A Propositional CONEstrip Algorithm

Erik Quaeghebeur*

Centrum Wiskunde & Informatica
Science Park 123, 1098 XG Amsterdam, Netherlands

Abstract We present a variant of the CONEstrip algorithm for checking whether the origin lies in a finitely generated convex cone that can be open, closed, or neither. This variant is designed to deal efficiently with problems where the rays defining the cone are specified as linear combinations of propositional sentences. The variant differs from the original algorithm in that we apply row generation techniques. The generator problem is WPMaxSAT, an optimization variant of SAT; both can be solved with specialized solvers or integer linear programming techniques. We additionally show how optimization problems over the cone can be solved by using our propositional CONEstrip algorithm as a preprocessor. The algorithm is designed to support consistency and inference computations within the theory of sets of desirable gambles. We also make a link to similar computations in probabilistic logic, conditional probability assessments, and imprecise probability theory.

Keywords: sets of desirable gambles, linear programming, row generation, satisfiability, SAT, PSAT, WPMaxSAT, consistency, coherence, inference, natural extension

1 Introduction

The CONEstrip algorithm [12] determines whether a finitely generated general convex cone contains the origin. A general convex cone can be open, closed, or ajar, i.e., neither open nor closed. This linear programming-based algorithm is designed for working with uncertainty models based on (non-simple) sets of desirable gambles [15,16,13] and their generalizations [14]. In particular, it can be used for checking the consistency criteria such models have to satisfy—specifically, coherence and avoiding partial loss—and for drawing deductive inferences from such models—typically, performing natural extension.

In the CONEstrip algorithm, the so-called gambles defining the cone had to be specified as vectors on some explicitly given, finite possibility space. The specification of the gambles as linear combinations of indicator functions of events

* This research was started while the author was a member of the SYSTeMS Research Group of Ghent University and was at that time a visiting scholar at the Department of Information and Computing Sciences of Utrecht University. It was finished during the tenure of an ERCIM "Alain Bensoussan" Fellowship Programme. The research leading to these results has received funding from the *European Union* Seventh Framework Programme (FP7/2007-2013) under *grant agreement* n° 246016.

belonging to some finite set often provides a more natural and economical formulation of the uncertainty model.

Events can be formalized as logical propositions; an elementary event of the underlying possibility space corresponds to a conjunction of all these propositions or their negation. The cardinality of the underlying possibility space is therefore exponential in the number of events. So even though it is possible to write down the gambles as vectors on the underlying possibility space and apply the CONEstrip algorithm, this would be very inefficient: the size of the linear programs involved is linear in the cardinality of the possibility space.

Therefore we need a variant of the algorithm that works efficiently in the propositional context sketched. We present such a variant in this paper. It preserves the structure of CONEstrip as an iteration of linear programs, each one 'stripping away' a superfluous part of the general cone. But now these linear programs are solved using a row generation technique—each row corresponding to an elementary event—as was already suggested by Walley et al. [17, Sec. 4, comment (d)] for a CONEstrip predecessor. Only the rows necessary for solving the linear programs are generated, and those already generated are carried over from one linear program to the next.

So instead of solving one big problem, we solve multiple smaller ones. The sub-problem to be solved to generate a row is WPMaxSAT, weighted partial maximum satisfiability. This was already discovered by Georgakopoulos et al. [6, Sec. 3] for the related dual PSAT, probabilistic satisfiability. WPMaxSAT is an optimization variant of SAT, propositional satisfiability [4]. Both can be tackled using binary linear programming techniques or specific algorithms [9,7,5].

The original impulse to work on this problem came from Cozman and di Ianni's recent contribution to the field [5], where many relevant references are listed, among which Hansen et al.'s classic review [8] cannot remain unmentioned here. The goal was to design a 'direct algorithm' [17] for sets of desirable gambles in their full generality. To present the result, we first get (re)acquainted with the standard CONEstrip algorithm in Section 2 and then build up towards the propositional variant in Section 3. In Section 4, we make a link to established problems that can be seen as special cases.

2 Preliminaries

A Representation of General Convex Cones. General convex cones can be open, closed, or ajar, i.e., neither open nor closed. Any finitely generated general convex cone can by definition be generated as the convex hull of a finite number of finitely generated open cones.

Let us formalize this definition: The possibility space is denoted by Ω. The set of all gambles is then $\mathcal{G} = \Omega \to \mathbb{R}$. Now consider a finite set $\mathcal{R}_0$ of finite subsets of $\mathcal{G}$, each containing the generators of an open cone, then a gamble f in $\mathcal{G}$ belongs to the general cone $\underline{\mathcal{R}_0}$ if and only if the following feasibility problem has a solution:

$$\begin{array}{rl} \text{find} & \lambda_{\mathcal{D}} \in [0,1] \quad \text{and} \quad \nu_{\mathcal{D}} \in (\mathbb{R}_{>0})^{\mathcal{D}} \quad \text{for all } \mathcal{D} \text{ in } \mathcal{R}_0 \\ \text{such that} & \sum_{\mathcal{D}\in\mathcal{R}_0} \lambda_{\mathcal{D}} = 1 \quad \text{and} \quad \sum_{\mathcal{D}\in\mathcal{R}_0} \lambda_{\mathcal{D}} \sum_{g\in\mathcal{D}} \nu_{\mathcal{D},g} g \gtreqless f. \end{array} \tag{1}$$

Here $h \coloneqq \sum_{\mathcal{D}\in\mathcal{R}_0} \lambda_{\mathcal{D}} \sum_{g\in\mathcal{D}} \nu_{\mathcal{D},g} g \gtreqless f$ represents $h(\omega) \leq f(\omega)$ for all ω in Ω_Γ and $h(\omega) \geq f(\omega)$ for all ω in Ω_Δ, with Ω_Γ and Ω_Δ problem-specific sets that satisfy $\Omega_\Gamma \cup \Omega_\Delta = \Omega$. Using inequality instead of equality constraints allows us to omit (up to $|\Omega|$) indicator functions of singletons or their negation that may be present in the representation $\mathcal{R}_0$.

The formulation of this problem includes strict inequalities and bilinear constraints. The former issue puts this problem outside of the class of standard mathematical programming problems. The latter issue makes it a non-linear problem, and therefore one with non-polynomial worst-case computational complexity. Below, we are going to go over the solution to both issues [12].

The subscript 0 in $\mathcal{R}_0$ indicates that it is the representation of the original cone. In the algorithms we discuss, smaller cones will iteratively be derived from it; their representation $\mathcal{R}_i$ gets the iteration number i as a subscript.

The CONEstrip Algorithm. To eliminate the strict inequalities $\nu_{\mathcal{D}} \in (\mathbb{R}_{>0})^{\mathcal{D}}$ from Problem (1) we are going to replace the $\nu_{\mathcal{D}}$ using $\tau_{\mathcal{D}} \in (\mathbb{R}_{\geq 1})^{\mathcal{D}}$ and $\sigma \in \mathbb{R}_{\geq 1}$ such that $\tau_{\mathcal{D}} = \sigma\nu_{\mathcal{D}}$ for all $\mathcal{D}$ in $\mathcal{R}_0$:

$$\begin{array}{rl} \text{find} & \lambda_{\mathcal{D}} \in [0,1] \quad \text{and} \quad \tau_{\mathcal{D}} \in (\mathbb{R}_{\geq 1})^{\mathcal{D}} \quad \text{for all } \mathcal{D} \text{ in } \mathcal{R}_0 \quad \text{and} \quad \sigma \in \mathbb{R}_{\geq 1} \\ \text{s.t.} & \sum_{\mathcal{D}\in\mathcal{R}_0} \lambda_{\mathcal{D}} \geq 1 \quad \text{and} \quad \sum_{\mathcal{D}\in\mathcal{R}_0} \lambda_{\mathcal{D}} \sum_{g\in\mathcal{D}} \tau_{\mathcal{D},g} g \gtreqless \sigma f. \end{array} \tag{2}$$

Note that we have also relaxed the convex coefficient constraint, because it gives us the flexibility we need next, without changing the problem.

To get rid of the non-linearity, we first replace $\lambda_{\mathcal{D}}\tau_{\mathcal{D}}$ with new variables $\mu_{\mathcal{D}}$ for all $\mathcal{D}$ in $\mathcal{R}_0$ and add a constraint that forces $\mu_{\mathcal{D}}$ to behave as $\lambda_{\mathcal{D}}\tau_{\mathcal{D}}$:

$$\begin{array}{rl} \text{find} & \lambda_{\mathcal{D}} \in [0,1] \quad \text{and} \quad \mu_{\mathcal{D}} \in (\mathbb{R}_{\geq 0})^{\mathcal{D}} \quad \text{for all } \mathcal{D} \text{ in } \mathcal{R}_0 \quad \text{and} \quad \sigma \in \mathbb{R}_{\geq 1} \\ \text{s.t.} & \sum_{\mathcal{D}\in\mathcal{R}_0} \lambda_{\mathcal{D}} \geq 1 \quad \text{and} \quad \sum_{\mathcal{D}\in\mathcal{R}_0} \sum_{g\in\mathcal{D}} \mu_{\mathcal{D},g} g \gtreqless \sigma f \\ & \lambda_{\mathcal{D}} \leq \mu_{\mathcal{D}} \leq \lambda_{\mathcal{D}}\mu_{\mathcal{D}} \text{ for all } \mathcal{D} \text{ in } \mathcal{R}_0. \end{array} \tag{3}$$

Notice that now $\lambda_{\mathcal{D}} \in \{0,1\}$ for any solution, functioning as a switch between $\mu_{\mathcal{D}} = 0\tau_{\mathcal{D}} = 0$ and $\mu_{\mathcal{D}} = 1\tau_{\mathcal{D}} \in (\mathbb{R}_{\geq 1})^{\mathcal{D}}$, so that $\mu_{\mathcal{D}}/\sigma$ effectively behaves as $\lambda_{\mathcal{D}}\nu_{\mathcal{D}}$.

We could replace the non-linear constraints $\mu_{\mathcal{D}} \leq \lambda_{\mathcal{D}}\mu_{\mathcal{D}}$ by $\mu_{\mathcal{D}} \leq \lambda_{\mathcal{D}} M_{\mathcal{D}}$, where $M_{\mathcal{D}}$ is a positive real number 'guaranteed to be' larger than $\max \mu_{\mathcal{D}}$, but this may be numerically problematic. Another approach is to remove the non-linear constraint, causing $\lambda_{\mathcal{D}} = 0$ to not force $\mu_{\mathcal{D}} = 0$ anymore—$\lambda_{\mathcal{D}} > 0$ still forces $\mu_{\mathcal{D}} \in (\mathbb{R}_{>0})^{\mathcal{D}}$. However, by maximizing $\sum_{\mathcal{D}\in\mathcal{R}_0} \lambda_{\mathcal{D}}$, the components of λ will function as a witness: if $\lambda_{\mathcal{D}} = 0$, then we *should* have $\mu_{\mathcal{D}} = 0$. This is the basis of the iterative CONEstrip algorithm:

```
***Initialization***. Set $i \coloneqq 0$.
***Iterand***. Does the linear programming problem below have a solution $(\bar{\lambda}, \bar{\mu}, \bar{\sigma})$?
```

$$\begin{array}{rl} \text{maximize} & \sum_{\mathcal{D} \in \mathcal{R}_i} \lambda_{\mathcal{D}} \\ \text{subject to} & \lambda_{\mathcal{D}} \in [0,1] \quad \text{and} \quad \mu_{\mathcal{D}} \in (\mathbb{R}_{\geq 0})^{\mathcal{D}} \quad \text{for all } \mathcal{D} \text{ in } \mathcal{R}_i \quad \text{and} \quad \sigma \in \mathbb{R}_{\geq 1} \\ & \sum_{\mathcal{D} \in \mathcal{R}_i} \lambda_{\mathcal{D}} \geq 1 \quad \text{and} \quad \sum_{\mathcal{D} \in \mathcal{R}_i} \sum_{g \in \mathcal{D}} \mu_{\mathcal{D},g} g \gneqq \sigma f \\ & \lambda_{\mathcal{D}} \leq \mu_{\mathcal{D}} \text{ for all } \mathcal{D} \text{ in } \mathcal{R}_i. \end{array} \tag{4}$$

```
  ***No***. $f \notin \underline{\mathcal{R}_0}$. ***Stop***.
  ***Yes***. Let $\mathcal{Q} \coloneqq \{\mathcal{D} \in \mathcal{R}_i \colon \bar{\lambda}_{\mathcal{D}} = 0\}$ and set $\mathcal{R}_{i+1} \coloneqq \mathcal{R}_i \setminus \mathcal{Q}$.
    Is $\{\mathcal{D} \in \mathcal{Q} \colon \bar{\mu}_{\mathcal{D}} = 0\} = \mathcal{Q}$?
    ***Yes***. Set $t \coloneqq i+1$; $f \in \underline{\mathcal{R}_t} \subseteq \underline{\mathcal{R}_0}$. ***Stop***.
    ***No***. Increase $i$'s value by 1. ***Reiterate***.
```

This algorithm terminates after at most $|\mathcal{R}_0| - 1$ iterations. The 'raw' complexity of the ith linear programming problem is polynomial in $|\mathcal{R}_i|$, $\sum_{\mathcal{D} \in \mathcal{R}_i} |\mathcal{D}|$, and $|\Omega|$. The terminal cone $\underline{\mathcal{R}_t}$ is the largest 'subcone' of $\underline{\mathcal{R}_0}$ that contains f in its relative interior; so $\underline{\mathcal{R}_t}$ is included in a face of $\underline{\mathcal{R}_0}$.

Optimization Problems. We can solve optimization problems with continuous objective functions over the general cone: First ignore the objective and run the CONEstrip algorithm on the representation $\mathcal{R}_0$ to obtain a terminal representation $\mathcal{R}_t$. Then we can optimize over the *topological closure* of the terminal general cone $\underline{\mathcal{R}_t}$, due to the continuity of the objective function:

$$\begin{array}{rl} \text{optimize} & \text{a continuous function of } \mu \\ \text{subject to} & \mu \in (\mathbb{R}_{\geq 0})^{\cup \mathcal{R}_t} \quad \text{and} \quad \sum_{g \in \cup \mathcal{R}_t} \mu_g g \gneqq f. \end{array} \tag{5}$$

When the objective function is linear in μ, we get a linear programming problem.

3 An Algorithm for Proposition-Based Gambles

We saw that Problem (4) was polynomial in the size $|\Omega|$ of the possibility space. When, as is often done, the gambles involved in the representation of the general cone are specified as linear combinations of indicator functions of Boolean propositions, $|\Omega|$ becomes exponential in the number of propositions. In this section we present an approach that avoids having to deal directly with an exponential number of constraints in such a propositional context.

How Structured Gambles Generate Structured Problems. We are not going to dive into the propositional framework directly, but first we are going to show how gambles that are linear combinations of a number of basic functions generate a specific exploitable structure in the problems we consider. Assume that any gamble g in $\{f\} \cup \bigcup \mathcal{R}_0$ can be written as a linear combination

$\sum_{\phi\in\Phi} g_\phi\phi$ with a finite set of basic functions $\Phi \subset \mathcal{G}$ and coefficients g_ϕ in $\mathbb{R}$.[1] Then $(\sum_{\mathcal{D}\in\mathcal{R}_i}\sum_{g\in\mathcal{D}}\mu_{\mathcal{D},g}g) - \sigma f = \sum_{\phi\in\Phi}\kappa_\phi\phi$ if for all ϕ in Φ we define κ_ϕ as $(\sum_{\mathcal{D}\in\mathcal{R}_i}\sum_{g\in\mathcal{D}}\mu_{\mathcal{D},g}g_\phi) - \sigma f_\phi$. So in Problem 4 we can rewrite the constraints on the variables μ and σ in terms of gambles as constraints on the variables κ in terms of basic functions by adding constraints linking the κ with the μ and σ.

Increasing the number of constraints in this way is productive only when we can deal with them more efficiently. How this can be done is discussed below.

Row Generation. A standard technique for dealing with large numbers of constraints in mathematical programming is row generation: The original problem is first relaxed by removing most or all constraints; in our problem, the constraints $\sum_{\phi\in\Phi}\kappa_\phi\phi \gneqq 0$ are removed. Then, in an iterative procedure, constraints are added back. Each such constraint or 'row' corresponds to some elementary event ω in Ω, i.e., is of the form $\sum_{\phi\in\Phi}\kappa_\phi\phi(\omega) \gneqq 0$. Each iteration the problem is solved under the present constraints, resulting in a solution vector $\bar{\kappa}$.

So which constraints should be added back? Constraints that are satisfied by the present solution $\bar{\kappa}$ will have no discernable impact, as $\bar{\kappa}$ will remain feasible. Therefore, constraints that are *violated* by $\bar{\kappa}$ must be generated. There may be many violated constraints and one would want to generate deep 'cuts', those that constrain κ most, as less are needed than when generating shallow cuts. However, generating deep cuts may be computationally more complex than generating shallow cuts, so a trade-off needs to be made between the number of iterations and the complexity of the constraint generation process. For our problem, $\operatorname{argmax}_{\omega\in\Omega}|\sum_{\phi\in\Phi}\bar{\kappa}_\phi\phi(\omega)|$ would generate a deep cut.

So when does the procedure stop? The original problem is infeasible if constraint generation is infeasible, or when an intermediate problem turns out to be infeasible, given that it is a relaxation of the original problem. When no violated constraint can be generated given a solution $\bar{\kappa}$, then the problem is feasible and—in case of an optimization problem—this solution is optimal. That the problem stops eventually is guaranteed, because we could in principle add all constraints back. But actually, the number of iterations needed is polynomial in the number of variables involved [6, Lemma 2]; so, for us, polynomial in $|\Phi|$.

The Propositional Context. Now we take the basic functions ϕ in Φ to be expressed as propositional sentences. The operations permitted in such sentences are the binary disjunction $\vee$—'or'—and conjunction $\wedge$—'and'—, and the unary negation $\neg$. The $\{0, 1\}$-valued binary variables appearing in the sentence are the so-called literals β_ℓ, where ℓ belongs to a given, finite index set $\mathcal{L}_\Phi := \bigcup_{\phi\in\Phi}\mathcal{L}_\phi$. For example, $\varphi(\beta) := (\beta_\spadesuit \vee \neg\beta_\clubsuit) \wedge \beta_\heartsuit$ is a sentence with $\mathcal{L}_\varphi := \{\spadesuit, \clubsuit, \heartsuit\}$.

The possibility space can be expressed in terms of the literals in the following way: $\Omega := \{\beta \in \{0,1\}^{\mathcal{L}_\Phi\cup\mathcal{L}_\psi} : \psi(\beta) = 1\}$, where ψ is a given sentence restricting

[1] The set Φ and objects that depend on it will in general also depend on the iteration number i: as gambles are effectively removed from $\mathcal{R}_0$ to obtain $\mathcal{R}_i$, basic functions appearing only in the expressions for removed gambles can be removed from Φ. We do not make this explicit in this paper to limit the notational burden.

the possible truth assignments—instantiations of β—that are valid elementary events. For example, there is a one-to-one relationship between $\{1,2,3\}$ and $\{\beta \in \{0,1\}^2 \colon \beta_1 \vee \beta_2 = 1\}$ with element vectors viewed as bit strings. Similarly, $\Omega_\Gamma \coloneqq \{\beta \in \Omega \colon \psi_\Gamma(\beta) = 1\}$ and $\Omega_\Delta \coloneqq \{\beta \in \Omega \colon \psi_\Delta(\beta) = 1\}$. In a propositional context, we use sentences instead of the corresponding sets.

Two important special cases [1,11, PSAT vs. CPA] are (i) no restrictions, i.e., ψ identically one, and (ii) $\mathcal{L}_\psi = \mathcal{L}_\Phi \coloneqq \Phi$ with $\phi(\beta) \coloneqq \beta_\phi$ for all ϕ in Φ. It is actually always possible to have $\phi(\beta) = \beta_\phi$ by using extensions $\mathcal{L} \coloneqq \mathcal{L}_\psi \cup \mathcal{L}_\Phi \cup \Phi$ and $\chi(\beta) \coloneqq \psi(\beta) \wedge \bigwedge_{\phi\in\Phi}\big((\beta_\phi \wedge \phi(\beta)) \vee (\neg\beta_\phi \wedge \neg\phi(\beta))\big)$. We will do so.

The propositional sentences we consider can in principle take any form. However, it is useful for algorithm design to write such sentences in some canonical 'normal' form. Given the connections of this work with PSAT, probabilistic satisfiability, we use the form standard in that field, CNF, conjunctive—or clausal—normal form. In this form, a sentence is written as a conjunction of clauses; a clause is a disjunction of literals and negated literals. Formally, $\chi(\beta) = \bigwedge_{m=1}^k \big(\bigvee_{\ell\in\mathcal{P}_m} \beta_\ell \vee \bigvee_{\ell\in\mathcal{N}_m} \neg\beta_\ell\big)$, where k is the number of conjuncts in the CNF of $\chi(\beta)$ and $\mathcal{P}_m \subseteq \mathcal{L}$ and $\mathcal{N}_m \subseteq \mathcal{L}$ with $\mathcal{P}_m \cap \mathcal{N}_m = \varnothing$ are the index sets of the mth conjunct's plain and negated disjuncts, respectively. The transformation of any sentence φ into CNF with a number of clauses linear in the number of operations in φ is an operation with polynomial complexity [10].

Row Generation in the Propositional Context. In the propositional context, we must in each iteration generate constraints of the form $\sum_{\phi\in\Phi}\kappa_\phi\bar{\beta}_\phi \gtreqless 0$, by generating some truth assignment $\bar{\beta}$ in $\{0,1\}^{\mathcal{L}}$. To generate deep cuts, the assignment $\bar{\beta}$ must be such that $|\sum_{\phi\in\Phi}\bar{\kappa}_\phi\bar{\beta}_\phi|$ is relatively large, where $\bar{\kappa}$ is the linear-program solution vector generated earlier in the iteration.

Generating valid truth assignments corresponds to solving a SAT problem: determining (whether there is) a β in $\{0,1\}^{\mathcal{L}}$ such that $\chi(\beta) = 1$. General SAT is NP-complete. There exist many specialized SAT solvers. Also, any SAT problem can be formulated as a binary linear programming problem:

$$\begin{aligned} \text{find} \quad & \beta \in \{0,1\}^{\mathcal{L}} \\ \text{such that} \quad & \textstyle\sum_{\ell\in\mathcal{P}_m}\beta_\ell + \sum_{\ell\in\mathcal{N}_m}(1-\beta_\ell) \geq 1 \quad \text{for all } 1 \leq m \leq k. \end{aligned} \tag{6}$$

Here, each constraint corresponds to a conjunct in the CNF of χ.

A SAT solver blindly generates instances, which will typically not result in deep cuts. To generate deep cuts, we need to take the constraint expression into account. A binary linear programming problem that does just this presents itself:

$$\begin{aligned} \text{optimize} \quad & \textstyle\sum_{\phi\in\Phi}\bar{\kappa}_\phi\beta_\phi \\ \text{subject to} \quad & \beta \in \{0,1\}^{\mathcal{L}} \\ & \textstyle\sum_{\ell\in\mathcal{P}_m}\beta_\ell + \sum_{\ell\in\mathcal{N}_m}(1-\beta_\ell) \geq 1 \quad \text{for all } 1 \leq m \leq k. \end{aligned} \tag{7}$$

In case of minimization, the β-instance generated is denoted $\bar{\delta}$; in case of maximization it is denoted $\bar{\gamma}$.

In essence, this Problem (7) is WPMaxSAT, weighted partial maximum SAT: now, part of the clauses are hard—those of χ—and part of them are weighted soft clauses—the $\bar{\kappa}_\phi\beta_\phi$, essentially—, whose total weight is maximized. General WPMaxSAT is NP-hard. Specialized WPMaxSAT solvers generally accept only positive integers as weights. The integral nature can be ensured by rescaling and rounding, so we can assume $\bar{\kappa}$ has integer components. For positivity, the weights are replaced by their absolute value; their sign is expressed through the corresponding soft clause: For maximization, one must use $\bar{\kappa}_\phi\beta_\phi$ if $\bar{\kappa}_\phi > 0$ and $|\bar{\kappa}_\phi|(\neg\beta_\phi)$ if $\bar{\kappa}_\phi < 0$. For minimization, one must use $\bar{\kappa}_\phi(\neg\beta_\phi)$ if $\bar{\kappa}_\phi > 0$ and $|\bar{\kappa}_\phi|\beta_\phi$ if $\bar{\kappa}_\phi < 0$. A big advantage of WPMaxSAT-based row generation is that $\bar{\kappa}$ satisfies all the constraints generated in earlier iterations, so these will not be generated again.

A Propositional CONEstrip Algorithm. We combine the original CONEstrip algorithm as described at the end of Section 2 with as constraint generators SAT for bootstrapping and WPMaxSAT subsequently:

Initialization. Set $i := 0$.

Are $\chi \wedge \psi_\Gamma$ and $\chi \wedge \psi_\Delta$ satisfiable?

Neither. The original problem is not well-posed. ***Stop***.

Either or both.

- If $\chi \wedge \psi_\Gamma$ is satisfied by some $\bar{\gamma}$, set $\Gamma_0 := \{\bar{\gamma}\}$; otherwise set $\Gamma_0 := \varnothing$.
- If $\chi \wedge \psi_\Delta$ is satisfied by some $\bar{\delta}$, set $\Delta_0 := \{\bar{\delta}\}$; otherwise set $\Delta_0 := \varnothing$.

Iterand.

1. Does the linear programming problem below have a solution $(\bar{\lambda}, \bar{\mu}, \bar{\sigma}, \bar{\kappa})$?

$$\begin{aligned}
\text{maximize}\quad & \textstyle\sum_{\mathcal{D}\in\mathcal{R}_i} \lambda_\mathcal{D} \\
\text{subject to}\quad & \lambda_\mathcal{D} \in [0,1] \quad\text{and}\quad \mu \in (\mathbb{R}_{\geq 0})^\mathcal{D} \quad \text{for all } \mathcal{D} \text{ in } \mathcal{R}_i \quad\text{and}\quad \sigma \in \mathbb{R}_{\geq 1} \\
& \textstyle\sum_{\mathcal{D}\in\mathcal{R}_i} \lambda_\mathcal{D} \geq 1 \quad\text{and}\quad \begin{cases} \sum_{\phi\in\Phi} \kappa_\phi\bar{\beta}_\phi \leq 0 \text{ for all } \bar{\beta} \text{ in } \Gamma_i \\ \sum_{\phi\in\Phi} \kappa_\phi\bar{\beta}_\phi \geq 0 \text{ for all } \bar{\beta} \text{ in } \Delta_i \end{cases} \\
& \lambda_\mathcal{D} \leq \mu_\mathcal{D} \text{ for all } \mathcal{D} \text{ in } \mathcal{R}_i \\
\text{where}\quad & \kappa_\phi := \left(\textstyle\sum_{\mathcal{D}\in\mathcal{R}_0} \sum_{g\in\mathcal{D}} \mu_{\mathcal{D},g} g_\phi\right) - \sigma f_\phi \in \mathbb{R} \text{ for all } \phi \text{ in } \Phi.
\end{aligned} \tag{8}$$

   No. $f \notin \underline{\mathcal{R}_0}$. ***Stop***.

   Yes. Let $\mathcal{Q} := \{\mathcal{D} \in \mathcal{R}_i \colon \bar{\lambda}_\mathcal{D} = 0\}$ and set $\mathcal{R}_{i+1} := \mathcal{R}_i \setminus \mathcal{Q}$.

2.
   - If $\Gamma_i \neq \varnothing$, let $\bar{\gamma}$ be the solution of the WPMaxSAT for maximizing $\sum_{\phi\in\Phi} \bar{\kappa}_\phi\beta_\phi$ under the hard clauses $\chi \wedge \psi_\Gamma$; set $\Gamma_{i+1} := \Gamma_i \cup \{\bar{\gamma}\}$. Otherwise, set $\bar{\gamma}$ identically zero.
   - If $\Delta_i \neq \varnothing$, let $\bar{\delta}$ be the solution of the WPMaxSAT for minimizing $\sum_{\phi\in\Phi} \bar{\kappa}_\phi\beta_\phi$ under the hard clauses $\chi \wedge \psi_\Delta$; set $\Delta_{i+1} := \Delta_i \cup \{\bar{\delta}\}$. Otherwise, set $\bar{\delta}$ identically zero.

   Is $\sum_{\phi\in\Phi} \bar{\kappa}_\phi\bar{\gamma}_\phi \leq 0 \leq \sum_{\phi\in\Phi} \bar{\kappa}_\phi\bar{\delta}_\phi$ and $\{\mathcal{D} \in \mathcal{Q} \colon \bar{\mu}_\mathcal{D} = 0\} = \mathcal{Q}$?

   Yes. Set $t := i + 1$; $f \in \underline{\mathcal{R}_t} \subseteq \underline{\mathcal{R}_0}$. ***Stop***.

   No. Increase i's value by 1. ***Reiterate***.

In this algorithm, the cone stripping and constraint generation iterations are merged. This can be done because if $\lambda_{\mathcal{D}} = 0$ for some $\mathcal{D}$ in $\mathcal{R}_i$, then certainly $\lambda_{\mathcal{D}} = 0$ when additional constraints are added.

The complexity of the algorithm is determined by (i) the number of iterations: polynomial in $|\varPhi|$ [6,11] and linear in $|\mathcal{R}_0|$; (ii) the complexity of the 'master' linear program: polynomial in $|\varPhi|$, $|\mathcal{R}_0|$, and $\sum_{\mathcal{D}\in\mathcal{R}_0}|\mathcal{D}|$; and (iii) the complexity of constraint generation—in the worst case exponential in k and polynomial in $|\mathcal{L}|$ [2]. So we have replaced a procedure with guaranteed exponential complexity due to an exponential number of constraints by a procedure that often has decent practical complexity. Because of the reduction to standard problems—SAT and WPMaxSAT, or binary linear programming—advances in solvers for those problems can directly be taken advantage of.

Before an implementation can be called mature, it must support 'restarting' of the SAT and WPMaxSAT solvers, meaning that preprocessing—such as variable elimination—needs to be done only once, before the first iteration. This could provide efficiency gains similar to those obtained by algorithms for probabilistic reasoning from the Italian school [1,3].

Optimization Problems. Again, we can solve optimization problems with continuous objective functions over the general cone: First ignore the objective and run the propositional CONEstrip algorithm on the representation $\mathcal{R}_0$ to obtain a terminal representation $\mathcal{R}_t$ and terminal instance sets $\varDelta_t$ and $\varGamma_t$. Then we optimize over the topological closure of the terminal general cone $\underline{\mathcal{R}_t}$:

Initialization. Set $i := t$.
Iterand.

1. Solve the following optimization problem to obtain the solution $(\bar{\mu}, \bar{\kappa})$:

$$\begin{aligned} &\text{optimize} && \text{a continuous function of } \mu \\ &\text{subject to} && \mu \in (\mathbb{R}_{\geq 0})^{\cup\mathcal{R}_t} \quad \text{and} \quad \begin{cases} \sum_{\phi\in\varPhi} \kappa_\phi \bar{\beta}_\phi \leq 0 \text{ for all } \bar{\beta} \text{ in } \varGamma_i \\ \sum_{\phi\in\varPhi} \kappa_\phi \bar{\beta}_\phi \geq 0 \text{ for all } \bar{\beta} \text{ in } \varDelta_i \end{cases} \\ &\text{where} && \kappa_\phi := \left(\textstyle\sum_{g\in\cup\mathcal{R}_t} \mu_g g_\phi\right) - f_\phi \in \mathbb{R} \text{ for all } \phi \text{ in } \varPhi. \end{aligned} \tag{9}$$

2.
   - If $\varGamma_i \neq \varnothing$, let $\bar{\gamma}$ be the solution of the WPMaxSAT for maximizing $\sum_{\phi\in\varPhi} \bar{\kappa}_\phi \beta_\phi$ under the hard clauses $\chi \wedge \psi_\varGamma$; set $\varGamma_{i+1} := \varGamma_i \cup \{\bar{\gamma}\}$. Otherwise, set $\bar{\gamma}$ identically zero.
   - If $\varDelta_i \neq \varnothing$, let $\bar{\delta}$ be the solution of the WPMaxSAT for minimizing $\sum_{\phi\in\varPhi} \bar{\kappa}_\phi \beta_\phi$ under the hard clauses $\chi \wedge \psi_\varDelta$; set $\varDelta_{i+1} := \varDelta_i \cup \{\bar{\delta}\}$. Otherwise, set $\bar{\delta}$ identically zero.

   Is $\sum_{\phi\in\varPhi} \bar{\kappa}_\phi \bar{\gamma}_\phi \leq 0 \leq \sum_{\phi\in\varPhi} \bar{\kappa}_\phi \bar{\delta}_\phi$?
   Yes. $\bar{\mu}$ is optimal. ***Stop***.
   No. Increase i's value by 1. ***Reiterate***.

When the objective function is linear in μ, we now get an iteration of linear programming problems.

4 Some Special Cases

The propositional CONEstrip algorithm presented in the preceding section can be directly applied to problems in desirability theory. We here make a link with other, more established theories by describing how the standard problems in those theories can be encoded as problems in desirability theory. In all of the cases discussed, ψ_Γ is identically one and ψ_Δ is identically zero.

A probability assessment $P(\phi) = p_\phi$ for some event ϕ corresponds to an open cone with representation $\mathcal{D}_{\phi,p_\phi} \coloneqq \{\phi - p_\phi 1, p_\phi 1 - \phi, 1\}$, where 1 denotes the constant gamble 1. For the classical PSAT problem, assessments are given for a set Φ of events and the questions is asked whether a probability mass function on the possibility space exists that satisfies these assessments. This problem can be solved applying the propositional CONEstrip algorithm to $\mathcal{R}_0 \coloneqq \{\mathcal{D}_{\phi,p_\phi} : \phi \in \Phi\}$ and f identically zero: there is a satisfying probability mass function if and only if the problem is infeasible; no such mass function is explicitly constructed.

This setup can be generalized to conditional probability assessments $P(\phi|\varphi) = p_{\phi|\varphi}$; these correspond to $\mathcal{D}_{(\phi|\varphi),p_{\phi|\varphi}} \coloneqq \{\phi \wedge \varphi - p_{\phi|\varphi}\varphi, p_{\phi|\varphi}\varphi - \phi \wedge \varphi, \varphi\}$. Given assessments for a set Φ_C of conditional events, the propositional algorithm can be applied to $\mathcal{R}_0 \coloneqq \{\mathcal{D}_{(\phi|\varphi),p_{\phi|\varphi}} : (\phi|\varphi) \in \Phi_\mathrm{C}\}$ and f identically zero to determine whether there exists a satisfying full conditional probability mass function.

A further generalization is to lower (and upper) conditional expectations of gambles g given as linear combinations of sentences: $\underline{P}(g|\varphi) = p_{g|\varphi}$ corresponds to $\mathcal{D}_{(g|\varphi),p_{g|\varphi}} \coloneqq \{g \wedge \varphi - p_{g|\varphi}\varphi, \varphi\}$. When given a set of such assessments, the propositional CONEstrip algorithm can be applied to check whether they incur partial loss. Also, in this context, to calculate the lower expectation via natural extension for a gamble h conditional on an event ξ, we need to include the set $\mathcal{D}_\xi \coloneqq \{-\xi, 0, \xi\}$ in the representation $\mathcal{R}_0$, use $f \coloneqq h \wedge \xi$, and maximize the objective $\mu_\xi - \mu_{-\xi}$.

5 Conclusions

We presented an algorithm for checking consistency of and doing inference with uncertainty models based on fully general finitely generated sets of desirable gambles, with gambles specified as linear combinations of propositional sentences. It is designed to avoid a sure exponential computational complexity. As far as we know, it is the first such algorithm presented in the literature.

We have made a preliminary implementation and verified that the algorithm works. Ongoing work consists in improving the implementation and setting up numerical experiments to test the practical efficiency relative to the standard CONEstrip algorithm—for the general problem—and other algorithms—for special cases. Further research directions will be determined by their results.

Acknowledgments. The author wishes to thank Fabio Cozman for useful discussion of ideas at the basis of this research, Marjan van den Akker for her enlightening explanation of linear programming row generation techniques, and the reviewers for their many useful comments.

References

1. Baioletti, M., Capotorti, A., Tulipani, S., Vantaggi, B.: Simplification rules for the coherent probability assessment problem. Annals of Mathematics and Artificial Intelligence 35(1-4), 11–28 (2002)
2. Bansal, N., Raman, V.: Upper bounds for MaxSat: Further improved. In: Algorithms and Computation, Lecture Notes in Computer Science, vol. 1741, pp. 247–258. Springer (1999)
3. Biazzo, V., Gilio, A., Lukasiewicz, T., Sanfilippo, G.: Probabilistic logic under coherence: complexity and algorithms. Annals of Mathematics and Artificial Intelligence 45(1–2), 35–81 (2005)
4. Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.): Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185. IOS Press (2009)
5. Cozman, F.G., di Ianni, L.F.: Probabilistic satisfiability and coherence checking through integer programming. In: van der Gaag, L.C. (ed.) Symbolic and Quantitative Approaches to Reasoning with Uncertainty. Lecture Notes in Artificial Intelligence, vol. 7958, p. 145–156. Springer (2013)
6. Georgakopoulos, G., Kavvadias, D., Papadimitriou, C.H.: Probabilistic satisfiability. Journal of Complexity 4, 1–11 (1988)
7. Gomes, C.P., Kautz, H., Sabharwal, A., Selman, B.: Satisfiability solvers. In: van Harmelen, F., Lifschitz, V., Porter, B. (eds.) Handbook of Knowledge Representation, chap. 2, pp. 89–134. Elsevier (2008)
8. Hansen, P., Jaumard, B., de Aragão, M.P., Chauny, F., Perron, S.: Probabilistic satisfiability with imprecise probabilities. International Journal of Approximate Reasoning 24(2–3), 171–189 (2000)
9. Manquinho, V., Marques-Silva, J., Planes, J.: Algorithms for weighted boolean optimization. In: Kullmann, O. (ed.) Theory and Applications of Satisfiability Testing - SAT 2009. Lecture Notes in Computer Science, vol. 5584, pp. 495–508. Springer (2009)
10. Prestwich, S.: CNF encodings. In: Biere, A., Heule, M., van Maaren, H., Walsh, T. (eds.) Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications, vol. 185, chap. 2, pp. 75–98. IOS Press (2009)
11. Pretolani, D.: Probability logic and optimization SAT: The PSAT and CPA models. Annals of Mathematics and Artificial Intelligence 43(1-4), 211–221 (2005)
12. Quaeghebeur, E.: The CONEstrip algorithm. In: Kruse, R., Berthold, M.R., Moewes, C., Ángeles Gil, M., Grzegorzewski, P., Hryniewicz, O. (eds.) Synergies of Soft Computing and Statistics for Intelligent Data Analysis. Advances in Soft Computing, vol. 190, p. 45–54. Springer (2013), `http://hdl.handle.net/1854/LU-3007274`
13. Quaeghebeur, E.: Desirability. In: Coolen, F.P.A., Augustin, T., de Cooman, G., Troffaes, M.C.M. (eds.) Introduction to Imprecise Probabilities. Wiley (2014)
14. Quaeghebeur, E., de Cooman, G., Hermans, F.: Accept & reject statement-based uncertainty models (submitted), `http://arxiv.org/abs/1208.4462`
15. Walley, P.: Statistical reasoning with imprecise probabilities, Monographs on Statistics and Applied Probability, vol. 42. Chapman & Hall (1991)
16. Walley, P.: Towards a unified theory of imprecise probability. International Journal of Approximate Reasoning 24(2–3), 125–148 (2000)
17. Walley, P., Pelessoni, R., Vicig, P.: Direct algorithms for checking consistency and making inferences from conditional probability assessments. Journal of Statistical Planning and Inference 126(1), 119–151 (2004)