Adaptive Elicitation of Preferences under Uncertainty in Sequential Decision Making Problems

Nawal Benabbou, Patrice Perny

Adaptive Elicitation of Preferences under Uncertainty in Sequential Decision Making Problems

Nawal Benabbou and **Patrice Perny**
Sorbonne Universités, UPMC Univ Paris 06, UMR 7606, LIP6
CNRS, UMR 7606, LIP6, F-75005, Paris, France
4 Place Jussieu, 75005 Paris, France
nawal.benabbou@lip6.fr, patrice.perny@lip6.fr

Abstract

This paper aims to introduce an adaptive preference elicitation method for interactive decision support in sequential decision problems. The Decision Maker's preferences are assumed to be representable by an additive utility, initially unknown or imperfectly known. We first study the determination of possibly optimal policies when admissible utilities are imprecisely defined by some linear constraints derived from observed preferences. Then, we introduce a new approach interleaving elicitation of utilities and backward induction to incrementally determine a near-optimal policy. We propose an interactive algorithm with performance guarantees and describe numerical tests demonstrating the practical efficiency of our approach.

1 Introduction

Automated preference elicitation and preference-based search are two important lines of research in AI for computational decision support, see e.g., [Boutilier, 2013]. Methods for automated preference elicitation are generally grounded on decision theory and consist in learning the parameters of a decision model to best fit the preferences of the Decision Maker (DM), e.g., [Chajewska *et al.*, 2001; Boutilier *et al.*, 2006; Fürnkranz and Hüllermeier, 2010; Perny *et al.*, 2016]. The resulting models are then involved in preference-based search algorithms, to explore all possible solutions and determine an optimal choice, see e.g., [Dasgupta *et al.*, 1995; Perny and Spanjaard, 2002; Boutilier *et al.*, 2004; Brafman *et al.*, 2010]. Thus, preference elicitation is often seen as a preliminary stage to preference-based search.

Yet, there is an alternative view which is being actively developed in AI, that consists in interleaving elicitation and search. The aim is to focus the elicitation burden on the useful part of preference information to solve a specific instance of a decision problem, rather than trying to obtain a full picture of the DM's preferences. This approach relies on an adaptive generation of preference queries in order to progressively reduce the indetermination attached to the parameters of the decision model until an optimal or near optimal solution can be identified. This incremental approach has proved successful for decision making on explicit sets in various contexts, e.g., for the elicitation of utility functions, see [White III *et al.*, 1984; Ha and Haddawy, 1997; Chajewska *et al.*, 2000; Wang and Boutilier, 2003; Braziunas and Boutilier, 2007; Hines and Larson, 2010].

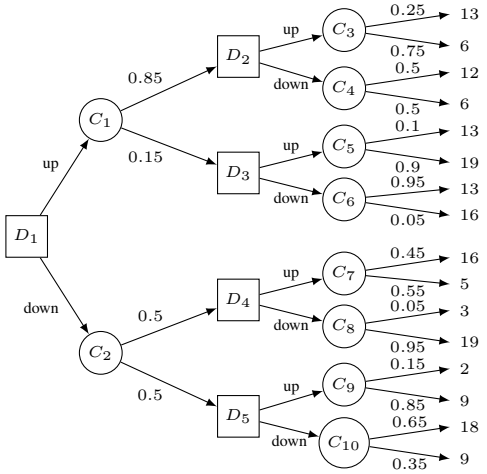
However, the extension of adaptive preference elicitation to combinatorial domains is generally not straightforward. The determination of informative preference queries and the progressive elimination of solutions are indeed harder to implement when the alternatives are very numerous and implicitly defined. Yet, some recent contributions propose incremental elicitation procedures for some specific combinatorial optimization problems considered in AI such as constraint satisfaction [Gelain *et al.*, 2010], matching [Drummond and Boutilier, 2013], planning [Weng and Zanuttini, 2013], and state space search [Benabbou and Perny, 2015a]. The aim of this paper is propose an incremental approach for another important problem, namely the determination of an optimal policy in a sequential decision problem under risk.

This problem appears in various contexts such as strategic resources allocation, investment management, optimization of medical treatment policies, or navigation problems. In these problems, DMs have to identify their preferred policy among a combinatorial set of possibilities, in a setting where the consequences of their acts depend on exogenous events. In such situations, decision support tools may facilitate the elicitation of their preferences and control the selection of actions in different possible states by taking into account anticipated future situations, possibly as a result of their own acts.

We consider a sequential decision problem under uncertainty represented by a decision tree T, i.e., an acyclic and connected graph including three types of nodes: a set $\mathcal{D}_T$ of decision nodes (represented by squares), a set $\mathcal{C}_T$ of chance nodes (represented by circles) and a set $\mathcal{X}_T$ of outcome nodes (leaves). Each decision node represents a decision variable and the branches starting from that node correspond to the possible decisions (the domain of the variable). The branches starting from a chance node correspond to different possible events and are labelled by their probabilities. The leaves represent the possible outcomes (e.g., profits, time). An example of decision tree is given in Figure 1.

The root of the tree is a decision node representing the initial decision to be made. The rank of the other decision nodes in the tree indicate the order in which decisions are made (from the left to the right). In the tree, the selection of an

Figure 1: An example of a decision tree.



action at node D_i is characterized by an edge (D_i, A) where A is a child of D_i. Note that this selection makes irrelevant all decision nodes outside from the tree rooted at A. For example, in the tree pictured in Figure 1, the decision *up* in D_1 makes irrelevant nodes D_4 and D_5. In the tree, *policies* are characterized by the sequential selection of an action at every (relevant) decision node. Every policy can therefore be represented by a set of edges in the tree. For example, in Figure 1, policy π selecting *up* in nodes D_1, D_2 and D_3 is represented by the following edges: $\{(D_1, C_1), (D_2, C_3), (D_3, C_5)\}$. Every policy induces a probability distribution on the leaves of the tree. This distribution can be represented by a lottery $\ell = (x_1, p_1; \ldots; x_m, p_m)$ yielding outcome x_k with probability p_k, where $p_k, k = 1, \ldots, m$, are strictly positive and add up to one. For example, policy π corresponds to the lottery $\ell = (6, .6375; 13, .2275; 19, .135)$. In the tree depicted in Figure 1, there exist 8 policies leading to 8 distinct lotteries.

We adopt here the Bayesian theory of rational behavior under risk [von Neumann and Morgenstern, 1947; Hammond, 1988]. The DM is assumed to be an expected utility maximizer and is consistent with the postulates of von Neuman and Morgenstern (vNM) theory. In particular, the DM will be indifferent between two lotteries if these lotteries yield the same outcomes with the same probabilities, even if the same probabilities result from different possible combinations of actions and events. The DM's preferences over lotteries are therefore represented by the expected utility model defined by $f(\ell, u) = \sum_{k=1}^{m} p_k u(x_k)$ for any lottery $\ell = (x_1, p_1; \ldots; x_m, p_m)$, where u is the vNM utility function of the DM. A lottery ℓ is preferred to a lottery ℓ' if and only if $f(\ell, u) \geq f(\ell', u)$. The set of policies inherits from the preferences defined over lotteries. Hence a policy is optimal in a tree T if the associated lottery maximizes function f over the set $\mathcal{L}_T$ of all lotteries attached to the policies in T.

In this context, we want to elicit preferences in order to determine the preferred policy without resorting to explicit enumerations and comparisons. As a preliminary to elicitation, in the first part of the paper, we address the computation of possibly optimal lotteries under incomplete preference information (Section 2). This topic has been recently addressed in various contexts related to AI, including multi-agent decision making [Konczak and Lang, 2005; Xia and Conitzer, 2008; Roijers *et al.*, 2014; Aziz *et al.*, 2015], valued constraint satisfaction [Gelain *et al.*, 2010; Marinescu *et al.*, 2013] and multiobjective optimization [Benabbou and Perny, 2015b]. We address here a similar issue in the context of sequential decision making under risk. This work departs from previous studies in decision trees that consider partial preferences induced by imprecise probabilities, see e.g., [Kikuti *et al.*, 2011]. Here the decision tree is completely specified and the probabilities of events are precisely known. The indetermination only derives from imprecise utilities. The second part of the paper is dedicated to preference elicitation (Section 3). We assume that the vNM utility u of the DM is initially unknown and we introduce a new approach interleaving the elicitation of u and the exploration of possibly optimal policies to determine an optimal policy in a decision tree. Finally, we introduce a representation of imprecise utilities using spline functions and we provide numerical tests showing the practical efficiency of the proposed approach (Section 4).

2 Computing Possibly Optimal Lotteries

We consider a situation where the DM's preferences are incomplete. The DM may have expressed some preference statements over various pairs of lotteries, but they are not sufficient to construct a precise vNM utility. These initial preference statements only induce some constraints on the space of utilities restricting the set U of admissible utility functions. Based on this partial information, we are interested in exploring the elements of $PO_U(\mathcal{L}_T)$, the set of possibly optimal lotteries in $\mathcal{L}_T$, i.e. lotteries that are f-optimal for at least one utility function in U. Formally, for any set L of lotteries:

$$PO_U(L) = \bigcup_{u \in U} \arg\max_{\ell \in L} f(\ell, u) \quad (1)$$

This set can be constructed using the U-dominance relation $\succ_U$ defined as follows:

Definition 1. *For any set L of lotteries and for any lottery ℓ_0: $L \succ_U \ell_0 \Leftrightarrow \forall u \in U, \exists \ell \in L, f(\ell, u) > f(\ell_0, u)$.*

For any set L of lotteries, we indeed have:

Proposition 1. $\ell_0 \in PO_U(L) \Leftrightarrow \forall L' \subseteq L, not(L' \succ_U \ell_0)$.

This result directly follows from Definition 1. It shows that $PO_U(L)$ is exactly the set of non U-dominated lotteries in L. Note that, for any set $L' \subseteq L$ and for any lottery $\ell_0 \in L$, if $L' \succ_U \ell_0$ then $L \succ_U \ell_0$; therefore, $PO_U(L) = \{\ell \in L : not(L \succ_U \ell)\}$. As a consequnce, to conclude on whether a given lottery $\ell_0 \in L$ is in $PO_U(L)$ or not, it is sufficient to check if $L \succ_U \ell_0$ holds (there is no need to enumerate and make the test for all subsets $L' \subseteq L$). Moreover, we have the following straightforward result:

Proposition 2. *For any set L of lotteries and for any lottery ℓ_0: $L \succ_U \ell_0 \Leftrightarrow \min_{u \in U} \max_{\ell \in L} \{f(\ell, u) - f(\ell_0, u)\} > 0$.*

This result which directly follows from Definition 1 allows one to check whether $L \succ_U \ell_0$ is satisfied or not by solving an optimization problem over U. We will see in Section 4 that

U can be represented by a convex polyhedron, which enables to solve this optimization problem in polynomial time using linear programming.

In order to compute $PO_U(\mathcal{L}_T)$, we now introduce a backward induction algorithm rolling back the tree from the leaves to the root. We assume that the root of the tree is a decision node and that the set $\mathcal{D}_T$ of decision nodes in T is topologically sorted: if there exists a path from node D_i to node D_j in the tree, then we necessarily have $i < j$. Starting from the last decision node in the topological order, one iteratively computes the set of non U-dominated lotteries attached to subpolicies rooted at decision nodes, until reaching the root of the tree. This idea is implemented in Algorithm 1. In this algorithm, each decision node D_i keeps a set L_i of lotteries attached to subpolicies rooted at that node (which is stored in the *lotteries* attribute). This set is computed by examining every child A of node D_i. If A is a decision node, then all lotteries that are stored in A are inserted in L_i. If A is a leaf of the tree, then the lottery $(A, 1)$ yielding outcome A with probability 1 is inserted in L_i. Otherwise, A is a chance node and all combinations of the lotteries stored in the descendants of A are inserted in L_i. These combinations are computed by recursive calls to *Combination* from the initial call *Combination*(A), where $p(A, A_k)$ denotes the probability on edge (A, A_k) (see Algorithm 2). Finally, only the non U-dominated elements of L_i are stored in the *lotteries* attribute of node D_i (see line 8). The following propositions will be used to prove the correctness of our algorithm:

Proposition 3 (Independence)**.** *For any lottery ℓ_0 and any set L of lotteries, if $L \succ_U \ell_0$ then, for any lottery ℓ' and any $\lambda \in (0, 1)$, we have: $\{\lambda\ell + (1-\lambda)\ell' : \ell \in L\} \succ_U \lambda\ell_0 + (1-\lambda)\ell'$.*

This result directly derives from the fact that the DM is an EU-maximizer and that vNM independence holds in EU theory. Moreover, let $\mathcal{L}_i$ be the set of lotteries attached to the policies rooted at node D_i for $i \in \{1, \ldots, |\mathcal{D}_T|\}$. We have:

Proposition 4. *Let $(D_i, D_j) \in \mathcal{D}_T^2$ be any pair of nodes such that D_j is a descendant of D_i in T. Let ℓ_i be the lottery associated with a policy π rooted at D_i and let ℓ_j be the lottery associated with the subpolicy of π rooted at D_j. If ℓ_j is U-dominated in $\mathcal{L}_j$ then ℓ_i is U-dominated in $\mathcal{L}_i$.*

Proof. Since ℓ_j is U-dominated in $\mathcal{L}_j$, there exists a set $L \subseteq \mathcal{L}_j$ such that $L \succ_U \ell_j$. We want to prove that $L' \succ_U \ell_i$ holds for some set $L' \subseteq \mathcal{L}_i$. Since D_j is a descendant of D_i in the tree, there exists a unique path $(A_1, \ldots, A_n)$ from node D_i to node D_j, with $A_1 = D_i$ and $A_n = D_j$. If $A_k \in \mathcal{D}_T$ for all $k \in \{1, \ldots, n\}$, then we necessarily have $\ell_i = \ell_j$ and $L \subseteq \mathcal{L}_i$. In that case, $L \succ_U \ell_i$ follows from the hypothesis. Otherwise, ℓ_i is necessarily of the form $\ell_i = \lambda\ell_j + (1-\lambda)\ell'$, where $\lambda = \prod_{k:A_k \in \mathcal{C}_T} p(A_k, A_{k+1})$ and $p(A_k, A_{k+1})$ is the probability labelling the edge (A_k, A_{k+1}). Since $L \succ_U \ell_j$ by hypothesis, we obtain $L' \succ_U \ell_i$ by Proposition 3, where $L' = \{\lambda\ell + (1-\lambda)\ell' : \ell \in L\} \subseteq \mathcal{L}_i$. □

Relation $\succ_U$ also satisfies the following property:

Proposition 5 (Transitivity)**.** *For all lotteries ℓ_0, ℓ_1 and for all sets L, L' of lotteries, if $L \succ_U \ell_1$ and $\{\ell_1\} \cup L' \succ_U \ell_0$ then $L \cup L' \succ_U \ell_0$.*

Algorithm 1: Backward induction.

Input: T: a decision tree; U: admissible utilities
1 **for** $i = |\mathcal{D}_T|, \ldots, 1$ **do**
2 $L_i \leftarrow \emptyset$;
3 **for** *each edge* (D_i, A) *in* T **do**
4 **if** $A \in \mathcal{D}_T$ **then** $L_i \leftarrow L_i \cup A.lotteries$;
5 **else if** $A \in \mathcal{C}_T$ **then** $L_i \leftarrow L_i \cup$ Combination(A);
6 **else** $L_i \leftarrow L_i \cup \{(A, 1)\}$;
7 **end**
8 $D_i.lotteries \leftarrow \{\ell \in L_i : not(L_i \succ_U \ell)\}$;
9 **end**
10 **return** $D_1.lotteries$

Algorithm 2: Combination.

Input: A: a chance node
1 Let $A_1, \ldots A_n$ denote the n children of node A;
2 **for** *each edge* (A, A_k) *in* T **do**
3 **if** $A_k \in \mathcal{D}_T$ **then** $L'_k \leftarrow A_k.lotteries$;
4 **else if** $A_k \in \mathcal{C}_T$ **then** $L'_k \leftarrow$ Combination(A_k);
5 **else** $L'_k \leftarrow \{(A_k, 1)\}$;
6 **end**
7 $L \leftarrow \emptyset$;
8 **for** *each* $(\ell_1, \ldots, \ell_n) \in L'_1 \times \ldots \times L'_n$ **do**
9 $L \leftarrow L \cup \{\sum_{k=1}^{n} p(A, A_k)\ell_k\}$
10 **end**
11 **return** L

This is a direct consequence of the transitivity of the preferences induced by $f(., u)$ for all $u \in U$. Due to Equation (1), we also have the following result:

Proposition 6. *For any set L of lotteries and any $\ell_0 \in L$, we have: $\ell_0 \notin PO_U(L) \Rightarrow PO_U(L) \succ_U \ell_0$.*

The last two results are now used to prove the following one:

Proposition 7. *For any two sets L, L' of lotteries, we have: $PO_U(L) \subseteq L' \subseteq L \Rightarrow PO_U(L') \subseteq PO_U(L)$.*

Proof. Let $\ell' \in PO_U(L')$. We want to prove that $\ell' \in PO_U(L)$. Since $\ell' \in PO_U(L')$, we have $not(L'' \succ_U \ell')$ for all $L'' \subseteq L'$ (by (1)). Moreover, we have $PO_U(L) \subseteq L'$ by hypothesis; therefore, $not(L'' \succ_U \ell')$ for all $L'' \subseteq PO_U(L)$. Now, we prove by contradiction that we have $not(L'' \succ_U \ell')$ for all $L'' \subseteq L$. Let us assume that there exists some $L''_0 \subseteq L$ such that $L''_0 \succ_U \ell'$. Since we proved that $not(L'' \succ_U \ell')$ for all $L'' \subseteq PO_U(L)$, we have $L''_0 \not\subseteq PO_U(L)$. Hence there exists some $\ell_0 \in L''_0$ such that $\ell_0 \in L$ and $\ell_0 \notin PO_U(L)$. Then, using Proposition 6, we have $PO_U(L) \succ_U \ell_0$. Moreover, since $L''_0 \succ_U \ell'$, we derive $PO_U(L) \cup L''_0 \backslash \{\ell_0\} \succ_U \ell'$ by transitivity (see Proposition 5). Let $L''_1 = PO_U(L) \cup L''_0 \backslash \{\ell_0\}$. Note that we have $PO_U(L) \subseteq L''_1$ by definition. If there exists $\ell_1 \in L''_1$ such that $\ell_1 \notin PO_U(L)$, then we can iterate the reasoning to derive $L''_2 \succ_U \ell'$ where $L''_2 = PO_U(L) \cup L''_1 \backslash \{\ell_1\}$. Note that $L''_2 = L''_1 \backslash \{\ell_1\}$ since $PO_U(L) \subseteq L''_1$ and $\ell_1 \notin PO_U(L)$. Thus, we can further iterate to construct an embedded sequence $L''_1 \supset \ldots \supset L''_n =$

$PO_U(L)$ such that $L''_k \succ_U \ell'$ for all $k \in \{1, \ldots, n\}$. Therefore, we have $PO_U(L) \succ_U \ell'$. Moreover, $PO_U(L) \subseteq L'$ by hypothesis, which implies $\ell' \notin PO_U(L')$ by Proposition 1. This yields a contradiction since $\ell' \in PO_U(L')$ by definition. Hence, there exists no $L''_0 \subseteq L$ such that $L''_0 \succ_U \ell'$. Moreover, $\ell' \in L' \subseteq L$ by hypothesis. Hence $\ell' \in PO_U(L)$. □

Proposition 8. *Algorithm 1 returns the set $PO_U(\mathcal{L}_T)$.*

Proof. Let L_A denote the output of Algorithm 1. We have to prove that $PO_U(\mathcal{L}_T) \subseteq L_A$ and $L_A \subseteq PO_U(\mathcal{L}_T)$.

- $PO_U(\mathcal{L}_T) \subseteq L_A$: using Proposition 4, we know that only subpolicies leading to U-dominated policies are deleted (in line 8). Since $PO_U(\mathcal{L}_T)$ is exactly the set of non U-dominated elements in $\mathcal{L}_T$ (due to Proposition 1), $PO_U(\mathcal{L}_T) \subseteq L_A$ holds.
- $L_A \subseteq PO_U(\mathcal{L}_T)$: this inclusion is implied by Proposition 7 (with $L = \mathcal{L}_T$ and $L' = L_A$). More precisely, we have $PO_U(\mathcal{L}_T) \subseteq L_A \subseteq \mathcal{L}_T$ which implies $PO_U(L_A) \subseteq PO_U(\mathcal{L}_T)$. Since $PO_U(L_A) = L_A$ (due to line 8), we conclude that $L_A \subseteq PO_U(\mathcal{L}_T)$. □

Example 1. *Coming back to Figure 1 and assuming that no preference information is available, relation $\succ_U$ eliminates* up *in D_5 (edge (D_5, C_9)),* down *in D_3 (edge (D_3, C_6)),* down *in D_2 (edge (D_2, C_4)) and nothing in D_4. This leads to three policies in D_1: $\pi_1 = \{(D_1, C_1), (D_2, C_3), (D_3, C_5)\}$, $\pi_2 = \{(D_1, C_2), (D_4, C_7), (D_5, C_{10})\}$ and $\pi_3 = \{(D_1, C_2), (D_4, C_8), (D_5, C_{10})\}$. Relation $\succ_U$ eliminates no policy in D_1. Hence $PO_U(\mathcal{L}_T) = \{\ell_2, \ell_3, \ell_4\}$, where ℓ_i denotes the lottery associated to policy π_i.*

Algorithm 1 relies on the nice properties of $\succ_U$ that make possible to construct efficiently the optimal policies by dynamic programming. However, there exist instances of binary decision trees of depth d for which all policies are possibly optimal and correspond to disctinct lotteries. In such cases, the set of possibly optimal lotteries includes $\Theta(2^{2^d})$ elements that obviously cannot be enumerated in polynomial time. This suggests inserting preference queries during the search so as to progressively reduce the number of possibly optimal lotteries until being able to determine a near-optimal lottery. The next section investigates this line.

3 Elicitation in Decision Trees

We propose to refine Algorithm 1 by interleaving preference elicitation and backward induction. The idea is to perform an implicit and progressive reduction of $PO_U(\mathcal{L}_T)$ by using new preferences statements to focus the search on the relevant lotteries. Collecting preference information during the search is made possible by the following straightforward result:

Proposition 9. *For any U, U' such that $U' \subseteq U$, for any set L of lotteries and for any lottery ℓ_0: $L \succ_U \ell_0 \Rightarrow L \succ_{U'} \ell_0$.*

This proposition shows that new preference statements obtained during the search do not invalidate the pruning operations (based on relation $\succ_U$) made so far while increasing pruning opportunities for the rest of the search.

In order to collect preference information, we use an adaptive elicitation approach based on regret minimization (as proposed in [Wang and Boutilier, 2003]) relying on the computations of pairwise max regrets (PMR), max regrets (MR) and minimax regrets (MMR) defined as follows:

- $PMR(\ell, \ell', U) = \max_{u \in U} \{f(\ell', u) - f(\ell, u)\}$
- $MR(\ell, \mathcal{L}_T, U) = \max_{\ell' \in \mathcal{L}_T} PMR(\ell, \ell', U)$
- $MMR(\mathcal{L}_T, U) = \min_{\ell \in \mathcal{L}_T} MR(\ell, \mathcal{L}_T, U)$

The optimal lotteries for the minimax regret criterion are the elements of $\arg\min_{\ell \in \mathcal{L}_T} MR(\ell, \mathcal{L}_T, U)$. By definition, the utility loss incurred by the choice of any of these lotteries is bounded above by the minimax regret value $MMR(\mathcal{L}_T, U)$. This suggests progressively collecting preference statements from the DM until $MMR(\mathcal{L}_T, U)$ drops below a given tolerance threshold $\delta \geq 0$. Ideally, we would like to ask preference queries until $MMR(\mathcal{L}_T, U) = 0$, which corresponds to the identification of an optimal lottery. However, to reduce the elicitation burden, it is more efficient to use a threshold $\delta > 0$ representing the maximum admissible gap to optimality.

To determine a lottery $\ell \in \mathcal{L}_T$ with a MR below δ, we propose an interactive backward induction algorithm that generates preference queries to discriminate between the lotteries attached to the decision nodes (see Algorithm 3). We ask preference queries until $MMR(L_i, U) \leq \delta/\eta$ where η denotes the maximum number of decision nodes included in a path from the root to a leaf of the tree. This ensures that the MR value of the returned lottery is bounded above by δ (because any path from the root to a leaf includes at most η decision nodes). Contrary to Algorithm 1, only one lottery is stored in each decision node D_i. This lottery is arbitrarily chosen among those minimizing $MR(\ell, L_i, U)$ over the set L_i of lotteries attached to node D_i (see lines 15-16).

In order to reduce $MMR(L_i, U)$ at a given decision node D_i, one may be tempted to ask the DM to compare two lotteries in L_i and state which one she prefers (as in the *Current Solution Strategy* presented in [Boutilier *et al.*, 2006]). However, lotteries associated with subpolicies in a decision tree are complex objects with multiple possible outcomes, which makes direct comparison cognitively difficult. We propose instead to ask the DM to compare lotteries of type $\ell^x = (x, 1)$ to lotteries of type $\ell^\lambda = (x^\top, \lambda; x^\bot, 1 - \lambda)$, where $x^\top$ and $x^\bot$ are respectively the best and worst possible outcomes. Note that we can impose $u(x^\bot) = 0$ and $u(x^\top) = 1$ since vNM utilities are unique up to positive affine transformations. Hence $f(\ell^\lambda, u) = \lambda u(x^\top) + (1 - \lambda) u(x^\bot) = \lambda$ for all $\lambda \in [0, 1]$. Moreover, we have $f(\ell_x, u) = u(x)$ for all $x \in \mathcal{X}_T$. Therefore, if the DM prefers ℓ^x to ℓ^λ, then we derive the constraint $u(x) \geq \lambda$; otherwise, we have $u(x) \leq \lambda$. This shows that we must choose $\lambda = (\max_{u \in U} u(x) + \min_{u \in U} u(x))/2$ to reduce the uncertainty attached to $u(x)$ as much as possible in the worst-case scenario of answer. At each iteration step, we choose an outcome $x \in \bigcup_{\ell \in L_i} S(\ell)$ maximizing $\max_{u \in U} u(x) - \min_{u \in U} u(x)$, where $S(\ell)$ denotes the set of possible outcomes $\{x_1, \ldots, x_m\}$ of lottery $\ell = (x_1, p_1; \ldots; x_m, p_m)$. This choice enables to obtain interesting performance guarantees as shown below.

Proposition 10. *For any set U of admissible utility functions and any set L of lotteries, if $\max_{u \in U} u(x) - \min_{u \in U} u(x) \leq \delta/(2\eta)$ for all $x \in \bigcup_{\ell \in L} S(\ell)$, then $MMR(L, U) \leq \delta/\eta$.*

Proof. Let u^* denote the actual vNM utility function of the DM. Let $\ell = (x_1, p_1; \ldots; x_n; p_n)$ be an element of $\arg\max_{\ell' \in L} f(\ell', u^*)$. We want to show that $MR(\ell, L, U) \leq \delta/\eta$. For all $\ell' = (x'_1, p'_1; \ldots; x'_m; p'_m) \in L$, we have:

$$PMR(\ell, \ell', U) + f(\ell, u^*) - f(\ell', u^*)$$
$$= \max_{u \in U} \{\sum_{k=1}^{m} (u(x'_k) - u^*(x'_k))p'_k - \sum_{k=1}^{n} (u(x_k) - u^*(x_k))p_k\}$$
$$\leq \sum_{k=1}^{m} \frac{\delta}{2\eta} p'_k - \sum_{k=1}^{n} -\frac{\delta}{2\eta} p_k = \frac{\delta}{2\eta}(\sum_{k=1}^{m} p'_k + \sum_{k=1}^{n} p_k) = \frac{\delta}{\eta}$$

Thus, $PMR(\ell, \ell', U) + f(\ell, u^*) - f(\ell', u^*) \leq \delta/\eta$. Since $f(\ell, u^*) - f(\ell', u^*) \geq 0$ by definition of ℓ, we obtain $PMR(\ell, \ell', U) \leq \delta/\eta$ and so $MR(\ell, L, U) \leq \delta/\eta$. □

Proposition 11. *Algorithm 3 has a time complexity of $O(poly(n_T, 1/\delta))$ and uses only $O(poly(n_T, 1/\delta))$ queries, where n_T is the number of nodes in tree T.*

Proof. On the number of queries: Initially, $\max_{u \in U} u(x) - \min_{u \in U} u(x) \leq u(x^\top) - u(x^\perp) = 1$ for all outcomes $x \in \mathcal{X}_T$ since $u \in U$ is an increasing function. At each iteration step, the question asked in line 12 allows us to divide by two the range of possible values for $u(x)$ for some $x \in \bigcup_{\ell \in L_i} S(\ell)$ such that $\max_{u \in U} u(x) - \min_{u \in U} u(x) > \delta/(2\eta)$. Hence, for each outcome x in the tree, we need at most $\lceil \log_2(2\eta/\delta) \rceil$ queries to reduce the range from 1 to $\delta/(2\eta)$. We know that this is sufficient to obtain $MMR(L_i, U) \leq \delta/\eta$ at every decision node D_i due to Proposition 10. Then, since we have $|\mathcal{X}_T| \leq n_T$ outcomes and $\eta \leq n_T$, the overall number of queries is bounded above by $n_T \lceil \log_2(2n_T/\delta) \rceil$.

On the time complexity: we assume that U-dominance tests and PMR computations can be performed in polynomial time (w.r.t. n_T) using linear programming (this point will be justified in the next section). At each decision node D_i, Algorithm 3 computes the set L_i composed of all the lotteries attached to the m_i children of D_i (see lines 2-7); the U-dominated lotteries are then removed from L_i (see line 8). Since only one lottery is stored in the *lotteries* attribute of each decision node, set L_i necessarily includes at most m_i elements. Hence, in the worst-case scenario, the computation of $MMR(L_i, U)$ requires to solve $m_i(m_i - 1)$ PMR-optimization problems. Since $m_i \leq n_T$ and PMR-optimizations are performed in polynomial time (w.r.t. n_T), we know that all values $MMR(L_i, U)$ are computed in polynomial time (w.r.t. n_T). Moreover, at each decision node D_i, the value $MMR(L_i, U)$ is computed exactly $q_i + 1$ times, where q_i is the number of queries generated at step i. Since we just proved that our algorithm uses only $O(poly(n_T, 1/\delta))$ queries in all, then it has a time complexity of $O(poly(n_T, 1/\delta))$. □

Example 2. *We illustrate Algorithm 3 on the tree of Figure 1 (with $\delta = 0.001$). We assume that no preference information is initially available. In node D_5, no preference query is needed since relation $\succ_U$ eliminates* up. *This is not the case in D_4 where $\succ_U$ does not discriminate between* up *and* down *whose associated lotteries are $\ell = (5, 0.55; 16, 0.45)$ and $\ell' = (3, 0.05; 19, 0.95)$. Since $MMR(\{\ell, \ell'\}, U) > \delta/2$, the DM is asked to compare lottery $(5, 1)$ to lottery $(x^\perp, 0.5; x^\top, 0.5)$ where $x^\perp = 0$ and $x^\top = 20$. If the DM prefers the latter, then U is updated by inserting the constraint $u(5) \leq 0.5$. Now, $MMR(\{\ell, \ell'\}, U) \leq \delta/2$ and* up *is eliminated. Nodes D_3 and D_2 are treated similarly as D_5 but* down *is eliminated instead of* up. *As a result, in D_1, we only obtain policies π_1 and π_3 (as defined in Example 1). Lottery ℓ_1 is now U-dominated in D_1 and therefore lottery ℓ_3 is returned. The problem has been solved with a single query.*

Algorithm 3: Interactive Backward induction.

Input: T: a tree; U: admissible utilities: δ: a threshold
1 **for** $i = |\mathcal{D}_T|, \ldots, 1$ **do**
2 $L_i \leftarrow \emptyset$;
3 **for** *each edge* (D_i, A) *in* T **do**
4 **if** $A \in \mathcal{D}_T$ **then** $L_i \leftarrow L_i \cup A.lotteries$;
5 **else if** $A \in \mathcal{C}_T$ **then** $L_i \leftarrow L_i \cup$ Combination(A);
6 **else** $L_i \leftarrow L_i \cup \{(A, 1)\}$;
7 **end**
8 $L_i \leftarrow \{\ell \in L : \text{not}(L \succ_U \ell)\}$;
9 **while** $MMR(L_i, U) > \delta/\eta$ **do**
10 Select $x \in \underset{x' \in \bigcup_{\ell \in L_i} S(\ell)}{\arg\max} \{\max_{u \in U} u(x) - \min_{u \in U} u(x)\}$;
11 $\lambda \leftarrow (\max_{u \in U} u(x) + \min_{u \in U} u(x))/2$;
12 Ask the DM to compare ℓ^x and ℓ^λ;
13 Update U according to the answer;
14 **end**
15 Select one lottery ℓ in $\arg\min_{\ell' \in L_i} MR(\ell', L_i, U)$;
16 $D_i.lotteries \leftarrow \{\ell\}$;
17 **end**
18 **return** $D_1.lotteries$

4 Implementation

To implement Algorithm 3 efficiently, we first discuss the representation of imprecise utility functions by convex polyhedra. Then, we will report some numerical results.

Representation of imprecise utilities. The first idea that probably comes to mind is to define a variable u_x representing the value $u(x)$ for every outcome $x \in \mathcal{X}_T$. With this representation, any preference statement comparing two lotteries translates into a linear inequality in variables $u_x, x \in \mathcal{X}_T$. Hence the set U is a convex polyhedron. However, this representation has two drawbacks. First, beside the constraints imposed by the observed preferences, a number (still polynomial) of additional constraints is needed to enforce monotonicity of utilities (i.e. $u_x > u_y$ whenever $x > y$). Moreover, the number of parameters to be learned grows with $|\mathcal{X}_T|$.

A more compact representation can be obtained if we define u as a spline function. Spline functions are piecewise polynomials whose elements connect with a high degree of smoothness. They are widely used in data interpolation due to their ability to approximate complex shapes [Beatty and Barsky, 1995; Ramsay, 1988]. Interestingly enough, spline functions can be generated by linear combinations of basis spline functions. This allows to reduce the elicitation of a spline function to the determination of its weights in the

spline basis. In order to model utility functions which are non-decreasing functions of outcomes, one particularly appealing basis of spline functions is the basis of I-spline functions denoted $I_i(x;k,t)$, $i=1,\ldots,m$, where k is the order of the spline function (controlling the degree and the scope of polynomial pieces), t is a subdivision of their definition interval and $m=|t|-k$ is the size of the spline basis. These functions are non-decreasing with respect to x. In this paper, we use I-spline functions of order 3 with a uniform subdivision of the unit interval (assuming that outcomes have been normalized to belong to the unit interval). This is a standard choice to keep a good controllability and flexibility because we keep low the degree of polynomials pieces while guaranteeing that adjacent pieces have matching first and second derivatives. Using the I-spline representation, the DM utility function writes: $u(x)=\sum_{j=1}^{m} b_j I_j(x;3,t),\quad b_j\geq 0,$ $j=1,\ldots,m$. Assuming that outcomes are defined in the $[0,1]$ interval, individual utilities are normalized in order to have $u(0)=0$ and $u(1)=1$ by imposing that $\sum_{j=1}^{m} b_j=1$.

In Figure 2, we give two examples of functions u (plotted in red) generated from a basis of functions $I_j(x;3,t), j=1,\ldots,5$ (plotted in black, blue, cyan, magenta, green), for a uniform subdivision of the interval. In the left part of the figure, the utility is defined from weights $(0.3,0.2,0.0,0.2,0.3)$ whereas in the right part of the figure, the utility is defined from weights $(0.5,0.3,0.2,0.0,0.0)$. Using only these five coordinates, we can generate an infinity of smooth functions covering or well approximating most functions. We might use a finer subdivision to cover even more complex functions including local irregularities.

The main advantage of this representation is to provide a compact definition of the parameter set. Indeed, only m variables are needed to characterize function u. This number is independent of the size of the tree. Moreover, although I-spline functions are not linear, it is worth noting that any preference statement of type "I prefer ℓ to ℓ'" where $\ell=(x_1,p_1;\ldots;x_q,p_q)$ and $\ell'=(x'_1,p'_1;\ldots;x'_r,p'_r)$ translates into a linear inequality in variables $b_j, j=1,\ldots,m$, which writes as follows:

$$\sum_{j=1}^{m} b_j \sum_{k=1}^{q} p_k I_j(x_k;3,t) \geq \sum_{j=1}^{m} b_j \sum_{k=1}^{r} p'_k I_j(x'_k;3,t)$$

Hence U is implicitly represented by a convex polyhedron defined as the set of all vectors $(b_1,\ldots,b_m)$ that are compatible with the available preference statements. Moreover, function u being defined as a convex combination of the I-spline basis functions, it is necessarily monotonic and we can get rid of any monotonicity constraint.

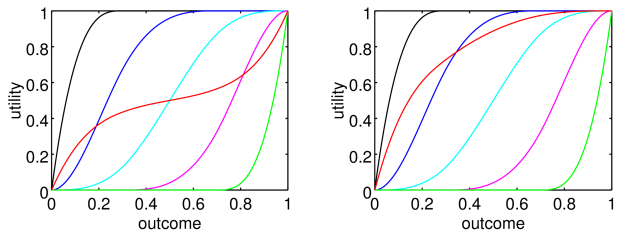


Figure 2: Two examples of utility functions

Numerical tests. Here we consider randomly generated instances of perfect binary trees which alternate decision nodes and chance nodes along each path. We vary d the depth of the tree from 8 to 18 and the tolerance threshold from $\delta=0.05$ to 0.1. As a baseline for comparison, we consider the two stage procedure (named TS) that consists in computing $L=PO_U(\mathcal{L}_T)$ using Algorithm 1 and then asking preference queries until $MMR(L,U)$ drops below δ (queries are generated as described in Section 3). Within both procedures, individual utilities are represented by spline functions. Starting from an empty set of preferences statements, simulated DMs answer to queries according to a randomly generated utility function. In Table 1, we report both t the computation times (in sec.) and q the number of queries; in this table, the symbols "-" simply means that the procedure was still running after 5 hours[1]. First, we see that Algorithm 3 is very efficient both in terms of number of queries and computation times. For example, for instances involving 2^{2^9-1} policies (i.e. $d=18$), it needs no more than 70 seconds and 23 queries to compute the result (for $\delta=0.1$). Moreover, Algorithm 3 is much faster than TS: when $\delta=0.05$ and $d=12$, Algorithm 3 needs about 1 second to compute the result while TS takes more than 35 minutes. This shows that interleaving search and elicitation enables to drastically speed-up the determination of a near-optimal solution in combinatorial domains.

	$\delta=0.1$				$\delta=0.05$			
	Algorithm 3		TS		Algorithm 3		TS	
d	t	q	t	q	t	q	t	q
8	0.15	12.1	0.90	7.2	0.22	17.0	1.69	8.9
10	0.39	16.4	11.94	8.6	0.42	21.0	14.46	10.9
12	1.21	19.7	725.51	9.4	1.20	22.3	2103.53	12.2
14	4.33	20.4	-	-	4.60	25.8	-	-
16	17.10	22.2	-	-	17.21	26.7	-	-
18	65.07	23.0	-	-	68.36	27.4	-	-

Table 1: Elicitation in trees (results averaged over 30 runs).

5 Conclusion

We have proposed an adaptive utility elicitation procedure for the interactive selection of policies in sequential decision problems under risk. Our procedure interleaves preference queries with backward induction to progressively reduce the set of admissible utility functions and therefore the set of possibly optimal lotteries. The proposed procedure determines a near-optimal policy in polynomial time, using a polynomial number of preference queries. In addition to these theoretical guarantees, our numerical tests show the practical efficiency of this approach, especially when utility functions are represented by spline functions. This approach can easily be extended to the elicitation of additive utilities under risk, either in a multi-agent setting or in a multiattribute setting.

[1]The tests are performed on a Intel Core i7-4770 CPU with 15GB of RAM. LPs are optimized using the Gurobi solver.

References

[Aziz *et al.*, 2015] H. Aziz, M. Brill, F. Fischer, P. Harrenstein, J. Lang, and H. G. Seedig. Possible and necessary winners of partial tournaments. *Journal of Artificial Intelligence Research*, 54:493–534, 2015.

[Beatty and Barsky, 1995] J. C. Beatty and B. A. Barsky. *An introduction to splines for use in computer graphics and geometric modeling*. Morgan Kaufmann, 1995.

[Benabbou and Perny, 2015a] N. Benabbou and P. Perny. Combining preference elicitation and search in multiobjective state-space graphs. In *IJCAI*, pages 297–303, 2015.

[Benabbou and Perny, 2015b] N. Benabbou and P. Perny. Incremental weight elicitation for multiobjective state space search. In *AAAI*, pages 1093–1099, 2015.

[Boutilier *et al.*, 2004] C. Boutilier, R. I. Brafman, C. Domshlak, H. H. Hoos, and D. Poole. Preference-based constrained optimization with cp-nets. *Computational Intelligence*, 20(2):137–157, 2004.

[Boutilier *et al.*, 2006] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based Optimization and Utility Elicitation using the Minimax Decision Criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006.

[Boutilier, 2013] C. Boutilier. Computational decision support: Regret-based models for optimization and preference elicitation. *CrowleyP. H. ZentallT. R.(Eds.), Comparative Decision Making: Analysis and Support Across Disciplines and Applications*, pages 423–453, 2013.

[Brafman *et al.*, 2010] R. I. Brafman, F. Rossi, D. Salvagnin, K. B. Venable, and T. Walsh. Finding the next solution in constraint-and preference-based knowledge representation formalisms. In *proc. of KR'10*. Citeseer, 2010.

[Braziunas and Boutilier, 2007] D. Braziunas and C. Boutilier. Minimax regret based elicitation of generalized additive utilities. In *UAI*, pages 25–32, 2007.

[Chajewska *et al.*, 2000] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *AAAI/IAAI*, pages 363–369, 2000.

[Chajewska *et al.*, 2001] U. Chajewska, D. Koller, and D. Ormoneit. Learning an agent's utility function by observing behavior. In *ICML*, pages 35–42, 2001.

[Dasgupta *et al.*, 1995] P. Dasgupta, P.P. Chakrabarti, and S.C. DeSarkar. Utility of *pathmax* in partial order heuristic search. *J. of algorithms*, 55:317–322, 1995.

[Drummond and Boutilier, 2013] J. Drummond and C. Boutilier. Elicitation and approximately stable matching with partial preferences. In *IJCAI*, 2013.

[Fürnkranz and Hüllermeier, 2010] J. Fürnkranz and E. Hüllermeier. *Preference learning: An introduction*. Springer, 2010.

[Gelain *et al.*, 2010] M. Gelain, Maria S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artificial Intelligence*, 174(3):270–294, 2010.

[Ha and Haddawy, 1997] V. Ha and P. Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *UAI'97*, pages 215–222, 1997.

[Hammond, 1988] P. J. Hammond. Consequentialist foundations for expected utility. *Theory and decision*, 25(1):25–78, 1988.

[Hines and Larson, 2010] G. Hines and K. Larson. Preference elicitation for risky prospects. In *AAMAS'10*, pages 889–896, 2010.

[Kikuti *et al.*, 2011] D. Kikuti, F. G. Cozman, and R. Shirota Filho. Sequential decision making with partially ordered preferences. *Artificial Intelligence*, 175(7):1346–1365, 2011.

[Konczak and Lang, 2005] K. Konczak and J. Lang. Voting procedures with incomplete preferences. In *Proc. IJCAI-05 Multidisciplinary Workshop on Advances in Preference Handling*, volume 20, 2005.

[Marinescu *et al.*, 2013] R. Marinescu, A. Razak, and N. Wilson. Multi-objective constraint optimization with tradeoffs. In *International Conference on Principles and Practice of Constraint Programming*, pages 497–512. Springer, 2013.

[Perny and Spanjaard, 2002] P. Perny and O. Spanjaard. On preference-based search in state space graphs. In *AAAI*, pages 751–756, 2002.

[Perny *et al.*, 2016] P. Perny, P. Viappiani, and A. Boukhatem. Incremental Preference Elicitation for Decision Making Under Risk with the Rank-Dependent Utility Model. In *UAI'16*, pages 597–606, 2016.

[Ramsay, 1988] J. O. Ramsay. Monotone regression spline in action. *Statistical Science*, page 425441, 1988.

[Roijers *et al.*, 2014] D. M. Roijers, S. Whiteson, and F. A. Oliehoek. Linear support for multi-objective coordination graphs. In *AAMAS'14*, pages 1297–1304, 2014.

[von Neumann and Morgenstern, 1947] J. von Neumann and O. Morgenstern. *Theory of games and economic behavior*. 2nd Ed. Princeton University Press, 1947.

[Wang and Boutilier, 2003] T. Wang and C. Boutilier. Incremental Utility Elicitation with the Minimax Regret Decision Criterion. In *IJCAI'03*, pages 309–316, 2003.

[Weng and Zanuttini, 2013] P. Weng and B. Zanuttini. Interactive value iteration for markov decision processes with unknown rewards. In *IJCAI'13*, pages 2415–2421, 2013.

[White III *et al.*, 1984] C. C. White III, A. P. Sage, and S. Dozono. A model of multiattribute decision making and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(2):223–229, 1984.

[Xia and Conitzer, 2008] L. Xia and V. Conitzer. Determining possible and necessary winners under common voting rules given partial orders. In *AAAI*, volume 8, pages 196–201, 2008.