Efficient Policies for Stationary Possibilistic Markov Decision Processes

Nahla Ben Amor [1], Zeineb EL khalfi [1,2], Hélène Fargier [2] and Régis Sabaddin [3]

[1] LARODEC,Le Bardo, Tunisie, email: nahla.benamor@gmx.fr, zeineb.khalfi@gmail.com
[2] IRIT, Toulouse, France, email: fargier@irit.fr
[3] INRA-MIAT, Toulouse, France, email: regis.sabbadin@inra.fr

Abstract. Possibilistic Markov Decision Processes offer a compact and tractable way to represent and solve problems of sequential decision under qualitative uncertainty. Even though appealing for its ability to handle qualitative problems, this model suffers from the *drowning effect* that is inherent to possibilistic decision theory. The present paper proposes to escape the drowning effect by extending to stationary possibilistic MDPs the lexicographic preference relations defined in [6] for non-sequential decision problems and provides a value iteration algorithm to compute policies that are optimal for these new criteria.

Keywords: Markov Decision process, Possibility theory, lexicographic comparisons, possibilistic qualitative utilities

1 Introduction

The classical paradigm for sequential decision making under uncertainty is the one of expected utility-based *Markov Decision Processes* (MDP) [11, 2], which assumes that the uncertain effects of actions can be represented by probability distributions and that utilities are additive. But the EU model is not tailored to problems where uncertainty and preferences are ordinal in essence. Alternatives to the EU-based model have been proposed to handle ordinal preferences/uncertainty. Remaining within the probabilistic, quantitative, framework while considering ordinal preferences has lead to *quantile-based* approaches [17, 15, 8, 18, 9]) Purely ordinal approaches to sequential decision under uncertainty have also been considered. In particular, possibilistic MDPs [13, 12, 4, 1] form a purely qualitative decision model with an ordinal evaluation of plausibility and preference. In this model, uncertainty about the consequences of actions is represented by possibility distributions and utilities are also ordinal. The decision criteria are either the optimistic qualitative utility or its pessimistic counterpart [5]. However, it is now well known that possibilistic decision criteria suffer from the *drowning effect* [6]. Plausible enough bad or good consequences may completely blur the comparison between policies, that would otherwise be clearly differentiable. [6] have proposed *lexicographic refinements* of possibilistic criteria for the one-step decision case, in order to remediate the drowning effect. In this paper, we propose an extension of the lexicographic preference relations to stationary possibilistic MDPs.

The next Section recalls the background about possibilistic MDPs, including the drowning effect problem. Section 3 studies the lexicographic comparison of policies in

finite horizon problems and presents a value iteration algorithm for the computation of lexi-optimal policies. Section 4 extends these results to the infinite-horizon case. Lastly, Section 5 reports experimental results. Proofs are omitted, but can be found in[1].

2 Possibilistic Markov decision process

2.1 Definition

A possibilistic Markov Decision Process (P-MDP) [12] is defined by:

- A finite set S of **states**.
- A finite set A of **actions**, A_s denotes the set of actions available in state s;
- A possibilistic transition function: each action $a \in A_s$ applied in state $s \in S$ is assigned a possibility distribution $\pi(.|s,a)$;
- A utility function μ: $\mu(s)$ is the intermediate satisfaction degree obtained in state s.

The uncertainty about the effect of an action a taken in state s is a possibility distribution $\pi(.|s,a) : S \to L$, where L is a qualitative ordered scale used to evaluate both possibilities and utilities (typically, and without loss of generality, $L = [0,1]$): for any s', $\pi(s'|s,a)$ measures to what extent s' is a plausible consequence of a when executed in s and $\mu(s')$ is the utility of being in state s'. In the present paper, we consider stationary problems, i.e. problems in which states, the actions and the transition functions do not depend on the stage of the problem. Such a possibilistic MDP defines a graph, where states are represented by circles and are labelled by utility degrees and actions are represented by squares. An edge linking an action to a state denotes a possible transition and is labeled by the possibility of that state given the action is executed.

Example 1. Let us suppose that a "Rich and Unknown" person runs a startup company. Initially, s/he must choose between Saving money (Sav) or Advertising (Adv) and may then get Rich (R) or Poor (P) and Famous (F) or Unknown (U). In the other states, Sav is the only possible action. Figure 1 shows the stationary P-MDP that captures this problem, formally described as follows: $S = \{RU, RF, PU\}, A_{RU} = \{Adv, Sav\}, \quad A_{RF} = \{Sav\}, A_{PU} = \{Sav\}, \pi(PU|RU, Sav) = 0.2, \pi(RU|RU, Sav) = 1; \pi(RF|RU, Adv) = 1; \pi(RF|RF, Sav) = 1, \pi(RU|RF, Sav) = 1, \mu(RU) = 0.5, \mu(RF) = 0.7, \mu(PU) = 0.3.$

Solving a stationary MDP consists in finding a (stationary) policy, i.e. a function $\delta : S \to A_s$ which is optimal with respect to a decision criterion. In the possibilistic case, as in the probabilistic case, the value of a policy depends on the utility and on the likelihood of its trajectories. Formally, let Δ be the set of all policies encoded by a P-MDP. When the horizon is finite, each $\delta \in \Delta$ defines a list of scenarios called trajectories. Each trajectory is a sequence of states and actions $\tau = (s_0, a_0, s_1, \ldots, s_{E-1}, a_{E-1}, s_E)$.

To simplify notations, we will associate the vector $v_\tau = (\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{E-1}, \mu_E)$ to each trajectory τ, where $\pi_{i+1} =_{def} \pi(s_{i+1}|s_i, a_i)$ and $\mu_i =_{def} \mu(s_i)$.

The possibility and the utility of τ given that δ is applied from s_0 are defined by:

$$\pi(\tau|s_0, \delta) = \min_{i=1..E} \pi(s_i|s_{i-1}, \delta(s_{i-1})) \quad \text{and} \quad \mu(\tau) = \min_{i=0..E} \mu(s_i) \tag{1}$$

[1] https://www.irit.fr/publis/ADRIA/PapersFargier/XKRU17MDP.pdf

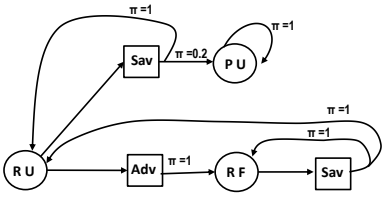

Fig. 1. A possibilistic stationary MDP

Two criteria, an optimistic and a pessimistic one, can then be used [5][13]:

$$u_{opt}(\delta, s_0) = \max_{\tau} \min\{\pi(\tau|s_0, \delta), \mu(\tau)\} \tag{2}$$

$$u_{pes}(\delta, s_0) = \max_{\tau} \min\{1 - \pi(\tau|s_0, \delta), \mu(\tau)\} \tag{3}$$

These criteria can be optimized by choosing, for each state, an action that maximizes the following counterparts of the Bellman equations [12]:

$$u_{opt}(s) = \max_{a \in A_s} \min\{\mu(s), \max_{s' \in S} \min(\pi(s'|s, a), u_{opt}(s'))\} \tag{4}$$

$$u_{pes}(s) = \max_{a \in A_s} \min\{\mu(s), \min_{s' \in S} \max(1 - \pi(s'|s, a), u_{pes}(s'))\} \tag{5}$$

This formulation is more general than the first one in the sense that it applies to both the finite and the infinite case. It has allowed the definition of a (possibilistic) *value iteration* algorithm which converges to an optimal policy in polytime $O(|S|^2 \cdot |A|^2 \cdot |L|)$ [12]. This algorithm proceeds by iterated modifications of a possibilistic value function $\tilde{Q}(s, a)$ which evaluates the "utility" (pessimistic or optimistic) of performing a in s.

2.2 The drowning effect

Unfortunately, possibilistic utilities suffer from an important drawback called the *drowning effect*: plausible enough bad or good consequences may completely blur the comparison between acts that would otherwise be clearly differentiated; as a consequence, an optimal policy δ is not necessarily Pareto eficient - it may exist a policy δ' such that $u_{pes}(\delta'_s) = u_{pes}(\delta_s)$ while $\forall s, u_{pes}(\delta'_s) \succeq u_{pes}(\delta_s)$ and (ii) $\exists s, u_{pes}(\delta'_s) \succ u_{pes}(\delta_s)$ where δ_s (resp. δ'_s) is the restriction of δ (resp. δ') to the subtree rooted in s.

Example 2. The P-MDP of Example 1; it admits two policies δ and δ': $\delta(RU) = Sav; \delta(PU) = Stay; \delta(RF) = Sav; \delta'(RU) = Adv; \delta'(PU) = Stay; \delta'(RF) = Sav$. For horizon $E = 2$:

- δ has 3 trajectories: $\tau_1 = (RU, PU, PU)$ with $v_{\tau_1} = (0.5\ 0.2\ 0.3\ 1\ 0.3)$; $\tau_2 = (RU, RU, PU)$ with $v_{\tau_2} = (0.5\ 1\ 0.5\ 0.2\ 0.3)$; $\tau_3 = (RU, RU, RU)$ with $v_{\tau_3} = (0.5\ 1\ 0.5\ 1\ 0.5)$.
- δ' has 2 trajectories: $\tau_4 = (RU, RF, RF)$ with $v_{\tau_4} = (0.5\ 1\ 0.7\ 1\ 0.7)$; $\tau_5 = (RU, RF, RU)$ with $v_{\tau_5} = (0.5\ 1\ 0.7\ 1\ 0.5)$.

Thus $U_{opt}(\delta) = U_{opt}(\delta') = 0.5$. However δ' seems better than δ since it provides utility 0.5 for sure while δ provides a bad utility (0.3) in some non impossible trajectories (τ_1 and τ_2). τ_3 which is good and totally possible "drowns" τ_1 and τ_2: δ is considered as good as δ'.

2.3 Lexi-refinements of ordinal aggregations

In ordinal (i.e. min-based and max-based) aggregation a solution to the drowning effect has been proposed, that is based on leximin and leximax comparisons [10]. It has then been extended to non-sequential decision making under uncertainty [6] and, in the sequential case, to decision trees [3]. Let us first recall the basic definition of these two preference relations. For any two vectors t and t' of length m built on L:

$$t \succeq_{lmin} t' \text{ iff } \forall i, t_{\sigma(i)} = t'_{\sigma(i)} \text{ or } \exists i^*, \forall i < i^*, t_{\sigma(i)} = t'_{\sigma(i)} \text{ and } t_{\sigma(i^*)} > t'_{\sigma(i^*)} \quad (6)$$

$$t \succeq_{lmax} t' \text{ iff } \forall i, t_{\mu(i)} = t'_{\mu(i)} \text{ or } \exists i^*, \forall i < i^*, t_{\mu(i)} = t'_{\mu(i)} \text{ and } t_{\mu(i^*)} > t'_{\mu(i^*)} \quad (7)$$

where, for any vector v (here, $v = t$ or $v = t'$), $v_{\mu(i)}$ (resp. $v_{\sigma(i)}$) is the i^{th} best (resp. worst) element of v.

[6] have extended these procedures to the comparison of matrices built on L. Given a complete preorder $\trianglerighteq$ on vectors, it is possible to order the lines of the matrices (say, A and B) according to $\trianglerighteq$ and to apply an $lmax$ or an $lmin$ procedure:

$$A \succeq_{lmin(\trianglerighteq)} B \Leftrightarrow \forall j, a_{(\trianglerighteq,j)} \cong b_{(\trianglerighteq,j)} \text{ or } \exists i \text{ s.t. } \forall j > i, a_{(\trianglerighteq,j)} \cong b_{(\trianglerighteq,j)} \text{ and } a_{(\trianglerighteq,i)} \rhd b_{(\trianglerighteq,i)} \quad (8)$$

$$A \succeq_{lmax(\trianglerighteq)} B \Leftrightarrow \forall j, a_{(\trianglerighteq,j)} \cong b_{(\trianglerighteq,j)} \text{ or } \exists i \text{ s.t.} \forall j < i, a_{(\trianglerighteq,j)} \cong b_{(\trianglerighteq,j)} \text{and } a_{(\trianglerighteq,i)} \rhd b_{(\trianglerighteq,i)} \quad (9)$$

where, for any $\boldsymbol{c} \in (L^M)^N$, $c_{(\trianglerighteq,i)}$ is the i^{th} largest sub-vector of $\boldsymbol{c}$ according to $\trianglerighteq$.

3 Lexicographic-value iteration for finite horizon P-MDPs

In (finite-horizon) possibilistic decision trees, the idea of [3] is to identify a strategy with the matrix of its trajectories, and to compare such matrices with a $\succeq_{lmax(lmin)}$ (resp. $\succeq_{lmin(lmax)}$) procedure for the optimistic (resp. pessimistic) case. We propose, in the following, a value iteration algorithm for the computation of such lexi-optimal policies in the finite (this Section) and infinite (Section 4) horizon cases.

3.1 Lexicographic comparisons of policies

Let E be the horizon of the P-MDP. A trajectory being a sequence of states and actions, a strategy can be viewed as a matrix where each line corresponds to a distinct trajectory. In the optimistic case each line corresponds to a vector $v_\tau = (\mu_0, \pi_1, \mu_1, \pi_2, \ldots, \pi_{E-1}, \mu_E)$ and in the pessimistic case to $w_\tau = (\mu_0, 1-\pi_1, \mu_1, 1-\pi_2, \ldots, 1-\pi_{E-1}, \mu_E)$.

This allow us to define the comparison of trajectories and strategies by [2]:

$$\tau \succeq_{lmin} \tau' \text{ iff } (\mu_0, \pi_1, \ldots, \pi_E, \mu_E) \succeq_{lmin} (\mu'_0, \pi'_2, \ldots, \pi'_E, \mu'_E) \quad (10)$$

$$\tau \succeq_{lmax} \tau' \text{ iff } (\mu_0, 1-\pi_1, \ldots, 1-\pi_E, \mu_E) \succeq_{lmax} (\mu'_0, 1-\pi'_1, \ldots 1-\pi'_E, \mu'_E) \quad (11)$$

$$\delta \succeq_{lmax(lmin)} \delta' \text{ iff } \forall i,\ \tau_{\mu(i)} \sim_{lmin} \tau'_{\mu(i)}$$
$$or\ \exists i^*,\ \forall i < i^*, \tau_{\mu(i)} \sim_{lmin} \tau'_{\mu(i)}\ \ and\ \ \tau_{\mu(i^*)} \succ_{lmin} \tau'_{\mu(i^*)} \quad (12)$$

[2] If a trajectory is shorter than E, neutral elements (0 for the optimistic case and 1 for the pessimistic one) are added at the end. If the policies have different numbers of trajectories, neutral trajectories (vectors) are added to the shortest one.

$$\delta \succeq_{lmin(lmax)} \delta' \text{ iff } \forall i,\ \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$$
$$or\ \exists i^*,\ \forall i < i^*, \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}\ \ and\ \ \tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)} \quad (13)$$

where $\tau_{\mu(i)}$ (resp. $\tau'_{\mu(i)}$) is the i^{th} best trajectory of δ (resp δ') according to $\succeq_{lmin}$ and $\tau_{\sigma(i)}$ (resp. $\tau'_{\sigma(i)}$) is the i^{th} worst trajectory of δ (resp δ') according to $\succeq_{lmax}$.

It is easy to show that we get efficient refinements of u_{opt} and u_{pes}.

Proposition 1. *If $u_{opt}(\delta) > u_{opt}(\delta')$ (resp. $u_{pes}(\delta) > u_{pes}(\delta')$) then $\delta \succ_{lmax(lmin)} \delta'$ (resp. $\delta \succ_{lmin(lmax)} \delta'$).*

Proposition 2. *Relations $\succeq_{lmin(lmax)}$ and $\succeq_{lmax(lmin)}$ are complete, transitive and satisfy the principle of strict monotonicity*[3].

Remark. We define the *complementary* MDP, $(S, A, \pi, \bar{\mu})$ of a given P-MDP (S, A, π, μ) where $\bar{\mu}(s) = 1 - \mu(s), \forall s \in S$. The complementary MDP simply gives complementary utilities. From the definitions of $\succeq_{lmax}$ and $\succeq_{lmin}$, we can check that:

Proposition 3. *$\tau \succeq_{lmax} \tau' \Leftrightarrow \bar{\tau}' \succeq_{lmin} \bar{\tau}$ and $\delta \succeq_{lmin(lmax)} \delta' \Leftrightarrow \bar{\delta}' \succeq_{lmax(lmin)} \bar{\delta}$.*

where $\bar{\tau}$ and $\bar{\delta}$ are obtained by replacing μ with $\bar{\mu}$ in the trajectory/P-MDP.

Therefore, all results which we will prove in the following for $\succeq_{lmax(lmin)}$ also hold for $\succeq_{lmin(lmax)}$, if we take care to apply them to complementary strategies. Since considering $\succeq_{lmax(lmin)}$ involves less cumbersome expressions (no $1 - \cdot$), we will give the results for this criterion. Moreover, abusing notations slightly, we identify trajectories τ (resp. strategies) with their v_τ vectors (resp. matrices of v_τ vectors).

3.2 Basic operations on matrices of trajectories

Before going further, we define some basic operations on matrices (typically, on $U(s)$ representing trajectories issued from s). For any matrix $U = (u_{ij})$ with n lines and m columns,$[U]_{l,c}$ denotes the restriction of U to its first l lines and first c columns.

Composition, $U \times (N_1, \ldots, N_a)$: Let U be a $a \times b$ matrix and $N_1, \ldots, N_a$ be a series of a matrices of dimension $n_i \times c$ (they all share the same number of columns). The composition of U with $(N_1, \ldots, N_a)$ denoted $U \times (N_1, \ldots, N_a)$ is a matrix of dimension $(\sum_{1 \leq i \leq a} n_i) \times (b + c)$. For any $i \leq a, j \leq n_j$, the $(\Sigma_{i'<i} n_{i'}) + j)^{th}$ line of $U \times (N_1, \ldots, N_a)$ is the concatenation of the i^{th} line of U and the j^{th} line of N_i. The composition of $U \times (N_1, \ldots, N_a)$ is done in $O(n \cdot m)$ operations, where $n = \sum_{1 \leq i \leq a} n_i$ and $m = b + c$. The matrix $U(s)$ is typically the concatenation of the matrix $U = ((\pi(s'|s, a), \mu(s')), s' \in succ(s, a))$ with the matrices $N_{s'} = U(s')$.

[3] A criterion O satisfies the principle of strict monotonicity iff: $\forall \delta, \delta', \delta'',\ \delta \succeq_O \delta' \iff \delta + \delta'' \succeq_O \delta' + \delta''$. $\delta + \delta''$ contains two disjoint sets of trajectories: the ones of δ and the ones of δ'' (and similarly for $\delta' + \delta''$). Then, adding or removing identical trajectories to two sets of trajectories does not change their comparison by $\succeq_{lmax(lmin)}$ (resp. $\succeq_{lmin(lmax)}$) - while it may transform a strict preference into an indifference if u_{opt} (resp. u_{pes}) were used.

Ordering matrices $U^{lmaxlmin}$: Let U be a $n \times m$ matrix, $U^{lmaxlmin}$ is the matrix obtained by ordering the elements of the lines of U in increasing order and the lines of U according to $lmax$ (in decreasing order). The complexity of the operation depends on the sorting algorithm: if we use QuickSort then ordering the elements within a line is performed in $O(m \cdot log(m))$, and the inter-ranking of the lines is done in $O(n \cdot log(n) \cdot m)$ operations. Hence, the overall complexity in $O(n \cdot m \cdot \log(n \cdot m))$.

Comparison of ordered matrices : Given two ordered matrices $U^{lmaxlmin}$ and $V^{lmaxlmin}$, we say that $U^{lmaxlmin} > V^{lmaxlmin}$ iff $\exists i, j$ such that $\forall i' < i, \forall j'$, $U^{lmaxlmin}_{i',j'} = V^{lmaxlmin}_{i',j'}$ and $\forall j' < j$, $U^{lmaxlmin}_{i,j'} = V^{lmaxlmin}_{i,j'}$ and $U^{lmaxlmin}_{i,j} > V^{lmaxlmin}_{i,j}$.
$U^{lmaxlmin} \sim V^{lmaxlmin}$ iff they are identical (comparison complexity: $O(n \cdot m)$).

3.3 Lexicographic-value iteration

In this section, we propose a value iteration algorithm (Algorithm 1 for the $lmax(lmin)$ variant; the $lmin(lmax)$ variant is similar) that computes a lexicographic optimal policy in a finite number of iterations. This algorithm is an iterative procedure that updates the utility of each state, represented by a finite matrix of trajectories, using the utilities of the neighboring states, until a halting condition is reached. At stage t, the procedure updates the utility of every states $s \in S$ as follows:

- For each $a \in A_s$, a matrix $Q(s, a)$ is built which evaluates the "utility" of performing a in s at stage t: this is done by combining $TU_{s,a}$ (comparison of the transition matrix $T_{s,a} = \pi(\cdot|s, a)$ and the utilities $\mu(s')$ of the states s' that may follows s when a is executed) with the matrices $U^{t-1}(s')$ of trajectories provided by these s'. The matrix $Q(s, a)$ is then ordered (the operation is made less complex by the fact that the matrices $U^{t-1}(s')$ have been ordered at $t - 1$).
- The $lmax(lmin)$ comparison is performed on the fly to memorize the best $Q(s, a)$
- The value of s at t, $U^t(s)$, is the one given by the action $\delta^t(s) = a$ which provides the best $Q(s, a)$. U^t and δ^t are memorized (and U^{t-1} can be forgotten).

Proposition 4. *lmax(lmin)-Value iteration provides an optimal solution for* $\succeq_{lmaxlmin}$.

Time and space complexities of this algorithm are nevertheless expensive, since it eventually memorizes all the trajectories. At each step t its size may be about $b^t \cdot (2 \cdot t + 1)$, where b is the maximal number of possible successors of an action; the overall complexity of the algorithm is $O(|S| \cdot |A| \cdot |E| \cdot b^E)$, which is problematic. Notice now that, at any stage t and for any state s $[U^t(s)]_{1,1}$ (i.e. the top left value in $U^t(s)$) is precisely equal to $u_{opt}(s)$ at horizon t for the optimal strategy. We have seen that making the choices on this basis is not discriminant enough. On the other hand, taking the whole matrix is discriminant, but exponentially costly. Hence the idea of considering more than one line and one column, but less than the whole matrix - namely the first l lines and c columns of $U^t(s)^{lmaxlmin}$; hence the definition of the following preference:

$$\delta \geq_{lmaxlmin,l,c} \delta' \text{ iff } [\delta^{lmaxlmin}]_{l,c} \geq [\delta'^{lmaxlmin}]_{l,c} \tag{14}$$

$\geq_{lmaxlmin,1,1}$ corresponds to $\succeq_{opt}$ and $\geq_{lmaxlmin,+\infty,+\infty}$ corresponds to $\geq_{lmaxlmin}$.

Algorithm 1: Lmax(lmin)-value iteration

Data: A possibilistic MDP and an horizon E
δ^*, the policy built by the algorithm, is a global variable
1 `// δ a global variable starts as an empty set`
Result: Computes and returns δ^* for MDP
2 **begin**
3 $t \leftarrow 0$;
4 **foreach** $s \in S$ **do** $U^t(s) \leftarrow ((\mu(s)))$;
5 **foreach** $s \in S$, $a \in A_s$ **do** $TU_{s,a} \leftarrow T_{s,a} \times ((\mu(s')), s' \in succ(s,a))$;
6 **repeat**
7 $t \leftarrow t+1$;
8 **foreach** $s \in S$ **do**
9 $Q^* \leftarrow ((0))$;
10 **foreach** $a \in A$ **do**
11 $Future \leftarrow (U^{t-1}(s'), s' \in succ(s,a))$; `// Gather the matrices provided by the successors of s;`
12 $Q(s,a) \leftarrow (TU_{s,a} \times Future)^{lmaxlmin}$;
13 **if** $Q^* \leq_{lmaxlmin} Q(s,a)$ **then** $Q^* \leftarrow Q(s,a)$; $\delta^t(s) \leftarrow a$;
14 $U^t(s) \leftarrow Q^*(s, \delta^t(s))$
15 **until** $t == E$;
16 $\delta^*(s) \leftarrow argmax_a Q(s,a)$
17 **return** δ^*;

The combinatorial explosion is due to the number of lines (because at finite horizon, the number of columns is bounded by $2 \cdot E + 1$), hence we shall bound the number of considered lines. The following proposition shows that this approach is sound:

Proposition 5. *For any* l, c $\delta \succ_{opt} \delta' \Rightarrow \delta \succ_{lmaxlmin,l,c} \delta'$.
For any l, c, l' *such that* $l' > l$, $\delta \succ_{lmaxlmin,l,c} \delta' \Rightarrow \delta \succ_{lmaxlmin,l',c} \delta'$.

Hence $\succ_{lmaxlmin,l,c}$ refines u_{opt} and the order over the strategies is refined for a fixed c when l increases. It tends to $\succ_{lmaxlmin}$ when $c = 2.E + 1$ and l tends to b^E.

Up to this point, the comparison by $\geq_{lmaxlmin,l,c}$ is made on the basis of the first l lines and c columns of the *full* matrices of trajectories. This does obviously not reduce their size. The important following Proposition allows us to make the l, c reduction of the ordered matrices *at each step* (after each composition), and not only at the very end, thus keeping space and time complexities polynomial.

Proposition 6. *Let* U *be a* $a \times b$ *matrix and* $N_1, \ldots, N_a$ *be a series of* a *matrices of dimension* $a_i \times c$ *. It holds that:*
$[(U \times (N_1, \ldots, N_a))^{lmaxlmin}]_{l,c} = [(U \times ([N_1^{lmaxlmin}]_{l,c}, \ldots, [N_a^{lmaxlmin}]_{l,c}))^{lmaxlmin)}]_{l,c}$.

In summary, the idea of our Algorithm, that we call bounded lexicographic-value iteration (BL-VI) is to compute policies that are close to lexi-optimality, by keeping a sub matrix of each current value matrix - namely the first l lines and c columns. The algorithm is obtained by replacing line 12 of Algorithm 1, with:

$$Line\ 12' \ :\ Q(s,a) \leftarrow [(TU_{s,a} \times Future)^{lmaxlmin}]_{l,c};$$

Proposition 7. *Bounded lmax(lmin)-Value iteration provides a solution that is optimal for* $\succeq_{lmaxlmin,l,c}$ *and its time complexity is* $O(|E| \cdot |S| \cdot |A| \cdot (l \cdot c) \cdot b \cdot log(l \cdot c \cdot b))$.

In summary, this algorithm provides in polytime a strategy that is always as least as good as the one provided by u_{opt} (according to $lmax(lmin)$) and tends to lexi optimality when $c = 2 \cdot E + 1$ and l tends to b^E.

4 Lexicogaphic-value iteration for infinite horizon P-MDPs

In the infinite-horizon case, the comparison of matrices of trajectories by equations (12) or (13) may not be enough to rank-order the policies. The length of the trajectories may be infinite, and their number infinite as well. This problem is well known in classical probabilistic MDP where a discount factor is used that attenuates the influence of later utility degrees - thus allowing the convergence of the algorithm [11]. On the contrary, classical P-MDPs do not need any discount factor and Value Iteration, based on the evaluation for $l = c = 1$, converges for infinite horizon P-MDPs [12].

In a sense, this limitation to $l = c = 1$ plays the role of a discount factor - which is too drastic; it is nevertheless possible to make the comparison using $\geq_{lmaxlmin,l,c}$. Let us denote $U^t(s)$ the matrix issued from s at horizon t when δ is executed. It holds that:

Proposition 8. $\forall l, c, \exists t$ *such that, forall* $t' > t$, $(U^t)^{lmaxlmin}_{l,c}(s) = (U^{t'})^{lmaxlmin}_{l,c}(s)$.

This means that from a given stage t, the value of a strategy is stable if computed with the bounded lmax(lmin) criterion. This criterion can thus be soundly used in the infinite-horizon case and bounded value iteration converges. To adapt the algorithm to the infinite case, we simply need to modify the halting condition at line 15 by:

Line 15' : **until** $(U^t)^{lmaxlmin}_{l,c} == \left(U^{t-1}\right)^{lmaxlmin}_{l,c}$.

Proposition 9. *Whatever* l, c*, Lmax(lmin)-Bounded Value iteration converges for infinite horizon P-MDPs.*

Proposition 10. *The overall complexity of Bounded lmax(lmin)-Value iteration algorithm is* $O(|L| \cdot |S| \cdot |A| \cdot (l \cdot c) \cdot b \cdot log(l \cdot c \cdot b))$.

5 Experiments

We now compare the performance of Bounded lexicographic value iteration $(BL\text{-}VI)$ as an approximation of (unbounded) lexicographic value iteration $(UL\text{-}VI)$, in the Lmax(lmin) variant. The two algorithms have been implemented in Java and the experiments have been performed on an Intel Core i5 processor computer (1.70 GHz) with 8GB DDR3L of RAM. We evaluate the performance of the algorithms by carrying out simulations on randomly generated P-MDPs with $|S| = 25$. The number of actions in each state is equal to 4. The output of each action is a distribution on two states randomly fired (i.e. the branching factor is equal to 2). The utility values are uniformly randomly fired in the set $L = \{0.1, 0.3, 0.5, 0.7, 1\}$. Conditional possibilities relative to decisions should be normalized. To this end, one choice is fixed to possibility degree 1

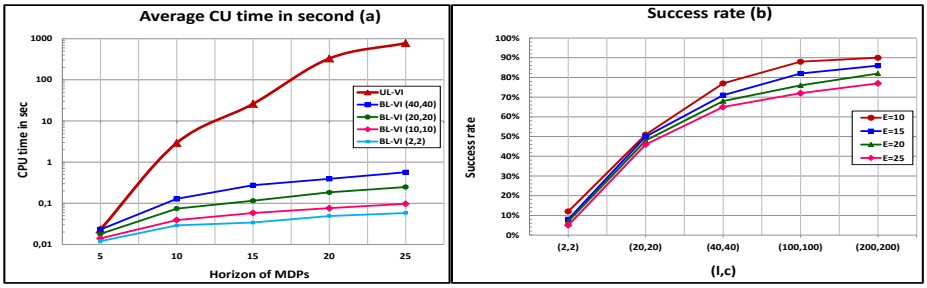

Fig. 2. Bounded lexicographic value iteration VS Unbounded lexicographic value iteration

and the possibility degree of the other one is uniformly fired in L. For each experience, 100 P-MDPs are generated. The two algorithms are compared w.r.t. 2 measures: (i) CPU time and (ii) Pairwise success rate: $Success$, the percentage of optimal solutions provided by Bounded value iteration with fixed (l, c) w.r.t. the lmax(lmin) criterion in its full generality. The higher $Success$, the more important the effectiveness of cutting matrices with $BL\text{-}VI$; the lower this rate, the more important the drowning effect.

Figure 2 presents the average execution CPU time for the two algorithms. Obviously, for both $UL\text{-}VI$ and $BL\text{-}VI$, the execution time increases with the horizon. Also, we observe that the CPU time of $BL\text{-}VI$ increases according to the values of (l, c) but it remains affordable, as the maximal CPU time is lower than $1s$ for MDPs with 25 states and 4 actions when $(l, c) = (40, 40)$ and $E = 25$. Unsurprisingly, we can check that the $BL\text{-}VI$ (regardless of the values of (l, c)) is faster than $UL\text{-}VI$ especially when the horizon increases: the manipulation of l, c-matrices is obviously less expensive than the one of full matrices. The saving increases with the horizon.

As with the success rate, the results are described in Figure 2. It appears that $BL\text{-}VI$ provides a very good approximation especially when increasing (l, c). It provides the same optimal solution as the $UL\text{-}VI$ in about 90% of cases, with an $(l, c) = (200, 200)$. Moreover, even when the success rate of $BL\text{-}VI$ decreases (when E increases), the quality of approximation is still good: never less than 70% of optimal actions returned, with $E = 25$. These experiments conclude in favor of bounded value iteration: its approximated solutions are comparable in terms of quality for high (l, c) and increase when (l, c) increase, while it is much faster than the unbounded version.

6 Conclusion

In this paper, we have extended to possibilistic Markov Decision Processes the lexicographic refinement of possibilitic utilities initially introduced in [6] for non-sequential problems. It can be shown that our approach is more discriminant than the refinement of binary possibilistic utility [16] since the latter does not satisfy strict monotonicity. Our lexicographic refinements criteria allowed us to propose a *Lmax(lmin)-Value Iteration* algorithm for stationary P-MDPs with two variants: (i) an unbounded version that converges in the finite horizon case, but is unsuitable for infinite-horizon P-MDPs,

since it generates matrices which size continuously increases with the horizon and (ii) a bounded version which has polynomial complexity. It bounds the size of the saved matrices and refines the possibilistic criteria, whatever the choice of the bounds. The convergence of this algorithm is shown for both the finite and the infinite horizon cases, and its efficiency has been observed experimentally even for low bounds.

There are two natural perspectives to this work. First, as far as the infinite horizon case is concerned, other types of lexicographic refinements could be proposed. One of these options could be to avoid the duplication of the set of transitions that occur several times in a single trajectory and consider only those which are observed. A second perspective of this work will be to define *reinforcement learning* [14] type algorithms for P-MDPs. Such algorithms would use samplings of the trajectories instead of full dynamic programming or quantile-based reinforcement learning approaches [7].

References

1. K. Bauters, W. Liu, and L. Godo. Anytime algorithms for solving possibilistic mdps and hybrid mdps. In *Proc FoIKS'2016*, pages 24–41, 2016.
2. R. Bellman. A Markovian decision process. *J. of Mathematics and Mechanics*, 6, 1957.
3. N. Ben Amor, Z. El Khalfi, H. Fargier, and R. Sabbadin. Lexicographic refinements in possibilistic decision trees. In *Proc ECAI'16*, pages 202–208, 2016.
4. N. Drougard, F. Teichteil-Konigsbuch, J.L. Farges, and D. Dubois. Qualitative possibilistic mixed-observable mdps. In *Proc UAI'13*, pages 192–201, 2013.
5. D. Dubois and H. Prade. Possibility theory as a basis for qualitative decision theory. In *Proc IJCAI'95*, pages 1925–1930, 1995.
6. H. Fargier and R. Sabbadin. Qualitative decision under uncertainty: back to expected utility. *Artificial Intelligence*, 164:245–280, 2005.
7. H. Gilbert and P. Weng. Quantile reinforcement learning. In *Proc JMLR'2016*, pages 1–16, 2016.
8. H. Gilbert, P. Weng, and Y. Xu. Optimizing quantiles in preference-based markov decision processes. In *Proc AAAI'2017*, 2017.
9. I. Montes, E. Miranda, and S. Montes. Decision making with imprecise probabilities and utilities by means of statistical preference and stochastic dominance. *European Journal of Operational Research*, 234(1):209–220, 2014.
10. H. Moulin. *Axioms of Cooperative Decision Making*. Cambridge University Press, 1988.
11. M.L. Puterman. *Markov Decision Processes*. John Wiley and Sons, 1994.
12. R. Sabbadin. Possibilistic Markov decision processes. *Engineering Applications of Artificial Intelligence*, 14:287–300, 2001.
13. R. Sabbadin, H. Fargier, and J. Lang. Towards qualitative approaches to multi-stage decision making. *International Journal of Approximate Reasoning*, 19:441–471, 1998.
14. R.S. Sutton and A.G. Barto. *Introduction to Reinforcement Learning*. MIT Press, 1998.
15. B. Szörényi, R. Busa-Fekete, P. Weng, and Eyke Hüllermeier. Qualitative multi-armed bandits: A quantile-based approach. In *Proc ICML'2015*, pages 1660–1668, 2015.
16. P. Weng. Qualitative decision making under possibilistic uncertainty: Toward more discriminating criteria. In *21st Conference in Uncertainty in Artificial Intelligence (UAI'05), July 26-29, Edinburgh, Scotland*, pages 615–622, 2005.
17. P. Weng. Markov decision processes with ordinal rewards: Reference point-based preferences. In *Proc ICAPS'2011*, 2011.
18. Y. Yue, J. Broder, R. Kleinberg, and T. Joachims. The k-armed dueling bandits problem. *Journal of Computer and System Sciences*, 78(5):1538–1556, 2012.