Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence

# Sequential Decision Making with Rank Dependent Utility: A Minimax Regret Approach

**Gildas Jeantet**
AiRPX
18 rue de la pépinière
75008 Paris, France
gildas.jeantet@airpx.com

**Patrice Perny**
LIP6-CNRS, UPMC
4 Place Jussieu
75252 Paris Cedex 05, France
patrice.perny@lip6.fr

**Olivier Spanjaard**
LIP6-CNRS, UPMC
4 Place Jussieu
75252 Paris Cedex 05, France
olivier.spanjaard@lip6.fr

**Abstract**

This paper is devoted to sequential decision making with Rank Dependent expected Utility (RDU). This decision criterion generalizes Expected Utility and enables to model a wider range of observed (rational) behaviors. In such a sequential decision setting, two conflicting objectives can be identified in the assessment of a strategy: maximizing the performance viewed from the initial state (optimality), and minimizing the incentive to deviate during implementation (deviation-proofness). In this paper, we propose a minimax regret approach taking these two aspects into account, and we provide a search procedure to determine an optimal strategy for this model. Numerical results are presented to show the interest of the proposed approach in terms of optimality, deviation-proofness and computability.

## Introduction

Decision making under uncertainty deals with situations where the consequences of an action depends on exogenous events. Such situations appear in several applications of artificial intelligence: medical diagnosis, troubleshooting under uncertainty (Breese and Heckerman 1996), designing bots for games (Maîtrepierre et al. 2008), etc. The standard way to handle uncertainty is to compare actions on the basis of *expected utility theory* (von Neumann and Morgenstern 1947). In this theory, a decision is viewed as a probability distribution over consequences – denoting $\mathcal{L}$ the set of probability distributions over the set $X$ of consequences, a decision is thus a *lottery* $L \in \mathcal{L}$. A lottery $L$ is then evaluated on the basis of its expected utility $EU(L) = \sum_{x \in X} L(x)u(x)$, where $L(x)$ is the probability of consequence $x$ and $u : X \to \mathbb{R}$ is a utility function that assigns a numerical value to each consequence.

Nevertheless, despite its intuitive appeal, expected utility has shown some limits to account for all rational decision behaviors. One of the most famous examples of such limits is due to Kahneman and Tversky (1979). This example is summarized in Table 1, where $X = \{\$0, \$3000, \$4000\}$ and each cell indicates probability $L(x)$.

Most of people prefer $L_1$ to $L'_1$ (preference for certainty) and $L'_2$ to $L_2$ (choosing $L'_2$ instead of $L_2$ almost amounts to exchange \$3000 for \$4000). By elementary calculus, one can show there exists no utility function $u$ such that $EU(L_1) > EU(L'_1)$ and $EU(L'_2) > EU(L_2)$. However, this preference reversal can be encompassed by a non-linear handling of probabilities. This had led researchers to generalize the definition of expected utility. One of the most popular generalizations of expected utility is the rank dependent expected utility (RDU) model (Quiggin 1993). In this model, a non-linear probability weighting function $\varphi$ is incorporated in the expectation calculus. Let us use a simple example to illustrate the principle. Consider lottery $L_2$ and assume that $u(x) = x$. The expected utility of $L_2$ can be reformulated as: $0 + 0.1 \times (3000 - 0) + 0 \times (4000 - 3000)$ (the utility of lottery $L_2$ is at least 0 with probability 1, then the utility might increase from 0 to 3000 with probability 0.1, and the utility cannot increase from 3000 to 4000). The rank dependent expected utility of $L_2$ is obtained from expected utility by inserting $\varphi$ as follows: $0 + \varphi(0.1) \times (3000 - 0) + \varphi(0) \times (4000 - 3000)$. By setting $\varphi(0.09) = \varphi(0.1) = 0.2$ and $\varphi(0.9) = 0.7$, the preference induced by RDU are then compatible with Kahneman and Tversky's example.

The topic of this paper is to study how to handle RDU in sequential decision making under uncertainty. A standard way of modeling a sequential decision problem under risk is to use *a decision tree* representing all decision steps and possible events. One does not make a simple decision but one follows a *strategy* (i.e. a sequence of decisions conditioned

| $x$ | \$0 | \$3000 | \$4000 |
|---|---|---|---|
| $L_1(x)$ | 0.00 | 1.00 | 0.00 |
| $L'_1(x)$ | 0.10 | 0.00 | 0.90 |
| $L_2(x)$ | 0.90 | 0.10 | 0.00 |
| $L'_2(x)$ | 0.91 | 0.00 | 0.09 |

Table 1: Kahneman and Tversky's example.

by events) resulting in a non deterministic outcome (a strategy is analogous to a *policy* in the literature dedicated to Markov decision processes). The set of feasible strategies is then combinatorial. As a consequence, the number of strategies exponentially increases with the size of the problem and computing an optimal strategy according to a given decision model requires an implicit enumeration procedure.

It is well-known that computing an optimal strategy according to EU in a decision tree can be performed in linear time by *rolling back* the decision tree, i.e. recursively computing the optimal EU value in each subtree by starting from the leaves of the decision tree. However, this approach is no longer valid when using RDU since Bellman's principle of optimality does not hold anymore. Actually, it has been shown that computing an optimal strategy according to RDU, viewed from the root of a decision tree, is an NP-hard problem (Jeantet and Spanjaard 2011). Furthermore, once a RDU-optimal strategy $s^*$ has been computed, it may be difficult to implement for a human decision maker because she may have an incentive to deviate after some decision steps. When entering a subtree the substrategy corresponding to $s^*$ may indeed be suboptimal, a consequence of the violation of the Bellman principle. This shows that the enhanced descriptive possibilities offered by RDU have drawbacks in terms of computational complexity and implementability.

These difficulties have led Jaffray and Nielsen (2006) to propose an operational approach eliminating some strategies whenever they appear to be largely suboptimal for RDU at some node. Their approach returns an RDU-optimal strategy (viewed from the root) among the remaining ones, such that no other strategy dominates it (w.r.t. stochastic dominance); unfortunately, by deliberately omitting a large subspace of strategies, this approach does not provide any information about the gap to RDU-optimality at the root nor at the subsequent decision steps (and therefore precludes any control on the incentive to deviate).

We propose here another implementation of RDU theory in decision trees. Our approach is based on the minimization of a weighted max regret criterion, where regrets measure, at all decision nodes, the opportunity losses (in terms of RDU) of keeping the strategy chosen at the root. Our aim in proposing this approach is to achieve a tradeoff between two possibly conflicting objectives: maximizing the quality of the strategy seen from the root (measured by the RDU value) and minimizing the incentive to deviate.

The paper is organized as follows: we first introduce the decision tree formalism and describe the main features of RDU. After recalling the main issues in using RDU in a sequential decision setting, we present the approach of Jaffray and Nielsen (2006), and we position our approach with respect to it. We then provide a solution procedure able to return a minimax regret strategy in a decision tree. Finally, we provide numerical tests on random instances to assess the operationality of the proposed approach.

## Decision Trees and RDU

The formalism of decision trees provides a simple and explicit representation of a sequential decision problem under uncertainty. A decision tree $\mathcal{T}$ involves three kinds of nodes:

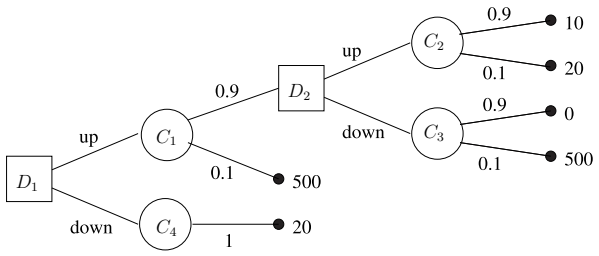


Figure 1: A decision tree representation.

a set $\mathcal{N}_D$ of decision nodes (represented by squares), a set $\mathcal{N}_C$ of chance nodes (represented by circles) and a set $\mathcal{N}_U$ of utility nodes (leaves of the tree). A decision node can be seen as a decision variable, the domain of which corresponds to the labels of the branches starting from that node. The branches starting from a chance node correspond to different possible events. These branches are labelled by the probabilities of the corresponding events. The values indicated at the leaves correspond to the *utilities* of the consequences. An example of decision tree is depicted in Figure 1.

A *strategy* consists in making a decision at every decision node. The decision tree in Figure 1 includes 3 feasible strategies: $(D_1 = up, D_2 = up)$, $(D_1 = up, D_2 = down)$ and $(D_1 = down)$ (note that node $D_2$ cannot be reached if $D_1 = down$). For convenience, a strategy can also be viewed as a set of edges, e.g. strategy $s = (D_1 = up, D_2 = up)$ can also be denoted by $\{(D_1, C_1), (D_2, C_2)\}$. A strategy is associated to a lottery over the utilities. For instance, strategy $s$ corresponds to lottery $(0.81, 10; 0.09, 20; 0.1, 500)$. More generally, to any strategy is associated a lottery $L = (p_1, u_1; \ldots; p_k, u_k)$ that yields utility $u_i$ with probability $p_i = P(\{u_i\})$, where $u_1 < \ldots < u_k$ represent the utilities at the leaves of the tree and $P$ is a probability distribution over $U = \{u_1, \ldots, u_k\}$. Comparing strategies amounts therefore to comparing lotteries. A useful notion to compare lotteries is the decumulative function $G_L$ given by $G_L(x) = \sum_{i:u_i \geq x} p_i$. For the sake of simplicity, we will consider a lottery $L$ as a function from $U$ to $[0, 1]$ such that $L(u_i) = p_i$. The set of lotteries is endowed with the usual mixture operation $pL_1 + (1-p)L_2$ ($p \in [0, 1]$), that defines, for any pair $L_1, L_2$, a lottery $L$ characterized by $L(u) = pL_1(u) + (1-p)L_2(u)$.

Using these notations, the rank dependent utility (Quiggin 1993) writes as follows:

$$RDU(L) = u_1 + \sum_{i=2}^{k} [u_i - u_{i-1}]\, \varphi\left(G_L\left(u_i\right)\right)$$

where $\varphi$ is a non-decreasing probability transformation function, proper to any agent, such that $\varphi(0)$=0 and $\varphi(1)$=1. The main interest of distorting cumulative probabilities, rather than probabilities themselves (as in Handa's model, 1977), is to get a choice criterion compatible with stochastic dominance. A lottery $L = (p_1, u_1; \ldots; p_k, u_k)$ is said to stochastically dominate a lottery $L' = (p'_1, u'_1; \ldots; p'_k, u'_k)$ if for all $x \in \mathbb{R}$, $G_L(x) \geq G_{L'}(x)$.

Coming back to strategy $s = (D_1 = up, D_2 = up)$ in Figure 1, the RDU value of $s$ is denoted by RDU$(s)$ and is

equal to $\text{RDU}(0.81, 10; 0.09, 20; 0.1, 500) = 10 + (20 - 10)\varphi(0.19) + (500 - 20)\varphi(0.1)$. If $\varphi(p) = p$ for all $p$, RDU clearly reduces to EU. For $\varphi(p) \neq p$, the probability weighting function $\varphi$ makes it possible to distinguish between weak risk aversion (i.e., if an option yields a guaranteed outcome, it is preferred to any other risky option with the same expectation) and strong risk aversion (i.e., if two lotteries have the same expectation, then the agent prefers the lottery with the minimum spread of possible outcomes). Within the RDU model, for a concave utility function $u$, an agent is weakly risk-averse iff $\varphi(p) \leq p$ for all $p \in [0, 1]$, and strongly risk-averse iff $\varphi$ is convex. Note that strong risk aversion implies weak risk aversion. Throughout the paper, we will assume that the decision maker is at least weakly risk-averse.

## Rolling Back Procedures

Before discussing the computational aspects of optimizing RDU in a decision tree, we first recall the standard procedure to get an EU-optimal strategy. It is well-known that rolling back the decision tree makes it possible to compute in linear time such an optimal strategy. Indeed, an EU-optimal 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 as indicated in Algorithm 1, where $V \equiv EU$ and $\pi_N$ is the expected utility computed at node $N$. The notations are the following: the children of node $N$ are denoted by $\Gamma(N)$, the utility of a node $N \in \mathcal{N}_U$ is denoted by $u(N)$, the probability on edge $(N, N')$ is denoted by $p(N, N')$ (with $N \in \mathcal{N}_C$), and $L_\pi(N)$ denotes the lottery characterized by $L(\pi_{N'}) = p(N, N')$ for $N' \in \Gamma(N)$.

Note that the rolling back procedure is no longer valid when using RDU. Due to the non-linearity of RDU w.r.t. the mixture operation defined above, the RDU value of a strategy cannot be inferred from the RDU values of its substrategies. Thus, in the case of $V \equiv$ RDU, the value returned by Algorithm 1 does not correspond in general to the RDU value of any strategy. This difficulty can be bypassed by rolling back lotteries instead of RDU values. This leads to a variant summarized in Algorithm 2, where $V \equiv$ RDU and $L_N$ is the lottery computed at node $N$. To illustrate how this variant operates, consider the decision tree of Figure 1 and assume that function $\varphi$ is defined by $\varphi(p) = 0$ if $p \leq 0.1$ and $\varphi(p) = p$ otherwise (note that $\varphi(p) \leq p$ and therefore the decision maker is risk-averse). Node $D_2$ is labelled by lottery $(0.9, 10; 0.1, 20)$, corresponding to substrategy $(D_2 = up)$, since $\text{RDU}(0.9, 10; 0.1, 20) = 10 > 0 = \text{RDU}(0.9, 0; 0.1, 500)$. Consequently, node $C_1$ is labelled by lottery $(0.81, 10; 0.09, 20; 0.1, 500)$. Finally, node $D_1$ is labelled by $(1, 20)$, corresponding to strategy $(D_1 = down)$, since $\text{RDU}(1, 20) = 20 > 11.9 = \text{RDU}(0.81, 10; 0.09, 20; 0.1, 500)$. The RDU value returned by the rolling back method corresponds thus to strategy $(D_1 = down)$. However, the optimal strategy is $(D_1 = up, D_2 = down)$ with $\text{RDU}(0.81, 0; 0.19, 500) = 95$. This shows that Algorithm 2 does not provide the optimal RDU value in general (which is not surprising since the problem is NP-hard, as indicated in the introduction).

**Algorithm 1**: Rolling back values.

**Input**: Decision tree $\mathcal{T}$
**for each** node $N$ in $\mathcal{T}$ from the leaves to the root **do**
  **case** $N \in \mathcal{N}_U$ : $\pi_N \leftarrow u(N)$
  **case** $N \in \mathcal{N}_D$ : $\pi_N \leftarrow \max_{N' \in \Gamma(N)} \pi_{N'}$
  **case** $N \in \mathcal{N}_C$ : $\pi_N \leftarrow V(L_\pi(N))$
**end**
**return** $\pi_{root}$

**Algorithm 2**: Rolling back lotteries.

**Input**: Decision tree $\mathcal{T}$
**for each** node $N$ in $\mathcal{T}$ from the leaves to the root **do**
  **case** $N \in \mathcal{N}_U$ : $L_N \leftarrow (1, u(N))$
  **case** $N \in \mathcal{N}_D$ : $L_N \leftarrow \arg \max_{N' \in \Gamma(N)} V(L_{N'})$
  **case** $N \in \mathcal{N}_C$ : $L_N \leftarrow \sum_{N' \in \Gamma(N)} p(N, N') L_{N'}$
**end**
**return** $V(L_{root})$

Seidenfeld (1988) justifies the choice of such a suboptimal strategy (the one returned by Algorithm 2) by claiming that a decision maker should adopt the following principle of *dynamic feasibility*: to assess a strategy at a decision node, anticipate how you will choose at the (potential) "future" decision nodes (by locally optimizing RDU) and declare infeasible all strategies incompatible with these choices. This strategy will be followed by a *sophisticated* decision maker, i.e. a decision maker who is able to anticipate her future actions, and who also adopts a *consequentialist* behavior, i.e. her decisions do not depend on the past nor on *counterfactual* events (events that could have occured but did not). Though appealing from an algorithmic viewpoint, one easily shows that Algorithm 2 can return a *stochastically dominated* strategy (Jaffray and Nielsen 2006). A strategy $s$ is said to be stochastically dominated if there exists another strategy $s'$ such that $L'$ stochastically dominates $L$, where $L$ (resp. $L'$) is the lottery corresponding to $s$ (resp. $s'$). Following a stochastically non-dominated strategy is obviously desirable from the normative viewpoint. Furthermore, as argued by Machina (1989), it is inappropriate to impose consequentialism on non-EU behavior since the very notion of a non-EU criterion (as RDU) implies a type of non-separability which contradicts the consequentialist assumption. For this reason, it seems reasonable to renounce consequentialism, and therefore Algorithm 2, when using a non-EU criterion. We assume here that, instead of following the dynamic feasibility principle, the (sophisticated) decision maker adopts a *resolute choice* behavior (McClennen 1990), i.e. she initially commits to a strategy and never deviates from it later.

## Resolute Choice Approaches

Two resolute choice approaches for using RDU in a decision tree have been studied in the literature.

The first approach is *resolute choice with root dictatorship*, that consists in determining an optimal strategy according to RDU viewed from the root of the decision tree. A branch and bound procedure to determine such a strategy has been proposed by Jeantet and Spanjaard (2011). Nevertheless, note that an RDU-optimal strategy can include poor substrategies, which can be an incentive for the decision maker to deviate from the predefined course of action. For instance, in the decision tree of Figure 1, consider a decision maker that wishes to follow the RDU-optimal strategy $(D_1 = up, D_2 = down)$. Assume that the decision maker reaches node $D_2$ at the second decision step, then the choice set is $(D_2 = up)$ with $\text{RDU}(D_2 = up) = 10$ or $(D_2 = down)$ with $\text{RDU}(D_2 = down) = 0$. Clearly, the decision maker has here an incentive to choose $(D_2 = up)$ and deviate from her initial resolutions.

The second approach is *resolute choice with selves*, that consists in considering each decision node as a *self* who represents the decision maker at the time and state when the decision is made (Jaffray and Nielsen 2006). One then aims at determining a strategy achieving a compromise between the different selves of a sophisticated decision maker, i.e. a strategy that remains suitable for all selves. This is the approach we follow here. The originality of our approach stems from the fact that one focuses on determining a *minimax regret* strategy between the selves, where the regret of a self for a given strategy $s$ is defined as the difference between the RDU value of $s$ at this node and the optimal RDU value in the subtree. This significantly differs from the approach of Jaffray and Nielsen where the compromise strategy is defined in a procedural manner. Their proposition can be interpreted as relaxing the notion of *dynamic feasibility* as follows: to assess a strategy at a decision node, determine a set of admissible strategies at the (potential) future decision nodes (where admissibility means they are both stochastically non-dominated and near optimal for RDU) and declare infeasible all strategies that are incompatible with these admissible strategies. The idea is to identify a subset of such feasible strategies at the root, and then to select among them the one maximizing RDU. Thus, each self, when making her decision, should have little incentive to deviate from the predefined strategy.

Algorithm 3 summarizes Jaffray and Nielsen's procedure, where the subset of feasible strategies at node $N$ is denoted by $\mathcal{S}_N$, and the cartesian product operator is denoted by $\Pi$. Parameter $\theta$ represents the maximal gap to optimality admissible for a strategy at a given decision node. Parameter $k$ is technical: if the number of feasible strategies becomes higher than $k$, then only the $k$ best ones w.r.t. RDU are kept. This prevents a combinatorial growth of the number of inspected strategies. It seems however difficult to select *a priori* proper values for parameter $\theta$ and $k$. Furthermore, the impact of these parameters on the quality of the solution is not obvious: even when $k$ is arbitrary large, the procedure does not guarantee a gap to optimality lower than $\theta$ (contrary to intuition). Actually, due to parameter $k$, it seems difficult to formulate a decision criterion characterizing to what extent the returned strategy can be seen as optimal.

**Algorithm 3**: Jaffray and Nielsen's procedure.

**Input**: Decision tree $\mathcal{T}$, real $\theta$, integer $k$
**for each** node $N \in \mathcal{N}_U$ **do** $\mathcal{S}_N \leftarrow \{(1, u(N))\}$
**for each** node $N$ in $\mathcal{T}$ **do**
  **case** $N \in \mathcal{N}_D$ :
    $\mathcal{S}_N \leftarrow \bigcup_{N' \in \Gamma(N)} \Big\{\{(N, N')\} \cup s : s \in \mathcal{S}_{N'}\Big\}$
    **for each** $s \in \mathcal{S}_N$ **do** $V_s \leftarrow \text{RDU}(s)$
    $V_{max} \leftarrow \max_{s \in \mathcal{S}_N} \{V_s\}$
    **for each** $s \in \mathcal{S}_N$ **do**
      **if** [$s$ is stochastically dominated] **or** [$V_s < V_{max} - \theta$] **then** $\mathcal{S}_N \leftarrow \mathcal{S}_N \setminus \{s\}$
    **end**
    **while** $|\mathcal{S}_N| > k$ **do**
      $\mathcal{S}_N \leftarrow \mathcal{S}_N \setminus \big\{\arg\min_{s \in \mathcal{S}_N} \{V_s\}\big\}$
  **end**
  **case** $N \in \mathcal{N}_C$ : $\mathcal{S}_N \leftarrow \prod_{N' \in \Gamma(N)} \mathcal{S}_{N'}$
**end**
**return** $\big\{\arg\max_{s \in \mathcal{S}_{root}} \{V_s\}\big\}$

## A Minimax Regret Approach

The resolute choice approach we propose is based on the optimization of a minimax regret criterion between the selves. For a strategy $s$ involving decision node $N$, the regret $r_N(s)$ of the self at node $N$ is defined by $r_N(s) = \text{RDU}^*(N) - \text{RDU}(s_N)$, where $s_N$ is the substrategy of $s$ at node $N$ and $\text{RDU}^*(N)$ is the RDU value of the optimal strategy at node $N$. Hence, the regret of a strategy $s$ is defined by $\overline{r}(s) = \max_{N \in s} \lambda_N r_N(s)$, where $\lambda_N$ is a weight assigned to the self at *decision* node $N$ and $N \in s$ denotes the fact that $N$ can be reached when following strategy $s$. This evaluation represents the maximum regret over decision nodes that can be reached by $s$. Then, our goal is to determine a strategy $s$ such that:

$$\overline{r}(s) = \min_{s' \in \mathcal{S}^\star_{root}} \overline{r}(s')$$

where $\mathcal{S}^\star_{root}$ denotes the set of stochastically non-dominated strategies at the root. The weights $\lambda_N$ allow greater flexibility, and can be interpreted in several ways, such as normalization factors or weighting coefficients, for instance:

• if $\lambda_N = 1$ for all $N$, then all selves are considered equivalently and this is the usual egalitarian view.

• if $\lambda_N$ is the probability to reach node $N$, i.e. is equal to the product of the probabilities along the path from the root to $N$ if $N \in s$, and to 0 otherwise, then the importance given to a self is proportional to the probability that the corresponding decision situation occurs.

• if $\lambda_{root} = \alpha$ and $\lambda_N = 1 - \alpha$ for $N \neq root$, then the value of a strategy $s$ can be reformulated as $\overline{r}(s) = \max\{\alpha r_{root}, (1 - \alpha) \max_{N \in s} r_N\}$. Parameter $\alpha$ can be interpreted as representing a tradeoff between optimality at the root (term $r_{root}$) and deviation-proofness of the strategy (term $\max_{N \in s} r_N$). We mean here *deviation-proofness* in the sense that the incentive to deviate is low for a self.

For illustration, consider strategy $s' = (D_1 = up, D_2 = down)$ in Figure 1. One has $\text{RDU}(s_{D_1}) = 95$, $\text{RDU}(s_{D_2}) = 0$, $\text{RDU}^*(D_1) = 95$ and $\text{RDU}^*(D_2) = 10$. Hence, with weights equal to 1, one gets $\overline{r}(s') = \max\{95 - 95, 10 - 0\} = 10$. Finally note that the determination of a weighted minimax regret strategy is NP-hard: if $\lambda_{root} = 1$ and $\lambda_N = 0$ for $N \neq root$, the problem amounts indeed to determine an RDU-optimal strategy at the root.

## A New Algorithm

The solution algorithm we propose is composed of two phases, the first phase aiming at computing optimal RDU values in all subtrees (these values are required to evaluate the regrets), and the second phase aiming at computing a minimax regret strategy:

*Phase 1. Computation of values RDU*$^*(N)$. This computation is performed via a branch and bound procedure. This branch and bound is launched for every subtree $\mathcal{T}(N)$, where $\mathcal{T}(N)$ is the subtree rooted in node $N \in \mathcal{N}_D$.

*Phase 2. Computation of a minimax regret strategy.* This computation is also based on a branch and bound procedure. Actually, both branch and bound procedures coincide in their branching part. The bounding part for computing a minimax regret strategy is explained in the sequel. This branch and bound is launched only once.

**Phase 1.** To compute values $\text{RDU}^*(N)$, the branch and bound we use only differs in the bounding part from the one proposed by Jeantet and Spanjaard (2011). One takes advantage of the specific shape of function $\varphi$ induced by risk aversion to compute tighter upper bounds on $\text{RDU}^*(N)$. We have the following (the proofs are omitted to save space):

**Proposition 1** *Let $EU^*(N)$ denote optimal expected utility at node $N$. If $\varphi(p) \leq p \; \forall p$, then RDU$^*(N) \leq EU^*(N)$.*

**Proposition 2** *Let $\overline{RDU}(N)$ denote value $\pi_N$ obtained at node $N$ by applying Algorithm 1 with $V \equiv$ RDU. If $\varphi$ is convex, then RDU$^*(N) \leq \overline{RDU}(N)$.*

Proposition 1 can be used to compute an upper bound as soon as the decision maker is weakly risk-averse. This upper bound can be strengthened if the decision maker is strongly risk-averse, by using also Proposition 2. For this reason, in case of weak risk-aversion, one applies Algorithm 1 once, with $V \equiv EU$, to obtain value $EU^*(N)$ at each node $N$. In case of strong risk-aversion, by applying a second time Algorithm 1 with $V \equiv$ RDU, the values at nodes $N$ become $\min\{EU^*(N), \overline{RDU}(N)\}$, from which one can compute an upper bound at least as good as the previous one.

**Phase 2.** We first describe the branching part of the branch and bound used in Phase 2. The 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*. Consider a decision tree $\mathcal{T}$ and a set of nodes $\mathcal{N}^s$ including the root of $\mathcal{T}$ and one and only one successor for every decision node $N \in \mathcal{N}^s_D = \mathcal{N}_D \cap \mathcal{N}^s$. The set of edges $s = \{(N, N') : N \in \mathcal{N}^s_D, N' \in \mathcal{N}^s\}$ defines a *partial strategy* of $\mathcal{T}$ if the subgraph induced by $\mathcal{N}^s$ is a tree. A complete strategy $s'$ is said to be *compatible* with a partial strategy $s$ if $s \subseteq s'$. The subset of strategies characterized by a partial strategy corresponds to the set of compatible strategies. A node of the enumeration tree is characterized by a partial strategy, and branching consists in choosing an edge among the ones starting from a given *candidate* decision node (one of the next future decision nodes compatible with the partial strategy). For illustration, the whole enumeration tree obtained for the decision tree of Figure 1 is given in Figure 2.

The aim of the branch and bound is of course to explore only a small portion of the enumeration tree. This goal is pursued by two means: a priority rule guiding the order in which the nodes are expanded, and bounding procedures enabling to prune subtrees (of the enumeration tree). The priorities in the branch and bound are denoted by $priority(N)$, the smaller the better. Algorithm 4 describes formally the branch and bound procedure we propose. For simplicity, we present a version that returns the *value* of an optimal strategy. The optimal strategy itself could of course be returned by using standard bookkeeping techniques. The branch and bound takes as arguments a partial strategy $s$ and the minimax regret value found so far, denoted by $best$. Function $bound(s)$ (resp. $heuristic(s)$) returns a lower bound (resp. upper bound) on the value of a minimax regret strategy compatible with $s$.

• $bound(s)$: the lower bounding procedure is based on the fact that the max regret over the decision nodes already instanciated in a partial strategy $s$ is a lower bound of the max regret of any completion of $s$. The formal procedure is described in Algorithm 5.

• $heuristic(s)$: in order to generate a good feasible strategy at each node of the branch and bound, one uses Algorithm 6. It consists in completing the partial strategy $s$ with optimal substrategies according to EU, and then computing, in each decision node $N$, the value of regret $r_N(s)$.

In order to ensure that the returned strategy is stochastically non-dominated (SND), one tests whether the new found strategy is SND before updating the incumbent. Since the initial incumbent is also SND (any EU-optimal strategy is SND), at any time the incumbent is SND, and therefore the returned strategy as well. Testing whether a strategy is SND amounts to solving a linear program (Jaffray and Nielsen 2006). The whole procedure is summarized in Algorithm 7.

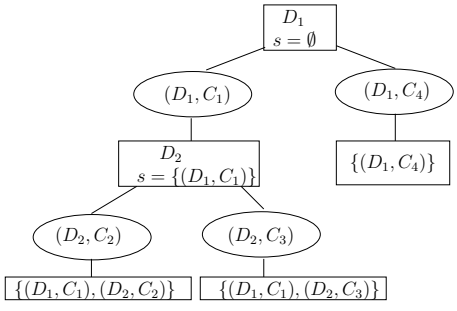


Figure 2: Enumeration tree of strategies in Figure 1.

**Algorithm 4**: $\mathbf{BB}(s, best)$

$\mathcal{N}_1 \leftarrow \{N_1 \in \mathcal{N}_D : N_1$ is candidate in $\mathcal{T}\}$;
$N_{min} \leftarrow \arg\min_{N \in \mathcal{N}_1} priority(N)$;
$\mathcal{E}_{min} \leftarrow \{(N_{min}, N') : N' \in \Gamma(N_{min})\}$;
**for each** $(N, N') \in \mathcal{E}_{min}$ **do**
  $best \leftarrow \min\{best, heuristic(s \cup \{(N, N')\})\}$;
  **if** $bound(s \cup \{(N, N')\}) < best$ **then**
    $temp \leftarrow \mathbf{BB}(s \cup \{(N, N')\}, best)$;
    **if** $temp < best$ **then** $best \leftarrow temp$;
  **end**
**end**
**return** $best$

**Algorithm 5**: $bound(s)$

**for each** node $N \in \mathcal{N}_D$ **do**
  **if** $\exists (N, N') \in s$ **then**
    $r_N \leftarrow RDU^*(N) - RDU^*(N')$;
  **else**
    $r_N \leftarrow 0$;
  **endif**
**end**
**return** $\max_{N \in \mathcal{N}_D} \lambda_N r_N$

**Algorithm 6**: $heuristic(s, V)$

**for each** node $N$ in $\mathcal{T}$ from the leaves to the root **do**
  **case** $N \in \mathcal{N}_U$: $L_N \leftarrow (1, u(N))$;
  **case** $N \in \mathcal{N}_C$: $L_N \leftarrow \sum_{N' \in \Gamma(N)} p(N, N') L_{N'}$;
  **case** $N \in \mathcal{N}_D$:
    **if** $\exists (N, N') \in s$ **then**
      $r_N \leftarrow RDU^*(N) - RDU(L_{N'})$;
    **else**
      $N' \leftarrow \arg\max_{N' \in \Gamma(N)} EU^*(N')$;
      $r_N \leftarrow RDU^*(N) - RDU(L_{N'})$;
    **endif**
  **end**
**end**
**return** $\max_{N \in \mathcal{N}_D} \lambda_N r_N$

**Algorithm 7**: Minimax Regret($\mathcal{T}$)

**compute** all $\mathrm{EU}^*(N)$ by Alg. 1 **with** $V \equiv \mathrm{EU}$;
**compute** all $\overline{RDU}(N)$ by Alg. 1 **with** $V \equiv \mathrm{RDU}$;
**for each** node $N$ in $\mathcal{T}$ **do compute** $\mathrm{RDU}^*(N)$;
$s^*_{EU} \leftarrow$ an EU-optimal strategy at the root;
$best \leftarrow \mathbf{BB}(\emptyset, \overline{r}(s^*_{EU}))$;
**return** $best$

## Numerical tests

Our algorithm, as well as Jaffray and Nielsen's procedure, have been implemented in C++. Solver CPLEX has been used to solve the linear programs involved in the non-dominance tests. The numerical tests were performed on a Pentium IV 2.13GHz CPU computer with 3GB of RAM. We consider complete binary decision trees where the nodes of even (resp. odd) depth are decision (chance) ones, and the leaves are utility nodes. The utilities are integers randomly drawn in [0,100). All probabilities have been randomly generated in (0,1). Concerning the decision model, function $\varphi$ is defined by $\varphi(p) = p^2$ in RDU, and all weights $\lambda_N$ are equal to 1 in the definition of the max regret.

| | | | Algorithm 3 | | |
|---|---|---|---|---|---|
| # Nodes | $k$ | $\theta$ | Time (sec.) | $\overline{r}(s)/r^*$ | $\mathrm{RDU}(s)/\mathrm{RDU}^*$ |
| 127 | 1 | 1 | $< 1$ | 1.1 | 0.98 |
| 127 | 1 | 5 | $< 1$ | 1.2 | 0.99 |
| 127 | 1 | 20 | $< 1$ | 2.4 | 0.99 |
| 127 | 5 | 1 | $< 1$ | 1.1 | 0.99 |
| 127 | 5 | 5 | $< 1$ | 1.4 | 0.99 |
| 127 | 5 | 20 | $< 1$ | 23.6 | 0.99 |
| 127 | 10 | 1 | $< 1$ | 1.1 | 0.99 |
| 127 | 10 | 5 | $< 1$ | 1.1 | 0.99 |
| 127 | 10 | 20 | $< 1$ | 35.5 | 0.99 |
| 511 | 1 | 1 | $< 1$ | 1.1 | 0.98 |
| 511 | 1 | 5 | $< 1$ | 1.7 | 0.99 |
| 511 | 1 | 20 | $< 1$ | 7.9 | 0.99 |
| 511 | 5 | 1 | $< 1$ | 1.2 | 0.99 |
| 511 | 5 | 5 | $< 1$ | 1.7 | 0.99 |
| 511 | 5 | 20 | 1.3 | 187.2 | 0.99 |
| 511 | 10 | 1 | $< 1$ | 1.1 | 0.98 |
| 511 | 10 | 5 | $< 1$ | 1.9 | 0.99 |
| 511 | 10 | 20 | 1.2 | 121.3 | 0.99 |
| 2047 | 1 | 1 | 124.4 | 1.1 | 0.97 |
| 2047 | 1 | 5 | 122.0 | 1.7 | 0.98 |
| 2047 | 1 | 20 | 138.7 | 1.5 | 0.99 |
| 2047 | 5 | 1 | 109.2 | 1.8 | 0.97 |
| 2047 | 5 | 5 | 144.1 | 13.1 | 0.97 |
| 2047 | 5 | 20 | 160.2 | 98.4 | 0.99 |
| 2047 | 10 | 1 | 156.5 | 1.6 | 0.97 |
| 2047 | 10 | 5 | 174.7 | 4.4 | 0.98 |
| 2047 | 10 | 20 | 184.9 | 316.6 | 0.99 |

Table 2: Performances of Algorithm 3.

Table 2 summarizes the performances of Jaffray and Nielsen's procedure (Algorithm 3). Parameter $k$ takes value in $\{1, 5, 10\}$ and parameter $\theta$ takes value in $\{1, 5, 20\}$. The execution times have been reported in the fourth column. The fifth column indicates value $\overline{r}(s)/r^*$, where $\overline{r}(s)$ is the maximum regret of strategy $s$ returned by Algorithm 3, and $r^*$ is the minimax regret (value obtained by our approach). In the sixth column, we indicate ratio $RDU(s)/RDU^*$, where $s$ denotes the strategy returned by Algorithm 3 and $RDU^*$ is the optimal RDU value.

As can be seen in Table 2 (sixth column), Algorithm 3 generally provides RDU values at the root close to the optimum. However, the performance concerning the dynamic feasibility (represented by ratio $\overline{r}(s)/r^*$ to be minimized) is less convincing. Although ratio $\overline{r}(s)/r^*$ is satisfactory in most cases, we observe a significant number of cases where the ratio becomes very large (up to 300), which proves that the algorithm does not provide any guarantee on the max regret. Note that a self with a high regret has a significant incentive to deviate, thus threatening the complete implementation of the strategy. One could be tempted to decrease the value of $\theta$ to overcome this difficulty, but a lower RDU value is to be expected at the root. It may even happen that Algorithm 3 returns no strategy, if all considered strategies are stochastically dominated (Jaffray and Nielsen 2006).

Table 3 indicates the performances of our approach (Al-

| | Phase 1 | Phase 2 | | |
|---|---|---|---|---|
| # Nodes | Time (sec.) | Time (sec.) | $\overline{r}(s)/r^*$ | RDU$(s)$/RDU$^*$ |
| 127 | $< 1$ | $< 1$ | 1 | 0.99 |
| 511 | 1.1 | $< 1$ | 1 | 0.98 |
| 2047 | 2.6 | $< 1$ | 1 | 0.96 |
| 8191 | 14.8 | 1.2 | 1 | 0.96 |

Table 3: Performances of Algorithm 7.

gorithm 7), based on minimax regret optimization. The second column indicates the execution time of phase 1. The third column indicates the execution time of phase 2. As in the previous table, ratios $\overline{r}(s)/r^*$ and $RDU(s)/RDU^*$ are indicated (fourth and fifth column). Obviously, the major strength of Algorithm 7 is its ability to control the incentive to deviate by optimally minimizing the max regret. As a consequence, the performance is uniformly 1 in the third column. Although RDU optimality at the root was not the primary objective, a nice feature empirically attested by Table 3 is the good quality of the returned strategies viewed from the root (never less than 96% of the optimum). Finally, our approach proves significantly faster than Algorithm 3: decision trees with up to 8191 nodes can be solved within 15 seconds, while Algorithm 3 needs 10 times more to solve instances with 2047 nodes. This can be explained by the fact that the stochastic dominance tests are much more numerous in Algorithm 3 than in our approach.

## Acknowledgments

This paper is dedicated to the memory of our colleague and friend Jean-Yves Jaffray, with whom we have had very stimulating discussions that remain today a constant source of inspiration. We also wish to thank the anonymous reviewers for their fruitful comments on an earlier version of the paper.

## References

J. Breese and D. Heckerman (1996). Decision-theoretic troubleshooting: A framework for repair and experiment. In P. Besnard and S. Hanks (eds.), *Proc. of the Twelfth Annual Conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers, 124–136.

J. Handa (1977). Risk, probabilities and a new theory of cardinal utility. *Journal of Political Economics* **85**:97–122.

R. Howard and J. Matheson (1984). *Influence Diagrams*. Menlo Park CA: Strategic Decisions Group.

J.-Y. Jaffray and T. Nielsen (2006). An operational approach to rational decision making based on rank dependent utility. *European J. of Operational Research* **169**(1):226–246.

G. Jeantet and O. Spanjaard (2011). Computing Rank Dependent Utility in Graphical Models. In *Artificial Intelligence Journal* **175**:1366–1389.

D. Kahneman and A. Tversky (1979). Prospect theory: An analysis of decision under risk. *Econometrica* **47**:263–291.

M.J. Machina (1989). Dynamic consistency and non-expected utility models of choice under uncertainty. *Journal of Economic Literature* **27**:1622–1688.

R. Maîtrepierre, J. Mary, and R. Munos (2008). Adaptive play in texas hold'em poker. In *ECAI*. 458–462.

E. McClennen (1990). *Rationality and Dynamic choice: Foundational Explorations*. Cambridge University Press.

J. Quiggin (1993). *Generalized Expected Utility Theory: The Rank-Dependent Model*. Kluwer.

T. Seidenfeld (1988). Decision theory without independence or without ordering. *Economics and Philosophy* **4**(2):267–290.

J. von Neumann and O. Morgenstern (1947). *Theory of games and economic behaviour*. Princeton, NJ. Princeton University Press.