Algorithms for Conditioning on Events of Zero Lower Probability

Fabio G. Cozman [*]
Escola Politécnica, Universidade de São Paulo
Av. Prof. Mello Moraes, 2231 - Cidade Universitária, São Paulo, SP, Brazil
fgcozman@usp.br, http://www.cs.cmu.edu/˜fgcozman

Abstract

This paper presents techniques for computing conditional lower expectations through sequences of pivoting operations, for conditioning events with positive or zero lower/upper probability. The algorithms depart from the sequences of linear programs currently employed in the literature.

Introduction

Suppose that an agent translates beliefs and preferences into lower expectations. For a collection of variables $\mathbf{X}$, a collection of functions $\{f_i(\mathbf{X})\}$ and a collection of events $\{B_i\}$, $i \in \{1, \ldots, k\}$, the agent assesses the lower expectations $\underline{E}[f_i(\mathbf{X})|B_i]$. These lower expectations must satisfy standards of *coherence* among each other (Coletti & Scozzafava 1999; Walley 1991). Coherence usually implies the existence of a set of probability measures, called a *credal set* (Levi 1980). A credal set is such that $\underline{E}[f_i(\mathbf{X})|B_i]$ is the minimum of $E_P[f_i(\mathbf{X})|B_i]$, the expected value of $f_i(\mathbf{X})$ conditional on B_i, taken over all probabilities in the set. Credal sets are often taken to be convex and assumed to be either closed or open; we focus only on convex closed sets in this paper. We do not deal with sets that have both closed and open boundaries, as allowed by the general theory of (Seidenfeld, Schervish, & Kadane 1995). The problem of interest here is, given a credal set, to compute $\underline{E}[f(\mathbf{X})|B]$, the smallest expected value for function $f(\mathbf{X})$ conditional on B.

This paper focuses on situations where at least one probability measure in the credal set assigns probability zero to the conditioning event B. The results have considerable practical significance because an event with zero *lower* probability may have non-zero *upper* probability — consequently the event may actually obtain in practice. Currently, algorithms that deal with zero probabilities employ sequences of linear programs, while algorithms that do not deal with zero probabilities require a single linear program.

The purpose of this paper is to explore computational techniques that reduce the cost of inference in the presence of zero probabilities. The paper contains an "algorithmic-minded" derivation of relevant results and an algorithm for handling zero probabilities by sequences of pivoting operations, not sequences of linear programs.

Inferences that discard zero probabilities

Given a credal set and an event B, either $\underline{P}(B) > 0$, or $\overline{P}(B) = 0$, or $\overline{P}(B) > \underline{P}(B) = 0$. Suppose first that every probability measure in the credal set assigns positive probability to B; that is, $\underline{P}(B) > 0$. The posterior credal set is then obtained by elementwise Bayes rule (Giron & Rios 1980; Levi 1980; Walley 1991):

$$\underline{E}[f(\mathbf{X})|B] = \min_P E_P[f(\mathbf{X})|B] = \min_P \frac{E_P[I_B(\mathbf{X})f(\mathbf{X})]}{E_P[I_B(\mathbf{X})]},$$

where $I_B(\mathbf{X})$ is the indicator function of B. If $\underline{P}(B) > 0$, $\underline{E}[f(\mathbf{X})|B]$ is the unique value of μ that satisfies the equation $\underline{E}[I_B(\mathbf{X})\,(f(\mathbf{X}) - \mu)] = 0$, called the *generalized Bayes rule* (Walley 1991, Chapter 6).

Consider a collection of variables with finitely many values. Every lower expectation $\underline{E}[f_i(\mathbf{X})|B_i]$ is a linear constraint in the space of probability distributions: $E_P[f_i(\mathbf{X})I_{B_i}(\mathbf{X})] \geq \underline{E}[f_i(\mathbf{X})|B_i]\, E_P[I_{B_i}(\mathbf{X})]$. The computation of an *unconditional* lower expectation $\underline{E}[f(\mathbf{X})]$ is then the minimization of $E_P[f(\mathbf{X})]$, a linear function, subject to a collection of linear constraints. This formulation goes back to the work of Boole, as reviewed and generalized in (Hailperin 1996), and stated in different contexts by (Bruno & Gilio 1980) and (Nilsson 1986), among others (Hansen *et al.* 2000).

Consider now the computation of a *conditional* lower expectation $\underline{E}[f(\mathbf{X})|B]$. The Charnes-Cooper transformation can be used to generate a linear program (Cozman 1999):

$$\min \textstyle\sum_{\mathbf{X}} I_B(\mathbf{x}) f(\mathbf{x}) Q(\mathbf{x}); \quad \text{subject to} \qquad (1)$$

$$\sum_{\mathbf{X}} I_{B_i}(\mathbf{x})\,(f_i(\mathbf{x}) - \underline{E}[f_i(\mathbf{X})|B_i])\, Q(\mathbf{x}) \geq 0$$

for all $f_i(\mathbf{X})$, $I_{B_i}(\mathbf{X})$; subject to $\sum_{\mathbf{X}} I_B(\mathbf{x})Q(\mathbf{x}) = 1$; and subject to $Q(\mathbf{x}) \geq 0$ for all values $\mathbf{x}$ of $\mathbf{X}$ ($\mathbf{x}$ refers to specific values of $\mathbf{X}$).

Now suppose that B has zero lower probability and positive upper probability; that is, probability zero is *acceptable* for B, but not sure. We must produce a value for $\underline{E}[f(\mathbf{X})|B]$

[*]The author was partially supported by a grant from CNPq, Brazil.

that is coherent with other assessments $\{\underline{E}[f_i(\mathbf{X})|B_i]\}_{i=1}^k$. One common approach is to focus only on the probability measures that assign positive probability to B:

$$\underline{E}[f(\mathbf{X})|B] = \min_{P:P(B)>0} E_P[f(\mathbf{X})|B]\,. \qquad (2)$$

The idea is to discard models that assign zero probability to conditioning events (Weichselberger 2000). We can follow the same approach when $\overline{P}(B) = 0$, declaring $\underline{E}[f(\mathbf{X})|B]$ undefined in such cases because every probability measure in the credal set assigns zero probability to B.

The minimizing $Q(\mathbf{X})$ in (1) will be proportional to a measure in the credal set that assigns non-zero probability to B. Consequently, the linear program does discard any model such that $P(B) = 0$. When $\overline{P}(B) = 0$ there is no $Q(\mathbf{X})$ that satisfies $\sum_{\mathbf{X}} I_B(\mathbf{x})Q(\mathbf{x}) = 1$, so the linear program has no feasible solution. In such a case we can report that the model must be revised.

The dual of linear program (1) is also of interest. To simplify notation, it is convenient to drop the dependency on $\mathbf{X}$ from all functions and to use the name of an event as the indicator function for the event (de Finetti's convention). Also, it is convenient to denote by G_i the quantity $B_i(f_i - \underline{E}[f_i|B_i])$, understood as the "reward" from buying f_i conditional on B_i at rate $\underline{E}[f_i|B_i]$ (Walley 1991). The dual of (1) is:

$$\max \mu; \quad \text{subject to} \qquad (3)$$

$$\sum_i \lambda_i G_i \leq B(f - \mu), \text{ for all values } \mathbf{x} \text{ of } \mathbf{X}, \text{ and } \lambda_i \geq 0.$$

Natural extensions

There is a different point of view that incorporates zero probabilities in a smoother fashion (de Finetti 1974). Suppose we require that credal sets contain *coherent* measures satisfying four conditions: 1. $P(A) \geq 0$ for any event A; 2. the probability of the universal event is 1; 3. for disjoint A and B, $P(A \cup B) = P(A) + P(B)$; and 4. $P(A \cap B) = P(A|B) \times P(B)$ (Coletti & Scozzafava 1999). Note that instead of defining $P(A|B)$ to be the ratio $P(A \cap B)/P(B)$, we require only that $P(A \cap B)$ be equal to $P(A|B) \times P(B)$. If $P(B)$ is detected to be zero, $P(A \cap B)$ and $P(A|B) \times P(B)$ are *always* equal, regardless of $P(A|B)$. All we require then is that $P(A|B)$ be chosen coherently, and so we may have a definite value for $P(A|B)$ even when $P(B) = 0$.

The value $\underline{E}[f(\mathbf{X})|B]$ obtained using the largest set of coherent probability measures (that is, including coherent probability measures when $P(B) = 0$) is called a *natural extension* by (Walley 1991).[1] The computation of natural extensions is a non-linear problem, because the constraint $P(A \cap B) = P(A|B)\,P(B)$ may have different effects depending on $P(B)$ being zero or not. To illustrate this, consider that an assessment such as $\underline{P}(A|B) = 1/2$, written as $2P(A \cap B) \geq P(B)$, can be ignored if $P(B) = 0$, but cannot be ignored if $P(B) > 0$. When checking coherency, we should look at the complete set of assessments $\{\underline{E}[f_i(\mathbf{X})|B_i]\}_{i=1}^k$ *and also* look at subsets of the assessments (de Finetti 1974). Walley, Pelessoni and Vicig demonstrate that the natural extension $\underline{E}[f(\mathbf{X})|B]$ is the solution of a program (Walley, Pelessoni, & Vicig 1999, referred to as WPV):

$$\max \mu; \text{ subject to } \textstyle\sum_{i \in I} \lambda_i G_i \leq B(f - \mu) \qquad (4)$$

for all values of $\mathbf{X}$; and subject to $\lambda_i \geq 0$ for all i running over a *single* subset I of $\{1, \ldots, k\}$. The set I is the largest subset of $\{1, \ldots, k\}$ for which $\underline{P}(B|B \cup (\cup_{i \in I} B_i)) > 0$, or alternatively, I is the largest subset of $\{1, \ldots, k\}$ such that

$$\sup_{\mathbf{x} \in B^c \cap (\cup_{i \in I} B_i)} \left(\sum_{i \in I} \lambda_i G_i(\mathbf{x}) \right) < 0, \quad \text{where } \lambda_i \geq 0. \quad (5)$$

WPV show that I can be constructed by a sequence of linear programs as follows. Initialize I_0 with $\{1, \ldots, k\}$, j with zero, and then maximize $\sum_{i \in I_j} \tau_i$ subject to

$$\sum_{i \in I_j} \lambda_i G_i + \sum_{i \in I_j} \tau_i B_i \leq 0, \qquad (6)$$

for $\lambda_i \geq 0$, $\tau_i \geq 0$, $\tau_i \leq 1$, and for all $\mathbf{x} \in B^c$. If the maximizing values of τ_i are 1 for all $i \in I_j$, stop and use I_j in (4). Otherwise, form I_{j+1} with the elements i of I_j such that $\tau_i = 1$, and repeat the process with I_{j+1} (maximize $\sum_{i \in I_{j+1}} \tau_i$, etc). This sequence of linear programs is at most as long as the number of assessments.

WPV also demonstrate that the natural extension $\underline{E}[f(\mathbf{X})|B]$ is the supremum value of μ for which there is $\epsilon > 0$ and $\lambda_i \geq 0$ such that

$$\sum_i \lambda_i (G_i + \epsilon B_i) \leq B(f - \mu). \qquad (7)$$

This condition is quite intuitive, as it asks for the "price" μ that can be paid for f, taking in consideration all other "prices" $(\underline{E}[f_i|B_i] - \epsilon)$ (and linear combinations of them). The supremum value of μ may be discontinuous at $\epsilon = 0$, as condition (7) for $\epsilon \to 0$ is not identical to (3).

There are several variations on "coherency conditions" in the literature (Coletti 1994; Coletti & Scozzafava 1999). Using coherency, logical relationships among the B_i can be accommodated and may actually reduce computational effort (Capotorti & Vantaggi 2000). Such techniques can be added to the algorithms presented in this paper.

The results just reviewed are quite elegant but still have a few weaknesses:

- There is no simple way to detect when the natural extension can be directly computed by (3) — this may lead to inefficiencies as conditioning events have positive probability more often than not. We must find algorithms that solve the "non-zero" case efficiently and that gracefully degrade when zero probabilities are present.
- There is a computational cost, both in time and in memory management, associated with setting up a sequence of linear programs. In particular, there is time spent for each linear program searching for an initial feasible solution. There is also a cost associated with handling the potentially many auxiliary variables τ_i.

[1]An alternative approach is to adopt (2) when $\overline{P}(B) > 0$ and require that $\underline{E}[f(\mathbf{X})|B]$ be the minimum of $E_P[f(\mathbf{X})|B]$ among *coherent* measures when $\overline{P}(B) = 0$. This strategy defines a *regular extension*, in the terminology of (Walley 1991, Appendix J).

As a way of avoiding sequences of linear programs, WPV suggest that ϵ should be fixed at a small value, say 10^{-8}. Because $\mu^*(\epsilon)$, the maximum μ as a function of ϵ, does not increase with ϵ, it is possible to bound the natural extension from above and below by solving two linear programs, first taking a small ϵ and then taking ϵ equal to zero. As the theoretical complexity of linear programming depends on the number of bits needed to store values (Bertsimas & Tsitsiklis 1997), small values of ϵ lead to larger computational effort — at the very least, more sophisticated linear programming techniques must be employed to prevent numerical instability from dwarfing the effect of small ϵ. In fact, numerical instability is certain to be present when $\mu^*(\epsilon)$ is discontinuous at zero ϵ.[2]

It seems that at least two linear programs must be solved to produce a natural extension $\underline{E}[f|B]$ with some confidence. One alternative is to first solve (3) (that is, take ϵ to be zero) and then to solve the first step of (6) — if all $\tau_i = 1$, then the natural extension is ready. A second alternative is to compute the unconditional $\underline{P}(B)$ and, if $\underline{P}(B) > 0$, then obtain the natural extension with an additional linear program (3). If we have to compute a sequence of τ_i, we always have to solve an additional linear program at the end of the sequence, just to check that the natural extension has been obtained.

The structure of natural extensions

WVP's algorithms for natural extension explore both coherency conditions (4) and (7). The result is a theory of considerable complexity. In this section, a simpler route is pursued, starting from condition (7) alone. The purpose of this effort is twofold: first, to simplify the results related to zero probabilities; second, to obtain insights on algorithms that can handle zero probabilities.

Define $\underline{f}$ to be $\inf f(\mathbf{X})$. Because any value of $\mu^*(\epsilon)$ must be larger than or equal to $\underline{f}$, we are interested in the program:

$$\max \mu; \quad \text{subject to} \sum_i \lambda_i (G_i + \epsilon B_i) \leq B\left((f - \underline{f}) - \mu\right), \tag{8}$$

where $\lambda_i \geq 0$ for i in $\{1, \ldots, k\}$, $\mu \geq 0$ and $\epsilon > 0$. An initial feasible solution can be easily found for program (8) for any value of ϵ: just take $\mu = 0$ and $\lambda_i = 0$ for i in $\{1, \ldots, k\}$. Consequently, program (8) is more attractive than its dual for any value of ϵ.

The maximum of (8) is a piecewise rational function $\mu^*(\epsilon)$ of ϵ (Bertsimas & Tsitsiklis 1997). Even though $\mu^*(\epsilon)$ may have many discontinuities for ϵ close to zero, we are only interested in a (continuous) piece of $\mu^*(\epsilon)$ in a vicinity of $\epsilon = 0$. Consider then a symbolic solution of (8) where we retain only the rational functions that are valid in the vicinity of $\epsilon = 0$ (the simplex method will produce a tableau with a rational function in each entry). At the end, the exact value of the natural extension can be read off of the tableau by setting ϵ to zero. In this process, *only one* test for equality is required for each pivoting operation, exactly the test that detects whether or not the current basic solution is degenerate (Bertsimas & Tsitsiklis 1997). All other comparisons are strict, and cannot be affected by $\epsilon > 0$, as the elements of the tableau are always rational functions of ϵ. We conclude that degeneracy necessarily happens in the course of optimization whenever the natural extension is different from $\mu^*(0)$.

The symbolic method is relatively easy to code and can be valuable in small problems where exact solutions are required.[3] In practice, the order of polynomials produced by the method can become unmanageably large. We might look at implementations that compute a few coefficients for each polynomial, increasing the number of available coefficients only when needed. Even then, the potential complexity of the method reduces its practical value.

Instead of focusing on (8) as a symbolic linear program, we can use this program to derive Expressions (4) and (5) and to obtain insights on algorithmic development. Suppose then that we solve (8) for $\epsilon = 0$. Denote the maximizing values by μ^*, λ^*. To investigate the properties of λ^* and μ^*, it is convenient to define a new concept. The constraint associated with a particular value $\mathbf{x}$ is *ϵ-active* (with respect to λ^*) if the constraint is satisfied with equality for $\epsilon = 0$ and if $\sum_i \lambda_i^* B_i(\mathbf{x}) > 0$.

Consider the constraints for $\mathbf{x} \in B^c$, which can be written as $\sum_i \lambda_i^* G_i + \epsilon \sum_i \lambda_i^* B_i \leq 0$. If such an inequality is not ϵ-active, it can remain valid for small enough positive ϵ; otherwise, the inequality is violated for any positive ϵ because

$$\sum_i \lambda_i^* G_i + \epsilon \sum_i \lambda_i^* B_i = \epsilon \sum_i \lambda_i^* B_i > 0.$$

Consider the constraints for $\mathbf{x} \in B$:

$$\sum_i \lambda_i^* G_i + \epsilon \sum_i \lambda_i^* B_i + \mu \leq f - \underline{f}.$$

Again, an inequality that is not ϵ-active can remain valid for small enough positive ϵ. Now suppose there are ϵ-active constraints for $\mathbf{x} \in B$. We can maintain feasibility by taking $\mu = (\mu^* - \epsilon k \overline{\lambda}^*)$, as we get

$$\sum_i \lambda_i^* G_i + \epsilon \sum_i \lambda_i^* B_i + (\mu^* - \epsilon k \overline{\lambda}^*) < f - \underline{f},$$

where k is the total number of assessments and $\overline{\lambda}^*$ is the largest value of λ_i^* for i in $\{1, \ldots, k\}$. Because $\lim_{\epsilon \to 0} (\mu^* - \epsilon k \overline{\lambda}^*) = \mu^*$, the value of the natural extension is μ^* as long as no ϵ-active constraint is present for $\mathbf{x} \in B^c$. We conclude:

Remark 1 *We obtain λ^* and μ^*, and if no constraint for B^c is ϵ-active, then μ^* is the natural extension.*

This verification step is simpler than the computation of $\underline{P}(B)$ or the solution of a linear program in τ_i, but it is equivalent to those schemes (as will be shown later).

[2]As an example, consider the linear program in WPV's Example 1. Solving this program with the semi-sophisticated algorithms presented in (Press *et al.* 1992), implemented in the Java language with 64-bit floating point arithmetic, produces a totally incorrect solution: the simplex method halts with an unbounded region for μ.

[3]Code implementing the symbolic tableau method can be obtained from the author.

Suppose we obtain λ^* and μ^*, and we detect ϵ-active constraints for B^c. How should we proceed? Take as starting point an ϵ-active constraint for some $\mathbf{x} \in B^c$ and λ^*. Assume that there are feasible values λ' and μ' such that $\sum_i \lambda'_i G_i(\mathbf{x}) < 0$. We can take the convex combinations $(1-\alpha)\lambda^*_i + \alpha\lambda'_i$ and $(1-\alpha)\mu^* + \alpha\mu'$ for some $\alpha \in (0,1)$, and then (note the strict inequality):

$$(1-\alpha)\sum_i \lambda^*_i G_i(\mathbf{x}) + \alpha \sum_i \lambda'_i G_i(\mathbf{x}) = \alpha \sum_i \lambda'_i G_i(\mathbf{x}) < 0.$$

In general, we can mix various values of λ with λ^* so as to force strict inequalities in all constraints for $\mathbf{x} \in B^c$ — except those constraints that are ϵ-active for *every* feasible λ. To guarantee feasibility for positive ϵ, we must at least "remove" the constraints that are always-active for $\epsilon = 0$:

Remark 2 *Consider the constraints for $\mathbf{x} \in B^c$ that are active when $\epsilon = 0$ for all feasible λ. Because these constraints satisfy $\sum_i \lambda_i G_i = 0$ for every feasible λ, we must set some λ_i to zero to enforce that $\sum_i \lambda_i B_i = 0$ for all such constraints.*

To "remove" the always-active constraints, we force some of the λ_i to be zero — in effect we add new constraints of the form $\lambda_i = 0$. We may lose optimality for μ, so we must again solve this linear program *with the added constraints on* λ. It is possible that the added constraints on λ cause some of the original constraints for $\mathbf{x} \in B^c$ to become active for all feasible values of λ. These new always-active constraints must then be "removed" and the process must be repeated until no always-active constraint remains. Denote by $\mu^\#$ the maximum value of μ subject to the remaining constraints, and by $\lambda^\#$ the corresponding value of λ.

When we obtain $\mu^\#$, it is possible that all remaining constraints for $\mathbf{x} \in B^c$ are strict inequalities for $\epsilon = 0$. In this case all constraints can be satisfied for small enough positive ϵ, and the natural extension is obviously $\mu^\#$. However in some problems there may be remaining constraints for $\mathbf{x} \in B^c$ such that $\sum_i \lambda^\#_i G_i(\mathbf{x}) = 0$. Note that, because always-active constraints have been "removed," we have reached a situation where some convex combination of values of λ can satisfy the remaining constraints with strict inequality. Take a convex combination $(1-\alpha_0)\lambda^\# + \sum_{j=1}^J \alpha_0 \lambda^j / J$ such that $\sum_{j=1}^J \lambda^j < 0$. By using this convex combination, we can find an α_0 guarantee that all constraints are strict for positive ϵ. Just choose $\alpha_0(\epsilon) = \min_{\mathbf{X}} -\frac{\epsilon B^\#}{G' + \epsilon(B' - B^\#)}$, where $G' = \sum_{ij} \lambda^j_i G_i(\mathbf{X})/J$, $B' = \sum_{ij} \lambda^j_i B_i(\mathbf{X})/J$, and $B^\# = \sum_i \lambda^\#_i B_i(\mathbf{X})$. Note that $\alpha_0(\epsilon)$ decreases when ϵ decreases, is positive for small enough ϵ, and $\lim_{\epsilon \to 0} \alpha_0(\epsilon) = 0$, so the effect of the "mixing terms" λ^j vanishes as ϵ tends to 0 from above. Consequently:

Remark 3 *The natural extension is equal to $\mu^\#$.*

Thus we may have to focus on a subset of the constraints (7) to compute the natural extension. The required constraints must be indexed by a subset I of $\{1, \dots, k\}$, and I must be such that $\sum_{i \in I} \lambda_i G_i(\mathbf{x}) < 0$ for all $\mathbf{x} \in B^c$ and for some λ. So we obtain the set I in (5).

Note that if a problem contains ϵ-active constraints for $\mathbf{x} \in B^c$, then $\underline{P}(B)$ must be zero. That is because $\underline{P}(B)$ is the maximum value of μ subject to $\sum_i \lambda_i G_i + \mu \leq B$ for $\lambda \geq 0$, $\mu \geq 0$. If we have an ϵ-active constraint for $\mathbf{x} \in B^c$, we must have $\mu \leq 0$, and then we must have $\underline{P}(B) = \mu = 0$. The opposite reasoning is valid also. Remark 1 just describes a test for detecting when $\underline{P}(B) = 0$ that is more efficient than existing methods summarized at the end of last section.

Remark 2 is essentially related to constraints that are always active. Because every constraint in (8) is an inequality, the solution of the linear program involves the association of *slack variable* y' with the constraint for $\mathbf{x}'$ (Bertsimas & Tsitsiklis 1997). The dual constraint associated with y' is $P(\mathbf{x}') \geq 0$. If the constraint for $\mathbf{x}' \in B^c$ is always active, the slack variable is always zero. Using the *null variable theorem* of linear programming (Bertsimas & Tsitsiklis 1997), we obtain that y' is a null variable if and only if there exists a probability measure in the credal set such that $P(B) = 0$ and such that $P(\mathbf{x}') \neq 0$. In other words, the constraint for $\mathbf{x}' \in B^c$ is always active if and only if there is a probability measure in the credal set such that $P(B) = 0$ and $P(\mathbf{x}') > 0$.

Imagine a credal set living in a multi-dimensional region of space; every point in this region has nonnegative components, and points with zero components must live in the border of the region. Suppose we have a credal set with vertices in the border for which $P(B) = 0$. Now, suppose we go to every such vertex and mark the constraints corresponding to $\mathbf{x} \in B^c$ such that $P(\mathbf{x}) > 0$ for that vertex. These constraints are exactly the constraints that are removed as we work towards $\mu^\#$.

An additional conclusion can be derived by analyzing the reduced cost vectors for program (1). Small positive values of ϵ can only affect the components of the reduced cost vector that are zero. Now, an optimal basic solution is degenerate if and only if the dual optimal basic solution has a reduced cost equal to zero (Bertsimas & Tsitsiklis 1997). So, we must have ϵ-active constraints when λ^* and μ^* are a degenerate basic solution of (3), showing again the interplay between zero probabilities and degeneracy.

A pivoting scheme for natural extensions

We can use the results developed so far to produce a new algorithm for computing natural extensions. The algorithm presented in this section attempts to eliminate as many inefficiencies as possible from the computation of natural extensions.

---

1) The starting point is to obtain λ^* and μ^* from (3). The solution of (3) introduces a slack variable y for each constraint, and we apply Remark 1 to decide whether we already have the natural extension. If y' is zero and $\sum_i \lambda^*_i B_i(\mathbf{x}') > 0$, then the constraint for $\mathbf{x}'$ is ϵ-active. If no ϵ-active constraint for $\mathbf{x} \in B^c$ is detected, we stop and declare μ^* to be the natural extension. Otherwise, we must verify which constraints are always-active. Again, we resort to slack variables. Sup-

pose that the constraint for $\mathbf{x}'$ is ϵ-active (that is, y' is zero). To detect whether this constraint is always-active, we maximize the associated slack variables and determine whether each y' can be larger than zero.
2) After we detect that the constraint for $\mathbf{x}'$ is ϵ-active, we must remove this constraint using Remark 2. We do so by setting some λ_i to zero — each one of these operations can be accomplished with a pivoting operation in the tableau for (3). We can set λ_i to zero simply by making it a nonbasic variable in the tableau. Instead of first detecting all the constraints that are ϵ-active and then removing them all, it is more efficient to remove a constraint as soon as possible. The very act of pivoting to remove a constraint may reveal that some other slack variables, at zero level for λ^*, are non null after all. And by setting some λ_i to zero, we may find that other ϵ-active constraints are removed simultaneously.
3) After removing always-active constraints, we return to (3), *now enforcing that some of the λ_i must be zero*. We then maximize μ, and return to the same process: check for ϵ-active constraints, pivot to maximize slack variables, pivot to remove constraints, etc. Eventually we reach a point where always-active constraints have been removed, and the simplex method yields $\mu^{\#}$. As stated by Remark 3, we then have the natural extension.

At first sight it may seem that maximizing some slack variables (end of step 1) incurs a large cost. There is no need to maximize slack variables to optimality: as soon as a pivoting operation leads to a positive value for y', the associated constraint cannot be always-active. And here we can use another important insight: all the information necessary to maximize the value of a slack variable y' *already* is in the tableau for (3). If y' is basic at λ^*, then the cost associated with y' is available in the tableau, and a feasible solution is already available. If y' is non-basic, then a single pivoting operation can bring y' to the basis and then the same argument is valid. An additional computational gain can be obtained by, after *every* pivoting operation, marking the slack variables that have positive values. These slack variables are associated with constraints that cannot be always-active. With this simple bookkeeping, we speed up the detection of always-active constraints.

Note that the algorithm runs through a sequence of pivoting operations. A single tableau is used, and the algorithm never spends any time looking for feasible solutions or initializing tableaus; there are no "small values" to set up, and no auxiliary variables τ_i to introduce.

Conclusion

This paper shows how to obtain an algorithmic understanding of natural extensions from the relatively simple and intuitive condition (7), and how to create a pivoting algorithm for computing natural extensions. The symbolic simplex method and the interplay between zero probability and active constraints/zero reduced costs can clarify subtle aspects of natural extensions.

A complete algorithmic understanding of natural extensions still requires empirical solution of practical problems and comparison of algorithms. From a theoretical point of view, it would be important to determine the average number of pivoting operations for the algorithm presented. This average number should be compared to the average complexity of existing algorithms (which use a sequence of linear programs) and to the average complexity of program (1). Finally, it would be useful to look at interior methods for the computation of natural extensions.

References

Bertsimas, D., and Tsitsiklis, J. N. 1997. *Introduction to Linear Optimization*. Belmont, Massachusetts: Athena Scientific.

Bruno, G., and Gilio, A. 1980. Applicazione del metodo del simplesso al teorema fondamentale per le probabilità nella concezione soggettivistica. *Statistica* 40:337–344.

Capotorti, A., and Vantaggi, B. 2000. A computational strategy for inference by strong local coherency. In *Proc. of the Eighth Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-based Systems (IPMU)*, volume 3.

Coletti, G., and Scozzafava, R. 1999. Coherent upper and lower Bayesian updating. *Proc. of the First Int. Symp. on Imprecise Probabilities and Their Applications*, 101–110. Ghent, Belgium: Imprecise Probabilities Project, Universiteit Gent.

Coletti, G. 1994. Coherent numerical and ordinal probabilistic assessments. *IEEE Transactions on Systems, Man and Cybernetics* 24(12):1747–1753.

Cozman, F. G. 1999. Calculation of posterior bounds given convex sets of prior probability measures and likelihood functions. *Journal of Computational and Graphical Statistics* 8(4):824–838.

de Finetti, B. 1974. *Theory of probability, vol. 1-2*. New York: Wiley.

Giron, F. J., and Rios, S. 1980. Quasi-Bayesian behaviour: A more realistic approach to decision making? In Bernardo, J. M.; DeGroot, J. H.; Lindley, D. V.; and Smith, A. F. M., eds., *Bayesian Statistics*. Valencia, Spain: University Press. 17–38.

Hailperin, T. 1996. *Sentential Probability Logic*. Bethlehem, United States: Lehigh University Press.

Hansen, P.; Jaumard, B.; de Aragão, M. P.; Chauny, F.; and Perron, S. 2000. Probabilistic satisfiability with imprecise probabilities. *Int. Journal of Approximate Reasoning* 24(2-3):171–189.

Levi, I. 1980. *The Enterprise of Knowledge*. Cambridge, Massachusetts: MIT Press.

Nilsson, N. J. 1986. Probabilistic logic. *Artificial Intelligence* 28:71–87.

Press, W. H.; Teukolsky, S. A.; Vetterling, W. T.; and Flannery, B. P. 1992. *Numerical Recipes in C*. Cambridgeshire: Cambridge University Press.

Seidenfeld, T.; Schervish, M. J.; and Kadane, J. B. 1995. A representation of partially ordered preferences. *Annals of Statistics* 23(6):2168–2217.

Walley, P.; Pelessoni, R.; and Vicig, P. 1999. Direct algorithms for checking coherence and making inferences from conditional probability assessments. Technical report, Dipartimento di Matematica Applicata B. de Finetti, Università di Trieste, Trieste, Italy.

Walley, P. 1991. *Statistical Reasoning with Imprecise Probabilities*. London: Chapman and Hall.

Weichselberger, K. 2000. The theory of interval-probability as a unifying concept for uncertainty. *Int. Journal of Approximate Reasoning* 24(2-3):149–170.