

Rank-Dependent Probability Weighting in Sequential Decision Problems under Uncertainty

Gildas Jeantet and Olivier Spanjaard

LIP6 - UPMC

104 av du Président Kennedy

75016 Paris, France

{gildas.jeantet, olivier.spanjaard}@lip6.fr

Abstract

This paper is devoted to the computation of optimal strategies in automated sequential decision problems. We consider here problems where one seeks a strategy which is optimal for rank dependent utility (RDU). RDU generalizes von Neumann and Morgenstern's expected utility (by probability weighting) to encompass rational decision behaviors that EU cannot accommodate. The induced algorithmic problem is however more difficult to solve since the optimality principle does not hold anymore. More crucially, we prove here that the search for an optimal strategy (w.r.t. RDU) in a decision tree is an NP-hard problem. We propose an implicit enumeration algorithm to compute optimal rank dependent utility in decision trees. The performances of our algorithm on randomly generated instances and real-world instances of different sizes are presented and discussed.

Introduction

Many AI problems can be formalized as planning problems under uncertainty (robot control, relief organization, medical treatments, games...). Planning under uncertainty concerns sequential decision problems where the consequences of actions are dependent on exogeneous events. Decision theory provides useful tools to deal with uncertainty in decision problems. The term *decision-theoretic planning* refers to planners involving such decision-theoretic tools (Blythe, 1999; Boutilier, Dean, and Hanks, 1999). A planner returns a *plan*, i.e. a sequence of actions conditioned by events. In planning under uncertainty, the consequence of a plan is obviously non deterministic. A plan can then be associated with a probability distribution (lottery) on the potential consequences. Comparing plans amounts therefore to comparing lotteries. The purpose of decision theory under risk is precisely to provide tools to evaluate lotteries in order to compare them. These tools are called hereafter *decision criteria*. Formally, the aim of a decision-theoretic planner is to find a plan optimizing a given decision criterion. A popular criterion is the expected utility (EU) model proposed by von Neuman and Morgenstern (1947). In this model, an agent is endowed with a *utility function* u that assigns a numerical value to each consequence. The evaluation of a plan is then performed via the computation of its utility

expectation. 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 (Allais, 1953). We present below a very simple version of this paradox due to Kahneman and Tversky (1979).

Example 1 (Allais' paradox) Consider a choice situation where two options are presented to a decision maker. She 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 problem (see Table 1). In the first problem she prefers L_1 to L'_1

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

Table 1: Lotteries in Allais' paradox.

(she is certain to earn 3000\$ with L_1 while she might earn nothing with L'_1), while in the second problem she 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 attitude, far from being paradoxical, corresponds to a reasonable behavior of *preference for security in the neighbourhood of certainty* (Allais, 1997). In other words, "a bird in the hand is worth two in the bush". It is known as the *certainty effect*. In the example, the probability of winning 3000\$ in L_1 (resp. 4000\$ in L'_1) is simply multiplied by 0.1 in L_2 (resp. L'_2). 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. This has led researchers to sophisticate the definition of expected utility. Among the most popular generalizations of EU, let

us mention the rank dependent utility (RDU) model introduced by Quiggin (1993). 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 the Allais' paradox. Furthermore, the probability weighting function φ is also useful to model the attitude of the agent towards the risk. Indeed, contrary to 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 (Perny, Spanjaard, and Storme, 2007). This concern of modeling risk-averse behaviors has also been raised in the context of Markov decision processes, for which Liu and Koenig (2008) have shown the interest of using one-switch utility functions.

In this paper, we investigate how to compute an RDU-optimal plan in planning under uncertainty. Several representation formalisms can be used for decision-theoretic planning, such as decision trees (e.g., Raiffa (1968)), influence diagrams (e.g., Shachter (1986)) or Markov decision processes (e.g., Dean et al. (1993); Kaelbling, Littman, and Cassandra (1999)). A decision tree is a direct 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. For simplicity, we study here the decision tree formalism. Note that an approach similar to the one we propose here could be applied to influence diagrams and finite horizon Markov decision processes, with a few customizations. The evaluation of a decision tree is the process of finding an optimal plan from the tree. The computation of a strategy maximizing RDU in a decision tree (in a decision tree, a plan is called a strategy) is a combinatorial problem since the number of potential strategies in a complete binary decision tree is in $\Theta(2^{\sqrt{n}})$, where n denotes the size of the decision tree. Contrary to the computation of a strategy maximizing EU, one cannot directly resort to dynamic programming for computing a strategy maximizing RDU. This raises a challenging algorithmic problem, provided the combinatorial number of potential strategies.

The paper is organized as follows. We first recall the main features of RDU and introduce the decision tree formalism. After showing that dynamic programming fails to optimize RDU in decision trees, we prove that the problem is in fact NP-hard. Next, we present our upper bounding procedure and the ensuing implicit enumeration algorithm. Finally, we provide numerical tests on random instances and on different representations of the game *Who wants to be a millionaire?*.

Rank Dependent Utility

Given a finite set $S = \{u_1, \dots, u_k\}$ of utilities, any strategy in a sequential decision problem can be seen as a *lottery*,

characterized by a probability distribution P over S . We denote by $L = (p_1, u_1; \dots; p_k, u_k)$ the lottery that yields utility u_i with probability $p_i = P(\{u_i\})$. For the sake of clarity, we will consider a lottery L as a function from S to $[0, 1]$ such that $L(u_i) = p_i$. Rank dependent utility, introduced by Quiggin (1993), is among the most popular generalizations of EU, and makes it possible to describe sophisticated rational decision behaviors. It involves a utility function on consequences as in EU, and also a probability transformation function φ . This is a non-decreasing function, proper to any agent, such that $\varphi(0) = 0$ and $\varphi(1) = 1$. Note that the distortion is not applied on probabilities themselves, but on cumulative probabilities. For any lottery $L = (p_1, u_1; \dots; p_k, u_k)$, the decumulative function G_L is given by $G_L(x) = \sum_{i:u_i \geq x} p_i$. We denote by $(G_L(u_1), u_1; \dots; G_L(u_k), u_k)$ the decumulative function of lottery L . The rank dependent utility of a lottery L is then defined as follows:

$$RDU(L) = u_{(1)} + \sum_{i=2}^k [u_{(i)} - u_{(i-1)}] \varphi(G_L(u_{(i)}))$$

where $(.)$ represents a permutation on $\{1, \dots, k\}$ such that $u_{(1)} \leq \dots \leq u_{(k)}$. This criterion can be interpreted as follows: the utility of lottery L is at least $u_{(1)}$ with probability 1; then the utility might increase from $u_{(1)}$ to $u_{(2)}$ with probability mass $\varphi(G_L(u_{(2)}))$; the same applies from $u_{(2)}$ to $u_{(3)}$ with probability mass $\varphi(G_L(u_{(3)}))$, and so on... When $\varphi(p) = p$ for all p , it can be shown that RDU 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 strategy returned by RDU is then compatible with the Allais' paradox. Indeed, we have:*

$$\begin{aligned} RDU(L_1) &= u(3000) = 3000 \\ RDU(L'_1) &= u(0) + \varphi(0.9)(u(4000) - u(0)) = 2800 \end{aligned}$$

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

$$\begin{aligned} RDU(L_2) &= u(0) + \varphi(0.1)(u(3000) - u(0)) = 600 \\ RDU(L'_2) &= u(0) + \varphi(0.09)(u(4000) - u(0)) = 800 \end{aligned}$$

We conclude that L'_2 is preferred to L_2 .

The main interest of distorting cumulative probabilities, rather than probabilities themselves (as in Handa's model, 1977), is to get a choice criterion that is compatible with stochastic dominance. A lottery $L = (p_1, u_1; \dots; p_k, u_k)$ is said to stochastically dominate a lottery $L' = (p'_1, u'_1; \dots; p'_k, u'_k)$ if for all $x \in \mathbb{R}$, $G_L(x) \geq G_{L'}(x)$. In other words, for all $x \in \mathbb{R}$, the probability to get a utility at least x with lottery L is at least as high as the probability with lottery L' . Compatibility with stochastic dominance means that $RDU(L) \geq RDU(L')$ as soon as L stochastically dominates L' . This property is obviously desirable to guarantee a rational behavior, and is satisfied by RDU model (contrary to Handa's model).

Decision Tree

A decision tree is an arborecence with three types of nodes: the *decision nodes* (represented by squares), the *chance nodes* (represented by circles), and the terminal nodes (the

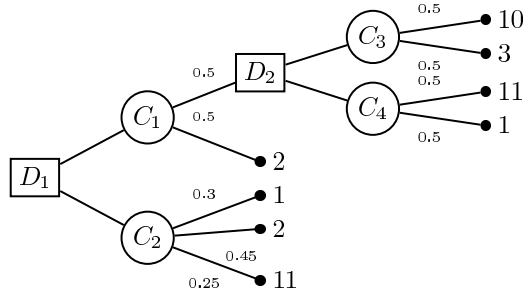


Figure 1: A decision tree representation.

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 Figure 1.

More formally, in a decision tree $\mathcal{T} = (\mathcal{N}, \mathcal{E})$, the root node is denoted by N_r , the set of decision nodes by $\mathcal{N}_D \subset \mathcal{N}$, the set of chance nodes by $\mathcal{N}_C \subset \mathcal{N}$, and the set of terminal nodes by $\mathcal{N}_T \subset \mathcal{N}$. The valuations are defined as follows: every edge $E = (C, N) \in \mathcal{E}$ such that $C \in \mathcal{N}_C$ is weighted by probability $p(E)$ of the corresponding event; every terminal node $N_T \in \mathcal{N}_T$ is labelled by its utility $u(N_T)$. Besides, we call *past*(N) the *past* of $N \in \mathcal{N}$, 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 (2006), 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 $N \in \mathcal{N}_C^\Delta = \mathcal{N}_C \cap \mathcal{N}^\Delta$.

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*.

Note that the number of potential strategies grows exponentially with the size of the decision tree, i.e. the number of decision nodes (this number has indeed the same order of magnitude as the number of nodes in \mathcal{T}). Indeed, one easily shows that there are $\Theta(2^{\sqrt{n}})$ strategies in a complete binary decision tree \mathcal{T} . For this reason, it is necessary to develop an optimization algorithm to determine the optimal strategy. 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. Starting from the leaves, one computes 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 Figure 1, the optimal expected utility at node D_2 is $\max\{6.5, 6\} = 6.5$. Consequently, the optimal expected utility at node C_1 is 4.25. The expected utility at node C_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 $\{(D_1, C_1), (D_2, C_3)\}$. Note that this is not an optimal strategy when using RDU to evaluate lotteries (see next section).

We show below that the computation of a strategy optimizing RDU is more delicate since dynamic programming no longer applies.

Monotonicity and Independence

It is well known that the validity of dynamic programming procedures strongly relies on a property of monotonicity (Morin, 1982) of the value function. In our context, this condition can be stated as follows on the value function V of lotteries:

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

where L, L', L'' are lotteries, α is a scalar in $[0, 1]$ and $\alpha L + (1 - \alpha)L''$ is the lottery defined by $(\alpha L + (1 - \alpha)L'')(x) = \alpha L(x) + (1 - \alpha)L''(x)$. This *algorithmic* condition can be understood, in the framework of decision theory, as a weak version of the *independence* axiom used by von Neuman and Morgenstern (1947) 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 $V \equiv EU$. However, it is not true anymore for $V \equiv RDU$, as shown by the following example.

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

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

The RDU values of lotteries L and L' are:

$$\begin{aligned} RDU(L) &= 3 + (10 - 3)\varphi(0.5) = 7.2 \\ RDU(L') &= 1 + (11 - 1)\varphi(0.5) = 7 \end{aligned}$$

Thus, we have $RDU(L) \geq RDU(L')$. By the monotonicity condition for $\alpha = 0.5$, one should therefore have $RDU(0.5L + 0.5L'') \geq RDU(0.5L' + 0.5L'')$. However, we have:

$$\begin{aligned} RDU(0.5L + 0.5L'') &= 2 + (3 - 1)\varphi(0.5) + (10 - 3)\varphi(0.25) = 5.75 \\ RDU(0.5L' + 0.5L'') &= 1 + (2 - 1)\varphi(0.75) + (11 - 2)\varphi(0.25) = 6.65 \end{aligned}$$

Therefore $RDU(0.5L + 0.5L'') < RDU(0.5L' + 0.5L'')$. Consequently, the monotonicity property does not hold.

It follows that one cannot directly resort to dynamic programming to compute an optimal strategy: it could yield a suboptimal strategy. Such a procedure could even yield a stochastically dominated strategy, as shown by the following example.

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

$$\begin{aligned} RDU(\{(D_1, C_2)\}) &= 5.8 \\ RDU(\{(D_1, C_1), (D_2, C_3)\}) &= 5.75 \\ RDU(\{(D_1, C_1), (D_2, C_4)\}) &= 6.65 \end{aligned}$$

Thus, the optimal strategy at the root is $\{(D_1, C_1), (D_2, C_4)\}$. However, by dynamic programming, one gets at node D_2 : $RDU(\{(D_2, C_3)\}) = 7.2$ and $RDU(\{(D_2, C_4)\}) = 7$. This is therefore the substrategy $\{(D_2, C_3)\}$ that is obtained at node D_2 . At node D_1 , this is thereafter the strategy $\{(D_1, C_2)\}$ (5.8 vs 5.75 for $\{(D_1, C_1), (D_2, C_3)\}$), stochastically dominated by $\{(D_1, C_1), (D_2, C_4)\}$, which is returned.

For this reason, a decision maker using the RDU criterion should adopt a *resolute choice* behavior (McClennen, 1990), i.e. she initially chooses a strategy and never deviates from it later (otherwise she could follow a stochastically dominated strategy). We focus here on determining an RDU-optimal strategy from the root. By doing so, we are sure to never encounter a stochastically dominated substrategy, contrarily to a method that would consist in performing backward induction with RDU. Other approaches of resolute choice have been considered in the literature. For example, Jaffray and Nielsen (2006) consider each decision node in the decision tree as an *ego* of the decision maker, and aim at determining a strategy achieving a compromise between the egos, such that all its substrategies are close to optimality for RDU and stochastically non-dominated.

Computational Complexity

We now prove that 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 *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 $X = \{x_1, \dots, x_n\}$ and $C = \{c_1, \dots, c_m\}$. 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 $\{c_{i_1}, \dots, c_{i_j}\} \subseteq C$ (resp. $\{c'_{i_1}, \dots, c'_{i_k}\} \subseteq C$). For every clause c_{i_h} ($1 \leq h \leq j$) 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} ($1 \leq h \leq k$), 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 ($1 \leq i \leq n$). 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, on Figure 2, 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 ($1 \leq i \leq m$) 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

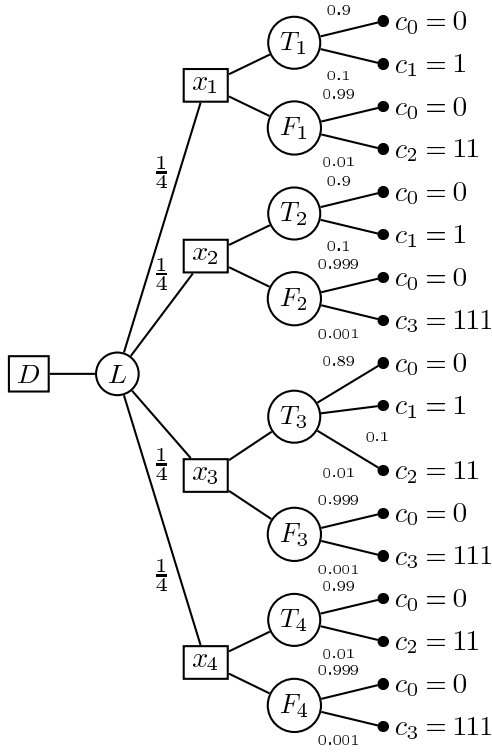


Figure 2: An example of reduction.

positive since the sum of p_i 's is strictly smaller than 1 by construction. Finally, function φ is defined as follows¹:

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

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

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

In the following, we consider some strategy Δ , inducing a lottery denoted by L , and we denote by $I \subseteq \{0, \dots, m\}$ 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 $RDU(L) = c_0 \times \varphi(1) + \sum_{i \in I} (c_i - c_{prev_I(i)}) \varphi(\sum_{j \geq i} \alpha_j \frac{p_j}{n})$

where $prev_I(i) = \max\{j \in I : j < i\}$. We now show that

$$\forall i \in I, \varphi(\sum_{j \geq i} \alpha_j \frac{p_j}{n}) = \varphi(\sum_{j \geq i} \frac{p_j}{n})$$

By increasingness of φ , we have

$$\varphi(\sum_{j \geq i} \frac{p_j}{n}) \leq \varphi(\sum_{j \geq i} \alpha_j \frac{p_j}{n}) \leq \varphi(\sum_{j \geq i} 3 \frac{p_j}{n})$$

¹Note 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.

Therefore

$$\varphi(\sum_{j \geq i} \frac{1}{n} (\frac{1}{10})^j) \leq \varphi(\sum_{j \geq i} \alpha_j \frac{p_j}{n}) \leq \varphi(\sum_{j \geq i} \frac{3}{n} (\frac{1}{10})^j)$$

Since $\varphi(\sum_{j \geq i} \frac{1}{n} (\frac{1}{10})^j) = \varphi(\sum_{j \geq i} \frac{3}{n} (\frac{1}{10})^j) = p_{i-1}$

we have by bounding $\varphi(\sum_{j \geq i} \alpha_j \frac{p_j}{n}) = \varphi(\sum_{j \geq i} \frac{p_j}{n})$

Hence $RDU(L) = \sum_{i \in I} (c_i - c_{prev_I(i)}) \varphi(\sum_{j \geq i} \frac{p_j}{n})$ 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

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

We note that for all $i \leq m$, $(c_i - c_{i-1}) \varphi(\sum_{j=i}^m \frac{p_j}{n}) = 10^{i-1} \times p_{i-1} = 10^{i-1} \times (\frac{1}{10})^{i-1} = 1$. 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 $I \subseteq \{0, \dots, m\}$ (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, $\{i \in I : i \neq k\} = \{i \in J : i \neq k\}$. We can therefore state the RDU value as a sum of three terms:

$$\begin{aligned} RDU(L) &= \sum_{i \leq k-1}^{i \in J} (c_i - c_{prev_J(i)}) \varphi(\sum_{j \geq i} \frac{p_j}{n}) \\ &+ (c_k - c_{prev_J(k)}) \varphi(\sum_{j \geq k} \frac{p_j}{n}) \\ &+ \sum_{i \geq k+1}^{i \in J} (c_i - c_{prev_J(i)}) \varphi(\sum_{j \geq i} \frac{p_j}{n}) \end{aligned}$$

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

$$\begin{aligned} RDU(L') &= \sum_{i \leq k-1}^{i \in J} (c_i - c_{prev_J(i)}) \varphi(\sum_{j \geq i} \frac{p_j}{n}) \\ &+ (c_k - c_{prev_J(k)}) \varphi(\sum_{j \geq k} \frac{p_j}{n}) \\ &+ \sum_{i \geq k+1}^{i \in J} (c_i - c_{prev_J(i)}) \varphi(\sum_{j \geq i} \frac{p_j}{n}) \end{aligned}$$

By increasingness of φ , we have

$$I \subseteq J \Rightarrow \forall i \leq k-1, \varphi(\sum_{j \geq i} \frac{p_j}{n}) \leq \varphi(\sum_{j \geq i} \frac{p_j}{n})$$

Thus the first term of $RDU(L)$ is smaller or equal to the first term of $RDU(L')$. One checks easily that $\varphi(\sum_{j \geq k} \frac{p_j}{n}) =$

$p_{succ_I(k)-1}$ and $\varphi(\sum_{j \geq k} \frac{p_j}{n}) = p_{prev_J(k)} = p_{k-1}$, where $succ_I(i) = \min\{j \in I : j > i\}$. 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. ■

In the following section, we describe an algorithm for determining an RDU-optimal strategy in a decision tree. We proceed by implicit enumeration since neither exhaustive enumeration nor backward induction are conceivable.

Implicit Enumeration

We now present a branch and bound method for determining an RDU-optimal strategy. 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}^Γ 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 $\Gamma = \{(N, N') : N \in \mathcal{N}_D^\Gamma, N' \in \mathcal{N}^\Gamma\} \subseteq \mathcal{E}$ defines a *partial strategy* of \mathcal{T} if the subgraph induced by \mathcal{N}^Γ 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 rg : if several decision nodes are candidates to enter \mathcal{N}^Γ , the one with the lowest priority rank will be considered first. For the sake of brevity, we do not elaborate here on this priority function.

Algorithm 1 describes formally the implicit enumeration procedure that we propose. 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}^Γ 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 the start of 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

Algorithm 1: $\mathbf{BB}(\Gamma, RDU_{opt})$

```

 $\mathcal{N}_1 \leftarrow \{N_1 \in \mathcal{N}_D : N_1 \text{ is candidate}\};$ 
 $N_{min} \leftarrow \arg \min_{N \in \mathcal{N}_1} rg(N);$ 
 $\mathcal{E}_{min} \leftarrow \{(N_{min}, C) \in \mathcal{E} : C \in S(N_{min})\};$ 
for each  $(N, C) \in \mathcal{E}_{min}$  do
  if  $ev(\Gamma \cup \{(N, C)\}) > RDU_{opt}$  then
     $RDU_{temp} \leftarrow \mathbf{BB}(\Gamma \cup \{(N, C)\}, RDU_{opt});$ 
    if  $RDU_{temp} > RDU_{opt}$  then
       $RDU_{opt} \leftarrow RDU_{temp};$ 
    end
  end
end
return  $RDU_{opt}$ 

```

evaluation is to determine a lottery that stochastically dominates any lottery corresponding to a strategy compatible with Γ , and then to evaluate this lottery according to RDU. This yields an upper bound since RDU is compatible with stochastic dominance, i.e. if L stochastically dominated L' then $RDU(L) \geq RDU(L')$. In order to compute such a lottery, one proceeds by dynamic programming in the decision tree. The initialization is performed as follows: at each terminal node $T \in \mathcal{N}_T$ is assigned a lottery $(1, u(T))$. Next, at each node $N \in \mathcal{N}$, one computes a lottery that stochastically dominates all the lotteries of subtree $T(N)$. More precisely, at a chance node C , one computes lottery L^C induced by the lotteries of its children as follows:

$$\forall u, L^C(u) = \sum_{N \in S(C)} p((C, N)) \times L^C(u)$$

where L^C corresponds to the lottery assigned to node N . Besides, at each decision node D , we apply the following recurrence relation on the decumulative functions²:

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

Finally, the value returned by ev is $RDU(L^{N_r})$. To prove the validity of this procedure, one can proceed by induction : the key point is that if lottery L stochastically dominates a lottery L' , then $\alpha L + (1 - \alpha)L''$ stochastically dominates $\alpha L' + (1 - \alpha)L''$ for any $\alpha \in [0, 1]$ and any lottery L'' .

Example 6 Let us come back to the decision tree of Figure 1, and assume that $\Gamma = \{(D_1, C_1)\}$. The lotteries assigned to the nodes are:

$$L^{C_3} = (\frac{1}{2}, 3; \frac{1}{2}, 10), L^{C_4} = (\frac{1}{2}, 1; \frac{1}{2}, 11), L^{D_2} = (\frac{1}{2}, 3; \frac{1}{2}, 11)$$

$$\text{since } G_{L^{D_2}} = \begin{cases} \max(G_{L^{C_3}}(1), G_{L^{C_4}}(1)), 1 \\ \max(G_{L^{C_3}}(3), G_{L^{C_4}}(3)), 3 \\ \max(G_{L^{C_3}}(10), G_{L^{C_4}}(10)), 10 \\ \max(G_{L^{C_3}}(11), G_{L^{C_4}}(11)), 11 \end{cases}$$

$$= (1, 3; \frac{1}{2}, 11)$$

$$L^{C_1} = (\frac{1}{2}, 2; \frac{1}{2} \times \frac{1}{2}, 3; \frac{1}{2} \times \frac{1}{2}, 11) = (\frac{1}{2}, 2; \frac{1}{4}, 3; \frac{1}{4}, 11)$$

$$L^{D_1} = L^{C_1} = (\frac{1}{2}, 2; \frac{1}{4}, 3; \frac{1}{4}, 11)$$

The returned value for $\Gamma = \{(D_1, C_1)\}$ is therefore $ev(\Gamma) = RDU((\frac{1}{2}, 2; \frac{1}{4}, 3; \frac{1}{4}, 11))$.

²Note that one can indifferently work with lotteries or decumulative functions, since only the decumulative function matters in the calculation of RDU.

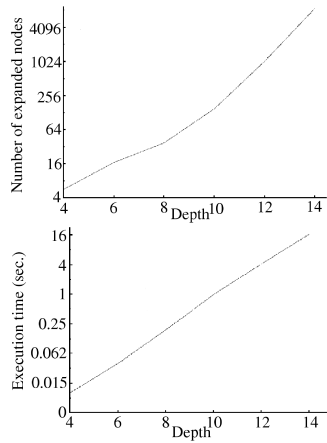


Figure 3: Behavior of the algorithm w.r.t. the depth.

Numerical Tests

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

Random instances. Our tests were performed on complete binary decision trees of even depth. The depth of these decision trees varies from 4 to 14 (5 to 5461 decision nodes), with an alternation of decision nodes and chance nodes. Utilities are real numbers randomly drawn within interval $[1, 500]$. Figure 3 presents the performances of the algorithm with respect to the depth of the decision tree. For each depth level, we give the average performance computed over 100 decision trees. The upper (resp. lower) curve gives the average number of expanded nodes in the enumeration tree (resp. the average execution time in sec.). Note that the y-axis is graduated on a logarithmic scale (basis 4) since the number of decision nodes is multiplied by 4 for each increment of the depth. For the sizes handled here, the number of expanded nodes and the execution time grow linearly with the number of decision nodes. Note that, for bigger instances (i.e. the depth of which is greater than 14), some rare hard instances begin to appear for which the resolution time becomes high. 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: she then quits the game with the monetary value of question $(k - 1)$. Following Perea and Puerto (2007), 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,

$\varphi(p)$	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

Table 2: Optimal strategies for various φ functions.

4500, 9000, 18000, 36000, 72000, 144000 and 300000 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, she 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 Perea and Puerto (2007) 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, modelling 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 2. 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 the second while there are 14400 decision nodes and the depth 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 (2007) 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 , she directly gives the correct answer, else she has to make a decision. A small part of the decision tree for this new modelling is represented in Figure 4 (the dotted lines represent omitted parts of 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 (2007) to evaluate the different probabilities at the chance nodes. The whole decision tree has more than 75 millions decision nodes. The problem becomes therefore much harder since the number

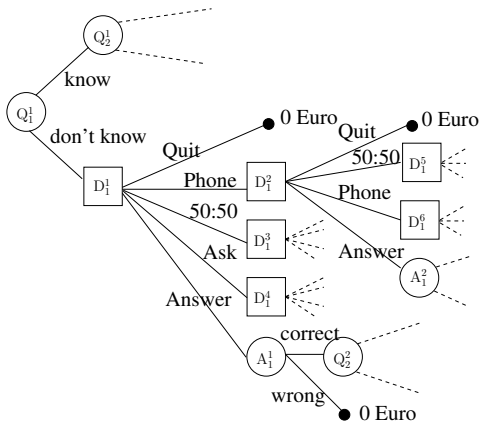


Figure 4: Refined decision tree for the TV game.

of potential strategies explodes. Unlike previous numerical tests, we had to use a computer with 64GB 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 resolution time increases with the concavity of the probability transformation function.

Conclusion

After showing the NP-hardness of the RDU-OPT problem in decision trees, we have provided an implicit enumeration algorithm to solve it. The upper bound used to prune the search is computed by dynamic programming, and is polynomial in the number of decision nodes. Note that this upper bound is actually adaptable to any decision criterion compatible with stochastic dominance. The tests performed show that the provided algorithm makes it possible to solve efficiently instances the number of decision nodes of which is near six thousands.

Concerning the representation formalism for planning under uncertainty, we would like to emphasize that the previous approach can be adapted to work in influence diagrams or finite horizon Markov decision processes. The standard solution methods to compute the optimal EU plan in these formalisms are indeed also based on dynamic programming. An approach similar to the one we developed here will therefore enable to compute an upper bound to prune the search in an enumeration tree.

Concerning the decision-theoretic aspect, we have focused here on the computation of an optimal strategy according to RDU among *pure* strategies. An interesting research direction for the future is to consider mixed strategies, i.e. strategies where one chooses randomly (according to a predefined probability distribution) the decision taken at each decision node. Indeed, this makes it possible to obtain a better RDU value than with pure strategies, as shown by the following example. 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 $\varphi(p) = 0.45$ for all $p \in (0; 1)$ (for simplicity). The RDU

value of the two pure strategies are respectively 5 and 5.05. Interestingly enough, 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.

Acknowledgments

This work has been supported by the ANR project PHAC which is gratefully acknowledged.

References

- Allais, M. 1953. Le comportement de l'homme rationnel devant le risque : critique des postulats de l'école américaine. *Econometrica* 21:503–546.
- Allais, M. 1997. An outline of my main contributions to economic science. *The American Economic Review* 87(6):3–12.
- Blythe, J. 1999. Decision-theoretic planning. *AI Mag.* 20.
- Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of AI Research* 11:1–94.
- Dean, T.; Kaelbling, L.; Kirman, J.; and Nicholson, A. 1993. Planning with deadlines in stochastic domains. In *Proc. of the 11th AAAI*, 574–579.
- Handa, J. 1977. Risk, probabilities and a new theory of cardinal utility. *Journal of Political Economics* 85:97–122.
- Jaffray, J.-Y., and Nielsen, T. 2006. An operational approach to rational decision making based on rank dependent utility. *European J. of Operational Research* 169(1):226–246.
- Kaelbling, L.; Littman, M.; and Cassandra, A. 1999. Planning and acting in partially observable stochastic domains. *Artificial Intelligence* 101:99–134.
- Kahneman, D., and Tversky, A. 1979. Prospect theory: An analysis of decision under risk. *Econometrica* 47:263–291.
- Liu, Y., and Koenig, S. 2008. An exact algorithm for solving mdps under risk-sensitive planning objectives with one-switch utility functions. In *Proc. of the Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS)*.
- McClellenn, E. 1990. *Rationality and Dynamic choice: Foundational Explorations*. Cambridge University Press.
- Morin, T. 1982. Monotonicity and the principle of optimality. *J. of Math. Analysis and Applications* 86:665–674.
- Perea, F., and Puerto, J. 2007. Dynamic programming analysis of the TV game “Who wants to be a millionaire?”. *European Journal of Operational Research* 183:805–811.
- Perny, P.; Spanjaard, O.; and Storme, L.-X. 2007. State space search for risk-averse agents. In *20th International Joint Conference on Artificial Intelligence*, 2353–2358.
- Quiggin, J. 1993. *Generalized Expected Utility Theory: The Rank-Dependent Model*. Kluwer.
- Raiffa, H. 1968. *Decision Analysis: Introductory Lectures on Choices under Uncertainty*. Addison-Wesley.
- Shachter, R. 1986. Evaluating influence diagrams. *Operations Research* 34:871–882.
- von Neuman, J., and Morgenstern, O. 1947. *Theory of games and economic behaviour*. Princeton University Press.