Algorithms for Multi-criteria optimization in Possibilistic Decision Trees

Nahla Ben Amor [1], Fatma Essghaier[1,2], Hélène Fargier [2]

[1] LARODEC, Le Bardo, Tunisie, email: nahla.benamor@gmx.fr, essghaier.fatma@gmail.com
[2] IRIT, Toulouse, France, email: fargier@irit.fr

Abstract. This paper raises the question of solving multi-criteria sequential decision problems under uncertainty. It proposes to extend to possibilistic decision trees the decision rules presented in [1] for non sequential problems. It present a series of algorithms for this new framework: *Dynamic Programming* can be used and provide an optimal strategy for rules that satisfy the property of monotonicity. There is no guarantee of optimality for those that do not - hence the definition of dedicated algorithms. This paper concludes by an empirical comparison of the algorithms.

Keywords: Possibility theory; Sequential decision problems; Multi-criteria decision making; Decision trees.

1 Introduction

When information about uncertainty cannot be quantified in a probabilistic way, possibilistic decision theory is a natural field to consider [2–7]. 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 [8–10]). But up to this point, the evaluation of the strategies was considered in a simple, mono-criterion context, while it is often the case that several criteria are involved in the decision [11].

A theoretical framework has been proposed for multi-criteria / multi-agent (non sequential) decision making under possibilistic uncertainty [1, 12]. In the present paper, we extend it to decision trees and we propose a detailed algorithmic study. After a refreshing on the background (Section 2), Section 3 presents our algorithms, and is completed, in Section 4, by an experimental evaluation.

2 Background

2.1 Multi-criteria decision making (MCDM) under uncertainty

Following Dubois and Prade's possibilistic approach of decision making under qualitative uncertainty, a non-sequential (i.e. one stage) decision can be seen

as a possibility distribution[1] over a finite set of outcomes, called a (simple) *possibilistic lottery* [2]. Such a lottery is denoted $L = \langle \lambda_1/x_1, \ldots, \lambda_n/x_n \rangle$ where $\lambda_i = \pi_L(x_i)$ is the possibility that decision L leads to outcome x_i; this possibility degree can also be denoted by $L[x_i]$. In this framework, a decision problem is thus fully specified by a set of possibilistic lotteries on X and a utility function $u : X \mapsto [0, 1]$. Under the assumption that the utility scale and the possibility scale are commensurate and purely ordinal, [2] proposes to evaluate each lottery by a qualitative, optimistic or pessimistic, global utility:

$$\text{Optimistic utility: } U^+(L) = \max_{x_i \in X} \min(\lambda_i, u(x_i)) \tag{1}$$

$$\text{Pessimistic utility: } U^-(L) = \min_{x_i \in X} \max(1 - \lambda_i, u(x_i)) \tag{2}$$

$U^+(L)$ is a mild version of the maximax criterion: L is good as soon as it is totally plausible that it gives a good consequence. On the contrary, the pessimistic index, $U^-(L)$ estimates the utility of an act by its worst possible consequence: its value is high whenever L gives good consequences in every "rather plausible" state.

This setting assumes a ranking of X by a *single* preference criterion, hence the use of a single utility function. When several criteria, say a set $Cr = \{1, \ldots, p\}$ of p criteria, have to be taken into account, u must be replaced by a vector $\boldsymbol{u} = \langle u_1, \ldots, u_p \rangle$ of utility functions u_j. If the criteria are not equally important, each j is equipped with a weight $w_j \in [0, 1]$ reflecting its importance.

In the absence of uncertainty, each decision leads to a unique consequence and the problem is a simple problem of qualitative MCDM aggregation; classically, such aggregation shall be either conjunctive (i.e. based on a weighted min) or disjunctive (i.e. based on a weighted max) - see [13] for more details about weighted min and weighted max aggregations.

In presence of uncertainty, the aggregation can be done *ex-ante* or *ex-post*:

- The *ex-ante* approach consists in computing the (optimistic or pessimistic) utility relative to each criterion j, and then performs the MCDM aggregation.
- The *ex-post* approach consists in first determining the aggregated utility (conjunctive or disjunctive) of each possible x_i; then the problem can be viewed as a mono-criterion problem of decision making under uncertainty.

Since the decision maker's attitude with respect to uncertainty can be either optimistic or pessimistic and the way of aggregating the criteria either conjunctive or disjunctive, [1, 12] propose four *ex-ante* and four *ex-post* approaches:

$$U^{-\min}_{ante}(L) = \min_{j \in Cr} \max((1 - w_j), \min_{x_i \in X} \max(u_j(x_i), (1 - L[x_i]))) \tag{3}$$

$$U^{-\max}_{ante}(L) = \max_{j \in Cr} \min(w_j, \min_{x_i \in X} \max(u_j(x_i), (1 - L[x_i]))) \tag{4}$$

$$U^{+\min}_{ante}(L) = \min_{j \in Cr} \max((1 - w_j), \max_{x_i \in X} \min(u_j(x_i), L[x_i])) \tag{5}$$

$$U^{+\max}_{ante}(L) = \max_{j \in Cr} \min(w_j, \max_{x_i \in X} \min(u_j(x_i), L[x_i])) \tag{6}$$

$$U^{-\min}_{post}(L) = \min_{x_i \in X} \max((1 - L[x_i]), \min_{j \in Cr} \max(u_j(x_i), (1 - w_j))) \tag{7}$$

$$U^{-\max}_{post}(L) = \min_{x_i \in X} \max((1 - L[x_i]), \max_{j \in Cr} \min(u_j(x_i), w_j)) \tag{8}$$

[1] A possibility distribution π is a mapping from the universe of discourse to a bounded linearly ordered scale, typically by the unit interval [0, 1].

$$U_{post}^{+\min}(L) = \max_{x_i \in X} \min(L[x_i], \min_{j \in Cr} \max(u_j(x_i), (1 - w_j))) \quad (9)$$

$$U_{post}^{+\max}(L) = \max_{x_i \in X} \min(L[x_i], \max_{j \in Cr} \min(u_j(x_i), w_j)) \quad (10)$$

In the notations above, the first (resp. second) sign denotes the attitude of the decision maker w.r.t. uncertainty (resp. the criteria). The $U_{ante}^{-\min}$ utility for instance considers that the decision maker is pessimistic and computes the pessimistic utility of each criterion. Then the criteria are aggregated on a cautions basis: the higher is the satisfaction of the least satisfied of the important criteria, the better is the lottery. Using the same notations, $U_{post}^{-\max}$ considers that a x_i is good as soon as one of the important criteria is satisfied: a max-based aggregation of the utilities is done, yielding a unique utility function $u()$ on the basis of which the pessimistic utility is computed. It should be noticed that the full pessimistic and full optimistic *ex-ante* utilities are equivalent to their *ex-post* counterparts [12], i.e. $U_{ante}^{-\min} = U_{post}^{-\min}$ and $U_{ante}^{+\max} = U_{post}^{+\max}$. But $U_{ante}^{-\max}$ (resp. $U_{ante}^{+\min}$) may differ from $U_{post}^{-\max}$ (resp. from $U_{post}^{+\min}$).

Example 1 *Consider two equally important criteria* 1 *and* 2 *($w_1 = w_2 = 1$), and a lottery $L = \langle 1/x_a, 1/x_b \rangle$ leading to two equi possible consequences x_a and x_b such that x_a is good for* 1 *and bad for* 2*, and x_b is bad for* 1 *and good for* 2*: $u_1(x_a) = u_2(x_b) = 1$ and $u_2(x_a) = u_1(x_b) = 0$. It is easy to check that $U_{ante}^{+\min}(L) = 0 \neq U_{post}^{+\min}(L) = 1$.*

2.2 Possibilistic Decision Trees [14]

Decision trees provide an explicit modeling of sequential problems by representing, simply, all possible scenarios. A decision tree is a labeled tree $\mathcal{DT} = (\mathcal{N}, \mathcal{E})$ where $\mathcal{N} = \mathcal{D} \cup \mathcal{C} \cup \mathcal{LN}$ contains three kinds of nodes (see Figure 1): $\mathcal{D}$ is the set of decision nodes (represented by squares); $\mathcal{C}$ is the set of chance nodes (represented by circles) and $\mathcal{LN}$ is the set of leaves.$Succ(N)$ denotes the set of children nodes of node N. For any $X_i \in \mathcal{D}, Succ(X_i) \subseteq \mathcal{C}$ i.e. a chance node (an action) must be chosen at each decision node. For any $C_i \in \mathcal{C},\ Succ(C_i) \subseteq \mathcal{LN} \cup \mathcal{D}$: the set of outcomes of an action is either a leaf node or a decision node (and then a new action should be executed).

In the possibilistic context, leaves are labeled by utility degrees in the $[0, 1]$ scale and the uncertainty pertaining to the possible outcomes of each $C_i \in \mathcal{C}$, is represented by a *conditional possibility distribution* π_i on $Succ(C_i)$, such that $\forall N \in Succ(C_i), \pi_i(N) = \Pi(N|path(C_i))$ where $path(C_i)$ denotes all the value assignments of chance and decision nodes on the path from the root to C_i [14].

Solving a decision tree amounts at building a *complete strategy* that selects an action (a chance node) for each decision node: a strategy is a mapping $\delta : \mathcal{D} \mapsto \mathcal{C} \cup \{\bot\}$. $\delta(D_i) = \bot$ means that no action has been selected for D_i (δ is partial). Leaf nodes being labeled with utility degrees, the rightmost chance nodes can be seen as *simple possibilistic lotteries*. Then, each strategy δ can be viewed as a connected sub-tree of the decision tree and is identified with a *possibilistic compound lottery* L_δ, i.e. with a possibility distribution over a set of (simple or compound) lotteries. A compound lottery $\langle \lambda_1/L_1, ..., \lambda_k/L_k \rangle$ (and thus any strategy) can then be reduced into an equivalent simple lottery as follows[2] [2]:

$$Reduction(\langle \lambda_1/L_1, ..., \lambda_k/L_k \rangle) = \langle \max_{j=1,k}(\min(\lambda_1^j, \lambda_j))/u_1, ..., \max_{j=1,k}(\min(\lambda_n^j, \lambda_j))/u_n \rangle.$$

[2] Obviously, the reduction of a simple lottery is the simple lottery itself.

The pessimistic and optimistic utility of a strategy δ can then be computed on the basis of the reduction of L_δ: the utility of δ is the one of $Reduction(L_\delta)$.

3 Multi-criteria optimization in Possibilistic Trees

Multi-criteria Possibilistic Decision Trees can now be defined: they are classical possibilistic decision trees, the leaves of which are evaluated according to several criteria - each leaf N is now labeled by a *vector* $u(N) = \langle u_1(N), \ldots, u_p(N)\rangle$ rather than by a single utility score (see Figure 1). A strategy still leads to compound lottery, and can be reduced, thus leading in turn to a simple (but multi-criteria) lottery. We propose to base the comparison of strategies on the comparison, according to the rules O previously presented, of their reductions:

$$\delta_1 \succeq_O \delta_2 \text{ iff } U_O(\delta_1) \geq U_O(\delta_2), \text{ where } \forall \delta, U_O(\delta) = U_O(Reduction(L_\delta)) \quad (11)$$

Example 2 *Consider the tree of Figure 1, involving two criteria that are supposed to be equally important and the strategy* $\delta(D_0) = C_1$, $\delta(D_1) = C_3$, $\delta(D_2) = C_5$. *It holds that* $L_\delta = \langle 1/L_{C_3}, 0.9/L_{C_5}\rangle$ *with* $L_{C_3} = \langle 0.5/x_a, 1/x_b\rangle$, $L_{C_5} = \langle 0.2/x_a, 1/x_b\rangle$. *Because* $Reduction(L_\delta) = \langle max(0.5, 0.2)/x_a, max(1, 0.9)/x_b\rangle = \langle 0.5/x_a, 1/x_b\rangle$, *we get* $U^{+min}_{ante}(\delta) = \min(\max\min(0.5, 0.3), \min(1, 0.6),\ \max(\min(0.5, 0.8)\min(1, 0.4))) = 0.5$.

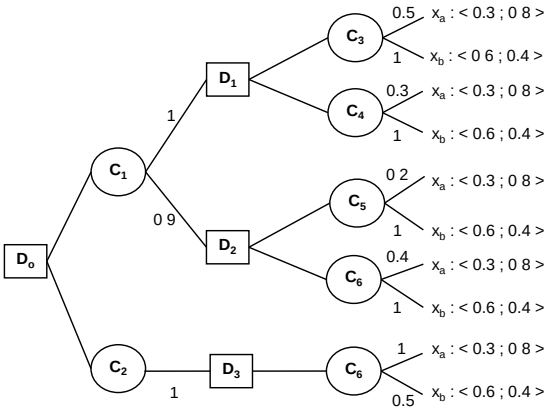

Fig. 1. A multi-criteria possibilistic decision tree

The definition proposed by Eq (11) is quite obvious but raises an algorithmic challenge: the set of strategies to compare is exponential w.r.t. the size of the tree which makes the explicit evaluation of the strategies not realistic. The sequel of the paper aims at providing algorithmic solutions to this difficulty.

3.1 Dynamic Programming as a tool for *ex-post* utilities

Dynamic Programming [15] is an efficient procedure of strategy optimization. It proceeds by *backward induction*, handling the problem from the end (and in our

case, from the leafs): the last decision nodes are considered first, and recursively until the root is reached. This algorithm is sound and complete as soon as the decision rule leads to complete and transitive preferences and satisfies the principle of weak monotonicity[3], that ensures that each sub strategy of an optimal strategy is optimal in its sub-tree. Hopefully, each of the *ex-post* criteria satisfy transitivity, completeness and weak monotonicity, because collapsing to either a classical U^- or a U^+ utility, which satisfy these properties [8, 14]. The adaptation of Dynamic Programming to the *ex-post* rules is detailed in Algorithm 1.

Algorithm 1: DynProgPost: *Ex-post* Dynamic Programming

Data: A Decision tree T, a node N in of T
Result: The value of the optimal strategy δ - δ is stored as a global variable
begin
  if $N \in \mathcal{LN}$ **then** // `Leaf : MCDM aggregation`
    for $i \in \{1, \ldots, p\}$ **do** $u_N \leftarrow (u_N \oplus (u_i \otimes \omega_i))$;
    // $\otimes = \min$, $\omega_i = w_i$, $\oplus = \max$ `for disjunctive aggregation`
    // $\otimes = \max$, $\omega_i = 1 - w_i$, $\oplus = min$ `for conjunctive aggregation` ;
  if $N \in \mathcal{C}$ **then** // `Chance Node: compute the qualitative utility`
    foreach $Y \in Succ(N)$ **do** $u_N \leftarrow (u_N \oplus (\lambda_Y) \otimes DynProgPost(Y))$;
    // $\otimes = \min$, $\lambda_Y = \pi(Y)$, $\oplus = \max$ `for optimistic utility`
    // $\otimes = \max$, $\lambda_Y = 1 - \pi(Y)$, $\oplus = \min$ `for pessimistic utility`
  if $N \in \mathcal{D}$ **then** // `Decision node: determine the best decision`
    $u^* \leftarrow 0$;
    foreach $Y \in Succ(N)$ **do**
      $u_Y \leftarrow DynProgPost(Y)$;
      if $u_Y \geq u^*$ **then** $\delta(N) \leftarrow Y$ and $u^* \leftarrow u_Y$;
  return u^*;

In short, this algorithm aggregates the utility values of each leaf, and then builds an optimal strategy from the last decision nodes to the root of the tree, using the principle defined by [10, 9] for classical (monocriterion) possibilistic decision trees.

3.2 Dynamic Programming for *ex-ante* utilities ?

The *ex-ante* variant of Dynamic Programming we propose is a little more tricky (see Algorithm 2). It keeps at each node a vector of p pessimistic (resp. optimistic) utilities, one for each criterion. The computation of the *ex-ante* utility can then be performed each time a decision is to be made. Recall that $U_{ante}^{-\min} = U_{post}^{-\min}$ and $U_{ante}^{+\max} = U_{post}^{+\max}$. Hence, for these two rules the optimization could also be performed by the *ex-post* algorithm. The two other rules, $U_{ante}^{-\max}$ and $U_{ante}^{+\min}$, unfortunately do not satisfy the monotonicity principle (see

[3] Formally, $\succeq_O$ is said to be weakly monotonic iff whatever L, L' and L'', whatever (α,β) such that $\max(\alpha, \beta) = 1$: $L \succeq_O L' \Rightarrow \langle \alpha/L, \beta/L'' \rangle \succeq_O \langle \alpha/L', \beta/L'' \rangle$.

Algorithm 2: DynProgAnte: *Ex-ante* Dynamic Programming

Data: A Decision tree T, a node N in of T
Result: The value of the optimal strategy δ - δ is stored as a global variable
begin
  if $N \in \mathcal{LN}$ **then** // `Leaf`
    for $i \in \{1, \ldots, p\}$ **do** $\boldsymbol{u_N}[i] \leftarrow u_i$;
  if $N \in \mathcal{C}$ **then** // `Chance Node: compute the utility vectors`
    // `Optimistic utility` $\otimes = \min$, $\lambda_Y = \pi(Y)$, $\oplus = \max$, $\epsilon \leftarrow 0$
    // `Pessimistic utility` $\otimes = \max$, $\lambda_Y = 1 - \pi(Y)$, $\oplus = \min$, $\epsilon \leftarrow 1$
    for $i \in \{1, \ldots, p\}$ **do** $\boldsymbol{u_N}[i] \leftarrow \epsilon$;
    foreach $Y \in Succ(N)$ **do**
      $\boldsymbol{u_Y} \leftarrow DynProgAnte(Y)$;
      for $i \in \{1, \ldots, p\}$ **do** $\boldsymbol{u_N}[i] \leftarrow (\boldsymbol{u_N}[i] \oplus (\lambda_Y \otimes \boldsymbol{u_Y}[i]))$;
  if $N \in \mathcal{D}$ **then** // `Decision node`
    // `Disjunctive MCDM: let` $\otimes = \min$, $\omega_i = w_i$, $\oplus = \max$, $\epsilon \leftarrow 0$
    // `Conjunctive MCDM: let` $\otimes = \max$, $\omega_i = 1 - w_i$, $\oplus = \ min$, $\epsilon \leftarrow 1$
    $u^* \leftarrow 0$
    foreach $Y \in Succ(N)$ **do**
      $v_Y \leftarrow \epsilon$; $\boldsymbol{u_Y} \leftarrow DynProgAnte(Y)$;
      for $i \in \{1, \ldots, p\}$ **do** $v_Y \leftarrow v_Y \oplus (\boldsymbol{u_Y}[i] \otimes \ \omega_i)$;
      if $v_Y > u^*$ **then** $\delta(N) \leftarrow Y$ and $\boldsymbol{u_N} \leftarrow u_Y$;
  return $\boldsymbol{u_N}$;

[1]). Hence, Algorithm 2 may provide a good strategy, but without any guarantee of optimality - it can be considered as an approximation algorithm in these two cases. Another approximation algorithm is the *ex-post* Algorithm described in the previous Section - even if it is not always the case, it often happens that $U_{post}^{-\max} = U_{ante}^{-\max}$ (resp. $U_{post}^{+\min} = U_{ante}^{+\min}$); if it is the case the solution provided by the *ex-post* Algorithm is optimal.

3.3 Optimization of $U_{ante}^{-\max}$ by Multi Dynamic Programming

The lack of monotonicity of $U_{ante}^{-\max}$ is not dramatic, even when optimality must be guaranteed. With $U_{ante}^{-\max}$ indeed, we look for a strategy that has a good pessimistic utility U_j^- for *at least* one criterion j. This means that if it is possible to get for each j a strategy that optimizes U_j^- (and this can be done by Dynamic Programming, since the classical pessimistic utility is monotonic), the one with the highest value for $U_{ante}^{-\max}$ is globally optimal. Formally:

Proposition 1 $U_{ante}^{-\max}(L) = \max_{j=1,p} \min(w_j, U_j^-(L))$
where $U_j^-(L)$ is the pessimistic utility of L according to the sole criterion j.

Corollary 1 *Let $\Delta^* = \{L_1^*, \ldots, L_p^*\}$ s.t. $\forall L$, $U_j^-(L_j^*) \geq U_j^-(L)$ and $L^* \in \Delta^*$. If $\max_{j=1,p} \min(w_j, U_j^-(L^*)) \geq \max_{j=1,p} \min(w_j, U_j^-(L_i^*)) \forall L_i^* \in \Delta^*$ then $U_{ante}^{-\max}(L^*) \geq U_{ante}^{-\max}(L), \forall L$.*

Hence, the optimization problem can be solved by a series of p calls to a classical (monocriterion) pessimistic optimization. This is the principle of the Multi Dynamic Programming approach detailed by Algorithm 3.

Algorithm 3: MultiDynProg: right optimization of $U_{ante}^{-\max}$

Data: A tree T
Result: An Optimal strategy δ^* and its value u^*
begin
  $u^* = 0$; `// Initialization`
  for $i \in \{1, \ldots, p\}$ **do**
    $\delta_i = PesDynProg(T, i)$ `// Call to classical possibilistic Dynamic Prog. [14] - returns an optimal strategy for` U_i^-;
    $u_i = \max_{j=1\ldots p} \min(w_j, U_j^-(\delta_i))$;
    if $u_i > u^*$ **then** $\delta^* \leftarrow \delta_i$; $u^* \leftarrow u_i$;
  return u^*;

3.4 Right optimization of $U_{ante}^{+\min}$: a Branch and Bound algorithm

Let us finally study the $U_{ante}^{+\min}$ utility. As previously said, it does not satisfy monotonicity and Dynamic Programming can provide a good strategy, but without any guarantee of optimality. To guarantee optimality, one can proceed by an implicit enumeration via a Branch and Bound algorithm, as done by [8] for Possibilistic Choquet integrals and by [16] for Rank Dependent Utility (both in the mono criterion case). The Branch and Bound procedure (see Algorithm 4) takes as argument a partial strategy δ and an upper bound of the $U_{ante}^{+\min}$ value of its best extension. It returns U^*, the $U_{ante}^{+\min}$ value of the best strategy found so far, δ^*. We can initialize δ^* with any strategy, e.g. the one provided by Algorithms 2 or 1. At each step of the Branch and Bound algorithm, the current partial strategy, δ, is developed by the choice of an action for some unassigned decision node. When several decision nodes are candidate, the one with the minimal rank (i.e. the former one according to the temporal order) is developed. The recursive procedure backtracks when either the current strategy is complete (then δ^* and U^* are updated) or proves to be worse than the current δ^* in any case. Function $UpperBound(D_0, \delta)$ provides an upper bound of the best completion of δ - in practice, it builds, for each criterion j, a strategy δ_j that maximizes U_j^+ (using [10, 9]'s algorithm, which is linear). It then selects, among these strategies, the one with the highest $U_{ante}^{+\min}$. It is important to note that $UpperBound(D_0, \delta) = U_{ante}^{+\min}(\delta)$ when δ is complete. Whenever the value returned by $UpperBound(D_0, \delta)$ is lower or equal to U^*, the value of the best current strategy, the algorithm backtracks, yielding the choice of another action for the last considered decision node.

Algorithm 4: B&B algorithm for the optimization of $U_{ante}^{+,min}$

Data: A decision tree T, a (partial) strategy δ, an upper Bound U of $U_{ante}^{+,min}(\delta)$
Result: U^*: the $U_{ante}^{+,min}$ value of δ^* the best strategy found so far
begin
  if $\delta(D_0) = \perp$ **then** $\mathcal{D}_{pend} \leftarrow \{D_0\}$;
  else $D_{pend} \leftarrow \{D_i \in \mathcal{D}$ s.t. $\exists D_j, \delta(D_j) \neq \perp$ and $D_i \in Succ(\delta(D_j))\}$;
  if $\mathcal{D}_{pend} = \emptyset$ **then** // δ is a complete strategy
    $\delta^* \leftarrow \delta$; $U^* \leftarrow U$;
  else
    $D_{next} \leftarrow arg\ min_{D_i \in D_{pend}}\ i$;
    foreach $C_i \in Succ(D_{next})$ **do**
      $\delta(D_{next}) \leftarrow C_i$;
      $U \leftarrow UpperBound(D_0, \delta)$;
      if $U > U^*$ **then** $U^* \leftarrow B\&B(U, \delta)$;
  return U^*;

4 Experiments

Beyond the evaluation of the feasibility of the algorithms proposed, our experiments aim at evaluating to what extent the optimization of the problematic utilities, $U_{ante}^{-\max}$ and $U_{ante}^{+\min}$, can be approximated by Dynamic Programming.

The implementation has been done in Java, on a processor Intel Core $i7$ 2670 QMCPU, $2.2Ghz$, $6Gb$ of RAM. The experiments were performed on complete binary decision trees. We have considered four sets of problems, the number of decisions to be made in sequence (denoted *seq*) varying from 2 to 6, with an alternation of decision and chance nodes: at each decision level l (i.e. odd level), the tree contains 2^{l-1} decision nodes followed by 2^l chance nodes [4]. In the present experiment, the number of criteria is set equal to 3. The utility values as well as the weights degrees are uniformly fired in the set $\{0, 0.1, 0.2, \ldots, 0.9, 1\}$. Conditional possibilities are chosen randomly in $[0, 1]$ and normalized. Each of the four samples of problems contains 1000 randomly generated trees.

Feasibility analysis and temporal performances : Table 1 presents the execution time of each algorithm. Obviously, for each one, the CPU time increases with the size of the tree. But it remains affordable even for very big trees (1365 decisions). We can check that $U_{ante}^{-\max}$ (resp. $U_{ante}^{+\min}$) the approximation performed by *ex-post* Dynamic Programming is faster than the one performed by *ex-ante* Dynamic Programming, both being faster than the exact algorithm (Multi Dynamic Programming and Branch and Bound, respectively).

Quality of the approximation : As previously mentioned the *ex-post* and the *ex-ante* Dynamic Programming algorithms are approximation algorithms for $U_{ante}^{-\max}$ and $U_{ante}^{+\min}$. The following experiments estimate the quality of these approximations. At this extent, we compute for each sample the success rate

[4] Hence, for a sequence length $seq = 2$ (resp. 3, 4, 5, 6), the number of decision nodes in each tree of the sample is equal to 5 (resp. 21, 85, 341, 1365)

# decision nodes		5	21	85	341	1365
$U_{post}^{-\min}$ $U_{ante}^{-\min}$	Post Dyn. Prog.	0.068	0.073	0.076	0.126	0.215
$U_{post}^{+\max}$ $U_{ante}^{+\max}$	Post Dyn. Prog.	0.071	0.075	0.082	0.128	0.207
$U_{post}^{-\max}$	Post Dyn. Prog.	0.068	0.083	0.090	0.140	0.235
$U_{post}^{+\min}$	Post Dyn. Prog.	0.067	0.075	0.082	0.132	0.211
$U_{ante}^{-\max}$	Multi Dyn. Prog.	0.172	0.203	0.247	0.295	1.068
$U_{ante}^{-\max}$	Ante Dyn. Prog.	0.079	0.096	0.120	0.147	0.254
$U_{ante}^{+\min}$	Branch & Bound	0.576	1.012	1.252	1.900	5.054
$U_{ante}^{+\min}$	Ante Dyn. Prog.	0.074	0.084	0.093	0.147	0.231

Table 1. Average CPU time, in milliseconds, of for each algorithms and for each rule, according the size of the tree (in number of decision nodes)

of the approximation algorithm considered, i.e. the number of trees for which the value provided by the approximation algorithm is actually optimal; then for the trees for which it fails to reach optimality, we report the average closeness value to $\frac{U_{Approx}}{U_{Exact}}$ where U_{Approx} is the utility of the strategy provided by the approximation algorithm and U_{Exact} is the optimal utility - the one of the solution by the exact algorithm (Branch and Bound for $U_{ante}^{+\min}$ and Multi Dynamic Programming for $U_{ante}^{-\max}$). The results are given in Table 2.

# decision nodes		5	21	85	341	1365
% of success						
$U_{ante}^{-\max}$	Ante Dyn. Prog	17.3%	19%	22.1%	26.4%	31%
$U_{ante}^{-\max}$	Post. Dyn. Prog	15.4%	23.6%	30.7%	35.6%	40.4%
$U_{ante}^{+\min}$	Ante Dyn. Prog.	87%	76.8%	68%	62.6%	59.6%
$U_{ante}^{+\min}$	Post Dyn. Prog.	91.7%	90.8%	88.2%	86.7%	76%
Closeness Value						
$U_{ante}^{-\max}$	Ante Dyn. Prog.	0.522	0.56	0.614	0.962	0.981
$U_{ante}^{-\max}$	Post Dyn. Prog.	0.473	0.529	0.556	0.58	0.62
$U_{ante}^{+\min}$	Ante Dyn. Prog.	0.97	0.95	0.94	0.93	0.91
$U_{ante}^{+\min}$	Post Dyn. Prog.	0.989	0.975	0.946	0.928	0.90

Table 2. Quality of approximation of $U_{ante}^{-\max}$ and $U_{ante}^{+\min}$ by Dynamic Programming

Clearly, *Ex-Post* Dynamic Programming provides a good approximation for $U_{ante}^{+\min}$ - its success rate decreases with the number of nodes but stay higher than 70 %, and above all it has a very high closeness value (above 0.9) ; notice that it is always better than its *ex-ante* counterpart, in terms of success rate, of closeness and of CPU time. This is good news since it is polynomial while Branch and Bound, the exact algorithm, is exponential in the number of nodes. As to $U_{ante}^{-\max}$, none of the approximation algorithms is good. However, this is not so bad news since Multi Dynamic Programming, the exact algorithm is polynomial and has very affordable CPU time.

5 Conclusion

This paper proposes to extend to possibilistic decision trees the decision rules presented in [1] for non sequential problems. We show that, for the *ex-post* decision rules, as well as for U_{ante}^{+max} and U_{ante}^{-min}, the optimization can be achieved by Dynamic Programming. For $U_{ante}^{+\min}$ the optimization can be carried either by an exact but costly algorithm (Branch&Bound) or by an approximation one, (*ex-post* Dynamic Programming). For $U_{ante}^{-\max}$ we propose an exact algorithm (Multi Dynamic Programming) that performs better than Dynamic Programming. As future work, we would like to study the handling of several criteria in more sophisticated qualitative decision models such as possibilistic influence diagrams [14] or possibilistic Markov decision models [10].

References

1. N. Ben Amor, F. Essghaier, and H. Fargier, "Solving multi-criteria decision problems under possibilistic uncertainty using optimistic and pessimistic utilities," in *Proceedings of IPMU*, 2014, pp. 269–279.
2. D. Dubois and H. Prade, "Possibility theory as a basis for qualitative decision theory," in *Proceedings of IJCAI'95*, 1995, pp. 1924–1930.
3. D. Dubois, L. Godo, H. Prade, and A. Zapico, "Making decision in a qualitative setting: from decision under uncertainty to case-based decision," in *Proceedings of KR*, 1998, pp. 594–607.
4. P. H. Giang and P. P. Shenoy, "A qualitative linear utility theory for spohn's theory of epistemic beliefs," in *Proceedings of UAI*, 2000, pp. 220–229.
5. D. Dubois, H. Prade, and R. Sabbadin, "Decision theoretic foundations of qualitative possibility theory," *EJOR*, vol. 128, pp. 459–478, 2001.
6. D. Dubois, H. Fargier, H. Prade, and P. Perny, "Qualitative decision theory: From savage's axioms to nonmonotonic reasoning," *JACM*, vol. 49, pp. 455–495, 2002.
7. D. Dubois, H. Fargier, and P. Perny, "Qualitative decision theory with preference relations and comparative uncertainty: An axiomatic approach," *Artificial Intelligence*, vol. 148, pp. 219–260, 2003.
8. N.Ben Amor, H. Fargier, and W.Guezguez, "Possibilistic sequential decision making," *International Journal Approximate Reasoning*, vol. 55, pp. 1269–1300, 2014.
9. R. Sabbadin, H. Fargier, and J. Lang, "Towards qualitative approaches to multi-stage decision making," *International Journal of Approximate Reasoning*, vol. 19, pp. 441–471, 1998.
10. R. Sabbadin, "Empirical comparison of probabilistic and possibilistic markov decision processes algorithms." in *Proceedings of ECAI*, 2000, pp. 586–590.
11. J. Harsanyi, "Cardinal welfare, individualistic ethics, and interpersonal comparisons of utility," *Journal of Political Economy*, vol. 63, pp. 309–321, 1955.
12. N. Ben Amor, F. Essghaier, and H. Fargier, "Egalitarian collective decision making under qualitative possibilistic uncertainty: Principles and characterization," in *Proceedings of AAAI*, 2015, pp. 3482–3488.
13. D. Dubois and H. Prade, "Weighted minimum and maximum operations in fuzzy set theory," *Journal of Information Sciences*, vol. 39, pp. 205–210, 1986.
14. L. Garcias and R. Sabbadin, "Possibilistic influence diagrams," in *Proceedings of ECAI*, 2006, pp. 372–376.
15. R. Bellman, *Dynamic Programming*. Princeton University Press, 1957.
16. G. Jeantet and O. Spanjaard, "Rank-dependent probability weighting in sequential decision problems under uncertainty," in *Proceedings of ICAPS*, 2008, pp. 148–155.