HAL
open science

Resolute Choice in Sequential Decision Problems with Multiple Priors

Hélène Fargier, Gildas Jeantet, Olivier Spanjaard

To cite this version:

Hélène Fargier, Gildas Jeantet, Olivier Spanjaard. Resolute Choice in Sequential Decision Problems with Multiple Priors. 22nd International Joint Conference on Artificial Intelligence (IJCAI 2011), Jul 2011, Barcelona, Spain. pp.2120-2125, 10.5591/978-1-57735-516-8/IJCAI11-354 . hal-01282520

HAL Id: hal-01282520

https://hal.science/hal-01282520v1

Submitted on 10 Jul 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Resolute Choice in Sequential Decision Problems with Multiple Priors

Hélène Fargier
IRIT-CNRS, UMR 5505
Université Paul Sabatier
Toulouse, France
fargier@irit.fr

Gildas Jeantet
LIP6-CNRS, UMR 7606
UPMC
Paris, France
gildas.jeantet@gmail.com

Olivier Spanjaard
LIP6-CNRS, UMR 7606
UPMC
Paris, France
olivier.spanjaard@lip6.fr

Abstract

This paper is devoted to sequential decision making under uncertainty, in the multi-prior framework of Gilboa and Schmeidler [1989]. In this setting, a set of probability measures (*priors*) is defined instead of a single one, and the decision maker selects a strategy that maximizes the minimum possible value of expected utility over this set of priors. We are interested here in the resolute choice approach, where one initially commits to a complete strategy and never deviates from it later. Given a decision tree representation with multiple priors, we study the problem of determining an optimal strategy from the root according to min expected utility. We prove the intractability of evaluating a strategy in the general case. We then identify different properties of a decision tree that enable to design dedicated resolution procedures. Finally, experimental results are presented that evaluate these procedures.

1 Introduction

In many sequential decision problems, agents must act under uncertainty. This issue arises in several applications of artificial intelligence (AI), e.g. medical diagnosis (sequence of medical treatments/tests), troubleshooting under uncertainty (sequence of observations/repairs), poker-playing program (sequence of raise/call/fold). Decision theory provides useful tools to deal with uncertainty in decision problems. A decision under uncertainty is classically viewed as a function that maps states of the word to consequences - denoting Θ the set of states, X the set of consequences, $D = \{f : \Theta \mapsto X\}$ is the set of potential acts. Following the theory of expected utility [von Neuman and Morgenstern, 1947], a probability P can model the uncertainty about the state of the world, and a utility function $u : X \mapsto U \subset \mathbb{R}$ will capture the utility of the consequences. Then each decision f can be evaluated by its expected utility $EU_{P,u}(f) = \Sigma_{\theta\in\Theta}\ u(f(\theta))\ .\ P(\theta)$ and compared to other decision. The best decision should be the one that maximizes expected utility.

However, it has been shown that expected utility cannot always capture the behaviour of rational decision makers. Ellsberg's counter example [1961] is the following: let U be an urn containing $\frac{1}{3}$ of red balls and $\frac{2}{3}$ of black or yellow balls. One ball will be drawn and guessing the right color leads to a gain of 1 (0 otherwise). The decision to bet on red (resp. black) is denoted by f_R (resp. f_B). The decision to bet on "red or yellow" (resp. "black or yellow") is denoted by f_{RY} (resp. f_{BY}).Most of people prefer f_R to f_B and f_{BY} to f_{RY}. But there exist no probability measure P and no utility function u such that $EU_{P,u}(f_R) > EU_{P,u}(f_B)$ and $EU_{P,u}(f_{BY}) > EU_{P,u}(f_{RY})$. The point is that the knowledge about the world is *ambiguous*: the underlying probability measure could be any probability in the set

$$\mathcal{P} = \{P\ :\ P(\{\text{red}\}) = \tfrac{1}{3}, P(\{\text{black, yellow}\}) = \tfrac{2}{3}\}.$$

Gilboa and Schmeidler [1989] have then shown that most decision makers do use the expected utility model, but on the basis of the whole *set* of priors: they maximize the min, over $\mathcal{P}$, of the possible values of the expected utility. In Ellsberg's example, for instance, the probability of {black} belongs to $[0, \frac{2}{3}]$: the minimum value of EU for act f_B is thus 0; for act f_R, $P(\{\text{red}\}) = \frac{1}{3}$, hence the expected utility is $\frac{1}{3}$ in any case: f_R is preferred to f_B. The probability of $P(\{\text{red, yellow}\})$ varies between $\frac{1}{3}$ and 1, hence the min expected utility of act f_{RY} is $\frac{1}{3}$ (it is possible that there is no yellow ball). Probability $P(\{\text{black, yellow}\})$ is equal to $\frac{2}{3}$, hence the expected utility of act f_{BY} is $\frac{2}{3}$ in any case: f_{BY} is preferred to f_{RY}.

The topic of this paper is to study how to optimize min expected utility in problems of sequential decision making under uncertainty involving multiple priors. More precisely, we tackle the problem of optimizing this criterion in a tree endowed with multiple priors. The choice of decision trees as the basic way of representing sequential problems is motivated by its generality: no linearity assumption on utility under certainty is indeed made, and even the Markov property is not required. Note that the algorithms proposed here can nevertheless be used to "solve" an influence diagram by "unfolding" the decision diagram in a decision tree, and then selecting an optimal strategy in this tree. This approach is suitable for small or medium size decision problems.

The question of selecting a strategy in a decision tree endowed with imprecise probabilities has received little attention in the literature. It raises two types of issues: decision-theoretic ones [Jaffray and Jeleva, 2007; Jaffray, 1999], i.e. the suitability –from the normative point of view– of various decision criteria under imprecise probability in a sequential decision setting, and algorithmic ones [Kikuti *et al.*, 2011;

Huntley and Troffaes, 2008], i.e. the effective computation of an optimal strategy. In this paper, we are more concerned with the algorithmic issues.

In a decision tree endowed with clear probabilities, it is well-known that an optimal strategy can be recovered in linear time by *rolling back* the decision tree, i.e. computing a strategy recursively in a dynamic programming manner. Nevertheless, this approach is not valid anymore when optimizing min expected utility in a decision tree with multiple priors, since Bellman's principle of optimality does not hold anymore. From this point, two research directions can be explored. The first one, followed by Kikuti *et al.* [2011] is to add an assumption of *dynamic feasibility*, which amounts at seeking the strategy returned by rolling back the decision tree. This strategy will be preferred by a *consequentialist* decision maker, i.e. a decision maker whose present decision does not depend on the past nor on what she plan to do when making her first decision. Though appealing from an algorithmic viewpoint, this approach does not necessarily prevent the decision maker from selecting a strategy that could be "dominated" in some sense [Hammond, 1988]. The alternative way, proposed by [McClennen, 1990], is the resolute choice approach, where the decision maker commits to an optimal strategy viewed from the root of the decision tree, and never deviates from it later. It raises a challenging algorithmic problem, provided the combinatorial number of potential strategies. Huntley and Troffaes [2008] have adressed this issue by proposing a very generic method to compute the set of *non-dominated* strategies (to be defined later), but without entering into computational considerations: they do not study the computational complexity of the problem, and the method they provide is not implementable as it stands.

The paper is organized as follows. After introducing the framework (Section 2), we show that evaluating a strategy according to min expected utility is NP-hard (Section 3) and we study how linear programming can be used to automatize this evaluation. We then provide operational algorithms to compute an optimal strategy (Section 4), and conclude by reporting the results of numerical tests (Section 5).

2 Decision Trees with Multiple Priors

In multistage decision making, one studies problems where one has to make a sequence of decisions conditionally to events. Decision trees provides a simple and explicit representation of a sequential decision problem under uncertainty. A decision tree $\mathcal{T}$ involves three kinds of nodes: a set $\mathcal{N}_D$ of decision nodes (represented by squares), a set $\mathcal{N}_C$ of chance nodes (represented by circles) and a set $\mathcal{N}_U$ of utility nodes (leaves of the tree). A decision node can be seen as a decision variable, the domain of which corresponds to the labels of the branches starting from that node. A random variable X is assigned to each chance node, and there is one outgoing branch for each possible value $x \in \mathcal{D}(X)$, where $\mathcal{D}(X)$ is the domain of X. These branches are labelled by the probabilities of the corresponding events. The values indicated at the leaves correspond to the *utilities* of the consequences. For illustration, we now present an example of a decision tree representing a sequential problem involving an Ellsberg's urn.

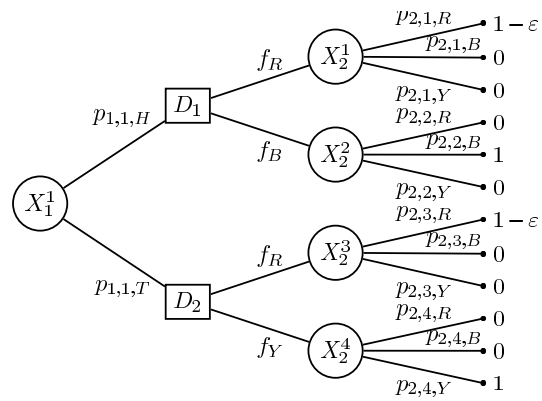

Figure 1: A decision tree involving an Ellsberg's urn.

Example 1 *Consider a game beginning with a toss coin, before a ball is drawn in an Ellsberg's urn (the same as in the introduction). Depending on whether the coin comes up heads or tails (random variable X_1, with $\mathcal{D}(X_1) = \{H, T\}$ for heads and tails), the decision maker is asked to bet on red or black (decision D_1), or red or yellow (decision D_2). The color of the ball is a random variable X_2, with $\mathcal{D}(X_2) = \{R, B, Y\}$ for red, black or yellow. If the decision maker bets on red (f_R) and a red ball is drawn, it yields a gain of $1 - \varepsilon$ (with $\varepsilon > 0$). If she bets on black or yellow (f_B or f_Y) and the guess is correct, it yields a gain of 1. The overall game can be represented by the decision tree of Figure 1. Note that random variable X_2 appears several times in the decision tree since the draw can be made in several contexts (the coin comes up heads and the decision maker chooses red, the coin comes up heads and the decision maker chooses black, etc.).*

When crisp probabilities are known, the joint probability measure over the random variables X_i is not given in extenso, but through the labelling on the tree by conditional probabilities : each branch starting from a chance node X_i^j represent an event $X_i = x$ and is endowed with a number, that represents the probability $P(X_i = x|past(X_i^j))$, where $past(X_i^j)$ denotes all the value assignments to random and decision variables on the path from the root to X_i^j. Furthermore, in this paper, we assume that $P(X_i = x|past(X_i^j))$ only depends on the random variables in $past(X_i^j)$. In the case of multiple priors, the joint probability is imprecise and it is not possible to represent it by a labeling of the edges with numbers. In the present paper, we propose to label them by variables representing the (ill-known) possible values of the conditional probabilities. Each branch starting from a chance node X_i^j represents an event $X_i = x$ and is labelled by a variable $p_{i,j,x}$ (for $P(X_i = x|past(X_i^j))$).

The joint imprecise probability should be expressed explicitly, by a set of joint probability measures, but it is not always feasible, e.g. when random variables have an infinite range of possible values. In this paper, we consider the case where the set of measures are represented by probability intervals $P(X_i \in E|past(X_i^j))$ on events E, where $E \subseteq \mathcal{D}(X_i)$. Formally, we have sets of linear constraints $\mathcal{C}_i^j$ locally defined over subsets of variables $p_{i,j,x}$ at each chance node X_i^j. A

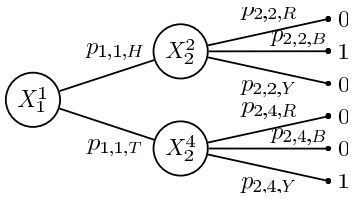

Figure 2: Lottery for strategy $(D_1 = f_B, D_2 = f_Y)$.

constraint $c \in \mathcal{C}_i^j$ takes the form $p_c^- \leq \sum_{x \in \mathcal{D}_c} p_{i,j,x} \leq p_c^+$, where $\mathcal{D}_c = \{x : p_{i,j,x} \text{ is in the scope of } c\} \subseteq \mathcal{D}(X_i)$ and p_c^-, p_c^+ are reals (if $p_c^- = p_c^+$ the interval is said *degenerated*). In Figure 1, these constraints are defined as follows:

$\mathcal{C}_1^1 = \{\frac{1}{2} \leq p_{1,1,H} \leq \frac{1}{2},\ \frac{1}{2} \leq p_{1,1,T} \leq \frac{1}{2}\}$

$\mathcal{C}_2^j = \{\frac{1}{3} \leq p_{2,j,R} \leq \frac{1}{3},\ \frac{2}{3} \leq p_{2,j,B} + p_{2,j,Y} \leq \frac{2}{3}\}$ for all j

Obviously, if the probabilities are precisely known at node X_i^j, then $\mathcal{C}_i^j$ simply assigns a value to each $p_{i,j,x}$.

Furthermore, some independencies can be taken into account by explicitly adding a set $\mathcal{C}_e$ of equality constraints over the $p_{i,j,x}$. In the example, since X_1 and X_2 are independent, we have $p_{2,1,x}$=$p_{2,2,x}$=$p_{2,3,x}$=$p_{2,4,x}$ for all x. We denote by $\mathcal{P}_\mathcal{T}$ the set of possible probability measures over decision tree $\mathcal{T}$ that are compatible with constraints $\mathcal{C} = (\bigcup_{i,j} \mathcal{C}_i^j) \cup \mathcal{C}_e$.

In a decision tree $\mathcal{T}$, a *strategy* consists in setting a value to every decision node, i.e. to every decision variable *conditionally to the past*. The decision tree in Figure 1 includes 4 feasible strategies, among which for instance strategy $s = (D_1 = f_B, D_2 = f_Y)$. In our setting, a strategy can be associated to a compound lottery over the utilities, where the probabilities of the involved events are ambigous. For instance, strategy s corresponds to the compound lottery pictured in Figure 2. Comparing strategies amounts therefore to compare compound lotteries. The evaluation of a strategy (more precisely, of the corresponding compound lottery) according to min expected utility depends on the multi-prior set $\mathcal{P}_\mathcal{T}$. In this setting, we denote by $\underline{\text{EU}}(s) = \min_{P \in \mathcal{P}_\mathcal{T}} EU(P, s)$ the min expected utility of strategy s, where $EU(P, s)$ is the expected utility of s for probability measure P. A strategy s is preferred to s' if $\underline{\text{EU}}(s) > \underline{\text{EU}}(s')$. We show in the next section that the evaluation of a strategy is a combinatorial problem in itself due to the combinatorial nature of $\mathcal{P}_\mathcal{T}$.

3 Evaluating a Strategy

We now prove that evaluating a strategy according to its min expected utility is an NP-hard problem, where the size of an instance is the number of involved chance nodes.

Proposition 1 *Evaluating a strategy according to its min expected utility is an NP-hard problem, even if all non-degenerated probability intervals are* $[0, 1]$.

Proof. The proof relies on a polynomial reduction from problem 3-SAT, which can be stated as follows: given a set X of boolean variables and a collection C of clauses on X such that $|c| = 3$ for every clause $c \in C$, does there exist an assignment of truth values to the boolean variables of X that satisfies simultaneously all the clauses of C ?

Let $X = \{x_1, \ldots, x_n\}$ and $C = \{c_1, \ldots, c_m\}$. Note that evaluating a strategy amounts to evaluating a compound lottery. One associates a random variable X_i to every booolean variable x_i, with domain $\{true, false\}$. The polynomial generation of a compound lottery from an instance of 3-SAT is performed as follows. One generates a subtree T_k for each clause $c_k \in C$. The indices of the boolean variables appearing in clause c_k are denoted u_k, v_k, w_k. For instance, for clause $c_2 = \overline{x_1} \vee x_3 \vee x_4$, $u_2 = 1$, $v_2 = 3$ and $w_2 = 4$. Subtree T_k is a complete binary tree such that:

- the root (depth 0) is a chance node labeled by X_{u_k},
- nodes of depth 1 are chance nodes labeled by X_{v_k},
- nodes of depth 2 are chance nodes labeled by X_{w_k},
- nodes of depth 3 are utility nodes.

Each chance node X_i^j has two outgoing branches:

- one corresponding to $X_i = true$ and labelled by $p_i \in [0, 1]$,
- one corresponding to $X_i = false$ and labelled by $1 - p_i$.

Finally, the value of a utility node is 1 if it is on the path making false all the literals of c_k, otherwise its value is 0.

Once all subtrees T_k have been generated, a unique root node X_0 is generated, which is parent of all subtrees T_k. A probability $1/m$ is assigned to each branch from X_0. This concludes the reduction. Clearly, the generated tree can be computed in polynomial time. As an illustration, in Figure 3, we give the compound lottery generated from the following instance of 3-SAT: $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_3 \vee x_4)$.

Let us show that the 3-SAT formula is satisfiable if and only if the min expected utility of the compound lottery is 0. If there is an instantiation $\{x_1 = a_1, x_2 = a_2, \ldots, x_n = a_n\}$ that satisfies the 3-SAT formula, then the min expected utility of the compound lottery L is 0. Indeed, setting each variable p_i to 1 if a_i is equal to "true", and 0 otherwise, yields a min expected utility equal to 0. This is related to the fact that each utility node of value 1 is the terminal node of the unique path in T_k that makes false clause c_k. As clause c_k is made true by the instanciation, this utility node has probability 0 to be reached.

Conversely, if the min expected utility of L is 0, then the 3-SAT formula is satisfiable. Indeed, given an instantiation of parameters p_i making zero the min expected utility of the lottery, the probability to reach a nonzero utility node is necessarily 0. This implies that, for every path to a nonzero utility node, at least one probability p_i is 0 (label p_i) or 1 (label $1 - p_i$). By setting value "true" (resp. "false") to x_i if $p_i = 1$ (resp. 0), the 3-SAT formula is satisfied. ■

We now describe a procedure to evaluate a strategy, based on a mathematical programming formulation. Before providing the formal expression of min expected utility $\underline{\text{EU}}(s)$ of a strategy s, we need to introduce some notations. We denote by $X_1, \ldots, X_n$ the set of random variables appearing in a decision tree $\mathcal{T}$, and by $X = \langle X_1, \ldots, X_n \rangle$ the corresponding random vector. We denote by $\mathcal{D}(Y)$ the domain of a random variable (or random vector) Y. Furthermore, given a subset $I \subseteq \{1, \ldots, n\}$ of indices, we denote by X_I the random vector whose components are X_i $(i \in I)$. Besides, given a node N, we denote by $\pi(N)$ the set of indices of the random variables on the path from the root to N (random variable

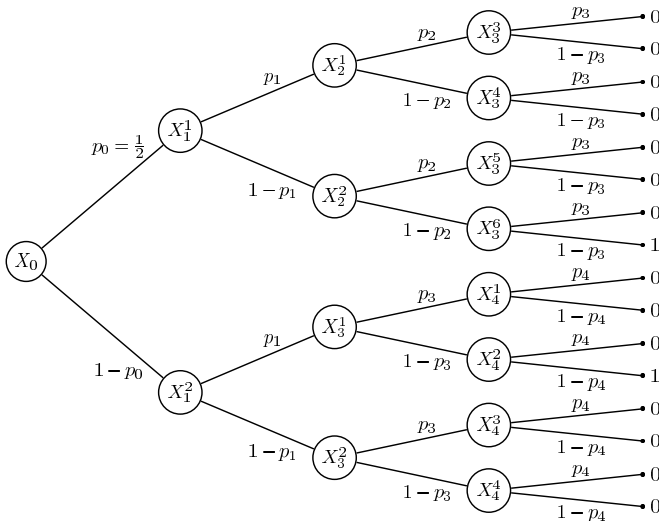

Figure 3: An example of reduction for the following instance of 3-SAT: $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_3 \vee x_4)$.

in N excluded if $N \in \mathcal{N}_C$). The complementary set is denoted by $\bar{\pi}(N) = \{1, \ldots, n\} \setminus \pi(N)$. Finally, we denote by $x(N)$ the assignment vector of $X_{\pi(N)}$ that leads to node N. For instance, consider the decision tree of Figure 1. We have $\pi(X_2^1) = \{1\}$ since random variable X_1 (node X_1^1) appears on the path from X_1^1 to X_2^1. Thus $X_{\pi(X_2^1)} = \langle X_1 \rangle$, and $x(X_2^1) = \langle H \rangle$ since $X_1 = H$ on the considered path.

In a decision tree $\mathcal{T}$, a strategy s is characterized by the subset $\mathcal{U}_s \subset \mathcal{N}_U$ of utility nodes it enables to reach. The min expected utility $\underline{\text{EU}}(s)$ of s can thus be written as follows:

$$\underline{\text{EU}}(s) = \min_{P \in \mathcal{P}_\mathcal{T}} \sum_{N \in \mathcal{U}_s} P(X_{\pi(N)} = x(N)) \times u(N)$$

where $P(X_{\pi(N)} = x(N))$ represents (in accordance with our notations) the probability to reach node N when following strategy s. For instance, strategy $(D_1 = f_B, D_2 = f_Y)$ in Figure 1 yields the following formula:

$$\underline{\text{EU}}(s) = \min_{P \in \mathcal{P}_\mathcal{T}} P(X_1 = H, X_2 = B) + P(X_1 = T, X_2 = Y)$$

Given the combinatorial nature of $\mathcal{P}_\mathcal{T}$, we formulate a mathematical program to evaluate a strategy s. The main difficulty consists in characterizing the set of measures $P \in \mathcal{P}_\mathcal{T}$. The fact that the crisp probabilities satisfies constraints $\mathcal{C}$ *is indeed not sufficient to ensure the global consistency of the probability measure* over $\mathcal{T}$ [Jeantet and Spanjaard, 2009], due to the possibility of several instances of the same random variable in $\mathcal{T}$. For instance, assume that the probabilities $P(X_1)$, $P(X_2)$ and $P(X_2|X_1)$ appear in the decision tree. Then, the total probability theorem should hold: $P(X_2) = \sum_{x \in \mathcal{D}(X_1)} P(X_2|X_1 = x)P(X_1 = x)$. In order to characterize set $\mathcal{P}_\mathcal{T}$ of "feasible" probability measures, we define constraints upon atomic probabilities $P(X = x)$ instead of variables $p_{i,j,x}$. Furthermore, we consider the probabilities $P(X(N) = x(N))$ to reach each of the nodes N of the tree, provided the node is compatible with the followed strategy. The variables used in our formulation are thus not the $p_{i,j,x}$, and their number is exponential in n (number of distinct random variables), but the interest is that they are sufficient to both properly characterize a probability measure and insure its compatibility with $\mathcal{C}$. So that Kolmogorov's axioms hold, it is necessary to satisfy the following constraint:

$$\sum_{x \in \mathcal{D}(X)} P(X = x) = 1$$

In Example 1, one has: $\sum_{\substack{x_1 \in \mathcal{D}(X_1) \\ x_2 \in \mathcal{D}(X_2)}} P(X_1 = x_1, X_2 = x_2) = 1.$

Besides, in order to link atomic probabilities $P(X = x)$ to probabilities $P(X_{\pi(N)} = x(N))$, we introduce the following linear constraints for all $N \in \mathcal{N}$ (there are some redundancies, but we do not elaborate here to save space):

$$P(X_{\pi(N)} = x(N)) = \sum_{y \in \mathcal{D}(X_{\bar{\pi}(N)})} P(X = (x(N), y))$$

where (x, y) denote the assignment of X such that $X_{\pi(N)} = x$ and $X_{\bar{\pi}(N)} = y$. In Example 1, for $N = X_2^1$, it writes:

$$P(X_1 = H) = \sum_{x_2 \in \mathcal{D}(X_2)} P(X_1 = H, X_2 = x_2)$$

Then, to insure that constraints $\mathcal{C}_i^j$ are satisfied, we introduce the following constraints for each chance node X_i^j by using Bayes' rule: $\forall c \in \mathcal{C}_i^j$,

$$P(X_{\pi(X_i^j)} = x(X_i^j)) \times p_c^- \leq \sum_{z \in \mathcal{D}_c} P(X_{\pi(X_i^j) \cup \{i\}} = (x(X_i^j), z))$$

$$\sum_{z \in \mathcal{D}_c} P(X_{\pi(X_i^j) \cup \{i\}} = (x(X_i^j), z)) \leq P(X_{\pi(X_i^j)} = x(X_i^j)) \times p_c^+$$

where $\mathcal{D}_c$ is the subdomain of X_i involved in constraint c (see Section 2), and $X_{\pi(X_i^j) \cup \{i\}} = (x, z)$ means that $X_{\pi(X_i^j)} = x$ and $X_i = z$. In Example 1, it writes for X_2^1, :

$\frac{1}{3}P(X_1 = H) \leq P(X_1 = H, X_2 = R) \leq \frac{1}{3}P(X_1 = H)$
$\frac{2}{3}P(X_1 = H) \leq P(X_1 = H, X_2 = B) + P(X_1 = H, X_2 = Y)$
$P(X_1 = H, X_2 = B) + P(X_1 = H, X_2 = Y) \leq \frac{2}{3}P(X_1 = H)$

Finally, to insure that constraints $\mathcal{C}_e$ are satisfied, we need the following bilinear constraint, by using Bayes' rule:

$$P(X_i = x, X_{\pi(X_i^j)} = x(X_i^j)) \times P(X_{\pi(X_{i'}^{j'})} = x(X_{i'}^{j'}))$$
$$= P(X_{i'} = x, X_{\pi(X_{i'}^{j'})} = x(X_{i'}^{j'})) \times P(X_{\pi(X_i^j)} = x(X_i^j))$$

for each constraint $p_{i,j,x} = p_{i',j',x'}$ in $\mathcal{C}_e$. In Example 1, for constraint $p_{2,1,R} = p_{2,3,R}$, it writes: $P(X_1 = T) \times P(X_1 = H, X_2 = R) = P(X_1 = H) \times P(X_1 = T, X_2 = R)$.

The characterization of $\mathcal{P}_\mathcal{T}$ thus requires $\mathcal{O}(|\mathcal{N}| + |\mathcal{D}(X)|)$ variables (representing the probabilities to reach each of the nodes, and the atomic probabilities) and $\mathcal{O}(|\mathcal{N}| + |\mathcal{C}|)$ constraints (the constraints to link the atomic probabilities to the probabilities to reach each of the nodes, and the constraints to insure compatibility with $\mathcal{C}$). Clearly, $|\mathcal{D}(X)|$ (and hence n) appears to be a key parameter for the spatial (and time) efficiency of this formulation, and the numerical tests will confirm it. Now that $\mathcal{P}_\mathcal{T}$ is characterized, we are able to evaluate a strategy by using an off-the-shelf optimization software.

4 Selecting a Strategy

Due to their combinatorial number, the enumeration of all strategies is quickly cumbersome. Furthermore, when trying to maximize min expected utility in a decision tree, it is

important to note that the optimality principle does not hold. For illustration, let us perform the rolling back procedure on the decision tree of Figure 1. In D_1 and D_2, the decision maker prefers decision f_R, since it provides a min expected utility of $(1-\varepsilon)/3$, while it is 0 for the other decision. The strategy returned by rolling back the decision tree is therefore $s = (D_1 = f_R, D_2 = f_R)$. The min expected utility of s is $(1-\varepsilon)/3$, while strategy $(D_1 = f_B, D_2 = f_Y)$ yields $1/3$. Hence, to compute a strategy optimizing min expected utility one cannot resort to standard dynamic programming in the general case. Nevertheless, there exist cases authorizing it. We describe one such case in the next paragraph.

Separable Decision Trees. For each chance node X_i^j, we denote by $\mathcal{P}_i^j$ the set of conditional probability distributions over $X_i|past(X_i^j)$ that satisfies constraints $\mathcal{C}_i^j$. A decision tree $\mathcal{T}$ is called *separable* (or *separately specified* [Kikuti *et al.*, 2011]) if $\mathcal{P}_\mathcal{T} = \prod_{X_i^j \in \mathcal{N}_C} \mathcal{P}_i^j$. In such a tree, it is valid to use a rolling back procedure for maximizing min expected utility. In this procedure, comparing strategies s and s' in a subtree amounts to compare the induced (ambiguous) lotteries L_s and $L_{s'}$. The validity of the rolling back procedure is related to the fulfillment of the following property:

$$\underline{\text{EU}}(\lambda L_s + (1-\lambda)L_{s'}) = \lambda \underline{\text{EU}}(L_s) + (1-\lambda)\underline{\text{EU}}(L_{s'})$$

where $\lambda \in [0,1]$ and $\lambda L_s + (1-\lambda)L_{s'}$ is the compound lottery that yields lottery L_s (resp. $L_{s'}$) with probability λ (resp. $(1-\lambda)$). The fulfillment of this property directly follows from the separability assumption [Jeantet and Spanjaard, 2009]. The optimal min expected utility $\underline{\text{EU}}^*(N)$ at node N is computed recursively by using the following formulas:

- At a decision node D_i: $\underline{\text{EU}}^*(D_i) = \max_{d \in \mathcal{D}(D_i)} \underline{\text{EU}}^*(D_i = d)$.

where $\mathcal{D}(D_i)$ is the domain of decision variable D_i.

- At a chance node X_i^j:

$$\underline{\text{EU}}^*(X_i^j) = \min_{P \in \mathcal{P}_i^j} \sum_{x \in \mathcal{D}(X_i)} P(X_i = x|past(X_i^j))\underline{\text{EU}}^*(N_x).$$

where N_x is the node reached if $X_i = x$.

The computation of $\underline{\text{EU}}^*(X_i^j)$ according to the above formula requires the resolution of a *linear* program whose set of constraints is C_i^j (note that the separability assumption implies of course that $\mathcal{C}_e = \emptyset$). The number of such (small size) linear programs solved during the running of the procedure is clearly linear in the number of chance nodes.

Non-separable Decision Trees. In the general (non-separable) case, the rolling back method does not work when directly operating with min expected utility. However, one can use the following simple property: if a strategy s is dominated by another strategy s', i.e. $EU(P,s) \leq EU(P,s')$ for all $P \in \mathcal{P}_T$, then $\underline{\text{EU}}(s) \leq \underline{\text{EU}}(s')$. The main idea is therefore to compute the set of all non-dominated strategies (w.r.t. the asymetric part of the previous relation) in a first phase, and then, in a second phase, to recover the optimal one in this set according to min expected utility. The evaluation procedure of Section 3 can be used to determine the best strategies in the second phase. It has been proved that the set of non-dominated strategies can be computed by rolling back the decision tree [Huntley and Troffaes, 2008]. We provide here an operational approach to perform this procedure in our setting. The set $S^*(N)$ of non-dominated strategies at node N is computed recursively:

- At a decision node N: $S^*(N) = ND(\bigcup_{i=1}^k S^*(N_i))$ where $N_1, \ldots, N_k$ are the children of N and $ND(S)$ returns the set of non-dominated strategies in S.

- At a chance node N:

$S^*(N) = ND(\{s_1 \oplus \ldots \oplus s_k : s_i \in S^*(N_i)\ \forall i\})$ where $N_1, \ldots, N_k$ are the children of N and $\oplus$ is the combination operation of substrategies.

The key primitive in this procedure is function $ND(\cdot)$. To compute the non-dominated elements of a set S, one performs pairwise comparisons of strategies in S. Such comparisons are called *dominance test* hereafter. A strategy s dominates a strategy s' if there does not exist $P \in \mathcal{P}_\mathcal{T}$ (i.e., satisfying the constraints presented in Section 3) such that:

$$\sum_{N \in \mathcal{U}_s} P(X_{\pi(N)}{=}x(N)) \times u(N) < \sum_{N \in \mathcal{U}_{s'}} P(X_{\pi(N)}{=}x(N)) \times u(N)$$

where $\mathcal{U}_s$ is the subset of utility nodes that s reaches. The test thus amounts to solve a set of linear constraints.

$h \setminus d$	2		3		4	
	# nodes	time	# nodes	time	# nodes	time
8	511	< 1	2073	< 1	5851	< 1
10	2047	< 1	12441	< 1	46811	< 1
12	8191	< 1	74639	1.2	374491	5.7
14	32767	< 1	447897	7.8	2995931	65.5
16	131071	2.6	2687385	57.3	X	
18	524287	10.8	X		X	
20	2097151	43.2	X		X	

Table 1: Separable case: average execution time (in sec.) according to height h and outdegree d of chance nodes.

5 Numerical Tests

The proposed algorithms have been implemented in C++, and the CPLEX solver has been used to solve the mathematical programs. The numerical tests have been performed on a Pentium IV 2.13GHz CPU computer with 3GB of RAM.

For the *separable case*, one considers decision trees of (even) depth h, where all chance (resp. decision) nodes have outdegree d (resp. 2). The utilities are real numbers randomly drawn in $[0, 100]$. To define the imprecise probability, or equivalently the constraints C_i^j, one generates $d-1$ constraints of arity $d-1$ (two constraints of arity 1 if $d = 2$). For each constraint c, one sets $p_c^+ - p_c^- = 0.1$. The average execution times on 20 such random instances are reported in Table 1. Timeout was set to 100 sec., and symbol X appears as soon as the running time exceeded the timeout for at least one instance. Notice that our procedure was able to solve decision trees with up to 2m nodes and that, unsurprizingly, ambiguity degree d has a great impact on the resolution times.

For the *non-separable case*, one considers binary and complete decision trees of even depth h, with real utilities in $[0, 100]$. The binary assumption allows to consider a larger range of depths for the decision trees. To generate the constraints C_i^j on the imprecise probabilities, we first generate a crisp probability distribution P over n random variables. Then, for each chance node, a random variable is randomly

h	4				6				8				10			
$n \setminus w$	0.01	0.05	0.1	0.5	0.01	0.05	0.1	0.5	0.01	0.05	0.1	0.5	0.01	0.05	0.1	0.5
5	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	37.2	1.0	1.4	3.5	X
6	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	107.6	6.4	9.3	11.7	X
7	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	X	19.1	22.0	23.0	X
8	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	1.1	1.4	2.0	X	35.2	51.6	58.0	X
9	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	1.9	2.2	2.4	X	87.8	114.9	142.3	X
10	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	4.1	4.3	4.6	X	199.2	253.6	328.1	X
11	< 1	< 1	< 1	< 1	< 1	< 1	< 1	< 1	6.5	7.4	7.7	X	409.0	590.7	X	X
12	< 1	< 1	< 1	< 1	< 1	< 1	< 1	1.6	12.1	14.3	17.8	X	X	X	X	X
13	< 1	< 1	< 1	< 1	< 1	< 1	< 1	6.3	20.2	28.1	33.5	X	X	X	X	X
14	-	-	-	-	1.1	1.3	1.4	11.9	57.7	59.6	66.4	X	X	X	X	X
15	-	-	-	-	3.4	3.8	4.2	20.7	104.2	121.3	137.5	X	X	X	X	X
16	-	-	-	-	6.7	7.0	7.2	41.0	231.4	259.0	286.2	X	X	X	X	X
17	-	-	-	-	13.9	15.5	16.0	98.2	493.1	533.8	581.9	X	X	X	X	X
18	-	-	-	-	30.1	34.9	38.3	180.9	X	X	X	X	X	X	X	X
19	-	-	-	-	65.5	68.0	73.9	352.1	X	X	X	X	X	X	X	X
20	-	-	-	-	143.4	155.3	164.8	702.8	X	X	X	X	X	X	X	X
21	-	-	-	-	307.3	312.1	322.4	X	X	X	X	X	X	X	X	X
ND	1	2	2	4	2	3	4	27	2	5	8	159	3	29	32	> 300

Table 2: Non-separable case: average exec. time (in sec.) according to height h, number n of random variables, and w=p^+-p^-.

n	14	15	16	17	18	19	20	21
	< 0.01	0.02	0.07	0.49	0.96	1.99	7.92	15.97

Table 3: Average execution time (in sec.) of a positive dominance test according to number n of random variables.

assigned. At each chance node, following Bayes' rule, we compute the crisp conditional probabilities according to P. Finally, each conditional probability p on a branch is replaced by an interval $[p^-, p^+]$ including p. The imprecision degree $w = p^+ - p^-$ ranges in $\{0.01, 0.05, 0.1, 0.5\}$ (one value for the whole tree). Furthermore, and importantly, one sets $\mathcal{C}_e = \emptyset$ in all instances. This assumption prevents from the insertion of bilinear constraints, and the CPLEX solver can therefore safely be used for evaluating strategies and testing dominance. The average execution times over 20 random instances are reported in Table 2 (symbol "-" means that n is greater than the number of chance nodes). Line ND gives the average number of non-dominated strategies at the root. Timeout was set to 1000 seconds. The size of the tackled instances are of course smaller than in the separable case, since the procedure handles sets of non-dominated elements, which is computationally more demanding than simply rolling back single values. As the dominance test is an important primitive in the procedure, Table 3 gives the average execution times of such tests according to the number n of random variables. We report here only the cases when a dominance relation is actually detected (average over 50 positive cases), since it takes longer than when it does not exist. Consistently with our expectation, the times strongly increase with n, that appears to be a key parameter for the efficiency of the procedure.

6 Conclusion

The optimization of min expected utility in sequential decision problems with multiple priors is known to be difficult from the algorithmic point of view, since the usual principles of dynamic programming do not apply. We have shown in this paper that the problem is indeed intractable. Evaluating a given strategy is NP-hard, even in a simple variant where the distinct random variables are mutually independent. Facing this difficulty, we distinguished two cases: separable and non-separable decision trees. For each case, we have proposed a dedicated procedure to tackle the problem. Our first experiments are promising, although classic improvements still have to be explored, e.g. column generation [Kikuti *et al.*, 2011], use of bounds to prune the search, etc.

The model proposed here is nevertheless quite simple from the viewpoint of expressivity. In a more general setting, one should provide the user with a richer way of expressing her knowledge about the random variables, e.g. by a credal network. Clearly, the more expressive the setting, the higher the risk of an increase in complexity. The challenge is to balance expressivity of the model and algorithmic efficiency.

References

[Ellsberg, 1961] D. Ellsberg. Risk, ambiguity and the savage axioms. *Quaterly Journal of Economics*, 75:643–669, 1961.

[Gilboa and Schmeidler, 1989] I. Gilboa and D. Schmeidler. Maxmin expected utility with non-unique prior. *Journal of Economic Theory*, 18:141–153, 1989.

[Hammond, 1988] P. Hammond. Consequentialist foundations for expected utility. *Theory and Decision*, 25:25–78, 1988.

[Huntley and Troffaes, 2008] N. Huntley and M. C.M. Troffaes. An efficient normal form solution to decision trees with lower previsions. In *Soft Methods for Hand. Var. and Imprecision*, pages 183–188, 2008.

[Jaffray and Jeleva, 2007] J.-Y. Jaffray and M. Jeleva. Information processing under imprecise risk with the Hurwicz criterion. In *5th Int. Symp. of Imp. Proba: Th. and App.*, pages 233–242, 2007.

[Jaffray, 1999] J.-Y. Jaffray. Rational decision making with imprecise probabilities. In *1st International Symposium on Imprecise Probability: Theories and Applications*, pages 183–188, 1999.

[Jeantet and Spanjaard, 2009] G. Jeantet and O. Spanjaard. Optimizing the Hurwicz criterion in decision trees with imprecise probabilities. In *Alg. Dec. Th., LNAI 5783*, pages 340–352, 2009.

[Kikuti *et al.*, 2011] D. Kikuti, F. G. Cozman, and R. S. Filho. Sequential decision making with partially ordered preferences. *Artificial Intelligence*, 175:1346–1365, 2011.

[McClennen, 1990] E.F. McClennen. *Rationality and Dynamic choice: Foundational Explorations*. Cambridge Press, 1990.

[von Neuman and Morgenstern, 1947] J. von Neuman and O. Morgenstern. *Theory of games and economic behaviour*. 1947.