section II. We then describe our approach of constraint generation procedure used to compute minimax regret in Section III. In Section IV, we discuss queries types used to reduce the regret and in Section V we investigate efficiency of the presented minimax regret method and reward elicitation procedure. We finally offer some conclusions and

[1]Thanks to Clément Pira for his suggestions and discussions related to defining types of queries.

suggestions for future work in Section VI.

II. Background

In this section we will present some notifications and formulations of Markov decision processes and minimax regret criterion.

A. Markov Decision Process(MDP)

A *Markov Decision Process(MDP)* [2] is defined by $\mathcal{M} = (S, A, p, r, \gamma, \beta)$ where S is a set of states, A is a set of actions (S and A are finite sets), $p(s'|s,a)$ is a probability of choosing state s' by being in state s and taking action a. $r : S \times A \to \mathbb{R}$ is a reward function, $0 \leq \gamma < 1$ is a discount factor and β is a distribution on initial states.
In this work, for convenience we use matrix-vector notations. Assume $|S| = n$ and $|A| = k$, then $\mathbf{r}$ is a matrix of length $n \times k$ whose elements of matrix are rewards $r(s,a)$. $\mathbf{P}$ is a $(n \times k) \times n$ probability transition matrix, where every row presents pair of (state, action) and every column is a state. Another definition is matrix $\mathbf{E}$ which has the same dimension as matrix $\mathbf{P}$ and is defined as follows:

$$\mathbf{E}(s'|s,a) = \begin{cases} p(s'|s,a) & \text{if } s' \neq s \\ p(s'|s,a) - 1 & \text{if } s' = s \end{cases}$$

A deterministic **policy** is a function like $\pi : S \longrightarrow A$ which allocates an action a to every state s. To compare policies with each other, they are evaluated by a **value function** v^π which is expected sum of rewards and is defined as follows:

$$v^\pi(s) = \mathbb{E}_{s_0 \sim \beta}[\sum_{t=0}^{\infty} \gamma^t r(s_t, \pi(s_t))|\pi] \tag{1}$$

Our goal as a user is to find the *optimal policy* that denotes by π^* such that:

$$\pi^* = argmax_{\pi'} \ \ v^{\pi'} \tag{2}$$

Remind that $v^\pi \succeq v^{\pi'} \Leftrightarrow \forall s \in S, \ v^\pi(s) \geq v^{\pi'}(s)$. The optimal policy π^* can be deduced from the optimal value function v^*, which can be computed with the *bellman equation* as follows:

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

$$\pi^*(s) = argmax_{a \in A} \ r(s,a) + \gamma \sum_{s' \in S} p(s'|s,a) v^*(s')$$

Note that these equations can be linearized, and then solved with any Linear Programming package.

Interestingly, there exists a dual formulation of the above Bellman's equations based on the notion of **occupancy frequency function** [2]. This dual formulation also leads us to an alternative way of computing the optimal policy. The occupancy frequency function $\mathbf{f}^\pi : S \times A \longrightarrow \mathbb{R}$ of a policy π is the total discounted probability of being in state s and taking action a:

$$\mathbf{f}^\pi(s,a) = \sum_{t=0}^{\infty} \gamma^t P\left(s_t = s, a_t = a\right)$$

The set $\mathcal{F}$ of valid occupancy frequencies for a given MDP is the polytope of all functions $\mathbf{f}$ satisfying $\gamma \mathbf{E}^\top \mathbf{f} + \beta = \mathbf{0}$ [2]. The value of the policy associated to the occupancy function $\mathbf{f}^\pi$ is:

$$v^\pi(s) = \mathbf{r}^\top . \mathbf{f}^\pi \tag{3}$$

We can now find the optimal occupancy frequency by solving the following Linear Programming problem:

$$\begin{gathered} \mathbf{f}^* = argmax_{\mathbf{f} \in F} \ \ \mathbf{r}^\top . \mathbf{f} \\ subj \ to \ \mathbf{f} \in \mathcal{F} \end{gathered} \tag{4}$$

Note that the optimal policy π^* can again be recovered from the optimal occupancy frequency by the following equation: $\pi(s,a) = \mathbf{f}^\pi(s,a) / \sum_{a' \in A} \mathbf{f}^\pi(s,a')$.

B. Imprecise Reward MDP

Inspired by *Ordered Reward MDP(ORMDP)* in [8], we introduce an MDP with unknown rewards where reward function is defined as $\hat{r} : S \times A \to E$. E is the set of possible reward values $E = \{\lambda_1, .., \lambda_d\}$, such that $\forall \ \lambda_i \in E \ \ 0 \leq \lambda_i \leq 1$. Again λ_is are unknown and will be estimated by interaction with the user.
As explained in [9] we reformulate ORMDP to an Imprecise Reward MDP $(S, A, p, \bar{r}, \gamma, \beta)$ where $\bar{r}(s,a)$ is a d-dimensional vector. If $\hat{r}(s,a) = \lambda_i$ then $\bar{r}(s,a) = (0, .., 0, 1, 0, .., 0)$ which 1 is the ith element of vector. The reward of state action s and a is $r(s,a) = [\lambda_1, .., \lambda_d].\bar{r}^\top(s,a)$. In this way we have $r(s,a) = \hat{r}(s,a)$. Now, we look for the optimal policy π^*, in other words we look for optimal occupancy function $\mathbf{f}^*$.
Based on our presented structure we have the following notification on rewards :

$$[r(s_1,a_1), \cdots, r(s_1,a_k), \cdots, r(s_n,a_1), \cdots, r(s_n,a_k)] = \underbrace{[\lambda_1, .., \lambda_d]}_{\boldsymbol{\lambda}} \underbrace{\begin{bmatrix} \bar{r}(s_1,a_1) \\ \vdots \\ \bar{r}(s_1,a_k) \\ \vdots \\ \bar{r}(s_n,a_1) \\ \vdots \\ \bar{r}(s_n,a_k) \end{bmatrix}^\top}_{\mathbf{r}_c^\top}$$

Then we have $\mathbf{r} = \boldsymbol{\lambda} \mathbf{r}_c^\top$.Here, $\mathbf{r}_c$ is known but $\boldsymbol{\lambda}$ is known and has be elicited. On the other side, we only know the feasible set of reward values λs which they are confined by a $d-$dimensional cube at the beginning. We denote this feasible set of λs by Λ.
As we have mentioned earlier, we look for

$$\mathbf{f}^* = sup_{\mathbf{g} \in \mathcal{F}} \ \mathbf{g}.\mathbf{r} = sup_{\mathbf{g} \in \mathcal{F}} \ \boldsymbol{\lambda} \mathbf{r}_c^\top . \mathbf{g} \tag{5}$$

C. Preference Elicitation

Among various types of preference elicitation methods, we focus on the one which defines a utility function on the set of possible outcomes. In our case, a set of outcome is a set of all possible occupancy functions ($\mathcal{F}$). If *utility function*

$U : \mathcal{F} \longrightarrow \mathbb{R}^+$ can be extended from user's preferences, we can easily predict the best decision for the user among set of outcomes (The prediction will be $argmax_{\mathbf{f}_x \in \mathcal{F}}\ U(\mathbf{f}_x)$). The problem is that these utility functions are not specified at the beginning in most real cases.
Based on Equation 3, we look for the policy with the max value function in comparison with other policies. In the other hand, value of optimal π^* corresponds to the value of optimal $\mathbf{f}^*$ and we have: $v_{\mathbf{r}}^{\pi^*} = \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{f}^*$. Therefore, considering Equation 5 and the relation between long term value functions and occupancy functions, we define $U(\mathbf{f}) = \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{f}$. If we had precise values of $\boldsymbol{\lambda}$s, we would choose $\mathbf{f}_{output}$ such that $\mathbf{f}_{output} = sup_{\mathbf{g}\in\mathcal{F}}\ U(\mathbf{g}) = sup_{\mathbf{g}\in\mathcal{F}}\ \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{g}$.
In our case like Regan and Boutilier's situations [4], we also don't have full utility functions information. Then, we can refine knowledge of λs through some constraints. In most cases decisions can be made even without knowing exact value utility functions [1] because of two reasons:

1- Optimal decisions can often be made with only little knowledge of utilities.
2- Value of certain utilities regarding their efficiency to make decisions is expensive and is not worthy.

Now, since our imprecise λs are bounded in the Λ polytope, we use *minimax regret decision criterion* to make decisions and find optimal policy $\mathbf{f}^*$. Minimax regret is a robust optimization method in presence of uncertain data and which we utilize it for approximating optimal policies.

Definition 1: The *pairwise regret* of policy $\mathbf{f}$ w.r.t $\mathbf{g}$ over feasible set Λ is defined as

$$R(\mathbf{f}, \mathbf{g}, \Lambda) = max_{\lambda\in\Lambda}\ \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{g} - \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{f}$$

If the system was forced to choose $\mathbf{f}$ instead of $\mathbf{g}$, the worst case loss with respect to possible set of $\lambda \in \Lambda$ would be $R(\mathbf{f}, \mathbf{g}, \Lambda)$.

Definition 2: The *maximum regret* of decision $\mathbf{f}$ is

$$\begin{aligned} MR(\mathbf{f}, \Lambda) &= max_{\mathbf{g}\in\mathcal{F}} R(\mathbf{f}, \mathbf{g}, \Lambda) \\ &= max_{\lambda\in\Lambda}\ max_{\mathbf{g}\in\mathcal{F}}\ \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{g} - \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{f} \end{aligned}$$

$MR(\mathbf{f}, \Lambda)$ is the maximum regret which system can make by choosing policy $\mathbf{f}$.

Definition 3: The *Minimax Regret* of feasible set Λ is:

$$\begin{aligned} &MMR(\Lambda) = min_{\mathbf{f}\in\mathcal{F}}\ MR(\mathbf{f}, \Lambda) \\ &= min_{\mathbf{f}\in\mathcal{F}} max_{\boldsymbol{\lambda}\in\Lambda} max_{\mathbf{g}\in\mathcal{F}}\ \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{g} - \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{f} \end{aligned}$$

Finally, a *minimax optimal decision* $\boldsymbol{f}^*$ is a decision which minimizes the max regret as bellow:

$$\mathbf{f}^* = argmin_{\mathbf{f}\in\mathcal{F}}\ MR(\mathbf{f}, \Lambda)$$

III. Computing Minimax Regret

Minimax regret is a robustness criterion to make a decision in partially known systems and it guides elicitation effectively. Minimax regret computation is a complex task. As an example Xu and Mannor [10] have implemented a computation method for Imprecise Reward MDPs which is NP-hard. Also inspiring *Benders decomposition*, Regan and Boutilier [3], [4] use a Mixed-Integer Program (MIP) model which is only applicable to small size of MDPs[2]. In this section, we present a calculation method namely the *random points method* to solve approximately the minimax regret problem.

A. Selected Random Points Method

Let Λ be a set of feasible $\boldsymbol{\lambda}$ vectors, which can be defined by a set of bounds on reward values specified priorly by a user or an expert. Assume feasible set of rewards Λ is given by a convex polytope $\Lambda = \{\boldsymbol{\lambda} | C\boldsymbol{\lambda} \leq d\}$. Then by using Minimax regret criterion, optimal policy $\mathbf{f}^*$ can be computed from the following equation :

$$\begin{aligned} \mathbf{f}^* = argmin_{\mathbf{f}\in\mathcal{F}}\ max_{\mathbf{g}\in\mathcal{F}}\ max_{\boldsymbol{\lambda}\in\Lambda}\ \ &\boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{g} - \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{f} \\ subject\ to:\ \ &\gamma\mathbf{E}^\top\mathbf{f} + \beta = \mathbf{0} \\ &\gamma\mathbf{E}^\top\mathbf{g} + \beta = \mathbf{0} \\ &C\boldsymbol{\lambda} \leq d \end{aligned} \tag{6}$$

Equation 6 is a quadratic program and solution of these kinds of equations are always complex. Then, we use the constraint generation approach [4] to simplify the equation by changing it to series of linear programmings :

$$\begin{aligned} &minimize_{\mathbf{f},\delta}\ \ \delta \\ &subject\ to:\ \ \delta \geq \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{g} - \boldsymbol{\lambda}\mathbf{r}_c^\top.\mathbf{f}\ \ \ \forall\mathbf{g}\in\mathcal{F},\ \boldsymbol{\lambda}\in\Lambda \\ &\qquad\qquad \gamma\mathbf{E}^\top\mathbf{f} + \beta = \mathbf{0} \end{aligned} \tag{7}$$

Equation 7 is a linear program with an infinite number of constraints, one fore each pair $\mathbf{g}$, $\boldsymbol{\lambda}$. Instead of enumerating all $\boldsymbol{\lambda}$s, we will draw randomly some $\boldsymbol{\lambda}$s from Λ. we inspire idea of Regan and Boutilier [5] in using non-dominated policies. First we select randomly many $\boldsymbol{\lambda}$ points from a given polytope Λ, then solve the following sub-Problem for a given $\boldsymbol{\lambda}_i$. By that, we will form a GEN set (generating set) made up of $(\boldsymbol{\lambda}_i, \mathbf{g}_i)$ pairs:

$$\begin{aligned} \mathbf{g}_i = Maximize_{\mathbf{g}}\ \ &\boldsymbol{\lambda}_i\mathbf{r}_c^\top.\mathbf{g} \\ subject\ to:\ \ &\gamma\mathbf{E}^\top\mathbf{g} + \beta = \mathbf{0} \end{aligned} \tag{8}$$

Considering three following algorithms, after preparing generation set we will find series of constraints and will compute our main linear programming problem (Based on Algorithm III-A) which will result in the optimal policy $\mathbf{f}^*$.

Algorithm 1 Find-$\mathbf{g}(\boldsymbol{\lambda}_i)$

Input: $\boldsymbol{\lambda}_i$
Output: $\mathbf{g}_i$
1: $\mathbf{g}_i \longleftarrow Maximize_{\mathbf{g}}\ \boldsymbol{\lambda}_i\mathbf{r}_c^\top.\mathbf{g}$
2: $\qquad subject\ to:\ \gamma\mathbf{E}^\top\mathbf{g} + \beta = \mathbf{0}$
3: **return** $\mathbf{g}_i$

We changed the quadratic problem (Equation 6) to the linear programming with $|S|.|A| + 1$ number of variables and $K + |S|$ number of constraints (K is number of randomly selected points from Λ polytope). Moreover, every constraint is a solution of another LP with $|S|.|A|$ number of variables and $|S|$ number of constraints.

[2]We will investigate their results as well in experimental part of this paper.

Algorithm 2 Calculate-GEN-set(N, Λ)

Input: let N be the number of randomly selected points from polytope Λ
Output: GEN set is including constraints for main problem

$\Lambda_{random} \longleftarrow \emptyset$ random selected points from polytope Λ
$GEN \longleftarrow \emptyset$
for $i = 1$ to $i \leq N$ **do**
  choose $\boldsymbol{\lambda_i}$ randomly in Λ
  $\Lambda_{random} \longleftarrow \Lambda_{random} \cup \boldsymbol{\lambda_i}$
end for
for $\boldsymbol{\lambda}_i \in \Lambda_{random}$ **do**
  $\mathbf{g}_i \longleftarrow$ find-$\mathbf{g}(\boldsymbol{\lambda_i})$
  $GEN \longleftarrow GEN \cup \{(\boldsymbol{\lambda}_i, \mathbf{g}_i)\}$
end for
return GEN

Algorithm 3 Calculate $\mathbf{f}^*$ for a given polytope $\Lambda = \{\boldsymbol{\lambda} | C\boldsymbol{\lambda} \leq d\}$

Input: N is the number of selected points from polytope Λ
Output: $\mathbf{f}^*$

Constraint $\longleftarrow \emptyset$
$GEN \longleftarrow$ Calculate-GEN-set(N, Λ)
for $i = 1$ to N **do**
  Choose $(\boldsymbol{\lambda}_i, \mathbf{g}_i) \in GEN$
  Constraint $\longleftarrow$ Constraint $\cup\ \boldsymbol{\lambda}_i \mathbf{r}_c^\top \mathbf{g}_i - \boldsymbol{\lambda}_i \mathbf{r}_c^\top \mathbf{f} \leq \delta$
end for
Constraint $\longleftarrow$ Constraint $\cup\ \gamma \mathbf{E}^\top \mathbf{f} + \beta = \mathbf{0}$
$\mathbf{f}^* \longleftarrow Min_{\delta, \mathbf{f}}\ \delta$
  $subject\ to:$ Constraint
return $\mathbf{f}^*$

IV. Reward Elicitation

We now wish to elicit and asses current knowledge of reward functions (in our case Λ polytope) by proposing informative and effective queries to the user after each interaction between her and the Decision Support System. As explained in [3], these queries are questions about user preferences in many different forms such as reward values (in our case are λs), states, state-action pairs or trajectories. After user response, every query implements a new bound on the reward scale and shrinks the feasible the reward set. Assume, the Λ polytope is given by a set of inequalities on a linear combination of $\lambda_1, .., \lambda_d$. At the beginning a given bound on λs is a unit cube[3] in d-dimensional space. These inequalities have the form "Is $w_1\lambda_1 + ... + w_d\lambda_d \geq a$?" where $W.\boldsymbol{\lambda} - a = 0$ is the plane in which $W = (w_1, .., w_d)$ is the unit normal vector[4] of plane and $a = W.\boldsymbol{\lambda}_0$ (The plane passes through the point $\boldsymbol{\lambda}_0$). We remind that the proposed queries to the user should be easy enough to be answered by the user.

We use a heuristic strategy namely *Halve Largest Gap (HLG)* [1] to find query forms, reduce regret and finally find the optimal policy. Based on HLG method for every two vertices (v_i, v_j) from polytope Λ, we define a Gap as :

$$\Delta(v_i, v_j) = max_{P \in Q}\ \{d(P, v_i) + d(P, v_j) \mid d(P, v_i) = d(P, v_j)\}$$

$$Q = \{\sum_{i=1}^{d} w_i \lambda_i - a = 0 | \sum w_i^2 = 1\}$$

$d(P, v_i)$ is the distance between plane P and point v_i. In the other words, $\Delta(v_i, v_j)$ is the max sum of distance of v_i and v_j from a plane like P which has the equal distance from two given vertices. Now we choose P^* plane such that:

$$P^* = argmax_{(v_i, v_j)}\ \Delta(v_i, v_j) \qquad s.t.\quad v_i, v_j \in vertices(\Lambda)^5$$

The query has the form : "Is $P^* \geq 0$?", which means which side of plane P^* is preferred by the user. Assume v_i and v_j are given, thus finding the appropriate plane $P = W\boldsymbol{\lambda} - a$ will change to the following problem:

$$\begin{aligned} &Max_{W, v_i, v_j}\ \ W.(v_i - v_j) \\ &subject\ to:\ Cv_i \leq d \\ &\qquad\qquad\ \ Cv_j \leq d \\ &\qquad\qquad\ \ ||W|| = 1 \end{aligned} \tag{9}$$

We mentioned that the plane has equal distance from two chosen points: $W.v_i - a = -W.v_j + a$. Thus after solving Equation 9, a will be calculated by:

$$a = \frac{W.(v_i + v_j)}{2} \tag{10}$$

Equation 9 is equivalent to the simple maximization:

$$\begin{aligned} &Max_{W, v_i, v_j}\ \ W.(v_i - v_j) \\ &subject\ to:\ ||W|| = 1 \quad \forall v_i, v_j\ \in vertices(\Lambda) \end{aligned} \tag{11}$$

To clarify the proceeding part of this section, consider W^*, v_i^* and v_j^* are d-dimensional vectors and they are solutions of Equation 11. Thus, the proper plane will be $W^*\boldsymbol{\lambda} - a^*$ where $a^* = W^*.(v_i^* + v_j^*)/2$. Equation 11 is a quadratic problem which is computationally complex in case of huge set of $vertices(\Lambda)$. To tackle this problem we form two LP problems inspired by the iterative solution to quadratic problem presented by [5] (Modified ICG-ND). If W_{fix} is a given normal vector of plane, then a linear programming which we call **Find-Pairs()** is:

$$\begin{aligned} &max_{v_i, v_j} W_{fix}.(v_i - v_j) \\ &subject\ to:\ Cv_i \leq d \\ &\qquad\qquad\ \ Cv_j \leq d \end{aligned}$$

In the other hand, for two given points v_{ifix}, v_{jfix}, the normal vector of plane is as following:

$$\begin{aligned} &max_W\ W.(v_{ifix} - v_{jfix}) \\ &subject\ to:\ ||W|| = 1 \end{aligned} \tag{12}$$

Equation 12 is quadratic problem again and can be simplified as a simple equation namely **Find-Normal-Vector()**:

$$W = \frac{v_{ifix} - v_{jfix}}{||v_{ifix} - v_{jfix}||}$$

In both Find-Pairs() and Find-Normal-Vectors(), we find value of a from Equation 10.

Lastly to solve Equation 11, we use Algorithm IV that is iteration of Find-Pairs() and Find-Normal-Vector().

[3] A cube whose sides are 1 unit long and it is located in the first orthant with one vertice on the origin.

[4] The normal vector perpendicular to the plane.

[5] $vertices(\Lambda)$ is a set of all Λ polytope vertices.

Algorithm 4 Find-Query()

Input: $C\boldsymbol{\lambda} \leq d$ polytope and ϵ constant are given
Output: we look for $W^*\boldsymbol{\lambda} - a^*$ plane

$(W^*\boldsymbol{\lambda} - a^*)_{first} \longleftarrow \mathbf{0}$
$x \longleftarrow$ choose a random point $\in vertices(C\boldsymbol{\lambda} \leq d)$
$y \longleftarrow$ choose a random point $\in vertices(C\boldsymbol{\lambda} \leq d)$
$(W^*\boldsymbol{\lambda} - a^*)_{final} \longleftarrow$ **Find-Normal-Vector**(x, y)
$\Delta \longleftarrow ||(W^*\boldsymbol{\lambda} - a^*)_{final} - (W^*\boldsymbol{\lambda} - a^*)_{first}||$
while $\Delta > \epsilon$ **do**
    $x, y \longleftarrow$ **Find-Pair**$(W^*{}_{final})$
    $(W^*\boldsymbol{\lambda} - a^*)_{first} \longleftarrow (W^*\boldsymbol{\lambda} - a^*)_{final}$
    $(W^*\boldsymbol{\lambda} - a^*)_{final} \longleftarrow$ **Find-Normal-Vector**(x, y)
    $\Delta \longleftarrow ||(W^*\boldsymbol{\lambda} - a^*)_{final} - (W^*\boldsymbol{\lambda} - a^*)_{first}||$
end while
return $(W^*\boldsymbol{\lambda} - a^*)_{final}$

V. Experiments

We will examine performance of our Minimax Regret calculation from two points of view including: Number of all possible value of rewards (Λ polytope dimension), and MDP size. Afterwards, we will investigate effectiveness of minimax regret as a conductor of reward elicitation.
Since this work is preliminary and we would like to compare our results with whole previous approaches, we used a set of randomly generated MDPs to evaluate our approach[6]. A random MDP is defined by several parameters including number of states n, number of actions k and number of possible value of rewards d. We utilize normalized set of feasible reward functions for each MDP, where the reward for each pair s, a is bounded between 0 and 1. We implement some structure on MDP with making a semi sparse transition function. From any state s restrict transitions to reach $\lceil \log_2(n) \rceil$ other states. For each pair of (s, a) draw reachable states based on uniform distribution over the set of states. For drawn states, transition probabilities are formed based on Gaussian distribution. For the rest of states transition probabilities are equal to 0. The initial state distribution β is a uniform distribution over states and we choose discount factor $\gamma = 0.95$. Remind that in the first step polytope of rewards is a unite d-dimensional cube. To have more effective results, we average the numeric results for computations over 10 times and we choose 1000 number of random λs for every iteration.

A. Computation efficiency

To measure the performance of our minimax regret computation method (selected random points method), we display the change of average minimax regret computation time with respect to MDP size. Figure 1 indicates how calculation time changes as a function of number of states or number of actions. In one of the diagrams, we fixed number of actions at 5 and varied the number of states while in the other one we fixed number of states at 10 and altered action numbers. Additionally, the Number of unknown rewards (d) is equal 10. Figure 2 and 3 are two results from Regan and Boutilier work. The former shows minimax calculation time using integer linear programming method namely ICG in [3] and the latter indicates minimax calculation time using nondominated policies [5]. **Both of these methods are not able to do calculations for huge size of MDPs because of complexity of their calculations, while our method is applicable for very huge MDPs (Figure 1)**. For instance, we can calculate minimax regret for a MDP with 200 states, while previous approaches don't have any results more than 10 states. Therefore, our approach is significant for MDPs with too many states and actions and few number of different reward values.

[6]We did these experimental results with R version 3.0.1 and "lpSolveAPI" package to solve LPs. We used lrslib Ver 4.3 as a library of the reverse search algorithm for vertex enumeration/convex *http://cgm.cs.mcgill.ca/ avis/C/lrs.html*.

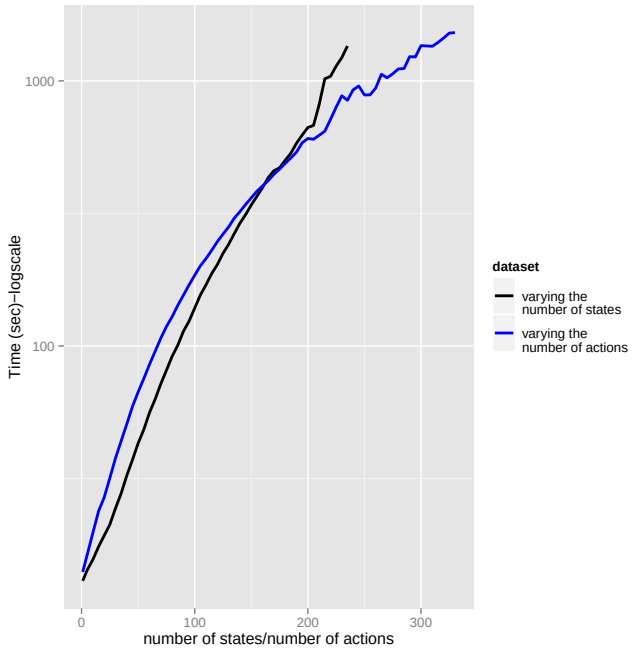

Fig. 1. Changing of states and actions vs calculation time for minimax regret.

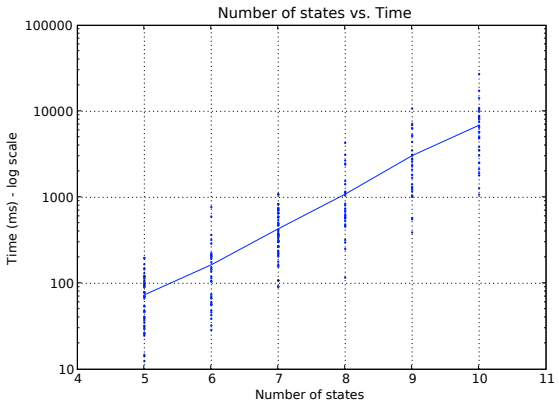

Fig. 2. The ICG can not see beyond 11 states [3]

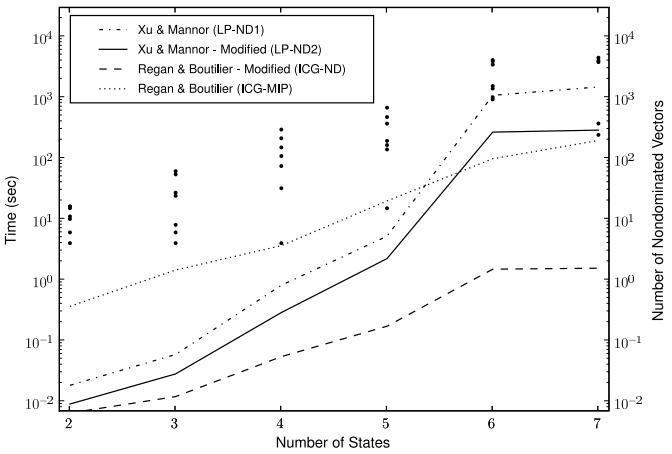

Fig. 3. Scaling of minimax regret computation (line plot on left y-axis) and nondominated policies (scatter plot on right y-axis) w.r.t. less than 7 number of states [5]

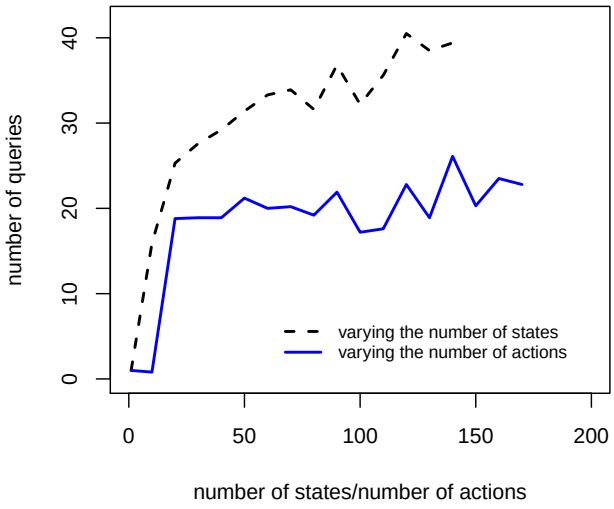

Fig. 4. Required number of queries when $d = 7$

B. Effectiveness of elicitation

In this section, we study our approach of regret-based reward elicitation on random MDPs. We set ϵ error of Algorithm IV equal 10^{-4} and compute the number of queries proposed to the user until minimax regret reduce to value of 0. In Figure 4, we observe changing of states and actions when $d = 7$. When sates change we fix $|A| = 5$, and set $|S| = 10$ during actions changing. The figure shows that query numbers increase linearly as a function of state parameter while it's changing w.r.t actions is approximately constant.
Weng and Zanuttini in [9] and Regan and Boutilier in [6] both have run their methods on a special size of MDP where $|A| = 6$ and $d = 14$ for $|S| \leq 500$. The former result with 46.5 queries is better than the latter result which is equal 60 queries. In comparison with our case, we have $|A| = 5$ and $d = 7$ in figure 4. As shown, the average required number of queries change linearly and doesn't reach 40 for less than 150 states. Moreover, we ran instances of the size $|A| = 5$ and $d = 10$. In this part, we could continue calculations until 70 states while queries had again linear form and didn't exceed 25. Our results are not strong enough to make any conclusion because of complexity of our calculations in reward elicitation. Then we can not fulfill calculations in a finite time for huge MDPs with value of $d \geq 10$. Also, we don't have any theoretical proof that shows this diagram will continue linearly for more number of states.

VI. Conclusion

In this work, we proposed a new structure on rewards in IRMDPs using [9]. Also we presented a robust policy computation method for this set of unknown rewards by using minimax regret criterion which is less complicated and computationally faster than previous works. Moreover, we proposed a reward elicitation approach in order to refine knowledge of rewards and reducing this regret with the goal of finding optimal policies. Since this work is preliminary, there will be many interesting subjects that we will work on in the future. One of them is defining non-dominated rewards and profit this definition in policy calculation and reward elicitation. Besides we will consider another type of structure on rewards, and present various types of queries in order to find more effective results.

References

[1] C. Boutilier, and R. Patrascu, and P. Poupart, and D. Schuurmans, *Constraint-based Optimization and Utility Elicitation using the Minimax Decision Criterion*. Artif. Intelligence, 170(8-9):686-713, 2006.

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

[3] K. Regan and C. Boutilier, *Regret-based Reward Elicitation for Markov Decision Processes* . NIPS-08 workshop on Model Uncertainty and Risk in Reinforcement Learning, 1, 2008.

[4] K. Regan and C. Boutilier, *Regret-based Reward Elicitation for Markov Decision Processes* . UAI-09 The 25th Conference on Uncertainty in Artificial Intelligence, 2009.

[5] K. Regan and C. Boutilier, *Robust Policy Computation in Reward-uncertain MDPs using Nondominated Policies* . Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010), 2010.

[6] K. Regan and C. Boutilier, *Robust Online Optimization of Reward-uncertain MDPs*. Twenty-Second Joint Conference on Artificial Intelligence (IJCAI 2011), 2011a.

[7] K. Regan and C. Boutilier, *Eliciting Additive Reward Functions for Markov Decision Processes*. Twenty-Second Joint Conference on Artificial Intelligence (IJCAI 2011), 2011b.

[8] P. Weng, *Markov Decision Processes with Ordinal Rewards: Reference Point-Based Preferences*. International Conference on Automated Planning and Scheduling, 21:282-289, 2011.

[9] P. Weng and B. Zanuttini, *Interactive Value Iteration for Markov Decision Processes with Unknown Rewards*. International Joint Conference on Artificial Intelligence, 2013.

[10] H. Xu and S. Mannor, *Parametric regret in uncertain Markov decision processes*. 48th IEEE Conference on Decision and Control,pages 3606-3613, 2009.