Lexicographic refinements in possibilistic decision trees and finite-horizon Markov decision processes

Nahla Ben Amor [a,*], Zeineb El Khalfi [a,b,**], Hélène Fargier [b,*], Régis Sabbadin [c,*]

[a] *LARODEC, University of Tunis, Tunisia*
[b] *IRIT, UPS-CNRS, 118 route de Narbonne, 31062 Toulouse, France*
[c] *MIAT, UR 875, Université de Toulouse, INRA, F-31320 Castanet-Tolosan, France*

Abstract

Possibilistic decision theory has been proposed twenty years ago and has had several extensions since then. Even though appealing for its ability to handle qualitative decision problems, possibilistic decision theory suffers from an important drawback. Qualitative possibilistic utility criteria compare acts through min and max operators, which leads to a drowning effect. To overcome this lack of decision power of the theory, several refinements have been proposed. Lexicographic refinements are particularly appealing since they allow to benefit from the Expected Utility background, while remaining qualitative. This article aims at extending lexicographic refinements to sequential decision problems i.e., to possibilistic decision trees and possibilistic Markov decision processes, when the horizon is finite. We present two criteria that refine qualitative possibilistic utilities and provide dynamic programming algorithms for calculating lexicographically optimal policies.

Keywords: Possibilistic decision trees; Markov decision processes; Possibilistic qualitative utilities; Lexicographic comparisons

1. Introduction

For many years, there has been an interest in the Artificial Intelligence community towards the foundations and computational methods of decision making under uncertainty (see e.g. [1–5]). The usual paradigm of decision under uncertainty is based on the *Expected Utility (EU) model* [6,7]. Its extensions to sequential decision making are *Decision Trees* (DT) [8] and *Markov Decision Processes* (MDP) [9], which assume 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 nature.

* Corresponding authors.
** Principal corresponding author.
E-mail addresses: nahla.benamor@gmx.com (N. Ben Amor), zeineb.khalfi@gmail.com (Z. El Khalfi), fargier@irit.fr (H. Fargier), regis.sabbadin@inra.fr (R. Sabbadin).

https://doi.org/10.1016/j.fss.2018.02.006

Alternatives to the EU-based model have been proposed to handle ordinal preferences/uncertainty. Considering ordinal preferences but remaining within a probabilistic quantification of uncertainty has led to quantile-based approaches [10–12], to the use of reference points [13] or to approaches by pairwise comparison [14]. Purely ordinal approaches to decision under uncertainty have been considered by [15–20]. In particular, possibilistic DTs and possibilistic MDPs (see [1,21–25]) use a common ordinal scale to model both the preferences and the uncertainty about the consequences of actions. These two framework rely on the qualitative decision theory (i.e. possibility theory) proposed and axiomatized by Dubois and Prade [20,17] — the decision criteria are either the optimistic qualitative utility or its pessimistic counterpart. However, it is now well known that possibilistic decision criteria suffer from the *drowning effect*. Acts (and policies in sequential problems, i.e., sets of conditional or unconditional decisions) are compared through min and max operators, which implies that plausible enough bad or good consequences may blur the comparison between acts that would otherwise be clearly differentiable.

In order to overcome the drowning effect, Fargier and Sabbadin have proposed *lexicographic refinements* of possibilistic criteria for non-sequential decision problems [26]. However, these refinements have not been extended yet to sequential decision under uncertainty, where the drowning effect is also due to the reduction of compound possibilistic policies into simple possibility distributions on the consequences. The present paper[1] proposes an extension of the lexicographic preference relations to finite horizon sequential problem, providing lexicographic possibilistic decision criteria that compare full policies (and not simply their reductions). This allow us to equip possibilistic decision trees and finite horizon Markov decision processes with backward induction algorithms that compute lexicographically optimal policies.

The paper is structured as follows: Section 2 presents possibilistic decision trees and finite-horizon possibilistic Markov decision processes in more details and highlights drowning effect. In Section 3, we define lexicographic orderings that refine the possibilistic decision criteria. Then, Section 4 proposes a dynamic programming algorithm for the computation of lexicographically optimal policies in possibilistic decision trees and in finite-horizon Markov decision processes. Section 5 shows that the lexicographic criteria proposed can be represented by expected utilities based on big-stepped probabilities/utilities. Finally, Section 6 presents experimental results.

2. Background and problematic

2.1. A short reminder on possibility theory

In decision-making problems, several types of uncertainty should be considered. Most of available decision models refer to probability theory for the representation of uncertainty [6,29]. Despite its success, probability theory is appropriate when all numerical informations are available or can be easily elicited. When information about uncertainty cannot be quantified in a probabilistic way, several non-classical theories of uncertainty can be considered in order to deal with imperfect, ordinal information namely, fuzzy sets theory [30], evidence theory [31] and possibility theory [32,33] etc. In what follows, we will focus on possibility theory the fundamental purely ordinal uncertainty theory.

Possibility theory, issued from fuzzy sets theory, was developed by Dubois and Prade [33]. The basic component of possibility theory is the notion of possibility distribution. It is a representation of a state of knowledge of an agent regarding the state of the world. A possibility distribution is denoted by π and it is a mapping from the universe of discourse Ω to a bounded linearly ordered scale L exemplified by the unit interval $[0, 1]$, we denote the function by: $\pi: \Omega \rightarrow [0, 1]$.

For vector $\omega \in \Omega$, $\pi(\omega) = 1$ means that realization ω is totally possible and $\pi(\omega) = 0$ means that ω is an impossible state. It is generally assumed that there exist at least one state ω which is totally possible: π is said then to be *normalized*.

In the possibilistic framework, extreme forms of partial knowledge can be captured, namely:

- Complete knowledge i.e. $\exists\, \omega_0$ s.t. $\pi(\omega_0) = 1$ and $\forall\, \omega \neq \omega_0, \pi(\omega) = 0$.
- Total ignorance i.e. $\forall \omega \in \Omega, \pi(\omega) = 1$ (all values in Ω are possible).

[1] This paper is an extended and revised version of a preliminary work presented in [27,28].

In possibilistic theory, there are two essential measures:

- **Possibility measure:** $\Pi(A) = \sup_{\omega \in A} \pi(\omega)$. $\Pi(A)$ is the possibility of an event A i.e. the degree of possibility of having one of the elements of A (for adventurous decision maker).
- **Necessity measure:** $N(A) = 1 - \Pi(\bar{A}) = 1 - \sup_{\omega \notin A} \pi(\omega)$. $N(A)$ expresses the need for an event A i.e. the certainty of having one of the elements of A (for cautious decision).

In possibilistic decision making, a decision can be seen as a possibility distribution over a finite set of outcomes [20]. In a single stage decision making problem, a utility function maps each outcome to a utility value in a totally ordered set $U = \{u_1, ..., u_n\}$. This function models the attractiveness of each outcome for the decision-maker.

While transition probabilities can be estimated through simulations of the process, transition possibilities may not. On the other hand, experts may be involved for the elicitation the possibility degrees and utilities of transitions. In the possibilistic framework, utility and uncertainty levels can be elicited jointly, by comparison of possibilistic lotteries, for example (e.g. by using certainty equivalent, as in [17]). Simulation can also be used jointly with expert evaluation when the underlying process is too costly to simulate a large number of times (large enough to get reliable estimates of probabilities): Simulation may be used to generate samples on which expert elicitation is applied.

2.2. Possibilistic decision trees

Decision trees (DTs) provide an explicit modeling of sequential decision problems by representing all possible scenarios [8,34]. The graphical component of a DT is a labeled directed tree $< \mathcal{N}, \mathcal{E} >$ which contains three kinds of nodes (see Fig. 1):

- $\mathcal{N}_D$ is the set of decision nodes (represented by squares);
- $\mathcal{N}_C$ is the set of chance nodes (represented by circles);
- $\mathcal{N}_U$ is the set of leaves, also called utility nodes.

Hence $\mathcal{N} = \mathcal{N}_D \cup \mathcal{N}_C \cup \mathcal{N}_U$.

For any node N, $Out(N) \subseteq \mathcal{E}$ denotes its outgoing edges, $Succ(N)$ the set of its children nodes (i.e. immediate successors) and $Succ(N, e)$ the child of N that is reached by edge $e \in Out(N)$. A DT represents a sequential decision problem in the following way:

- Leaf nodes correspond to states of the world in which a utility is obtained; the utility of a leaf node $L_i \in \mathcal{N}_U$ is denoted $u(L_i)$. For the sake of simplicity we assume that utilities are attached to leaves only.
- Decision nodes correspond to states of the world in which a decision is to be made: $D_i \in \mathcal{N}_D$ represents a decision variable Y_i. Its domain corresponds to the labels a of the edges starting from D_i.
  These edges lead to chance nodes, i.e., $Succ(D_i) \subseteq \mathcal{N}_C$.
- A state variable X_j is assigned to each chance node $C_j \in \mathcal{N}_C$. Its domain corresponds to the labels x of the edges starting from that node. Each edge starting from a chance node C_j represents an event $X_j = x$. For any $C_j \in \mathcal{N}_C$, $Succ(C_j) \subseteq \mathcal{N}_U \cup \mathcal{N}_D$ i.e., after the execution of a decision, either a leaf node or a decision node is reached.

Given a decision tree DT, $Start(DT)$ denotes the set of its first decision nodes (it is a singleton containing the root of the tree if this root is a decision node, or its successors if the root is a chance node). For the sake of simplicity, we suppose that all the paths from the root to a leaf in the tree have the same length. h, the horizon of the decision tree is the number of decision nodes along these paths. Given a node $N \in \mathcal{N}$, we shall also consider the subproblem DT_N defined by the tree rooted in N.

The joint knowledge on the state variables is not given in extenso, but through the labeling of the edges issued from chance nodes. In a possibilistic context, the uncertainty pertaining to the possible outcomes of each X_j is represented by a possibility distribution: each edge starting from C_j, representing an event $X_j = x$, is endowed with a number $\pi_{C_j}(x)$, the possibility $\pi(X_j = x | past(C_j))$ where $past(C_j)$ denotes all the value assignments to chance and

decision nodes on the path from the root to C_j.[2] A possibilistic ordered scale, $L = \{\alpha_0 = 0_L < \alpha_1 < \ldots < \alpha_l = 1_L\}$, is used to evaluate the utilities and possibilities.

Solving a decision tree amounts to building a *policy* (or "*strategy*"), i.e., a function $\delta : \mathcal{N}_D \mapsto A$, where A is the set of possible actions, including a special "undefined" action $\perp$, chosen for action nodes which are left unexplored by a given policy.

Admissible policies assign a chance node to each reachable decision node, i.e., must be:

- *sound*: $\forall D_i \in \mathcal{N}_D, \delta(D_i) \in Out(D_i) \cup \{\perp\} \subseteq A$, and
- *complete*: (i) $\forall D_i \in Start(DT), \delta(D_i) \neq \perp$ and
  (ii) $\forall D_i$ s.t. $\delta(D_i) \neq \perp, \forall N \in Succ(Succ(D_i, \delta(D_i)))$ either $\delta(N) \neq \perp$ or $N \in \mathcal{N}_{\mathcal{U}}$.

We denote by Δ_N (or simply Δ, when there is no ambiguity) the set of admissible policies relative to the tree rooted in N. Each policy δ defines a connected subtree of DT, the branches of which represent possible scenarios, or *trajectories*. Formally, a trajectory $\tau = (a_{j_0}, x_{i_1}, a_{j_1}, \ldots, a_{j_{h-1}}, x_{i_h})$ is a sequence of value assignments to decision and chance variables along a path from a starting decision node (a node in $Start(DT)$) to a leaf. $Y_0 = a_{j_0}$ is the first decision in the trajectory, x_{i_1} the value taken by its first chance variable, X_{j_0} in this scenario, $Y_{i_1} = a_{j_1}$ is the second decision, etc. For the sake of notational simplicity, we associate to each τ the vector $\pi_\tau = (\pi_1, \ldots, \pi_h, u_\tau)$ that gathers the possibility and utility degrees encountered on the trajectory; formally, u_τ is the utility $u(x_{i_h})$ of the leaf of τ and $\pi_k = \pi_{C_{j_{k-1}}}(x_{i_k})$ (where $\pi_{C_{j_{k-1}}}$ is the possibility distribution at chance node $C_{j_{k-1}}$) is the possibility of x_{i_k} given that action $a_{j_{h-1}}$ is executed.

We often identify a policy δ, the corresponding subtree and the set of its trajectories (hence the notation $\tau \in \delta$ to mean that τ is a trajectory of δ). We also consider subtrees of the original DT, and thus sub-policies: let C_j be a chance node, $D_{i_1}, \ldots, D_{i_k}$ its successors and, for $l = 1, k$, the policies $\delta_{i_l} \in \Delta_{D_{i_l}}$ which solve the subproblem rooted in D_{i_l}. In order to simplify the indices, we shall abuse of notations and designate by $\pi_\tau = (\pi_1, \ldots, \pi_h, u_\tau)$ the vector associated to a trajectory that belong to a sub-policy.

Thus, $\delta_{i_1} + \cdots + \delta_{i_k}$ is the policy of Δ_{C_j} resulting from the composition of the δ_{i_l}: $(\delta_{i_1} + \cdots + \delta_{i_k})(N) = \delta_{i_l}(N)$ iff N belongs to the subtree rooted in D_{i_l}.

Example 1. Let us suppose that a "Rich and Unknown" person runs a startup company. In every states they must choose between Saving money (Sav) or Advertising (Adv) and they may then get Rich (R) or Poor (P) and Famous (F) or Unknown (U). Fig. 1 shows the possibilistic decision tree (with horizon $h = 2$) that represents this sequential decision problem. This tree has 8 possible policies and 16 trajectories:

$$\tau_1 = (Adv, R\&U, Adv, R\&U), \quad \tau_2 = (Adv, R\&U, Adv, R\&F), \quad \tau_3 = (Adv, R\&U, Sav, P\&U),$$
$$\tau_4 = (Adv, R\&U, Sav, R\&U), \quad \tau_5 = (Adv, R\&F, Adv, R\&U), \quad \tau_6 = (Adv, R\&F, Adv, R\&F), \text{ etc.}$$

The evaluation of a policy, as proposed by [25], relies on the qualitative optimistic and pessimistic decision criteria axiomatized by [20]. The utility of the policy is computed on the basis of the transition possibilities and the utilities of its trajectories. For each trajectory $\tau = (a_{j_0}, x_{i_1}, a_{j_1}, \ldots, x_{i_h})$, with associated vector $\pi_\tau = (\pi_1, \ldots, \pi_h, u_\tau)$:

- Its utility, $u(\tau)$, is the utility u_τ of its leaf.
- The possibility of τ given that a policy δ is applied from initial node D_0 is defined by:

$$\pi(\tau|\delta, D_0) = \begin{cases} \min_{\pi_k \in \pi_\tau} \pi_k & \text{if } \tau \in \delta, \\ 0 & \text{otherwise.} \end{cases}$$

Following [20], Fargier et al. define the optimistic and pessimistic utility degrees of a policy $\delta \in \Delta$ [25]:

$$u_{opt}(\delta) = \max_{\tau \in \delta} \min(\pi(\tau|\delta, D_0), u(\tau)) \tag{1}$$

[2] As in classical probabilistic decision trees, it is assumed that $\pi(X_j = x|past(C_j))$ only depends on the variables in $past(C_j)$ and often only on the decision made in the preceding node and on the state of the preceding chance node.

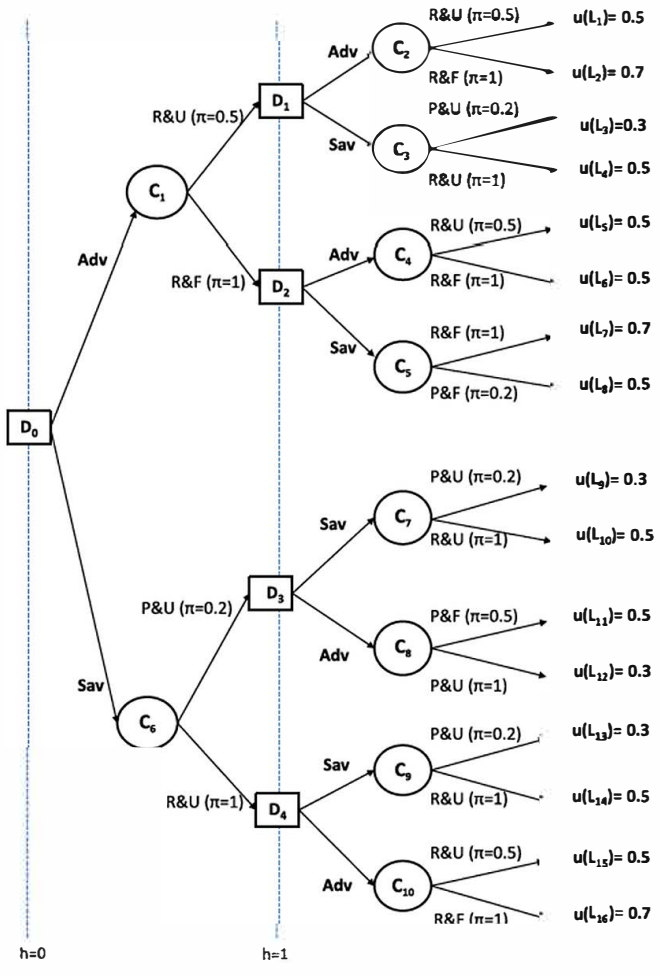

Fig. 1. The possibilistic decision tree of Example 1.

$$u_{pes}(\delta) = \min_{\tau \in \delta} \max\left(1 - \pi(\tau|\delta, D_0), u(\tau)\right) \tag{2}$$

This approach is purely ordinal (only min and max operations are used to aggregate the evaluations of the possibility of events and the utility of states). We can check that the preference orderings $\succeq_O$ between policies, derived either from u_{opt} ($O = u_{opt}$) or from u_{pes} ($O = u_{pes}$), satisfy the principle of weak monotonicity:

$$\forall C_j \in \mathcal{N}_C, \forall D_i \in Succ(C_j), \delta, \delta' \in \Delta_{D_i}, \delta'' \in \Delta_{Succ(C_j)\setminus D_i}:$$
$$\delta \succeq_O \delta' \Longrightarrow \delta + \delta'' \succeq_O \delta' + \delta''$$

This property guarantees that *dynamic programming* [35] applies, and provides an optimal policy in time polynomial with the size of the tree: [24,25] have proposed qualitative counterparts of the stochastic dynamic programming algorithm *backwards induction* for decision trees (see Algorithm 1, written in a recursive style) that optimizes the decisions from the leaves of the tree to its root.

Decision trees represent sequential decision problems under the assumption of complete observation [8,34]. However, DTs have serious limitations in their ability to model complex situations, especially when the horizon is long: the number of nodes at step $t \leq h$ is about b^t, b being the branching factor of the tree. This is why DTs are often replaced with the use of Markov Decision Processes (MDP) [9] which offer a more compact representations of sequential decision problems.

Algorithm 1: Backward-Induction-DT-uopt(N:Node).

Data: A possibilistic DT; the policy, δ, is memorized as global variable
Result: Set δ for the tree rooted in N and returns its optimistic utility

```
begin
    // Leaves
    if N ∈ N_U then u_opt ← u(N)
    // Chance nodes
    if N ∈ N_C then
        foreach D_i ∈ Succ(N) do
            u^i_opt ← Backward-Induction-DT-uopt(D_i);
        u_opt ← max_{D_i ∈ Succ(N)} min(π_N(D_i), u^i_opt);
    // Decision nodes
    if N ∈ N_D then
        u_opt ← 0;
        foreach C_j ∈ Succ(N) do
            u ← Backward-Induction-DT-uopt(C_j);
            if u > u_opt then
                u_opt ← u;
                δ(N) ← label(N, C_j);
    return u_opt;
```

2.3. Possibilistic Markov decision processes

A possibilistic finite-horizon Markov decision process $< T, S, A, \pi, u >$ [24] is defined by:

- A finite set of **stages** $T = \{0, \ldots, h\}$; h is called the horizon of the problem.
- Finite **state spaces**, S_t, for each $t = 0 \ldots h$; $S = S_0 \cup \ldots \cup S_h$ denotes the set of all possible states at every time steps.
- Sets A_s of available **actions** in state $s \in S_t$; $A = A_{S_0} \cup \ldots \cup A_{S_h}$ denotes the full action space.
- In a possibilistic context, the uncertainty of the agent about the effect of an action a taken in state $s \in S_{t-1}$ is represented by a possibility distribution $\pi(.|s,a) : S_t \to L$, where, $L = \{0_L < \ldots < 1_L\}$ is a possibilistic ordered scale. For $s' \in S_t$, $\pi(s'|s,a)$ measures to what extent s' is a plausible consequence of a in s. In the same way, consequences are ordered in terms of levels of satisfaction by a qualitative utility function $u : S_h \to L$.
- The **rewards** $u(s)$ that are obtained in the final states $s \in S_h$. In this article we do not consider intermediate satisfaction degrees. However, our results could be easily extended to handle them.

Decision trees and finite-horizon Markov decision processes are two frameworks that are very close to each other. A finite-horizon (possibilistic) MDP can be translated into a decision tree by representing explicitly every possible trajectories. However, the number of such trajectories may be exponential in the horizon ($O((|S| \times |A|)^h)$) and so is the size of the DT representation of the finite horizon MDP. Thus, a naive application of the backwards induction algorithm to a DT translation of a MDP is inefficient. Fortunately, [24] has shown that a backwards induction algorithm could be defined for possibilistic MDP, which complexity is only polynomial in the representation size of the MDP (similarly to the stochastic MDP case).

Example 2. Let us consider the problem introduced in Example 1 — it is possible to represent it as a possibilistic finite-horizon MDP. Fig. 2 represents the possibilistic MDP, in the form of an acyclic graph, when the horizon $h = 2$ (here, utilities are also shown). We have: $S_0 = \{R\&U_0\}$, $S_1 = \{R\&F_1, R\&U_1, P\&U_1\}$ and $S_2 = \{R\&F_2, R\&U_2, P\&F_2, P\&U_2\}$ also $\forall t = 0, 2, s \in S_t A_s = \{Adv, Sav\}$.

Solving a finite-horizon MDP consists in finding an (optimal) policy i.e., is a function that maps each state to an admissible action δ: $S \to A$, s.t. $\delta(s) \in A_s$.

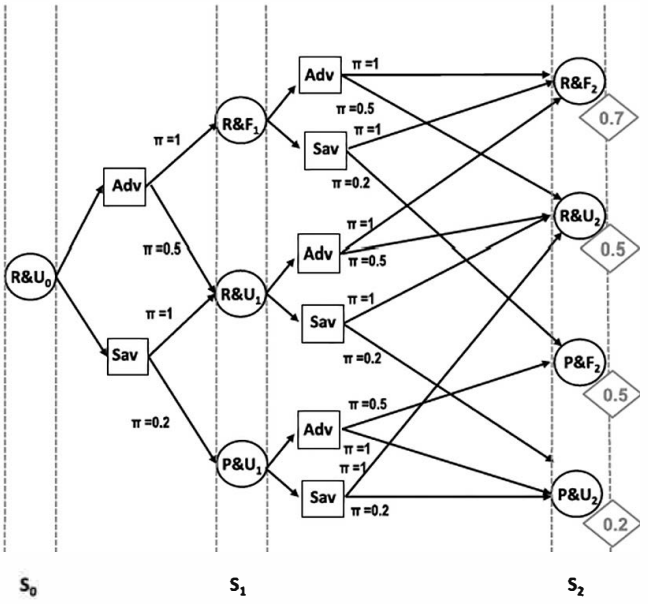

Fig. 2. Example of finite-horizon possibilistic MDP ($h = 2$).

When applied from a state $s_{i0} \in S_0$, such a policy defines a list of trajectories, as for the decision trees case. A trajectory τ is a sequence of actions and states along a path following (and excluding) a first state s_{i0} to a final state $s_{ih} \in S_h$. Formally,[3] $\tau = (a_{j0}, \ldots, s_{ik}, a_{jk}, \ldots, s_{ih})$, where $s_{ik} \in S_k$ and $a_{jk} = \delta(s_{ik})$.

Note that s_{i0} is not part of the trajectory but given alongside the MDP model. a_{j0} is the first action in the trajectory — the one prescribed by δ for s_{i0}, etc. The reward associated to τ is the utility $u_\tau = u(s_{ih})$ obtained in the final state of the trajectory s_{ih}. For the sake of notational simplicity, we will associate to each trajectory τ the vector $\pi_\tau = (\pi_1, \ldots, \pi_h, u_\tau)$ where $\pi_k = \pi(s_{ik}|s_{ik-1}, a_{jk-1})$ denotes the possibility degree to reach s_k applying action a_{jk-1} from state s_{ik-1}.

The qualitative pessimistic utility of a policy δ in state s_0 is defined by the qualitative *minmax* expectation of the degrees of satisfaction of the final states of the possible trajectories, and the optimistic utility as the *maxmin* expectation of the same:

$$u_{pes}(s_0, \delta) = \min_{\tau \in \delta} \max\{1 - \Pi(\tau|s_0, \delta), u_\tau\} \tag{3}$$

$$u_{opt}(s_0, \delta) = \max_{\tau \in \delta} \min\{\Pi(\tau|s_0, \delta), u\tau\} \tag{4}$$

The possibility $\Pi(\tau|s_0, \delta)$ of τ given that policy δ is applied from initial state s_0 is defined by:

$$\Pi(\tau|s_0, \delta) = \begin{cases} \min_{\pi_k \in \pi_\tau} \pi_k & \text{if } \tau \in \delta, \\ 0 & \text{otherwise.} \end{cases}$$

These criteria can be optimized by choosing, for each state, an action that maximizes the following counterparts of the Bellman equations [24]:

- In the pessimistic case:

$$u_{pes}(s) = \max_{a \in A_s} \min\{u(s), \min_{s' \in S_{t+1}} \max\{1 - \pi(s'|s, a), u^{t+1}_{pes}(s')\}\} \forall t < h, s \in S_t \tag{5}$$

$$u_{pes}(s) = u(s) \; \forall s \in S_h$$

[3] We suppose, without loss of generality, that all trajectories have the same length h.

- In the optimistic case:

$$u_{opt}(s) = \max_{a \in A_s} \min\{u(s), \max_{s' \in S_{t+1}} \min(\pi(s'|s,a), u_{opt}(s'))\} \forall t < h, s \in S_t \quad (6)$$
$$u_{opt}(s) = u(s)\ \forall s \in S_h.$$

[24,36] have shown that any policy computed backwards by successive applications of equation (5) (resp. (6)) is optimal according to u_{pes} (resp. u_{opt}). As a matter of fact, any policy returned by the possibilistic backward induction algorithm (Algorithm 2) is optimal with respect to u_{opt}.

Algorithm 2: Backward-Induction-MDP-uopt.

Data: A possibilistic MDP
Result: Computes and returns an optimal policy δ

begin
  $t \leftarrow h$;
  for $s \in S_h$ **do** $u_{opt}(s) \leftarrow u(s)$
  while $t \geq 1$ **do**
    $t \leftarrow t - 1$;
    foreach $s \in S_t$ **do**
      $u_{opt}(s) \leftarrow \max_{a \in A_s} \max_{s' \in S_{t+1}} \min\{\pi(s'|s,a), u_{opt}(s')\}$;
      $\delta(s) \leftarrow \arg\max_{a \in A_s} \left\{ \max_{s' \in S_{t+1}} \min\{\pi(s'|s,a), u_{opt}(s')\} \right\}$;
  return δ;

2.4. The possibilistic drowning effect

The basic pessimistic and optimistic utilities present a severe drawback, known as the "drowning effect", due to the use of idempotent operations. In particular, when two policies give an identical and extreme (either good, for u_{opt} or bad, for u_{pes}) utility in some plausible trajectory, they may be undistinguished although having given significantly different consequences in other possible trajectories. In what follows we illustrate the drowning effect of qualitative utilities on possibilistic decision trees (the same counter example holds for finite-horizon MDPs, using the modeling proposed in Fig. 2).

Example 3. Let δ and δ' be the two policies of Example 1 defined by:

$$\delta(D_0) = \delta'(D_0) = Adv; \delta(D_1) = Sav; \delta'(D_1) = Adv; \delta(D_2) = \delta'(D_2) = Adv.$$

δ has 4 trajectories, $\tau_3, \tau_4, \tau_5, \tau_6$ with $\pi(\tau_3|D_0,\delta) = 0.2$ and $u(\tau_3) = 0.3$; $\pi(\tau_4|D_0,\delta) = 0.5$ and $u(\tau_4) = 0.5$; $\pi(\tau_5|D_0,\delta) = 0.5$ and $u(\tau_5) = 0.5$; $\pi(\tau_6|D_0,\delta) = 1$ and $u(\tau_6) = 0.5$. Hence $u_{opt}(\delta) = u_{pes}(\delta) = 0.5$. δ' is also composed of 4 trajectories $(\tau_1, \tau_2, \tau_5, \tau_6)$. As before, we can show that $u_{opt}(\delta') = u_{pes}(\delta') = 0.5$.

Thus $u_{opt}(\delta) = u_{opt}(\delta')$ and $u_{pes}(\delta) = u_{pes}(\delta')$: δ', which provides at least utility 0.5 in all trajectories, is not preferred to δ that provides a bad utility (0.3) in some non-impossible trajectory (τ_3). τ_4, which is good and possible "drowns" the bad consequence of δ in τ_3 in the optimistic comparison; in the pessimistic one, the bad utility of τ_3 is drowned by its low possibility, hence a global degree $u_{pes}(\delta)$ that is equal to the one of δ' (which, once again, guarantees a utility degree of 0.5 at least).

The two possibilistic criteria, thus, may fail to satisfy the principle of *Pareto efficiency*, that may be written as follows, for any optimization criterion O (here u_{pes} or u_{opt}):
$\forall \delta, \delta' \in \Delta$, if (i) $\forall N \in \text{Common}(\delta, \delta'), \delta_N \succeq_O \delta'_N$ and (ii) $\exists N \in \text{Common}(\delta_N, \delta'_N)$,

$$\delta_N \succ_O \delta'_N, \text{ then } \delta \succ_O \delta',$$

where Common(δ, δ') is the set of situations (decision nodes in decision trees, states in the MDP framework) for which both δ and δ' provide an action and δ_D (resp. δ'_D) is the restriction of δ (resp. δ') to the subtree rooted in D.

Moreover, neither u_{opt} nor u_{pes} fully satisfy the classical, strict, monotonicity principle, that can be written as follows, for any optimization criterion O

$$\forall C_j \in \mathcal{N}_C, D_i \in Succ(C_j), \delta, \delta' \in \Delta_{D_i}, \delta'' \in \Delta_{Succ(C_j)\setminus D_i},$$
$$\delta \succeq_O \delta' \iff \delta + \delta'' \succeq_O \delta' + \delta''.$$

It may indeed happen that $u_{pes}(\delta) > u_{pes}(\delta')$ while $u_{pes}(\delta + \delta'') = u_{pes}(\delta' + \delta'')$ (or that $u_{opt}(\delta) > u_{opt}(\delta')$ while $u_{opt}(\delta + \delta'') = u_{opt}(\delta' + \delta'')$).

The purpose of the present work is to build efficient preference relations that agree with the qualitative utilities when the latter can make a decision, and break ties when not — to build refinements[4] that satisfy the principle of Pareto efficiency.

In order to overcome the drowning effect, Fargier and Sabbadin have proposed *lexicographic refinements* of possibilistic criteria for non-sequential decision problems [26]. However, these refined criteria cannot be used in *sequential* decision problems, where the drowning effect is also due to the reduction of compound possibilistic policies into simple possibility distributions. In the next section, we propose an extension of these lexicographic preference relations to finite horizon sequential problems.

3. Escaping the drowning effect by lexicographic comparisons

The possibilistic drowning effect is due to the use of min and max operations. In ordinal aggregations, this drawback is well known and it has been overcome by means of leximin and leximax comparisons [37]. More formally, for any two vectors t and t':

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

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

The lexicographic refinements of the preference relations induced by u_{opt} and u_{pes} have been considered by [26] for non-sequential problems.

In a one step decision problem each decision can indeed be identified with a possibility distribution π over the states (i.e., utility degrees), i.e., a vector of pairs $(\pi(u), u)$. Then it is possible to write:

- $\pi \trianglerighteq_{lmax(lmin)} \pi'$ iff $\forall i, (\pi(u), u)_{\lambda(i)} \sim_{lmin} (\pi'(u), u)_{\lambda(i)}$ or
  $\exists i^*, \forall i < i^*, (\pi(u), u)_{\lambda(i)} \sim_{lmin} (\pi'(u), u)_{\lambda(i)}$ and $(\pi(u), u)_{\lambda(i^*)} \succ_{lmin} (\pi'(u), u)_{\lambda(i^*)}$.
- $\pi \trianglerighteq_{lmin(lmax)} \pi'$ iff $\forall i, (1 - \pi(u), u)_{\sigma(i)} \sim_{lmax} (1 - \pi'(u), u)_{\sigma(i)}$ or
  $\exists i^*, \forall i < i^*, (1 - \pi(u), u)_{\sigma(i)} \sim_{lmax} (1 - \pi'(u), u)_{\sigma(i)}$ and $(1 - \pi(u), u)_{\sigma(i^*)} \succ_{lmax} (1 - \pi'(u), u)_{\sigma(i^*)}$.

where $(\pi(u), u)_{\lambda(i)}$ is the ith best pair among the $(\pi(u), u)$, according to $lmin$ and $(1 - \pi(u), u)_{\sigma(i)}$ is the ith worst pair among the $(1 - \pi(u), u)$, according to $lmax$.

A straightforward way of applying lexicographic comparisons to sequential decision is to associate to any policy the possibility distribution that it induces on the utility rewards, as usually done in possibilistic (and probabilistic) decision trees. We call this possibility distribution the *reduction* of δ.

Formally, for a policy δ and any of its trajectories ($\tau = (a_{j_0}, x_{i_1}, a_{j_1}, \ldots, x_{i_h})$ in a DT or $\tau = (a_{j0}, ..., s_{ik}, a_{jk}, s_{ih})$ in a MDP with starting state s_{i0}), the vector $\pi_\tau = (\pi_1, \ldots, \pi_h, u_\tau)$ has been defined, that gathers the possibility and utility degrees encountered on the trajectory. The possibility of τ is equal to $\min_{\pi_{k=1,\ldots,h} \in \pi_\tau} \pi_k$ and the possibility of getting

[4] Formally, a relation $\succeq'$ refines a relation $\succeq$ if and only if whatever δ, δ', if $\delta \succ \delta'$ then $\delta \succ' \delta'$: whenever $\succeq$ strictly prefers δ to δ', $\succeq'$ makes the same decision; but it can be more decisive and break ties — it may happen that $\delta \sim \delta'$ and $\delta \succ' \delta'$.

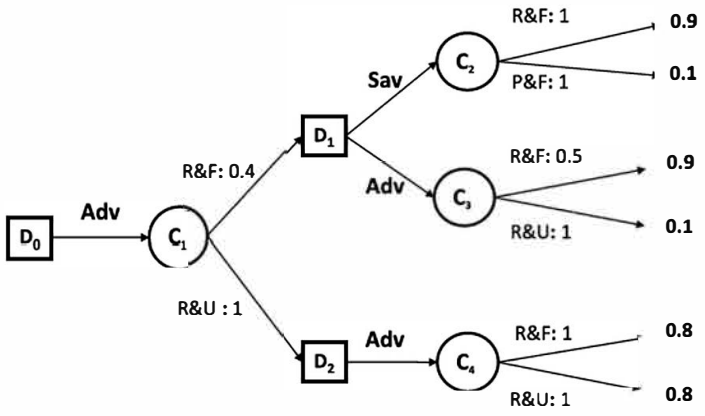

Fig. 3. A counter example showing that $\trianglerighteq_{lmax(lmin)}$ does not satisfy Pareto efficiency.

u_τ is the possibly of the set of trajectories of δ that provide this utility degree. Hence, for any δ, the reduction of δ is the distribution π_δ on the utility degrees, defined by:

$$\pi_\delta(u) = \max_{\tau \in \delta,\ u_\tau = u} \ \min_{\pi_k \in \pi_\tau} \pi_k.$$

This is the principle of reduction that is used when qualitative decision theory is used by [24,25] to compare policies: the pessimistic (resp. optimistic) utility of a policy is simply the one of its reduction. Because the π_δ are single stepped, one can think on applying lexicographic comparisons as such, and can write:

$$\delta \trianglerighteq_{lmax(lmin)} \delta' \text{ iff } \pi_\delta \trianglerighteq_{lmax(lmin)} \pi_{\delta'},$$

$$\delta \trianglerighteq_{lmin(lmax)} \delta' \text{ iff } \pi_\delta \trianglerighteq_{lmin(lmax)} \pi_{\delta'}.$$

$\trianglerighteq_{lmax(lmin)}$ (resp. $\trianglerighteq_{lmin(lmax)}$) refines $\succeq_{u_{opt}}$ (resp. $\succeq_{u_{pes}}$), but neither $\trianglerighteq_{lmax(lmin)}$ nor $\trianglerighteq_{lmin(lmax)}$ do satisfy Pareto efficiency, as shown by the following counterexample.

Example 4. Consider a modified version of Example 1 (Fig. 3). δ and δ' are the two policies defined by: $\delta(D_0) = \delta'(D_0) = Adv$, $\delta(D_1) = Sav$, $\delta' = (D_1) = Adv$, $\delta(D_2) = \delta'(D_2) = Adv$. $Common(\delta, \delta') = \{D_0, D_1, D_2\}$, $\delta_{D_0} = \delta'_{D_0}$, $\delta_{D_2} = \delta'_{D_2}$ and δ_{D_1} dominates δ'_{D_1} with respect to $lmax(lmin)$, since $((1, 0.1), (1, 0.9)) \rhd_{lmax(lmin)} ((1, 0.1)(0.5, 0.9))$. δ should then be strictly preferred to δ'. Let us compute the reduction of δ: $\pi_\delta(0.9) = \pi_\delta(0.1) = \min(0.4, 1) = 0.4$ and $\pi_\delta(0.8) = \min(1, 1) = 1$ and for δ' we have $\pi_{\delta'}(0.9) = \min(0.4, 0.5) = 0.4$, $\pi_{\delta'}(0.1) = \min(0.4, 1) = 0.4$ and $\pi_{\delta'}(0.8) = \min(1, 1) = 1$: δ and δ' are indifferent for $\trianglerighteq_{lmax(lmin)}$. This contradicts Pareto efficiency.

The drowning effect here is due to the reduction of the policies, namely to the fact that the possibility of a trajectory is drowned by the one of the least possible of its transitions. That is why we propose to give up the principle of reduction and to build lexicographic comparisons on policies considered *in extenso*. For any $\pi_\tau = (\pi_1, \ldots, \pi_h, u_\tau)$ and $\pi_{\tau'} = (\pi'_1, \ldots, \pi'_h, u'_h)$, we define $\succeq_{lmin}$ and $\succeq_{lmax}$ by:

- $\tau \succeq_{lmin} \tau'$ iff $(\pi_1, \ldots, \pi_h, u_\tau) \succeq_{lmin} (\pi'_1, \ldots, \pi'_h, u'_h)$
- $\tau \succeq_{lmax} \tau'$ iff $(1 - \pi_1, \ldots, 1 - \pi_h, u_\tau) \succeq_{lmax} (1 - \pi'_1, \ldots, 1 - \pi'_h, u'_h)$

Hence the proposition of the following preference relations:

- $\delta \succeq_{lmax(lmin)} \delta'$ iff $\forall i,\ \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ or $\exists i^*, \forall i \leq i^*, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$,
- $\delta \succeq_{lmin(lmax)} \delta'$ iff $\forall i,\ \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ or $\forall i,\ \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ or $\exists i^*, \forall i \leq i^*, \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ and $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$,

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

These relations are relevant refinements and escape the drowning effect — they are those we are looking for:

Proposition 1. *$\succeq_{lmax(lmin)}$ is complete, transitive and refines $\succeq_{u_{opt}}$; $\succeq_{lmin(lmax)}$ is complete, transitive and refines $\succeq_{u_{pes}}$.*

Proposition 2. *$\succeq_{lmax(lmin)}$ and $\succeq_{lmin(lmax)}$ both satisfy the principle of Pareto efficiency as well as strict monotonicity.*

Propositions 1 and 2 have important consequences. From a prescriptive point of view, they outline the rationality of $lmax(lmin)$ and $lmin(lmax)$. Moreover, the fact that these preference relations are weak orders and satisfy strict monotonicity suggest a probabilistic interpretation, which we develop in Section 5. From a practical point of view, Propositions 1 and 2 allow us to define a backward induction algorithm to get lexicographically optimal solutions. This is the topic of the next section.

4. Dynamic programming for lexicographic criteria

4.1. Dynamic programming for lexicographic criteria in possibilistic DTs

The algorithm we propose proceeds in the classical way, by backward induction (see Algorithm 3 for the $lmax(lmin)$ variant (the $lmin(lmax)$ variant is similar). The difference is that the lexicographic comparison of policies is done on the basis of their trajectories. To this extend, the algorithm needs, for each possible policy, the matrix ρ of the vectors $\pi_\tau = (\pi_1, \ldots, \pi_h, u_\tau)$ of its trajectories (in the optimistic case) or the one of the vectors $(1-\pi_1, \ldots, 1-\pi_h, u_\tau)$ (in the pessimistic case). Formally, for line z corresponding to the zth trajectory and for a criterion O:

$$\rho_{z,t} = \begin{cases} \pi_t & \text{if } t \leq h \text{ and } O = lmax\big(lmin\big) \\ 1-\pi_t & \text{if } t \leq h \text{ and } O = lmin\big(lmax\big) \\ u_\tau & \text{if } t = h+1. \end{cases}$$

Algorithm 3: Backward-Induction-DT-lmax(lmin)(N:Node).

Data: A possibilistic DT; the policy, δ, is memorized as global variable
Result: Set δ for the tree rooted in N and returns the matrix ρ of the π_τ vectors corresponding its trajectories

begin
  // Leaves
  if $N \in \mathcal{N}_{\mathcal{U}}$ **then** $\rho = [u(N)]$
  // Chance nodes
  if $N \in \mathcal{N}_C$ **then**
    foreach $D_i \in Succ(N)$ **do**
      $\rho^{D_i} \leftarrow Backward\text{-}Induction\text{-}DT\text{-}lmax(lmin)(D_i)$
    $\rho \leftarrow ConcatAndOrder(\pi_N, \rho^{D_1}, ..., \rho^{D_k})$;
    // with $k = |Succ(N)|$;
  // Decision nodes
  if $N \in \mathcal{N}_D$ **then**
    $\rho \leftarrow [0]$;
    foreach $C_j \in Succ(N)$ **do**
      $\rho^{C_j} \leftarrow Backward\text{-}Induction\text{-}DT\text{-}lmax(lmin)(C_j)$;
      if $\rho^{C_j} \succeq_{lmax(lmin)} \rho$ **then**
        $\rho \leftarrow \rho^{C_j}$;
        $\delta(N) \leftarrow label(N, C_j)$;
  return ρ;

The algorithm is written in a recursive manner, and proceeds as follows: when the node N reached is a chance node, an optimal sub-policy is recursively built for each of its children D_i — these recursive calls return for each D_i a matrix ρ^{D_i} that contains the π_τ vectors of the trajectories τ of this sub-policy.[5] The matrix corresponding to the trajectories beginning at N, namely ρ is obtained by combining the ρ^{D_i} according to π_N, the possibility distribution associated to N; this matrix is not reduced contrarily to what is classically done. When a decision node is reached, an optimal sub-policy is computed for every child C_j. The best of them is selected, $\delta(N)$ receives the action corresponding to this chance node and the corresponding ρ matrix is returned.

If N is a chance node, its matrix depends on the ones of its successors, $D_1, \ldots, D_k$ (on the ρ^{D_i}, recursively computed) and on the possibility distribution on them. It is built by calling function $ConcatAndOrder(\pi_N, \rho^{D_1}, \ldots, \rho^{Dk})$. This function adds a column to each ρ^{D_i}, filled with $\pi_N(D_i)$ the possibility degrees of getting D_i when choosing the action represented by N; the matrices are vertically concatenated. In order to get faster lexicographic comparisons, the elements in the lines are then ordered in decreasing (resp. increasing) order, and the lines (the rows) are reordered by decreasing (resp. increasing) order with respect to $lmax$ (resp. $lmin$). As a matter of fact, once ρ has been reordered, $\rho_{1,1}$, the first element of ρ, is always equal to the optimistic utility (resp. the pessimistic utility) of the sub-policy represented by ρ.

Procedure "ConcatAndOrder" is given by Algorithm 4. Given a matrix ρ, we use these notations:

- L_ρ: number of lines of ρ,
- C_ρ: number of columns of ρ,
- ρ_i: the line i in ρ,
- ρ_{ij}: the element in line i and column j in ρ.

Algorithm 4: ConcatAndOrder(π, $\rho^1, \ldots, \rho^k$).

```
   Data: k matrices ρ^1, ..., ρ^k and a distribution π on {1, ..., k}
   Result: ρ, the combination of ρ^1, ..., ρ^k according to π
1  begin
2      NbLines ← Σ_{m=1}^{k} L_{ρ^m};
3      max_C ← max_{m=1,k}(C_{ρ^m});
4      Creates a matrix ρ with NbLines lines and max_C + 1 columns
5      // Concatenation
6      i ← 0;
7      for m = 1, k do
8          for i' = 1, L_{ρ^m} do
9              i ← i + 1;
10             for j = 1, C_{ρ^m} do ρ_{i,j} ← ρ^m_{i'j}
11             for j = C_{ρ^m} + 1, max_C do ρ_{i,j} ← 0
12             ρ_{i,max_C+1} ← π(m);
13     // Ordering the elements of each line by increasing order
14     for i = 1, NbLines do
15         sortIncreasing(ρ_i, ≥);
16     // Ordering the lines by decreasing order according to lmax
17     sortDecreasing(ρ, ≥_{lmax});
18     return ρ;
```

Because the ρ matrices are ordered, the lexicographic comparison of two decisions (line 15) is performed by scanning the elements of their ρ matrices, line by line from the first one. The first pair of different values determines the best matrix/chance node.

[5] To make the lexicographic comparison of trajectories, and thus of policies, we only need to compare their π_τ vectors — hence we memorize the matrices of numbers rather than explicit trajectories.

If a trajectory is shorter than the horizon h, neutral elements (1 for the optimistic case and 0 for the pessimistic one) are added at the end. If the policies (or sub-policies) have different numbers of trajectories, we compare the two matrices based on the number of trajectories of the shortest matrix, then two cases arise:

- if we have a strict preference between the two matrices, we get a strict preference between the policies (or between the sub-policies);
- if we have an indifference, we deduce that the shortest matrix is the best one w.r.t. the lexicographic criterion, since it expresses less uncertainty in the corresponding policy (or in the sub-policy).

If the matrices have different numbers of lines, neutral lines are added at the bottom of the shortest one (filled with 0 for the optimistic case, with 1 for the pessimistic one).

Since it requires to memorize the trajectories that follow from the current policy (i.e., the ρ matrices). Let b be the branching factor of the tree; the size of the tree is thus equal to b^{2h}. Now, consider the size of a matrix ρ: it is in the order of $b^h \times (h+1)$ in the worst case (i.e., at the end of the backward induction) — the same order of magnitude as the one of the size of the tree.

Let us now study the time complexity. The complexity of ordering matrices depends on the sorting algorithm: for instance, if we use QuickSort on an $n \times m$ matrix, 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 of ordering matrices is in $O(n \cdot m \cdot \log(n \cdot m))$.

At each step t, from $t = h-1$ to $t = 1$, there is a chance phase and a decision phase — b decision nodes, each followed by b chance nodes. The chance phase is more expensive than the decision one — it makes the same number of recursive calls than the decision phase, but the decision phase does not increase the size of the matrices it receives while the chance phase builds and orders, for each of its b^2 chance nodes a matrix that is bigger than the one it receives: it receives (for a given chance node, again) b matrices b^{h-t-1} lines and $h-t-1$ columns and concatenate them in a matrix with b^{h-t} lines and $h-t$ columns (concat). The ordering then costs $b^{h-t} \cdot (h-t) \cdot \log(b^{h-t} \cdot (h-t))$. The comparison is cheaper, since it costs $b^{h-t} \cdot (h-t)$ in the worst case. So the worst time complexity at step t is a function $T_t = b \cdot b^{h-t} \cdot (h-t) \cdot log(b^{h-t} \cdot (h-t))$. The order of magnitude of worst case complexity of the algorithm is thus $U = \Sigma_{t=h-1}^{1} T_t = \Sigma_{t=h-1}^{1} b \cdot b^{h-t} \cdot (h-t) \cdot log(b^{h-t} \cdot (h-t))$. Letting $y = b^{h-t} \cdot (h-t)$, we get $y = b^{h-1} \cdot (h-1)$ at $t = 1$ and $y = b$ at $t = h-1$. Thus $U = \Sigma_{y=b}^{b^{h-1} \cdot (h-1)} b \cdot y \cdot log(y)$, i.e., $U = b^2 \cdot (b^{h-1} \cdot (h-1))^2 \cdot log(b^{h-1} \cdot (h-1))$: the time complexity is in $O(b^{2h} \cdot (h-1)^2 log((h-1) \cdot b^{h-1})$ — it is polynomial with respect to the horizon and the size of the tree (which, again, is in b^{2h}).

4.2. Dynamic programming for lexicographic criteria in possibilistic MDPs

A first way to solve a finite-horizon possibilistic Markov Decision Process would be to compute a Decision Tree that is equivalent to the MDP (this is always possible, through the duplication of the nodes with several predecessors) and to apply the algorithm presented in the previous section. However, as already mentioned this approach may lead to algorithms which are exponential in time and space (with respect to the MDP description), since the size of the decision trees associated to a MDP may be exponential in the size of the MDP. In this section, we propose an algorithm that calculates a lexicographically optimal policy by a backward induction on the MDP itself. Algorithm 5 below performs the optimization of $lmax(lmin)$ (the $lmin(lmax)$ variant is similar).

As in the case of possibilistic decision trees (Section 4.1), the comparison of decisions (here, of actions a) is done on the basis of the trajectories they induce, given the decisions made for the future state. To this extent, one memorizes, for each state s for which a decision has been made, the matrix $\rho(s)$ corresponding to the trajectories obtained when the current policy is applied from s. For $s \in S_t$, $\rho(s)$ is defined as follows: for $lmax(lmin)$ each line gathers the possibility degrees $\pi(s'|a,s)$ of reaching the following state $s' \in S_{t+1}$, given that $\delta(s) = a$ is executed (resp. $1 - \pi(s'|s,a)$ for $lmin(lmax)$), combined with a trajectory in the matrix of the next state s'. The matrices corresponding to the final states simply contain their utility.

The principle of the backward induction algorithm can be summarized as follows: suppose being at period t (e.g. $t = h-1$). Since the algorithm proceeds backwards, a decision $\delta(s')$ has been made for all future states (the s' in $S_{t'}, t' > t$). We have to decide the best action for the states of S_t. For each state $s \in S_t$ and each action $a \in A_s$ we build the matrix M corresponding to trajectories that would be obtained if a was chosen for s, using the *ConcatAndOrder*

Algorithm 5: Backward-induction-MDP-lmax(lmin).

Data: A possibilistic finite horizon $\mathcal{MDP}$
Result: Computes and returns δ for $\mathcal{MDP}$

```
1  begin
2      // Initialization
3      ∀s ∈ S_h, ρ(s) ← [u(s)];
4      t ← h;
5      // Backward induction
6      while t ≥ 1 do
7          Future ← {ρ(s'), s' ∈ S_t};
8          t ← t − 1;
9          foreach s ∈ S_t do
10             ρ(s) ← [0];
11             foreach a ∈ A_s do
12                 M ← ConcatAndOrder(π(.|a, s), Future);
13                 if M ≥_{lmax(lmin)} ρ(s) then
14                     ρ(s) ← M;
15                     δ(s) ← a;
16     return δ;
```

procedure described in the previous section — M is built from $\pi(.|s,a)$ and from the matrices already computed for the s'. M is then compared with the best matrix $\rho(s)$ found so far: if better, a becomes the current best decision for s ($\delta(s) \leftarrow a$) and M becomes the new $\rho(s)$. The lexicographic comparison of matrices is the one described in Section 3 and is made easier by the fact that the matrices have been ordered on the fly. The process is continued for each $S_{t'}, t' = t-1, \ldots, 0$ (by moving backward in time) until we reach the present time period ($t = 0$) and get an optimal policy.

The backwards induction algorithm only makes a polynomial number (in the size of the MDP definition) of calls to the *ConcatAndOrder* function: there are as many number of calls to this function as the number of actions in the MDP, which is $\sum_{t=1}^{h-1} \sum_{s \in S_t} |A_s|$. At each step t, for each state s in S_t: for each action in $|A_s|$, b matrices of size $b^{h-t-1} \cdot (h-t-1)$ are received. Concatenated as a $b^{h-t} \cdot (h-t)$ matrix, ordered — which costs $b^{h-t} \cdot (h-t) log(b^{h-t}.(h-t))$ and compared with the best matrix found so far — which costs $b^{h-t} \cdot (h-t)$. Hence, we have a time complexity in $U = \sum_{t=1}^{h-1} \sum_{s \in S_t} |A_s| \cdot b^{h-t} \cdot (h-t) log(b^{h-t} \cdot (h-t))$. Denote n the (maximal) number of states in S_t and a the (maximal) number of actions in $|A_s|$, we get $U = n \cdot a \sum_{t=1}^{h-1} b^{h-t} \cdot (h-t) \cdot log(b^{h-t} \cdot (h-t))$ i.e., a time complexity in $O(n \cdot a \cdot (h-1)^2 \cdot b^{h-1} \cdot log((h-1) \cdot b^{h-1}))$.[6]

In summary, lexicographic comparisons, which take into account the whole matrix of subsequent trajectories, overcome the drowning effect but are very costly — exponential in the size of the MDP. On the other hand selecting decisions on the basis of their sole optimistic/pessimistic utility is cheap but not discriminant enough. Hence the idea to restrict the reasoning to a sub-matrix — namely to the first l lines of the matrices of trajectories, i.e., π_τ vectors of the l most important trajectories (the l best for the optimistic case, the l worst for the pessimistic ones). Bounding the number of columns is not necessary, since the combinatorial explosion in Algorithm 5 is due to the number of lines in the matrices (because for the finite horizon, the number of columns is bounded by $h+1$). We get variant of Algorithm 5, which we call "Bounded Lexicographic Backward Induction" (BL-BI) by simply replacing line 14 by line 14′:

$$14': \quad \rho(s) \leftarrow \langle M \rangle_l$$

where $\langle M \rangle_l$ denote the restriction of M to its first l lines.

Clearly, this algorithm is not guaranteed to provide a lexicographically optimal solution, but the policy is always at least as good as the one provided by u_{opt} (according to $lmax(lmin)$). Indeed, bounding the matrices is done *after* they have been ordered. Hence $M_{1,1}$ is equal to u_{opt} in the unbounded case and because the bounding is done after

[6] The details of the calculation are similar to the ones made for decision trees.

reordering, this property still holds when using $BL\text{-}BI$. Hence the order on matrices (and thus on policies) refines the one provided by classical optimistic BI algorithm. The optimal policy for Bounded Lexicographic Backward Induction is optimal for the optimistic utility. Actually, the greater l, the more refined the comparison over the policies. This comparison tends to $\succ_{lmax(lmin)}$ when l tends to b^h. The same holds in the pessimistic case: the optimal policy for Bounded Lexicographic Backward Induction is optimal for the pessimistic utility and the greater l, the more refined the comparison over the policies and the comparison tends to $\succ_{lmin(lmax)}$ l tends to b^h.

The temporal complexity of Bounded Lexicographic Backward Induction is decreased, compared to that of the Lexicographic Backward Induction. Indeed, the number of calls to *ConcatAndOrder* of the algorithm does not change. However, the *ConcatAndOrder* algorithm is only called on sets of matrices which have at most l lines, instead of b^h. We deduce that the complexity of the algorithm is bounded by $= O(n \cdot l \cdot (h-1)^2 \cdot log((h-1) \cdot l)$, where n is the (maximal) number of states in S_t and a is the (maximal) number of actions in $|A_s|$.

5. Lexicographic comparisons on decision trees and expected utility

When the problem is not sequential, it has been shown that the comparison of possibilistic lotteries (i.e., a possibilistic distribution on utility degrees) by $\succeq_{lmax(lmin)}$ and $\succeq_{lmin(lmax)}$ satisfy the axioms of EU. Fargier and Sabbadin have indeed shown that these decision criteria can be captured by an EU criterion — namely, relying on big-stepped probabilities and utilities [26]. In what follows, we claim that such a result can be extended to decision trees (and, because a finite horizon MDP can always be transformed into an equivalent decision tree, to MDPs).

The proof relies on a transformation of the possibilistic decision tree into a probabilistic one. The graphical components are identical and so are the sets of admissible policies. In the optimistic case the probability and utility distributions are chosen in such a way that the $lmax(lmin)$ and EU criteria provide the same preference relation on Δ. To this extent, we build a transformation function $\phi : L \subseteq [0,1] \to [0,1]$ that maps each possibility degree to an additive probability and each utility level to an additive one.

For any chance node C_j, a local transformation ϕ_j is derived from ϕ, such that ϕ_j satisfies the normalization condition. In addition, ϕ and ϕ_j required to satisfy the following condition:

$$(R') : \forall \alpha, \alpha' \in L \text{ such that } \alpha > \alpha' : \phi^-(\alpha)^{h+1} > b^h \phi^+(\alpha'),$$

where ϕ^- and ϕ^+ are two functions defined as follows:

- $\phi^-(\alpha) = \min\{\phi(\alpha),\ min_j\ \phi_j(\alpha)\}, \forall \alpha \in L,$
- $\phi^+(\alpha) = \max\{\phi(\alpha),\ max_j\ \phi_j(\alpha)\}, \forall \alpha \in L.$

Condition (R') guarantees that if $u_{opt}(\delta) = \alpha > u_{opt}(\delta') = \alpha'$, then a comparison based on a sum-product approach on the new probabilistic tree will also decide in favor of δ.

EU_{opt} denotes the preference relation provided by the EU-criterion on the probabilistic tree obtained by replacing each π_j by $\phi_j \circ \pi_j$ and the utility function u by $\phi \circ u$. We show that:

Proposition 3. *If (R') holds, then $\succeq_{EU_{opt}}$ refines $\succeq_{u_{opt}}$.*

Proposition 4. $\delta \succeq_{lmax(lmin)} \delta'$ *iff* $\delta \succeq_{EU_{opt}} \delta', \forall (\delta, \delta') \in \Delta$.

Example 5. We illustrate in the following an example of transformation of the decision tree of Fig. 3 with $h = b = 2$. First, let us build a function ϕ satisfying (R'). For this purpose, it is sufficient to construct the function ϕ: $L \to \mathcal{R}$ as follows:

$$\phi(1_L) = 1,\ \phi(\alpha_i) < \frac{\phi(\alpha_{i+1})^3}{4} \text{ (if } V = 1_L, \alpha_1, \ldots, \alpha_k = 0_L).$$

Using the transformation function ϕ, we get: $\phi(1) = \mathbf{1}, \phi(0.9) = \mathbf{0.2}, \phi(0.8) = \mathbf{0.001}, \phi(0.5) = \mathbf{10^{-10}}, \phi(0.4) = \mathbf{10^{-30}}, \phi(0.1) = \mathbf{10^{-91}}$.

We obtain the transformed conditional distributions by normalizing on each node and according to (R'). For instance:

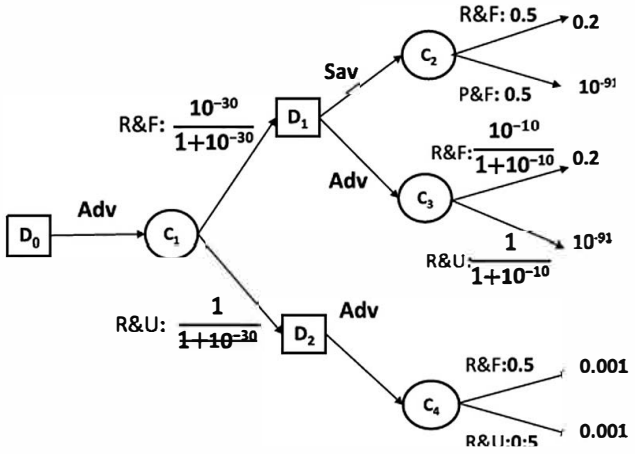

Fig. 4. Transformed probabilistic decision tree of possibilistic decision tree of (counter) – Example 4.

for node C_1, $\phi_1(10^{-30}) = \frac{10^{-30}}{1+10^{-30}}$ and $\phi_1(1) = \frac{1}{1+10^{-30}}$,
for node C_2, $\phi_2(1) = \frac{1}{1+1}$ and $\phi_2(1) = 0.5$,
for node C_3, $\phi_3(10^{-10}) = \frac{10^{-10}}{1+10^{-10}}$ and $\phi_3(1) = \frac{1}{1+10^{-10}}$,
for node C_4, $\phi_4(1) = 0.5$ and $\phi_4(1) = 0.5$.
Hence, we get the probabilistic decision tree presented in Fig. 4.

The construction is a little more complex if we consider the $\succeq_{lmin(lmax)}$ comparison, where the utility degrees are not directly compared to possibility degrees π but to degrees $1 - \pi$. However it is possible to rely on the results obtained for the optimistic case, since the optimistic and pessimistic utilities are dual of each other.

Proposition 5. *Let DT^{inv} be the tree obtained from DT by using utility function $u' = 1 - u$ on leaves. It holds that: $u_{pes,DT}(\delta) \geq u_{pes,DT}(\delta')$ iff $u_{opt,DT^{inv}}(\delta') \geq u_{opt,DT^{inv}}(\delta)$.*

As a consequence, we build an EU-based equivalent of $\succeq_{lmin(lmax)}$, denoted $\succeq_{EU_{pes}}$, by replacing each possibility distribution π_i in DT by the probability distribution $\phi_i \circ \pi_i$, as for the optimistic case and each utility degree u by$\phi(1) - \phi(u)$. It is then possible to show that:

Proposition 6. $\delta \succeq_{lmin(lmax)} \delta'$ *iff* $\delta \succeq_{EU_{pes}} \delta', \forall(\delta, \delta') \in \Delta$.

Propositions 4 and 6 show that lexicographic-comparisons have a probabilistic interpretation — actually, using big-stepped probabilities and utilities. This result comfort the idea, first proposed by [38] and then by [26], of a bridge between qualitative approaches and probabilistic ones, through the notion of big-stepped probabilities [38,39]. We make here a step further by identifying transformations that support sequential decision making.

Beyond this theoretical argument, this result suggests an alternative algorithm for the optimization of $lmax(lmin)$ (resp. $lmin(lmax)$): simply transform the possibilistic decision tree into a probabilistic one and use a classical, EU-based algorithm of dynamic programming. In a perfect world, both approaches solve the problem in the same way and provide the same optimal policies — the difference being that the first one is based on the comparison of matrices, the second one on expected utilities in $\mathbb{R}^+$. The point is that the latter handles very small numbers because of big-stepped probabilities/utilities; then either the program is based on an explicit handling of small numbers, and proceeds just like the matrix-based comparison, or it lets the programming language handle these numbers in its own way — and, given the precision of the computation, provides approximations.

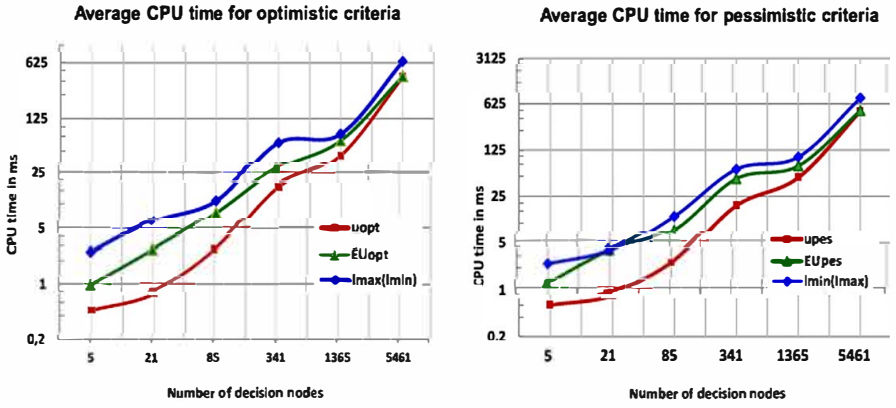

Fig. 5. Average CPU time (in ms) for $h = 2$ to 7.

6. Experiments

We have three criteria for each of the pessimistic and optimistic approaches: the basic possibilistic ones, the lexicographic refinements, and the EU approximations of the latter presented in Section 5. All of these criteria aim at solving the same kind of decision problems: sequential decision under possibilistic uncertainty, represented by a possibilistic decision tree or a finite-horizon possibilistic MDP. We compare the algorithms with two measures: the CPU time and a pairwise success rate: $Success_{\frac{A}{B}}$ is the percentage of policies provided by an algorithm optimizing criterion A that are optimal with respect to criterion B; for instance, the lower $Success_{\frac{u_{opt}}{lmax(lmin)}}$, the more important the drowning effect.

The algorithms corresponding to these criteria have been implemented in Java. The experiments have been performed on an Intel Core i5 processor computer (1.70 GHz) with 8 GB DDR3L of RAM.

6.1. Experimental results on possibilistic decision trees

The tests were performed on randomly generated complete binary decision trees, from $h = 2$ to $h = 7$. The first node is a decision node: at each decision level from the root ($i = 1$) to the last level ($i = 7$) the tree contains 2^{i-1} decision nodes. This means that with $h = 2$ (resp. 3, 4, 5, 6, 7), the number of decision nodes is equal to 5 (resp. 21, 85, 341, 1365, 5461) The utility values are sampled uniformly and randomly in the set $L = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Conditional possibilities relative to chance nodes are normalized, one edge having possibility one and the possibility degree of the other being uniformly sampled in L. For each value of h, 100 DTs are generated. As to the EU-based approaches, the transformation function depends on the horizon h and the branching factor b (here $b = 2$). We used $\phi(1_L) = 1$, $\phi(\alpha_i) = \frac{\phi(\alpha_{i+1})^{h+1}}{b^h * 1.1}$, each ϕ_j being obtained by normalization of ϕ on C_j.

Fig. 5 presents the average execution CPU time for the six criteria (the three optimistic ones and the three pessimistic ones). We observe that, whatever the optimized criterion, the CPU time increases linearly with respect to the number of decision nodes, which is in line with what we could expect. Furthermore, it remains affordable with big trees: the maximal CPU time is lower than 3125 ms for the DTs with 5461 decision nodes. It appears that u_{opt} is always faster than EU_{opt}, which is 1.5 or 2 times faster than $lmax(lmin)$. The same conclusions are drawn when comparing $lmin(lmax)$ to u_{pes} and EU_{pes}. These results are easy to explain: (i) the manipulation of matrices is more expensive than the one of numbers and (ii) the handling of numbers by min and max operations is faster than sum-product manipulations of small numbers.

As to the success rate, the results are described in Fig. 6. The percentage of policies optimal for u_{opt} (resp. for u_{pes}) that are also optimal for $lmax(lmin)$ (resp. $lmin(lmax)$) is never more than 82%, and decreases when the horizon increases: the drowning effect is not negligible and increases with the length of the trajectories. Moreover EU_{opt} (resp. EU_{pes}) does not perform well as an approximation of $lmax(lmin)$ (resp. $lmin(lmax)$): the percentage of

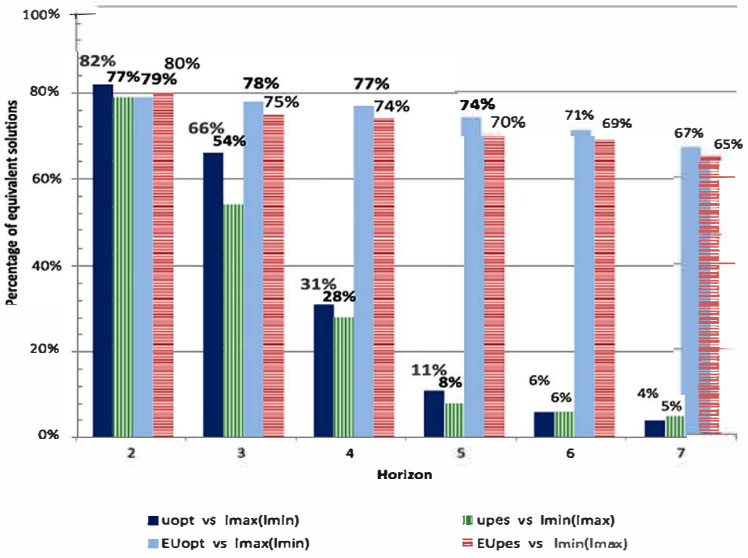

Fig. 6. Success rate for $h = 2$ to 7.

policies optimal for the former which are also optimal for the latter is lower than 80% in all cases, and decreases when h increases.

This is easily explained by the fact that the probabilities are small numbers and tend to 0 when the length of the branches (and thus the number of factors in the products) increase, as suggested in Section 5, leading to numerical approximations due to machine precision.

These experiments conclude in favor of the lexicographic refinements in their full definition — their approximation by expected utilities are comparable in terms of CPU efficiency but not precise enough. The EU criteria nevertheless offer a suitable approximation than u_{opt} and u_{pes} when space is limited (or when h increases).

6.2. *Experimental results on possibilistic Markov decision processes*

We propose to compare the performance of Bounded Lexicographic Backward Induction (BL-BI) as an approximation of (unbounded) Lexicographic Backward Induction (UL-BI). We also compare it to the classical Backward Induction algorithms classically used to for pessimistic and optimistic utilities (BI-u_{opt} and BI-u_{pes}) in randomly generated possibilistic MDPs for $h = 2$ to $h = 7$. In each stage, the MDP contains 20 states and the number of actions in each state is equal to 4. The output of each action is a distribution on two states randomly sampled (i.e., the branching factor is equal to 2). The utility values are uniformly randomly sampled in the set $L = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1\}$. Conditional possibilities relative to decisions should be normalized. To this end, one edge receives the possibility degree 1 and the possibility degree of the remaining ones is uniformly sampled in L. For each value of h, 100 possibilistic MDPs are generated.

Fig. 7 presents the average execution CPU time for the three algorithms. We observe that the CPU time increases linearly with respect to the horizon for $BI - u_{opt}$ and $BI - u_{pes}$. It seems to be also the case for BL-BI. On the other hand, it increases exponentially for UL-BI. We also observe that the CPU time of BL-BI is affordable, with a maximal value of 625 ms for the MDPs with 20 states when $l = 20$ and $h = 5$. Unsurprisingly, we can check that the BL-BI is faster than UL-BI especially when the horizon increases: the manipulation of $l \times (h + 1)$-matrices is less expensive than the one of full matrices. The saving increases with the horizon.

As to the success rate, the results for the optimistic case are described in Fig. 8. The percentage of optimal policies for u_{opt} that are also optimal for $lmax(lmin)$ when considering the whole matrices is never more than 60%, and decreases when the horizon increases. Indeed, if we take an arbitrary optimistic optimal policy, the higher the problem size the lower its chance of being lexicographically optimal. We observe that the drowning effect increases with the length of the trajectories.

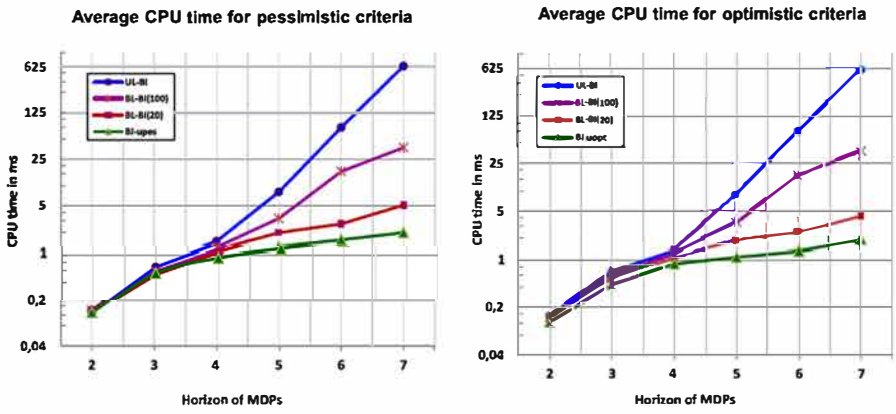

Fig. 7. Average CU time (in ms) for $h = 2$ to $h = 7$.

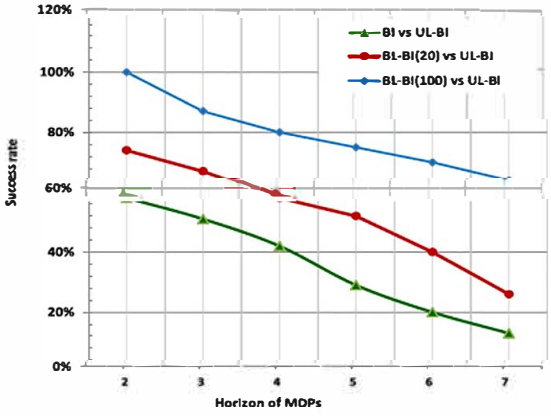

Fig. 8. Success rate for optimistic criteria.

It also appears that $BL\text{-}BI$ provides a very good approximation for reasonable values of l. Of course, the greater l the greater the quality of the approximation. When $l = 100$ $BL\text{-}BI$ provides a lexi optimal strategy in about 80% of cases. Moreover, even when the success rate of $BL\text{-}BI$ decreases (when h increases), the quality of approximation is still good: when h $l = 100$ never less than 70% of the strategies returned are lexi optional.

These experiments conclude in favor of bounded backward induction: the policies it returns are lexi — optimal policies in terms of quality for high l while it is much faster than the unbounded version.

Its approximated solutions are comparable with the optimal policy in terms of quality for high l and increase when l increase, while it is much faster than the unbounded version.

7. Conclusion

This article has proposed an extension to sequential decision problems of the lexicographic refinements of qualitative utilities proposed by [26], namely to finite horizon possibilistic decision trees and possibilistic Markov decision processes. On the theoretical side, this work generalizes to sequential problems the links established in [26] between possibilistic utilities and expected utilities. On the practical one, it allows to overcome the drowning effect at work in possibilistic decision trees and MDPs and provides dynamic programming algorithms for calculating lexicographically optimal policies.

Notice that the lexicographic refinements studied in this paper perform better that the refinement of binary possibilistic utilities (BPU) proposed in [16] for Binary Possibilistic Utilities and as a particular case, to classical, optimistic and pessimistic, possibilistic utilities. In [16]'s treatment, two similar trajectories of the same policy are merged. The resulting criterion thus suffers from a drowning effect and does not satisfy strict monotonicity: as such, it cannot be

represented by an EU-based criterion which "counts" trajectories (weighted by their probabilities). We actually refine [16]'s criterion. Incorporating our lexicographic refinements in BPU would lead to a more powerful refinement and suggests a probabilistic interpretation of efficient BPU. It also leads to new planning algorithms that are more decisive than their original counterparts.

The extensions of this work are twofold. First, we shall study undefined/infinite horizon problems — typically infinite-horizon possibilistic MDPs — and propose Lexicographic Value Iteration and Policy Iteration algorithms, that we already begun to address in [40] but it is beyond the scope of the present paper. As far as the infinite horizon case is concerned, other types of lexicographic refinements shall also 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.

The second perspective of this work, not unrelated, is to develop simulation-based algorithms for finding lexicographic solutions to MDPs. Reinforcement Learning algorithms [41] allow to solve large size MDPs by making use of simulated trajectories of states and actions. It is not immediate to develop RL algorithms for possibilistic MDPs, since no unique stochastic transition function corresponds to a possibility distribution [42]. However, uniform simulation of trajectories (with random choice of actions) may be used to generate an approximation of the possibilistic decision tree (provided that both transition possibilities and utility of the leaf are given with the simulated trajectory). So, interleaving simulations and lexicographic dynamic programming may lead to RL-type algorithms for approximating lexicographically optimal policies for (large) possibilistic MDPs.

Appendix. Proofs of propositions

Proof of Proposition 1

- **Completeness**. It is a consequence of the completeness of $\succeq_{lmax}$ and $\succeq_{lmin}$ [43].
- **Transitivity**. We prove that $\succeq_{lmax(lmin)}$ is transitive. The proof relies on the transitivity of $\succeq_{lmin}$. Let us consider three policies, δ, δ' and δ'' and assume $\delta \succeq_{lmax(lmin)} \delta'$ and $\delta' \succeq_{lmax(lmin)} \delta''$. Since $\delta \succeq_{lmax(lmin)} \delta'$ and $\delta' \succeq_{lmax(lmin)} \delta''$, then we are in either of the following cases:
  1. $\forall i, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)} \sim_{lmin} \tau''_{\lambda(i)}$. This happens when $\delta \sim_{lmax(lmin)} \delta' \sim_{lmax(lmin)} \delta''$. And then by transitivity of $\succeq_{lmin}$, we have $\forall i, \tau_{\lambda(i)} \sim_{lmin} \tau''_{\lambda(i)} \Leftrightarrow \delta \sim_{lmax(lmin)} \delta''$.
  2. When either $\delta \succ_{lmax(lmin)} \delta'$ or $\delta' \succ_{lmax(lmin)} \delta''$, then by definition of $\succeq_{lmax(lmin)}$, there exists i^*, such that, (a) $\forall i < i^*, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)} \sim_{lmin} \tau''_{\lambda(i)}$, (b) $\tau_{\lambda(i^*)} \succeq_{lmin} \tau'_{\lambda(i^*)} \succeq_{lmin} \tau''_{\lambda(i^*)}$ and (c), either $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ or $\tau'_{\lambda(i^*)} \succ_{lmin} \tau''_{\lambda(i^*)}$, or both. Then, once again by transitivity of $\succeq_{lmin}$, $\tau_{\lambda(i^*)} \succ_{lmin} \tau''_{\lambda(i^*)}$. So, $\delta \succ_{lmax(lmin)} \delta''$.

  So, points 1 and 2 imply, together, that $\delta \succeq_{lmax(lmin)} \delta'$ and $\delta' \succeq_{lmax(lmin)} \delta''$ imply $\delta \succeq_{lmax(lmin)} \delta''$.
- Similarly, it can be checked that $\succeq_{lmin(lmax)}$ is transitive. Let us consider three policies, δ, δ' and δ'' and assume $\delta \succeq_{lmin(lmax)} \delta'$ and $\delta' \succeq_{lmin(lmax)} \delta''$. Since $\delta \succeq_{lmin(lmax)} \delta'$ and $\delta' \succeq_{lmin(lmax)} \delta''$, then we are in either following cases:
  1. $\forall i, \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)} \sim_{lmax} \tau''_{\sigma(i)}$. This happens when $\delta \sim_{lmin(lmax)} \delta' \sim_{lmin(lmax)} \delta''$. And then by transitivity of $\succeq_{lmax}$, we have $\forall i, \tau_{\sigma(i)} \sim_{lmax} \tau''_{\sigma(i)} \Leftrightarrow \delta \sim_{lmin(lmax)} \delta''$.
  2. When either $\delta \succ_{lmin(lmax)} \delta'$ or $\delta' \succ_{lmin(lmax)} \delta''$, then by definition of $\succeq_{lmin(lmax)}$, there exists i^*, such that, (a) $\forall i < i^*, \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)} \sim_{lmax} \tau''_{\sigma(i)}$, (b) $\tau_{\sigma(i^*)} \succeq_{lmax} \tau'_{\sigma(i^*)} \succeq_{lmax} \tau''_{\sigma(i^*)}$ and (c), either $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ or $\tau'_{\sigma(i^*)} \succ_{lmax} \tau''_{\sigma(i^*)}$, or both. Then, once again by transitivity of $\succeq_{lmax}$, $\tau_{\sigma(i^*)} \succ_{lmax} \tau''_{\sigma(i^*)}$. So, $\delta \succ_{lmin(lmax)} \delta''$.

  So, points 1 and 2 imply, together, that $\delta \succeq_{lmin(lmax)} \delta'$ and $\delta' \succeq_{lmin(lmax)} \delta''$ imply $\delta \succeq_{lmin(lmax)} \delta''$.
- **Refinement**. We prove that $\succeq_{lmax(lmin)}$ refines $\succeq_{u_{opt}}$. Let us consider two policies δ and δ'. If $u_{opt}(\delta) > u_{opt}(\delta')$

$$\Leftrightarrow \quad \max_{\tau \in \delta} \min\{\min_{\pi_k \in \pi_\tau} \pi_k, u_\tau\} > \max_{\tau' \in \delta'} \min\{\min_{\pi'_k \in \pi_{\tau'}} \pi'_k, u'_h\}$$

$$\Rightarrow \max_{\tau \in \delta} \min(\pi_1, \ldots, \pi_h, u_\tau) > \max_{\tau' \in \delta'} \min(\pi'_1, \ldots, \pi'_h, u'_h).$$

Since $\min(\pi_1, \ldots, \pi_h, u_\tau) > \min(\pi'_1, \ldots, \pi'_h, u'_h)$

$$\Rightarrow (\pi_1, \ldots, \pi_h, u_\tau) \succ_{lmin} (\pi'_1, \ldots, \pi'_h, u'_h)$$

(leximin ordering refines min ordering), so, $\tau_{\lambda(1)} \succ_{lmin} \tau'_{\lambda(1)} \Rightarrow \delta \succ_{lmax(lmin)} \delta'$ where $\tau_{\lambda(1)}$ (resp. $\tau'_{\lambda(1)}$) is the best trajectory of δ (resp. δ') according to $\succeq_{lmin}$.
So by the definition of $\succeq_{lmax(lmin)}$ we have $\delta \succ_{lmax(lmin)} \delta'$. And we deduce that $\succeq_{lmax(lmin)}$ refines $\succeq_{u_{opt}}$.

- We show in the same way that $\succeq_{lmin(lmax)}$ refines $\succeq_{u_{pes}}$. Let us consider two policies δ and δ'. $u_{pes}(\delta) > u_{pes}(\delta')$

$$\Leftrightarrow \quad \min_{\tau\in\delta} \max\{\min_{\pi_k\in\pi_\tau}(1-\pi_k), u_\tau\} > \min_{\tau'\in\delta'} \max\{\min_{\pi'_k\in\pi_{\tau'}}(1-\pi'_k), u'_h\}$$

$$\Leftrightarrow \quad \min_{\tau\in\delta} \max\{\min(1-\pi_1,\ldots,1-\pi_h,u_\tau)) > \min_{\tau'\in\delta'} \max(1-\pi'_1,\ldots,1-\pi'_h,u'_h).$$

Since $\max(1-\pi_1,\ldots,1-\pi_h,u_\tau) > \max(1-\pi'_1,\ldots,1-\pi'_h,u'_h)$

$$\Rightarrow (1-\pi_1,\ldots,1-\pi_h,u_\tau)) \succ_{lmax} (1-\pi'_1,\ldots,1-\pi'_h,u'_h)$$

(leximax ordering refines max ordering). Then $\tau_{\sigma(1)} \succ_{lmax} \tau'_{\sigma(1)} \Rightarrow \delta \succ_{lmin(lmax)} \delta'$ where $\tau_{\sigma(1)}$ (resp. $\tau'_{\sigma(1)}$) is the worst trajectory of δ (resp. δ') according to $\succeq_{lmax}$. So, by definition of $\succeq_{lmin(lmax)}$ we have $\delta \succ_{lmin(lmax)} \delta'$.

Proof of Proposition 2

(i) We first prove that $\succeq_{lmax(lmin)}$ and $\succeq_{lmin(lmax)}$ are strictly monotonic. Note that $\forall C_j \in \mathcal{N}_C, \forall D_i \in Succ(C_j), \delta, \delta' \in \Delta_{D_i}, \delta'' \in \Delta_{Succ(C_j)\setminus D_i}$, the trajectories of $\delta + \delta''$ are composed of two disjoint sets of trajectories: One for δ and one for δ''. The same holds for $\delta' + \delta''$. Then, note that adding or removing identical trajectories to two sets of trajectories does not change the $\succeq_{lmax(lmin)}$ or the $\succeq_{lmin(lmax)}$ ordering between these two sets.
To be more precise, assume, for example, that $\delta \succ_{lmax(lmin)} \delta'$. Then, $\exists i^*, \forall i < i^*, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$. The trajectories corresponding to δ'' are composed of trajectories which rank before $\tau_{\lambda(i^*)}$, and after $\tau_{\lambda(i^*)}$. Obviously, the ones that rank before $\tau_{\lambda(i^*)}$ are added to both lists of trajectories, and thus simply delay i^* while not inducing a new preference. And the ones that rank after $\tau_{\lambda(i^*)}$ are not taken into consideration in the comparison of $\delta + \delta''$ and $\delta' + \delta''$. In the same way, by definition of $\succeq_{lmin(lmax)}$ we get $\delta \succ_{lmin(lmax)} \delta'$ i.e., $\exists i^*, \forall i < i^*, \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ and $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$. The same result as for $\tau_{\lambda(i^*)}$ applies to $\tau_{\sigma(i^*)}$. Thus, $\succeq_{lmax(lmin)}$ and $\succeq_{lmin(lmax)}$ are strictly monotonic.

(ii) Now we prove that $\succeq_{lmax(lmin)}$ satisfy the principle of Pareto efficiency. So, suppose that $\delta \succeq_{lmax(lmin)} \delta'$. Two cases arise:
- if $\forall i, \quad \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and then $\delta \sim_{lmax(lmin)} \delta'$,
- if $\exists i^*$, s.t. $\forall i < i^*, \quad \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$. Then, $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ implies that there exist a pair of different $(\beta_{i^*,k}, \beta'_{i^*,k})$, where $\beta_{i^*,k}$ (resp. $\beta'_{i^*,k}$) is an element of $\tau_{\lambda(i^*)}$ (resp. $\tau'_{\lambda(i^*)}$), that determines the best policy. Here we get $\beta_{i^*,k} > \beta'_{i^*,k}$ i.e., $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ and thus $\delta \succ_{lmax(lmin)} \delta'$.

In summary, if we have $\delta \succeq_{lmax(lmin)} \delta'$ and $\exists i^*$, s.t. $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$ we get $\delta \succ_{lmax(lmin)} \delta'$ which expresses exactly the principle of Pareto efficiency in the case $\succeq_{lmax(lmin)}$.

(iii) Let us prove $\succeq_{lmin(lmax)}$ satisfy the principle of Pareto efficiency. When considering the $\succeq_{lmin(lmax)}$ order, the same kind of result as for the $\succeq_{lmax(lmin)}$ can be obtained.
So, suppose that $\delta \succeq_{lmin(lmax)} \delta'$. Two cases arise:
- if $\forall i, \quad \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ and then $\delta \sim_{lmin(lmax)} \delta'$,
- if $\exists i^*$, s.t. $\forall i < i^*, \quad \tau_{\sigma(i)} \sim_{lmax} \tau'_{\sigma(i)}$ and $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$. Then, $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ implies that there exist a pair of different $(\beta_{i^*,k}, \beta'_{i^*,k})$, where $\beta_{i^*,k}$ (resp. $\beta'_{i^*,k}$) is an element of $\tau_{\sigma(i^*)}$ (resp. $\tau'_{\sigma(i^*)}$), determining the best policy. We get $\beta_{i^*,k} > \beta'_{i^*,k}$ i.e., $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ and thus $\delta \succ_{lmin(lmax)} \delta'$.

In summary, if we have $\delta \succeq_{lmin(lmax)} \delta'$ and $\exists i^*$, s.t. $\tau_{\sigma(i^*)} \succ_{lmax} \tau'_{\sigma(i^*)}$ we get $\delta \succ_{lmin(lmax)} \delta'$ which expresses exactly the principle of Pareto efficiency in the case of $\succeq_{lmin(lmax)}$.

Proof of Proposition 3

For any transformation function ϕ s.t. $\forall(\alpha,\alpha') \in L, \quad \alpha > \alpha'$ it holds that $\phi^-(\alpha)^{h+1} > b^h\phi^-(\alpha')$. Let δ and δ' be two strategies. Assume that $u_{opt}(\delta) = \alpha > u_{opt}(\delta') = \alpha'$ and let us show that $EU_{opt}(\delta) > EU_{opt}(\delta')$.

- $u_{opt}(\delta) = \alpha \Rightarrow \exists \tau^* = (\pi_{j_0}(x_{i_1}), \ldots, \pi_{j_{h-1}}(x_{i_h}), u(x_{i_h}))$ in δ s.t.

  $$\min(\pi_{j_0}(x_{i_1}), \ldots, \pi_{j_{h-1}}(x_{i_h}), u(x_{i_h})) \geq \alpha.$$

  Then $EU_{opt}(\delta) = \sum_{\tau}(\prod_{k=1}^{h} \phi_k(\pi_{j_{k-1}}(x_{i_k})) * \phi(\mu(x_{i_k})))$.
  By keeping only trajectory τ^* in the sum, we get:

  $$EU_{opt}(\delta) \geq \prod_{k=1}^{h} \phi_k(\alpha) * \phi(\alpha).$$

  Since[7] $\phi^-(\alpha) \leq \phi_k(\alpha), \forall k, \alpha > \alpha'$, $EU_{opt}(\delta) \geq \prod_{k=1}^{h} \phi^-(\alpha) * \phi(\alpha)$.
  Then, since $\phi^-(\alpha) \leq \phi(\alpha)$, $EU_{opt}(\delta) \geq \prod_{k=1}^{h} \phi^-(\alpha) * \phi^-(\alpha)$.
  Thus, we get:

  $$EU_{opt}(\delta) \geq \phi^-(\alpha)^{h+1} \tag{7}$$

- $u_{opt}(\delta') = \alpha' \Rightarrow \forall \tau,\ \min(\pi_{j'_0}(x_{i_1}), \ldots, \pi_{j'_{h-1}}(x_{i'_h}), u(x_{i'_h})) \leq \alpha'$.
  We have $EU_\tau = \prod_{k=1}^{h} \phi_k(\alpha') * \phi(\alpha') \leq \prod_{k=1}^{h} \phi^+(\alpha') * \phi(\alpha') \leq \prod_{k=1}^{h} \phi^+(\alpha') * \phi^+(\alpha')$, since $\phi^+(\alpha) \geq \phi(\alpha) \geq \phi_k(\alpha), \forall k, \alpha > \alpha'$.
  Then, $EU_\tau \leq \phi^+(min(\pi_{j'_0}(x_{i_1}), \ldots, \pi_{j'_{h-1}}(x_{i'_h}), u(x_{i'_h}))) \prod_{k=1}^{h} \phi^+(\alpha') * \phi^+(\alpha') \leq \phi^+(min(\pi_{j'_0}(x_{i_1}), \ldots, \pi_{j'_{h-1}}(x_{i'_h}), u(x_{i'_h}))) \leq \phi^+(\alpha')$ for b^h trajectory of δ'. Thus,

  $$EU_{opt}(\delta') \leq b^h.\phi^+(\alpha'). \tag{8}$$

Finally, using (1), (2) and (R'), we get $u_{opt}(\delta) > u_{opt}(\delta') \Rightarrow EU_{opt}(\delta) > EU_{opt}(\delta')$.

Proof of Proposition 4

For the sake of notational simplicity, we will associate any trajectory τ (resp. τ') with the vector $\vec{\alpha} = (\alpha_1, \ldots, \alpha_k, \ldots, \alpha_{h+1})$ (resp. $\vec{\alpha'} = (\alpha'_1, \ldots, \alpha'_k, \ldots, \alpha'_{h+1})$) consisting in reordering $(\pi_1, \ldots, \pi_h, u_\tau)$ (resp. $(\pi'_1, \ldots, \pi'_h, u'_\tau)$) in increasing order. Obviously, $\tau \succeq_{lmin} \tau'$ iff $\vec{\alpha} \succeq_{lmin} \vec{\alpha'}$. Note that $\delta \succeq_{lmax(lmin)} \delta'$ iff either

1. $\forall i, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ or
2. $\exists i^*, \forall i \leq i^*, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$ and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$.

Note the following facts concerning pairs of trajectories, (τ, τ'):

1. $\tau \sim_{lmin} \tau' \Leftrightarrow \prod_{k=1}^{h+1} \phi_k(\alpha_k) = \prod_{k=1}^{h+1} \phi_k(\alpha'_k)$, where $\phi_{k+1} \equiv \phi$ since $\tau \sim_{lmin} \tau' \leftrightarrow \vec{\alpha} = \vec{\alpha'}$.
   Then: $\delta \cong_{lmax(lmin)} \delta' \Leftrightarrow \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}, \forall i$.

   $$\Rightarrow \sum_{t}(\prod_{k=1}^{h+1} \phi_k(\alpha_k) = \sum_{t}(\prod_{k=1}^{h+1} \phi_k(\alpha'_k)$$
   $$\Leftrightarrow EU_{opt}(\delta) = EU_{opt}(\delta').$$

   Thus $\delta \cong_{lmax(lmin)} \delta'$

   $$\Rightarrow EU_{opt}(\delta) = EU_{opt}(\delta').$$

2. If $\delta >_{lmax(lmin)} \delta' \Leftrightarrow \exists i^*\ s.t.\ \tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)}$, then $\exists j^*, \forall j < j^*, \alpha_j = \alpha'_j$ and $\alpha_{j^*} > \alpha'_{j^*}$. Then, let us compare the product of transformed possibilities/utilities along τ and τ':

[7] $\phi^-(\alpha) \leq \phi_k(\alpha) \leq \phi(\alpha)$.

$$\prod_{k=1}^{h+1}\frac{\phi(\alpha_k)}{\phi(\alpha'_k)} = \prod_{k=j^*}^{h+1}\frac{\phi(\alpha_k)}{\phi(\alpha'_k)}$$

$$\geq \frac{\phi(\alpha_{j^*})}{\phi(\alpha'_{j^*})} * \frac{\phi(\alpha_{j^*})^{h-j^*}}{\phi(\phi(1_L)^{h-j^*})} \tag{9}$$

(lower and upper bounds on trajectories degrees)

$$\geq \frac{\phi(\alpha_{j^*})^{h-j^*+1}}{\phi(\alpha'_{j^*})} \qquad (\phi(1_L) \leq 1)$$

$$\geq \frac{\phi(\alpha_{j^*})^{h}}{\phi(\alpha'_{j^*})} \qquad (\phi(\alpha_{j^*}) \leq 1)$$

$$> b^h. \qquad \text{(From R)}$$

Then, since trajectories are ordered along $\succeq_{lmax(lmin)}$, $\tau'_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i)}, \forall i > i^*$, and since there are no more than b^h such trajectories $\tau'_{\lambda(i)}$, we get that $EU_{opt}(\delta) > EU_{opt}(\delta')$. Thus $\delta \succ_{lmax(lmin)} \delta' \Rightarrow EU_{opt}(\delta) > EU_{opt}(\delta')$.

So, we have just proved that $\delta \succeq_{lmax(lmin)} \delta' \Rightarrow \delta \succeq_{EU_{opt}} \delta'$. Thus, $\succeq_{lmax(lmin)}$ is equivalent to $\succeq_{EU_{opt}}$.

Proof of Proposition 5

Let l_δ (resp. $l_{\delta'}$) be the equivalent simple lottery of the compound lottery representing the policy δ. $l_\delta = \langle \pi_1/u_1, ..., \pi_i/u_i, ..., \pi_p/u_p \rangle$ (resp. $l_{\delta'} = \langle \pi'_1/u'_1, ..., \pi'_i/u'_i, ..., \pi'_p/u'_p \rangle$) i.e., $\pi_i = \pi(u_i)$ (resp. $\pi'_i = \pi(u'_i)$) is the possibility that the policy leads to the outcome utility u_i (resp. u'_i).

Let us show that $u_{pes,\mathcal{DT}}(\delta) \geq u_{pes,\mathcal{DT}}(\delta') \Leftrightarrow u_{opt,\mathcal{DT}^{inv}}(\delta) \leq u_{opt,\mathcal{DT}^{inv}}(\delta')$ where $u_{pes,\mathcal{DT}}(\delta)$ denotes the pessimistic utility of δ when considering the original decision tree $\mathcal{DT}$ and $u_{opt,\mathcal{DT}^{inv}}(\delta)$ denotes the optimistic utility of δ when considering the decision tree $\mathcal{DT}^{inv}$, obtained from $\mathcal{DT}$ by using utility function $u' = 1 - u$.

$$\begin{aligned}
& u_{pes,\mathcal{DT}}(\delta) \;\geq\; u_{pes,\mathcal{DT}}(\delta') \\
\Leftrightarrow\quad & u_{pes,\mathcal{DT}}(l_\delta) \;\geq\; u_{pes,\mathcal{DT}}(l_{\delta'}) \\
\Leftrightarrow\quad & \min\max(n(\pi_i), u_i) \;\geq\; \min\max(n(\pi'_i), u'_i) \\
\Leftrightarrow\quad & n(\min\max(n(\pi'_i), u'_i)) \;\geq\; n(\min\max(n(\pi_i), u_i)) \\
\Leftrightarrow\quad & \max n(\max((n(\pi'_i), u'_i))) \;\geq\; \max n(\max((n(\pi_i), u_i))) \\
\Leftrightarrow\quad & \max\min(n(n(\pi'_i), n(u'_i))) \;\geq\; \max\min(n(n(\pi_i), n(u_i))) \\
\Leftrightarrow\quad & \max\min(\pi'_i, n(u'_i)) \;\geq\; \max\min(\pi_i, n(u_i)) \\
\Leftrightarrow\quad & u_{opt,\mathcal{DT}^{inv}}(l_{\delta'}) \;\geq\; u_{opt,\mathcal{DT}^{inv}}(l_\delta) \\
\Leftrightarrow\quad & u_{opt,\mathcal{DT}^{inv}}(\delta') \;\geq\; u_{opt,\mathcal{DT}^{inv}}(\delta).
\end{aligned}$$

Proof of Proposition 6

It is sufficient to show that:

$$\delta \succeq^{\mathcal{DT}}_{lmax(lmin)} \delta' \Leftrightarrow \delta' \succeq^{\mathcal{DT}^{inv}}_{lmin(lmax)} \delta, \tag{10}$$

where $\succeq^{\mathcal{DT}^{inv}}_{lmin(lmax)}$ is the pessimistic lexicographic comparison of policies in the decision tree where the utilities of all leaves have been reversed ($\bar{u}(N) = 1 - u(N)$).

For any two policies δ and δ':

- if $\delta \succ^{\mathcal{DT}}_{lmax(lmin)} \delta'$ then $\exists i^*, \forall i \leq i^*, \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)}$and $\tau_{\lambda(i^*)} \succ_{lmin} \tau'_{\lambda(i^*)} \Leftrightarrow (\pi_1, \ldots, \pi_h, u_\tau) \succ_{lmin} (\pi'_1, \ldots, \pi'_h, u'_\tau)$.

Now let us invert each degree in both trajectories, we get:

$$((1-\pi_1),\ldots,(1-\pi_h),(1-u_\tau)) \prec_{lmax} ((1-\pi'_1),\ldots,(1-\pi'_h),(1-u'_h))$$
$$\Leftrightarrow \quad ((1-\pi_1),\ldots,(1-\pi_h)),\bar{u}_h) \prec_{lmax} ((1-\pi'_1),\ldots,(1-\pi'_h),\bar{u}'_h)$$

i.e., $\tau_{\sigma(i^*)} \prec_{lmax} \tau'_{\sigma(i^*)}$ which is the $\succeq_{lmax}$ relation when considering $\mathcal{DT}^{inv}$. We get $\delta' \succeq^{\mathcal{DT}^{inv}}_{lmin(lmax)} \delta$.

- If $\delta \cong^{\mathcal{DT}}_{lmax(lmin)} \delta'$ then $\forall i,\ \tau_{\lambda(i)} \sim_{lmin} \tau'_{\lambda(i)} \Leftrightarrow$

$$(\pi_1,\ldots,\pi_h,u_\tau) \cong_{lmin} (\pi'_1,\ldots,\pi'_h,u'_\tau).$$

If we inverse each degree in both trajectories, we get:

$$((1-\pi_1),\ldots,(1-\pi_h),(1-u_\tau)) \cong_{lmax} ((1-\pi'_1,\ldots,(1-\pi'_h),(1-u'_\tau))$$
$$\Leftrightarrow \quad ((1-\pi_1),\ldots,(1-\pi_h),\bar{u}_h) \cong_{lmax} ((1-\pi'_1),\ldots,(1-\pi'_h),\bar{u}'_h)$$

i.e., $\tau_{\sigma(i)} \cong_{lmax} \tau'_{\sigma(i)}$. We get $\delta' \cong^{\mathcal{DT}^{inv}}_{lmin(lmax)} \delta$.

Thus, $\succeq_{EU_{pes}}$ is equivalent to $\succeq_{lmin(lmax)}$.

References

[1] K. Bauters, W. Liu, L. Godo, Anytime algorithms for solving possibilistic MDPs and hybrid MDPs, in: Proceedings of FoIKS'2016, 2016, pp. 24–41.
[2] P. Weng, Axiomatic foundations for a class of generalized expected utility: algebraic expected utility, in: Proceedings of UAI'2006, 2006, pp. 520–527.
[3] F.C. Chu, J.Y. Halpern, Great expectations. Part I: on the customizability of generalized expected utility, in: Proceedings of IJCAI'2003, 2003, pp. 291–296.
[4] B. Bonet, H. Geffner, Arguing for decisions: a qualitative model of decision making, in: Proceedings of UAI'1996, 1996, pp. 98–105.
[5] D.J. Lehmann, Generalized qualitative probability: savage revisited, in: Proceedings of UAI'2005, 1996, pp. 381–388.
[6] J.V. Neumann, O. Morgenstern, Theory of Games and Economic Behavior, 1948.
[7] L.J. Savage, The Foundations of Statistics, Wiley, 1954.
[8] H. Raiffa, Decision Analysis: Introductory Lectures on Choices Under Uncertainty, Addison Wesley, 1968.
[9] M.L. Puterman, Markov Decision Processes, John Wiley and Sons, 1994.
[10] B. Szörényi, R. Busa-Fekete, P. Weng, E. Hüllermeier, Qualitative multi-armed bandits: a quantile-based approach, in: Proceedings of ICML'2015, 2015, pp. 1660–1668.
[11] H. Gilbert, P. Weng, Y. Xu, Optimizing quantiles in preference-based Markov decision processes, in: Proceedings of AAAI'2017, 2017, pp. 3569–3575.
[12] I. Montes, E. Miranda, S. Montes, Decision making with imprecise probabilities and utilities by means of statistical preference and stochastic dominance, Eur. J. Oper. Res. 234 (2014) 209–220.
[13] P. Weng, Markov decision processes with ordinal rewards: reference point-based preferences, in: Proceedings of ICAPS'2011, 2011, pp. 282–289.
[14] Y. Yue, J. Broder, R. Kleinberg, T. Joachims, The k-armed dueling bandits problem, J. Comput. Syst. Sci. 78 (5) (2012) 1538–1556.
[15] P. Giang, P.P. Shenoy, Two axiomatic approaches to decision making using possibility theory, Eur. J. Oper. Res. 162 (2005) 450–467.
[16] P. Weng, Qualitative decision making under possibilistic uncertainty: toward more discriminating criteria, in: Proceedings of UAI'2005, 2005, pp. 615–622.
[17] D. Dubois, H. Prade, R. Sabbadin, Decision-theoretic foundations of qualitative possibility theory, Eur. J. Oper. Res. 128 (2001) 459–478.
[18] L. Godo, A. Zapico, On the possibilistic-based decision model: characterization of preference relations under partial inconsistency, Appl. Intell. 14 (2001) 319–333.
[19] D. Dubois, L. Godo, H. Prade, A. Zapico, Making decision in a qualitative setting: from decision under uncertainty to case-based decision, in: Proceedings of KR'1998, 1998, pp. 594–605.
[20] D. Dubois, H. Prade, Possibility theory as a basis for qualitative decision theory, in: Proceedings of IJCAI'95, 1995, pp. 1925–1930.
[21] N. Ben Amor, H. Fargier, W. Guezguez, Possibilistic sequential decision making, Int. J. Approx. Reason. 55 (2014) 1269–1300.
[22] N. Drougard, F. Teichteil-Konigsbuch, J.-L. Farges, D. Dubois, Structured possibilistic planning using decision diagrams, in: Proceedings of AAAI'2014, 2014, pp. 2257–2263.
[23] N. Drougard, F. Teichteil-Königsbuch, J.-L. Farges, D. Dubois, Qualitative possibilistic mixed-observable MDPs, in: Proceedings of UAI'13, 2013, pp. 192–201.
[24] R. Sabbadin, Possibilistic Markov decision processes, Eng. Appl. Artif. Intell. 14 (2001) 287–300.
[25] R. Sabbadin, H. Fargier, J. Lang, Towards qualitative approaches to multi-stage decision making, Int. J. Approx. Reason. 19 (1998) 441–471.
[26] H. Fargier, R. Sabbadin, Qualitative decision under uncertainty: back to expected utility, Artif. Intell. 164 (2005) 245–280.

[27] N. Ben Amor, Z.E. Khalfi, H. Fargier, R. Sabbadin, Lexicographic refinements in possibilistic decision trees, in: Proceedings of ECAI'2016, 2016, pp. 202–208.
[28] N. Ben Amor, Z.E. Khalfi, H. Fargier, R. Sabbadin, Lexicographic refinements in possibilistic Markov decision processes: the finite horizon case, in: Proceedings of LFA'2016, 2016.
[29] L.J. Savage, The Foundations of Statistics, J. Wiley, New York, 1954, second revised ed. 1972.
[30] L. Zadeh, Fuzzy sets, Inf. Control 8 (1965) 338–353.
[31] G. Shafer, A Mathematical Theory of Evidence, Princeton University Press, 1976.
[32] L. Zadeh, Fuzzy sets as a basis for theory of possibility, Fuzzy Sets Syst. 1 (1978) 3–28.
[33] D. Dubois, H. Prade, Possibility Theory: An Approach to Computerized Processing of Uncertainty, Plenum Press, New York, 1988.
[34] L. Garcia, R. Sabbadin, Possibilistic influence diagrams, in: Proceedings of ECAI'06, 2006, pp. 372–376.
[35] R. Bellman, Dynamic Programming, Princeton University Press, 1957.
[36] R. Sabbadin, A possibilistic model for qualitative sequential decision problems under uncertainty in partially observable environments, in: Proceedings of UAI'1999, 1999, pp. 567–574.
[37] H. Moulin, Axioms of Cooperative Decision Making, Cambridge University Press, 1988.
[38] S. Benferhat, D. Dubois, H. Prade, Possibilistic and standard probabilistic semantics of conditional knowledge bases, J. Log. Comput. 9 (1999) 873–895.
[39] P. Snow, Diverse confidence levels in a probabilistic semantics for conditional logics, Artif. Intell. 113 (1999) 269–279.
[40] N. Ben Amor, Z.E. Khalfi, H. Fargier, R. Sabbadin, Efficient policies for stationary possibilistic Markov decision processes, in: Proceedings of ECSQARU'2017, 2017.
[41] R.S. Sutton, A.G. Barto, Reinforcement Learning: An Introduction, MIT Press, 1998.
[42] R. Sabbadin, Towards possibilistic reinforcement learning algorithms, in: Proceedings of IEEE Fuzzy Systems, 2001, pp. 404–407.
[43] D. Dubois, H. Fargier, H. Prade, Refinements of the maximin approach to decision-making in fuzzy environment, Fuzzy Sets Syst. 81 (1996) 103–122.