Deterministic Policies Based on Maximum Regrets in MDPs with Imprecise Rewards

Pegah Alizadeh [a], Emiliano Traversi [b] and Aomar Osmani [b]
[a] *Léonard de Vinci Pôle Universitaire, Research Center, 92 916 Paris, La Défense, France*
E-mail: pegah.alizadeh@devinci.fr
[b] *LIPN-UMR CNRS 7030, Université Sorbonne Paris Nord, Villetaneuse, France*
E-mails: emiliano.traversi@lipn.univ-paris13.fr, aomar.osmani@lipn.univ-paris13.fr

Abstract.
Markov Decision Process Models (MDPs) are a powerful tool for planning tasks and sequential decision-making issues. In this work we deal with MDPs with imprecise rewards, often used when dealing with situations where the data is uncertain. In this context, we provide algorithms for finding the policy that minimizes the maximum regret. To the best of our knowledge, all the regret-based methods proposed in the literature focus on providing an optimal stochastic policy. We introduce for the first time a method to calculate an optimal deterministic policy using optimization approaches. Deterministic policies are easily interpretable for users because for a given state they provide a unique choice. To better motivate the use of an exact procedure for finding a deterministic policy, we show some (theoretical and experimental) cases where the intuitive idea of using a deterministic policy obtained after "determinizing" the optimal stochastic policy leads to a policy far from the exact deterministic policy.

Keywords: Markov Decision Process, Minimax Regret, Unknown Rewards, Branch-and-Bound, Deterministic Policy, Stochastic Policy

1. Introduction

Markov processes are useful for modeling the stochastic environments in reinforcement learning frameworks. In recent years, attempts were made to apply methods from reinforcement learning and optimization approaches to construct decision support systems for action selection in stochastic environments with uncertain parameters. These uncertainties parameters can be unknown/partially-known rewards or transition functions. Although the conventional methods in planning under uncertainty suggest stochastic decision support systems, they can not be applied to automatic decision support systems such as those in robotics or autonomous vehicles. As they do not suggest an exact action in each sequence to the user and leave her with stochastic based selection system [1, 2]. These methods can not gain ground with users of such systems because the made decisions can not be easily interpreted by the users.

Typically, in MDPs the reward functions are estimated either from observations or external sources. Mannor et al. [3] demonstrate that the policy found via an optimization process under the MDPs with exact numerical parameters, sometimes can be much worse than the anticipated policy under bounded and imprecise reward values. This motivates using MDPs that account for this ambiguity in model parameters. To cope with this problem, *MDPs with imprecise rewards (IRMDP)* should be used. There exists several alternative models in the literature including symbolic-based rewards approaches [4, 5] and numerical-based rewards approaches [1, 2, 6].

One common approach in computing a robust solution is the *maximin* method, which computes a policy maximising the value with respect to the worst-case scenario [7–10]. *maximin* policies are conservative naturally [11], thus *minimax regret* approach [1, 2] has been introduced as solution to cope with this issue. The minimax robustness can be considered as a game between two adversaries, one finds a policy with maximum values while the adversary chooses an instantiation of the reward functions that minimize the expected value. There are some recent works that propose some techniques for dependent uncertainties in MDPs [12, 13]. In this paper, the uncertain reward functions are independent from each other.

Several methods in the past have only focused on computing *optimal stochastic policies* for IRMDPs optimization approaches [1, 2, 14].When it comes to deterministic policies, only heuristic algorithms are available in the literature. As an example, Ahmed et al. [15] proposed a mixed integer linear program for computing approximation of deterministic policies when a set of samples from uncertain rewards is given. In this paper we propose an exact approach receiving the set of whole uncertain rewards without any sampling. To the best of our knowledge, there is no work that handle deterministic policy computation on MDPs under uncertainties using exact optimization approaches. The existing works on deterministic policies computation deal usually with MDPs with precise parameters [16, 17]. To find the best *policy* (strategy), we use the *minimax regret criterion*. The basic idea is to find the policy with the minimum lost in comparison with other possible policies and reward instantiations. Minimizing the max regret is more optimistic than minimizing the worst case scenario and has been widely used in the literature.

The most of the exact and approximate methods for solving an MDP accept to have *stochastic policies* as feasible policies for the MDP. The use of stochastic policies presents two main advantages. From an algorithmic point of view, finding an optimal stochastic policy is usually easier than finding the optimal deterministic policy. Moreover, with stochastic policies the solution space increases, allowing to have optimal policies with a better value than the optimal deterministic policies. However, a deterministic policy is easier to understand from a user's point of view and therefore it is more likely to be used in practice. Finally, in several situations the nature of the problems does not allow any choices and requires a deterministic policy, this is due to either the discrete/combinatorial nature of the problem studied or to the fact that the algorithm must be executed only once, losing the relevance of the stochastic aspects.

In this paper, we introduce a first study of finding the deterministic policy that minimizes the maximum regret in an MDP with uncertain rewards. After giving some preliminaries concepts (see section 2) we present an exact enumerative scheme to find the optimal deterministic solution in section 3. Our method finds the best deterministic policy in a computing time that is relatively close to the one needed to compute the optimal stochastic policy. Section 4 presents some theoretical analysis of the optimal deterministic policy. Finally, section 5 reports an experimental study on random and diamond MDPs, in which we analyze the performances of our algorithms.

2. Preliminaries

Markov Decision Process. An *MDP* [18] is defined by a tuple $M(S, A, P, r, \gamma, \beta)$, where: S is a finite set of states; A is finite set of actions, $P : S \times A \times S \longrightarrow [0, 1]$ is a *transition function* where $P(s'|s, a)$ encodes the probability of going to state s' by being in state s, and choosing action a; $r : S \times A \longrightarrow \mathbb{R}$ is a *reward function* (or penalty, if negative) obtained by choosing action a in state s; $\gamma \in [0, 1[$ is the discount factor; and $\beta : S \longrightarrow [0, 1]$ is an *initial state distribution function* indicating probability of initiating in state s by $\beta(s)$.

A (stationary) *deterministic policy* is a function $\pi : S \longrightarrow A$, which takes action $\pi(s)$ when in state s. A (stationary) *stochastic policy* is a function $\tilde{\pi} : S \times A \longrightarrow [0, 1]$ which indicates with probability $\tilde{\pi}(s, a)$, action a is chosen in state s according to policy $\tilde{\pi}$. A policy $\tilde{\pi}$ induces a *visitation frequency function* $f^{\tilde{\pi}}$ where $f^{\tilde{\pi}}(s, a)$ is the total discounted frequency of being in state s and choosing action a under the policy $\tilde{\pi}$ (see [1] or Section 6.9 in [18]):

$$f^{\tilde{\pi}}(s,a) = \sum_{s' \in S} \beta(s') \sum_{t=0}^{\infty} \gamma^{t-1} P(S_t = s' | A_t = a, S_1 = s)$$

where the sum is taken over trajectories defined by $S_0 \sim \beta, A_t \sim \tilde{\pi}(S_t)$ and $S_{t+1} \sim P(.|S_t, A_t)$. The policy is computable from $f^{\tilde{\pi}}$, via

$$\tilde{\pi}(s,a) = \frac{f^{\tilde{\pi}}(s,a)}{\sum_{a'} f^{\tilde{\pi}}(s,a')} \ . \tag{1}$$

For a deterministic policy we have that $f^{\pi}(s, a) = 0$, $\forall a \neq \pi(s)$.

Policies are evaluated by expectation of discounted sum of rewards w.r.t to the infinite horizon discounted criterion, namely *value function* $V : S \longrightarrow \mathbb{R}$:

$$V^{\tilde{\pi}}(s) = \mathbb{E}(\sum_{t=0}^{\infty} \gamma^t r(s_t, \tilde{\pi}(s_t))).$$

Another way for defining the quality of policies is the *Q-value function* $Q : S \times A \longrightarrow \mathbb{R}$ given by:

$$Q^{\tilde{\pi}}(s,a) = r(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V^{\tilde{\pi}}(s') \ . \tag{2}$$

For a given initial state distribution β, the value of the optimal policy is $\beta \cdot V^{\tilde{\pi}}$, this quantity can be expressed in terms of the visitation frequency function (see [18]):

$$\beta \cdot V^{\tilde{\pi}} = r \cdot f^{\tilde{\pi}} \ . \tag{3}$$

An MDP always has an optimal policy π^* such that; $\pi^* = \text{argmax}_{\pi} \beta \cdot V^{\pi}$ or $f^* = \text{argmax}_{f} r \cdot f$, where the optimal policy can be recovered from f^* using Equation 1.

MDPs with Imprecise Rewards. When designing real cases as MDPs, specifying the reward function is generally a hard problem. For instance preferences stating which (state, action) pairs are good or bad should be interpreted into numerical costs. Note that even knowing all these preferences is time consuming. In order to tackle this complexity, we use MDPs with *imprecise reward values* (IRMDP). An IRMDP [1] is a tuple $M(S, A, P, \mathcal{R}, \gamma, \beta)$ where S, A, P, γ and β are defined as in the previous section, while $\mathcal{R}$ is a set of possible reward functions on $S \times A$. $\mathcal{R}$ models the uncertainty on the vector of real reward values r.

Similar to several previous works in the literature [1, 15, 19–22], we assume that the set of possible rewards is modelled as a polytope $\mathcal{R} = \{r : Cr \leqslant d\}$. The only real restriction we make concerning the uncertainty set is that we want it to be convex because it is well known that any convex set can be approximated with a polytope. For simplicity, in our experiments we work with a box as uncertainty set. However, our method is valid for any generic polytope $\mathcal{R}$.

Limited stochastic policies. A stochastic policy can potentially recommend many actions for each state. It is harder for a user, especially under time pressure, to select an action among a list of available actions suggested by the stochastic policy. One of the motivations of looking for a deterministic policy is precisely to have a solution that is easy to interpret. On the other hand, in this section we propose to improve the readability for the user by still allowing multiple possible actions per state but, at the same time, limiting the total number of choices in each state.

More formally, we define a (stationary) *limited stochastic policy* of cardinality k as a function $\tilde{\pi} : S \times A \longrightarrow [0, 1]$ which indicates with probability $\tilde{\pi}(s, a)$, action a is chosen in state s according to policy $\tilde{\pi}$. For each state s, we must have that $\tilde{\pi}(s, a) > 0$ for at most k actions, while $\tilde{\pi}(s, a) = 0$ for the others. It is clear that a limited stochastic policy of cardinality 1 is a deterministic policy, on the other hand. For completeness, we have that a limited stochastic policy of cardinality ∞ is a stochastic policy in the general sense.

Minimax Regret. In order to solve the IRMDP we use the *minimax regret criterion* (see [1, 2]).

The *regret* of policy f^{π} over reward function $r \in \mathcal{R}$ is the loss or difference in value between f and the optimal policy under r and is defined as

$$R(f^{\pi}, r) = \max_{g \in \mathcal{F}} r \cdot g - r \cdot f \ .$$

where $\mathcal{F}$ represents the set of all valid policies. The *maximum regret* for policy f^{π} is the maximum regret of this policy w.r.t the reward set $\mathcal{R}$:

$$MR(f^{\pi}, \mathcal{R}) = \max_{r \in \mathcal{R}} R(f^{\pi}, r) \ .$$

In other words, when we should select the f policy, what is the worst case loss over all possible rewards $\mathcal{R}$. Considering it as a game, the adversary tries to find a reward value in order to maximize our loss for the given f^{π}.

Finally we define the *minimax regret* of feasible reward set $\mathcal{R}$ as

$$MM(\mathcal{R}) = \min_{f^{\pi}} MR(f^{\pi}, r) \ .$$

Any policy f^* that minimizes the maximum regret is the *minimax-regret optimal policy* for the MDP. We recall that usually such optimal policies are considered stochastic and not deterministic. There are several approaches for computing the minimax regret [1, 2, 19, 20, 23]. In this paper, we use the approach presented by Regan and Boutilier [1] based on *Benders Decomposition* [24]. The idea is to formulate the problem as series of Linear Programs (LPs) and Mixed Integer Linear Programs (MILPs):

```
Master Program (MP)
```

$$\begin{aligned} &\text{minimize}_{\delta, f} && \delta && (4)\\ &\text{subject to:} && r \cdot g - r \cdot f \leqslant \delta \quad \forall \langle g_r, r \rangle \in \text{GEN} && (5)\\ &&& \gamma E^{\top} f + \beta = 0 && (6) \end{aligned}$$

```
Slave Program (SP)
```

$$\begin{aligned} &\text{maximize}_{Q,V,I,r} && \beta \cdot V - r \cdot f && (7)\\ &\text{subject to:} && Q_a = r_a + \gamma P_a V \quad \forall a \in A && (8)\\ &&& V \geqslant Q_a \quad \forall a \in A && (9)\\ &&& V \leqslant m(1 - I_a) M_a + Q_a \quad \forall a \in A && (10)\\ &&& Cr \leqslant d && (11) \end{aligned}$$

$$\sum_{a \in A} I_a = 1 \tag{12}$$

$$I_a(s) \in \{0,1\} \quad \forall s \in S,\ a \in A \tag{13}$$

$$M_a = M^{\top} - M_a^{\perp} \quad \forall a \in A \tag{14}$$

The Master Program (MP) is a Linear Program computing the minimum regret with respect to all the possible combinations of rewards and adversary policies.The set containing all the combinations of rewards and adversary policies is called GEN. In the first set of constraints, one constraint for each element $\langle g_r, r \rangle \in$ GEN is considered. The second set of constraints of the Master Program, $\gamma E^{\top} f + \beta = 0$ guaranties that f is a valid visitation frequency function. For the sake of abbreviation, the E matrix is generated according to the transition function P; E is a $|S||A| \times |S|$-matrix with a row for each state action, and one column for each state: $E_{sa,s'} = \begin{cases} P(s'|s,a) & \text{if } s' \neq s \\ P(s'|s,a) - \frac{1}{\gamma} & \text{if } s' = s \end{cases}$.

The intuition behind this constraint is related to the dual linear program of the Bellman Equation (see for example [25], Chapter 4 or [18], Section 6.9).

From a practical point of view, it is not convenient to enumerate a-priori all the constraints (5). Benders decomposition is based on the idea of starting with a small (maybe empty) subset of constraints (5) and interacting with the Slave Program to have either a certificate of optimality of the Master Program or a new inequality that can potentially change the value of the master.

The Slave Program receives a feasible policy f^* and searches for a policy and a reward value that maximize the regret of the given policy, in other words, it finds a r and g such that $r \cdot g - r \cdot f^* > \delta^*$. If such a (r, g) is found, it is added to GEN and the Master Program is solved again. If this is not the case, the procedure stops and f^* is the (stochastic) policy that minimizes the maximum regret.

The interaction between Master and Slave Programs can be viewed as a game between two players. The Master Program finds an optimal policy that minimizes the regret w.r.t the adversarial choices found so far by the Slave Program, while the Slave Program searches for an adversarial choice that gives the maximum gain against the current master policy. This game continues until the Slave Program can not find neither a policy as g nor a reward as r to generate a higher regret for the given f by the Master Program.

The Slave Program is a reformulation of the MR(f, $\mathcal{R}$) for the received policy f from the Master Program. According to equation (3), the objective function $r \cdot g - r \cdot f$ is rewritten as $\beta \cdot V - r \cdot f$. Constraint (8) ensures that equation (2) is satisfied and constraints (9) and (10) ensure that $Q(s,a) = V(s), \forall a$. For each a, we have that the constant M_a is equal to $M^{\top} - M^{\perp}$, where $M^{\top}$ is the value of the optimal policy for maximum reward values. In case of a box uncertainty set, we compute the optimal policy for $M(S, A, P, r_{\max}, \gamma, \beta)$, where $r_{\max}$ is the vector corresponding to the maximum values for each reward. This can be found using the classical methods such as value iteration or policy iteration [25]. Similarly $M^{\perp}$ is the Q-value for the optimal policy with the minimum rewards on $\mathcal{R}$.

I is a $|S| \times |A|$-matrix defining the policy related to V. Constraints (12) and (13) impose to have a deterministic policy, i.e., with one and only one selected action a per state s. Notice that the Slave Program proposes a deterministic adversary to the Master Program, while the Master Program always approximates a stochastic policy. Since the adversary policy proposes an extreme policy w.r.t the given f, a MILP model for the Slave Program is sufficient.

3. Finding an optimal deterministic (or limited stochastic) policy

We are now focusing on the description of an algorithm that can provide an optimal either deterministic or limited stochastic policy for an IRMDP. We use a branch-and-bound framework with the Benders decomposition as bounding procedure to achieve this goal. We show how to obtain a limited stochastic policy of cardinality k. To obtain a deterministic policy, it is sufficient to fix $k = 1$.

The branch-and-bound algorithm is one of the most popular and efficient algorithms for solving combinatorial problems for decades (see [26] section 11 for a full explanation of a generic version of the branch-and-bound algorithm). A branch-and-bound algorithm consists of a clever enumeration of the space of feasible policies through a space search: the set of limited stochastic policies that can potentially be the optimal is represented with a rooted tree with the full set associated to the root. The algorithm explores branches of this tree, where each branch represents a subset of the solution set. Once a new branch of the tree is created, and before that branches is split again in additional subbranches, a (lower) bounding procedure is

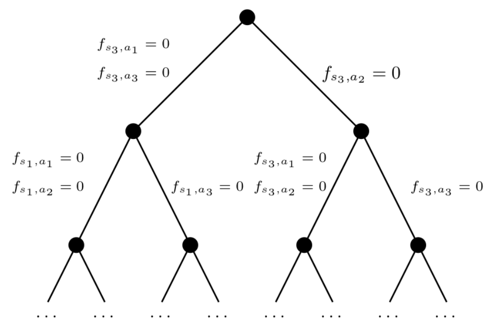

Fig. 1. Example of a branch-and-bound tree for an MDP with 4 states and 3 actions per state.

executed on that branch. The bounding procedure gives an underestimation of the optimal solution of the problem over the feasible set associated to the given branch. The branch is hence discarded if it cannot produce a better solution than the best one found so far by the algorithm.

In our case, the root of the branch-and-bound tree is associated to the full set of limited stochastic policies, while a branch is obtained by selecting a couple (s, a) of state and action and subsequently imposing the following disjunction on the two child nodes:

- $f_{s,a'} = 0, \forall a' \neq a$ for the “left” child node.
- $f_{s,a} = 0$ for the “right” child node.

The disjunctions impose to the left child represents only limited stochastic policies with $f_{s,a} \neq 0$ (i.e. $\pi(s, a) = 1$). On the other hand, the right child represents limited stochastic policies with $f_{s,a} = 0$ (i.e. $\pi(s, a) = 0$) [1]. Figure 1 presents an example of a branch-and-bound tree for an MDP with 4 states and 3 actions.

As already mentioned, to avoid exploring the whole tree, we need a lower bounding procedure to *prune* some of the nodes that do not contain the optimal policy. In our application, we use the optimal stochastic policy as under-estimator of the optimal limited stochastic policy for a given branch of the tree (we remind that we are looking for the policy with minimal value, for this reason the under-estimator can be viewed as an optimistic estimation of the policy). In this way, if a node has a stochastic policy higher than the best limited stochastic policy found so far it is not necessary to continue exploring that branch and the node can be pruned. Every time that a stochastic policy computed in the bounding procedures is also limited stochastic, its value can be used to update the value of the best known limited stochastic policy.

[1]The total number of choices (i.e., the number of state-action pairs) is finite, therefore also the size of the branch-and-bond tree is finite.

The final ingredient of a branch-and-bound is a procedure to find feasible limited stochastic policies. In our implementation, every time a stochastic policy computed in the bounding procedures is also limited stochastic, its value can be used to update the value of the best known limited stochastic policy.

In Figure 2 we show the pseudo-code of our implementation of the branch-and-bound algorithm. The algorithm starts by initializing the value of the best known limited stochastic policy to $+\infty$ and the list of unexplored nodes to the root node (i.e., the one with no constraints on the f variables). The while loop extracts one unexplored node from the list, fixes the f corresponding to its sub-region of feasible limited stochastic policies and computes a lower bound with Benders decomposition (if N is the empty set, then no f variable is fixed to zero). If the resulting optimal stochastic policy has a maximum regret δ^* greater or equal than the lower maximum regret found so far for a limited stochastic policy, no additional nodes are created and the loop extracts another node from the list. If the node is not pruned but the stochastic policy is also limited stochastic, the maximum regret of the best limited stochastic solution is updated to δ^*. As last option, if the stochastic solution is not limited stochastic, a state s with more than one f different from zero is found and the $f^*_{s,a}$ with the highest value is used to create the next two child nodes.

Cut-and-branch version of the algorithm. In the computational experiments, we test also a modification of the algorithm, called *cut-and-branch*. In this version of the algorithm, we decide to solve the root node of the branch-and-bound tree as usual. Once the algorithm starts to branch, additional Benders cuts are added only if the policy found by the Master Program is limited stochastic. In this way we are sure to compute correctly the value of the maximum regret of a limited stochastic solution. In Figure 2, the cut-and-branch option is activated if C&B=TRUE.

The advantage of the proposed approach is that the computing time needed to process a node is lower than the one needed by the basic version of the algorithm. On the other hand, the lower bounds obtained in the second case are weaker, this means that the total number of nodes explored can potentially be higher. In the computation section we show how the cut-and-branch version of the algorithm outperforms the basic implementation.

```
Algorithm   branch-and-bound search for an optimal limited stochastic policy of cardinality $k$:
$BestVal := +\infty$ /* best value fixed to infinity */
$\mathcal{N} = \{\{\emptyset\}\}$ /* the collection of open nodes is initialized with the empty set */
while $\mathcal{N}$ is not empty do
    extract node $N$ from $\mathcal{N}$
    for each $f$ in $N$ do:
        fix $f = 0$ in the MP
    if ($N = \{\emptyset\}$ or C&B=FALSE) then:
            solve the MP and generate additional cuts
    else:
            solve the MP
    $(\delta^*, f^*) :=$ optimal solution of the MP
    if $\delta^* < BestVal$ then:
    /* comparing the stoc. pol. with the best det. pol. */
        if $f^*$ is limited stochastic of cardinality $k$ then:
            $BestVal = \delta^*$ /* update best det. policy */
        else:    /* create the two child nodes */
            find $f^*_{s,a}$ of highest value among the ones associated to a non det. state
            $N_L := N \cup_{s' \neq s} f_{s',a}$, $N_R := N \cup f_{s,a}$
            $\mathcal{N} := \mathcal{N} \cup N_L \cup N_R$
```

Fig. 2. Algorithm to compute an optimal limited stochastic policy of cardinality k. If $k = 1$ we obtain an optimal deterministic policy. If C&B=TRUE, the cut-and-branch option is activated.

4. Theoretical analysis of the optimal deterministic policy

The goal of this section is to give a theoretical motivations of the importance of studying specific algorithms for finding an optimal deterministic policy.
We introduce the intuitive concept of *determinised policy*, a way to obtain a feasible deterministic policy starting from a stochastic policy. This procedure is based on the assumption that, if only one action is possible for each state, it is reasonable to assume that the most probable action should be chosen. This procedure is fast and it can be applied to any feasible stochastic policy. However, there exists no guarantee on the quality of the policy obtained, regardless of the quality of the stochastic policy used. In this section we theoretically show that using this "common sense" idea of adapting a stochastic policy can lead to solution significantly sub-optimal. In Section 5 we show how the computational results obtained are consistent with the findings of this section.

The determinised policy. Let $\tilde{f}$ be a given visitation frequency value for the optimal stochastic policy. The corresponding "rounding" deterministic policy $\hat{\pi}$ can be computed as follows:

- for each $s' \in S$:
  - ◇ find the action $a' = \text{argmax}_{a \in A} f_{s',a}$.
  - ◇ fix the rest of the action to zero: $\hat{f}_{s',a} = 0, \forall a \neq a'$
- recover the deterministic policy $\hat{\pi}$ obtained from the above fixing $\hat{f}$.

The approach computes the deterministic policy by selecting the action with the highest probability for each state. Despite being pretty simple, this approach represents a plausible behavior of a user that want to derive a deterministic policy starting from a stochastic one. From now on, we will name this rounding deterministic policy as "*deteminised policy*".

A small counterexample.. We define the *Trident IRMDP* (see Figure 3) as follows:

- Three states: s_0, s_1, s_2, three actions a_0, a_1, a_2 and a discout factor $\gamma = 1$.
- A transition function:
  $P(s_0|s_2, a_0) = 1$, $P(s_1|s_2, a_1) = 1$,
  $P(s_0|s_2, a_2) = T_0$ and $P(s_1|s_2, a_2) = T_1$.
- Two unknown rewards associated to s_0 and s_1: $r(s_0) = r_0 \in [-A, +A]$ and $r(s_1) = r_1 \in [-A + B, +A + B]$ with $A, B > 0$ and $A \gg B$. Thus, $\mathcal{R} = [-A, +A] \times [-A + B, A + B]$.[2]
- An initial distribution on states $\beta(s_0) = \beta(s_1) = 0$ and$\beta(s_2) = 1$.

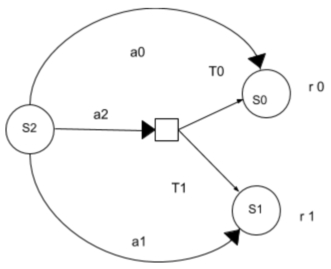

Fig. 3. Trident IRMDP with 3 states and 3 actions.

The following propositions give a complete characterization of the optimal stochastic and deterministic policies for the Trident MDP. With a slight abuse of notation, we use the subscript a instead of using s_2, a, i.e. instead of writing $\pi(s_2, a)$ we use π_a. We also use r_0 in

[2]in this formulation rewards are dependent on states. They can be easily modified to the reward function notation given in this paper $r(s, a)$

place of $r(s_0)$ and r_1 in place of $r(s_1)$. Each stochastic policy on the trident MDP can be demonstrated as a tuple $\pi = (\pi_0, \pi_1, \pi_2)$. Similarly, the visitation frequency functions are presented as $f = (f_0, f_1, f_2)$.

Proposition 1. *An optimal stochastic policy that minimizes the maximum regret (see Section 2) for the Trident MDP is the policy $\tilde{\pi} = (\pi_0, \pi_1, \pi_2)$ defined as:*

$$\pi_0 = \frac{2A - B}{4A}, \quad \pi_1 = \frac{2A + B}{4A}, \quad \pi_2 = 0 \; .$$

Proof. We first observe that for every policy $\pi' = (\pi'_0, \pi'_1, \pi'_2)$ with $\pi'_2 > 0$, it is possible to construct a policy $\pi'' = (\pi''_0, \pi''_1, \pi''_2)$ with $\pi''_2 = 0$ and with the same value in the following way:

$$\pi''_0 = \pi'_0 + \pi'_2 T_0, \quad \pi''_1 = \pi'_1 + \pi'_2 T_1 \; .$$

If we compute the value of the first policy we notice that:

$$\begin{aligned} \beta \cdot V^{\pi'} = V^{\pi'}(s_2) = \\ r_0 \pi'_0 + r_1 \pi'_1 + r_0 T_0 \pi'_2 + r_1 T_1 \pi'_2 = \\ r_0(\pi'_0 + T_0 \pi'_2) + r_1(\pi'_1 + T_1 \pi'_2) = r_0 \pi''_0 + r_1 \pi''_1 \\ = V^{\pi''}(s_2) = \beta \cdot V^{\pi''} \end{aligned}$$

showing that both policies have the same value. Moreover, the equivalence shows that $\beta \cdot V^{\pi'} = \beta \cdot V^{\pi''}$, $\forall r \in \mathcal{R}$. This implies that π' and π'' have equivalent maximum regret, because:

$$MR(\pi', \mathcal{R}) = \max_r \max_g r \cdot g - \beta \cdot V^{\pi'} =$$

$$\max_r \max_g r \cdot g - \beta \cdot V^{\pi''} = MR(\pi'', \mathcal{R}).$$

We can hence suppose that there exists an optimal stochastic policy with $\pi_2 = 0$ as a solution for minimax regret.

As second part of the proof, we compute the value of the optimal policy considering $\tilde{\pi} = (\pi_0, \pi_1, 0)$ where $\pi_0, \pi_1 \geqslant 0$ similarly its equivalent visitation frequency is $\tilde{f} = (f_0, f_1, 0)$. We notice that the adversary policy by its equivalent visitation frequency g giving a maximum regret is always deterministic (see Section 2). For this reason, we have only two adversary policies: we can have either $g = (g_0, g_1, g_2)$ where $g_0 = g_2 = 0$ and $g_1 > 0$ or the opposite, $g' = (g_0, g_1, g_2)$ where $g_0 > 0$ and $g_1 = g_2 = 0$. (With arguments analogous to the ones used in the first part of the proof, we can rule out the case where $g_2 \geqslant 0$.)

Knowing that the maximum regret is the maximum among two choices for the adversary policies, the maximum regret associated to the policy $g = (0, g_1, 0)$ (obtained by fixing $r_0 = -A$ and $r_1 = A + B$) is the following:

$$r \cdot g - r \cdot \tilde{f} = A + B + A\pi_0 - (A + B)\pi_1 \qquad (15)$$

and the maximum regret associated to the policy $g' = (g_0, 0, 0)$ where $g_0 > 0$ is obtained by fixing $r_0 = A$ and $r_1 = -A + B$, leading to a value of

$$r \cdot g - r \cdot \tilde{f} = A - A\pi_0 - (B - A)\pi_1 \; . \qquad (16)$$

We are interested in minimising the max regret, this means that we want to find the values of π_0 and π_1 that minimize $\max\{(15), (16)\}$. The optimal stochastic policy can hence be obtained by solving the following system of two equations:

$$\begin{cases} A + B + A\pi_0 - (A + B)\pi_1 = A - A\pi_0 - (B - A)\pi_1 \\ \pi_0 + \pi_1 \qquad\qquad\qquad\quad = 1 \end{cases}$$

It has as optimal solution the values $\pi_0 = \dfrac{2A - B}{4A}$ and $\pi_1 = \dfrac{2A + B}{4A}$, concluding the proof. $\square$

Proposition 1 implies the following Lemma:

Lemma 1. *The determinised policy (rounding policy) for the Trident MDP is $\hat{\pi} = (0, 1, 0)$ and its maximum regret is $MR(\hat{f}, \mathcal{R}) = 2A - B$.*

Proof. It is a direct consequence of the fact that in the optimal stochastic policy we always have $\pi_1 > \pi_2$ and $\pi_0 = 0$. $\square$

Proposition 2. *If $T_1 > T_0$, the optimal deterministic policy is $\pi^* = (0, 0, 1)$ and its maximum regret is $MR(f^*, \mathcal{R}) = A - AT_0 + (A - B)T_1$.*

Proof. We prove the statement by explicitly computing the maximum regret of the three possible deterministic policies: $\pi = (1, 0, 0), \pi' = (0, 1, 0)$ and $\pi'' = (0, 0, 1)$.

Maximum regret of $\pi = (1, 0, 0)$. We want to find the adversary policy that maximizes the regret for the policy π. We do that by computing all possible combinations of adversary policies and rewards:

- If a visitation frequency for adversary policy is $g = (0, g_1, 0)$ where $g_1 > 0$, the reward maximizing the regret is $r_0 = -A$ and $r_1 = A + B$, leading to a maximum regret of

$$A + B - (-A) = 2A + B \tag{17}$$

- If the adversary policy is $g' = (0, 0, g_2)$ where $g_2 > 0$, we need to check all four combinations of extreme rewards:
  - ◇ $r_0 = -A$ and $r_1 = A + B$. Maximum regret of

$$-AT_0 + (A + B)T_1 + A = (1 - T_0 + T_1)A + T_1 B \tag{18}$$

  - ◇ $r_0 = A$ and $r_1 = A + B$. Maximum regret of

$$AT_0 + (A + B)T_1 - A = (-1 + T_0 + T_1)A + T_1 B \tag{19}$$

  - ◇ $r_0 = A$ and $r_1 = -A + B$. Maximum regret of

$$AT_0 + (-A + B)T_1 - A = (-1 + T_0 - T_1)A + T_1 B \tag{20}$$

  - ◇ $r_0 = -A$ and $r_1 = -A + B$. Maximum regret of

$$-AT_0 + (-A + B)T_1 + A = (1 - T_0 - T_1)A - T_1 B \tag{21}$$

  By hypothesis we have that $A \gg B$ and $T_0 + T_1 = 1$, this implies that

$$(17) \geqslant \max\{(18), (19), (20), (21)\}.$$

  Therefore, the maximum regret if $g_2 > 0$ is $MR(f^{\pi}, \mathcal{R}) = 2A + B$.

Maximum regret of $\pi' = (0, 1, 0)$. It is trivial to check, with calculations analogous to the one used above to compute the regret of π, that the maximum regret in this case is equal to $MR(f^{\pi'}, \mathcal{R}) = 2A - B$.

Maximum regret of $\pi'' = (0, 0, 1)$. Also in this case, we need to consider the two cases of $g = (g_0, 0, 0)$ where $g_0 > 0$ and $g' = (0, g_1, 0)$ with $g_1 > 0$. For g, we fix $r_0 = A$ and $r_1 = -A + B$, obtaining a regret equal to

$$A - AT_0 + (A - B)T_1 \tag{22}$$

And for g' we fix $r_0 = -A$ and $r_1 = A + B$, obtaining a regret equal to

$$A + B + AT_0 - (A + B)T_1 \ . \tag{23}$$

The maximum between (22) and (23) depends on the values of T_0 and T_1. By imposing $A - AT_0 + (A - B)T_1 \geqslant A + B + AT_0 - (A + B)T_1$ we obtain:

$$2AT_1 \geqslant 2AT_0 + B \ .$$

We recall that by construction we have $A \gg B$, this implies that if $T_1 > T_0$ (resp. $T_1 \leqslant T_0$) we have that the maximum regret is equal to (22) (resp. (23)). The minimum maximum regret found so far is the one obtained for $\pi = \pi' = (0, 1, 0)$, and it is equal to $MR(f^{\pi'}, \mathcal{R}) = 2A - B$. Therefore, it remains to check for which values of $T_0 > T_1$ we have that $2A - B \geqslant$ (22):

$$A - AT_0 + (A - B)T_1 \leqslant 2A - B$$
$$\Leftrightarrow A - A(1 - T_1) + (A - B)T_1 \leqslant 2A - B$$
$$\Leftrightarrow (2A - B)T_1 \leqslant 2A - B \Leftrightarrow T1 \leqslant 1 \ .$$

Since we have by construction that $T_1 \leqslant 1$ we can conclude that for any $T_1 > T_0$ the optimal deterministic policy is $\pi* = \pi'' = (0, 0, 1)$ and its maximum regret is equal to $MR(f^{\pi''}, \mathcal{R}) = A - AT_0 + (A - B)T_1$. □

Proposition 2 and Lemma 1 show that for any Trident MDP we have that the optimal deterministic policy and the determinised policy (rounding policy) are always different.

The following Lemma shows that the rounding policy could be significantly worse than the optimal deterministic policy:

Lemma 2. *The ratio between the maximum regret of the determinised policy and the optimal deterministic policy goes to* 2 *with the increase of the value of A with respect to B and the increase of* T_1. *In other words:*

$$\lim_{A/B \to \infty, T_1 \to \frac{1}{2}^+} \frac{2A - B}{A - AT_0 + (A - B)T_1} = 2$$

Proof. The statement follows from the definition of the limit. □

From a theoretical point of view, it is still unknown what is the highest possible value we can have for the ratio between the maximum regret of the determinised policy and the optimal deterministic policy. From a practical point of view, such small example shows how the use of the determinised policy can lead to a maximum regret 100% far from the optimal. One possible intuition for explaining this high value of the ratio is that the number of states that can be reached with an executed action are limited. For this reason, in Section 5 we introduce a new set of random MDPs, called *limited*, with the mentioned property. In this way, it is possible to experimentally verify if the intuition is correct in practice.

5. Experimental results

In this section, we provide an experimental evaluation of our algorithms based on three classes of test instances: random MDPs with unlimited connections among states, random MDPs with limited connections among states and diamond MPDs. The aim of this section is twofold: we first investigate how in practice the optimal deterministic policy is different from the determinised policy obtained from the optimal stochastic policy (We recall that we compare our results against the determinised policies because it reflects what a normal system user would do when forced to obtain a deterministic policy from a stochastic one). Secondly, we show how the new cut-and-branch version of the algorithm helps to solve faster the considered instances.

5.1. Instance description

Unlimited Random MDPs. A random MDP with unlimited connections (`Random -unlim` MDP) is defined by a given number of states $|S|$ and actions $|A|$. The rewards are bounded between two real random values uniformly selected in the intervall $[-1.0, 1.0]$. The transition function has the following properties: from any state s we restrict transitions to reach $\lceil \log_2(n) \rceil$ number of next states. For each pair of (s, a) we draw reachable states based on uniform distribution over the set of states. For drawn states, the transition probabilities are formed based on Gaussian distribution. The initial state distribution β is uniform and we choose a discount factor $\gamma = 0.95$. In our tests we use instances with $|S| \in \{10, 15\}$ and $|A| \in \{2, 3, 4, 5, 10\}$. The instances with $|S| = 15$ and $|A| = 10$ are excluded because their solution time exceeds the time limit of 24 hours.

Random MDPs, limited connections (`Random-lim`). These instances are inspired by the Trident IRMDP presented in Section 4 (See Figure 3). A random-lim MDP is a random MDP where the number of next states that can be reached with a given action are limited in comparison with random-unlimit MDPs. A random-lim MDP is identified by its number of states $|S|$ and a fixed number of reachable states n. The rewards are bounded between two real random values uniformly selected in the interval $[-1.0, 1.0]$. The transition function has the following properties: from any state s the number of reachable states S' is equal to n, uniformly selected over the set of states. The number of actions depends on the value of n. First, we define n actions, where each of them reaches only a single state from S'. Secondly, each of the remaining actions reaches all the possible pairs of states from S'. Therefore, we have a total of $n + \frac{n(n-1)}{2}$ actions. The initial state distribution β is uniform and we choose a discount factor $\gamma = 0.95$. In our tests we use instances with $|S| \in \{5, 6, 7, 8, 9, 10\}$ and $|A| \in \{3, 6\}$.

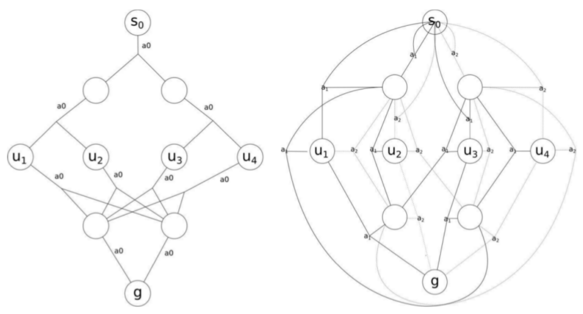

Fig. 4. Diamond MDP: actions a_0 (left) and a_1, a_2 (right) (Figure taken from [20]).

Diamond MDPs (`Diamond`). This class of MDPs has been introduced for the first time in Benavent and Zanuttini [20].

This class of MDPs has a diamond structure, with one top and one bottom state (playing the role of start and terminal states in the MDP), one intermediate layer of states, containing all the uncertainties on rewards, plus two intermediate layers between the extreme states and the intermediate layer.

The diamond MDP structure is given in Figure 4. In diamond MDP, each action a_0 has probability 0.5 to reach each child node. On the other hand a_1 (resp. a_2) has a probability of p (resp. $1 - p$) to reach the left (resp. right) child node and to reach its parent otherwise. The imprecise values of the rewards for the middle layer are $[-600, 600]$, while the one of the bottom node is $[600, 1000]$ (see [20] for more details on the structures). We propose a generalization of this family of MDP by testing a range of parameters for the probability $p \in \{0.05, 0.10, \ldots, 0.40, 0.45\}$.

|S|	10					15			
|A|	2	3	4	5	10	2	3	4	5
VR_{DD}, Avg.	1.11	1.15	1.04	1.06	1.01	1.03	1.05	1.02	1.03
VR_{DD}, Max.	1.27	1.78	1.13	1.18	1.03	1.15	1.13	1.06	1.04
VR_{DS}, Avg.	0.74	0.80	0.86	0.86	0.91	0.78	0.84	0.85	0.87
VR_{DS}, Max.	0.81	0.86	0.97	0.93	0.94	0.84	0.96	0.88	0.88

Table 1

Value Ratio for `Random-unlim` MDPs.

|S|	5		6		7		8		9		10	
|A|	3	6	3	6	3	6	3	6	3	6	3	6
VR_{DD}, Avg.	1.10	1.11	1.23	1.09	1.11	1.55	1.42	1.20	1.17	1.11	1.07	1.14
VR_{DD}, Max.	1.29	1.35	1.68	1.14	1.45	2.76	1.87	1.96	1.74	1.29	1.21	1.33
VR_{DS}, Avg.	0.83	0.87	0.83	0.87	0.84	0.88	0.86	0.85	0.84	0.91	0.91	0.94
VR_{DS}, Max.	0.90	0.94	0.91	0.97	0.92	0.98	0.93	0.98	0.93	0.98	0.98	0.98

Table 2

Value Ratio for `Random-lim` MDPs.

p	5	10	15	20	25	30	35	40	45
VR_{DD}	1.66	1.24	1.16	1.13	1.15	1.15	1.15	1.14	1.16
VR_{DS}	0.94	0.91	0.89	0.88	0.88	0.87	0.83	0.82	0.83

Table 3

Value Ratio for `Diamond`.

5.2. *Comparison with the determinised policy*

For a given MDP, let $MR(f^{\hat{\pi}}, \mathcal{R})$ be the maximum regret of the determinised (rounding deterministic) policy, $MR(f^{\pi^*}, \mathcal{R})$ be the maximum regret of the optimal deterministic policy and $MR(f^{\tilde{\pi}}, \mathcal{R})$ be the maximum regret of the optimal stochastic policy. We define the Deterministic-Determinised (DD) *Value Ratio* of such MDPs as: $VR_{DD} = \dfrac{MR(f^{\hat{\pi}}, \mathcal{R})}{MR(f^{\pi^*}, \mathcal{R})}$. Moreover, we define the Deterministic-Stochastic (DS) *Value Ratio* of such MDPs as: $VR_{DS} = \dfrac{MR(f^{\tilde{\pi}}, \mathcal{R})}{MR(f^{\pi^*}, \mathcal{R})}$. The VR_{DD} gives an idea about how far is the determinised policy from the optimal policy[3]. For example, a VR_{DD} of 1.20 means that the rounding deterministic policy gives a value that is 20% worse in comparison with the optimal deterministic policy.

[3] It is a deterministic policy.

In Tables 1, 2 and 3 we present the Value Ratios for the different classes of instances considered. In case of multiple instances for each combination of settings, we report the average and the maximum values. We first notice that the `Random-lim` and `Diamond` MDPs have higher VR_{DD} in comparison to `Random-unlim` MDPs. The explanation for this difference is that, for `Random-lim` and `Diamond` MDPs, the set of states that can be reached with a give action is considerably smaller in comparison to that state that can be reached with an action in a `Random-unlim` MDP. Generally speaking, a stochastic policy can easily allow to "spread" the choice from a state to the different next states by allowing fractional values of π. On the other side, a deterministic policy allows to move only to a few states (the ones reached by the single action selected). The higher is the difference between the optimal deterministic and stochastic policy, the higher is the probability of taking a suboptimal choice when using the determinising procedure. With `Random-unlim` MDPs, each action leads to $\lceil \log_2(n) \rceil$ states (while for the other two classes of MDPs each action leads to two states). Therefore, for `Random-unlim` MDPs, even a deterministic policy can visit a significant amount of states. The relative small value of VR_{DD} for the `Random-unlim` MDPs is probably due to the mentioned behaviour: for `Random-unlim` MDPs the stochastic policy is usually sufficiently close to the deterministic policy to allow a good approximation when determinising it.

As second observation, we see that the maximum values of VR_{DD} are quite high for almost all the combinations of parameters considered. Furthermore, for a `Random-lim` instance with 7 states and 6 actions we find a VR_{DD} of 2.76, showing that the theoretical worse case showed in Section 4 can be increased. Therefore, choosing to determinise may turn out to be an unsafe choice, specially considering that it is not possible to check how far a determinised policy is from the optimal. We finally notice that the computing time for obtaining the optimal deterministic policy is on average six time bigger than the time needed to obtain the optimal stochastic policy.

5.3. *Impact of the cut-and-branch improvement*

In figures 5 and 6 we show how the use of the cut-and-branch presented at the end of Section 3 improves the computing times on the `Random` MDPs. For each couple of states-actions we show the average

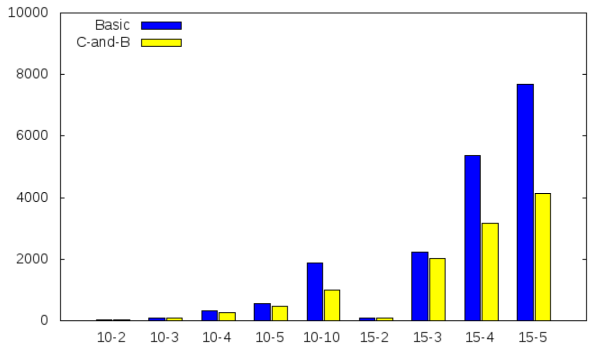

Fig. 5. Impact of cut-and-branch, `Random-unlim` MDPs (states-actions vs computing time, in seconds).

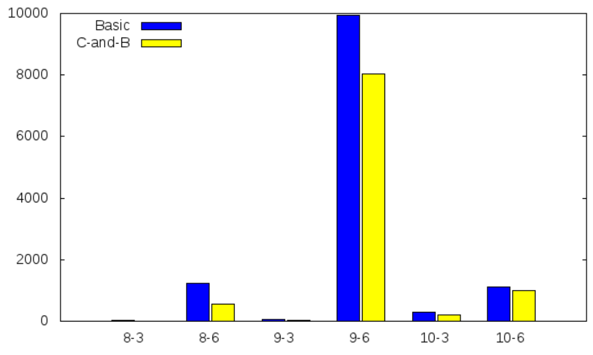

Fig. 6. Impact of cut-and-branch, `Random-lim` MDPs (states-actions vs computing time, in seconds).

computing times of the baseline implementation of the Branch-and-bound and of the cut-and-branch version.

The use of the cut-and-branch option allows to decrease the computing times for all the instances considered. This increase is particularly clear for the instances of `Random-unlim` with 10 or 15 nodes, where the computing time can be reduced by almost one half on the more difficult instances.

5.4. Limited stochastic policy

In Figure 7 we show how the value of the optimal policy decreases when the maximum number of allowed actions increases. Each curve represents a `Random-lim` MDP with 6 actions and 4 to 7 states. On the horizontal axis we have the maximum number of actions allowed in the optimal policy; a value equal to 1 corresponds to the case of a deterministic policy and the value equal to 6 corresponds to the stochastic policy. Vertically we show the difference between the value of the optimal policy for a given number of allowed actions and the optimal stochastic policy. We normalize the results by dividing them by the maximum gap (for this reason we always have a gap of 100% for the deterministic policy and a gap of 0% for the stochastic policy). We report the results for `Random-lim` MDPs while we experimentally observed that the behavior is the same for all the studied instances. We were able to obtain such results by a generalization of our branch-and-bound algorithms.

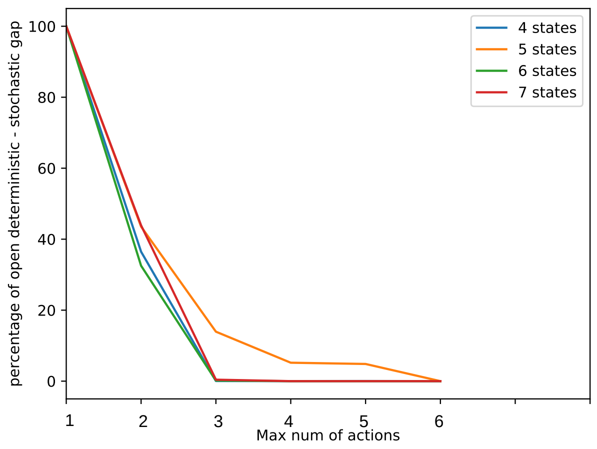

Fig. 7. Change in the value of the optimal policy for a given maximum amount of actions per state. For each MDP instance, there are in total 6 actions per state. The x axis determines possible limited number of actions per state ($1 =$ deterministic policy, $6 =$ stochastic policy).

It is important to notice that allowing to have a maximum of two possible actions per state leads to a policy with a maximum regret significantly closer to the optimal stochastic policy. These results suggest that an interesting trade-off is to allow a maximum of two or three actions per state. In this way the policy remains easy to interpret for the user as the value of the maximum regret is only marginally worse than the one of the optimal stochastic policy. If a future theoretical study confirms our observations for certain categories of applications, it will open a concrete perspective to the use of “determinization”. It leaves reasonable choices to the user while keeping a quality of the policy close to the optimal.

6. Conclusions

We presented for the first time in the literature an algorithm to find an optimal deterministic policy that minimizes the maximum regret of a Markov Decision Processes (MDP) with imprecise rewards. The proposed algorithm consists of a branch-and-bound that uses Benders decomposition as bounding procedure. In addition to a basic implementation, we propose a cut-and-branch implementation that turns out to reduce the overall computing time by up to 50%. We motivate the use of deterministic over stochastic policies by showing theoretically that basic rounding procedures find deterministic policies far from the optimal. Secondly, we show that the additional computational effort of computing the optimal deterministic policy in comparison to the one needed to compute the optimal stochastic policy is acceptable (approximately one order of magnitude slower).

References

[1] K. Regan and C. Boutilier, Regret-based Reward Elicitation for Markov Decision Processes, in: *UAI*, AUAI Press, 2009, pp. 444–451.

[2] H. Xu and S. Mannor, Parametric regret in uncertain Markov decision processes, in: *CDC*, IEEE, 2009, pp. 3606–3613.

[3] S. Mannor, D. Simester, P. Sun and J.N. Tsitsiklis, Bias and Variance Approximation in Value Function Estimates, *Management Science* **53**(2) (2007), 308–322.

[4] 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) (2012), 123–156.

[5] P. Weng, Ordinal Decision Models for Markov Decision Processes, in: *ECAI*, Frontiers in Artificial Intelligence and Applications, Vol. 242, IOS Press, 2012, pp. 828–833.

[6] D.E. Bell, Regret in Decision Making under Uncertainty, *Operations Research* **30**(5) (1982), 961–981.

[7] R. Givan, S. Leach and T. Dean, Bounded-parameter Markov decision processes, *Artificial Intelligence* **122**(1) (2000), 71–109.

[8] G.N. Iyengar, Robust Dynamic Programming, *Mathematics of Operations Research* **30**(2) (2005), 257–280.

[9] A. Mastin and P. Jaillet, Loss bounds for uncertain transition probabilities in Markov decision processes (2012), 6708–6715.

[10] A. Nilim and L.E. Ghaoui, Robust Control of Markov Decision Processes with Uncertain Transition Matrices, *Operations Research* **53**(5) (2005), 780–798.

[11] E. Delage and S. Mannor, Percentile Optimization in Uncertain Markov Decision Processes with Application to Efficient Exploration, in: *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, ACM, New York, NY, USA, 2007, pp. 225–232.

[12] S. Mannor, O. Mebel and H. Xu, Lightning Does Not Strike Twice: Robust MDPs with Coupled Uncertainty, in: *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, 2012, pp. 451–458.

[13] W. Wiesemann, D. Kuhn and B. Rustem, Robust Markov Decision Processes, *Mathematics of Operations Research* **38**(1) (2013), 153–183.

[14] K. Regan and C. Boutilier, Robust Policy Computation in Reward-Uncertain MDPs Using Nondominated Policies, in: *AAAI*, AAAI Press, 2010.

[15] A. Ahmed, P. Varakantham, M. Lowalekar, Y. Adulyasak and P. Jaillet, Sampling Based Approaches for Minimizing Regret in Uncertain Markov Decision Processes (MDPs), *J. Artif. Intell. Res.* **59** (2017), 229–264.

[16] D. Dolgov and E. Durfee, Stationary Deterministic Policies for Constrained MDPs with Multiple Rewards, Costs, and Discount Factors, in: *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, IJCAI'05, Morgan Kaufmann Publishers Inc., 2005, pp. 1326–1331.

[17] G. Montúfar, K. Ghazi-Zahedi and N. Ay, Geometry and Determinism of Optimal Stationary Control in Partially Observable Markov Decision Processes, *CoRR* **abs/1503.07206** (2015).

[18] M.L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*, 1st edn, John Wiley & Sons, Inc., New York, NY, USA, 1994. ISBN 0471619779.

[19] P. Alizadeh, Y. Chevaleyre and J. Zucker, Approximate regret based elicitation in Markov decision process, in: *RIVF*, IEEE, 2015, pp. 47–52.

[20] F. Benavent and B. Zanuttini, An Experimental Study of Advice in Sequential Decision-Making under Uncertainty, in: *32nd AAAI Conference on Artificial Intelligence*, 2018.

[21] P. Alizadeh, Y. Chevaleyre and F. Lévy, Advantage based value iteration for Markov decision processes with unknown rewards, in: *2016 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2016, pp. 3837–3844.

[22] P. Weng and B. Zanuttini, Interactive Value Iteration for Markov Decision Processes with Unknown Rewards, IJCAI/AAAI, 2013, pp. 2415–2421.

[23] V.F. da Silva and A.H.R. Costa, A Geometric Approach to Find Nondominated Policies to Imprecise Reward MDPs, in: *Proceedings of the 2011 European Conference on Machine Learning and Knowledge Discovery in Databases - Volume Part I*, ECML PKDD'11, Springer-Verlag, Berlin, Heidelberg, 2011, pp. 439–454.

[24] J.F. Benders, Partitioning Procedures for Solving Mixed-variables Programming Problems, *Numer. Math.* **4**(1) (1962), 238–252.

[25] R.S. Sutton and A.G. Barto, *Introduction to Reinforcement Learning*, 1st edn, MIT Press, Cambridge, MA, USA, 1998. ISBN 0262193981.

[26] D. Bertsimas and R. Weismantel, *Optimization Over Integers*, Dynamic Ideas, 2005. ISBN 9780975914625.