Probabilistic Satisfiability and Coherence Checking through Integer Programming

Fabio Gagliardi Cozman and Lucas Fargoni di Ianni

Universidade de Sao Paulo
Av. Prof. Mello Moraes, 2231, Sao Paulo, SP - Brazil

Abstract. This paper presents algorithms based on integer programming, both for probabilistic satisfiability and coherence checking. That is, we consider probabilistic assessments for both standard probability measures (Kolmogorovian setup) and full conditional measures (de Finettian coherence setup), and in both cases verify satisfiability/coherence using integer programming. We present empirical evaluation of our method, with evidence of phase-transitions.

1 Introduction

The analysis of arguments that combine propositions and probabilities has deserved attention for quite some time. For instance, in Boole's work [8] we find interesting examples such as:

> The probability that it thunders upon a given day is p, the probability that it both thunders and hails is q, but of the connexion of the two phenomena of thunder and hail, nothing further is supposed to be known. Required the probability that it hails on the proposed day.

Here we have propositions A and B, assessments $P(A) = p$ and $P(A \cap B) = q$. Boole asks for $P(B)$ and obtains the tight interval $[q, 1-(p-q)]$. The assessments are *coherent*: there is a probability measure that *satisfies* them.

Suppose we have propositional sentences $\{\phi_i\}_{i=1}^{M}$, each containing a subset of atomic propositions $\{A_j\}_{j=1}^{n}$. We may associate one or more of these sentences with probabilities, writing for instance $P(\phi_i) = \alpha_i$. To establish semantics for these assessments, we consider a probability measure over the set of truth assignments. The *Probabilistic Satisfiability (PSAT)* problem is to determine whether it is possible to find a probability measure over truth assignments such that all assessments are satisfied [14, 18–20, 23]. When assessments involve conditional probabilities such as $P(\phi'_i|\phi''_i) = \alpha_i$, there are two paths to follow. The Kolmogorovian setup reduces such assessments to ratios of probabilities. The other path is to use de Finetti's theory of coherent probabilities, where full conditional measures are used to interpret conditional assessments [11, 12, 32]. The *Coherent Probability Assessment (CPA)* problem is to determine whether it is possible to find a full conditional measure that satisfies all assessments [3, 4].

Probabilistic satisfiability and coherence checking are central problems in reasoning under uncertainty. They serve as a foundation for logical and probabilistic inference, as a basis for probabilistic rules [30], and as an initial necessary step in the understanding of combinations of first-order logic and probabilities [21, 27, 31].

The most direct way to solve a PSAT problem is to write down the problem as a linear consistency problem [19]. The difficulty is that the resulting linear program may be too large. One may resort to column generation techniques [25], or to inference rules that capture probabilistic relationships [3, 16], or even to combinations of column generation and inference rules [24]. There is also a different approach to probabilistic satisfiability that tackles it by transformation into logical satisfiability [15].

In this paper we present another approach to Probabilistic Satisfiability, where the original problem is written as an integer linear program of size that is polynomial on the size of the original problem. The algorithm is extremely simple to state; our implementation shows that it is quite efficient compared to alternatives. Using our implementation we have studied the issue of phase transitions. We report these experiments in this paper.

Section 2 summarizes necessary background. Our basic algorithm is described in Section 3. Implementation and experiments, with a discussion of phase transitions, are presented in Section 4. Conditional probabilities are handled in Section 5, and inference problems are discussed in Section 6.

2 SAT and PSAT

Consider n atomic propositions A_j and M sentences ϕ_i in propositional logic. If a truth assignment ω is such that sentence ϕ is True, write $\omega \models \phi$. The Satisfiability (SAT) problem is to determine whether or not there exists a truth assignment to all variables such that all sentences evaluate to True [10, 17]. If every sentence ϕ_i is a conjunction of clauses, then we have a SAT problem in *Conjunctive Normal Form (CNF)*. A SAT problem in CNF is a k-SAT problem when each clause has k literals. The 2-SAT problem has a polynomial solution, while k-SAT is NP-complete for $k > 2$.

For a fixed n, m and k, one may generate a random k-SAT with n propositions and a single sentence in CNF with m clauses, as follows. For each one of the m clauses: select k variables at random, and for each variable produce a literal that may be negated or not, with probability half. There has been intense study of *phase transition* phenomena in random k-SAT; that is, study of the observed fact that for small values of m/n the probability that a random k-SAT is satisfiable tends to one as n grows (at fixed m/n), while for large values of m/n the probability that a random k-SAT is satisfiable tends to zero as n grows. Moreover, in the regions where satisfiability has probability approaching zero or one we observe that generated random k-SAT problems can be easily solved, while in the transition between the two regions we find hard problems.

Suppose that some sentences, say ϕ_1 to ϕ_q, for $q \leq M$, are associated with probabilities through *assessments* of the form $P(\phi_i) \bowtie \alpha_i$, where $\bowtie$ is one of $\geq$, $=$, $\leq$. The semantics of such an assessment is as follows. Take the set of 2^n truth assignments that can be generated for the n propositions. A probability measure P over this set satisfies the assessments if, for each assessment $P(\phi_i) \bowtie \alpha_i$,

$$\sum_{\omega \models \phi_i} P(\omega) \bowtie \alpha_i. \tag{1}$$

The Probabilistic Satisfiability (PSAT) problem is to determine whether a given set of sentences and probabilistic assessments can be satisfied. That is, to determine whether there is a probability measure over those truth assignments that satisfy sentences not associated with probabilities, such that all assessments are satisfied by this probability measure. The k-PSAT problem is a PSAT problem where each sentence is in CNF and where each clause has k literals. The k-PSAT is NP-complete for *all* values of $k > 1$; note that even for $k = 2$ we obtain NP-completeness [23]. A few polynomial special cases of PSAT are known [2].

There are many algorithms for PSAT. The most obvious one is to write down M constraints of the form (1), one for each sentence; some will be actually associated with assignments $P(\phi_i) \bowtie \alpha_i$, while others will encode "pure" logical sentences as $P(\phi_i) = 1$. Each constraint can be written as

$$\sum_{j=1}^{2^n} I_{\phi_i}(\omega_j) P(\omega_j) \bowtie \alpha_i, \qquad \text{where } I_{\phi_i}(\omega_j) = \begin{cases} 1 & \text{if } \omega_j \models \phi_i \\ 0 & \text{otherwise,} \end{cases} \tag{2}$$

while truth assignments ω_j are ordered from 1 to 2^n (say by the n-bit binary number obtained by writing 0 for False and 1 for True as assigned to $A_1, \ldots, A_n$). Add to these M linear constraints the necessary constraints $\sum_\omega P(\omega) = 1$ and $P(\omega) \geq 0$ for all ω. Probabilistic Satisfiability is then obtained when the resulting set of linear constraints has a solution. The challenge is that we have 2^n truth assignments, so the size of the linear constraints is exponential in the input.

The most efficient algorithms for PSAT combine linear programming techniques and inference rules to simplify the problem [24]. These algorithms use the fact that a PSAT problem is satisfiable if and only if there is a probability measure that assigns positive probability mass to $(M + 1)$ truth assignments; all other truth assignments get zero probability mass [18]. Hence we can write down a $(M + 1) \times (M + 1)$ matrix $\mathbf{C}$ and write the PSAT problem as feasibility of $\mathbf{C}\mathbf{p} \bowtie \boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ denotes a vector of values α_i and $\bowtie$ refers to $\geq$, $=$ or $\leq$ as appropriate. Each column of $\mathbf{C}$ corresponds to a truth assignment; the challenge is to select $(M + 1)$ truth assignments. This is done through column generation techniques from linear programming [5]. Initially a set of $(M + 1)$ columns is selected, and then pivoting operations exchange columns until the problem is determined to be satisfiable or not. At each pivoting operation, a column is removed from $\mathbf{C}$, and the choice of the column to enter $\mathbf{C}$ happens through an auxiliary optimization problem (there are several possible formulations for this auxiliary problem) [23, 24]. Performance improvements are obtained if column generation is preceded by application of inference rules.[1] This combination has produced the best results so far, being able to solve PSAT problems with up to 200 propositions and 800 clauses, each one of them a clause associated with a probability.

An entirely different approach to PSAT has been developed by Finger and De Bona [15]; here the selection of columns of $\mathbf{C}$ is reduced to a SAT problem. All operations with linear constraints are encoded into SAT by careful analysis of numerical precision, and a SAT solver is used to solve the PSAT problem.

The resulting methods are fairly sophisticated and require numerical care. Moreover, the extension of such methods to conditional probabilities in de Finetti's coherency framework is difficult, and existing methods require sequences of linear programs (Section 5). In this paper we propose a novel approach that addresses these concerns.

A PSAT is in *Normal Form* if a single sentence ϕ is given, and each probabilistic assessment is an equality associated with a single proposition (that is, every probabilistic assessment is of the form $P(A_i) = \alpha_i$) [15]. Even though this form may seem restrictive, every PSAT can be brought to it with polynomial effort: basically, for each assessment $P(\phi_i) \bowtie \alpha_i$, introduce if necessary fresh propositions to transform the assessment into $P(\phi'_i) = \alpha_i$; then introduce a new proposition A'_i and exchange the original assessment by a sentence $A'_i \Leftrightarrow \phi'_i$ and an assessment $P(A'_i) = \alpha_i$; finally, generate a single sentence ϕ that is a conjunction of all previous sentences. Every k-PSAT for $k > 2$ can be reduced to Normal Form with q assessments $P(A_i) = \alpha_i$ plus one CNF ϕ consisting of clauses with exactly 3 literals each.

3 PSAT through Integer Programming

Assume our PSAT problem is in Normal Form with assessments $\{P(A_j) = \alpha_j\}_{j=1}^{q}$ and a sentence ϕ in CNF with m clauses, each clause with k literals. So our problem is parameterized by the number of propositions n, the number of assessments q, the number of clauses m, and the number of literals per clause k. Such a parameterized Normal Form neatly separates the probabilistic and the propositional aspects of Probabilistic Satisfiability.

Our problem is: find the $(q + 1)$ columns of $\mathbf{C}$, each one corresponding to a truth assignment ω such that $\omega \models \phi$, in such a way that $\mathbf{C}\mathbf{p} = \boldsymbol{\alpha}$.

Hence we have $(q + 1)^2$ optimization variables (elements of $\mathbf{C}$ to look for); all of them are binary with values 0 and 1. As noted previously, Finger and De Bona reduce the search for these variables to a SAT problem [15]. We instead find $\mathbf{C}$ by solving an integer program.

Consider looking for the jth column of $\mathbf{C}$; denote it by $\mathbf{C}_j$. Such a column corresponds to a truth assignment that satisfies ϕ. We explore the well known connection between SAT and integer programming to find such a truth assignment [10]. Start by generating a vector $\mathbf{a}_j$ with n binary variables $\{a_{i,j}\}_{i=1}^{n}$, all

[1] An example of an inference rule [24]: if $P(A_1) \in [\underline{\alpha}_1, \overline{\alpha}_1]$ and $P(\neg A_1 \vee A_2) \in [\underline{\alpha}_1, \overline{\alpha}_2]$ for $\overline{\alpha}_1 + \overline{\alpha}_2 \geq 1$, then $P(A_2) \in [\max(0, \underline{\alpha}_1 + \underline{\alpha}_2 - 1), \min(1, \overline{\alpha}_2)]$.

```
1: procedure PSAT-IP(propositions $\{A_j\}_{j=1}^n$, assessments $\{P(A_i) = \alpha_i\}_{i=1}^q$, sentence $\phi$ in CNF with $m$ clauses)
2:         ▷ Variables $a_{i,j}$ are binary; variables $b_{i,j}$ and $p_j$ are real-valued in $[0,1]$.
3:     for $j \in \{1,\dots,q+1\}$ and each clause $(\vee_{l'=1}^{k'} A_{i_{l'}}) \vee (\vee_{l''=1}^{k''} \neg A_{i_{l''}})$ of $\phi$ do
4:         Generate linear constraint $(\sum_{l'=1}^{k'} a_{i_{l'},j}) + (\sum_{l''=1}^{k''}(1 - a_{i_{l''},j})) \geq 1$.
5:     for $i \in \{1,\dots,q+1\}$ do
6:         Generate linear constraint $\sum_{j=1}^{q+1} b_{i,j} = \alpha_i$.
7:         for $j \in \{1,\dots,q+1\}$ do
8:             Generate linear constraints $0 \leq b_{i,j} \leq a_{i,j}$ and $a_{i,j} - 1 + p_j \leq b_{i,j} \leq p_j$.
9:     return Satisfiable if linear constraints have a solution, Unsatisfiable otherwise.
```

Fig. 1. PSAT solution based on integer linear program.

with values 0 and 1. Now take one clause of ϕ; suppose it is written as

$$(\vee_{l'=1}^{k'} A_{i_{l'}}) \vee (\vee_{l''=1}^{k''} \neg A_{i_{l''}}).$$

For this clause, generate the linear inequality:

$$\left(\sum_{l'=1}^{k'} a_{i_{l'},j}\right) + \left(\sum_{l''=1}^{k''} (1 - a_{i_{l''},j})\right) \geq 1. \tag{3}$$

Consider the m inequalities generated this way (one per clause). A vector $\mathbf{a}_j$ that satisfies these m inequalities yields a truth assignment for ϕ by assigning True to A_i when $a_{i,j}$ is one, and assigning False to A_i when $a_{i,j}$ is zero. Note that the elements of $\mathbf{C}_j$ are exactly $a_{1,j}$ to $a_{q,j}$.

We generate the whole matrix $\mathbf{C}$ by generating $(q+1)$ sets of variables $\mathbf{a}_j$ and their related inequalities. We now have inequalities for all elements of $\mathbf{C}$, and we need to solve $\mathbf{C}\mathbf{p} = \boldsymbol{\alpha}$. To do so, note that each row of $\mathbf{C}$ represents an equality as follows:

$$\sum_{j=1}^{q+1} a_{i,j} p_j = \alpha_i, \tag{4}$$

where p_j denotes the jth element of $\mathbf{p}$. The challenge is to reduce the bilinear term $a_{i,j}p_j$ to linear constraints. We do that by introducing a new fresh variable $b_{i,j}$ and the constraints:

$$0 \leq b_{i,j} \leq a_{i,j} \qquad \text{and} \qquad a_{i,j} - 1 + p_j \leq b_{i,j} \leq p_j. \tag{5}$$

Note that if $a_{i,j} = 0$, then $b_{i,j} = 0$; and if $a_{i,j} = 1$, then $b_{i,j} = p_j$.

The whole algorithm is presented in Figure 1; it basically collects constraints from Expressions (3), (4), and (5). The algorithm produces an integer linear program that has a solution if and only if the original PSAT problem is satisfiable.

4 Implementation, Experiments, and Phase Transition

We have coded our PSAT method using the Java language with calls to CPLEX version 12, and run experiments in iMac computers with 4GBytes of memory. The algorithm is very compact, using only 45 lines of code (basically a direct translation of the algorithm in Figure 1 into CPLEX calls).

We focused on two values of k, namely, 2 and 3. We investigated $k = 2$ because 2-SAT is polynomial and 2-PSAT is NP-complete, a property not shared by any other k-PSAT. And we investigated $k = 3$ because any PSAT problem can be polynomially reduced to a 3-PSAT problem; in fact, Finger and De Bona pay attention to 3-PSAT for this reason [15].

Additionally, we were particularly interested in investigating phase transition phenomena. Until the work of Baioletti et al. [4], and Finger and De Bona [15], there was no evidence of phase transition in the literature. Consequently it makes more sense at this stage to examine the behavior of PSAT for various values of n, m and q, rather than to randomly try out large problems that may in the end be easy.

Figure 2 summarizes a number of experiments for $k = 2$. In all of them, PSAT problems were randomly generated from parameters n, m, q and k: m clauses with k literals each were randomly generated by selecting propositions randomly out of the n propositions; each literal was negated or not with probability $1/2$; finally, the first q propositions were associated with probabilities randomly selected in the interval $[0, 1]$. Each point in each graph conveys mean values for 50 different random PSAT problems. We set a time limit of 10 minutes per problem; some of the more difficult problems did not finish within this time limit.

The left graphs in Figure 2 show typical behavior for random 2-PSAT: the darker line indicates the percentage of satisfiable problems, and the lighter line indicates mean time spent in their solution (mean of 50 distinct random PSAT problems). The top graph deals with 2-PSAT problems with 1000 variables and up to 1500 clauses; these are rather large problems and the phase transition phenomenon is clear (note that we are using larger values of q than in the previous investigation by Finger and De Bona [15]). The lower graph conveys the same information, but now for $n = 100$. The main point to note is that the phase transition seems to occur for much smaller m/n. Indeed the presence of probabilities seems to create relationships between n, q and m in ways that are not observed in 2-SAT (where the phase transition occurs for $m/n = 1$). An interesting display of this phenomenon can be found in the right graph, where one can see that the phase transition is affected by q.

Similar results are displayed in Figure 3. In the left graph we see typical phase transition behavior, now centered around $m/n \approx 3.5$. The reason we show this particular graph (with $n = 40$, $q = 4$) is that the same experiment is reported by Finger and De Bona [15]; their reported times are about 10 times larger than ours. The right graph shows the change in the location of phase transition as q varies, similarly to what happens with 2-PSAT.

To give a better feel of the times involved in solving PSAT problems with our method, Table 1 summarizes a large variety of tests; each entry is the mean

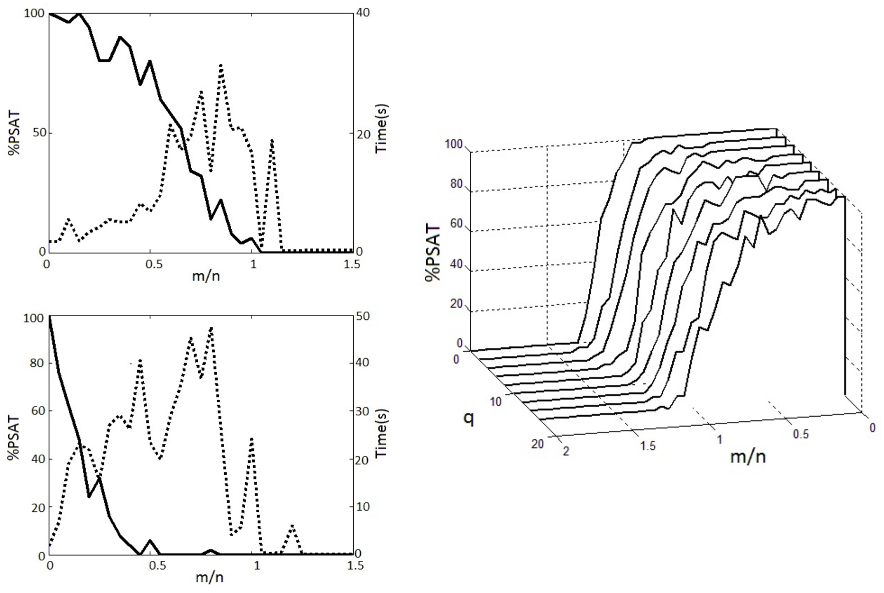

Fig. 2. Experiments with 2-PSAT.

of 50 distinct PSAT problems. Note that it is not correct to expect that the larger the problem, the more time it takes; due to phase transition, some large problems may be easy, while some apparently small problems may be hard.

5 Checking Coherence of Conditional Assessments through Integer Programming

Suppose that conditional assessments $P(\phi'_i|\phi''_i) = \alpha_i$ must be processed. In the standard, Kolmogorovian style, probability theory, this assessment means that $P(\phi'_i \wedge \phi''_i)/P(\phi''_i) = \alpha_i$ if $P(\phi''_i) > 0$. This holds if and only if

$$P(\phi'_i \wedge \phi''_i) - \alpha_i P(\phi''_i) = 0. \qquad (6)$$

The only change from "unconditional" PSAT is that each element of the matrix $\mathbf{C}$ is now a linear expression. Indeed, if we only take conditional assessments of the form $P(A'_i|A''_i) = \alpha_i$, then the element $\mathbf{C}_{i,j}$ is given by the nonlinear expression $a'_{i,j}a''_{i,j} - \alpha_i a''_{i,j}$, where $a'_{i,j}$ and $a''_{i,j}$ are binary variables corresponding to propositions $A'_{i,j}$ and $A''_{i,j}$ respectively. To handle this, the only change in our previous algorithm is that the constraints in its line 6 must be replaced by $\sum_{j=1}^{q+1} \left(b'_{i,j} - \alpha_i b''_{i,j}\right) = 0$, and constraints in line 8 must be replaced by

$$0 \le b'_{i,j} \le a'_{i,j}, \qquad 0 \le b'_{i,j} \le a''_{i,j}, \qquad a'_{i,j} + a''_{i,j} - 2 + p_j \le b'_{i,j} \le p_j,$$

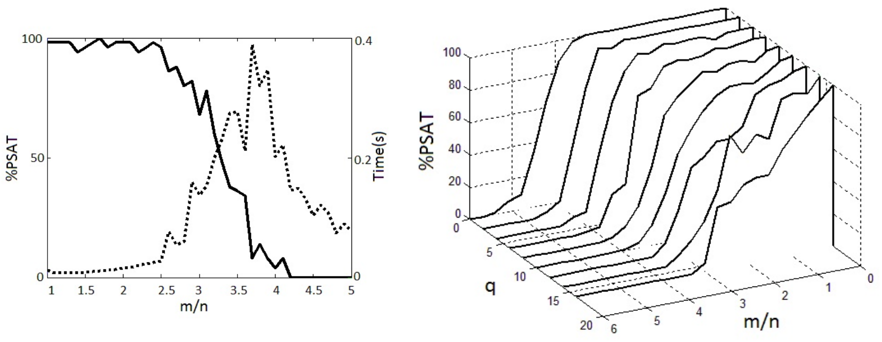

Fig. 3. Experiments with 3-PSAT.

n	q	m	2-PSAT, mean time (sec.)	3-PSAT, mean time (sec.)
500	25	500	2.5441801	1.8612529
500	25	750	0.1588879	1.8934227
500	25	1000	0.1802416	12.0530050
500	50	500	18.271062	16.1097538
500	50	750	0.3348024	48.0028159
500	50	1000	0.9317918	177.037949
750	25	500	8.9984422	4.6699495
750	25	750	7.6501204	5.5394115
750	25	1000	0.2109049	3.6071930
750	50	500	0.3424100	19.9149997
750	50	750	0.4221426	23.5807273
750	50	1000	1.0847125	30.1025869
1000	25	500	2.4382244	2.1816687
1000	25	750	1.5742295	2.2077639
1000	25	1000	0.5740323	3.1606671
1000	25	1500	0.3041526	3.0211587
1000	25	2000	0.5616025	28.1456910
1000	50	500	3.5237384	16.2340889
1000	50	750	1.0613374	15.0277582
1000	50	1000	0.4471337	17.4252168

Table 1. Experiments with 2-PSAT and 3-PSAT.

$$0 \leq b''_{i,j} \leq a''_{i,j}, \qquad a''_{i,j} - 1 + p_j \leq b''_{i,j} \leq p_j.$$

This Kolmogorovian setup requires some care when interpreting conditional assessments. Suppose first that A''_i has probability zero in every probability measure that satisfies all assessments. In this case we may take $P(A'_i|A''_i) = \alpha_i$ to be a misguided assignment to a quantity that should really be left undefined. However suppose that some satisfying probability measures assign zero probability to A''_i, while others do not. The most reasonable interpretation of this situation is that only those probability measures that assign positive probability to A''_i should be retained; the others do not satisfy the fact that $P(A'_i|A''_i)$ has been actually assessed.

An entirely different view of conditional probability can be found in de Finetti's theory of coherence. Here conditional probability is not a derived concept, but rather the primary object of interest. Assessments can be given on arbitrary events, and coherence of assessments is equated to existence of a *full* *conditional measure* that satisfies the assessments. The selection of a particular conditional measure imposes considerable structure on events, while de Finetti's approach assumes little algebraic structure on the assessments [7, 11]. A full conditional measure $P : \mathcal{B} \times (\mathcal{B}\backslash\emptyset) \to \Re$, where $\mathcal{B}$ is a Boolean algebra over a set Ω, is a two-place set-function such that for every nonempty event C [13]:

- $P(C|C) = 1$ and $P(A|C) \geq 0$ for all A;
- $P(A \cup B|C) = P(A|C) + P(B|C)$ when $A \cap B = \emptyset$;
- $P(A \cap B|C) = P(A|B \cap C)P(B|C)$ when $B \cap C \neq \emptyset$.

Note that conditioning is defined for every nonempty event; whenever the conditioning event is Ω, we suppress it and write the "unconditional" probability $P(A)$. There are other names for full conditional measures in the literature, such as *conditional probability measures* [26]; and *complete conditional probability systems* [29]. Full conditional measures have been applied in economics [22, 29], philosophy [1, 28], artificial intelligence [9].

So, suppose we have the same propositions and assessments as before, and events are interpreted as sets of truth assignments. We say the assessments are *coherent* if there is a full conditional measure that satisfies them [11, 12]. Note that a set of assessments may be coherent even if $P(B) = 0$ and $P(A|B) = \alpha > 0$; a probability measure that assigns probability zero to a conditioning event need not be discarded.

There are algorithms for coherency checking that basically work by dividing the space of truth assignments into "layers": the first layer contains the truth assignments with positive unconditional probability; the second layer contains the truth assignments with positive conditional probability given the complement of the first layer, and so on [9, 11]. For each layer an appropriately specified PSAT problem is solved, and the collection of PSAT problems yields the desired coherency check. Alternative algorithms employ local rules that mimic logical inference [3, 4]. To the extent that these methods solve linear programs in intermediate steps, and these linear programs are of size exponential in the input, the reductions to integer programming that we have explored before can be used.

To illustrate the last comment, consider the formulation of coherence checking that is due to Walley et al. [32]. They consider that assessments are of the form $P(A'_i|A''_i) \geq \alpha_i$, and show that existence of a satisfying full conditional measure is equivalent to:

$$\sup_{\omega \models S_\lambda} \left(\sum_{i=1}^{q} \lambda_i G_i(\omega) \right) \geq 0 \qquad \text{whenever } \forall i : \lambda_i \geq 0 \text{ and } \exists i : \lambda_i > 0,$$

where $G_i(\omega) = I_{A''_i}(\omega)(I_{A'_i}(\omega) - \alpha_i)$, $S_\lambda = \vee_{i:\lambda_i>0} A''_i$ and $I_A(\omega)$ is the indicator function defined in Expression (2): 1 if $\omega \models A$ and 0 otherwise.

Walley et al. offer the following algorithm to check coherence, where a sequence of linear programs with more than 2^n constraints each is generated. First, set $\mathcal{I} = \{1, \ldots, q\}$. Solve the linear program in Expression (7) below. If $\tau_i = 1$ for all $i \in \mathcal{I}$, then coherence fails (problem is Unsatisfiable). Otherwise, replace

$\mathcal{I}$ by $\{i \in \mathcal{I} : \tau_i = 1\}$. If $\mathcal{I}$ becomes empty, then coherence holds (problem is Satisfiable); otherwise solve linear problem in Expression (7) again, and so on.

$$\max \sum_{i\in\mathcal{I}} \tau_i \tag{7}$$
$$\text{s.t.: } \forall\omega : \sum_{i\in\mathcal{I}} \lambda_i G_i(\omega) + \sum_{i\in\mathcal{I}} \tau_i I_{A_i''}(\omega) \leq 0; \forall\lambda_i : \lambda_i \geq 0; \forall i \in \mathcal{I} : \tau_i \in [0,1].$$

Note that at each step we have the dual of a linear program with 2^n optimization variables; one can therefore write a compact integer linear program at each step, using the techniques described in previous sections.

6 Inference

In this section we offer a few brief comments on the *inference* problem: given a satisfiable or coherent set of assessment, find all possible values for $P(\varphi)$, the probability of an additional sentence φ, such that all assessments together are still satisfiable/coherent.

In the Kolmogorovian setup that is usually adopted for PSAT, both $\min P(\varphi)$ and $\max P(\varphi)$ can be obtained by adding appropriate linear objective functions to our methods. If additionally one wants tight bounds on a conditional probability $P(\varphi'|\varphi'')$, then linear fractional programming [5] can be used in the Kolmogorovian setup to transform $\min P(\varphi' \wedge \varphi'')/P(\varphi'')$ into a linear objective function (and similarly for $\max P(\varphi' \wedge \varphi'')/P(\varphi'')$).

The inference problem is considerably more complex in de Finetti's framework. Walley et al. [32, Algorithm 5] present solutions for such a situation that rely on sequences of linear programs. The discussion in Section 5 applies to that case. Here the verification of coherence is a preliminary step, because only coherent assessments are allowed to be used in inference [6].

7 Conclusion

In this paper we have introduced an approach to probabilistic satisfiability and coherence checking that translates these problems into integer linear programming. Our algorithms have the advantage of simplicity when compared to alternative approaches. Because we can rely on existing highly optimized linear programming solvers, we do not worry about numerical stability; likewise, our algorithms can inherit any gains from parallelization and heuristics applied to integer linear programming.

Experiments indicate that our algorithms are quite effective for random PSAT problems. Moreover, we have presented an analysis of phase transition in PSAT that improves previous results in the literature. Of course more testing is necessary to fully understand the properties of probabilistic satisfiability and coherence checking.

Acknowledgements

Both authors received support by CNPq. We thank the reviewers for very useful suggestions, in particular for pointing us to Ref. [3] and [4].

References

1. Ernest W. Adams. *A Primer of Probability Logic.* CSLI Publications, Stanford, CA, 2002.
2. Kim Allan Andersen and Daniele Pretolani. Easy cases of probabilistic satisfiability. *Annals of Mathematics and Artificial Intelligence*, 33(1):69–91, 2001.
3. Marco Baioletti, Andrea Capotorti, Sauro Tulipani, and Barbara Vantaggi. Simplification rules for the coherent probability assessment problem. *Annals of Mathematics and Artificial Intelligence*, 35(1-4):11-28, 2002.
4. Marco Baioletti, Andrea Capotorti, Sauro Tulipani. An empirical complexity study for a 2CPA solver. In Bouchon-Meunier, G. Coletti, R. R. Yager, editors, *Modern Information Processing: From Theory to Applications*, pages 73–84, 2005.
5. Dimitris Bertsimas and John N. Tsitsiklis. *Introduction to Linear Optimization.* Athena Scientific, Belmont, Massachusetts, 1997.
6. V. Biazzo and Angelo Gilio. A generalization of the fundamental theorem of de Finetti for imprecise conditional probability assessments. *International Journal of Approximate Reasoning* 24:251–272, 2000.
7. V. Biazzo, Angelo Gilio, Thomas Lukasiewicz, G. Sanfilippo. Probabilistic logic under coherence: Complexity and algorithms. *Annals of Mathematics and Artificial Intelligence*, 45(1-2):35–81, 2005.
8. George Boole. *The Laws of Thought.* Dover edition, 1958.
9. Andrea Capotorti, L. Galli, and Barbara Vantaggi. How to use locally strong coherence in an inferential process based on upper-lower probabilities. *Soft Computing*, 7(5):280–287, 2003.
10. Vijay Chandru and John Hooker. *Optimization Methods for Logical Inference.* John Wiley & Sons Inc., 1999.
11. Giulianella Coletti and Romano Scozzafava. *Probabilistic Logic in a Coherent Setting.* Trends in logic, 15. Kluwer, Dordrecht, 2002.
12. Bruno de Finetti. *Theory of Probability, vol. 1-2.* Wiley, New York, 1974.
13. Lester E. Dubins. Finitely additive conditional probability, conglomerability and disintegrations. *Annals of Statistics*, 3(1):89–99, 1975.
14. Ronald Fagin, Joseph Y. Halpern, and N. Megiddo. A logic for reasoning about probabilities. *Information and Computation*, 87:78–128, 1990.
15. Marcelo Finger and Glauber De Bona. Probabilistic satisfiability: Logic-based algorithms and phase transition. In *IJCAI*, pages 528–533, 2011.
16. A. M. Frisch and Peter Haddawy. Anytime deduction for probabilistic logic. *Artificial Intelligence*, 69:93–122, 1994.
17. I. P. Gent and T. Walsh. The SAT phase transition. In *European Conference on Artificial Intelligence*, pages 105–109, 1994.
18. G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *Journal of Complexity*, 4:1–11, 1988.
19. Theodore Hailperin. Best possible inequalities for the probability of a logical function of events. *American Mathematical Monthly*, 72:343–359, 1965.

20. Theodore Hailperin. *Boole's Logic and Probability: a Critical Exposition from the Standpoint of Contemporary Algebra, Logic, and Probability Theory*. North-Holland, Amsterdam, 1976.
21. Joseph Y. Halpern. *Reasoning about Uncertainty*. MIT Press, Cambridge, Massachusetts, 2003.
22. Peter J. Hammond. Elementary non-Archimedean representations of probability for decision theory and games. In P. Humphreys, editor, *Patrick Suppes: Scientific Philosopher; Volume 1*, pages 25–59. Kluwer, Dordrecht, The Netherlands, 1994.
23. Pierre Hansen and Brigitte Jaumard. Probabilistic Satisfiability. Technical Report G-96-31, Les Cahiers du GERAD, École Polytechique de Montréal, 1996.
24. Pierre Hansen and Sylvain Perron. Merging the local and global approaches to probabilistic satisfiability. *International Journal of Approximate Reasoning*, 47(2):125–140, 2008.
25. Brigitte Jaumard, Pierre Hansen, and Marcus Poggi de Aragão. Column generation methods for probabilistic logic. *ORSA Journal on Computing*, 3(2):135–148, Spring 1991.
26. Peter Krauss. Representation of conditional probability measures on Boolean algebras. *Acta Mathematica Academiae Scientiarum Hungaricae*, 19(3-4):229–241, 1968.
27. Thomas Lukasiewicz. Expressive probabilistic description logics. *Artificial Intelligence*, 172(6-7):852–883, April 2008.
28. V. McGee. Learning the impossible. In E. Bells and B. Skyrms, editors, *Probability and Conditionals*, pages 179–199. Cambridge University Press, 1994.
29. R. Myerson. *Game Theory: Analysis of Conflict*. Harvard University Press, Cambridge, MA, 1991.
30. Raymond Ng and V. S. Subrahmanian. Probabilistic logic programming. *Information and Computation*, 101(2):150–201, 1992.
31. N. J. Nilsson. Probabilistic logic. *Artificial Intelligence*, 28:71–87, 1986.
32. Peter Walley, Renato Pelessoni, and Paolo Vicig. Direct algorithms for checking consistency and making inferences from conditional probability assessments. *Journal of Statistical Planning and Inference*, 126(1):119–151, 2004.