ECAI 2016
G.A. Kaminka et al. (Eds.)

doi:10.3233/978-1-61499-672-9-202

Lexicographic Refinements in Possibilistic Decision Trees

Nahla Ben Amor[1] and **Zeineb El Khalfi**[2] and **Hélène Fargier**[3] and **Régis Sabbadin**[4]

Abstract. Possibilistic decision theory has been proposed twenty years ago and has had several extensions since then. Because of the lack of decision power of possibilistic decision theory, several refinements have then been proposed. Unfortunately, these refinements do not allow to circumvent the difficulty when the decision problem is sequential. In this article, we propose to extend lexicographic refinements to possibilistic decision trees. We show, in particular, that they still benefit from an Expected Utility (EU) grounding. We also provide qualitative dynamic programming algorithms to compute lexicographic optimal strategies. The paper is completed with an experimental study that shows the feasibility and the interest of the approach.

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, 28, 7, 5, 16]). The usual paradigm of decision under uncertainty is based on the *Expected Utility (EU) model* [18, 23]. Its extensions to sequential decision making are *Decision Trees* (DT) [20] and *Markov Decision Processes* (MDP) [6, 19], where the uncertain effects of actions are represented by probability distributions.

When information about uncertainty cannot be quantified in a probabilistic way, possibilistic decision theory is a natural field to consider [14, 27, 12, 15, 10, 11, 15]. Qualitative decision theory is relevant, among other fields, for applications to planning under uncertainty, where a suitable *strategy* (i.e. a set of conditional or unconditional decisions) is to be found, starting from a qualitative description of the initial world, of the available decisions, of their (perhaps uncertain) effects and of the goal to reach (see [1, 3, 9, 8, 21, 22]).

Even though appealing for its ability to handle qualitative problems, possibilisitic decision theory suffers from an important drawback. Acts (and strategies in sequential problems) are compared through $\min$ and $\max$ operators, which leads to a *drowning effect*: 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, refinements of possibilistic decision criteria have been proposed in the non-sequential case [13, 27]. Some refinements have the very interesting property to remain qualitative while satisfying the properties of EU. But these refinements do not extend to sequential decision under uncertainty (in the context of the present work, to decision trees) where the drowning effect is also due to the reduction of compound possibilistic strategies into simple ones [13].

[1] LARODEC, Tunisie, email: nahla.benamor@gmx.fr
[2] LARODEC, Tunisie, IRIT, France, email: zeineb.khalfi@gmail.com
[3] IRIT, France, email: fargier@irit.fr
[4] INRA-MIAT, France, email: rsabbadin@toulouse.inra.fr

The present paper proposes lexicographic refinements that compare full strategies (and not simply their reductions) and provides a dynamic programming algorithm to compute a lexicographic optimal strategy. It is a technical challenge to establish results of equivalence between lexicographic refinements of utilities of strategies in possibilistic decision trees and EU-based criteria. We prove such results, which opens the way to define dynamic programming solutions or even reinforcement learning algorithms for possibilistic MDPs [26, 25], which would not suffer from the drowning effect.

The paper is structured as follows ; the next Section recalls some results about the comparison of strategies in possibilistic decision trees. In Section 3, we define lexicographic orderings that refine the possibilistic criteria. Section 4 then proposes a dynamic programming algorithm for the computation of lexi-optimal strategies. Section 5 shows that the lexicographic criteria can be represented by *infinitesimal* expected utilities. The last Section reports experiments highlighting the feasibility and interest of the approach[5].

2 Possibilistic decision trees

Decision trees provide an explicit modeling of sequential decision problems by representing, simply, all possible scenarios. The graphical component of a decision tree is a labelled graph $\mathcal{DT} = (\mathcal{N}, \mathcal{E})$. $\mathcal{N} = \mathcal{N}_D \cup \mathcal{N}_C \cup \mathcal{N}_U$ contains three kinds of nodes (see Figure 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.

For any node N, $Out(N)$ denotes its outgoing edges, $Succ(N)$ the set of its children nodes and $Succ(N, e)$ the child of N that is reached by edge $e \in Out(N)$. This tree represents a sequential decision problem as follows:

- Leaf nodes correspond to states of the world in which a utility is obtained (for the sake of simplicity we assume that utilities are attached to leaves only); the utility of a leaf node $L_i \in \mathcal{N}_U$ is denoted $u(L_i)$.
- 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 the domain of which 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 $C_j \in \mathcal{N}_C$, the domain of which 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.

[5] The proofs are omitted for the sake of brevity but are available at https://www.irit.fr/publis/ADRIA/PapersFargier/ecai2016.pdf

$Start(\mathcal{DT})$ denotes the first decision nodes of the tree (it is a singleton containing the root of the tree if it 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 of $\mathcal{DT}$, we shall also consider the subproblem $\mathcal{DT}_N$ 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 $\pi_j(x)$, the possibility $\pi(X_j = x|past(C_j))$[6]. 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 *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 strategy. Admissible strategies 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) $\forall D_i \in Start(\mathcal{DT}), \delta(D_i) \neq \perp$ 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 strategies built from a tree rooted in N. Each strategy 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 $Start(\mathcal{DT})$) 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.

We identify a strategy δ, the corresponding subtree and the list of its trajectories represented by a matrix. We also consider subtrees, and thus sub-strategies: let C_j be a chance node, $D_{i_1}, \ldots, D_{i_k}$ its successors and, for $l = 1, k$, the strategies $\delta_{i_l} \in \Delta_{D_{i_l}}$ which solve the subproblem rooted in D_{i_l}. $\delta_{i_1} + \cdots + \delta_{i_k}$ is the strategy 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 state she must choose between Investing (Inv) or Advertising (Adv) and she may be then Rich (R) or Poor (P) and Famous (F) or Unknown (U). Figure 1 shows the possibilistic decision tree (with horizon $h = 2$) that represents this decision problem. This tree has 8 strategies, 16 trajectories:*
$\tau_1 = (Adv, R\&U, Inv, P\&U)$, $\tau_2 = (Adv, R\&U, Inv, R\&U)$, $\tau_3 = (Adv, R\&U, Adv, R\&U)$, $\tau_4 = (Adv, R\&U, Adv, R\&F)$, $\tau_5 = (Adv, R\&F, Adv, R\&U)$, $\tau_6 = (Adv, R\&F, Adv, R\&F)$, *etc.*

The evaluation of a possibilistic strategy, as proposed by [22], relies on the qualitative optimistic and pessimistic decision criteria axiomatized by [11]. The utility of the strategy is computed on the basis of the transition possibilities and the utilities of its trajectories. For each trajectory $\tau = (a_{j_0}, x_{i_1}, a_{j_1}, \ldots, x_{i_h})$:

[6] 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 actually only on the decision made in the preceding node and on the state of the preceding chance node.

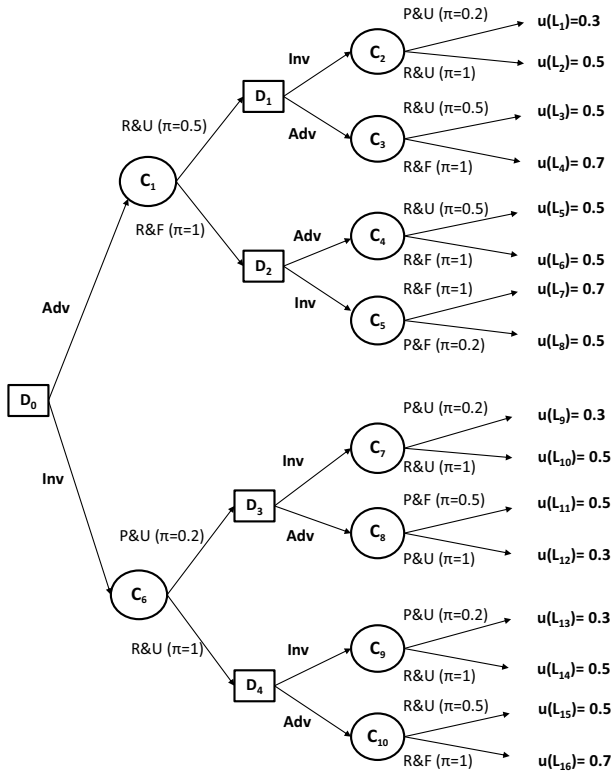

Figure 1. The possibilistic decision tree of Example 1

- Its utility denoted $u(\tau)$, is the utility $u(x_{i_h})$ of its leaf.
- The possibility of τ given that a strategy δ is applied from initial node D_0 is defined by:

$$\pi(\tau|\delta, D_0) = \begin{cases} \min\limits_{k=1..h} \pi_{j_{k-1}}(x_{i_k}) & \text{if } \tau \text{ is a trajectory of } \delta, \\ 0 & \text{otherwise.} \end{cases}$$

where $\pi_{j_{k-1}}$ is the possibility distribution at $C_{j_{k-1}}$.

It is now possible to compute, for any $\delta \in \Delta$ its optimistic and pessimistic utility degrees (the higher, the better):

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

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

This approach is purely ordinal (only min and max operations are used to aggregate the evaluations of the possibility of events and the ones of the utility of states). We can check that the preference orderings $\succeq_O$ between strategies, derived either from u_{opt} ($O = u_{opt}$) or from u_{pes} ($O = u_{pes}$), satisfy the principle of weak monotonicity: $\forall C_j \in \mathcal{N}_{C_j}, \forall D_i \in Succ(C_j), \delta, \delta' \in \Delta_{D_i}, \delta" \in \Delta_{Succ(C_j)\setminus D_i}$:

$$\delta \succeq_O \delta' \implies \delta + \delta" \succeq_O \delta' + \delta''$$

This property guarantees that *dynamic programming* [2] applies, and provides an optimal strategy in time polynomial with the size of the tree: [21, 22] have proposed qualitative counterparts of stochastic dynamic programming algorithms: in the finite horizon case *backwards induction*, or in the infinite horizon case *value and policy iteration*.

The basic pessimistic and optimistic utilities nevertheless present a severe drawback, known as the "drowning effect", due to the use of idempotent operations. In particular, when two strategies 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 they may give significantly different consequences in other possible trajectories, as illustrated in Example 2.

Example 2 *Let δ and δ' be the two strategies of Example 1 defined by $\delta(D_0) = \delta'(D_0) = Adv; \delta(D_1) = Inv; \delta'(D_1) = Adv; \delta(D_2) = \delta'(D_2) = Adv$. δ gathers 4 trajectories, $\tau_1, \tau_2, \tau_5, \tau_6$ with $\pi(\tau_1|D_0, \delta) = 0.2$ and $u(\tau_1) = 0.3$; $\pi(\tau_2|D_0, \delta) = 0.5$ and $u(\tau_2) = 0.5$; $\pi(\tau_5|D_0, \delta) = 0.5$ and $u(\tau_5) = 0.5$; $\pi(\tau_6|D_0, \delta) = 1$ and $u(\tau_5) = 0.5$. Hence $u_{opt}(\delta) = u_{pes}(\delta) = 0.5$.*
- δ' is also composed of 4 trajectories ($\tau_3, \tau_4, \tau_5, \tau_6$). Hence $u_{opt}(\delta') = u_{pes}(\delta') = 0.5$.
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_1). τ_2, which is good and totally possible "drowns" the bad consequence of δ in τ_1 in the optimistic comparison; in the pessimistic one, the bad utility of τ_1 is drowned by its low possibility, hence a global degree u_{pes} that is equal to the one of δ' (that, once again, guarantees a 0.5 utility degree 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) $\forall D \in \text{Common}(\delta, \delta'), \delta_D \succeq_O \delta'_D$ and (ii) $\exists D \in \text{Common}(\delta_D, \delta'_D), \delta_D \succ_O \delta'_D$, then $\delta \succ_O \delta'$
where $\text{Common}(\delta, \delta')$ is the set of nodes 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} or u_{pes} do fully satisfy the classical, strict, monotonicity principle, that can be written as follows:
$\forall C_j \in \mathcal{N}_C, D_i \in Succ(C_j), \delta, \delta' \in \Delta_{D_i}, \delta" \in \Delta_{Succ(C_j) \setminus D_i}$,

$$\delta \succeq_O \delta' \iff \delta + \delta" \succeq_O \delta' + \delta''$$

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 [7] that satisfy the principle of Pareto efficiency.

3 Escaping the drowning effect by leximin and leximax 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 [17]. 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 $\forall i, t_{\mu(i)} = t'_{\mu(i)}$ or $\exists i^*, \forall i < i*, t_{\mu(i)} = t'_{\mu(i)}$ and $t_{\mu(i*)} > t'_{\mu(i*)}$

[7] Formally, a preference relation $\succeq'$ refines a preference relation $\succeq$ if and only if whatever δ, δ', if $\delta \succ \delta'$ then $\delta \succ' \delta'$.

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

The refinements of u_{opt} and u_{pes} by lexicographic principles have been considered by [13] for non sequential problems; in this context, a decision is a possibility distribution π over the utility degrees, i.e. a vector of pairs $(\pi(u), u)$. Then it is possible to write:

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

where $(\pi(u), u)_{\mu(i)}$ is the i^{th} best pair of $(\pi(u), u)$ according to $lmin$ and $(1 - \pi(u), u)_{\sigma(i)}$ is the i^{th} worst pair of $(1 - \pi(u), u)$ according to $lmax$.

A straightforward way of applying this to sequential decision is to reduce the compound possibility distribution corresponding to the strategy, as usually done in possibilistic (and probabilistic) decision trees. The reduction of δ yields the distribution π_δ on the utility degrees, defined by: $\pi_\delta(u) = \max_{\tau, u(\tau)=u} \pi(\tau|\delta, D_0)$. Then we can write:

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

$\trianglerighteq_{lmax(lmin)}$ (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 3 *Consider a modified version of the problem of Example 1 (Figure 2). δ and δ' are the two strategies defined by: $\delta(D_0) = \delta'(D_0) = Adv$, $\delta(D_1) = Inv$, $\delta' = (D_1) = Adv$, $\delta(D_2) = \delta'_{D_2} = Adv$. $Common(\delta, \delta') = \{D_0, D_1, D_2\}$, $\delta_{D_0} = \delta'_{D_0}$, $\delta_{D_2} = \delta'_{D_2}$ and δ_{D_1} dominates δ'_{D_1} w.r.t. $lmax(lmin)$, since $((1, 0.1), (1, 0.9)) \triangleright_{lmax(lmin)} ((1, 0.1)(0.5, 0.9))$. δ should then be strictly preferred to δ'. By reduction, we get $\pi_\delta(0.9) = \pi_\delta(0.1) = \min(0.4, 1) = 0.4$ 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.*

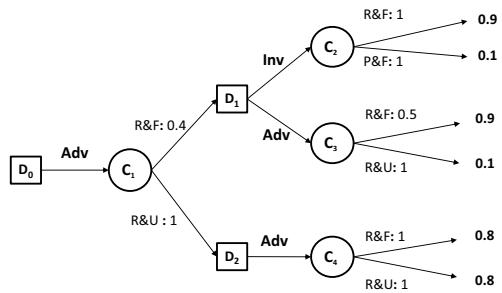

Figure 2. A counter example at the efficiency of $\trianglerighteq_{lmax(lmin)}$

The drowning effect at work here is due to the reduction of strategies, namely to the fact that the possibility of a trajectory

is drowned by the one of the least possible of its edges. That is why we propose to give up the principle of reduction and to build lexicographic comparisons on strategies considered *in extenso*. Recall that: $u_{opt}(\delta) = \max_{\tau \in \delta} \min \left\{ \min_{k=1..h} \pi_{j_{k-1}}(x_{i_k}); u(x_{i_h}) \right\}$. Then, for any $\tau = (a_{j_0}, x_{i_1}, \ldots, a_{j_{h-1}}, x_{i_h})$ and $\tau' = (a_{j'_0}, x_{i'_1}, \ldots, a_{j'_{h-1}}, x_{i'_h})$, we define $\succeq_{lmin}$ and $\succeq_{lmax}$ by:

- $\tau \succeq_{lmin} \tau'$ iff $(\pi_{j_0}(x_{i_1}), \ldots, \pi_{j_{h-1}}(x_{i_h}), u(x_{i_h})) \succeq_{lmin} (\pi_{j'_0}(x_{i'_1}), \ldots, \pi_{j'_{h-1}}(x_{i'_h}), u(x_{i'_h}))$
- $\tau \succeq_{lmax} \tau'$ iff $(1 - \pi_{j_0}(x_{i_1}), \ldots, 1 - \pi_{j_{h-1}}(x_{i_h}), u(x_{i_h})) \succeq_{lmax} (1 - \pi_{j'_0}(x_{i'_1}), \ldots, 1 - \pi_{j'_{h-1}}(x_{i'_h}), u(x_{i'_h}))$

Hence the proposition of the following preference relations[8]:

- $\delta \succeq_{lmax(lmin)} \delta'$ iff $\forall i,\ \tau_{\mu(i)} \sim_{lmin} \tau'_{\mu(i)}$ or $\exists i^*, \forall i \leq i*, \tau_{\mu(i)} \sim_{lmin} \tau'_{\mu(i)}$ and $\tau_{\mu(i^*)} \succ_{lmin} \tau'_{\mu(i^*)}$,
- $\delta \succeq_{lmin(lmax)} \delta'$ 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 $\tau_{\mu(i)}$ (resp. $\tau'_{\mu(i)}$) is the i^{th} best trajectory of δ (resp δ') according to $\succeq_{lmin}$ and $\tau_{\sigma(i)}$ (resp. $\tau'_{\sigma(i)}$) is the i^{th} worst trajectory of δ (resp δ') according to $\succeq_{lmax}$.

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 $lmin(lmax)$ and suggest a probabilistic interpretation, which we develop in Section 5. From a practical point of view, they allow us to define a dynamic programming algorithm to get lexi optimal solutions - this is the topic of the next Section.

4 Dynamic Programming for lexi qualitative criteria

The algorithm we propose (Algorithm 1 for the $lmax(lmin)$ variant; the $lmin(lmax)$ variant is similar) proceeds in the classical way, by backwards induction: when a chance node is reached, an optimal substrategy is recursively built for each of its children; these substrategies are combined but the resulting strategy is NOT reduced, contrarily to what is classically done; when a decision node is reached, the program is called for each child and the best of them is selected.

The comparison of strategies is done on the basis of the matrices of their trajectories (denoted ρ ; each line gathers the possibility and utility degrees of a trajectory $\tau = (a_{j_0}, x_{i_1}, a_{j_1}, \ldots, a_{j_h}, x_{i_h})$):

$$\rho_{lt} = \begin{cases} \pi_{j_{t-1}}(x_{i_t}) & \text{if } t \leq h, O = lmax(lmin) \\ 1 - \pi_{j_{t-1}}(x_{i_t}) & \text{if } t \leq h, O = lmin(lmax) \\ u(x_{i_h}) & \text{if } t = h+1. \end{cases}$$

So as to allow fast comparisons, the matrices are built incrementally and ordered on the fly by the function $ConcatAndOrder$: when a chance node, say C_j is reached, $k = |Succ(C_j)|$ substrategies are built recursively and their matrices $\rho_1, \ldots, \rho_k$ are computed. Matrix ρ of the current (compound) strategy, for the subtree rooted in C_j, is obtained by calling $ConcatAndOrder\left(\rho_1, \ldots, \rho_k, \pi_{C_j}\right)$. This function adds a column to each ρ_i, filled with $\pi_j(x_i)$; the matrices are vertically concatenated; then the elements in the lines are ordered in decreasing (resp. increasing) order, and the lines are reordered by decreasing (resp. increasing) order w.r.t. to $lmax$ (resp. $lmin$). As a matter of fact, once ρ has been reordered, $\rho_{1,1}$ is always equal to $u_{opt}(\delta)$ (resp. $u_{pes}(\delta)$).

The lexicographic comparison of two strategies δ and δ' is performed by scanning the elements $\rho_{l,t}$ and $\rho'_{l,t}$ of ρ and ρ' in parallel, line by line from the first one. The first pair of different $(\rho_{l,t}, \rho'_{l,t})$ determines the best matrix/strategy. 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).

Even if working with matrices rather than numerical values, the algorithm is polynomial w.r.t. the size of the original tree. This is because (i) the algorithm crosses each edge of the tree only once (as in the classical version), (ii) the matrices are never bigger than the strategies and (iii) the comparison of strategies is done in time linear with their size - thus linear with the size of the original tree.

Algorithm 1: DynProgLmaxLmin(N:Node)
Data: δ, the strategy built by the algorithm, is a global variable
Result: Computes δ for $\mathcal{DT}_N$ and returns the maxtrix of its trajectories, ρ

```
begin
    // Leaves
    if N ∈ N_U then ρ = [u(N)];
    // Chance nodes
    if N ∈ C then
        k = |Succ(N)|;
        for D_i ∈ Succ(N) do
        ρ_i ← DynProgLmaxLmin(D_i);
        ρ ← ConcatAndOrder(ρ_1, ..., ρ_k, π_N);
    // Decision nodes
    if N ∈ D then
        ρ ← [0]
        foreach a_j ∈ Out(N) do
            ρ_j ← DynProgLmaxLmin(Succ(N, a_j));
            if ρ_j ≽_{lmax(lmin)} ρ then
            ρ ← ρ_j and δ(N) ← a_j;
    return ρ;
```

5 Lexi comparisons and Expected Utility

If the problem is not sequential, it is easy to see that the comparison of possibilistic utility distributions by $\succeq_{lmax(lmin)}$ and $\succeq_{lmin(lmax)}$ do satisfy the axioms of EU. [13] have indeed shown that these decision criteria can be captured by an EU - namely, relying on infinitesimal probabilities and utilities. In this Section, we claim that such a result can be extended to sequential problems - for decision trees.

The proof relies on a transformation of the possibilistic tree into a probabilistic one. The graphical components are identical and so are the sets of admissible strategies. In the optimistic case the probability and utility distributions are chosen in such a way that the $lmax(lmin)$ and EU criteria do provide the same preference on Δ. To this extent, we build a transformation $\phi : L \subseteq [0,1] \rightarrow [0,1]$

[8] If the strategies have different numbers of trajectories, neutral trajectories (vectors) are added to the shortest strategy, at the bottom of the shortest list of trajectories

that maps each possibility distribution to an additive distribution and each utility level into an additive one; this transformation is 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 b is the branching factor of the tree. 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 tree will also decide in favor of δ.

For any chance node C_j, a local transformation ϕ_j is then derived from ϕ, such that ϕ_j satisfies both condition (R) and the normalization condition of probability theory. 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 4 $\phi(1) = \mathbf{1}, \phi(0.9) = \mathbf{0.2}, \phi(0.8) = \mathbf{0.001}, \phi(0.5) = \mathbf{10^{-10}}, \phi(0.4) = \mathbf{10^{-30}}, \phi(0.1) = \mathbf{10^{-91}}$.

It holds that $\phi(\alpha)^3 > \phi(\alpha') * 2^2$*, for all* $\alpha > \alpha'$*. We obtain the transformed conditional distributions by normalizing on each node. For instance 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* $\phi_4(1) = 0.5$.

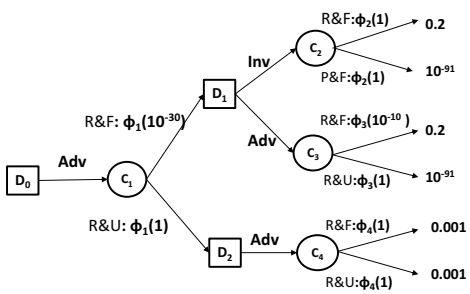

Figure 3. Transformed probabilistic decision tree of possibilisic decision tree of (counter)-example 3

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

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 $\mathcal{DT}$ 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 lexi-comparisons have a probabilistic interpretation - actually, using infinitesimal probabilities and utilities. This result comforts the idea, first proposed by [4] and then by [13], of a bridge between qualitative approaches and probabilities, through the notion of big stepped probabilities [4, 24]. We make here a step further, by the identification of 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 strategies - 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 infinitesimals; then either the program is based on an explicit handling of infinitesimals, 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.

6 Experiments

We thus get three criteria for each of the pessimistic and optimistic approaches: the basic possibilistic ones, the lexicographic refinements described in Section 3, and the EU approximations of the latter. We compare the 3 variants within each series with two measures: the CPU time and a pairwise success rate: $Success_{\frac{A}{B}}$ is the percentage of solutions 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 backward induction algorithms corresponding to the six criteria have been implemented in Java. 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. The experiments have been performed on an Intel Core i5 processor computer (1.70 GHz) with 8GB DDR3L of RAM..

The tests were performed on complete binary decision trees, for $h = 2$ to $h = 7$, that are randomly generated. 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 uniformly randomly fired 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 is uniformly fired in L. For each value of h, 100 decision trees are generated.

Figure 4 presents the average execution CPU time for the six criteria. We observe that, whatever the optimized criterion, the CPU time increases linearly w.r.t. 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 $1s$ for a decision tree 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 conclusion is drawn when comparing $lmin(lmax)$ to u_{pes} and EU_{pes}. These results are easy to explain: (i) the manipulation of matrices is obviously 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 infinitesimal.

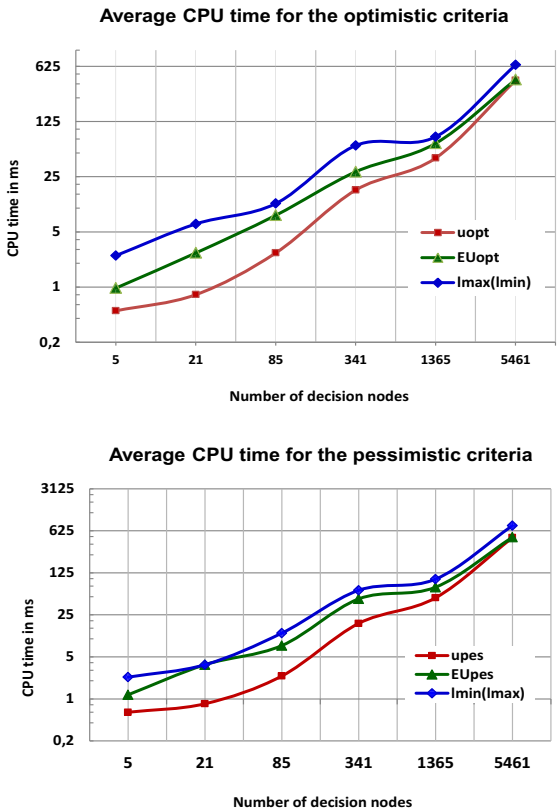

Figure 4. Average CPU time (in ms) for h=2 to 7

As to the success rate, the results are described in Figure 5. The percentage of solutions 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 solutions 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 infinitesimals and converge to 0 when the length of the branches (and thus the number of factors in the products) increase, as suggested in Section 5.

These experiments conclude in favor of the lexi 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 better approximation than u_{opt} and u_{pes} when space is limited (or when h increases).

7 Concluding remarks

This work has both theoretical and practical implications. It extends and generalizes to sequential problems the theoretical links established in [13] between possibilistic utilities and expected utilities. It performs better that the refinement of binary possibilistic utilities

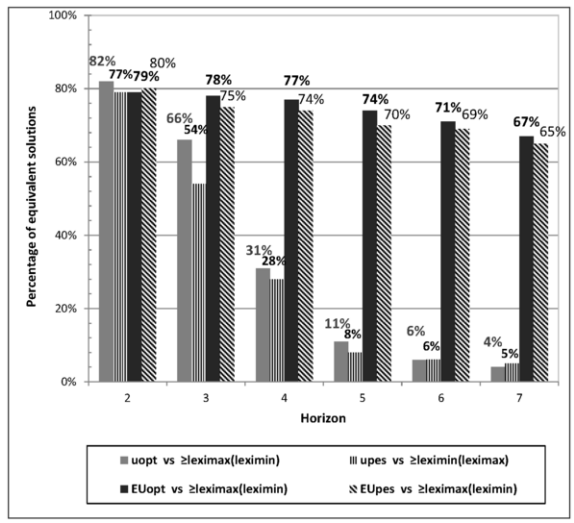

Figure 5. Sucess rate

(BPU) proposed in [27] for Binary Possibilistic Utilities and as a particular case, to classical, optimitic and pessimistic, possibilitistic utilities. In [27]'s treatment indeed, two similar trajectories of the same strategy are merged. The resulting criterion thus suffers from a drowning effect and does no satisfy strict monotonicity: as such, it cannot be represented by an EU-based criterion which "counts" trajectories (weighted by their probabilities). We actually do refine [27]'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 perspectives of our work are twofold. First, our approach could be naturally extended to solve possibilistic Markov Decision Processes.This extension seems theoretically straightforward, since a finite-horizon MDP can be translated into a set of decision trees (one for each state). Thus, our theoretical results hold for finite-horizon MDPs as well. However, the direct application of the lexicographic approach to possibilistic MDPs may lead to algorithms which are exponential in time and space (w.r.t. the MDP description), since the decision trees associated to a MDP may be of exponential size, while (possibilistic) MDPs can be solved in polynomial time [22, 21]. Determining whether computing lexicographic optimal solutions to possibilistic MDPs is tractable is a perspective of this work.

The second perspective of this work, not unrelated, is to develop simulation-based algorithms for finding lexicographic solutions to MDPs. Reinforcement Learning algorithms [26] allow to solve large size MDPs by making use of simulated trajectories of states to optimize a strategy. It is not immediate to develop RL algorithms for possibilistic MDPs, since no unique stochastic transition function corresponds to a possibility distribution. 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 lexicographic-optimal policies for (large) possibilistic MDPs.

REFERENCES

[1] Kim Bauters, Weiru Liu, and Llu'is Godo, 'Anytime algorithms for solving possibilistic MDPs and hybrid MDPs', in *9th International Symposium on Foundations of Information and Knowledge Systems (FoIKS'16)*, eds., Marc Gyssens and Guillermo Simari, Lecture Notes in Artificial Intelligence, pp. 1–18. Springer International Publishing Switzerland, (2016).

[2] Richard Bellman, *Dynamic Programming*, Princeton University Press, 1957.

[3] Nahla Ben Amor, Hélène Fargier, and Wided Guezguez, 'Possibilistic sequential decision making', *International Journal of Approximate Reasoning*, **55**, 1269–1300, (2014).

[4] Salem Benferhat, Didier Dubois, and Henri Prade, 'Possibilistic and standard probabilistic semantics of conditional knowledge bases', *Journal of Logic and Computation*, **9**, 873–895, (1999).

[5] Blai Bonet and Hector Geffner, 'Arguing for decisions: A qualitative model of decision making', in *12th Conference on Uncertainty in Artificial Intelligence (UAI-96), August 1-4, Portland, Oregon, USA*, pp. 98–105, (1996).

[6] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman, 'Acting optimally in partially observable stochastic domains', in *12th National Conference on Artificial Intelligence (AAAI'13), July 31 - August 4 Seattle, WA, USA*, pp. 1023–1028, (1994).

[7] Francis C. Chu and Joseph Y. Halpern, 'Great expectations. part I: on the customizability of generalized expected utility', in *18th International Joint Conference on Artificial Intelligence (IJCAI-03), August 9-15,2013, Acapulco, Mexico*, pp. 291–296, (2003).

[8] Nicolas Drougard, Florent Teichteil-Konigsbuch, Jean-Loup Farges, and Didier Dubois, 'Qualitative possibilistic mixed-observable MDPs', in *29th Conference on Uncertainty in Artificial Intelligence (UAI'13), August 11-15,2013, Bellevue, WA, USA*, pp. 192–201, (2013).

[9] Nicolas Drougard, Florent Teichteil-Konigsbuch, Jean-Loup Farges, and Didier Dubois, 'Structured possibilistic planning using decision diagrams', in *28th Conference on Artificial Intelligence (AAAI'14), July 27 -31, 2014, Québec City, Québec, Canada.*, pp. 2257–2263, (2014).

[10] Didier Dubois, Lluis Godo, Henri Prade, and Adriana Zapico, 'Making decision in a qualitative setting: from decision under uncertainty to case-based decision', in *6th International Conference on Principles of Knowledge Representation and Reasoning (KR'98), June 2-5, Trento, Italy*, pp. 594–605, (1998).

[11] Didier Dubois and Henri Prade, 'Possibility theory as a basis for qualitative decision theory', in *14th international joint conference on Artificial intelligence (IJCAI'95), August 20-25, Montreal, Quebec Canada*, pp. 1925–1930, (1995).

[12] Didier Dubois, Henri Prade, and Régis Sabbadin, 'Decision-theoretic foundations of qualitative possibility theory', *European Journal of Operational Research*, **128**, 459–478, (2001).

[13] Hélène Fargier and Régis Sabbadin, 'Qualitative decision under uncertainty: back to expected utility', *Artificial Intelligence*, **164**, 245–280, (2005).

[14] Phan Giang and Prakash P Shenoy, 'Two axiomatic approaches to decision making using possibility theory', *European Journal of Operational Research*, **162**, 450–467, (2005).

[15] Lluis Godo and Adriana Zapico, 'On the possibilistic-based decision model: Characterization of preference relations under partial inconsistency', *Applied Intelligence*, **14**, 319–333, (2001).

[16] Daniel J. Lehmann, 'Generalized qualitative probability: Savage revisited.', in *21st Conference in Uncertainty in Artificial Intelligence (UAI '05), July 26-29, Edinburgh, Scotland*, pp. 381–388, (1996).

[17] Hervi Moulin, *Axioms of Cooperative Decision Making*, Cambridge University Press, 1988.

[18] John Von Neumann and Oskar Morgenstern, *Theory of games and economic behavior*, 1948.

[19] Martin L. Puterman, *Markov Decision Processes*, John Wiley and Sons, 1994.

[20] Howard Raiffa, *Decision Analysis: Introductory Lectures on Choices under Uncertainty*, Addison Wesley, 1968.

[21] Régis. Sabbadin, 'Possibilistic Markov decision processes', *Engineering Applications of Artificial Intelligence*, **14**, 287–300, (2001).

[22] Régis Sabbadin, Hélène Fargier, and Jŕome Lang, 'Towards qualitative approaches to multi-stage decision making', *International Journal of Approximate Reasoning*, **19**, 441–471, (1998).

[23] Leonard J. Savage, *The Foundations of Statistics*, Wiley, 1954.

[24] Paul Snow, 'Diverse confidence levels in a probabilistic semantics for conditional logics', *Artificial Intelligence*, **113**, 269–279, (1999).

[25] Richard S. Sutton, 'Learning to predict by the methods of temporal differences', in *Machine Learning*, pp. 9–44, (1988).

[26] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning:An Introduction*, MIT Press, 1998.

[27] Paul Weng, 'Qualitative decision making under possibilistic uncertainty: Toward more discriminating criteria', in *21st Conference in Uncertainty in Artificial Intelligence (UAI'05), July 26-29, Edinburgh, Scotland*, pp. 615–622, (2005).

[28] Paul Weng, 'Axiomatic foundations for a class of generalized expected utility: Algebraic expected utility', in *22nd Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-06), July 13-16 , Arlington, Virginia*, pp. 520–527, (2006).