Artificial Intelligence 175 (2011) 1366–1389

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Computing rank dependent utility in graphical models for sequential decision problems ☆

Gildas Jeantet, Olivier Spanjaard *

Laboratoire d'Informatique de Paris 6 (LIP6-CNRS), Université Pierre et Marie Curie (UPMC), 4 Place Jussieu, F-75252 Paris Cedex 05, France

ARTICLE INFO

Article history:
Received 2 March 2009
Received in revised form 31 August 2010
Accepted 31 August 2010
Available online 2 December 2010

Keywords:
Algorithmic decision theory
Rank dependent utility
Decision trees
Influence diagrams
Planning under uncertainty

ABSTRACT

This paper is devoted to automated sequential decision in AI. More precisely, we focus here on the Rank Dependent Utility (RDU) model. This model is able to encompass rational decision behaviors that the Expected Utility model cannot accommodate. However, the non-linearity of RDU makes it difficult to compute an RDU-optimal strategy in sequential decision problems. This has considerably slowed the use of RDU in operational contexts. In this paper, we are interested in providing new algorithmic solutions to compute an RDU-optimal strategy in graphical models. Specifically, we present algorithms for solving decision tree models and influence diagram models of sequential decision problems. For decision tree models, we propose a mixed integer programming formulation that is valid for a subclass of RDU models (corresponding to risk seeking behaviors). This formulation reduces to a linear program when mixed strategies are considered. In the general case (i.e., when there is no particular assumption on the parameters of RDU), we propose a branch and bound procedure to compute an RDU-optimal strategy among the pure ones. After highlighting the difficulties induced by the use of RDU in influence diagram models, we show how this latter procedure can be extended to optimize RDU in an influence diagram. Finally, we provide empirical evaluations of all the presented algorithms.

1. Introduction

In many AI problems, agents must act under uncertainty (e.g. in robot control, relief organization, medical diagnosis, games, etc.). When the consequences of an action only depend on events whose probabilities are known, *decision theory under risk* provides useful tools to automate decisions. The purpose of this theory is indeed to design *decision criteria* to evaluate probability distributions on outcomes (called hereafter *lotteries*) according to the preferences of a decision maker. A popular criterion is the expected utility (EU) model proposed by von Neumann and Morgenstern [49]. In this model, an agent is endowed with a *utility function* u that assigns a numerical value to each outcome. The evaluation of a lottery $L = (p_1, x_1; \ldots; p_n, x_n)$ (i.e., the lottery that yields outcome x_i with probability p_i) is then performed via the computation of its utility expectation: $EU(L) = \sum_{i=1}^{n} p_i u(x_i)$. However, despite its intuitive appeal, the EU model does not make it possible to account for all rational decision behaviors. An example of such impossibility is the so-called Allais' paradox [2] (Table 1). We present below a very simple version of this paradox due to Kahneman and Tversky [23] (Table 2).

Example 1 *(Kahneman and Tverky's example).* Consider a choice situation where two options are presented to a decision maker. He chooses between lottery L_1 and lottery L'_1 in a first problem, and between lottery L_2 and lottery L'_2 in a second

☆ This paper extends preliminary results of the two authors [20].

* Corresponding author.
E-mail address: olivier.spanjaard@lip6.fr (O. Spanjaard).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.11.019

Table 1
Original Allais' paradox. Columns represent probabilities, 100M stands for 100 millions. Most people prefers simultaneously L_1 to L'_1 and L'_2 to L_2.

Lottery	0.01	0.1	0.89
L_1	\$100M	\$100M	\$100M
L'_1	\$0M	\$500M	\$100M
L_2	\$100M	\$100M	\$0M
L'_2	\$0M	\$500M	\$0M

Table 2
Kahneman and Tversky's version. Columns represent outcomes. Most people prefers simultaneously L_1 to L'_1 and L'_2 to L_2.

Lottery	\$0	\$3000	\$4000
L_1	0.00	1.00	0.00
L'_1	0.10	0.00	0.90
L_2	0.90	0.10	0.00
L'_2	0.91	0.00	0.09

problem (see Table 2). In the first problem he prefers L_1 to L'_1 (he is certain to earn \$3000 with L_1 while he might earn nothing with L'_1), while in the second problem he prefers L'_2 to L_2 (the probability of earning \$4000 with L'_2 is almost the same as the probability of earning only \$3000 with L_2). The EU model cannot simultaneously account for both preferences. Indeed, the preference for L_1 over L'_1 implies $u(3000) > 0.1u(0) + 0.9u(4000)$. This is equivalent to $0.1u(3000) > 0.01u(0) + 0.09u(4000)$, and therefore to $0.9u(0) + 0.1u(3000) > 0.91u(0) + 0.09u(4000)$ (by adding $0.9u(0)$ on both sides). Hence, whatever utility function is used, the preference for L_1 over L'_1 implies the preference for L_2 over L'_2 in the EU model.

Actually, Allais points out that this preference reversal, far from being paradoxical, is the consequence of a reasonable behavior of *preference for security in the neighborhood of certainty* [3]. In other words, "a bird in the hand is worth two in the bush" (preference for L_1 over L'_1). It is known as the *certainty effect*. The preference reversal can be explained as follows: when the probability of winning becomes low, the sensitivity to the value of earnings increases while the sensitivity to the probabilities decreases. To encompass the certainty effect in a decision criterion, the handling of probabilities should therefore not be linear. Given this situation, new models have been developed: some models are grounded on an alternative representation of uncertainty such as the theory of possibility [12], others try to sophisticate the definition of expected utility such as prospect theory [23], cumulative prospect theory [48] or the rank dependent utility (RDU) model introduced by Quiggin [40]. This latter model is one of the most popular generalization of EU. In this model, a non-linear probability weighting function φ is incorporated in the expectation calculus, which gives a greater expressive power. In particular, the RDU model is compatible with both versions of Allais' paradox. Furthermore, the probability weighting function φ is also useful to model the attitude of the agent towards the risk. Indeed, unlike the EU model, the RDU model makes it possible to distinguish between weak risk aversion (i.e., if an option yields a guaranteed utility, it is preferred to any other risky option with the same expected utility) and strong risk aversion (i.e., if two lotteries have the same expected utility, then the agent prefers the lottery with the minimum spread of possible outcomes). For this reason, the RDU criterion has been used in search problems under risk in state space graphs, with the aim of finding optimal paths for risk-averse agents [38]. Note that, within the AI community, the rank dependent utility function is best known under the name of *Weighted Ordered Weighted Averaging* operator [45,46]. In particular, the WOWA operator has been studied in several fields of AI where an aggregation function is required: synthesis of information [45], decision making under risk [34,36], metadata aggregation problems [10], interactive techniques in multicriteria optimization [35].

The algorithmic issues related to the use of RDU in *sequential decision problems* have prevented its adoption in this setting until today. In a sequential decision problem under risk, one does not make a simple decision but one follows a *strategy* (i.e. a sequence of decisions conditioned by events) resulting in a non-deterministic outcome. This type of problem is in particular encountered in *decision-theoretic planning* [6,7]. This term refers to planners involving decision-theoretic tools. Formally, the aim of a decision-theoretic planner is to find a plan optimizing a given decision criterion. For this purpose, the lottery induced by each plan is evaluated according to a decision criterion (usually EU). Several representation formalisms can be used for sequential decision problems, such as decision trees [e.g. 41], influence diagrams [e.g. 44] or Markov decision processes [e.g. 11,22]. A decision tree is an explicit representation of a sequential decision problem, while influence diagrams or Markov decision processes are compact representations and make it possible to deal with decision problems of greater size. It is important to note that, in all these formalisms, the set of potential strategies is combinatorial (i.e., its size increases exponentially with the size of the instance). The computation of an optimal strategy for a given representation and a given decision criterion is then an algorithmic issue in itself. Contrary to the computation of a strategy maximizing EU, one cannot directly resort to dynamic programming for computing a strategy maximizing RDU (due to its non-linearity). Evaluating a decision tree or an influence diagram according to RDU (i.e., computing an optimal strategy according to RDU) raises therefore a challenging algorithmic problem. This is precisely the issue we tackle in this paper.

The paper is organized as follows. In Section 2, we recall the main features of RDU. In Section 3, we place our work in the stream of research aiming at incorporating risk-sensitivity in probabilistic planning problems. Then, in Section 4, after detailing how the RDU criterion should be used in a sequential decision problem, we propose two approaches for optimizing RDU in a decision tree, and we provide numerical tests for both approaches. In Section 5, we investigate the optimization of RDU in an influence diagram. After recalling the influence diagram formalism, we highlight a difficulty that was not present in the decision tree formalism, namely that not all strategies are considered in an influence diagram. We then propose an approach to overcome this difficulty, and provide numerical tests that show its interest.

2. Rank dependent utility

Given a finite set $S = \{x_1, \ldots, x_n\}$ of outcomes, any strategy in a sequential decision problem can be seen as a *lottery*, characterized by a probability distribution P over S. In this paper, unless explicitely mentioned, we assume that the outcomes are real numbers ordered as follows: $x_1 < \cdots < x_n$. We denote by $L = (p_1, x_1; \ldots; p_n, x_n)$ the lottery that yields outcome x_i with probability $p_i = P(\{x_i\})$. The decumulative function G_L is given by $G_L(\alpha) = \sum_{i:x_i \geqslant \alpha} p_i$, and is denoted by $(G_L(x_1), x_1; \ldots; G_L(x_n), x_n)$. For the sake of clarity, we will consider a lottery L as a function from S to $[0, 1]$ such that $L(x_i) = p_i$. As indicated above, in decision problems, lotteries are compared according to a decision criterion, as for instance EU.

Rank Dependent Utility (RDU), introduced by Quiggin [40], is among the most popular generalizations of EU, and makes it possible to describe sophisticated rational decision behaviors. From the axiomatic viewpoint, the RDU model is grounded on a weakening of the *sure thing principle* [43] that we now detail. It is indeed well known that the sure thing principle holds when using the EU criterion. In the framework of decision under risk, this principle can be stated as follows: let lotteries $L_1 = (p_1, x_1; \ldots; p_n, x_n)$ and $L'_1 = (p_1, x'_1; \ldots; p_n, x'_n)$ be such that the outcomes are not necessarily ranked in increasing order and $x_{i_0} = x'_{i_0}$ (where i_0 is an arbitrary index in $\{1, \ldots, n\}$), then L_1 preferred to L'_1 implies L_2 preferred to L'_2, for lotteries L_2, L'_2 obtained from lotteries L_1 and L'_1 by merely replacing the common outcome x_{i_0} by another common outcome y_{i_0}. In Allais' paradox (Table 1), the sure thing principle is clearly violated. There is indeed a 89 percent chance to win \$100 millions in L_1 and L'_1, while there is a 89 percent chance to win nothing in L_2 and L'_2, *ceteris paribus*. In order to encompass such examples, the validity of the axiom has to be restricted to cases where the common outcome is ranked similarly in both lotteries, and where its replacement does not affect the ranking in both lotteries: L_1 preferred to L'_1 implies L_2 preferred to L'_2, for lotteries L_2, L'_2 obtained from lotteries L_1 and L'_1 by merely replacing the i_0^{th} common outcome x_{i_0} by a common outcome y_{i_0}, again in i_0^{th} rank both in L_2 and L'_2. This weaker version of the axiom is called *comonotonic sure thing principle* [9]. It allows preference reversals in cases of extreme change in the level of risk. For instance, in Allais' paradox, there is an extreme change in the level of risk since, in the first comparison, the probability to earn nothing is about 1 percent, while, in the second comparison, it is about 90 percent. Comonotonic sure thing principle, together with a continuity axiom and an axiom of compatibility with stochastic dominance, characterize the RDU model. A lottery $L = (p_1, x_1; \ldots; p_k, x_k)$ is said to *stochastically dominate* a lottery $L' = (p'_1, x'_1; \ldots; p'_k, x'_k)$ if for all $\alpha \in \mathbb{R}$, $G_L(\alpha) \geqslant G_{L'}(\alpha)$. In other words, for all $\alpha \in \mathbb{R}$, the probability to get an outcome at least α with lottery L is at least as high as the probability with lottery L'. Compatibility with stochastic dominance means that lottery L is preferred to lottery L' as soon as L stochastically dominates L'. This property is obviously desirable to guarantee a rational behavior.

In order to allow preference reversals in cases of extreme change in the level of risk, the handling of probabilities in the RDU model is non-linear. In this purpose, the first proposal that may come into mind consists in distorting individual probabilities by a non-linear function φ, which yields a decision criterion of the form $\sum_{i=1}^{n} \varphi(p_i)u(x_i)$ where u denotes an increasing utility function. Actually, this choice criterion has been proposed by Handa [16], but is not compatible with stochastic dominance. For this reason, the distortion in the RDU model is not performed on the probabilities themselves, but on reverse cumulative probabilities. The formula of rank dependent expected utility can be easily obtained by rewriting the one of expected utility with respect to reverse cumulative probabilities: $EU(L) = \sum_{i=1}^{n} p_i u(x_i) = u(x_1) + \sum_{i=2}^{n} [u(x_i) - u(x_{i-1})](G_L(x_i))$ (the utility of lottery L is at least $u(x_1)$ with probability 1; then the utility might increase from $u(x_1)$ to $u(x_2)$ with probability $G_L(x_2)$; the same applies from $u(x_2)$ to $u(x_3)$ with probability $G_L(x_3)$, and so on, etc.). The rank dependent utility of a lottery L is then defined as follows:

$$RDU(L) = u(x_1) + \sum_{i=2}^{n} \big[u(x_i) - u(x_{i-1})\big]\varphi\big(G_L(x_i)\big)$$

Rank dependent utility thus involves an increasing utility function on consequences $u : S \to \mathbb{R}$ as in EU, and also a transformation function on probabilities $\varphi : [0, 1] \to [0, 1]$. It is compatible with stochastic dominance, i.e. $RDU(L) \geqslant RDU(L')$ as soon as L stochastically dominates L'. The transformation function φ is a non-decreasing function, proper to any agent, such that $\varphi(0) = 0$ and $\varphi(1) = 1$. When $\varphi(p) = p$ for all p, RDU obviously reduces to EU.

Example 2. Coming back to Example 1, we define the utility function by $u(x) = x$, and we set $\varphi(0.09) = \varphi(0.1) = 0.2$, $\varphi(0.9) = 0.7$. The preferences induced by RDU are then compatible with Kahneman and Tversky's example. Indeed, we have:

$$RDU(L_1) = u(3000) = 3000$$
$$RDU\big(L'_1\big) = u(0) + \varphi(0.9)\big(u(4000) - u(0)\big) = 2800$$

Therefore L_1 is preferred to L'_1. Similarly, we have:

$$RDU(L_2) = u(0) + \varphi(0.1)\big(u(3000) - u(0)\big) = 600$$
$$RDU\big(L'_2\big) = u(0) + \varphi(0.09)\big(u(4000) - u(0)\big) = 800$$

We conclude that L'_2 is preferred to L_2.

In order to elicit function φ, various methods have been used. Several functional forms have been proposed for function φ in the economic literature. As indicated by Quiggin [40], the simplest are functions of the form $\varphi(p) = p^\gamma$ with $\gamma \in (0, 1)$. Actually, when trying to exactly reproduce the behavior of human agents, there is empirical evidence that the function φ is in general inverse S-shaped, i.e., first concave and then convex [48,8]. This means that the probabilities of best consequences are overweighted (*potential effect*) while the probabilities of worst consequences are underweighted (*certainty effect*). The functional form proposed by Karmarkar [24] in this purpose is: $p^\gamma/(p^\gamma + (1-p)^\gamma)$ with $\gamma \in (0, 1)$. After setting the functional form of φ, one estimates the right value for parameter γ through standard non-linear regression methods (maximum likehood, least squares). For a more detailed discussion about the different possible functional forms of φ, the interested reader may refer to the book of Quiggin [40]. Note that there also exist parametric-free elicitation methods, i.e. that works without assuming a prespecified shape of function φ [1,13,47].

3. Position of the paper

By studying the use of RDU from the computational viewpoint, we place our work in the stream of research aiming at incorporating risk sensitivity in probabilistic planning problems. The non-linearity of risk-sensitive criteria raises new algorithmic difficulties. A pioneering work on this topic has been carried out by Howard and Matheson, in the framework of Markov decision processes [17]. They show how risk sensitivity may be treated by evaluating a plan via an expected utility instead of an expectation. In the case of risk-averse agents, the practicality of the approach relies on the use of an exponential utility function u (to maximize) defined by $u(x) = -\gamma^x$ for an outcome $x \in \mathbb{R}^+$ ($\gamma \in (0, 1)$). The adoption of such a utility function involves the agreement of the decision maker with the "Δ-property", namely that his attitude towards risk does not depend on his wealth level. Koenig and Simmons have performed a similar work with a slightly different representation of probabilistic planning problems, that required the design of new algorithms [25]. The representation they use is called a "probabilistic decision graph", and resembles the decision trees we study here. Then, back to the framework of MDPs, Liu and Koenig have proposed to resort to a "one-switch" utility function in order to take into account the wealth level in the preferences. For a risk-averse agent that becomes risk-neutral in the limit as its level of wealth increases, the one-switch utility function u is of the form $u(x) = x - D\gamma^x$, where $D > 0$ and $\gamma \in (0, 1)$. After exhibiting conditions guaranteeing that the optimal expected utilities of the total plan-execution reward exist and are finite for fully observable MDP models with risk-sensitive utility functions [26], the authors have proposed a functional value iteration algorithm to approximate optimal expected utilities for *one-switch utility functions* [27]. Finally, they have also proposed a policy iteration algorithm in a subsequent paper, that enables to return an optimal policy [28].

All these results are real advances to take into account more accurately risk sensitivity in probabilistic planning problems, especially in high-stakes situations. However, the induced preferences reproduce the biases of EU theory. To explain this in more details, we need previously to introduce the notions of *weak risk-aversion* and *strong risk-aversion*. An agent is said to be *weakly risk-averse* if, for any lottery L, he considers that sure lottery $(1; E(L))$ is as least as good as L, where $E(L) = \sum_{i=1}^n p_i x_i$ for $L = (p_1, x_1; \ldots; p_n, x_n)$ [4,39]. In EU theory, risk-aversion means that the agent's utility function u on outcomes is increasing and *concave*, the coefficient of risk-aversion of any agent being measured by $-u''(x)/u'(x)$ [4]. Strong risk-aversion is defined from the notion of *mean preserving spread* [42]. Basically, an agent is said to be strongly risk-averse if, between two lotteries with the same expectation, he always prefers the less spread one. Interestingly, it has been shown that a lottery L is a mean preserving spread of a lottery L' if and only if $EU(L) \leqslant EU(L')$ for all increasing and concave utility functions u, where $EU(L) = \sum_{i=1}^n p_i u(x_i)$ [40]. Consequently, when using EU to compare lotteries, any weakly risk-averse agent is also strongly risk-averse. A nice virtue of the RDU model is precisely that it enables the distinction between weak and strong risk-aversion (contrary to EU). Within this model, for a concave utility function u, the agent is weakly risk-averse iff $\varphi(p) \leqslant p$ for all $p \in [0, 1]$, and strongly risk-averse iff φ is convex (this directly follows from a result of Quiggin [40]). As stated above, rank dependent utility is thus a powerful tool for modeling risk-sensitive agents, as illustrated by Allais' paradox. This motivates our study.

4. Computing RDU in a decision tree

4.1. Decision tree formalism

A decision tree is an arborescence with three types of nodes: the *decision nodes* (represented by squares), the *chance nodes* (represented by circles), and the terminal nodes (the leaves of the arborescence). The branches starting from a decision node correspond to different possible decisions, while the ones starting from a chance node correspond to different possible events, the probabilities of which are known. The values indicated at the leaves correspond to the *utilities* of the consequences. Note that one omits the orientation of the edges when representing decision trees. For the sake of illustration, a decision tree representation of a sequential decision problem (with three strategies) is given in Fig. 1.

More formally, in a decision tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$, the set $\mathcal{N}$ of nodes is $\mathcal{N}_D \cup \mathcal{N}_A \cup \mathcal{N}_U$, where $\mathcal{N}_\mathcal{D}$ is the set of decision nodes, $\mathcal{N}_A$ the set of chance nodes and $\mathcal{N}_U$ the set of terminal nodes. The root node is denoted by $N_r \in \mathcal{N} \setminus \mathcal{N}_U$. The valuations are defined as follows: every edge $E = (A, N) \in \mathcal{E}$ such that $A \in \mathcal{N}_A$ is weighted by probability $p(E)$ of the corresponding event; every terminal node $N_U \in \mathcal{N}_U$ is labeled by its utility $u(N_U)$. Besides, we call $past(N)$ the *past* of

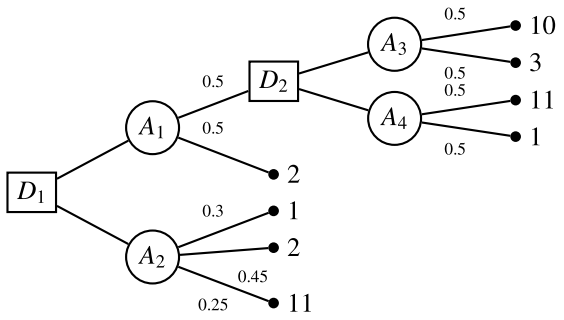


Fig. 1. A decision tree representation.

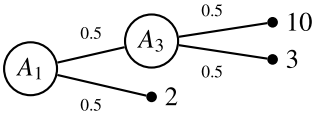


Fig. 2. A compound lotery.

$N \in \mathcal{N}$, i.e. the set of edges along the path from N_r to N in $\mathcal{T}$. Finally, we denote by $S(N)$ the set of successors of N in $\mathcal{T}$, and by $\mathcal{T}(N)$ the subtree of $\mathcal{T}$ rooted in N.

Following Jaffray and Nielsen [19], one defines a *strategy* as a set of edges $\Delta = \{(N, N'):\ N \in \mathcal{N}_D^{\Delta}, N' \in \mathcal{N}^{\Delta}\} \subseteq \mathcal{E}$, where $\mathcal{N}^{\Delta} \subseteq \mathcal{N}$ is a set of nodes including:

- the root N_r of $\mathcal{T}$,
- one and only one successor for every decision node $N \in \mathcal{N}_D^{\Delta} = \mathcal{N}_D \cap \mathcal{N}^{\Delta}$,
- all successors for every chance node $N \in \mathcal{N}_A^{\Delta} = \mathcal{N}_A \cap \mathcal{N}^{\Delta}$.

Given a decision node N, the restriction of a strategy in $\mathcal{T}$ to a subtree $\mathcal{T}(N)$, which defines a strategy in $\mathcal{T}(N)$, is called a *substrategy*.

In order to evaluate a strategy, it is important to note that a strategy can be associated to a compound lottery over the utilities. For instance, in the decision tree of Fig. 1, strategy $\{(D_1, A_1), (D_2, A_3)\}$ corresponds to the compound lottery depicted in Fig. 2. Evaluating a strategy amounts therefore to evaluating a compound lottery. From now on, for the ease of presentation, we will manipulate lotteries directly defined over utilities rather than outcomes. A lottery is then a function from U to $[0, 1]$ such that $L(x_i) = u_i$, where $U = \{u_1, \ldots, u_n\}$ with $u_1 < u_2 < \cdots < u_n$ is the image set of S with respect to u. Coming back to our example, it is natural to assume that the compound lottery of Fig. 2 is equivalent to lottery $L = (0.5, 2; 0.25, 3; 0.25, 10)$ (actually, this assumption is known as the *reduction of compound lotteries* axiom [29], which is used in the axiomatizations of EU and RDU). Given a value function V that maps every lottery with a real number (e.g. $V \equiv EU$ or $V \equiv RDU$), the evaluation of the strategy is then $V(L)$. For example, if $V \equiv EU$, then the evaluation of strategy $\{(D_1, A_1), (D_2, A_3)\}$ is $0.5 \times 2 + 0.25 \times 3 + 0.25 \times 10 = 4.25$.

4.2. Computing RDU in a decision tree: from decision theory to combinatorial optimization

In a decision tree $\mathcal{T}$, the number of potential strategies may grow exponentially with the size of the decision tree. For example, in a binary decision tree with n nodes and a strict alternation of decision/chance nodes, one can easily show that the number of strategies is in $\Theta(2^{\sqrt{n}})$. For this reason, it is necessary to develop an optimization algorithm to determine an optimal strategy in a decision tree. It is well known that the rolling back method makes it possible to compute in linear time an optimal strategy w.r.t. EU. Indeed, such a strategy satisfies the optimality principle: any substrategy of an optimal strategy is itself optimal. The optimality principle is closely related to a condition of *monotonicity* [32] on the value function. In our context, given a value function V (e.g. $V \equiv EU$), this condition can be stated as follows:

$$V(L) \geqslant V(L') \quad \Rightarrow \quad V(\alpha L + (1-\alpha)L'') \geqslant V(\alpha L' + (1-\alpha)L'')$$

where L, L', L'' are lotteries, α is a scalar in $[0, 1]$ and $\alpha L + (1-\alpha)L''$ is the lottery defined by $(\alpha L + (1-\alpha)L'')(x) = \alpha L(x) + (1-\alpha)L''(x)$. In the framework of decision theory, this condition can be seen as a weak version of the *independence* axiom used by von Neumann and Morgenstern [49] to characterize the EU criterion. This axiom states that the mixture of two lotteries L and L' with a third one should not reverse preferences (induced by V): if L is strictly preferred to L', then $\alpha L + (1-\alpha)L''$ should be strictly preferred to $\alpha L' + (1-\alpha)L''$. The monotonicity condition holds for $V \equiv EU$, which justifies that the optimality principle holds for EU.

Hence, starting from the leaves, one can compute recursively for each node the expected utility of an optimal substrategy: the optimal expected utility for a chance node equals the expectation of the optimal utilities of its successors; the optimal expected utility for a decision node equals the maximum expected utility of its successors.

Example 3. In Fig. 1, the optimal expected utility at node D_2 is $\max\{6.5, 6\} = 6.5$. Consequently, the optimal expected utility at node A_1 is 4.25. The expected utility at node A_2 is $0.3 \times 1 + 0.45 \times 2 + 0.25 \times 11 = 3.95$. The optimal expected utility

at the root node D_1 is therefore $\max\{4.25, 3.95\} = 4.25$, and the correspond strategy is $\{(D_1, A_1), (D_2, A_3)\}$. Note that this strategy can be suboptimal when using RDU to evaluate lotteries (see below).

In decision theory, the behavior of an agent that adopts such a recursively computed strategy is called *consequentialist*. More precisely, consequentialism means that the preferences between substrategies in a subtree does not depend on the rest of the decision tree. Besides, an agent is said to be *dynamically consistent* whenever at any decision node he is willing to carry out the plan of action that he determined to be optimal ex-ante. It has been proved that the preferences of an agent that is both consequentialist and dynamically consistent follow the EU model [14,15]. An agent whose preferences follow the RDU model (named hereafter RDU-maximizer) should therefore renounce either consequentialism or dynamic consistency. Assume first that the agent adopts a consequentialist behavior, that is, he computes recursively a strategy from the leaves by selecting optimal substrategies for RDU, and then follows this strategy. As shown by Example 4, this strategy, determined ex-ante, could be suboptimal since the monotonicity condition does not hold for $V \equiv RDU$, as already noticed by Nielsen and Jensen [33]. By committing himself to consequentialism, the agent thus renounces dynamic consistency (he does not follow the strategy that is optimal ex-ante).

Example 4. Consider lotteries $L = (0.5, 3; 0.5, 10)$ (corresponding to chance node A_3 in Fig. 1), $L' = (0.5, 1; 0.5, 11)$ (corresponding to chance node A_4 in Fig. 1) and $L'' = (1, 2)$. Assume that the decision maker preferences follow the RDU model with the following φ function:

$$\varphi(p) = \begin{cases} 0, & \text{if } p = 0 \\ 0.45, & \text{if } 0 < p \leqslant 0.25 \\ 0.6, & \text{if } 0.25 < p \leqslant 0.5 \\ 0.75, & \text{if } 0.5 < p \leqslant 0.7 \\ 0.8, & \text{if } 0.7 < p \leqslant 0.75 \\ 1, & \text{if } p > 0.75 \end{cases}$$

The RDU values of lotteries L and L' are:

$$RDU(L) = 3 + (10 - 3)\varphi(0.5) = 7.2$$
$$RDU(L') = 1 + (11 - 1)\varphi(0.5) = 7$$

Thus, we have $RDU(L) \geqslant RDU(L')$ (substrategy $\{(D_2, A_3)\}$ is preferred to substrategy $\{(D_2, A_4)\}$ in D_2 in Fig. 1). By the monotonicity condition for $\alpha = 0.5$, one should therefore have $RDU(0.5L + 0.5L'') \geqslant RDU(0.5L' + 0.5L'')$. However, we have:

$$RDU(0.5L + 0.5L'') = 2 + (3 - 1)\varphi(0.5) + (10 - 3)\varphi(0.25) = 5.75$$
$$RDU(0.5L' + 0.5L'') = 1 + (2 - 1)\varphi(0.75) + (11 - 2)\varphi(0.25) = 6.65$$

Therefore $RDU(0.5L + 0.5L'') < RDU(0.5L' + 0.5L'')$ (strategy $\{(D_1, A_1), (D_2, A_4)\}$ is preferred to strategy $\{(D_1, A_1), (D_2, A_3)\}$ in Fig. 1). Consequently, the monotonicity property does not hold.

More seriously, a consequentialist RDU-maximizer could even follow a stochastically dominated strategy, as shown by the following example.

Example 5. Consider the decision tree of Fig. 1. In this decision tree, the *RDU* values of the different strategies are (at the root):

$$RDU(\{(D_1, A_2)\}) = 5.8$$
$$RDU(\{(D_1, A_1), (D_2, A_3)\}) = 5.75$$
$$RDU(\{(D_1, A_1), (D_2, A_4)\}) = 6.65$$

Thus, the optimal strategy at the root is $\{(D_1, A_1), (D_2, A_4)\}$. However, by recursion, one gets at node D_2: $RDU(\{(D_2, A_3)\}) = 7.2$ and $RDU(\{(D_2, A_4)\}) = 7$. This is therefore substrategy $\{(D_2, A_3)\}$ that is obtained at node D_2 (see Example 4). At node D_1, this is thereafter the strategy $\{(D_1, A_2)\}$ (5.8 vs. 5.75 for $\{(D_1, A_1), (D_2, A_3)\}$), stochastically dominated by $\{(D_1, A_1), (D_2, A_4)\}$, which is finally obtained.

At first sight, such an example could be misinterpreted as illustrating a weakness of the RDU model in sequential decision situations. Actually, it primarily shows that the RDU model is inappropriate for consequentialist agents. Conversely, the EU model is unable to reproduce non-consequentialist behaviors. This type of behaviors is however routinely displayed by many rational agents. An intuitive example of a non-consequentialist behavior has been proposed by Machina [30].

Assume that Mom has a single treat which she can give to either daughter Abigail or son Benjamin. She is indifferent between Abigail getting the treat and Benjamin getting the treat, but she strictly prefers a coin flip over either of the sure outcomes. Assume that Abigail wins the coin flip. We are in a different state than before the coin flip, and Mom now prefers Abigail getting the treat over a new flip. History, in this case, matters as far as preferences: Benjamin *had his chance*, and therefore the fact that Benjamin could have won still matters after the coin is flipped. This is the very justification of non-consequentialism. More generally, a non-consequentialist agent still gives importance to events that could have occurred, contrary to a consequentialist agent that does not take a risk into account once it has been borne.

In this paper, we consider therefore a dynamically consistent agent (he sets a plan initially and never deviates from it later [31]) with a non-consequentialist behavior (the optimal strategy at the root can include substrategies that appear suboptimal in their subtrees). In other words, we study in the sequel how to compute an RDU-optimal plan viewed from the initial situation, and comply to it. By doing so, we are sure to never encounter a stochastically dominated substrategy, contrary to a method that would consist in performing backward induction with RDU. Unfortunately, the determination of an RDU-optimal strategy in a decision tree is an NP-hard problem (where the size of an instance is the number of involved decision nodes):

Proposition 1 *(Jeantet and Spanjaard, 2008 [20]). The determination of an RDU-optimal strategy* (*problem* RDU-OPT) *in a decision tree is an NP-hard problem.*

Proof. This is proved by polynomial reduction from 3-SAT (see Appendix A). □

Note that Jaffray and Nielsen also studied the use of RDU in decision trees [19]. Nevertheless, their approach differs from ours, since they focus on how RDU should be used by agents close to be consequentialist. Consequently, they do not compute an optimal strategy viewed from the initial situation.

4.3. Two approaches for computing RDU

We propose here two approaches for determining an RDU-optimal strategy in a decision tree. One approach uses a mixed integer linear programming formulation, and the other one proceeds by implicit enumeration (neither exhaustive enumeration nor backward induction are conceivable since RDU-OPT is NP-hard).

4.3.1. A Mixed Integer Linear Programming formulation

We now present a Mixed Integer Linear Programming (MIP) formulation of problem RDU-OPT, in the case where function φ is concave piecewise linear. Consider a decision tree $\mathcal{T}$. We first detail the set of constraints defining feasible strategies. For this purpose, a boolean variable $y_{(i,j)}$ is created in the MIP formulation for every decision branch (D_i, A_j). The $|\mathcal{N}_D|$ constraints defining the set of feasible strategies are then:

$$\sum_j y_{(1,j)} = 1$$

$$\sum_j y_{(i,j)} = y_{prev_D(i)} \quad \forall i \in \{2, \ldots, |\mathcal{N}_D|\}$$

where $y_{(i,j)} = 1$ (resp. $y_{(i,j)} = 0$) if (D_i, A_j) is selected (resp. not selected), and $prev_D(i)$ is the last decision branch preceding D_i on the path from the root (in the temporal order).

Example 6. For the decision tree represented in Fig. 3, the constraints defining the set of feasible strategies are:

$$y_{(1,1)} + y_{(1,2)} = 1$$
$$y_{(2,3)} + y_{(2,4)} = y_{(1,1)}$$
$$y_{(3,5)} + y_{(3,6)} = y_{(1,1)}$$
$$y_{(4,7)} + y_{(4,8)} = y_{(1,2)}$$
$$y_{(5,9)} + y_{(5,10)} = y_{(1,2)}$$

We now detail the modeling of the objective function. The set of utilities at the leaves of $\mathcal{T}$ is denoted by $U = \{u_1, u_2, \ldots, u_n\}$, with $u_1 \leqslant u_2 \leqslant \cdots \leqslant u_n$, and the probability to obtain utility u_h is denoted by p_h. Probability p_h is the product of all probabilities on the path from the root to utility u_h. The rank dependent utility is then written as follows:

$$u_1 + \sum_{h=2}^{n} (u_h - u_{h-1}) \varphi\left(\sum_{j \geqslant h} p_j y_{prev_u(j)} \right)$$

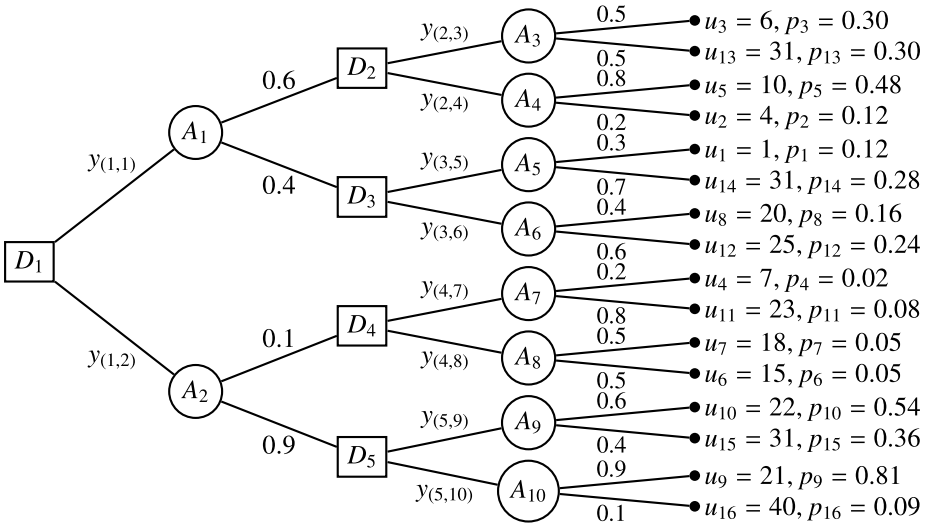


Fig. 3. A decision tree and the corresponding variables and parameters in the MIP formulation.

where $y_{prev_u(j)}$ is the last decision branch preceding u_j on the path from the root. By introducing $(n-1)$ variables φ_h, it can be rewritten as follows:

$$u_1 + \sum_{h=2}^{n} (u_h - u_{h-1})\varphi_h$$

where $\varphi_h = \varphi(\sum_{j \geqslant h} p_j y_{prev_u(j)})$ for $h = 2, \ldots, n$.

Example 7. For the decision tree represented in Fig. 3, the objective function is written:

$$1 + (4-1)\varphi_2 + \cdots + (31-25)\varphi_{13} + (31-31)\varphi_{14} + (31-31)\varphi_{15} + (40-31)\varphi_{16}$$
$$= 1 + 3\varphi_2 + 2\varphi_3 + \varphi_4 + 3\varphi_5 + 5\varphi_6 + 3\varphi_7 + 2\varphi_8 + \varphi_9 + \varphi_{10} + 2\varphi_{11} + 2\varphi_{12} + 6\varphi_{13} + 9\varphi_{16}$$

Note that variables φ_{14} and φ_{15} do not appear in the final objective function, and can therefore be eliminated from the program.

The expression defining the value of φ_h is of course non-linear, due to the presence of function φ. This difficulty can be overcome in the case where φ is concave piecewise linear. Recall that a concave function φ reflects a risk-seeking behavior for certain forms of utility function (e.g., convex).By using a concave piecewise linear function φ, one can approximate any concave regular function. This family of functions is therefore interesting to study. It is well known that any concave piecewise linear function can be written as the lower envelope of a set of affine functions. Let $\{f_1, f_2, \ldots, f_m\}$ denote this set, where $f_k(p) = a_k p + b_k$. We have $\varphi(p) = \min\{f_1(p), f_2(p), \ldots, f_m(p)\}$. The value of φ_h can be obtained by optimization:

$$\varphi_h = \max_{\alpha} \alpha$$
$$\alpha \leqslant f_k\left(\sum_{j \geqslant h} p_j y_{prev_u(j)}\right) \quad \forall k \in \{1, \ldots, m\}$$
$$\alpha \geqslant 0$$

The above problem is a linear program for a given strategy y.

Example 8. Assume that $\varphi(p) = 1.8p$ for $p \leqslant 0.5$, and $\varphi(p) = 0.4p + 0.6$ for $p > 0.5$. We have then $\varphi(p) = \min\{f_1(p), f_2(p)\}$ with $f_1(p) = 1.8p$ and $f_2(p) = 0.4p + 0.6$. For a given assignment of boolean values to variables $y_{(i,j)}$ in the decision tree represented in Fig. 3, the value of φ_{13} can be written as the result of the following linear program:

$$\varphi_{13} = \max_{\alpha} \alpha$$
$$\alpha \leqslant 1.8(0.30y_{(2,3)} + 0.28y_{(3,5)} + 0.36y_{(5,9)} + 0.09y_{(5,10)})$$
$$\alpha \leqslant 0.4(0.30y_{(2,3)} + 0.28y_{(3,5)} + 0.36y_{(5,9)} + 0.09y_{(5,10)}) + 0.6$$
$$\alpha \geqslant 0$$

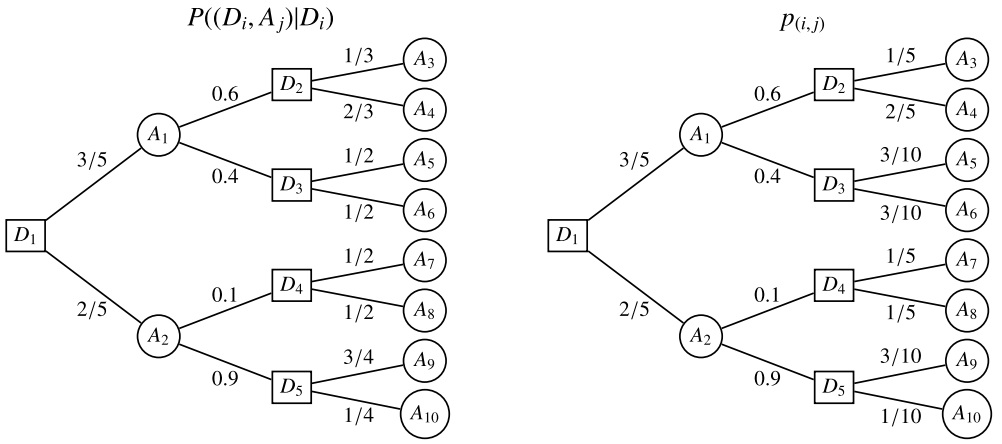


Fig. 4. One-to-one correspondence between assignments of probabilities $P((D_i, A_j)|D_i)$ and $p_{(i,j)}$.

Since $u_h - u_{h-1} \geqslant 0$, the objective function (to maximize) will benefit from maximizing every φ_h, and the MIP formulation is therefore the following in the concave piecewise case:

$$\max_{\varphi} u_1 + \sum_{h=2}^{n} (u_h - u_{h-1})\varphi_i$$

$$\varphi_h \leqslant f_k\left(\sum_{j \geqslant h} p_j y_{prev_u(j)}\right) \quad \forall h \in \{2, \ldots, n\}\ \forall k \in \{1, \ldots, m\}$$

$$\sum_j y_{(1,j)} = 1$$

$$\sum_j y_{(i,j)} = y_{prev_D(i)} \quad \forall i \in \{2, \ldots, |\mathcal{N}_D|\}$$

$$y_{(i,j)} \in \{0, 1\} \quad \varphi_h \geqslant 0$$

This program includes $(n-1)$ continuous variables, $|\mathcal{N}_D|$ binary variables, and $(n-1)m + |\mathcal{N}_D|$ constraints (since m constraints are created for every variable φ_h). Its size is therefore linear in the size of $\mathcal{T}$ for a fixed number m of pieces in φ. We recall however that the complexity of the solution procedure is of course exponential in the number of binary variables in the worst case (for a fixed number m of pieces in φ).

Let us now briefly study a relaxation of the problem, where one considers not only *pure strategies* but also *mixed strategies*. In a mixed strategy, one chooses randomly (according to a predefined probability distribution) the decision taken at each decision node. When using expected utility to evaluate a strategy, it is not worth considering mixed strategies since there always exists a pure strategy yielding the same expected utility than the best mixed strategy. This is no longer the case when using rank dependent utility to evaluate a strategy. Consider a decision tree with a single decision node leading to two different options: a sure outcome lottery $(1, 5)$, and a lottery $(0.5, 1; 0.5, 10)$. Assume that the probability transformation function is defined by $\varphi(0) = 0$, $\varphi(p) = 0.45$ if $p \in (0; 0.7)$ and $\varphi(p) = 1$ if $p > 0.7$. The RDU values of the two pure strategies are respectively 5 and 5.05. Comparatively, the mixed strategy where one chooses the sure outcome lottery with probability 0.6, and the other one with probability 0.4, results in a RDU value of 7.25. We now show that the optimal mixed strategy can be polynomially computed in the two previous cases (concave piecewise linear φ, piecewise constant φ), by slighting modifying the MIP formulations so that no boolean variables appear anymore (and the number of constraints remains the same). It yields of course a linear programming formulation. In a mixed strategy, the probability of obtaining utility u_h equals the product of the probabilities on the chance *and decision* branches along the path from the root to u_h. For this reason, a real variable $p_{(i,j)}$ is created in the MIP formulation for every decision branch (D_i, A_j), instead of a boolean variable. However, to obtain linear constraints, this variable does not represent probability $P((D_i, A_j)|D_i)$ (probability to make decision (D_i, A_j) conditionally to reach node D_i) but the product of the probabilities of the decision branches from the root to node A_j. There is a one-to-one correspondence between assignments of probabilities $P((D_i, A_j)|D_i)$ and $p_{(i,j)}$, since $P((D_i, A_j)|D_i)$ equals $p_{(i,j)}/p_{prev_D(i)}$. This is illustrated in Fig. 4. The probability of obtaining utility u_h is then $p_h p_{prev_u(h)}$ (on the path to u_h, p_h is the product of the probabilities on the chance branches, and $p_{prev_u(h)}$ is the product of the probabilities on the decision branches). The objective function is therefore written:

$$u_1 + \sum_{h=2}^{n} (u_h - u_{h-1}) \varphi\left(\sum_{j \geqslant h} p_j p_{prev_u(j)} \right)$$

Let us now study the constraints that must satisfy variables $p_{(i,j)}$. By definition, we have $\sum_j p_{(i,j)} = \sum_j p_{prev_D(i)} P((D_i, A_j)|D_i)$. Then, $\sum_j p_{prev_D(i)} P((D_i, A_j)|D_i) = p_{prev_D(i)} \sum_j P((D_i, A_j)|D_i) = p_{prev_D(i)}$. Consequently, $\sum_j p_{(i,j)} = p_{prev_D(i)}$. The constraints on variables $p_{(i,j)}$ are thus very similar to the previous constraints on variables $y_{(i,j)}$:

$$\sum_j p_{(1,j)} = 1$$

$$\sum_j p_{(i,j)} = p_{prev_D(i)} \quad \forall i \in \{2, \ldots, |\mathcal{N}_D|\}$$

where $p_{(i,j)} \in [0, 1]$. All the other constraints are identical to the ones in pure strategies (with $y_{(i,j)}$ replaced by $p_{(i,j)}$). This proves the polynomial solvability of the determination of an RDU-optimal mixed strategy in the case where function φ is concave piecewise linear (since the size of the linear program is linear in the size of the decision tree).

4.3.2. Implicit enumeration algorithm

We now present a branch and bound method for determining an RDU-optimal (pure) strategy. Unlike the previous approach, this method has the advantage of remaining valid for any probability transformation function φ. The branching principle is to partition the set of strategies in several subsets according to the choice of a given edge (N, N') at a decision node N. More formally, the nodes of the enumeration tree are characterized by a *partial strategy*, that defines a subset of strategies. Consider a decision tree $\mathcal{T}$ and a set of nodes $\mathcal{N}^\Gamma$ including:

- the root N_r of $\mathcal{T}$,
- one and only one successor for every decision node $N \in \mathcal{N}_D^\Gamma = \mathcal{N}_D \cap \mathcal{N}^\Gamma$.

The set of edges $\Gamma = \{(N, N')\colon\ N \in \mathcal{N}_D^\Gamma, N' \in \mathcal{N}^\Gamma\} \subseteq \mathcal{E}$ defines a *partial strategy* of $\mathcal{T}$ if the subgraph induced by $\mathcal{N}^\Gamma$ is a tree. A strategy Δ is said *compatible* with a partial strategy Γ if $\Gamma \subseteq \Delta$. The subset of strategies characterized by a partial strategy corresponds to the set of compatible strategies. At each iteration of the search, one chooses an edge among the ones starting from a given decision node. The order in which the decision nodes are considered is given by a priority function $rk : \mathcal{N}_D \to \{1, 2, \ldots, |\mathcal{N}_D|\}$: if several decision nodes are candidates to enter $\mathcal{N}^\Gamma$, the one with the lowest priority rank will be considered first. The ranking function rk is defined by:

$$\begin{cases} rk(N_r) = 1 \\ |past(N)| > |past(N')| \quad \Rightarrow \quad rk(N) > rk(N') \\ |past(N)| = |past(N')| \text{ and } EU(\mathcal{T}(N)) > EU(\mathcal{T}(N')) \quad \Rightarrow \quad rk(N) < rk(N') \end{cases}$$

where $EU(\mathcal{T}(N))$ is the optimal value of EU in $\mathcal{T}(N)$ (we recall that $\mathcal{T}(N)$ is the subtree rooted in N).

Example 9. For the decision tree in Fig. 3, there is a unique ranking function rk defined by: $rk(D_1) = 1$, $rk(D_2) = 5$, $rk(D_3) = 3$, $rk(D_4) = 4$ and $rk(D_5) = 2$ (since $EU(\mathcal{T}(D_5)) < EU(\mathcal{T}(D_3)) < EU(\mathcal{T}(D_4)) < EU(\mathcal{T}(D_2))$).

Algorithm 1 describes formally the implicit enumeration procedure that we propose:

Algorithm 1: BB(Γ, RDU_{opt})

```
N_1 ← {N_1 ∈ N_D: N_1 is candidate};
N_min ← arg min_{N∈N_1} rk(N);
E_min ← {(N_min, A) ∈ E: A ∈ S(N_min)};
for each (N, A) ∈ E_min do
  if ev(Γ ∪ {(N, A)}) > RDU_opt then
    RDU_temp ← BB(Γ ∪ {(N, A)}, RDU_opt);
    If RDU_temp > RDU_opt then
      RDU_opt ← RDU_temp;
    end
  end
end
return RDU_opt
```

It takes as an argument a partial strategy Γ and the best RDU value found so far, denoted by RDU_{opt}. The search is depth-first. The decision nodes that are candidates to enter $\mathcal{N}^\Gamma$ are denoted by $\mathcal{N}_1$. Among them, the node with the lowest

priority rank is denoted by N_{min}. The set of its incident edges is denoted by $\mathcal{E}_{min}$. It defines the set of possible extensions of Γ considered in the search (in other words, the children of the node associated to Γ in the enumeration tree). For every partial strategy Γ (in other words, at every node of the enumeration tree), one has an evaluation function ev that gives an upper bound of the RDU value of any strategy compatible with Γ. The optimality of the returned value RDU_{opt} is guaranteed since only suboptimal strategies are pruned during the search as soon as ev is an upper bound.

We give below the main features of our algorithm.

Initialization. A branch and bound procedure is notoriously more efficient when a good solution is known before starting the search. In our method, the lower bound (RDU_{opt}) is initially set to the RDU value of the EU-optimal strategy.

Computing the lower bound. At each node of the search, one computes the EU-optimal strategy among strategies that are compatible with Γ. When its RDU value is greater than the best found so far, we update RDU_{opt}. This makes it possible to prune the search more quickly.

Computing the upper bound. The evaluation function is denoted by ev. It returns an upper bound on the RDU value of any strategy compatible with Γ. The principle of this evaluation is to determine a lottery that stochastically dominates any lottery corresponding to a strategy compatible with Γ, and then to evaluate this ideal lottery according to RDU. This yields an upper bound since RDU is compatible with stochastic dominance, i.e. if L stochastically dominates L' then $RDU(L) \geqslant RDU(L')$. In order to compute such a lottery, one proceeds by dynamic programming in the decision tree. Actually, one can indifferently manipulate decumulative functions or lotteries, since both sets are in bijection (we recall that a lottery on utilities is considered as a function from U to $[0,1]$ in this paper). For the sake of clarity, we describe the recursion by referring to decumulative functions. The initialization is performed as follows: at each terminal node $T \in \mathcal{N}_U$ is assigned a decumulative function $G_{L^T} = (1, u(T))$. Next, at each node $N \in \mathcal{N}$, one computes the decumulative function of a lottery that stochastically dominates all the lotteries of subtree $\mathcal{T}(N)$. More precisely, at a chance node A, one computes the decumulative function G_{L^A} induced by the decumulative functions of its children as follows:

$$\forall u, \quad G_{L^A}(u) = \sum_{N \in S(A)} p\big((A,N)\big) G_{L^N}(u)$$

where G_{L^N} corresponds to the decumulative function assigned to node $N \in \mathcal{N}$. Besides, at each decision node D, we apply the following recurrence relation on the decumulative functions:

$$\begin{cases} \forall u, \quad G_{L^D}(u) = G_{L^N}(u) \quad \text{if } \exists N \in S(D)\text{: } (D,N) \in \Gamma \\ \forall u, \quad G_{L^D}(u) = \max_{N \in S(D)} G_{L^N}(u) \quad \text{otherwise} \end{cases}$$

Finally, the value returned by ev is $RDU(L^{N_r})$ where L^{N_r} corresponds to the lottery of decumulative function $G_{L^{N_r}}$. The complexity of this recursive procedure for $\Gamma = \emptyset$ is in $O(|\mathcal{N}|.|U|)$ (where $|U|$ is the number of distinct utilities at the leaves) since each node in $\mathcal{N}$ is examined once, and the support set of a lottery is upper bounded by $|U|$. However, during the branch and bound procedure, when an edge (D_i, N_j) is inserted into Γ, it is not necessary to recompute G_{L^N} in every node N. One can indeed use functions G_{L^N} already computed for evaluating $ev(\Gamma)$: it is sufficient to update functions G_{L^N} only on nodes N that belong to the path from N_r to D_i. Since the length of a path in $\mathcal{T}$ is upper bounded by height h, the complexity of computing $ev(\Gamma)$ for $\Gamma \neq \emptyset$ is therefore in $O(h.|U|)$. To prove the validity of the recursive procedure, one proceeds by induction:

- *At a chance node A*: consider a tuple $(L_N)_{N \in S(A)}$ of lotteries such that L^N stochastically dominates L_N for all $N \in S(A)$. We have: $G_{L^A}(u) = G_{\sum_{N \in S(A)} p((A,N)) L^N}(u) = \sum_{N \in S(A)} p((A,N)) G_{L^N}(u) \geqslant \sum_{N \in S(A)} p((A,N)) G_{L_N}(u) \ \forall u$ where $\sum_{N \in S(A)} p((A,N)) L^N$ denotes a compound lottery. This proves that L^A stochastically dominates any lottery corresponding to a strategy in $\mathcal{T}(A)$.
- *At a decision node D*: if there exists $N \in S(D)$ with $(D,N) \in \Gamma$, then the validity is obvious. In the other case, consider again a tuple $(L_N)_{N \in S(D)}$ of lotteries such that L^N stochastically dominates L_N for all $N \in S(D)$. By definition, $G_{L^D}(u) = \max_{N \in S(D)} G_{L^N}(u) \ \forall u$. Thus $G_{L^D}(u) \geqslant G_{L^N}(u) \geqslant G_{L_N}(u) \ \forall u$ for any lottery L_N of the tuple. This proves that L^D stochastically dominates any lottery corresponding to a strategy in $\mathcal{T}(D)$.

Consequently, lottery L^{N_r} stochastically dominates all lotteries corresponding to a strategy compatible with Γ.

Example 10. Let us come back to the decision tree of Fig. 1. Assume that $\Gamma = \{(D_1, A_1)\}$. The decumulative functions assigned to nodes A_3 and A_4 are $G_{L^{A_3}} = (1, 3; \frac{1}{2}, 10)$ and $G_{L^{A_4}} = (1, 1; \frac{1}{2}, 11)$. They are represented in the left part of Fig. 5. The decumulative function $G_{L^{D_2}}$ computed by dynamic programming is then the upper envelope of $G_{L^{A_3}}$ and $G_{L^{A_4}}$. More formally, the decumulative function computed in D_2 is defined by: $\forall x, G_{L^{D_2}}(x) = \max\{G_{L^{A_3}}(x), G_{L^{A_4}}(x)\}$. This decumulative function is represented in bold in the right part of Fig. 5. Then we have: $G_{L^{D_2}} = (1, 3; \frac{1}{2}, 11)$. Similarly, one computes:

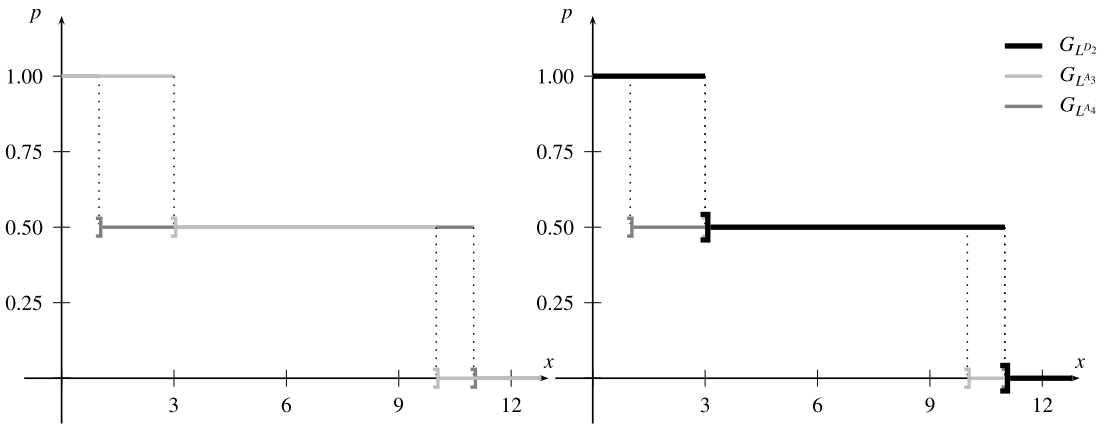


Fig. 5. Computation of a stochastically dominating lottery.

Table 3
Execution times for a concave piecewise linear function φ (in seconds).

Approach	Height (nodes)				
	4 (11)	6 (127)	8 (511)	10 (2047)	12 (8191)
MIP	< 1	< 1	< 1	$2.1 - 2.3 - 2.8$	$79.4 - 86.0 - 107.3$
Implicit enumeration	< 1	< 1	< 1	$< 0.1 - 2.5 - 5.0$	$0.3 - 5.6 - 20.2$

- $G_{L^{A_1}} = 0.5G_{L^{D_2}} + 0.5G_{L^{u(2)}} = (1, 2; \frac{1}{2}, 3; \frac{1}{4}, 11)$ where $G_{L^{u(2)}}$ is the decumulative function associated to the node of utility $u(2)$,
- $G_{L^{D_1}} = G_{L^{A_1}} = (1, 2; \frac{1}{2}, 3; \frac{1}{4}, 11)$ since $\Gamma = \{(D_1, A_1)\}$.

Decumulative function $G_{L^{D_1}}$ corresponds to lottery $(\frac{1}{2}, 2; \frac{1}{4}, 3; \frac{1}{4}, 11)$. The upper bound for $\Gamma = \{(D_1, A_1)\}$ is therefore $ev(\Gamma) = RDU((\frac{1}{2}, 2; \frac{1}{4}, 3; \frac{1}{4}, 11))$.

4.4. Numerical tests

Algorithms were implemented in C++ and the computational experiments were carried out on a PC with a Pentium IV CPU 2.13 GHz processor and 3.5 GB of RAM.

Tests on random instances. We have first compared the performances of the MIP approach with the ones of the implicit enumeration approach. Our tests were performed on complete binary decision trees of even height. The height of these decision trees varies from 4 to 12, with an alternation of decision nodes and chance nodes. The utilities at the leaves are real numbers randomly drawn within interval $[1, 1000]$, and the conditional probabilities at the chance nodes are randomly drawn positive real numbers summing up to 1. Since the MIP approach requires a concave piecewise linear function φ, we used function φ defined by $\varphi(p) = \min\{f_1(p), \ldots, f_5(p)\}$ with: $f_1(p) = 4p$, $f_2(p) = 2p + 0.2$, $f_3(p) = p + 0.5$, $f_4(p) = \frac{1}{2}p + 0.7$, $f_5(p) = \frac{1}{4}p + 0.85$. Table 3 shows the average execution CPU times obtained by both approaches. The mixed integer linear programs are solved using the ILOG CPLEX v11.1.0 solver. Note that the solution times indicated in Table 3 for the MIP approach do not take into account the preprocessing time. When it is informative, the min and max values are indicated under the following format: *min – average – max*. Both approaches instantly give an optimal strategy for trees of height less than 8. However, when height exceeds 12, the implicit enumeration approach is much more efficient than the MIP approach.

Next, we have gone further into the study of our implicit enumeration algorithm. For this purpose, we have measured the performances of the implicit enumeration approach with function φ defined by: $\varphi(p) = p^\gamma/(p^\gamma + (1-p)^\gamma)$. This function is not concave nor piecewise linear. This is the one usually proposed to model sophisticated behaviors that the EU model is unable to describe [24]. Parameter γ takes value in interval $[0, 1]$. For $\gamma = 1$, we have $\varphi(p) = p$ and RDU reduces to EU. We have tested our algorithm for several values of parameter γ, namely $\gamma = 0.2$, $\gamma = 0.5$ and $\gamma = 0.8$. Table 4 presents the performances of the algorithm with respect to parameter γ and the height of the decision tree. When it is informative, the min and max values are indicated. For each value of γ and height, we give the average performance computed over 50 decision trees. Unsurprisingly, the performances improve when γ is near 1 (i.e. RDU is close to EU). Note that, for bigger instances (i.e. the height of which is greater than 14), some hard instances begin to appear for which the solution time becomes high.

Table 4
Execution time.

	Depth (nodes)					
	5 (11)	7 (127)	9 (511)	11 (2047)	13 (8191)	15 (32767)
$\gamma = 0.2$	< 0.1	< 0.1	< 0.1	$< 0.1 - 0.4 - 1.1$	$0.3 - 4.3 - 13.8$	$3.0 - 62.1 - 512.9$
$\gamma = 0.5$	< 0.1	< 0.1	< 0.1	$< 0.1 - 0.1 - 0.5$	$0.1 - 2.7 - 7.4$	$1.2 - 17.3 - 99.4$
$\gamma = 0.8$	< 0.1	< 0.1	< 0.1	$< 0.1 - 0.1 - 0.5$	$0.1 - 1.6 - 8.0$	$0.6 - 10.9 - 101.4$

Table 5
Quality of upper and lower bound.

Bound	Height (nodes)				
	4 (11)	6 (127)	8 (511)	10 (2047)	12 (8191)
$\gamma = 0.2$					
$RDU(L_{EU^*})/RDU*$	95.1%	92.46%	90.0%	91.2%	90.9%
$RDU(L_{SD})/RDU*$	104.4%	105.1%	106.9%	112.4%	117.5%
$\gamma = 0.5$					
$RDU(L_{EU^*})/RDU*$	99.6%	99.3%	99.0%	98.8%	98.2%
$RDU(L_{SD})/RDU*$	105.5%	109.6%	109.4%	109.1%	110.2%
$\gamma = 0.8$					
$RDU(L_{EU^*})/RDU*$	99.8%	99.5%	99.4%	98.9%	98.6%
$RDU(L_{SD})/RDU*$	106.4%	106.3%	108.8%	107.8%	107.2%

Finally, in order to evaluate the quality of the lower bound and the upper bound, we have investigated ratios $RDU(L_{EU})/RDU^*$ and $RDU(L_{SD})/RDU^*$ with respect to parameter γ and the height of the decision tree, where L_{EU} is the lottery corresponding to an optimal strategy for EU, L_{SD} is the lottery computed for evaluating the upper bound and RDU^* is the value of an optimal strategy for RDU. The results are presented in Table 5, where each value is an average over 50 instances. One can observes that $RDU(L_{EU})$ provides a good lower bound that naturally deteriorates when γ becomes close to 0 (i.e. the probabilities are very distorted). The upper bound appears to be within 10% of the optimal value on most instances.

However, the complete binary trees considered here are actually the "worst cases" that can be encountered. In fact, in many applications, the decision trees are much less balanced and therefore, for the same number of decision nodes, an RDU-optimal strategy will be computed faster, as illustrated now on a TV game example.

Application to *Who wants to be a millionaire?* *Who wants to be a millionaire?* is a popular game show, where a contestant must answer a sequence of multiple-choice questions (four possible answers) of increasing difficulty, numbered from 1 to 15. This is a double or nothing game: if the answer given to question k is wrong, then the contestant quits with no money. However, at each question k, the contestant can decide to stop instead of answering: he then quits the game with the monetary value of question $(k-1)$. Following Perea and Puerto [37], we study the Spanish version of the game in 2003, where the monetary values of the questions were 150, 300, 450, 900, 1800, 2100, 2700, 3600, 4500, 9000, 18 000, 36 000, 72 000, 144 000 and 300 000 Euros respectively. Note that, actually, after the 5th and 10th questions, the money is banked and cannot be lost even if the contestant gives an incorrect response to a subsequent question: for example, if the contestant gives a wrong answer to question 7, he quits the game with 1800 Euros. Finally, the contestant has three *lifelines* that can be used once during the game: Phone a friend (call a friend to ask the answer), 50:50 (two of the three incorrect answers are removed), Ask the audience (the audience votes and the percentage of votes each answer has received is shown).

We applied our algorithm to compute an RDU-optimal strategy for this game. For this purpose, we first used the model proposed by [37] to build a decision tree representing the game. In this model, a strategy is completely characterized by giving the question numbers where the different lifelines are used, and the question number where the contestant quits the game. We have carried out experimentations for various probability transformation functions, modeling different attitudes towards risk. The identity (resp. square, square root) function corresponds to an expected reward maximizer (resp. a risk averse, risk seeker decision maker). The results are reported in Table 6. For each function φ, we give the expected reward (column Exp.) of the optimal strategy, as well as the maximum possible reward (column Max.) and the probability to win at least 2700 Euros (column $G_L(2.7K)$). Note that, in all cases, the response time of our procedure is less than one second while there are 14 400 decision nodes and the height is 30. This good behavior of the algorithm is linked to the shape of the decision tree, that strongly impacts on the number of potential strategies.

A limitation of the model introduced by Perea and Puerto [37] is that the choice to use a lifeline is not dependent on whether the contestant knows the answer or not. For this reason, we introduced the following refinement of the model: if the contestant knows the answers to question k, he directly gives the correct answer, else he has to make a decision. A small part of the decision tree for this new modeling is represented in Fig. 6 (the dotted lines represent omitted parts of

Table 6
Optimal strategies for various φ functions.

$\varphi(p)$	50:50	Phone	Ask	Quit	Exp.	Max.	$G_L(2.7K)$
p	9	10	12	13	2387	36K	0.10
p^2	4	5	5	8	1536	2.7K	0.35
$\sqrt{p}$	14	15	13	X	1987	300K	0.06

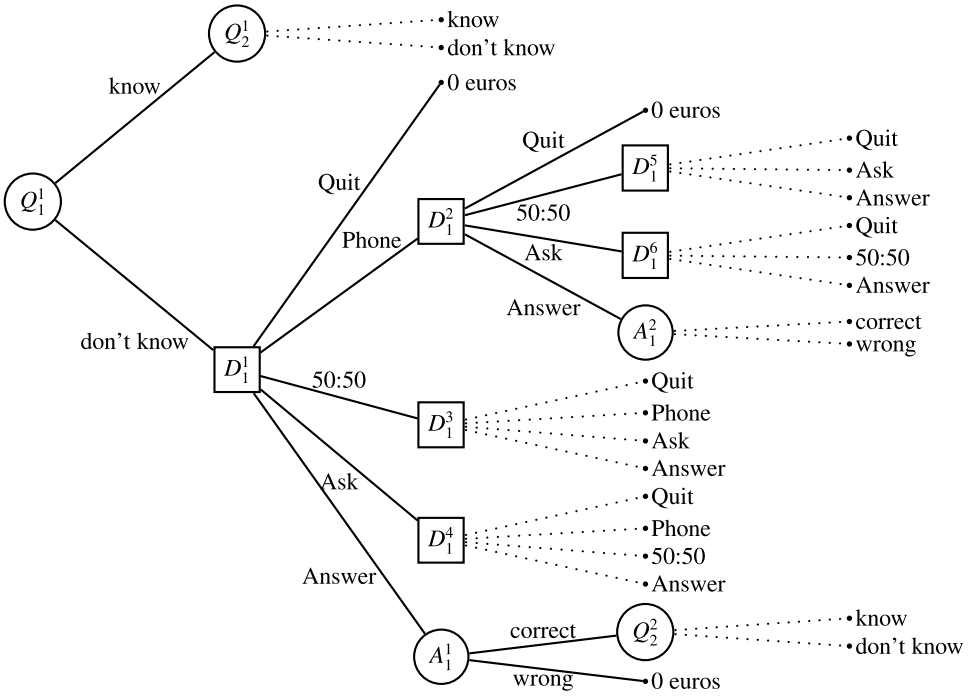


Fig. 6. Refined decision tree for the TV game.

the tree). Chance nodes Q_1^i's (resp. Q_2^i's) represent question 1 (resp. 2), with two possible events (know the answer or not). Decision nodes D_1^i's represent the decision to use an available lifeline, answer or quit facing question 1 (in fact, this latter opportunity becomes realistic only from question 2). Finally, the answer is represented by a chance node (A_1^i's) where the probabilities of the events (correct or wrong answer) depend on the used lifelines. We used the data provided by Perea and Puerto [37] to evaluate the different probabilities at the chance nodes. The whole decision tree has more than 75 millions of nodes. The problem becomes therefore much harder since the number of potential strategies explodes. Unlike previous numerical tests, we had to use a computer with 64 GB of RAM so as to be able to store the instance. Despite the high size of the instance, the procedure is able to return an optimal strategy in 2992 sec. for $\varphi(p) = p^2$ (risk averse behavior) and 4026 sec. for $\varphi(p) = p^{(2/3)}$ (risk seeker behavior). Note that, for risk seeker behaviors, the solution time increases with the concavity of the probability transformation function.

5. Computing RDU in an influence diagram

The previous approaches face the main inconvenience of decision trees: their size grows quickly when the number of decision stages increases. For this reason, we study the optimization of RDU in influences diagrams, that provide compact representations of sequential decision problems.

5.1. Influence diagram formalism

An *influence diagram* [18] is a graphical model for a sequential decision problem. Unlike decision trees, the emphasis is put on the decomposability of the underlying probability structure. By taking advantage of independences between involved random variables and utility variables, one gets a much more compact representation than the one obtained by expliciting all possible scenarios, as would be done in a decision tree. An influence diagram including random variables $A_1, \ldots, A_p$ and decision variables $D_1, \ldots, D_n$ is an acyclic digraph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ such that:

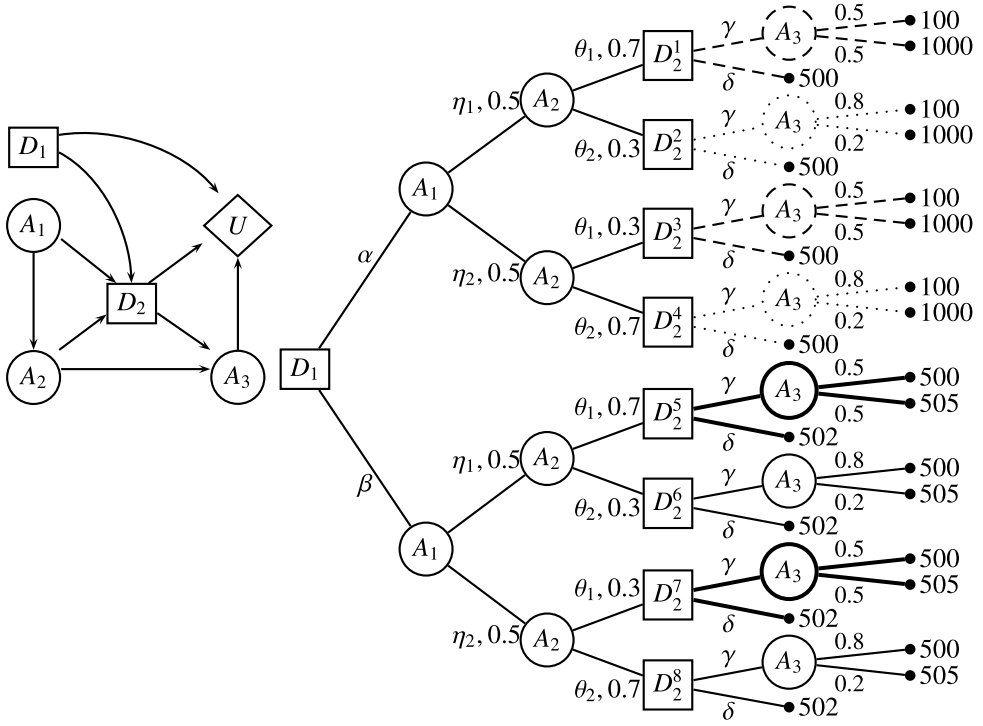


Fig. 7. Influence diagram and corresponding decision tree.

- set $\mathcal{N}$ is partitioned into three subsets: a set $\mathcal{N}_D = \{D_1, \ldots, D_n\}$ of decision variables (represented by squares), a set $\mathcal{N}_A = \{A_1, \ldots, A_p\}$ of random variables (represented by circles), and a set $\mathcal{N}_U = \{U_1, \ldots, U_m\}$ of utility nodes (represented by diamonds);
- set $\mathcal{E}$ of directed edges is partitioned into three subsets: a set of functional edges from a decision variable or a random variable to a random variable or a utility node (edges representing dependences), a set of informational edges from a decision variable or a random variable to a decision variable (edges representing the observed variables before making a decision);
- the nodes representing random variables are endowed with a conditional probability table, indicating the probability of every event conditionally to the parent nodes;
- the utility nodes are endowed with a table indicating the utility conditionally to the parent nodes.

The following structural condition on the graph must also hold: there exists a path (including functional and informational edges) connecting all the nodes representing decision variables.

An influence diagram is represented in the left part of Fig. 7, where the conditional probability tables and the utility tables have been deliberately omitted for the sake of brevity. The two possible decisions in D_1 (resp. D_2) are denoted by α and β (resp. γ and δ). The two modalities of random variable A_1 (resp. A_2) are η_1 and η_2 (resp. θ_1 and θ_2). The order in which the decisions and the observations are made is assumed to be $D_1 - A_1 - A_2 - D_2 - A_3 - U$. We adopt here the convention that the temporal order of the decisions are read from left to right. By "unfolding" the diagram, one obtains the decision tree represented in the right part of Fig. 7 (note that the probabilities and the utilities indicated in the tree comply with the conditionings imposed by the diagram). Since both A_3 and U are independent from A_1 conditionally to D_1, several subtrees are identical (see Fig. 7). Note that the presence of identical subtrees leads to repeat several times the same calculations when determining a strategy maximizing EU in the decision tree. Influence diagrams make it possible to avoid this pitfall, as detailed in the following section.

5.2. RDU, influence diagram and consequentialism

The purpose of our work is to "solve" an influence diagram when the preferences of the decision maker do not follow the EU model but the RDU model. The aim of solving the diagram is to determine the best strategy according to a decision criterion (EU, RDU or others). In order to define a strategy, it is necessary to know the random variables already observed when making each decision, as well as the temporal order in which the decisions are made. A strategy consists then in setting a value to every decision variable conditionally to its past. In a decision tree, the past of a decision variable is simply defined as the set of random variables and decision variables lying on the path from the root to that variable. The decision

tree in the right part of Fig. 7 includes 32 feasible strategies, among which strategy $\{D_1 = \alpha, D_2^1 = \gamma, D_2^2 = \delta, D_2^3 = \delta, D_2^4 = \delta\}$ (note that nodes D_2^5, D_2^6, D_2^7 and D_2^8 cannot be reached when $D_1 = \alpha$). In an influence diagram, the temporal order is less apparent. For this reason, set $\mathcal{N}_A$ is partitioned into disjoint sets $I_0, I_1, \ldots, I_n$. Set I_0 includes the random variables observed before first decision D_1 is made (corresponding to the parents of D_1), I_k the random variables observed between D_k and D_{k+1} (corresponding to the parents of D_{k+1}), and finally I_n the remaining random variables, i.e. the ones that are never observed or are observed after the last decision D_n is made (the variables that are not parent of any decision variable). This induces a partial order $\prec$ on $\mathcal{N}_D \cup \mathcal{N}_A$: $I_0 \prec D_1 \prec I_1 \prec \cdots \prec D_n \prec I_n$. For instance, for the diagram in the left part of Fig. 7, the partial order is $D_1 \prec \{A_1, A_2\} \prec D_2 \prec \{A_3\}$. The past of a decision variable D_k is then the set of variables X such that $X \prec D_k$. Formally, a strategy in an influence diagram is a set of *decision rules* for variables D_k, where a decision rule for D_k maps each instantiation of the variables in the past of D_k with a value in the domain of D_k. However, in practice, only *consequentialist* strategies (i.e., where every decision made only depends on the variables influencing parameters of the future) are considered in influence diagrams. Therefore, it is not necessary to know the values of all variables in the past to set the value of a decision variable. This property is crucial since the description of a consequentialist strategy remains linear in the size of the diagram, which is not the case of non-consequentialist strategies. For instance, in the diagram of Fig. 7, only variables D_1 and A_2 (and not A_1) have an influence on the future of D_2. Therefore, in a consequentialist strategy, the decision made in D_2 only depends on D_1 and A_2 (and not A_1). A decision rule in D_2 is for instance $\{(D_2 = \gamma | D_1 = \alpha, A_2 = \theta_1), (D_2 = \delta | D_1 = \alpha, A_2 = \theta_2), (D_2 = \delta | D_1 = \beta, A_2 = \theta_1), (D_2 = \gamma | D_1 = \beta, A_2 = \theta_2)\}$. It means: decision γ is made in D_2 if decision α was made in D_1 and event θ_1 occurred in A_2, decision δ is made in D_2 if decision α was made in D_1 and event θ_2 occurred in A_2, etc. As a result, the set of strategies in the diagram of Fig. 7 includes 16 strategies. It is important to note that the set of strategies considered in the influence diagram is only a subset of the strategies considered in the corresponding decision tree (for instance the above-mentioned strategy for the decision tree is not in this subset). When optimizing EU, this does no harm since it is well known that there always exists an EU-optimal strategy included in this subset (since there always exists an EU-optimal strategy that is consequentialist). *A contrario*, a strategy optimizing RDU is not necessarily included in this subset (since there does not always exist an RDU-optimal strategy that is consequentialist), as illustrated in the following example.

Example 11. Consider the decision tree in the right part of Fig. 7. A strategy maximizing EU consists in making decision α in D_1, decision γ in D_2^1 and D_2^3, and decision δ in D_2^2 and D_2^4. This strategy is consequentialist. Now, assume that one adopts the following probability transformation function φ: $\varphi(p) = 0$, if $0 \leqslant p < 0.175$, $\varphi(p) = 0.09$, if $0.175 \leqslant p < 0.25$, $\varphi(p) = 0.1$ if $0.25 \leqslant p < 0.35$, $\varphi(p) = 0.15$ if $0.35 \leqslant p < 0.75$, $\varphi(p) = 0.5$, if $0.75 \leqslant p < 0.825$, $\varphi(p) = 0.9$ if $0.825 \leqslant p < 1$ and $\varphi(1) = 1$. Then, the unique strategy maximizing RDU consists in making decision α in D_1, decision γ in D_2^1 and decision δ in D_2^2, D_2^3 and D_2^4. This strategy is not consequentialist since two distinct decisions are made in D_2^1 and D_2^3.

5.3. Algorithms

Note that determining an EU-optimal strategy in an influence diagram is proved NP-hard. Determining an RDU-optimal strategy is even harder since the description of a non-consequentialist strategy may require a memory space whose size is exponential in the size of the diagram.

5.3.1. Two-phases method

Since an RDU-optimal strategy is not necessarily consequentialist, a first method that seems natural consists of two phases: (1) "unfolding" the influence diagram in a decision tree, then (2) determining an optimal strategy according to RDU directly in the decision tree. For phase 2, one can use the implicit enumeration approach proposed in Section 4.3.2. However, although it enables the determination of a real RDU-optimal strategy, this method is of course costly in memory space, and becomes impracticable when the size of the decision tree is prohibitive.

5.3.2. Δ-relaxation method

For this reason, we propose another method that takes advantage of the compact structure of influence diagrams (without unfolding them in decision trees), at the cost of a reduction of the set of considered strategies. For this purpose, a first idea would be to explore only the space of consequentialist strategies, but one would then lose an important part of the descriptive power of the RDU model. Consequently, the approach we propose here does not renounce consequentialism: we introduce a relaxed form of consequentialism in order to realize a compromise between descriptive power and compactness of representation. By inserting additional functional edges in the diagram, one enlarges the space of considered strategies. Note that creating fictitious dependences between independent variables does not change the problem itself but only its representation. In Example 12 below, we illustrate the changes induced by the insertion of a new functional edge.

Example 12. After inserting edge (A_1, U) in the influence diagram of Fig. 7, one obtains the diagram represented in Fig. 8. From the point of view of compactness of the representation, it should be noted that, if variable A_1 has two modalities, it will double the number of lines in the table assigned to variable U: the old table (before insertion of the edge) represented indeed utility $U(D_1, D_2, A_3)$, while the new table (after insertion of the edge) represents $U(D_1, A_1, D_2, A_3)$. However, the

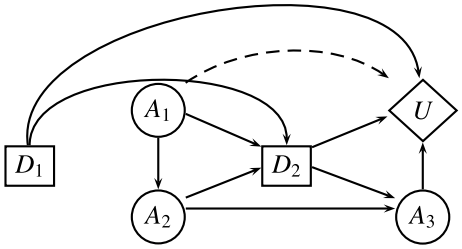


Fig. 8. Modified influence diagram.

benefit is that the space of considered strategies is enlarged: since A_1 now influences the parameters in the future of D_2, the decision made in D_2 is conditioned by the value of A_1. Consequently, a decision rule in D_2 now takes into account variables D_1, A_1 and A_2. In other words, in this case, the insertion of edge (A_1, U) makes it possible to take into account all the non-consequentialist strategies.

In the sequel, we name Δ-relaxation of consequentialism the fact of adding Δ dependences on every decision variable. At worst, one adds $|\mathcal{N}_D| \times \Delta$ supplementary edges. Adding a dependence on a decision variable D amounts to inserting an edge[1] from V to V' with $V \prec D \prec V'$, provided D does not already depend on V. This latter condition can be graphically formalized as follows: $(V, W) \notin \mathcal{E}\ \forall W \notin \mathcal{N}_D \wedge W \succ D$. The inserted edges are of course not chosen arbitrarily. We now detail the procedure to select them. Our aim is primarily to keep the representation as much compact as possible. In this purpose, our heuristic to select edges to insert consists in choosing greedily the ones that have the least impact on the sizes of the tables in the diagram. Note first that, given two variables V and V', inserting edge (V, V') increases the size of the table in V', i.e. the number of entries. More precisely, the size of the table in V' is equal to $|V'|.\prod_{W \in \text{Pred}(V')} |W|$, where $|V'|$ is the number of modalities of variable V' and $\text{Pred}(V')$ is the set of predecessors of V' in the influence diagram. Inserting edge (V, V') multiplies therefore the size of the table in V' by $|V|$. In other words, for a table in V' of size $s(V')$ before insertion, the size becomes $s(V').|V|$ after insertion, thus yielding an increment $i(V, V') = s(V').(|V| - 1)$. The aim of our heuristic in the greedy procedure to insert edges is to minimize this increment at each step. To this end, each time an edge has to be selected among a set of candidate edges, we choose an edge (V, V') such that $i(V, V')$ is minimum. Algorithm 2 summarizes the whole procedure, where **update table** is a primitive that makes the table in V' depends on V (by duplicating the entries).

Algorithm 2: AddEdges

foreach $D \in \mathcal{N}_D$ **do**
  $i \leftarrow 0$;
  while $i < \Delta$ and $[\exists V \prec D\colon\ (V, V') \notin \mathcal{E}\ \forall V' \notin \mathcal{N}_D \wedge V' \succ D]$ **do**
    $V \leftarrow \arg\min_{W \prec D}\{|W|\colon\ (W, W') \notin \mathcal{E}\ \forall W' \notin \mathcal{N}_D \wedge V' \succ D\}$;
    $V' \leftarrow \arg\min_{W \notin \mathcal{N}_D}\{|W|.\prod_{W' \in \text{Pred}(W)} |W'|\colon\ W \succ D\}$;
    $\mathcal{E} \leftarrow \mathcal{E} \cup \{(V, V')\}$;
    update table in V';
    $i \leftarrow i + 1$;
  end
end

We now detail the procedure for determining a strategy maximizing RDU after a Δ-relaxation of the diagram. As for decision trees, we propose here a branch and bound procedure to solve the influence diagram. The main differences are in the way the strategies are enumerated (branching rule) and in the dynamic programming procedure used to compute an upper bound (more precisely, a lottery stochastically dominating all possible lotteries). We give below the main features of the implicit enumeration scheme.

Initialization. As for decision trees, one determines a strategy optimizing EU. Several solving algorithms have been proposed in the literature to accomplish this task [44,21]. Schachter's algorithm [44] consists in eliminating incrementally the nodes of the diagram while respecting the partial order on them. For this purpose, edge reversals (of edges representing probabilistic dependences) are performed, that are sometimes very costly in computation time. Jensen et al.'s algorithm [21] (inspired from inference algorithms used in *Bayesian networks*) consists in transforming the influence diagram in a *junction tree*, and then applying non-serial dynamic programming [5] in the junction tree. We have adopted this latter approach in our implementation, since its performances are generally assumed better than the ones of Schachter's algorithm.

[1] Note that the insertion of a single edge (V, V'), with $V \in \mathcal{N}_A \cup \mathcal{N}_D$ and $V' \in \mathcal{N}_A \cup \mathcal{N}_U$, is likely to create additional dependences on other variables D' such that $V \prec D' \prec V'$.

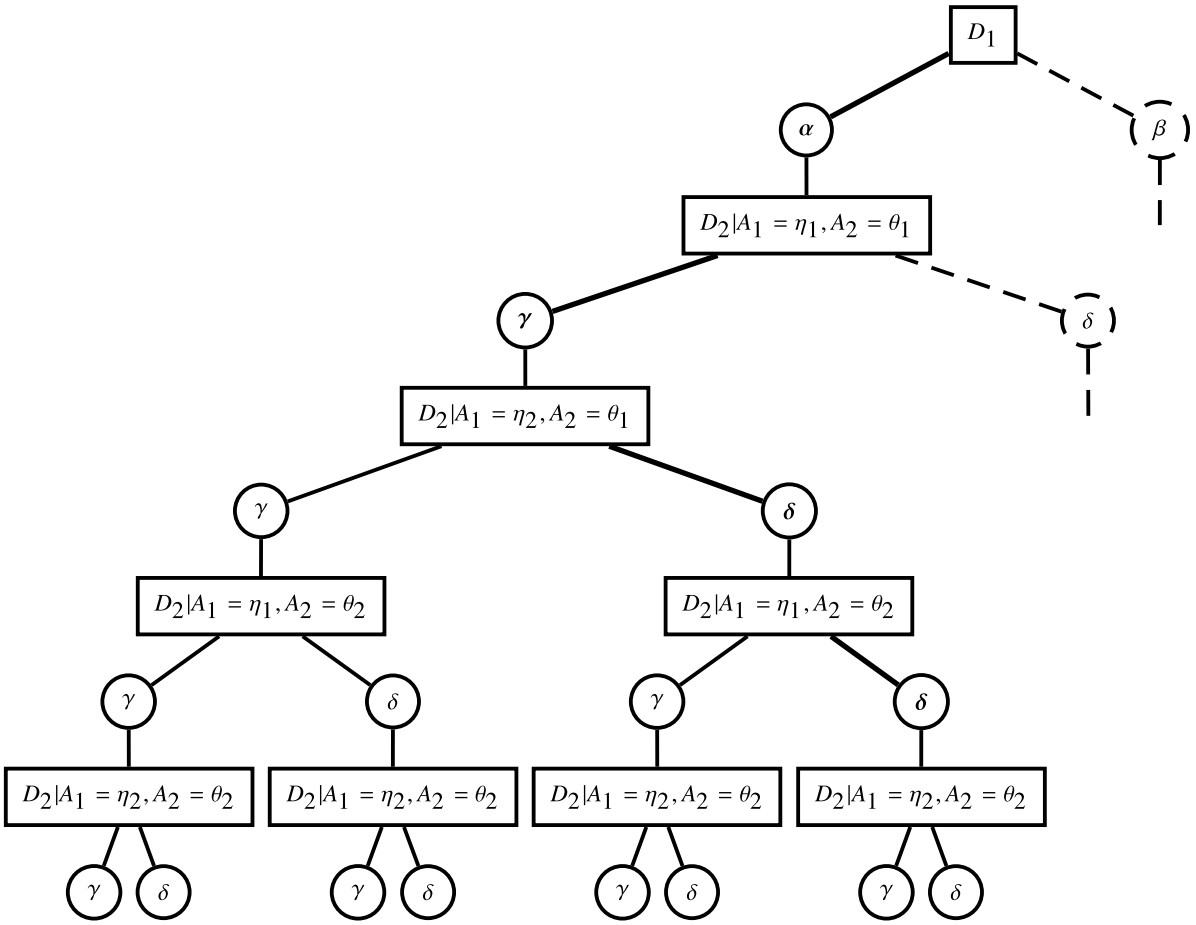


Fig. 9. Enumeration tree.

Branching rule. The branching rule we use is to set a value to a decision variable conditionally to the variables that influence it. For instance, for the diagram in Fig. 8, one separates the set of strategies characterized by: $(D_1 = \boldsymbol{\alpha}) \wedge (D_2 = \boldsymbol{\gamma} | D_1 = \alpha, A_1 = \eta_1, A_2 = \theta_1) \wedge (D_2 = \boldsymbol{\delta} | D_1 = \alpha, A_1 = \eta_2, A_2 = \theta_1) \wedge (D_2 = \boldsymbol{\delta} | D_1 = \alpha, A_1 = \eta_1, A_2 = \theta_2)$ into two subsets characterized respectively by:

- $(D_1 = \boldsymbol{\alpha}) \wedge (D_2 = \boldsymbol{\gamma} | D_1 = \alpha, A_1 = \eta_1, A_2 = \theta_1) \wedge (D_2 = \boldsymbol{\delta} | D_1 = \alpha, A_1 = \eta_2, A_2 = \theta_1) \wedge (D_2 = \boldsymbol{\delta} | D_1 = \alpha, A_1 = \eta_1, A_2 = \theta_2) \wedge (\boldsymbol{D_2 = \gamma | D_1 = \alpha}, \boldsymbol{A_1 = \eta_2}, \boldsymbol{A_2 = \theta_2})$,
- $(D_1 = \boldsymbol{\alpha}) \wedge (D_2 = \boldsymbol{\gamma} | D_1 = \alpha, A_1 = \eta_1, A_2 = \theta_1) \wedge (D_2 = \boldsymbol{\delta} | D_1 = \alpha, A_1 = \eta_2, A_2 = \theta_1) \wedge (D_2 = \boldsymbol{\delta} | D_1 = \alpha, A_1 = \eta_1, A_2 = \theta_2) \wedge (\boldsymbol{D_2 = \delta | D_1 = \alpha}, \boldsymbol{A_1 = \eta_2}, \boldsymbol{A_2 = \theta_2})$.

The enumeration tree is represented in Fig. 9. The instanciated nodes are indicated in bold. The order in which the variables are considered in the enumeration tree is compatible with the temporal order on the decision nodes (remember that there always exists a path linking all the decision nodes in the diagram).

Computing the lower bound. This is similar to what is done in decision trees. At each node of the enumeration tree, one computes a strategy optimizing EU in the subset of considered strategies, and one evaluates it according to RDU.

Computing the upper bound. As for decision trees, we need to compute a lottery stochastically dominating any lottery corresponding to a feasible strategy (so that its RDU value is an upper bound). For this purpose, we use a dynamic programming procedure. We recursively compute such a lottery for every decision variable and every possible instantiation of the variables in its past (compatible with decisions already set). In the following, for simplicity, we assume that there is only one single utility variable U, that takes value in $\{u_1, \ldots, u_m\}$ with $u_1 \leqslant \cdots \leqslant u_m$. We use a backward induction procedure. The initialization in U is performed by the following formula:

$$\forall i \geqslant 1, \quad P(U = u_i | I_0 \cdot\cdot I_n, D_1 \cdot\cdot D_n) = \begin{cases} 1 & \text{if } U(I_0 \cdot\cdot I_n, D_1 \cdot\cdot D_n) = u_i \\ 0 & \text{otherwise} \end{cases}$$

where $U(I_0 \cdot\cdot I_n, D_1 \cdot\cdot D_n)$ gives the value taken by variable U according to the values taken by variables $I_0, \ldots, I_n, D_1, \ldots, D_n$. Consider now a decision variable D_k. For a given realization of random variables $I_0, \ldots, I_{k-1}$ and a given choice of decisions in $D_1, \ldots, D_{k-1}$, the stochastically dominating lottery in D_k is computed by the following recurrence relation for all $i \geqslant 1$:

$$P(U \geqslant u_i | I_0 \cdot\cdot I_{k-1}, D_1 \cdot\cdot D_{k-1}) = \max_{D_k} \sum_{j=i}^{m} \sum_{I_k} P(I_k | I_0 \cdot\cdot I_{k-1}, D_1 \cdot\cdot D_k) \times P(U = u_j | I_0 \cdot\cdot I_{k-1}, I_k, D_1 \cdot\cdot D_k)$$

The stochastically dominating lottery is then easily determined by the following formula:

$$P(U = u_i | I_0 \cdot\cdot I_{k-1}, D_1 \cdot\cdot D_{k-1}) = P(U \geqslant u_i | I_0 \cdot\cdot I_{k-1}, D_1 \cdot\cdot D_{k-1}) - P(U \geqslant u_{i+1} | I_0 \cdot\cdot I_{k-1}, D_1 \cdot\cdot D_{k-1})$$

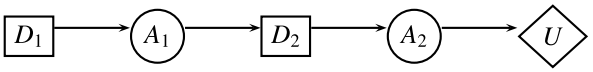
The value finally returned is computed by applying RDU to the lottery obtained in D_1. Note that the conditionings are in fact performed on much smaller sets of variables thanks to the independences displayed in the influence diagram. In order to optimize the calculations, we used a junction tree, like in Jensen's algorithm [21].

5.4. Numerical tests

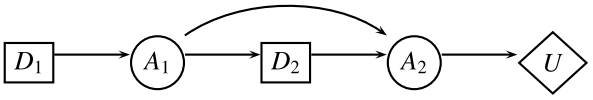
The two proposed algorithms (the two-phases method and the one operating directly in the influence diagram endowed with fictitious dependences) have been implemented in C++ and the computational experiments were carried out on a PC with a Pentium IV CPU 2.13 GHz processor and 3.5 GB of RAM (as for decision trees). We present below the results obtained on randomly generated influence diagrams. We first describe the mechanism of the random generator. We then analyze the performances of the algorithms, both in terms of computation time and solution quality. We used the same function φ as for decision trees: $\varphi(p) = p^\gamma/(p^\gamma + (1-p)^\gamma)$, with $\gamma = 0.2$ (value of γ for which the probabilities are the most distorted).

5.5. Random generator

In order to control the size of the generated influence diagrams, all nodes of the diagram must be taken into account when computing an optimal strategy. For this purpose, given a fixed number n of decision nodes, one first creates a path of length $2n+1$ alternating decision variables and random variables, with a utility variable at the leaf node. For instance, for two decision variables, it gives:



Then, to avoid that some variables in the diagram play no real role (i.e., have no impact on the choice of an optimal strategy), one imposes that every random variable influences another random variable (as long as there exists at least one random variable in its future). Coming back to the previous example, one can obtain:



To densify the diagram, edges are randomly inserted between some nodes. Concerning utilities and conditional probabilities, they were randomly generated. Finally, we used the greedy algorithm described in Section 5.3 to perform the Δ-relaxation of consequentialism.

5.6. Numerical results

To evaluate the computational gain when directly working in the influence diagram, we compared the execution times of the implicit enumeration algorithm in the two-phases method with the execution times of the algorithm performing a Δ-relaxation of consequentialism for various values of Δ. Table 7 indicates the execution times obtained (average time in seconds over 40 randomly generated instances for each entry). In each column, we vary the number of decision variables (i.e. we vary the value of n) and in each row is varied the value of Δ. The time taken by the two-phases method is indicated in the last line of the table. Symbol "–" appears when there exist instances for which the execution time is very important (beyond 30 min). Not surprisingly, the more n and Δ increase, the more the execution times increase. The two-phases method does not make it possible to solve instances with more than 7 decision nodes. In comparison, the Δ-relaxation method enables to solve instances with up to 10 decision nodes, at the cost of optimality.

In order to evaluate the quality of the solutions returned by the Δ-relaxation method, we compared their RDU values with the optimal ones in the corresponding decision trees (obtained by applying the two-phases method). Given an influence

Table 7
Execution time (sec).

Δ	*n*						
	4	5	6	7	8	9	10
0	0	0	0	0.21	0.85	2.78	9.11
1	0	0	0	0.23	0.62	3.94	12.15
2	0	0	0	0.38	1.10	4.79	23.36
3	0	0	0.10	0.72	2.33	9.25	29.97
4	0	0	0.56	1.07	4.46	18.61	93.52
5	0	0.08	1.91	3.43	19.84	87.40	–
2 ph.	0.17	0.93	5.02	17.38	–	–	–

Table 8
Influence of parameter Δ on strategy quality.

n	$\Delta = 0$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 5$
2	97%	100%			
3	93%	96%	98%	100%	
4	92%	93%	95%	97%	97%
5	93%	94%	95%	95%	97%
6	92%	93%	93%	95%	96%

diagram *ID*, we denote by $RDU^*(ID)$ (resp. $RDU_\Delta(ID)$) the RDU value of a strategy optimizing RDU in the corresponding decision tree (resp. optimizing RDU in *ID* after performing a Δ-relaxation of consequentialism). In order to evaluate the influence of parameter Δ on the strategy quality (w.r.t. RDU), we computed ratio $RDU_\Delta(ID)/RDU^*(ID)$ for different values of Δ and *n*. For each value of *n*, 200 instances were randomly generated. Then, on these instances, we performed a Δ-relaxation of consequentialism for Δ varying from 0 to 5. Table 8 shows the average ratio obtained (in percentage) over the 200 instances. The empty entries correspond to cases where all dependences are already present, and therefore the value is 100%. When $\Delta = 0$, one considers only consequentialist strategies. One can see that, for the instances considered here, the values of the returned strategies are significantly closer to the optimum for $\Delta = 5$ than for $\Delta = 0$ (thanks to the consideration of some non-consequentialist strategies).

6. Conclusion

In this article, we have proposed algorithms for optimizing RDU in sequential decision problems represented by decision trees or influence diagrams. The problem is NP-hard for both representations. For optimizing RDU in decision trees, we have provided a MIP formulation (adapted to risk seeker decision makers) and a branch and bound procedure (adapted to any attitude toward risk). The upper bound used in the branch and bound procedure is computed by dynamic programming, and is polynomial in the number of decision nodes. Note that this upper bound is actually customizable to any decision criterion compatible with stochastic dominance. The tests performed show that the dedicated branch and bound algorithm performs better than CPLEX applied to the MIP formulation. Our method makes it possible to solve efficiently random instances the number of decision nodes of which is near six thousands, and real-world instances the number of decision nodes of which is 75 millions (thanks to the particular shape of the decision tree).

However, when the number of stages grows, the decision tree formalism does not make it possible to compactly store the problem instance. Influence diagrams, that expresses in a concise way the problem instance by using independences between variables, are then very useful. When optimizing RDU in influence diagrams, we aim therefore at conciliating optimality and compactness issues (determining a strategy the RDU value of which is close to optimum, and the storage of which does not require too much memory space). We have shown that the space of strategies considered in an influence diagram is smaller than the space of strategies considered in the corresponding decision tree, since it is restricted to consequentialist strategies. For this reason, we have proposed an approach where the influence diagram is endowed with new edges representing fictitious dependences (Δ-relaxation of consequentialism). This enables to enrich the space of strategies with non-consequentialist strategies. Such strategies are indeed appreciated by RDU-maximizers. The numerical experiments carried out show the interest of Δ-relaxation since the RDU value of the computed strategy significantly improves the value obtained for a consequentialist strategy.

In order to enhance the approach proposed here, the most promising research direction is to identify crucial edges, i.e. edges whose insertion in the influence diagram would significantly increase the RDU value of the returned strategy. It would indeed limit the growth of the size of the diagram, and it would improve the quality of the returned strategy. In a different research direction, it is worth investigating a new compact graphical model able to handle non-consequentialist decision criteria, since the influence diagram representation seems to be not very suitable for it.

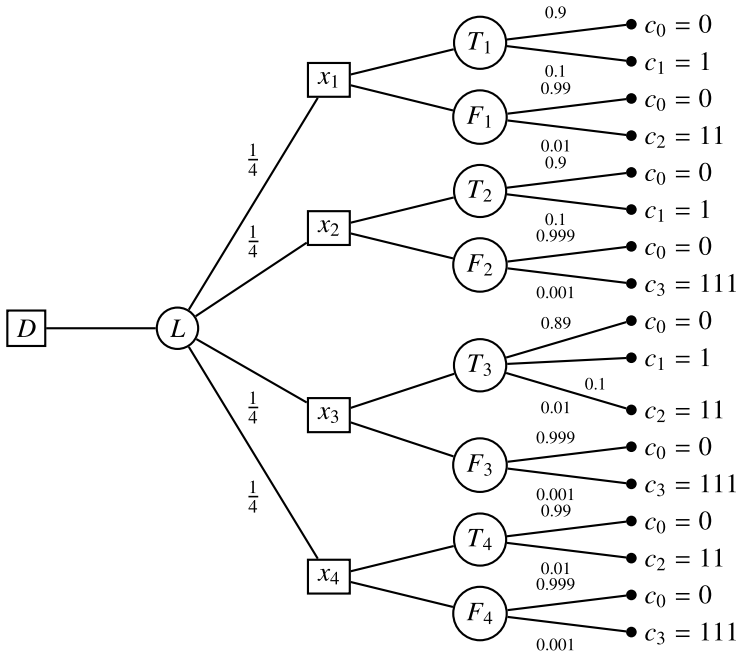


Fig. 10. An example of reduction.

Appendix A. Proofs

Proposition 1. *The determination of an RDU-optimal strategy* (*problem* RDU-OPT) *in a decision tree is an NP-hard problem.*

Proof. The proof relies on a polynomial reduction from problem 3-SAT, which can be stated as follows:

INSTANCE: a set X of boolean variables, a collection C of clauses on X such that $|c| = 3$ for every clause $c \in C$.
QUESTION: 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 $C = \{c_1, \ldots, c_m\}$. We now show how to reduce problem 3-SAT to finding a strategy of value greater or equal to m in a polynomially generated decision tree. The polynomial generation of a decision tree from an instance of 3-SAT is performed as follows. One defines a decision node for every variable of X. Given x_i a variable in X, the corresponding decision node in the decision tree, also denoted by x_i, has two children: the first one (chance node denoted by T_i) corresponds to the statement 'x_i has truth value "true"', and the second one (chance node denoted by F_i) corresponds to the statement 'x_i has truth value "false"'. The subset of clauses which includes the positive (resp. negative) literal of x_i is denoted by $\{c_{i_1}, \ldots, c_{i_j}\} \subseteq C$ (resp. $\{c_{i'_1}, \ldots, c_{i'_k}\} \subseteq C$). For every clause c_{i_h} $(1 \leqslant h \leqslant j)$ one generates a child of T_i denoted by c_{i_h} (terminal node). Besides, one generates an additional child of T_i denoted by c_0, corresponding to a fictive consequence. Similarly, one generates a child of F_i for every clause $c_{i'_h}$ $(1 \leqslant h \leqslant k)$, as well as an additional child corresponding to fictive consequence c_0. Node T_i has therefore $j + 1$ children, while node F_i has $k + 1$ children. In order to make a single decision tree, one adds a chance node C predecessor of all decision nodes x_i $(1 \leqslant i \leqslant n)$. Finally, one adds a decision node as root, with C as unique child. The obtained decision tree includes $n + 1$ decision nodes, $2n + 1$ chance nodes and at most $2n(m + 1)$ terminal nodes. Its size is therefore in $O(nm)$, which guarantees the polynomiality of the transformation. For the sake of illustration, in Fig. 10, we represent the decision tree obtained for the following instance of 3-SAT: $(x_1 \vee x_2 \vee x_3) \wedge (\overline{x_1} \vee x_3 \vee x_4) \wedge (\overline{x_2} \vee \overline{x_3} \vee \overline{x_4})$.

Note that one can establish a bijection between the set of strategies in the decision tree and the set of assignments in problem 3-SAT. For that purpose, it is sufficient to set $x_i = 1$ in problem 3-SAT iff edge (x_i, T_i) is included in the strategy, and $x_i = 0$ iff edge (x_i, F_i) is included in the strategy. An assignment such that the entire expression is true in 3-SAT corresponds to a strategy such that every clause c_i $(1 \leqslant i \leqslant m)$ is a possible consequence (each clause appears therefore from one to three times). To complete the reduction, we now have to define, on the one hand, the probabilities assigned to the edges from nodes C, T_i and F_i, and, on the other hand, the utilities of the consequences and function φ. The reduction consists in defining them so that strategies maximizing RDU correspond to assignments for which the entire expression is true in 3-SAT. More precisely, we aim at satisfying the following properties;

(i) the RDU value of a strategy only depends on the *set* (and not the multiset) of its possible consequences (in other words, the set of clauses that become true with the corresponding assignment),

(ii) the RDU value of a strategy corresponding to an assignment that makes satisfiable the 3-SAT expression equals m;

(iii) if a strategy yields a set of possible consequences that is strictly included in the set of possible consequences of another strategy, the RDU value of the latter is strictly greater.

For that purpose, after assigning probability $\frac{1}{n}$ to edges originating from C, one defines the other probabilities and utilities as follows ($i \neq 0$):

$$\begin{cases} p_i = \left(\frac{1}{10}\right)^i \\ u(c_i) = \sum_{j=1}^{i} 10^{j-1} \end{cases}$$

where p_i is the probability assigned to all the edges leading to consequence c_i. For the edges of type (T_j, c_0) (or (F_j, c_0)), one sets $u(c_0) = 0$ and one assigns a probability such that all the probabilities of edges originating from T_j (or F_j) sum up to 1. Note that this latter probability is positive since the sum of p_i's is strictly smaller than 1 by construction. Finally, function φ is defined as follows[2]:

$$\varphi(p) = \begin{cases} 0 & \text{if } p \in \left[0; \frac{p_m}{n}\right) \\ p_i & \text{if } p \in \left[\frac{p_{i+1}}{n}; \frac{p_i}{n}\right) \text{ for } i < m \\ 1 & \text{if } p \in \left[\frac{p_1}{n}; 1\right) \end{cases}$$

For the sake of illustration, we now give function φ obtained for the instance of 3-SAT indicated above:

$$\varphi(p) = \begin{cases} 0 & \text{if } p \in \left[0; \frac{1}{4 \times 1000}\right) \\ \frac{1}{100} & \text{if } p \in \left[\frac{1}{4 \times 1000}; \frac{1}{4 \times 100}\right) \\ \frac{1}{10} & \text{if } p \in \left[\frac{1}{4 \times 100}; \frac{1}{4 \times 10}\right) \\ 1 & \text{if } p \in \left[\frac{1}{4 \times 10}; 1\right) \end{cases}$$

In the following, we consider some strategy Δ, inducing a lottery denoted by L, and we denote by $I \subseteq \{0, \ldots, m\}$ the set of indices of the possible consequences of Δ. Note that consequence c_0 is always present in a strategy Δ. We denote by $\alpha_i \in \{1, 2, 3\}$ the number of occurrences of consequence c_i in Δ. By abuse of notation, we use indifferently c_i and $u(c_i)$ below.

Proof of (i). The RDU value of Δ is $RDU(L) = c_0 \times \varphi(1) + \sum_{i \in I}(c_i - c_{prev_I(i)})\varphi(\sum_{\substack{j \in I \\ j \geqslant i}} \alpha_j \frac{p_j}{n})$ where $prev_I(i) = \max\{j \in I : j < i\}$. We now show that $\forall i \in I, \varphi(\sum_{\substack{j \in I \\ j \geqslant i}} \alpha_j \frac{p_j}{n}) = \varphi(\sum_{\substack{j \in I \\ j \geqslant i}} \frac{p_j}{n})$. By increasingness of φ, we have

$$\varphi\left(\sum_{\substack{j \in I \\ j \geqslant i}} \frac{p_j}{n}\right) \leqslant \varphi\left(\sum_{\substack{j \in I \\ j \geqslant i}} \alpha_j \frac{p_j}{n}\right) \leqslant \varphi\left(\sum_{\substack{j \in I \\ j \geqslant i}} 3\frac{p_j}{n}\right)$$

Therefore

$$\varphi\left(\sum_{\substack{j \in I \\ j \geqslant i}} \frac{1}{n}\left(\frac{1}{10}\right)^j\right) \leqslant \varphi\left(\sum_{\substack{j \in I \\ j \geqslant i}} \alpha_j \frac{p_j}{n}\right) \leqslant \varphi\left(\sum_{\substack{j \in I \\ j \geqslant i}} \frac{3}{n}\left(\frac{1}{10}\right)^j\right)$$

[2] Note that function φ is not strictly increasing here, but the reader can easily convince himself that it can be slightly adapted so that it becomes strictly increasing.

Since

$$\varphi\left(\sum_{\substack{j\in I\\ j\geqslant i}}\frac{1}{n}\left(\frac{1}{10}\right)^j\right)=\varphi\left(\sum_{\substack{j\in I\\ j\geqslant i}}\frac{3}{n}\left(\frac{1}{10}\right)^j\right)=p_{i-1}$$

we have by bounding

$$\varphi\left(\sum_{\substack{j\in I\\ j\geqslant i}}\alpha_j\frac{p_j}{n}\right)=\varphi\left(\sum_{\substack{j\in I\\ j\geqslant i}}\frac{p_j}{n}\right)$$

Hence

$$RDU(L)=\sum_{i\in I}(c_i-c_{prev_I(i)})\varphi\left(\sum_{\substack{j\in I\\ j\geqslant i}}\frac{p_j}{n}\right)\quad\text{since } c_0\times\varphi(1)=0$$

Proof of (ii). Consider a strategy Δ^* corresponding to an assignment that makes the expression true, and the induced lottery L^* where all the consequences c_i of C are possible. By (i), we have

$$RDU\big(L^*\big)=\sum_{i=1}^{m}(c_i-c_{i-1})\varphi\left(\sum_{j=i}^{m}\frac{p_j}{n}\right)$$

We note that for all $i\leqslant m$, $(c_i-c_{i-1})\varphi(\sum_{j=i}^{m}\frac{p_j}{n})=10^{i-1}\times p_{i-1}=10^{i-1}\times(\frac{1}{10})^{i-1}=1$. Consequently, $RDU(L^*)=m$.

Proof of (iii). Let Δ (resp. Δ') denote some strategy the induced lottery of which is L (resp. L') and let $I\subseteq\{0,\ldots,m\}$ (resp. $J=I\cup\{k\}$) denote the set of indices of its possible consequences. We assume here that $k<\max I$, the case $k=\max I$ being obvious. By definition, $\{i\in I\colon\ i\neq k\}=\{i\in J\colon\ i\neq k\}$. We can therefore state the RDU value as a sum of three terms:

$$RDU(L)=\sum_{\substack{i\in J\\ i\leqslant k-1}}(c_i-c_{prev_J(i)})\varphi\left(\sum_{\substack{j\in I\\ j\geqslant i}}\frac{p_j}{n}\right)+(c_k-c_{prev_J(k)})\varphi\left(\sum_{\substack{j\in I\\ j\geqslant k}}\frac{p_j}{n}\right)+\sum_{\substack{i\in J\\ i\geqslant k+1}}(c_i-c_{prev_J(i)})\varphi\left(\sum_{\substack{j\in J\\ j\geqslant i}}\frac{p_j}{n}\right)$$

Similarly, the RDU value of strategy Δ' can also be stated as a sum of three terms:

$$RDU\big(L'\big)=\sum_{\substack{i\in J\\ i\leqslant k-1}}(c_i-c_{prev_J(i)})\varphi\left(\sum_{\substack{j\in J\\ j\geqslant i}}\frac{p_j}{n}\right)+(c_k-c_{prev_J(k)})\varphi\left(\sum_{\substack{j\in J\\ j\geqslant k}}\frac{p_j}{n}\right)+\sum_{\substack{i\in J\\ i\geqslant k+1}}(c_i-c_{prev_J(i)})\varphi\left(\sum_{\substack{j\in J\\ j\geqslant i}}\frac{p_j}{n}\right)$$

By increasingness of φ, we have

$$I\subseteq J\quad\Rightarrow\quad\forall i\leqslant k-1,\varphi\left(\sum_{\substack{j\in I\\ j\geqslant i}}\frac{p_j}{n}\right)\leqslant\varphi\left(\sum_{\substack{j\in J\\ j\geqslant i}}\frac{p_j}{n}\right)$$

Thus the first term of $RDU(L)$ is smaller or equal to the first term of $RDU(L')$. One checks easily that $\varphi(\sum_{\substack{j\in I\\ j\geqslant k}}\frac{p_j}{n})=p_{succ_I(k)-1}$ and $\varphi(\sum_{\substack{j\in J\\ j\geqslant k}}\frac{p_j}{n})=p_{prev_J(k)}=p_{k-1}$, where $succ_I(i)=\min\{j\in I\colon\ j>i\}$. But $p_{succ_I(k)-1}<p_{k-1}$ since $succ_I(k)-1>k-1$. Therefore the second term of $RDU(L)$ is strictly smaller than the second term of $RDU(L')$. Finally, the third term of $RDU(L)$ is of course equal to the third term of $RDU(L')$. Consequently $RDU(L)<RDU(L')$.

From (i), (ii) and (iii) we conclude that any strategy corresponding to an assignment that does not make the expression true has a RDU value strictly smaller than m, and that any strategy corresponding to an assignment that makes the expression true has a RDU value exactly equal to m. Solving 3-SAT reduces therefore to determining a strategy of value m in RDU-OPT. □

References

[1] M. Abdellaoui, Parameter-free elicitation of utilities and probabilities weighting functions, Management Science 46 (11) (2000) 1497–1512.
[2] M. Allais, Le comportement de l'homme rationnel devant le risque: critique des postulats de l'école américaine, Econometrica 21 (1953) 503–546.
[3] M. Allais, An outline of my main contributions to economic science, The American Economic Review 87 (6) (1997) 3–12.
[4] K.J. Arrow, The Theory of Risk Aversion, Ynjo Jahnsonin Saatio, Helsinki, 1965.
[5] U. Bertelé, F. Brioschi, Nonserial Dynamic Programming, Academic Press, 1972.
[6] J. Blythe, Decision-theoretic planning, AI Magazine 20 (1999) 1–15.
[7] C. Boutilier, T. Dean, S. Hanks, Decision-theoretic planning: Structural assumptions and computational leverage, Journal of AI Research 11 (1999) 1–94.
[8] C. Camerer, T. Ho, Non-linear weighting of probabilities and violations of the betweenness axiom, Journal of Risk and Uncertainty 8 (1994) 167–196.
[9] A. Chateauneuf, Comonotonicity axioms and rank-dependent expected utility theory for arbitrary consequences, Journal of Mathematical Economics 32 (1999) 21–45.
[10] E. Damiani, S. De Capitani di Vimercati, P. Samarati, M. Viviani, A WOWA-based aggregation technique on trust values connected to metadata, Electronic Notes in Theoretical Computer Science 157 (2006) 131–142.
[11] T. Dean, L. Kaelbling, J. Kirman, A. Nicholson, Planning with deadlines in stochastic domains, in: Proc. of the 11th AAAI, 1993, pp. 574–579.
[12] D. Dubois, H. Prade, R. Sabbadin, Decision-theoretic foundations of qualitative possibility theory, European Journal of Operational Research 128 (3) (2001) 459–478.
[13] R. Gonzales, G. Wu, On the shape of the probability weighting function, Cognitive Psychology 38 (1999) 129–166.
[14] P. Hammond, Consequentialism and the independence axiom, in: Risk, Decision and Rationality, D. Reidel Publishing Co, Dordrecht, Holland, 1988, pp. 503–515.
[15] P. Hammond, Consequentialist foundations for expected utility, Theory and Decision 25 (1) (1988) 25–78.
[16] J. Handa, Risk, probabilities and a new theory of cardinal utility, Journal of Political Economics 85 (1977) 97–122.
[17] R. Howard, J. Matheson, Risk-sensitive Markov decision processes, Management Science 18 (7) (1972) 356–369.
[18] R. Howard, J. Matheson, Influence diagrams, in: Readings on the Principles and Applications of Decision Analysis, Strategic Decisions Group, Menlo Park, CA, 1984, pp. 721–762. Reprinted: Decision Analysis 2 (3) (2005) 127–143.
[19] J.-Y. Jaffray, T. Nielsen, An operational approach to rational decision making based on rank dependent utility, European Journal of Operational Research 169 (1) (2006) 226–246.
[20] G. Jeantet, O. Spanjaard, Rank-dependent probability weighting in sequential decision problems under uncertainty, in: Proc. of the International Conference on Automated Planning and Scheduling (ICAPS), 2008, pp. 148–155.
[21] F. Jensen, F.V. Jensen, S.L. Dittmer, From influence diagrams to junction trees, in: Proc. of the Tenth Annual Conference on Uncertainty in Artificial Intelligence (UAI), 1994, pp. 367–373.
[22] L. Kaebling, M. Littman, A. Cassandra, Planning and acting in partially observable stochastic domains, Artificial Intelligence 101 (1999) 99–134.
[23] D. Kahneman, A. Tversky, Prospect theory: An analysis of decision under risk, Econometrica 47 (1979) 263–291.
[24] U. Karmarkar, Subjectively weighted utility and the Allais paradox, Organisational Behavior and Human Performance 24 (1) (1979) 67–72.
[25] S. Koenig, R.G. Simmons, Risk-sensitive planning with probabilistic decision graphs, in: Proc. of the International Conference on Principles of Knowledge Representation and Reasoning (KR), 1994, PP. 363–373.
[26] Y. Liu, S. Koenig, Existence and finiteness conditions for risk-sensitive planning: Results and conjectures, in: Proc. of the Twentieth Annual Conference on Uncertainty in Artificial Intelligence (UAI), 2005, pp. 354–363.
[27] Y. Liu, S. Koenig, Risk-sensitive planning with one-switch utility functions: Value iteration, in: Proc. of the Twentieth National Conference on Artificial Intelligence, AAAI, 2005, pp. 993–999.
[28] Y. Liu, S. Koenig, An exact algorithm for solving MDPs under risk-sensitive planning objectives with one-switch utility functions, in: Proc. of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2008, pp. 453–460.
[29] R. Luce, H. Raiffa, Games and Decisions, John Wiley, New York, 1957.
[30] M.J. Machina, Dynamic consistency and non-expected utility models of choice under uncertainty, Journal of Economic Literature 27 (4) (1989) 1622–1668.
[31] E. McClennen, Rationality and Dynamic Choice: Foundational Explorations, Cambridge University Press, 1990.
[32] T. Morin, Monotonicity and the principle of optimality, Journal of Mathematical Analysis and Applications 86 (1982) 665–674.
[33] T.D. Nielsen, F.V. Jensen, Learning a decision maker's utility function from (possibly) inconsistent behavior, Artificial Intelligence Journal 160 (2004) 53–78.
[34] W. Ogryczak, T. Sliwinski, On decision support under risk by the WOWA optimization, in: ECSQARU 2007, pp. 779–790.
[35] W. Ogryczak, WOWA enhancement of the preference modeling in the reference point method, in: Proc. of the Fifth International Conference on Modeling Decisions for Artificial Intelligence, in: Lecture Notes in Artificial Intelligence, vol. 5285, Springer, 2008, pp. 38–49.
[36] W. Ogryczak, T. Sliwinski, On efficient WOWA optimization for decision support under risk, International Journal of Approximate Reasoning 50 (2009) 915–928.
[37] F. Perea, J. Puerto, Dynamic programming analysis of the TV game "Who wants to be a millionaire?, European Journal of Operational Research 183 (2007) 805–811.
[38] P. Perny, O. Spanjaard, L.-X. Storme, State space search for risk-averse agents, in: 20th International Joint Conference on Artificial Intelligence, 2007, pp. 2353–2358.
[39] J. Pratt, Risk aversion in the small and in the large, Econometrica 32 (1) (1964) 122–136.
[40] J. Quiggin, Generalized Expected Utility Theory: The Rank-Dependent Model, Kluwer, 1993.
[41] H. Raiffa, Decision Analysis: Introductory Lectures on Choices under Uncertainty, Addison–Wesley, 1968.
[42] M. Rotshild, J. Stiglitz, Increasing risk I: A definition, Journal of Economic Theory 2 (3) (1970) 225–243.
[43] L. Savage, The Foundations of Statistics, Wiley, 1954.
[44] R. Shachter, Evaluating influence diagrams, Operations Research 34 (1986) 871–882.
[45] V. Torra, Weighted OWA operators for synthesis of information in: Proc. of the Fifth IEEE International Conference on Fuzzy Systems (IEEE-FUZZ'96), 1996, pp. 966–971.
[46] V. Torra, The weighted OWA operator, International Journal of Intelligent Systems 12 (1997) 153–166.
[47] V. Torra, The WOWA operator and the interpolation function W^*: Chen and Otto's interpolation method revisited, Fuzzy Sets and Systems 113 (3) (2000) 389–396.
[48] A. Tversky, D. Kahneman, Advances in prospect theory: Cumulative representation of uncertainty, Journal of Risk and Uncertainty 5 (1992) 297–323.
[49] J. von Neumann, O. Morgenstern, Theory of Games and Economic Behaviour, Princeton University Press, 1947.