Artificial Intelligence 175 (2011) 1346–1365

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Sequential decision making with partially ordered preferences ☆

Daniel Kikuti, Fabio Gagliardi Cozman *, Ricardo Shirota Filho

Escola Politécnica, Universidade de São Paulo, Av. Prof. Mello Moraes, 2231 São Paulo, SP, Brazil

ARTICLE INFO

Article history:
Received 28 February 2009
Received in revised form 11 August 2010
Accepted 11 August 2010
Available online 2 December 2010

Keywords:
Sequential decision making under uncertainty
Partially ordered preferences
Sets of probability measures
Criteria of choice
Consequentialist and resolute norms
Linear and multilinear programming

ABSTRACT

This paper presents new insights and novel algorithms for strategy selection in sequential decision making with partially ordered preferences; that is, where some strategies may be incomparable with respect to expected utility. We assume that incomparability amongst strategies is caused by indeterminacy/imprecision in probability values. We investigate six criteria for consequentialist strategy selection: Γ-Maximin, Γ-Maximax, Γ-Maximix, Interval Dominance, Maximality and E-admissibility. We focus on the popular decision tree and influence diagram representations. Algorithms resort to linear/multilinear programming; we describe implementation and experiments.

1. Introduction

It is often possible, in a decision problem, to express preferences that are completely ordered; that is, for every two alternatives, the decision maker either prefers one to the other, or is indifferent between them. In fact, expected utility theory is based on the assumption that revealed preferences are completely ordered. However, preferences are often *partially* ordered; examples can be found in the theory of CP-nets and the theory of nondeterministic planning, as briefly discussed in Section 2. When preferences are partially ordered, two alternatives may be *incomparable* and incomparability may *fail* to be transitive.

In this paper we focus on preferences that can be represented by a single utility function and a *set* of probability measures. Whenever there are incomplete or partial beliefs, or disagreements amongst experts concerning chances, one may fail to assign a precise probability value to every event, thus producing a partial order with respect to expected utility [3,46,73]. This is the situation we wish to focus on. Section 2 contains the necessary background on these topics.

The literature describes many criteria of choice when preferences are partially ordered [71]. These criteria are covered in Section 3 and can be roughly divided into two groups: (1) criteria that enforce a complete ordering amongst choices (Γ-*Maximin*, Γ-*Maximax* and Γ-*Maximix*); and (2) criteria that select a set of incomparable actions (*Interval Dominance*, *Maximality* and *E-admissibility*). Practical approaches to decision making with sets of probabilities have been mainly limited to the first category; however, recent discussions [60] have highlighted theoretical and behavioral problems when using this group of criteria, and the second group of criteria has been advocated as a more adequate approach. Nevertheless, incomparability comes at a cost, and very little has been observed in the literature in terms of algorithmic progress, mainly due to computational complexity and inability to deal with incomparable choices.

☆ This work has been financially supported by FAPESP, grants 2003/11165-9, 2004/09568-0, 2005/58090-9, 2008/03995-5.
* Corresponding author.
E-mail addresses: danielkikuti@yahoo.com.br (D. Kikuti), fgcozman@usp.br (F.G. Cozman), ricardo.shirota@poli.usp.br (R.S. Filho).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2010.11.017

There are also distinct behavioral norms when it comes to sequential decision making with partially ordered preferences; that is, whenever a sequence of decisions must be made. For instance, a decision maker may be *resolute* in that she commits herself to a complete strategy once and for all, or *consequentialist* in that she allows herself to change the current strategy in case another one is appropriate in view of the future possible choices.

The interplay between criteria of choice, behavioral norms, and models such as decision trees and influence diagrams has not been explored in the literature; this paper aims at filling this gap to some extent. There are indeed insights to be learned from an organized discussion of criteria of choice and behavioral norms; for instance, we discuss in Section 5 the fact that the standard LIMID model clashes with a consequentialist stance.

Another, more substantial, contribution of the paper is the development of algorithms for consequentialist sequential decision making expressed through *decision trees* [56] and *influence diagrams* [34]. Algorithms for decision making under Γ-Maximin and similar criteria have appeared in many settings [58,69,74], while algorithms for decision making under Maximality and E-admissibility have been suggested by Kyburg and Pittarelli [43] and proposed more recently by Kikuti et al. [42] and Utkin and Augustin [72].[1] Section 3 presents algorithms and computational analysis for several criteria of choice. The most valuable contribution of Section 3 is the algorithm for E-admissibility. We also present a new algorithm for strategy selection using linear programming in a family of decision trees where partial preferences have considerable regularity.

Sections 4 and 5 respectively present algorithms for decision making in problems specified through decision trees and influence diagrams. We should note the scarcity of previous literature on influence diagrams under partially ordered preferences, perhaps due to the fact that several criteria of choice require the manipulation of an exponential number of strategies. To reduce this complexity, we examine "ordered" LIMIDs, and we analyze both their conceptual foundation (in particular their clash with consequentialism) and their computational properties.

In short, we present novel results and algorithms for sequential decision making with decision trees and influence diagrams, plus new insights for single-stage decision making under Interval Dominance and E-admissibility. The broader goal of the paper is to combine both the philosophical underpinnings and the computational properties of partially ordered preferences, a combination we feel is missing in the current literature.

2. Partially ordered preferences, behavioral norms, and credal sets

Throughout, our decision makers must select one or more *actions* within a finite set of possible alternatives $\mathbb{A} = \{a_1, \ldots, a_m\}$. Performing action a yields a reward $a(\omega)$ for each state of nature ω; the set of states of nature is assumed to be a finite set $\Omega = \{\omega_1, \ldots, \omega_n\}$. We assume that $a(\omega)$ is a real number expressed in utiles. Even though some theories of preference allow multiple utilities to be defined for a single decision problem [2], in this paper we assume that utilities are precisely fixed in a given decision problem, and consequently every action is identified with a single real-valued function over the states of nature. Note that a utility function is a function that returns a value in utiles for each possible outcome; so we are assuming that a single utility function is fixed.

The connection between preference and expected utility, in decision making under *risk* [47], is based on the axiomatization of preference relations. Denote the strict preference of a_i over a_j by $a_i \succ a_j$, and define indifference between two actions as $a_i \sim a_j \Leftrightarrow \neg(a_i \succ a_j) \wedge \neg(a_j \succ a_i)$. Suppose $\succ$ satisfies (recall that actions are functions that can be multiplied and added) [23]:

Axiom 1 *(completeness).* The relation $\succ$ is complete and negatively transitive (recall that $\succ$ is negatively transitive if it satisfies for all a_i, a_j, a_k: $(a_i \not\succ a_j) \wedge (a_j \not\succ a_k) \Rightarrow (a_i \not\succ a_k)$).

Axiom 2 *(independence).* For $\alpha \in (0, 1]$, $a_i \succ a_j \Rightarrow \alpha a_i + (1-\alpha)a_k \succ \alpha a_j + (1-\alpha)a_k$ (this axiom says that whenever $a_i \succ a_j$, a compound action made of a_i and a_k will be preferred to a compound action made of a_j and a_k, where α denotes the ratio of mixture between the actions).

Axiom 3 *(continuity).* If $a_i \succ a_j \succ a_k$, then there exists $\alpha, \beta \in (0, 1)$ such that $\alpha a_i + (1-\alpha)a_k \succ a_j \succ \beta a_i + (1-\beta)a_k$.

Then there must exist a single probability measure P and a related expected utility representation for $\succ$; that is, the value of an action a_i is given by $E[a_i] = \sum_{j=1}^{n} P(\omega_j)a_i(\omega_j)$, and $a_i \succ a_j$ if and only if $E[a_i] > E[a_j]$.

Several theories relax these axioms, attempting to accommodate various observed decision making patterns [1,20,40]. For instance, *lexicographic preferences* violate Axiom 3 and are encoded through expected utility vectors, ordered with respect to a lexicographic hierarchy [5,23]. Other theories violate Axiom 2 and lead to non-additive functionals that represent preferences [49, Section 2.3]. *Partially ordered preferences* violate Axiom 1 by assuming that preferences are not completely ordered; this is exactly the situation we examine in the present paper. If we assume a single utility function and do not require the preference relation $\succ$ to be complete, we have that $a_i \succ a_j$ if and only if $E_P[a_i] > E_P[a_j]$ for all P in

[1] Most results are based on material presented in Kikuti et al. [42] and Kikuti and Cozman [41]. The third author participated in developing the column generation method, and the experiments, reported in Section 4.3.

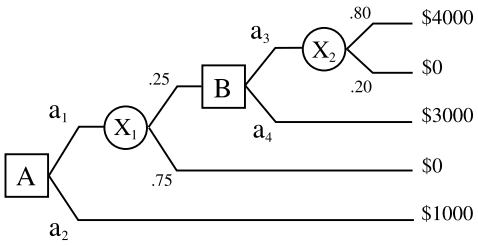


Fig. 1. A sequential decision problem represented through a decision tree.

a set of probability measures [26,61,62]. That is, the preference relation $\succ$ can be completely represented by a set of probability measures K. Incomparability between two actions a_i and a_j appears when one probability measure $P_1 \in K$ produces $E_{P_1}[a_i] > E_{P_1}[a_j]$ while another probability measure $P_2 \in K$ fails to produce $E_{P_2}[a_i] > E_{P_2}[a_j]$.

As alluded to in Section 1, several circumstances may preclude the assessment of a complete preference ordering, from incomplete understanding of a decision scenario, or perhaps a desire to abstract elements of a complex decision situation, to disagreements amongst experts involved in decision making. Sometimes the very language in which preferences are expressed allows for partial specification; this is particularly relevant in artificial intelligence applications. For instance, the semantics of "nondeterministic" actions in planning [6] is that these actions have effects whose probabilities are unknown, and consequently it is not possible to completely order them with respect to expected utility [69]. Another suggestive example is the theory of CP-nets [7], in which a graph-theoretical language organizes preferences about *features* of outcomes rather than outcomes themselves. A CP-net may generate a single preference ordering for outcomes, but in general it specifies a partial ordering. While a CP-net deals with outcomes (and thus reflects incomplete specification of utilities), a similar language for actions would be within the confines of the present paper. Hopefully the present paper will help shorten the gap between the current theories of nondeterministic planning and CP-nets, and the foundational literature on partially ordered planning.

When preferences are partially ordered, there may be no single "best" action to select. Before we examine criteria of choice in Section 3, we review behavioral norms for sequential decision problems (Section 2.1) and some properties of sets of probability measures (Section 2.2).

2.1. Sequential decision problems: strategies and behavior norms

In a sequential decision problem, a decision maker faces a sequence of decisions, and each decision may impact future decisions. A convenient language to introduce sequential decision problems is through *decision trees* [56]. A decision tree $\mathcal{T}$ is a connected graph without cycles, where each node belongs to one of three categories. A decision node $D \in \mathbb{D}$, typically drawn as a square, represents the place where the decision maker must choose an action. A chance node $C \in \mathbb{C}$, typically drawn as a circle, represents an event out of control of the decision maker. A utility node $U \in \mathbb{U}$ is associated with a real-valued utility. In decision trees, a leaf node is a utility node and vice versa. Edges out of a decision node represent the possible actions that the decision maker can choose and edges out of a chance node represent the possible outcomes of the event. A *subtree* of $\mathcal{T}$ is a tree $\mathcal{T}'$ whose nodes and edges form subsets of $\mathcal{T}$. We assume that any tree or subtree is rooted at a decision node. A *strategy* is a complete set of actions specifying how the decision maker should act when she is actually called to decide. A strategy for a subtree is called a *substrategy*. We are interested in selecting *strategies*. The next example clarifies this notion.

Example 1. The decision tree in Fig. 1 is adapted from [40]. It has three strategies: $s_1 = (a_1, a_3)$, $s_2 = (a_1, a_4)$ and $s_3 = (a_2)$; where (a_1, a_3) means that the decision maker will choose action a_1 at A and a_3 if she reaches the decision node B (if she does not reach B then she receives \$0). (One might instead consider all conceivable combinations of decisions, as often done in game theory; in this case we would have strategies such as (a_2, a_3) and (a_2, a_4). We do not follow this route.)

There are two widely debated *behavioral norms* for decision makers engaged in sequential decision problems [21]. A *resolute* decision maker commits herself to a complete strategy once and for all, comparing simultaneously all strategies rooted at the first decision node [50]. A *sophisticated* or *consequentialist* decision maker selects strategies to follow out of a decision node only looking at the subtree rooted at that decision node, and may actually change the strategy previously selected [28,29]. There are other possible norms that we do not investigate further; for instance, a *myopic* decision maker constructs her strategy by selecting actions, at each decision node, independently of future choices [28,67].

The following example illustrates the behavioral norms in the light of a non-expected utility model of preference.

Example 2. Consider again Example 1. Suppose the decision maker has adopted a *rank-dependent utility* model of preference [40], where strategies are ranked using a single utility/probability pair and the function

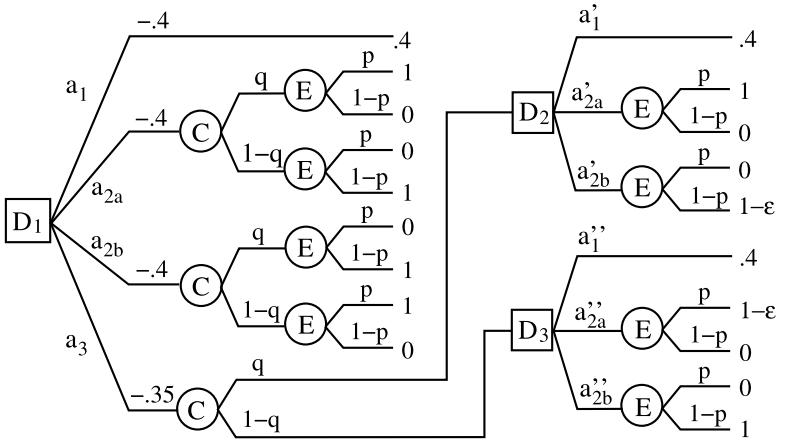


Fig. 2. Decision tree for Example 3.

$$V(a) = a(\omega_1) + \sum_{j=2}^{n} \exp\bigl(-\bigl(-\ln(x_j)\bigr)^{0.5}\bigr)\bigl[a(\omega_j) - a(\omega_{j-1})\bigr],$$

where $x_j = \sum_{k=j}^{n} P(\omega_k)$ and the inequalities $a(\omega_1) \leqslant \cdots \leqslant a(\omega_n)$ are assumed to hold. At node A we have $V(s_1) = 1124.86$, $V(s_2) = 924.23$ and $V(s_3) = 1000$, so at node A, $s_1 \succ s_3 \succ s_2$. However at node B, action a_4 is preferred to a_3 as $V(a_3) = 2494.06$ and $V(a_4) = 3000$. If the decision maker is *resolute*, she selects s_1 and implements it; at B she must choose a_3 even if a_4 is locally better. If the decision maker is *consequentialist*, she would anticipate that in B she would prefer a_4 to a_3, so s_1 is infeasible for her: comparing s_2 and s_3 at A, she must choose s_3. If the decision maker is *myopic*, she would select s_1 at A, but when she reaches B she deviates from s_1 by choosing a_4 (the best option locally), thus actually implementing s_2.

In general, resolute behavior demands examination of all strategies at the beginning. Even though the resolute norm has been forcefully defended when moral aspects of commitments are taken into account [9,51], resolution faces several problems as to how preferences are to be elicited and modeled. By renouncing consequentialism, the decision maker does not have well defined local preferences that can be revealed from her choices [57].[2] A consequentialist norm makes even more sense if we consider a *resource bounded* decision maker who cannot possibly optimize over the space of all strategies [66]. For this reason, we adopt the consequentialist norm throughout this paper.

If a decision maker eliminates an action at a non-root node, and this action might be selected from the perspective of the root node, we have an episode of *incoherent choice*. Another situation is that of *inconsistent choice*, where the decision maker selects a strategy but subsequently deviates from it [28,49]. Incoherent and inconsistent choices do not occur when a decision maker ranks preferences through expected utility with a single utility function and a single non-zero probability measure (in fact, resolute and consequentialist norms are equivalent for such a decision maker). However, incoherence may befall the decision maker when preferences are partially ordered:

Example 3 *(Adapted from Seidenfeld [60]).* In the decision tree depicted in Fig. 2, $p \in [0.25, 0.75]$, $q = 1/2$ and $\epsilon > 0$. There is a charge of 0.4 utiles to take action a_1, a_{2a} or a_{2b}, and a charge of 0.35 utiles to take action a_3, thus $E_P[a_1] = 0$ and $E_P[a_{2a}] = E_P[a_{2b}] = 0.1$ for every possible P. There are 9 additional strategies to consider, by combining actions out of D_2 and D_3. Suppose we are interested in strategies with maximum minimum expected utility (that is, we adopt the Γ-Maximin criterion to be detailed later). For small ϵ, at D_2 the Γ-Maximin action is a'_1, and at D_3 the Γ-Maximin action is a''_1; however the strategy (a_3, a'_1, a''_1) is not Γ-Maximin at D_1, as (a_3, a'_{2a}, a''_{2b}) is a strategy with larger minimum expected utility. This is an episode of incoherent choice, since we throw away the actions that would lead to better strategy at the root node.

We will further discuss incoherent and inconsistent choices in Section 4.1, after detailing some criteria of choice.

2.2. Partially ordered preferences through sets of probability measures

As noted previously, partially ordered preferences can often be represented by sets of probability measures. We call a set of probability measures a *credal set* [46], and denote by $K(X)$ a credal set that contains distributions for a random

[2] Jaffray [38] combines consequentialist preferences and non-consequentialist behavior. Nielsen and Jaffray [52] use the rank-dependent utility model with preferences function revealed by the *anticipated utility theory* of Quiggin [55].

variable X. We assume throughout that credal sets are given by a finite set of linear constraints, thus being closed convex with finitely many vertices (Example 4 shows one such credal set).

Given a set of assessments containing constraints on probability values, any credal set that satisfies the constraints is an *extension* of the assessments. Given an event A, $\underline{P}(A) = \min_{P \in K} P(A)$ and $\overline{P}(A) = \max_{P \in K} P(A)$ are respectively the *lower probability* and the *upper probability* of A. Given a random variable X, $\underline{E}[X] = \min_{P \in K} E[X]$ and $\overline{E}[X] = \max_{P \in K} E[X]$ are respectively the *lower expectation* and the *upper expectation* of X (we use expected value and expectation as synonyms). A *conditional credal set* $K(\cdot|A)$ is obtained by conditioning (Bayes' rule) every measure in a credal set with respect to A (likewise, $K(\cdot|Y)$ is produced by elementwise conditioning with respect to a random variable Y). We assume that any conditioning event has lower probability strictly larger than zero ($\underline{P}(A) > 0$). If the (joint) credal set $K(X, Y)$ is such that every one of its vertices satisfies stochastic independence of X and Y (that is, all vertices factorize as $P(X)P(Y)$), then X and Y are said to be *strongly independent*. There are several other concepts of independence for credal sets in the literature [11,12]; strong independence is perhaps the most popular, and we adopt it in this paper. Conditional strong independence is defined in the obvious manner, by requiring conditional stochastic independence for every vertex of the conditional credal set of interest.

We shall, when we deal with influence diagrams, use elements of the theory of credal networks. A *credal network* is a graph-theoretical representation for a joint credal set $K(X_1, \ldots, X_n)$ that mimics the structure of a Bayesian network [13]. A credal network consists of a directed acyclic graph such that each node is identified with a random variable X_i. The parents of variable X_i in the graph are denoted by $\mathrm{pa}(X_i)$. Each variable is associated with a conditional credal set $K(X_i|\mathrm{pa}(X_i) = \pi_k)$ for each value π_k of $\mathrm{pa}(X_i)$, and every variable is assumed strongly independent of its nondescendants in the graph given its parents in the graph. Thus the largest extension of all assessments in a credal network, called the *strong extension*, is a joint credal set $K(X_1, \ldots, X_n)$ given by the convex hull of the set of joint distributions: $\{\prod_{i=1}^{n} p(X_i|\mathrm{pa}(X_i)):\ p(X_i|\mathrm{pa}(X_i) = \pi_k) \in K(X_i|\mathrm{pa}(X_i) = \pi_k)\}$ where this expression refers to densities induced by the appropriate probability distributions. An *inference* is then the computation of lower/upper probabilities for the values of some variable. In general, inference with strong extensions is NP^{PP}-complete; with constraints on the induced width of credal networks, the complexity of inferences is in NP [17]. The best available algorithms for inference with strong extensions optimize a multilinear polynomial $\prod_{i=1}^{n} p(X_i|\mathrm{pa}(X_i))$ subject to assessments in the credal network [16]. Some variables may be discarded when computing a particular inference, using the *d-separation* property that strong extensions inherit from Bayesian networks [12].[3]

We will need to solve multilinear programs several times in this paper. The most refined algorithm for solution of multilinear programming problems arising from judgements of independence seems to be the adaptation of Sherali and Tuncbilek [65]'s RL method by de Campos and Cozman [17]. We have used the adapted RL method in our implementation (Section 5.2), and we usually take the solution of a multilinear program to be a "unit" of computation, even though such a solution may require substantial effort in itself.

3. Criteria of choice in single-stage decision making

In this section we study several criteria of choice for partially ordered preferences; that is, criteria that select one or more actions from a given set of actions. We present the basic computations that must be performed in a single-stage decision making problem, and we present short code fragments that are used later. While several of these algorithms have appeared in the literature [70,71], the discussion contributes with new analyses both for Interval Dominance and for E-admissibility [42]. The computational cost of the algorithms is presented as a function of the number of auxiliary optimization programs that must be solved. The following example clarifies the nature of these programs (the example deals with linear constraints; later we face situations where auxiliary programs are multilinear).

Example 4. Consider a decision problem with three states, x_1, x_2 and x_3 that are values of a random variable X. Actions and utilities are given in Fig. 3. Suppose the credal set $K(X)$ is specified through $P(X = x_1) \in [1/10; 7/20]$, $P(X = x_2) \in [1/5; 2/5]$, and $P(X = x_3) \in [7/20; 13/20]$, as depicted in Fig. 3. To obtain lower and upper expectations for actions a_i (also in Fig. 3), we must solve linear programs of the form $\min / \max E[a_i] = \sum_j p_j a_i(x_j)$ subject to $\sum_j p_j = 1$ and $\underline{P}(X = x_j) \leqslant p_j \leqslant \overline{P}(X = x_j)$ for $j = 1, 2, 3$.

Existing criteria can be roughly divided into two groups. *Indecision-resistant* criteria force a single ordering of choices, and thus select either a single action or a set of equally ranked (with respect to choice) actions. *Indecision-prone* criteria may return a set of actions that are deemed incomparable (with respect to preference).

We start by briefly examining indecision-resistant criteria: Γ-Maximin, Γ-Maximax, and Γ-Maximix. The *Γ-Maximin* criterion selects an action with highest lower expectation, a "pessimistic" solution that focuses on worst case scenarios [3,25]. Algorithm 1 is an easy translation of the Γ-Maximin criterion. The *Γ-Maximax* criterion selects an action with highest upper expectation, an "optimistic" solution that focuses on best case scenarios [58]. The *Γ-Maximix* criterion selects

[3] Given three collections of variables **X**, **Y** and **Z**, suppose that along every path between a variable in **X** and a variable in **Y** there is a variable W such that: either W has two converging arrows and is not in **Z** and none of its descendants are in **Z**, or W is in **Z**. Then **X** and **Y** are d-separated by **Z** [54].

	x_1	x_2	x_3	$\underline{E}[a_i]$	$\overline{E}[a_i]$
a_1	9	7	1	3.3	5.6
a_2	0	5	8	4.3	6.45
a_3	5	5	5	5.0	5.0
a_4	6	4	4	4.2	4.7
a_5	2	6	5	4.2	5.1

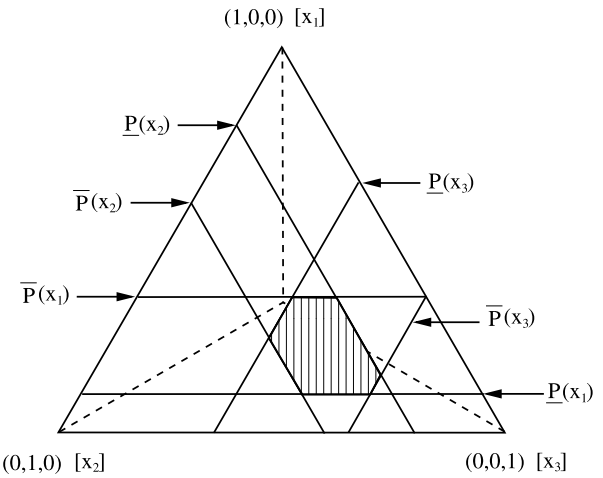


Fig. 3. Left: Actions, utilities, and lower and upper expected utilities for Example 4. Right: Credal set $K(X)$ (hatched area) defined by probability intervals in Example 4; axes denote probability of x_1, x_2 and x_3.

Algorithm 1: Criterion_Γ-Maximin

Input: Set of actions $\mathbb{A}$, and set of constraints K on probability values.
Output: A Γ-Maximin action.

```
1 a* ← a_1, x ← E̲[a_1];
2 foreach a_i ∈ 𝔸\a_1 do if E̲[a_i] > x then {a* ← a_i, x ← E̲[a_i]};
3 return a*;
```

Algorithm 2: Criterion_Maximality

Input: Set of actions $\mathbb{A}$ containing $\#\mathbb{A}$ actions (each action has attribute "admissible", initially set to true), and set of constraints K on probability values.
Output: The set of maximal actions.

```
1 for i ← 1 to (#𝔸 − 1) do
2     for j ← i + 1 to (#𝔸) do
3         if E̲[a_i − a_j] > 0 then a_j.admissible ← false;
4         else if Ē[a_i − a_j] < 0 then a_i.admissible ← false;
5 return All actions with attribute "admissible" set to true;
```

$a^* = \arg\max_{a_i \in \mathbb{A}}(\eta \underline{E}[a_i] + (1-\eta)\overline{E}[a_i])$, where $\eta \in [0,1]$ reflects the degree of ambiguity aversion [72], and is already sketched by Hurwicz [36]. Algorithm 1 can be easily modified to deal with the Γ-Maximax and Γ-Maximix criteria. For these three criteria, Algorithm 1 returns the selected action by solving a number of optimization programs, each subject to constraints in K. The number of optimization programs is clearly linear on the number of actions. In Example 4, the Γ-Maximin criterion selects a_3, while the Γ-Maximax criterion selects a_2 and the Γ-Maximix criterion also selects a_2 for $\eta = 0.5$.

Consider indecision-prone criteria, starting with Maximality. An action a_i is *maximal* if there is no action a_j such that, for each possible probability measure $P \in K$, $E_P[a_j] > E_P[a_i]$. The maximality criterion is based on pairwise comparisons amongst actions, as indicated by Algorithm 2 (in Algorithm 2 we use the fact that $E_P[a_i] > E_P[a_j]$ for all P is equivalent to $\underline{E}[a_i - a_j] > 0$ [73]). To handle n actions, the algorithm must solve at most $(n^2 - n)$ optimization programs. In Example 4, to determine whether a_4 can be maximal in the presence of a_3, we must solve the linear program $\max(6p_1 + 4p_2 + 4p_3 - 5p_1 - 5p_2 - 5p_3)$ subject to $\sum_{i=1}^{3} p_i = 1$, $1/10 \leqslant p_1 \leqslant 7/20$, $1/5 \leqslant p_2 \leqslant 2/5$, and $7/20 \leqslant p_3 \leqslant 13/20$. We find that the maximum is $3/10$, hence a_4 is not maximal when a_3 is present.

We now examine Interval Dominance and E-admissibility in more detail.

3.1. Interval Dominance

Interval Dominance selects one or more *I-admissible* actions as follows [70]. Action a_j is I-inadmissible when a_i is present if $\underline{E}[a_i] > \overline{E}[a_j]$. The I-admissible actions are the actions that are never I-inadmissible. Note that Interval Dominance does not identify dominance in the sense that, given two actions a_i and a_j, a_i dominates a_j if for all probability measures P,

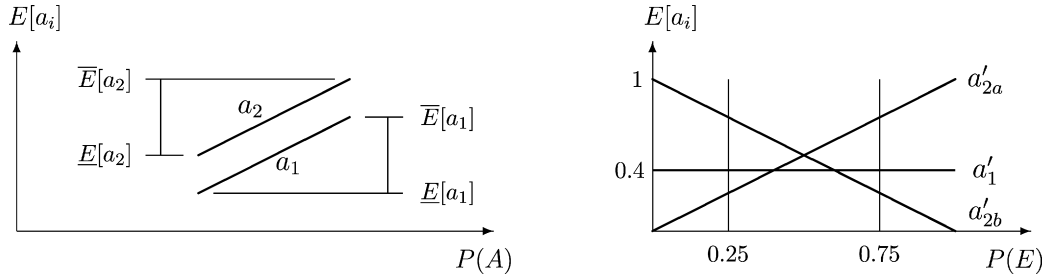


Fig. 4. Left: Interval Dominance does not capture "true" dominance (slanted lines denote the expectations of actions a_1 and a_2 as the probability of event A varies). Right: Actions and their expected utilities in Example 5.

Algorithm 3: Criterion_IntervalDominance

Input: Set of actions $\mathbb{A}$ (each action has attribute "admissible", initially set to true), and set of constraints K on probability values.
Output: The set of I-admissible actions.

1 $a^* \leftarrow$ Criterion_Γ-Maximin($\mathbb{A}, K$); $x \leftarrow \underline{E}[a^*]$;
2 **foreach** $a_i \in \mathbb{A}$ **do** **if** $x > \overline{E}[a_i]$ **then** a_i.admissible $\leftarrow$ false;
3 **return** All actions with attribute "admissible" set to true;

$E_P[a_i] \geqslant E_P[a_j]$: in Fig. 4, a_2 has higher expectation than action a_1 for every probability value $P(A) \in [0.3, 0.7]$, but Interval Dominance does not choose between these actions.

A naive method to generate I-admissible actions would compare every pair of actions. Algorithm 3 avoids unnecessary computation of lower and upper expectations by using the Γ-Maximin solution a^*. Action a^* is always I-admissible: suppose otherwise that a^* is I-inadmissible; then there is a' with higher lower expectation, contradicting the hypothesis. The comparison of all actions with a^* generates the I-admissible actions (an action a' dominated by another action a'' is also dominated by action a^* with the maximum lower expectation, because $\underline{E}[a] \geqslant \underline{E}[a'']$). To find a^* we must solve n optimization programs; to determine the set of admissible actions we must solve $n-1$ additional optimization programs. Consequently, Algorithm 3 returns the I-admissible actions by solving a linear (on the number of actions) number of optimization programs.

3.2. E-admissibility

The criterion of E-admissibility, where E stands for "expectation" [45], focuses on actions that maximize expected utility. Given a set of actions $\mathbb{A}$ and a credal set K, the action $a_i \in \mathbb{A}$ is E-admissible when, for at least one $P \in K$, a_i maximizes expected utility [59]:

$$a_i \text{ is E-admissible when } \exists (P \in K)\colon\ \forall (a_j \in \mathbb{A},\ j \neq i) : E_P[a_i - a_j] \geqslant 0. \tag{1}$$

A variant of E-admissibility has been explored for Markov decision processes with imprecise probabilities by Itoh and Nakamura [37].

Example 5. Take a'_1, a'_{2a} and a'_{2b} as in Example 3, and assume $\epsilon = 0$. Expected utilities are shown in Fig. 4 (right). Only actions a'_{2a} and a'_{2b} are E-admissible: action a'_{2b} maximizes expected utility for $P(E) \in [0.25, 0.5]$, while action a'_{2a} maximizes expected utility for $P(E) \in [0.5, 0.75]$. Even though a'_1 is the Γ-Maximin action, it never maximizes expected utility and is not E-admissible.

E-admissibility is qualitatively different from the previous criteria in that it does not depend on pairwise comparisons; rather, it is based on the existence of specific probability measures in the underlying credal set. Thus one might think that E-admissibility is more difficult to handle computationally than the other criteria. This feeling transpires in the literature on decision making with partially ordered preferences, as best expressed in Troffaes' [70] excellent review. However, it is possible to reduce the search for E-admissible actions to a linear sequence of optimization programs, using insights first derived by Kyburg and Pittarelli [43]. The original discussion by Kyburg and Pittarelli did not focus on computational cost, and it lay dormant until the same techniques surfaced independently in work by Kikuti et al. [42] and Utkin and Augustin [72], in response to Troffaes' [70] analysis.

The basic idea is that an action a_i is E-admissible if there is $P \in K$ such that all constraints generated by Expression (1) are satisfied. If these constraints cannot be satisfied, then a_i is not E-admissible. Algorithm 4 generates exactly these constraints: the linear expression $E_P[a_i - a_j] \geqslant 0$ in line 3, stored in the set $\mathcal{C}$, denotes a symbolic constraint on the free (not yet bound) values of P. This way the algorithm avoids the need to represent credal sets explicitly (that is, the need to enumerate vertices). As every action is verified only once:

Algorithm 4: Criterion_E-admissibility

Input: Set of actions $\mathbb{A}$ containing $\#\mathbb{A}$ actions (each action has attribute "admissible", initially set to true), and set of constraints K on probability values.
Output: The set of E-admissible actions.

```
for i ← 1 to (#𝔸) do
    C ← K;
    for j ← 1 to (#𝔸) do if i ≠ j then C ← C ∪ {E_P[a_i − a_j] ⩾ 0};
    if C is not feasible then a_i.admissible ← false;
return All actions with attribute "admissible" set to true;
```

Proposition 1. *Algorithm* 4 *returns the E-admissible actions by solving a linear* (*in the number of actions*) *number of optimization programs.*

In Example 4, a_1, a_2 and a_3 are E-admissible actions. To verify whether a_1 is E-admissible, we must verify whether the following linear constraints can be satisfied together: $\sum_{i=1}^{3} p_i = 1$ and

$$1/10 \leqslant p_1 \leqslant 7/20, \quad 1/5 \leqslant p_2 \leqslant 2/5, \quad 7/20 \leqslant p_3 \leqslant 13/20,$$
$$9p_1 + 7p_2 + p_3 - 0p_1 - 5p_2 - 8p_3 \geqslant 0, \qquad 9p_1 + 7p_2 + p_3 - 5p_1 - 5p_2 - 5p_3 \geqslant 0,$$
$$9p_1 + 7p_2 + p_3 - 6p_1 - 4p_2 - 4p_3 \geqslant 0, \qquad 9p_1 + 7p_2 + p_3 - 2p_1 - 6p_2 - 5p_3 \geqslant 0.$$

4. Algorithms for sequential decision making: decision trees

In this section we derive algorithms for sequential decision making with decision trees that display indeterminacy/imprecision in probability values.

4.1. Preliminaries: the option for consequentialism

In a decision tree where chance nodes are associated with credal sets we may face differences between resolute and consequentialist behaviors.

We start by noting that E-admissibility and Maximality never lead to incoherent choice as both resolute and consequentialist norms produce identical sets of strategies [35,60].[4] Thus with E-admissibility and Maximality it is possible to run backward induction and produce a sequence of substrategies that satisfy consequentialism and that reach the strategies complying with the resolute norm. As a digression, note that such backward induction scheme is exactly given by Algorithm 5, detailed later, when this algorithm is specialized to the E-admissibility and Maximality criteria [42]; we also note that recent work by Huntley and Troffaes [35] yields simplifications to Algorithm 5 when applied to Maximality.

The remaining criteria in Section 3 may produce distinct consequentialist and resolute behaviors. The Γ-Maximin criterion was considered in Example 3. Clearly the same applies to the Γ-Maximix and Γ-Maximax criteria, since they do not guarantee that a discarded substrategy is not part of an optimal strategy. To illustrate this for Γ-Maximax criterion, consider again Example 3. At D_2, the Γ-Maximax action is a'_{2a}, and at D_3, the Γ-Maximax action is a''_{2b}. However at D_1 the Γ-Maximax strategy is either (a_3, a'_{2a}, a''_{2a}) or (a_3, a'_{2b}, a''_{2b}).

Incoherent choice can also happen with Interval Dominance, a fact that apparently has not been indicated before:

Example 6. In the decision tree depicted in Fig. 5, suppose $p \in [1/10, 3/10]$, $q_1 \in [1/5, 2/5]$, $q_2 \in [2/5, 3/5]$ and $q_3 \in [3/5, 4/5]$. At D_2, action a'_1 dominates action a'_2, but the strategy (a_1, a'_2) is I-admissible at D_1.

In this paper we adopt the consequentialist position that the best substrategy starting at a decision node can depend only on the subtree starting at that node, motivated by the fact that the sensible alternative, resolute behavior, is computationally unfeasible in general for bounded agents (in a sequential decision problem, the resolute behavior can be viewed as a "brute-force" method, that demands the enumeration of all possible strategies).

4.2. Selecting strategies

Algorithm 5 presents a general framework for consequentialist sequential decision making in the presence of indeterminacy/imprecision in probabilities — that is, under partially ordered preferences. The algorithm can be specialized by replacing throughout CRITERION by the desired criterion of choice. The intuition behind the algorithm is that a strategy can

[4] Nevertheless, both E-admissibility and Maximality still may lead to inconsistent choice, where the decision maker plans for an action but then executes a different action.

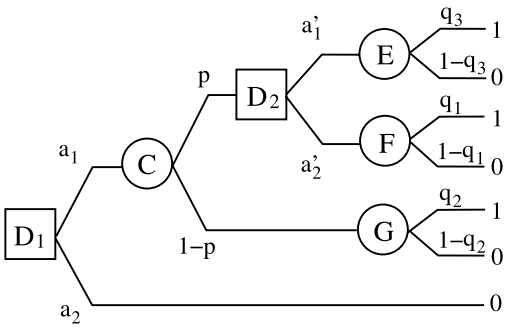


Fig. 5. Decision tree for Example 6.

Algorithm 5: DecisionTree_CRITERION

Input: A decision tree $\mathcal{T}$ as described in Section 4.1, with a set $\mathcal{T}.K$ of constraints on probability values.
Output: A set of strategies selected by CRITERION.

```
1  m ← Number of decision nodes;
2  for i ← m until 1 do
3      S ← ∅;
4      foreach D' that is a child of D_m do
5          if D' ∈ 𝔻 then
6              foreach s_j ∈ D'.substrategies do S ← S ∪ {D', s_j};
7          else if D' ∈ ℂ then
8              foreach s_j ∈ Combination(D') do S ← S ∪ {D', s_j};
9          else S ← S ∪ {D'}
10     K ← GenerateConstraints(S, D_m, T.K);
11     D.substrategies ← Criterion_CRITERION(S, K);
12 return Content of attribute substrategies for root decision node;
```

be constructed by visiting the nodes backwards, i.e. from last (leaf) to first (root), by selecting the optimal choice as decision nodes are encountered. The optimal choice at a decision node is then combined to the array of optimal choices selected at previously visited nodes, until the root node is resolved. In order to accomplish this, Algorithm 5 assumes that each decision node keeps a list of admissible substrategies (according to CRITERION) rooted at that node; this list is kept in the "substrategies" attribute. The remainder of this section is dedicated to a detailed discussion of technical aspects of this algorithm. In addition, we propose a simple transformation that allows the algorithm to be solved using linear programming instead of the more computationally demanding multilinear program. We also show that this linear program can benefit from the use of the column generation technique in order to obtain solutions more efficiently.

We assume that the decision nodes in $\mathbb{D}$ are topologically sorted, that is, they follow a linear (temporal) ordering of decisions such that if $\mathcal{T}$ contains a path from decision node D_x to D_y, then D_x appears before the decision node D_y in the ordering. We assume that D_1 is the root of $\mathcal{T}$ and D_m is the last decision node in such ordering.

As mentioned, the construction of strategies in our algorithm starts backwards at a leaf of the tree, that is, at a utility node (line 9 of Algorithm 5). We visit each decision node D_m (from last to first) and examine its children. If a child D' of D is a decision node, the substrategies rooted at D' are combined with the action that prescribes a move to D' (this is indicated by storing D' at the beginning of the substrategies). If D' is a chance node, there are two cases to handle. If a chance node has no decision nodes as successors, the only substrategy rooted at D is a substrategy that simply moves to D'. If a chance node has decision nodes as successors, it is then necessary to combine all substrategies that can branch out of D'. For instance, if there are three decision nodes out of a chance node, and each one of these decision nodes leads to two substrategies, then eight substrategies must be produced. We assume that these substrategies are returned by the function Combination in line 8 of Algorithm 5.

The substrategies $\mathcal{S}$ generated in the loop from line 4 to line 9 are fed to the appropriate function Criterion_CRITERION in line 11. This function treats each substrategy as an action, and it must also receive the constraints and parameters of optimization programs that are run so as to select substrategies in Criterion_CRITERION. These constraints are basically contained in the input set $\mathcal{T}.K$, and given in terms of local assignments.

Once the constraints are generated, they are processed by Criterion_CRITERION in a sequence of optimization programs. Fix a decision node D and strategy s; the expectation of s is:

$$\sum_{x_1,\ldots,x_N\in\{0,1\}} P\big(X_1 = x_1 \big| \mathrm{an}(X_1)\big)\ldots P\big(X_N = x_N \big| \mathrm{an}(X_N)\big) U(s, x_1, \ldots, x_N), \tag{2}$$

where X_i denotes indicator functions for each one of the events in the subtree rooted at D only with branches selected by s, $\mathrm{an}(X_i)$ is the set of nodes in the path from D to X_i, and $P(X_i = x_i|\mathrm{an}(X_i))$ is a local assignment on the probability

that the event X_i obtains x_i. This expression is clearly in a multilinear form, and optimizing it subject to constraints on probability values (the terms in the product) takes us to nonlinear programming, as has been pointed out before [14,15,42]. Nonlinear programming is known to be a class of very difficult problems to solve, and specific optimization methods have been recently applied to the particular case of sequential decision making under the Γ-Maximax criteria by de Campos and Ji [19].

There are, however, interesting situations where all optimization programs in Criterion_CRITERION can be transformed into *linear* programs, as discussed in the next section.

4.3. Selecting strategies with linear programming

A linear formulation is obtained for strategy selection when assessments are linear constraints and are *separately specified* in the sense that probability constraints for the event at a particular chance node do not depend on probability values for any other event. For instance, assessments are separately specified if $\mathcal{T}.K$ contains only bounds on probabilities such as $P(A|B) \in [\alpha, \beta]$ where A is an event at a chance node and B is a conjunction of events in the decision tree from the root to that node (as illustrated by Example 6).

To obtain linear programs, we first note that Expression (2) can be written as

$$\sum_{x_1,\ldots,x_N \in \{0,1\}} P(X_1 = x_1, \ldots, X_N = x_N) U(s, x_1, \ldots, x_N), \tag{3}$$

where $P(X_1 = x_1, \ldots, X_N = x_N)$ are the probability values to optimize over. For instance, consider again constraints such as $P(A|B) \in [\alpha, \beta]$, where A and B are (conjunctions of) events in the decision tree. We can use Bayes' rule of conditioning to transform these constraints into the form $\alpha P(B) \leqslant P(A \cap B) \leqslant \beta P(B)$ under the assumption that probabilities are strictly positive. In Example 6, we have the constraints:

$$1/10 \leqslant P(C) \leqslant 3/10, \qquad 3/5 P(C) \leqslant P(E \cap C) \leqslant 4/5 P(C),$$
$$1/5 P(C) \leqslant P(F \cap C) \leqslant 2/5 P(C), \qquad 2/5 P(C^c) \leqslant P(G \cap C^c) \leqslant 3/5 P(C^c).$$

It is instructive to analyze the size of the linear programs that are generated by this method. One extreme (favorable) situation is represented by a symmetric decision tree where each decision node in the same slice branches into a constant number of chance nodes containing the same event. That is, the first decision node branches into several nodes containing event A; then the decision nodes out of these two chance nodes branch into several chance nodes all labeled with event B, and so on, as depicted in Fig. 6 (left) for the case of branching factor equal to two. For a fixed strategy, we have a symmetric binary tree, and if the problem deals with N events, this binary tree contains $2^N - 1$ nodes. Each complete path from the root to a utility node corresponds to a complete conjunction of events and complements of events, and the linear program to be built has as many optimization variables as there are paths in this symmetric binary tree. We reach the satisfying conclusion that, for symmetric decision trees, the linear programs that must be solved within Algorithm 5 are polynomial on the size of the decision tree (this is of course not entirely comforting as the size of these decision trees is exponential on the number of events). Fig. 6 (right) shows running times for the computation of lower expected value for a given strategy, as this is the basic operation for all criteria of choice. Points in that graph have been produced by generating symmetric decision trees with randomly generated utilities and lower/upper probabilities for events. The implementation is coded in AMPL and uses the CPLEX commercial package as linear programming solver; experiments were run in a microcomputer with two dual-core processors and 4 GBytes of memory. One sees that running times are quite small for $N \leqslant 10$; it is hard to imagine a symmetric decision tree with more than 10 chance nodes. Just to compare, we have also solved the multilinear formulation (2) using the programming package Multilin [17]. The multilinear programs took minutes even for $N = 5$ and often failed to converge (Multilin produces successive approximations and typically reaches a vicinity of the solution quickly, but then converges very slowly).

Now consider the other extreme situation, where for every fixed strategy we have a symmetric binary tree such that each chance node contains a *different* event. Here the linear program that computes the lower/upper expectation of a strategy is exponentially larger than the decision tree: for a binary tree with height H, there are $N = 2^H - 1$ chance nodes and $2^N = 2^{(2^H - 1)}$ optimization variables. That is, we may have a relatively small decision tree that leads to very large linear programs: for example, in a binary tree of height 5 (5 levels), there are $N = 2^5 - 1 = 31$ chance nodes and $2^N = 2^{31}$ optimization variables. As indicated in Fig. 6 (right), running times grow substantially as N grows beyond 10. Here we face a situation that is similar to *probabilistic logic*; that is, we have a relatively small set of constraints on N events and we must handle 2^N configurations of these events [24,27,31]. Such problems have been tackled with column generation [39] and redundancy detection [32,48]. We are interested in minimizing/maximizing an objective function given by Expression (3). We can write this expression as a product of vectors $u \cdot p$, where u contains the values of the utility nodes and p the probability values over the 2^N possible configuration of events. We also know from $\mathcal{T}.K$ the constraints p is subject to, that can also be easily written in matrix form by $\mathbf{A}p \geqslant \mathbf{b}$.

It is clear that $\mathbf{A}$ has a very large number of columns (more precisely, 2^N, one for each possible configuration of events). Storing and manipulating such an amount of columns is very inefficient and time consuming. However, by using column

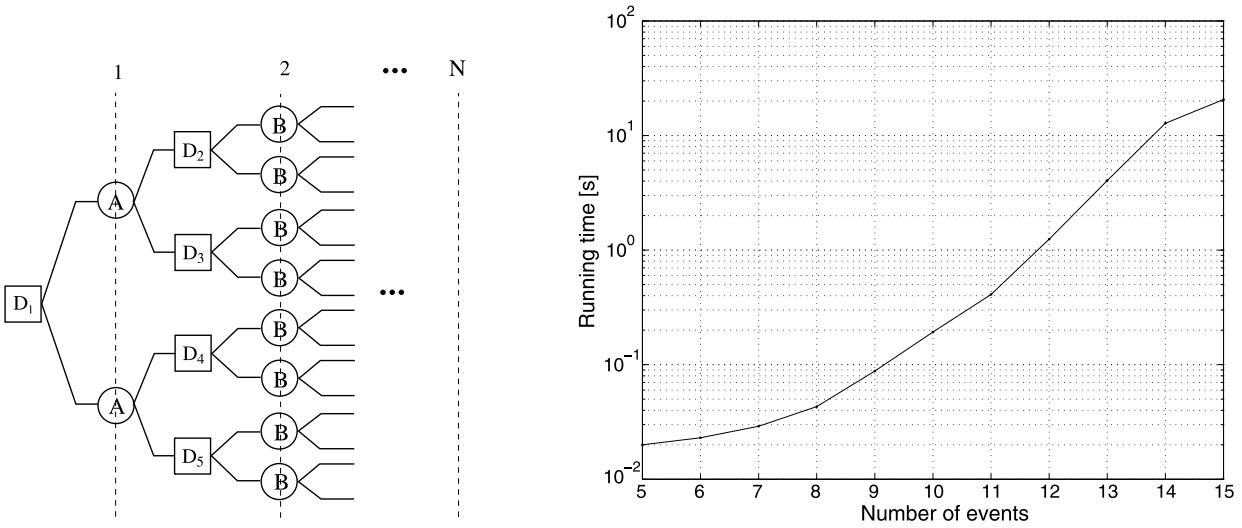


Fig. 6. Linear programming solution for exponentially large, separately specified decision trees. Left: Symmetric decision tree. Right: Running times, in logarithmic scale, for growing N (number of events).

generation, it is possible to solve this linear program by manipulating only a smaller sub-matrix $\mathbf{A}'$ of $\mathbf{A}$ with as many rows and columns as there are rows in $\mathbf{A}$. The challenge is that, to run the simplex method, at each iteration we must select one of the previously discarded columns of $\mathbf{A}$ to replace an existing column in $\mathbf{A}'$. This is done by computing the reduced cost $c - y\mathbf{A}$, where y is the dual cost of the current solution [4]. We can write every column of $\mathbf{A}$ as a vector of multilinear expressions $[A_1B_1 - \alpha_1 B_1, \ldots, A_m B_m - \alpha B_m]^T$, where A_i is an event and B_i is a conjunction of events, and α_i is an assessment. Thus $c - y\mathbf{A}$ is a multilinear expression on optimization variables that are either 0 or 1; there are standard techniques to reduce such an optimization problem to integer programming [18, Section 4.2]. To summarize: we run the simplex method with $\mathbf{A}'$, and to decide which column of $\mathbf{A}$ to enter into $\mathbf{A}'$, we run an auxiliary integer program. We note that problems with hundreds of variables in probabilistic logic have been solved using column generation with relative ease [32], so the exact solution of large decision trees can be obtained. Note also that these techniques can be extended to chance nodes that are associated with random variables with finitely many values. The added effort is to binarize the random variables into sets of binary variables, and to add Boolean constraints so that these binary variables only take on possible values. The result is again a probabilistic logic problem that can be solved using linear programming, possibly with column generation if N is large.

4.4. Consequentialist backward induction...?

One might argue that constraints $\mathcal{K}$ should be generated only once before any other computation in Algorithm 5, as assessments $\mathcal{T}.K$ are available as input. However such an approach may miss significant simplifications, because there may be chance nodes that are discarded during execution of the algorithm, given our consequentialist perspective. For instance, in Fig. 5 the event F can be discarded when one is at D_1, because action a_2' and the subsequent nodes are not admissible for any criteria of choice previously discussed. Thus it makes sense to generate constraints "inside" the loop (line 10) in Algorithm 5. Nevertheless, in the worst case the function GenerateConstraints may build, if implemented as described in the previous section, an exponentially large optimization program at the root node. This is somewhat unsatisfying, particularly when compared to backward induction in standard decision trees.

In a standard decision tree, an already processed decision node is completely summarized by the unique expected value of the selected substrategy from that node on. Instead in the function GenerateConstraints described in the previous section, the programs built at decision nodes grow in size. The natural question is: Can we have a function GenerateConstraints that summarizes the already processed decision nodes through an interval of expected utility? For instance, in Example 6 we would like the choice between a_1 and a_2 to be resolved at D_1 only by processing an expectation interval from D_2. Alas, such an interval-based backward induction fails in general:

Example 7. Consider the decision tree in Fig. 7, adapted from Hammond [30]. Here $p \in [\epsilon, 1-\epsilon]$ and $q = \epsilon$, for some small $\epsilon > 0$. Actions a and b are maximal and E-admissible at D_2, and c and d are maximal and E-admissible at D_3, but strategy $\{a', a\}$ dominates $\{a'', d\}$, and $\{a'', c\}$ dominates $\{a', b\}$. If D_2 and D_3 were to return the expectation intervals for their maximal/E-admissible actions, it would not be possible to detect that some strategies are dominated; in fact, all four strategies have overlapping expectation intervals.

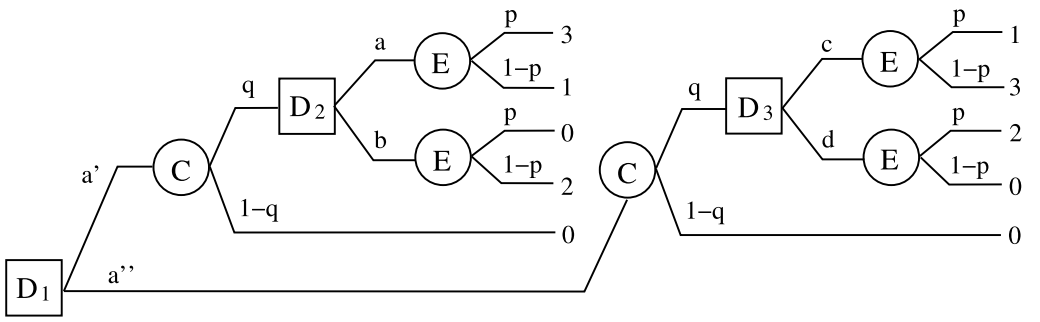


Fig. 7. Decision tree for Example 7.

Maximality and E-admissibility may fail in an interval-based backward induction because the necessary information about the constraints on probability values may be lost. We now wish to show that an interval-based backward induction can succeed for Γ-Maximin, Γ-Maximax, Γ-Maximix and Interval Dominance. That is, for these criteria and under the assumption that constraints are separately specified, we can evaluate actions at a decision node D as a one-step decision problem where each decision node reached from D is replaced by a single expectation interval. We start by rehearsing, in the next paragraph, an argument by Danielson and Ekenberg [14], who derived algorithms for computation of lower/upper expectations of a fixed strategy in the presence of local bounds on probabilities and expectations.

The central idea by [14] is as follows. Fix a strategy s; using Expression (2),

$$\underline{E}[s] = \min_{P_1,\ldots,P_N} \sum_{X_1} \cdots \sum_{X_N} P_1\big(X_1\big|\mathrm{an}(X_1)\big)\ldots P_N\big(X_N\big|\mathrm{an}(X_N)\big)U(s, X_1, \ldots, X_N),$$

where we have subscripted the probability distributions so as to emphasize the scope of the minimization. We can move the summations to the right, eliminating variables one by one from X_N to X_1, and we can place the minimization over a probability distribution right before the probability distribution of interest; this is only possible because all constraints are local and independently specified for each path from root to the node of interest. Hence to compute the lower expectation of s, we can run a backward induction scheme where a *reduced* optimization program is built at each decision node D by encoding constraints in the subtree rooted at D, with action selected by s, and with branches cut at future decision nodes. These latter decision nodes are replaced by their lower and upper expected utilities; and as D produces its own lower and upper expected utilities, it passes only these values back to its ancestors. Reduced programs are multilinear due to the presence of probability constraints *and* expectation intervals; that is, in Expression (2) we have both values of P_i and of U as free variables to optimize for. Danielson and Ekenberg [14] present techniques that simplify the solution of these local multilinear programs.

Returning to our problem, consider the criterion of Interval Dominance. At decision node D we can build a complete optimization program with all chance nodes in the subtree rooted at D, except those nodes in subtrees eliminated in previous stages of the backward induction method. Alternatively, we can build a reduced optimization problem with the chance nodes between D and the decision nodes that are direct descendants, with the proviso that these descendants summarize the content of their subtrees by intervals of expected utility. Recall that each decision node that is direct descendant of D is attached to a fixed set of substrategies, because our decision maker is consequentialist. Thus the only question is whether the set of I-admissible actions at D is the same regardless of whether we use the complete or the reduced program. Given that all that matters for Interval Dominance are lower/upper expectations, the argument in the previous paragraph leads to a positive answer: the two programs produce identical results, as the complete program can be divided into several smaller programs that are all encoded in the reduced program and its expectation intervals.

Similar arguments work for Γ-Maximin, Γ-Maximax and Γ-Maximix. Each descendant D' of D is attached to a single selected substrategy s' rooted at D'. Every expected utility in the expectation interval for s' can be attained by selecting probability distributions in the subtree rooted at D'. Consequently, at D we lose nothing by restricting attention to the expectation interval of s', and likewise for every descendant of D.

To finish this section, we compare several kinds of assessments and criteria of choice in the context of Example 3:

Example 8. Consider Example 3, and suppose ϵ is small. Algorithm 5 starts at D_2, where three substrategies, corresponding to actions a'_1, a'_{2a} and a'_{2b}, are evaluated by the appropriate function Criterion_CRITERION. The same happens at node D_3. At node D_1, the actions a_1, a_{2a}, a_{2b} are evaluated, together with all combinations of selected strategies from D_2 and D_3. We have the implicit constraint that a single value p refers to several probability values ($P(E|C, D_1)$, $P(E|C^c, D_2)$, etc.); that is, constraints are not separately specified. Running the complete backward induction algorithm, we obtain the following selected strategies. The Γ-Maximin actions are a'_1 and a''_1 at D_2 and D_3, and the Γ-Maximin strategy is either (a_{2a}) or (a_{2b}) at D_1. (Note that action a'_{2a} is inadmissible at D_2, action a''_{2b} is likewise inadmissible at D_3, but their combination is identical to a_{2a}, a Γ-Maximin action at D_1!) The Γ-Maximax criterion prescribes a'_{2a} and a''_{2b} at D_2 and D_3, and then

(a_3, a'_{2a}, a''_{2b}) at D_1. Interval Dominance selects all actions at nodes D_2 and D_3, so we have twelve strategies to evaluate at D_1, five of which are inadmissible ((a_1), (a_{2a}), (a_{2b}), (a_3, a'_1, a''_1) and (a_3, a'_{2b}, a''_{2a})). E-admissibility discards only a'_1 and a''_1 at D_2 and D_3, while Maximality does not discard any of them. At D_1 Maximality discards the same strategies as Interval Dominance plus the strategies (a_3, a'_1, a''_{2a}) and (a_3, a'_{2a}, a''_1). Finally, at D_1 E-admissibility discards the same strategies as Maximality plus the strategies (a_3, a'_1, a''_{2b}) and (a_3, a'_{2a}, a''_1). There are three E-admissible strategies: (a_3, a'_{2a}, a''_{2a}), (a_3, a'_{2a}, a''_{2b}), and (a_3, a'_{2b}, a''_{2b}).

Suppose we change Example 3 so that the probability of E depends on the path from root to the chance node labeled with E; that is, we have eight probability values $p_i \in [0.25, 0.75]$ (constraints are separately specified). A linear programming solution is now possible. However, if we wish we can deal with reduced programs through multilinear programming. Note that we face a decrease in selectivity by separating assessments: the *only* inadmissible strategy for Interval Dominance, Maximality and E-admissibility is (a_1).

To summarize the discussion on decision trees: the same strategies are selected under resolute and consequentialist norms for Maximality and E-admissibility, and there, a backward induction procedure is fully justified; for the remaining criteria, a backward induction method is only justified by a consequentialist position (as adopted in this paper). Another point is that the size of optimization programs generated by Algorithm 5 may grow exponentially; however for Γ-Maximin, Γ-Maximax, Γ-Maximix and Interval Dominance it is possible to run reduced programs when constraints are separately specified. And finally, a linear programming formulation is possible in some cases but multilinear programming is required in general, and in particular when dealing with reduced programs.

5. Influence diagrams with partially ordered preferences

Decision trees can hardly represent large, or even medium size, decision problems, as the number of nodes in a decision tree increases exponentially with the number of chance and decision variables. A more compact way to represent sequential decision problems is through influence diagrams. A seminal work on influence diagrams with interval probabilities was presented by Breese and Fertig [8,22]; no substantial advance seems to have appeared in the literature after that work. We now expand that analysis by considering several criteria of choice.

The section is organized as follows. In Section 5.1 we introduce influence diagrams with partially ordered preferences, we define a class of problems that we are interested in, we present an algorithm to solve such class of problems and provide an analysis of complexity for the algorithm. In Section 5.2 we discuss several examples and experiments.

5.1. Preliminaries, strategy selection and algorithm

An influence diagram with imprecise probabilities is a directed acyclic graph over a set of decision nodes $\mathbb{D}$ (square shaped), chance nodes $\mathbb{C}$ (circle shaped) and utility nodes $\mathbb{U}$ (diamond shaped). Edges into a chance node indicate stochastic dependence; edges into a decision node indicate the available information at the time of the decision; edges into a utility node indicate functional dependence. Each decision node is associated with a finite set of actions conditional on its parents. Each chance node is associated with a random variable C and with a set of credal sets: for each instantiation π_i of the parents of C ($\mathrm{pa}(C)$), we have a credal set $K(C|\mathrm{pa}(C) = \pi_k)$ specified as in credal networks[5]; each utility node U is associated with a function $u(\mathrm{pa}(U))$ that depends only on the parents of U. If more than one utility node is specified, then the total utility is the sum of all functions in utility nodes [68]. The standard definition of influence diagrams [34] requires a linear temporal order of all decisions (typically represented by a directed path comprising all decision nodes) and the *no forgetting* assumption, that is, at each decision node the decision maker knows all her previous decisions and past observations. However, some past information may be irrelevant and should not be considered for computational reasons [53,64]. In *Limited Memory Influence Diagrams (LIMIDs)* [44], the no forgetting assumption is relaxed, that is, the decision maker knows only the past decisions and observations that are explicitly linked to the decision nodes. This allows the representation of a broad class of decision problems, including situations with many decision makers.

A *policy* δ_D for decision node D is a mapping from the parents of D to the possible actions in D. A *strategy* s is an ordered set of prescribed actions for all decision nodes, where each action depends on the parents of the decision node; that is, an ordered set of policies $s = \{\delta_{D_1}, \ldots, \delta_{D_n}\}$. The expression of expected utility for a strategy s, for a fixed probability distribution P, is

$$\sum_{U \in \mathbb{U}, X \in \{\mathbb{C}, \mathbb{D}\}} \left(u\big(\mathrm{pa}(U)\big) \prod_X P\big(X|\mathrm{pa}(X)\big) \right), \tag{4}$$

where we note that if $X \in \mathbb{D}$, then its value is fixed by strategy s (the variable is associated to zero/one probabilities given s). In standard influence diagrams and standard LIMIDs, an *optimal* strategy is a strategy with maximum expected utility (such strategies are called *global maximum strategies* in LIMIDs [44]).

It is important to pause for a moment and consider the properties of LIMIDs. First, any influence diagram is a LIMID where a decision node is informed about all previous decisions. However, if a LIMID contains a small number of arcs into

[5] In this paper all variables have finitely many values.

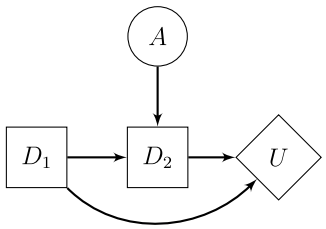


Fig. 8. A simple influence diagram with too many strategies.

decision nodes, the number of possible strategies is small when compared to the number of strategies in an influence diagram with identical graph. Hence the number of arcs into decision nodes is a critical parameter in a LIMID. Another important property of LIMIDs is that decision nodes are not necessarily ordered, so a decision maker contemplating a particular decision node may have no clue as to which decisions are implemented already and which decisions are still to be reached.

The lack of ordering amongst decisions in LIMIDs brings about a point that seems to have been missed in the literature. Namely, that LIMIDs are intrinsically inappropriate for consequentialist behavior. Clearly a decision maker can enumerate all strategies in a LIMID and then select a strategy with maximum expected utility, presumably to follow it all the way (in a resolute manner). A consequentialist behavior is harder to describe in the context of a LIMID. Suppose a decision maker seats at a decision node, considering only future moves in an attempt to evaluate its current decision; that is, in a consequentialist position. But how is this decision maker to know what are the "future" moves in a LIMID? There may be decisions that are not ordered with respect to the current decision, and the only way to examine the relative value of strategies is to consider all possible orderings. This may only be possible by considering the set of strategies from the outset, as a resolute decision maker would do. Indeed, the popular algorithm *Single Policy Update* (SPU) [44] finds non-optimal strategies in LIMIDs by updating policies in some given order; the resulting strategies are not guaranteed to be optimal and, more importantly, the whole reasoning behind SPU cannot be given a consequentialist justification. For this reason, we do not attempt in this paper to adapt SPU to LIMIDs with imprecise probabilities; quite the contrary, we use a direct multilinear formulation of the strategy-selection problem. Note that a version of SPU for indecision-resistant criteria is not so difficult to conceive (as every decision node must yield a single policy) but a version of SPU for indecision-prone criteria seems not to be possible.

For these reasons, in this paper we are interested only in those LIMIDs that have a temporal order for decisions, so that we can always consider consequentialist behavior. Such an assumption about LIMIDs clearly limits the scope of models we can use, but it should be noted that the resulting class of LIMIDs is substantially larger than the class of influence diagrams; even though there is an ordering on decisions, the set of strategies that is allowed for a LIMID does not necessarily require each decision node to be aware of all previous decisions in the ordering. That is, the number of possible strategies in the LIMID may be substantially smaller than the number of strategies in an influence diagram of identical structure. In fact, the reason why we focus on "ordered" LIMIDs is exactly so that we can limit the number of possible strategies as compared to influence diagrams proper.

Example 9. Consider Fig. 8. Suppose that A has four possible outcomes and D_1 has four possible actions. A policy for D_2 specifies one action for each configuration of the parents. If D_2 has two possible actions, then there are 2^{16} policies. In general, if we have m configurations and n actions, we can generate n^m policies. This example shows that the number of policies grows quickly (exponentially) when decision nodes depend on several parents. In this case the search for optimal strategies may easily become intractable. For instance, suppose we intend to apply the Maximality criterion, then we have a total of n^{m^2} optimization programs to evaluate.

As noted, our approach to strategy selection is to use a (consequentialist) backward induction process, rather than to resort to a variant of SPU or any other approximate scheme. The algorithm proceeds from the last decision node up to the first, building the strategies during execution by selecting admissible actions for each configuration of $\mathrm{pa}(D)$, and then combining only the selected actions. The subtle point is that, for indecision-prone criteria, we may have to consider more than one admissible policy for a decision node D as long as the algorithm iterates. This approach can save computations in two ways: (1) the computations are done locally (we do not need to consider all variables on the graph, thus, the optimization programs are smaller), and (2) if the number of selected actions is smaller than n, then we reduce the number of possible policies to be considered. Algorithm 6 summarizes this idea. Once the fact that we adopt a consequentialist position is understood, the algorithm can be viewed as a mix of the algorithm for policy selection in standard influence diagrams (where consequentialism is natural) and our previous algorithms for decision trees.

In Algorithm 6, we keep track of a list $\mathcal{S}_i$ associated to decision node D_i. The list $\mathcal{S}_i$ is used to hold the set of (sub)strategies suggested by indecision-prone criteria. It contains all admissible substrategies, evaluated by CRITERION, rooted at D_i, $\mathcal{S}_i = \{\delta_{D_i}, \ldots, \delta_{D_m}\}$. The initialization in line 2 indicates that previous to the first iteration there is no substrategy. As

Algorithm 6: InfluenceDiagram_CRITERION

Input: An influence diagram $\mathcal{D}$ as described in Section 5.1, with a set $\mathcal{D}.K$ of constraints on probability values.
Output: A set of strategies selected by CRITERION.

```
1  m ← Number of decision nodes;
2  S_{m+1} ← ∅;
3  for D_i ∈ 𝔻, i ← m until 1 do
4      𝔾_i ← Required variables to evaluate actions in D_i;
5      A_i ← CombineActions(D_i, S_{i+1});
6      foreach configuration π_k of pa(D_i) do
7          K_i ← GenerateConstraints(D_i, A_i, π_k, 𝔾_i, D.K);
8          Admissible[π_k] ← Criterion_CRITERION(A_i, K_i);
9      S_i ← CombineSubstrategies(Admissible, S_{i+1});
10 return Criterion_CRITERION(S_1, K_1);
```

we have pointed out before, to evaluate an action in a given decision node, we do not need to consider all variables in the graph. This is exactly what we do in line 4 of Algorithm 6. First we note that a utility node U is *relevant* to a decision D if there exists a directed path connecting D and U. Then we use a standard d-separation algorithm [63] to obtain the needed variables, that is, the set of variables that are not d-separated from the set $\mathbb{U}_i$ of utility nodes relevant to D_i, given that the set $\{D_i, \text{pa}(D_i)\}$ are "observed" (the decision node is clamped to the selected action).[6] The function CombineActions takes the list of substrategies $\mathcal{S}_{i+1}$ and attaches an action of D_i to each $s \in \mathcal{S}_{i+1}$. Suppose that in Example 9 there are two admissible policies in D_2, then at decision node D_1 the function CombineActions returns to $\mathcal{A}_i$ eight possible combinations. These combined actions will be evaluated by the criteria of choice before building the policies for decision node D_i. We rely on the criteria for reducing the number of admissible policies. This is done in the inner loop.

The function GenerateConstraints is responsible for creating the constraints on probabilities that are passed to the function Criterion_CRITERION. This function must encode the state space, as done for decision trees, and then encode constraints on probability values based on the input constraints $\mathcal{D}.K$. Differently from decision trees, the constraints in $\mathcal{K}_i$ must take into account the fact that lower/upper expectations are now: (1) restricted to variables in $\mathbb{G}_i$; and (2) conditional on a set of "observed" nodes $\{D_i, \text{pa}(D_i)\}$. Thus the new element here is that we must minimize/maximize conditional expectations; this is done by introducing an auxiliary variable z and a new constraint,

$$\sum_{\mathbb{U}_i, \mathbb{G}_i \setminus \mathbb{E}_i} P(\mathbb{G}_i) u(\mathbb{U}_i) - z P(\mathbb{E}_i) = 0, \tag{5}$$

where $\mathbb{E}_i$ denotes the set of nodes "observed" at D_i. This constraint forces z to be the desired conditional expectation. Hence the inner loop in Algorithm 6 builds the optimization programs as in decision trees, using Expression (5) in symbolic form whenever necessary.

The function CombineSubstrategies is responsible for building $\mathcal{S}_i$. It receives the set of admissible actions and the set of substrategies S_{i+1}, builds the possible policies for D_i, and appends them to the substrategies in S_{i+1}.

The complexity of the algorithm obviously depends on the criteria of choice. For indecision-resistant criteria, we always consider one optimal policy at each decision node and, consequently, we have only one strategy (similar to influence diagrams with precise probabilities). This is also the best case for indecision-prone criteria (when the criterion is very selective). The worst case happens when the criterion does not discard any action. This implies that we need to consider all possible combination of policies as long as we proceed in a backward fashion, and at the first decision node we have the same amount of strategies as a resolute decision maker.

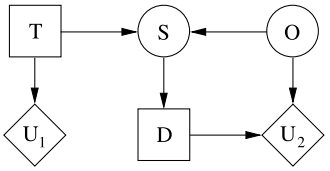
5.2. Examples and experiments

The algorithm presented in the previous section has been implemented and run in several well-known examples. Most of the implementation was coded in the Java language, with calls to optimization packages (the multilinear programming package Multilin [17] and the CPLEX and Minos commercial packages respectively for linear and nonlinear optimization). Tests were run in a microcomputer with two dual-core processors and 4 GBytes of memory. We report the character of selected strategies and the computational effort spent to select these strategies.

We start with the classic oil wildcatter problem [56]; this is a small influence diagram, so we can indicate the steps of the algorithm in some detail.

Example 10. The oil wildcatter problem is depicted in Fig. 9. An oil wildcatter must decide whether to drill or not to drill (decision D). The cost of drilling is \$70K. If the decision is to drill, the soil may be soaking, wet or dry (with a return

[6] Due to their independence relations, influence diagrams and LIMIDs can be viewed as extended Bayesian networks [10,63,64]. Our framework can be viewed as an extended credal network, so d-separation applies.



Test (S)	Amount of oil (O) dry	wet	soaking
ns	[0.60, 0.65]	[0.30, 0.35]	[0.10, 0.10]
os	[0.25, 0.30]	[0.40, 0.40]	[0.40, 0.45]
cs	[0.10, 0.10]	[0.25, 0.30]	[0.45, 0.50]

Fig. 9. Influence diagram for the oil wildcatter problem and probability intervals for seismic test given amount of oil.

of \$270K, \$120K or \$0 respectively). Suppose probabilities for the amount of oil (O) are: $P(O = \text{soaking}) \in [0.2, 0.2]$, $P(O = \text{wet}) \in [0.3, 0.35]$, $P(O = \text{dry}) \in [0.45, 0.5]$. At the cost of \$10K, the oil wildcatter can take seismic soundings of the site. The result of this test (S) may be ns (no oil), os (some oil), cs (abundance of oil), with interval probabilities in Fig. 9. If the test is not conducted, then $P(S = s|O) = 1$ if $\{S = \text{nt}\}$ and $P(S|O) = 0$ otherwise, where nt is a special value of S that indicates absence of test.

To select strategies, we start with the decision D; node U_2 is required and all nodes but U_1 are returned by GetdConnected. The expected utility of not drilling is 0.00 regardless of S. The expected utility of drilling depends on S. To compute the lower expected utility of drilling given that $\{S = \text{ns}\}$, we must minimize auxiliary variable z subject to constraints on probabilities and to

$$\sum_{O} P(O)P(S = \text{ns}|O, T)u(D = \text{yes}, O) - zP(S = \text{ns}) = 0.$$

Running this and similar multilinear programs, we obtain:

S	nt	ns	os	cs
$\underline{E}[D = \text{yes}\|S]$	20.00	−32.76	32.86	82.61
$\overline{E}[D = \text{yes}\|S]$	26.00	−21.27	50.00	91.29

Using the Γ-Maximix criterion with $\eta = 0.5$ we find that the action not to drill is not admissible, except when the seismic test indicates ns. Thus we have the policy $\delta_D^*(S) = \text{ns}$ if $\{S = \text{no}\}$ and $\delta_D^*(S) = \text{yes}$ otherwise. As the Γ-Maximix criterion specifies only one policy in D, we have only two policies to analyze at T ($\delta_T = \text{yes}$ and $\delta_T = \text{no}$). The overall utility is given by the sum of U_1 and U_2; we then obtain $E[T = \text{yes}] \in [-10K, -10K] + [31.75K, 37.23K] = [21.75K, 27.23K]$ and $E[T = \text{no}] \in [0, 0] + [20K, 26K] = [20K, 26K]$. The selected action at T is to take the seismic test ($\delta_T^* = \text{yes}$). The selected strategy is $s^* = \{\delta_T^*, \delta_D^*\}$. This strategy is also selected by the Γ-Maximin and Γ-Maximax criteria. With E-admissibility, we obtain at D the same policy suggested by the Γ-Maximix criterion. At node T, we have two E-admissible policies, $\delta_T = \text{yes}$ and $\delta_T = \text{no}$. The combination of optimal policies leads to two strategies with expected utilities respectively in $[21.75K, 27.23K]$ and $[20.0K, 26.0K]$. Interval Dominance and Maximality select the same strategies as E-admissibility.

The next influence diagram we examine is the "Breeding Pigs" problem described by Lauritzen and Nilsson [44], and represented by the influence diagram in Fig. 10. A pig breeder is growing pigs for a period of four months and subsequently selling them. During this period the pig may or may not develop a certain disease (h_i represents the pig's health, healthy or ill, in the ith month). Once a month, a doctor makes a test for the presence of the disease (t_i represents the test's results, disease free or otherwise), and the doctor may or may not treat the pig for the disease by injecting a certain drug (decision node d_i). The utility nodes u_1, u_2, u_3 represent the cost of treating the pig, and u_4 represents the payoff for selling the pig. Additionally, we have: the price of a pig with disease is 300DKK (Danish kroner) and of a disease-free pig is 1000DKK (utility node u_4); the cost of an injection is 100DKK (utility nodes u_1, u_2, u_3); the test is correct when the pig is ill with probability 0.80, and correct when the pig is healthy with probability 0.90 (chance nodes t_i); a healthy pig develops the disease in the subsequent month with probability 0.20 without injection, whereas a healthy and treated pig develops the disease with probability 0.10; an untreated pig that is unhealthy will remain so in the subsequent month with probability 0.90, whereas the similar probability is 0.5 for an unhealthy pig that is treated (chance nodes h_2, h_3, h_4). The decision maker is uncertain about the pig's health in the first month (h_1). In our experiments we assume two intervals for h_1: the first one is a small interval defined near the probability of the original problem ($P(\text{ill}) = [0.1, 0.2]$); the second one is a large interval ($P(\text{ill}) = [0.0, 0.5]$).

In this example the full previous treatment and test history are available when decisions are made. At decision node d_3, there are 5 conditioning nodes. A policy at d_3 specifies an action for 32 configurations (all five conditioning nodes have just two possible values). An indecision-resistant criterion must run 64 inferences to find out the best policy, while an indecision-prone criterion may have to keep track of many incomparable substrategies. It is indeed possible that the large number of incomparable substrategies makes it impossible to finish the algorithm, as we will see in a moment.

Now, consider a LIMID for the Breeding Pigs problem, where the decision maker does not remember past decisions and results of tests from previous months (the decision maker remembers only the result of the test taken in the current month). The resulting LIMID is also depicted in Fig. 10.

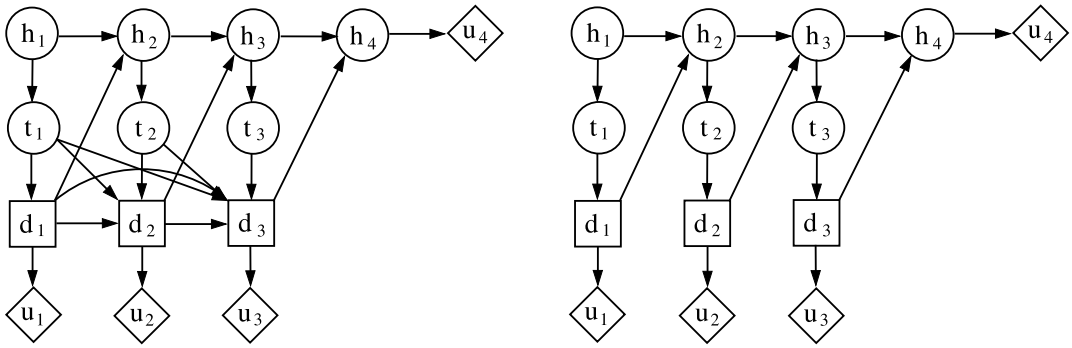


Fig. 10. Left: Influence diagram (with complete history) for the Breeding Pigs problem. Right: LIMID version for the Breeding Pigs problem.

Table 1
Experiments with the Breeding Pigs problem.

$P(h_1 = \text{ill})$	Criteria	# of admissible strategies LIMIDs	Elapsed time (s) LIMIDs	# of admissible strategies IDs	Elapsed time (s) IDs
[0.1, 0.2]	Γ-Maximin	1	2.41	1	10.20
	Γ-Maximax	1	2.63	1	11.45
	Γ-Maximix	1	3.59	1	16.03
	I. Dominance	2	5.35	16	89.95
	Maximality	1	1.84	2	17.59
	E-admissibility	1	2.39	2	23.58
[0.0, 0.5]	Γ-Maximin	1	3.74	1	16.48
	Γ-Maximax	1	5.27	1	32.23
	Γ-Maximix	1	7.57	1	43.39
	I. Dominance	8	23.42	–	–
	Maximality	2	3.76	–	–
	E-admissibility	1	4.14	7	584.64

Table 1 presents results for all criteria we have discussed. The column labeled *# of admissible strategies* reports the number of strategies selected by our implementation, and the column labeled *Elapsed time* shows the average time of execution over thirty runs for each criterion. An interesting fact is that the smaller probability interval leads to smaller execution times: the sharper the probabilities, the smaller the number of incomparable substrategies to process. Another point to note is that Interval Dominance and Maximality could not be run to termination for the larger probability interval, due to the huge number of strategies that these criteria fail to discard during execution. A curious fact is that E-admissibility does not crash the system, and indeed leads to a relatively small number of selected strategies: even though E-admissibility seems more complex computationally, the fact that it is more selective than Interval Dominance and Maximality is quite valuable in practice.

Another curious fact in Table 1 is that, with the LIMID, Maximality and E-admissibility seem to be faster than the indecision-resistant criteria. This result can be traced to a few computational aspects that are not apparent from a superficial analysis. As we can define the set of maximal/E-admissible actions without computing exact probability values (just find a distribution satisfying the constraints), we use a fast approximated solver in Minos to produce a preliminary selection of actions. After this, we use the exact solver Multilin to compute lower/upper expectations. The approximated solver quickly gets close to the exact solution, so the overall computing time is greatly benefited. It is also possible to explain the weak showing of Interval Dominance, as it requires the use of the exact solver to compute lower and upper expectations so as to compare actions.[7] It is noteworthy that Interval Dominance is a simple criterion to apply in single-stage decision problems while in sequential decision problems it faces difficulties due to its low selectivity.

Our final example deals with a relatively large LIMID that has been proposed by de Campos and Ji [19] as a model for Effects-Based Operation planning (EBO). The LIMID is shown in Fig. 11; all variables are binary, and the decision nodes have two possible actions (yes/no). The cost of actions, given by $\{U_i\}_{i=1}^{11}$, is: if D_i is yes, then cost is 50 for U_3, 150 for U_8, 80 for U_{10}, 100 for U_{11} and 20 for the all others. The reward for achieving the main goal is 1000, while not achieving it costs 500 (utility node U_H). The chance nodes C_i represent the rate of success with an interval probability $P(C_i = 1|D_i = \text{yes}) \in [0.9, 1.0]$. The chance nodes B_j and A_k have probability 1 if all parents are positive, probability 0.5 if only one parent is positive, and probability 0 otherwise. The probability of chance node G is 1 given that its parents

[7] If we use the solver Minos to find I-admissible actions, Interval Dominance takes 2.68 and 56.70 seconds in the LIMID and influence diagram respectively, when $P(h_1 = \text{ill}) \in [0.1, 0.2]$; and it takes 9.45 seconds in the LIMID, and crashes with the influence diagram, when $P(h_1 = \text{ill}) \in [0.0, 0.5]$.

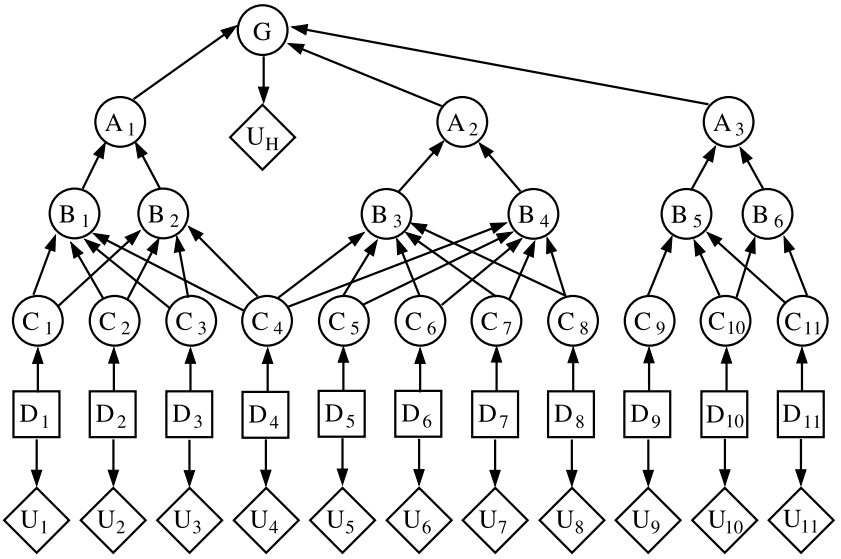


Fig. 11. Influence diagram for the EBO problem.

are positive, 0.6 if only one parent is negative, 0.3 if two parents are negative, and 0 otherwise. Decision nodes have no parents (an extreme LIMID), hence an arbitrary ordering of decision nodes is adopted when selecting strategies. Using the Γ-Maximin criterion, the selected strategy is to take action yes in all decision nodes except D_5, D_6, D_7 and D_8 (approximately 40 seconds were taken to select it). The lower expected utility of this strategy is -55.28; the elapsed time to select it was approximately 40 seconds. The only E-admissible strategy is to take action yes in all decision nodes except D_8 (approximately 120 seconds were taken to select it). The expected utility of this strategy belongs to $[68.97, 330]$; note that the consequentialist E-admissible strategy is *always* better than the consequentialist Γ-Maximin strategy.[8]

6. Conclusion

This paper has examined the selection of strategies in sequential decision making when preferences are partially ordered. In particular, we have focused on preference patterns that are encoded through a single utility and a set of probability measures. A partial order over strategies introduces subtle ingredients into the decision problem, as we have several criteria of choice and behavioral norms, episodes of incoherent/inconsistent choice, and varying degrees of computational gain. We have tried to shed some light into these matters from a consequentialist perspective, and to present algorithms that select strategies by solving sequences of optimization programs. Most algorithms employ multilinear programming, and some particular cases can be tackled by linear programming. Clearly, algorithms based on multilinear programming can be adapted to handle interval-valued utility, as we then have products between probabilities and utilities that must be optimized over; we have refrained from discussing interval-valued utility so as to limit the length of the paper.

The current literature on sequential decision making with partially ordered preferences can be roughly divided in two streams. The philosophical debate tends to favor abstract comparisons amongst criteria and norms, with little consideration of computational costs. On the other hand, if one looks at techniques that do involve decision making with partially ordered preferences, such as nondeterministic planning and CP nets, and even the theory of LIMIDs, one finds detailed study of computational costs but little attention to criteria, norms, and consistency. We hope that this paper strikes some needed balance between conceptual discussion and computational development, and helps shorten some of the gaps between these viewpoints. In particular we believe that the effect of the consequentialist perspective is currently not appreciated in the artificial intelligence literature, exactly where this norm is most adequate as one must deal with bounded agents.

We can summarize our contributions as follows. We have first derived new algorithmic techniques for Interval Dominance and E-admissibility (using insights by Kyburg and Pittarelli [43], as done independently by Utkin and Augustin [72]). We also presented a brief analysis of incoherent choice with Interval Dominance. More importantly, we have studied decision trees with partially ordered preferences, by presenting a consequentialist backward induction framework with multilinear and linear programming instantiations, and by noting that different criteria do affect the computational properties of backward induction. We have then applied these insights to influence diagrams, and actually to ordered LIMIDs, where the technology of credal networks (d-separation and multilinear programming) is used as much as possible. We have

[8] By brute-force enumeration of strategies, we find that there is an E-admissible strategy with lower expected utility 156.41 (and upper expected utility 480.00); the consequentialist approach misses this possibility.

also presented experiments with a complete implementation of the algorithms for influence diagrams. Given the lack of literature on influence diagrams with indeterminacy and imprecision in probability values, our results are a first step in understanding this powerful but intricate model. We have noted already that recently Huntley and Troffaes [35] and de Campos and Ji [19] have presented specialized algorithms respectively for Maximality in decision trees and for Γ-Maximax in influence diagrams, that can be more efficient than the algorithms discussed in this paper in particular problems.

An important subject this paper has not tackled is Markov Decision Processes (MDPs). MDPs are probably the most prominent model for sequential decision making under uncertainty in use in artificial intelligence today, and could likewise benefit from partial preference orderings. It should be noted that the initial translation into the framework of MDPs is not difficult, in fact there is a considerable amount of publications that deal with MDPs with sets of probability (usually referred to as Markov Decision Processes with Imprecise Probabilities) [33,58,74]. The topic deserves an investigation of its own; additionally, there are many questions that must be answered before partial preferences can be extensively used in MDPs, such as how to deal with act-state dependence and how to choose between the set of incomparable choices suggested by criteria such as Maximality and E-admissibility.

We certainly leave many avenues for future exploration. A necessary next step is a detailed empirical characterization of computational effort in solving decision trees and influence diagrams, including a comparison between multilinear and linear programming schemes whenever the latter are possible. It would also be important to characterize the class of influence diagrams that can be solved through *reduced* programs (that is, programs that deal with small subsets of chance and decision nodes, passing back interval-valued expected utility). There are also conceptual questions that deserve further scrutiny, as there are other criteria of choice and behavioral norms besides the ones investigated in this paper. Even within the scope of criteria discussed in this paper, there are questions to answer. For instance: we have produced algorithms that compute *all* E-admissible strategies; perhaps in practice one should be content with just one E-admissible strategy?

References

[1] M. Allais, O. Hagen, Expected Utility Hypotheses and the Allais Paradox, D. Reidel Publishing Company, Dordrecht, Holland, 1979.
[2] R.J. Aumann, Utility theory without the completeness axiom, Econometrica 30 (3) (July 1962) 445–461.
[3] J.O. Berger, Statistical Decision Theory and Bayesian Analysis, Springer, New York, 1985.
[4] D. Bertsimas, J.N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific, Belmont, Massachusetts, 1997.
[5] L. Blume, A. Brandeburger, E. Dekel, Lexicographic probabilities and choice under uncertainty, Econometrica 59 (1) (January 1991) 61–79.
[6] B. Bonet, R. Givan, in: 5th International Planning Competition: Non-deterministic Track, call for participation, 2005.
[7] C. Boutilier, R.I. Brafman, H.H. Hoos, D. Poole, CP-nets: A tool for representing and reasoning with conditional ceteris paribus preference statements, Journal of Artificial Intelligence Research 21 (2004) 135–191.
[8] J. Breese, K. Fertig, Decision making with interval influence diagrams, in: Sixth Conference on Uncertainty in Artificial Intelligence, Elsevier Science, New York, 1990, pp. 122–129.
[9] K. Bykvist, Time-partial morality and dynamic choice, in: W. Rabinowicz (Ed.), Value and Choice – Some Common Themes in Decision Theory and Moral Philosophy, Lund Philosophy Reports, 2000, pp. 53–64.
[10] G.F. Cooper, A method for using belief networks as influence diagrams, in: Proceedings of the 4th Conference on Uncertainty in Artificial Intelligence, Minneapolis, 1988, pp. 55–63.
[11] I. Couso, S. Moral, P. Walley, A survey of concepts of independence for imprecise probabilities, Risk, Decision and Policy 5 (2) (2000) 165–181.
[12] F.G. Cozman, Separation properties of sets of probabilities, in: C. Boutilier, M. Goldszmidt (Eds.), Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, San Francisco, July 2000, pp. 107–115.
[13] F.G. Cozman, Graphical models for imprecise probabilities, International Journal of Approximate Reasoning 39 (2–3) (June 2005) 167–184.
[14] M. Danielson, L. Ekenberg, Computing upper and lower bounds in interval decision trees, European Journal of Operational Research 181 (2) (September 2007) 808–816.
[15] M. Danielson, L. Ekenberg, J. Johansson, A. Larsson, The DecideIT decision tool, in: J.-M. Bernard, T. Seidenfeld, M. Zaffalon (Eds.), Proceedings of the 3rd International Symposium on Imprecise Probabilities and Their Applications, Carleton Scientific, Lugano, Switzerland, July 2003, pp. 204–217.
[16] C.P. de Campos, F.G. Cozman, Inference in credal networks using multilinear programming, in: Proceedings of the 2nd European Starting AI Researcher Symposium, IOS Press, Valencia, Spain, August 2004, pp. 50–61.
[17] C.P. de Campos, F.G. Cozman, The inferential complexity of Bayesian and credal networks, in: Proceedings of the 9th International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July–August 2005, pp. 1313–1318.
[18] C.P. de Campos, F.G. Cozman, Inference in credal networks through integer programming, in: International Symposium on Imprecise Probability: Theories and Applications, Prague, 2007, pp. 145–154.
[19] C.P. de Campos, Q. Ji, Strategy selection in influence diagrams using imprecise probabilities, in: Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence, Helsinki, Finland, July 2008, pp. 121–128.
[20] D. Ellsberg, Risk, ambiguity, and the Savage axioms, The Quarterly Journal of Economics 75 (4) (1961) 643–669.
[21] N. Etchart, Adequate moods for Non-EU decision making in a sequential framework, Theory and Decision 52 (February 2002) 1–28.
[22] K. Fertig, J. Breese, Probability intervals over influence diagrams, IEEE Transactions on Pattern Analysis and Machine Intelligence 15 (3) (1993) 280–286.
[23] P.C. Fishburn, Utility Theory for Decision Making, Kriefer Publishing Company, New York, 1970.
[24] G. Georgakopoulos, D. Kavvadias, C.H. Papadimitriou, Probabilistic satisfiability, Journal of Complexity 4 (1) (March 1988) 1–11.
[25] I. Gilboa, D. Schmeidler, Maxmin expected utility with non-unique prior, Journal of Mathematical Economics 18 (2) (1989) 141–153.
[26] F.J. Giron, S. Rios, Quasi-Bayesian Behaviour: A More Realistic Approach to Decision Making? University Press, Valencia, 1980.
[27] T. Hailperin, Best possible inequalities for the probability of a logical function of events, American Mathematical Monthly 72 (1965) 343–359.
[28] P.J. Hammond, Changing tastes and coherent dynamic choice, The Review of Economic Studies 43 (1) (1976) 159–173.
[29] P.J. Hammond, Consequentialism and the independence axiom, in: B.R. Munier (Ed.), Risk, Decision and Rationality (Proceedings of the 3rd International Conference on the Foundations and Applications of Utility, Risk and Decision Theories), Dordrecht, Holland, 1988, pp. 503–516.
[30] P.J. Hammond, Orderly decision theory: a comment on Professor Seidenfeld, Economics and Philosophy 4 (1988) 272–297.
[31] P. Hansen, B. Jaumard, Probabilistic satisfiability, Tech. Rep. G-96-31, Les Cahiers du GERAD, École Polytechique de Montréal, 1996.
[32] P. Hansen, S. Perron, Merging the local and global approaches to probabilistic satisfiability, International Journal of Approximate Reasoning 47 (2) (2008) 125–140.

[33] D. Harmanec, Generalizing Markov decision processes to imprecise probabilities, Journal of Statistical Planning and Inference 105 (1) (June 2002) 199–213.
[34] R.A. Howard, J.E. Matheson, Influence diagrams, Decision Analysis 2 (3) (2005) 127–143.
[35] N. Huntley, M. Troffaes, An efficient normal form solution to decision trees with lower previsions, in: International Workshop on Soft Methods in Probability and Statistics, 2008, pp. 419–426.
[36] L. Hurwicz, A class of criteria for decision-making under ignorance, Cowles Comission Paper 356, 1951.
[37] H. Itoh, K. Nakamura, Partially observable Markov decision processes with imprecise parameters, Artificial Intelligence 171 (8–9) (2007) 453–490.
[38] J.-Y. Jaffray, Rational decision making with imprecise probabilities, in: G.D. Cooman, F.G. Cozman, S. Moral, P. Walley (Eds.), Proceedings of the 1st International Symposium on Imprecise Probabilities and Their Applications, Ghent, Belgium, June 1999, pp. 183–188.
[39] B. Jaumard, P. Hansen, M.P. de Aragão, Column generation methods for probabilistic logic, ORSA Journal on Computing 3 (2) (1991) 135–148.
[40] D. Kahneman, A. Tversky, Prospect theory: An analysis of decisions under risk, Econometrica 47 (1979) 262–291.
[41] D. Kikuti, F.G. Cozman, Influence diagrams with partially ordered preferences, in: 3rd Multidisciplinary Workshop on Advances in Preference Handling, 2007.
[42] D. Kikuti, F.G. Cozman, C.P. de Campos, Partially ordered preferences in decision trees: Computing strategies with imprecision in probabilities, in: Workshop on Advances in Preference Handling, Edinburgh, United Kingdom, July 2005, pp. 118–123.
[43] H.E. Kyburg Jr., M. Pittarelli, Set-based Bayesianism, IEEE Transactions on Systems, Man and Cybernetics, Part A 26 (3) (1996) 324–339.
[44] S.L. Lauritzen, D. Nilsson, Representing and solving decision problems with limited information, Management Science 47 (9) (2001) 1235–1251.
[45] I. Levi, On indeterminate probabilities, The Journal of Philosophy 71 (1974) 391–418.
[46] I. Levi, The Enterprise of Knowledge, The MIT Press, Massachusetts, 1980.
[47] R.D. Luce, H. Raiffa, Games and Decisions, Wiley, New York, 1957.
[48] C. Luo, C. Yu, J. Lobo, G. Wang, T. Pham, Computation of best bounds of probabilities from uncertain data, Computational Intelligence 12 (4) (1996) 541–566.
[49] M.J. Machina, Dynamic consistency and non-expected utility models of choice under uncertainty, Journal of Economic Literature 27 (4) (December 1989) 1622–1688.
[50] E.F. McClennen, Rationality and Dynamic Choice: Foundational Explorations, Cambridge University Press, Cambridge, 1990.
[51] E.F. McClennen, Pragmatic rationality and rules, Philosophy and Public Affairs 26 (3) (1997) 210–258.
[52] T.D. Nielsen, J.-Y. Jaffray, An operational approach to rational decision making based on rank dependent utility, 2001, unpublished manuscript available on http://www.cs.aau.dk/~tdn/papers/nielsen-jaffray-01.pdf.
[53] T.D. Nielsen, F.V. Jensen, Welldefined decision scenarios, in: Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, Morgan Kaufmann, Stockholm, Sweden, July 1999, pp. 502–511.
[54] J. Pearl, Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference, Morgan Kaufmann Publishers, Los Altos, CA, 1988.
[55] J.C. Quiggin, A theory of anticipated utility, Journal of Economic Behavior & Organization 3 (4) (December 1982) 323–343.
[56] H. Raiffa, Decision Analysis: Introductory Lectures on Choices under Uncertainty, Addison-Wesley, Massachusetts, 1968.
[57] P.A. Samuelson, Consumption theory in terms of revealed preference, Econometrica 15 (1948) 243–253.
[58] J.K. Satia, R.E. Lave Jr., Markovian decision processes with uncertain transition probabilities, Operations Research 21 (3) (May–June 1973) 728–740.
[59] M.J. Schervish, T. Seidenfeld, J.B. Kadane, I. Levi, Extensions of expected utility theory and some limitations of pairwise comparisons, in: Proceedings of the 3rd International Symposium on Imprecise Probabilities and Their Applications, Lugano, Switzerland, July 2003, pp. 496–510.
[60] T. Seidenfeld, A contrast between two decision rules for use with (convex) sets of probabilities: Γ-Maximin versus E-Admissibility, Synthese 140 (1–2) (May 2004) 69–88.
[61] T. Seidenfeld, M.J. Schervish, J.B. Kadane, Decisions without ordering, in: W. Sieg (Ed.), Acting and Reflecting, Kluwer Academic Publishers, Dordrecht, 1990, pp. 143–170.
[62] T. Seidenfeld, M.J. Schervish, J.B. Kadane, A representation of partially ordered preferences, Annals of Statistics 23 (6) (December 1995) 2168–2217.
[63] R. Shachter, Bayes-Ball: the rational pastime (for determining irrelevance and requisite information in belief networks and influence diagrams), in: Proceedings of the 14th Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), Morgan Kaufmann, San Francisco, CA, 1998, pp. 480–487.
[64] R. Shachter, Efficient value of information computation, in: Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence (UAI-99), Morgan Kaufmann, San Francisco, CA, 1999, pp. 594–601.
[65] H.D. Sherali, C.H. Tuncbilek, A global optimization algorithm for polynomial programming problems using a reformulation–linearization technique, Journal of Global Optimization 2 (1) (March 1992) 101–112.
[66] H.A. Simon, A behavioral model of rational choice, The Quarterly Journal of Economics 69 (1) (1955) 99–118.
[67] R. Strotz, Myopia and inconsistency in dynamic utility maximization, The Review of Economic Studies 23 (3) (1956) 165–180.
[68] J.A. Tatman, R.D. Shachter, Dynamic programming and influence diagrams, IEEE Transactions on Systems, Man and Cybernetics 20 (2) (1990) 365–379.
[69] F.W. Trevizan, F.G. Cozman, L.N. de Barros, Planning under risk and Knightian uncertainty, in: International Joint Conference on Artificial Intelligence, 2007, pp. 2023–2028.
[70] M.C.M. Troffaes, Decision making with imprecise probabilities: A short review, in: F. Cozman (Ed.), Society for Imprecise Probability Theory and Applications Newsletter, Manno, Switzerland, December 2004, pp. 4–7.
[71] M.C.M. Troffaes, Decision making under uncertainty using imprecise probabilities, International Journal of Approximate Reasoning 45 (1) (2007) 17–29.
[72] L.V. Utkin, T. Augustin, Powerful algorithms for decision making under partial prior information and general ambiguity attitudes, in: Proceedings of the 4th International Symposium on Imprecise Probabilities and Their Applications, Pittsburgh, Pennsylvania, July 2005, pp. 349–358.
[73] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, London, 1991.
[74] C.C. White III, H.K. El-Deib, Markov decision processes with imprecise transition probabilities, Operations Research 42 (4) (July–August 1994) 739–749.