International Journal of Approximate Reasoning 58 (2015) 57–70

Contents lists available at ScienceDirect

International Journal of Approximate Reasoning

www.elsevier.com/locate/ijar

Probabilistic satisfiability and coherence checking through integer programming

Fabio G. Cozman *, Lucas Fargoni di Ianni

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

ARTICLE INFO

Article history:
Received 13 May 2014
Received in revised form 1 September 2014
Accepted 11 September 2014
Available online 18 September 2014

Keywords:
Probabilistic logic
Probabilistic satisfiability
Coherence
Integer programming
Phase transitions

ABSTRACT

This paper presents algorithms, both for probabilistic satisfiability and for coherence checking, that rely on reduction to integer programming. That is, we verify whether probabilistic assessments can be satisfied by standard probability measures (Kolmogorovian setting) or by full conditional probabilities (de Finettian coherence setting), and in both cases verify satisfiability or coherence using integer programming techniques. We present an empirical evaluation of our method, the results of which show 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 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. [13, Chapter XVIII, Ex. 1]

Here we have propositions A and B, assessments $\mathbb{P}(A) = p$ and $\mathbb{P}(A \wedge B) = q$. Boole asks for $\mathbb{P}(B)$ and obtains the tight interval $[q, 1 - (p - q)]$. There is a probability measure that *satisfies* the assessments; for this reason, they are *coherent*.

Suppose we have atomic propositions $\{A_j\}_{j=1}^{n}$ and propositional sentences $\{\phi_i\}_{i=1}^{M}$ involving those atomic propositions. We may associate one or more of these sentences with probabilities, writing for instance $\mathbb{P}(\phi_i) = \alpha_i$. As detailed later, 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 [25,28,31,33,34,37]. When assessments involve conditional probabilities such as $\mathbb{P}(A|B) = \alpha$, there are two paths to follow. Kolmogorovian probability theory reduces such assessments to ratios of probabilities. The other path is to use de Finetti's theory of coherent probabilities, where full conditional probabilities are used to interpret conditional assessments [19,23,58]. The *Coherence Checking (CCHECK)* problem is to determine whether it is possible to find a full conditional probability that satisfies all assessments, without requiring that assessments are over an algebra or any other structure [5,6]. Coherence checking has been explored in a variety of settings; a typical example is:

* Corresponding author.

http://dx.doi.org/10.1016/j.ijar.2014.09.002
0888-613X/

A doctor considers three possible diagnoses, H_1 (ileum), H_2 (peritonitis), and H_3 (appendicitis with local peritonitis), with the logical condition that $H_3 \rightarrow (\neg H_1) \wedge H_2$, and assessments $\mathbb{P}(H_1) = 1/2$, $\mathbb{P}(H_2) = 1/5$, and $\mathbb{P}(H_3) = 1/8$. Note that diagnoses do not constitute a partition of the certain event. The assessments are *coherent* in that they are satisfied by at least a probability measure. The doctor now considers E (no pain in abdomen), notes that $H_3 \rightarrow \neg E$, and declares $\mathbb{P}(E|H_2) = 2/5$ and $\mathbb{P}(E|\neg H_2) = 1/8$. Now the whole set of assessments fails to be coherent.[1]

Probabilistic satisfiability and coherence checking are central problems in reasoning under uncertainty. They serve not only as a foundation for logical and probabilistic inference, but as a basis for probabilistic rules [50], and as an initial necessary step in the understanding of combinations of first-order logic and probabilities [35,46,51].

The most direct way to solve a PSAT problem is to write it down as a linear feasibility problem [33]. The difficulty is that the resulting linear program may be too large; for n propositions, we must build a matrix with up to 2^n columns. When conditional probabilities are present, coherence checking may require sequences of such linear programs. To avoid dealing with exponentially many columns, one may resort to column generation techniques [40], to inference rules that capture probabilistic relationships [6], or even to combinations of column generation and inference rules [38]. There is also a different approach to probabilistic satisfiability that reduces it to logical satisfiability [4,26]. Overall, results in the literature save computations by applying increasingly complicated methods.

In this paper we present another approach to probabilistic satisfiability and coherence checking, where these problems are turned into integer programs. Our basic algorithm for PSAT is rather concise and easy to implement when a linear solver is available. In fact, our goal is to present methods that can be applied to medium sized problems, with say some 20 to 200 atomic propositions, by exploiting the fact that integer programming technology has improved dramatically in recent years. So, instead of explicitly resorting to inference rules and column generation, our methods simply outsource such schemes to the linear solver, as top solvers do apply sophisticated heuristics and numerical stabilization internally. We show that our techniques can be easily extended to expectation assessments, and describe ways to reduce coherence checking to (sequences of) integer linear programs.

As our experiments show, integer programming techniques are not yet capable of beating the fastest methods in the literature for large problems, but they offer a robust basis for PSAT and CCHECK. Using our implementation we study the issue of phase transition in probabilistic satisfiability, showing evidence of interesting phenomena in PSAT.

Section 2 summarizes necessary background in satisfiability and probability satisfiability. Our basic algorithm for probabilistic satisfiability is described in Section 3. We consider extensions of probabilistic satisfiability in Section 4, and then study coherence checking in Section 5. Implementation and experiments, with a discussion of phase transitions, are presented in Section 6.

2. SAT and PSAT

Consider n atomic propositions A_j and M sentences ϕ_i in propositional logic involving those atomic propositions. A truth assignment is an assignment of truth values (True or False) to each atomic proposition, that induces an assignment of truth values for all sentences involving the atomic propositions. 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 atomic propositions such that all sentences evaluate to True [18,30].

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 (note that literals may appear more than once in a clause, so in fact we can have up to k distinct 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 m clauses, as follows. For each one of the m clauses: select k propositions at random, and for each proposition 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, 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 [30]. 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 $\mathbb{P}(\phi_i) \bowtie \alpha_i$, 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 $\mathbb{P}$ over this set satisfies the assessments if, for each assessment $\mathbb{P}(\phi_i) \bowtie \alpha_i$,

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

[1] The example is due to Coletti and Scozzafava [21, Example 1], and appears edited here. Failure of coherence can be verified as follows. We must have $\mathbb{P}(H_2) = 1/5 = \mathbb{P}(H_2 \wedge E) + \mathbb{P}(H_2 \wedge \neg E) \geq \mathbb{P}(H_2 \wedge E) + \mathbb{P}(H_2 \wedge \neg E \wedge (\neg H_1 \wedge H_3))$; but $\mathbb{P}(H_2 \wedge E) = 2/25$ (multiplying $\mathbb{P}(H_2) = 1/5$ and $\mathbb{P}(E|H_2) = 2/5$) and $\mathbb{P}(H_2 \wedge \neg E \wedge (\neg H_1 \wedge H_3)) = \mathbb{P}(H_3) = 1/8$ (because $H_2 \wedge \neg E \wedge (\neg H_1 \wedge H_3)$ is equivalent to H_3 as $H_3 \rightarrow (\neg H_1) \wedge H_2$ and $H_3 \rightarrow \neg E$).

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 the possible truth assignments, 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 [37]. 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 sentences are associated with assignments $\mathbb{P}(\phi_i) \bowtie \alpha_i$. If a sentence ϕ_i is not associated with a probabilistic assessment, just add $\mathbb{P}(\phi_i) = 1$ to the assessments. Then each constraint can be written as

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

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 $\sum_j \mathbb{P}(\omega_j) = 1$ and $\mathbb{P}(\omega_j) \geq 0$ for all ω_j. 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 [38]. 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 [31]. Hence we can write down an $(M + 1) \times (M + 1)$ matrix $\mathbf{C}$ and write the PSAT problem as feasibility of $\mathbf{Cp} \bowtie \boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ denotes a vector of values α_i and $\bowtie$ refers to $\geq$, $=$ or $\leq$ as appropriate. Each row of $\mathbf{C}$ corresponds to an assessment, except one row that corresponds to the constraint $\sum_j \mathbb{P}(\omega_j) = 1$. Each column of $\mathbf{C}$ corresponds to a truth assignment; the challenge is to select $(M + 1)$ truth assignments. This is usually done by column generation techniques from linear programming [7]. 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) [37,38]. Performance improvements are obtained if column generation is preceded by application of inference rules.[2] This combination has produced the best results so far, being able to solve PSAT problems with hundreds of atomic propositions and hundreds of assessments. Variations on such techniques have been applied successfully even to probabilistic description logics, in problems with hundreds of propositions and clauses [41]. A different approach to PSAT is to reduce it to SAT [4,26]. For instance, the method by Finger and De Bona [26] reduces the selection of columns of $\mathbf{C}$ to a SAT 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).

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 $\mathbb{P}(A_i) = \alpha_i$) [26]. Even though this form may seem restrictive, every PSAT can be brought to it with polynomial effort: basically, for each assessment $\mathbb{P}(\phi_i) \bowtie \alpha_i$, introduce if necessary fresh atomic propositions to transform the assessment into $\mathbb{P}(\phi_i') = \alpha_i$; then introduce a new proposition A_i' and exchange the original assessment by a sentence $A_i' \Leftrightarrow \phi_i'$ and an assessment $\mathbb{P}(A_i') = \alpha_i$; finally, generate a single sentence ϕ that is a conjunction of all previous sentences. Such a sentence ϕ can be turned into CNF in polynomial time, for instance using Tseitsin's transformation [18]. Thus every k-PSAT for $k > 2$ can be reduced to Normal Form with q assessments $\mathbb{P}(A_i) = \alpha_i$ plus one sentence ϕ in CNF where each clause has exactly 3 literals.

3. Probabilistic satisfiability and integer programming techniques

Assume our PSAT problem is in Normal Form with assessments $\{\mathbb{P}(A_j) = \alpha_j\}_{j=1}^q$ 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 $\mathbf{Cp} = \boldsymbol{\alpha}$.

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, the method by Finger and De Bona [26] reduces PSAT to SAT, and in doing so it encodes the search for optimization variables as a SAT problem. 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 [18].

[2] An example of an inference rule [38]: if $\mathbb{P}(A_1) \in [\underline{\alpha}_1, \overline{\alpha}_1]$ and $\mathbb{P}(\neg A_1 \vee A_2) \in [\underline{\alpha}_1, \overline{\alpha}_2]$ for $\overline{\alpha}_1 + \overline{\alpha}_2 \geq 1$, then $\mathbb{P}(A_2) \in [\max(0, \underline{\alpha}_1 + \underline{\alpha}_2 - 1), \min(1, \overline{\alpha}_2)]$. There are many inference rules in the literature, but not always generating sound inferences [27].

Algorithm 1 PSAT solution based on integer linear program.

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

Start by generating a vector $\mathbf{a}_j$ with n binary variables $\{a_{i,j}\}_{i=1}^n$, all with values 0 and 1. Now take one clause of ϕ; suppose it is written as

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

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 $\mathbf{Cp} = \boldsymbol{\alpha}$. 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 $\mathbf{p}$. We must handle the bilinear term $a_{i,j}p_j$; we do that by introducing a fresh new variable $b_{i,j}$ and the linear constraints:

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

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 Algorithm 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. As noted before, we have $(q+1)^2$ optimization variables that encode $\mathbf{C}$. However, we have introduced $n(q+1)$ binary optimization variables $a_{i,j}$, and these integer variables do demand effort by the linear solver. We have $m(q+1)$ linear constraints that encode clauses, plus $2q(q+1)$ linear constraints that encode products $a_{i,j}p_j$.

This integer program is rather orderly built: each value of j corresponds to a SAT problem with identical clauses; in fact each j corresponds to a column that would be produced by a column generation method. Hence we have a block of constraints for each j, and such regularity must be exploited by the linear solver. If in addition the constraints are specified in some organized fashion, say because they are expressed through graphs or other regular language [29], then there will be additional structure to be exploited by the solver.

We present tests with the PSAT-IP algorithm in Section 6; before we examine those results, we discuss more general assessments and conditional probabilities, both in Kolmogorovian and de Finettian settings, in Sections 4 and 5.

4. Variations on a theme

An obvious change to PSAT-IP is to allow assessments such as $\mathbb{P}(A_i) \bowtie \alpha_i$. This introduces no difficulty: simply change the corresponding constraint in Line 5 of the algorithm either to $\geq$ or $\leq$ as appropriate.

Another possible change to the input of PSAT-IP is to depart a little from the Normal Form, allowing for instance assessments of the form $\mathbb{P}(\phi_i) \bowtie \alpha_i$ where ϕ_i is a propositional sentence. Of course we can simply return to Normal Form by introducing a fresh proposition A and turning the sentence $A \Leftrightarrow \phi_i$ into CNF, perhaps using Tseitsin's transformation where clauses are generated with the addition of fresh atomic propositions [18]. However, it is interesting to deal directly with the

Algorithm 2 ESAT solution based on integer linear program.

1: **procedure** ESAT-IP(propositions $\{A_j\}_{j=1}^n$, assessments $\{\sum_{k=1}^{r_i} \beta_{i,k}\mathbb{P}(A_{i(k)}) = \alpha_i\}_{i=1}^q$, sentence ϕ in CNF with m clauses)
COMMENT: $a_{i,j}$ are binary; $b_{i,j}$, $b'_{i,j}$, p_j are real-valued in $[0,1]$.
2: **for** $j \in \{1,\ldots,q+1\}$ and each clause $(\bigvee_{l'=1}^{k'} A_{i_{l'}}) \vee (\bigvee_{l''=1}^{k''} \neg A_{i_{l''}})$ of ϕ **do**
3: Generate constraint $(\sum_{l'=1}^{k'} a_{i_{l'},j}) + (\sum_{l''=1}^{k''}(1 - a_{i_{l''},j})) \geq 1$.
4: **for** $i \in \{1,\ldots,q\}$ **do**
5: Generate constraint $\sum_{j=1}^{q+1} b_{i,j} = \alpha_i$.
6: **for** $j \in \{1,\ldots,q+1\}$ **do**
7: Generate constraint $b_{i,j} = \sum_k \beta_{i,k} b'_{i(k),j}$
8: **for** each $b'_{i(k),j}$ that is used in the previous line, **do**
9: generate constraints $0 \leq b'_{i(k),j} \leq a_{i(k),j}$, $a_{i(k),j} - 1 + p_j \leq b'_{i(k),j} \leq p_j$.
10: Generate constraint $\sum_{j=1}^{q+1} p_j = 1$.
11: **return** Satisfiable if constraints have a solution, Unsatisfiable otherwise.

sentence ϕ without introducing atomic propositions, as each atomic proposition will lead to several integer variables within the PSAT-IP algorithm. We will use as example the case where ϕ_i consists of a single clause, as this case has been used in the past to test PSAT algorithms [38,40].

Suppose then that we have, for some i, the assessment

$$\mathbb{P}\left(\left(\bigvee_{l'=1}^{k'} A_{i_{l'}}\right) \vee \left(\bigvee_{l''=1}^{k''} \neg A_{i_{l''}}\right)\right) = \alpha_i. \tag{6}$$

We follow Algorithm 1, with a change in Line 7. We must now generate the following linear constraints:

$$a_{i_{l'},j} - 1 + p_j \leq b_{i,j} \leq p_j, \quad \text{for } l' \in \{1,\ldots,k'\}; \tag{7}$$

$$(1 - a_{i_{l''},j}) - 1 + p_j \leq b_{i,j} \leq p_j, \quad \text{for } l'' \in \{1,\ldots,k''\}; \tag{8}$$

$$0 \leq b_{i,j} \leq \left(\sum_{l'=1}^{k'} a_{i_{l'},j}\right) + \left(\sum_{l''=1}^{k''}(1 - a_{i_{l''},j})\right). \tag{9}$$

And again, the set of linear constraints has a solution if and only if the PSAT problem if satisfiable.

Suppose, to extend the example further, that ϕ_i is a conjunction $\phi'_i \wedge \phi''_i$ where ϕ'_i and ϕ''_i are clauses. Then we must introduce two variables $b'_{i,j}$ and $b''_{i,j}$ and add the constraints $b'_{i,j} + b''_{i,j} \leq b_{i,j} + 1$, $b'_{i,j} \geq b_{i,j}$ and $b''_{i,j} \geq b_{i,j}$. These constraints force $b_{i,j}$ to behave as a conjunction. We now recurse by adding constraints on $b'_{i,j}$ and $b''_{i,j}$ as required by ϕ'_i and ϕ''_i respectively. This sort of recursive procedure can be applied to any propositional sentence ϕ_i.

Another extension that does not introduce conceptual challenges, even though it may introduce some computational burden, is to allow expectations in assessments. That is, instead of just considering assessments such as $\mathbb{P}(\phi_i) \bowtie \alpha_i$, we also take assessments such as $\mathbb{E}[f_i] \bowtie \alpha_i$, where f_i is a real-valued function of truth assignments, and where $\mathbb{E}[f_i] = \sum_\omega f_i(\omega)\mathbb{P}(\omega)$. We denote by ESAT (Expectation Satisfiability) the problem of determining whether a set of expectation assessments can be satisfied by at least one probability measure over truth assignments.

Expectation assessments are quite powerful as they can uniquely characterize any closed convex set of probability measures [48,57,61]. Such sets, often referred to as *credal sets* [45], offer a flexible and general language for uncertainty representation, encompassing many other formalisms such as belief functions, random sets, Choquet capacities, possibility measures, and p-boxes [3].

Without loss of generality, we assume that any function f_i is given as a linear combination of indicator functions; that is,

$$f_i(\omega) = \sum_{k=1}^{r_i} \beta_{i,k}\mathbb{I}_{A_{i(k)}}(\omega);$$

we abuse notation by using $i(k)$ to refer to the function that returns the index of an atomic proposition. Hence $\mathbb{E}[f_i] = \sum_k \beta_{i,k}\mathbb{P}(A_{i(k)})$.

An algorithm for ESAT, based on integer linear programming, is presented in Algorithm 2. Note that an assessment such as $\mathbb{E}[f_i] \geq \alpha_i$ can be dealt with easily; the only change is to use $\geq$ instead of $=$ in the corresponding constraint in Line 5 of the algorithm. Again, there are $n(q+1)$ binary optimization variables, plus a set of $m(q+1)$ linear constraints that encode clauses, and a polynomially large set of linear constraints that encode products $a_{i,j}p_j$ (the number of constraints depends on the number of terms in each function f_i).

Yet another variation on PSAT is obtained when we contemplate assessments of conditional probability within a Kolmogorovian framework. That is, suppose that a conditional assessment $\mathbb{P}(\phi_i'|\phi_i'') = \alpha_i$ is given. In the Kolmogorovian setting, this assessment means that

$$\frac{\mathbb{P}(\phi_i' \wedge \phi_i'')}{\mathbb{P}(\phi_i'')} = \alpha_i \quad \text{whenever } \mathbb{P}(\phi_i'') > 0.$$

This holds if and only if

$$\mathbb{P}(\phi_i' \wedge \phi_i'') - \alpha_i \mathbb{P}(\phi_i'') = 0;$$

that is, if and only if

$$\mathbb{E}[\mathbb{I}_{\phi_i' \wedge \phi_i''} - \alpha_i \mathbb{I}_{\phi_i''}] = 0. \tag{10}$$

The only change from "unconditional" PSAT is that each element of the matrix $\mathbf{C}$ is now a linear expression (an expectation). Thus we can apply the ESAT-IP algorithm to this problem. There are few changes if instead of an assessment $\mathbb{P}(\phi'|\phi'') = \alpha_i$, we have $\mathbb{E}[f_i|\phi''] = \alpha_i$ for $f_i = \sum_{k=1}^{r_i} \beta_{i,k} \mathbb{I}_{A_{i(k)}}$. We simply have to transform the latter assessment into

$$\mathbb{E}[f_i \mathbb{I}_{\phi''} - \alpha_i \mathbb{I}_{\phi''}] = 0;$$

that is,

$$\mathbb{E}\left[\left(\sum_{k=1}^{r_i} \beta_{i,k} \mathbb{I}_{A_{i(k)} \wedge \phi''}\right) - \alpha_i \mathbb{I}_{\phi''}\right] = 0. \tag{11}$$

Again, ESAT-IP can be used directly.

Matters are considerably simpler for a conditional assessment that only involves atomic propositions; that is, $\mathbb{P}(A_i'|A_i'') = \alpha_i$. 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' and A_i'' respectively. To handle this, the only change that must be made to the PSAT-IP algorithm is that the constraints in its Line 5 must be replaced by $\sum_{j=1}^{q+1}(b_{i,j}' - \alpha_i b_{i,j}'') = 0$, and constraints in Line 7 must be replaced by

$$0 \le b_{i,j}' \le a_{i,j}', \qquad 0 \le b_{i,j}' \le a_{i,j}'', \qquad a_{i,j}' + a_{i,j}'' - 2 + p_j \le b_{i,j}' \le p_j, \tag{12}$$

$$0 \le b_{i,j}'' \le a_{i,j}'', \qquad a_{i,j}'' - 1 + p_j \le b_{i,j}'' \le p_j. \tag{13}$$

To summarize the previous comments:

1. The most general algorithm is ESAT-IP. To handle constraints such as $\mathbb{E}[f_i] \bowtie \alpha_i$, just replace $=$ by $\bowtie$ in Line 5. To handle conditional probabilities and conditional expectations, use Expressions (10) and (11) respectively.
2. As stated, ESAT-IP does not collapse to PSAT-IP when all assessments are of the form $\mathbb{P}(A_i) = \alpha_i$; however, the change that must be made to guarantee this collapse is trivial. It is enough to change Lines 7 and 8 of ESAT-IP, so that an assessment $\mathbb{P}(A_i) = \alpha_i$ does not generate any $b_{i(k),j}'$; instead, $b_{i,j}$ is then directly used in Line 9.
3. In addition, when an additional check in Line 7 detects that an assessment is of the form $\mathbb{P}(A_i'|A_i'') = \alpha_i$, then ESAT-IP should not generate any $b_{i(k),j}'$; instead $b_{i,j}'$ and $b_{i,j}''$ as in Expressions (12) and (13) should be directly used in Line 9.

In our presentation we have decided not to include the checks for assessments $\mathbb{P}(A_i) = \alpha_i$ and $\mathbb{P}(A_i'|A_i'') = \alpha_i$ in ESAT-IP, to make it more readable.

To finish this section, we note that both PSAT-IP and ESAT-IP can be used to compute the maximum (or minimum) value of a probability, subject to all other assessments. Suppose that we have a set of assessments and a sentence in CNF as before, and we wish to know the maximum possible value of $\mathbb{P}(A_{i^*})$, where i^* is a fixed index in $\{1, \ldots, n\}$. To do so, we adapt existing techniques that have been applied to PSAT [6,40].

We start by writing $\mathbb{P}(A_{i^*})$ as the summation $\sum_{j=1}^{q+1} b_{i^*,j}$, where the $b_{i^*,j}$ are new optimization variables (real-valued in $[0, 1]$). We must then equate each $b_{i^*,j}$ with the product $a_{i^*,j} p_j$; we do so by introducing the constraints

$$0 \le b_{i^*,j} \le a_{i^*,j} \quad \text{and} \quad a_{i^*,j} - 1 + p_j \le b_{i^*,j} \le p_j. \tag{14}$$

Finally, we compute $\max \sum_{j=1}^{q+1} b_{i^*,j}$ subject to constraints (14) for each $j \in \{1, \ldots, q+1\}$, and to all constraints generated by PSAT-IP (or ESAT-IP). The maximum is exactly $\max \mathbb{P}(A_{i^*})$, as desired.

5. Coherence checking and integer programming techniques

In this section we consider assessments of conditional probability from a different light; that is, here we interpret such assessments within de Finetti's theory of coherence. Both PSAT and Coherence Checking (CCHECK) are identical when

all probabilities are unconditional. There are however significant conceptual differences when conditional assessments are present, particularly when some conditioning propositions have probability zero.

Most existing algorithms for CCHECK build a sequence of linear programs whose solution yields the desired coherence check. In particular, a CCHECK problem can be reduced to a sequence of PSAT problems [8,10]. Indeed, our later Algorithm 4 is nothing but Biazzo et al.'s [8,10] solution for the CCHECK problem, and if the reader is familiar with that method, the text up to Algorithm 4 can be skipped. We have included this text because, first, our presentation emphasizes connections to linear programming and duality; second, and more importantly, this particular path allows us to find alternative algorithms that resort to disjunctive programming.

We first review some basic and known results concerning coherence checking (Section 5.1). We then examine the reduction of CCHECK to disjunctive programming and to sequences of PSAT problems (Section 5.2).

5.1. Coherence and full conditional probabilities

Consider a set of assessments $\{\mathbb{P}(A_i) = \alpha_i\}_{i=1}^q$ and a sentence ϕ that must hold. That is, every truth assignment ω we consider must be such that $\omega \models \phi$. In coherence checking one often ignores ϕ for the sake of simplicity, leaving it implicit that truth assignments must satisfy this background set of logical constraints. We will leave ϕ implicit in this section.

The assessments are *coherent* if, for any set of real numbers λ_i, there is always a truth assignment ω such that [19,23]:

$$\sum_{i=1}^{q} \lambda_i \big(\mathbb{I}_{A_i}(\omega) - \alpha_i\big) \geq 0. \tag{15}$$

Obviously, if coherence fails, there is a set of λ_i such that for all ω, we have $\sum_{i=1}^q \lambda_i(\mathbb{I}_{A_i}(\omega) - \alpha_i) < 0$. That is, there is a linear combination of accepted assessments that leads to a sure loss.

A slightly more general version of coherence, variously called *g-coherence* [8] or *sure-loss-avoidance* [57], assumes that assessments are of the form $\mathbb{P}(A_i) \geq \alpha_i$. Such assessments are quite general; for instance we can represent the assessment $\mathbb{P}(A_i) \leq \beta_i$ through $A_i' \Leftrightarrow \neg A_i$ and $\mathbb{P}(A_i') \geq 1 - \beta_i$. Assessments are *g-coherent* if, for any set of *non-negative* real numbers λ_i, there is always a truth assignment ω such that Expression (15) obtains.

That is, g-coherence obtains if and only if the maximum value of μ is zero in the following program (note: if g-coherence fails, the maximum is infinite):

$$\max \mu \quad \text{s.t.} \quad \forall i : \lambda_i \geq 0, \quad \forall j : \sum_i \lambda_i \big(\mathbb{I}_{A_i}(\omega_j) - \alpha_i\big) + \mu \leq 0.$$

The dual of this program gives us a PSAT problem [15]:

$$\min 0 \quad \text{s.t.} \quad \forall i : \sum_j \big(\mathbb{I}_{A_i}(\omega_j) - \alpha_i\big) p_j \geq 0; \quad \forall j : p_j \geq 0; \quad \sum_j p_j = 1.$$

Now consider conditional assessments $\{\mathbb{P}(A_i|B_i) \geq \alpha_i\}_{i=1}^q$. Note that to simplify matters, we only consider assessments over a pair of propositions A_i and B_i; we leave to the background constraint ϕ any logical relation between these propositions. Note also that if a particular B_i is logically equivalent to a tautology, the corresponding assessment is unconditional. There are two ways to specify a suitable concept of g-coherence.

First, take g-coherence to mean that the set of constraints

$$\forall j : \sum_i \lambda_i \, \mathbb{I}_{B_i}(\omega_j)\big(\big(\mathbb{I}_{A_i}(\omega_j) - \alpha_i\big) + \epsilon\big) \leq 0 \tag{16}$$

has no solution for $\epsilon > 0$ and $\lambda_i \geq 0$ such that $\sum_i \lambda_i > 0$. Such a concept of coherence was first proposed by Williams [61,62], and later studied in detail by Walley [57]. The advantages of this formulation of g-coherence are explored by Wagner [56]. The motivation here is simple: if such a condition fails, so that the constraints have a solution, one can produce a sure loss from a linear combination of acceptable assessments (even more acceptable as we are throwing in a positive amount ϵ).

Now if g-coherence holds, for any set $\{\lambda_i\}_{i=1}^q$ of non-negative numbers, there is always $\omega \models \bigvee_{i:\lambda_i>0} B_i$ such that

$$\sum_{i=1}^{q} \lambda_i \mathbb{G}_i(\omega) \geq 0, \tag{17}$$

where $\mathbb{G}_i(\omega) = \mathbb{I}_{B_i}(\omega)(\mathbb{I}_{A_i}(\omega) - \alpha_i)$. Conversely, if assessments are such that there is no $\omega \models \bigvee_{i:\lambda_i>0} B_i$ satisfying Expression (17), then for every $\omega \models \bigvee_{i:\lambda_i>0} B_i$ we have $\sum_i \lambda_i \mathbb{G}_i(\omega) < 0$, and for some $\epsilon > 0$ we obtain the inequality in the definition of g-coherence.

Hence g-coherence can be defined as: for any set $\{\lambda_i\}_{i=1}^q$ of non-negative numbers, there is always $\omega \models \bigvee_{i:\lambda_i>0} B_i$ such that $\sum_{i=1}^q \lambda_i \mathbb{G}_i(\omega) \geq 0$. This concept is directly based on suggestions by de Finetti [23,43] that were later expanded [39,44,54] and then adapted to assessments based on inequalities [8,20].

One can instead consider assessments of conditional expectations such as $\{\mathbb{E}[f_i|B_i] \geq \alpha_i\}_{i=1}^q$, where f_i is a real-valued function of truth assignments. In fact this is the strategy adopted by Williams [61,62] and by Walley [57]. To simplify the presentation, we keep the focus on probability assessments, but computations do not change substantially if we move to expectations on functions $f_i = \sum_k \beta_{i,k} \mathbb{I}_{A_{i(k)}}$.

The non-trivial result about g-coherence is that a set of assessments is g-coherent if and only if there exists a Boolean algebra $\mathcal{B}$ such that the assessments can be *extended* to a full conditional probability on $\mathcal{B}$.[3] To explain this, recall that a *full conditional probability* $\mathbb{P} : \mathcal{B} \times (\mathcal{B} \backslash \emptyset) \to \Re$, where $\mathcal{B}$ is a Boolean algebra of events (with $\emptyset$ denoting the bottom element) over a set Ω, is a two-place set-function such that for every event $C \neq \emptyset$ [24]:

- $\mathbb{P}(C|C) = 1$ and $\mathbb{P}(A|C) \geq 0$ for all A;
- $\mathbb{P}(A \vee B|C) = \mathbb{P}(A|C) + \mathbb{P}(B|C)$ when $A \wedge B = \emptyset$;
- $\mathbb{P}(A \wedge B|C) = \mathbb{P}(A|B \wedge C)\mathbb{P}(B|C)$ when $B \wedge C \neq \emptyset$.

There are other names for full conditional probabilities in the literature, such as *conditional probability measures* [42]; and *complete conditional probability systems* [49]. Full conditional probabilities have been applied in economics [36,49], philosophy [1,47], and artificial intelligence [16].

Full conditional probabilities behave differently from (Kolmogorovian-style) probability measures with respect to conditioning: a set of assessments may be g-coherent even if $\mathbb{P}(B) = 0$ and $\mathbb{P}(A|B) = \alpha > 0$.

5.2. Checking coherence

In this section we discuss algorithms that can be used to check coherence of a set of conditional probability assessments. As will be clear, there is considerable literature on this topic, most of which starts from the second definition of g-coherence (Expression (17)) to generate a sequence of dual linear programs that are in essence PSAT problems. An exception is the work of Walley, Pelessoni and Vicig [58], where the first definition (Expression (16)) is directly used to derive algorithms. Here we start from the first definition and later move towards the second.

One naive approach might be to fix a very small $\epsilon > 0$ and check whether constraints in Expression (16) are consistent. This may lead to numerical instability problems when ϵ is very small. Another naive approach is to check feasibility symbolically by leaving ϵ as a parameter in an implementation of the simplex method. This may work for very small problems, but it will not scale easily as it requires manipulating polynomials in ϵ.

To move to more numerically robust methods, we note that g-coherence fails if and only if the following (disjunctive) linear constraints are feasible:

$$\forall i : \lambda_i \geq 0, \quad \sum_i \lambda_i \geq 1, \quad \forall j : \sum_i \lambda_i \mathbb{G}_i(\omega_j) + \sum_i \tau_i \mathbb{I}_{B_i}(\omega_j) \leq 0, \tag{18}$$

$$\forall i : \tau_i \in \{0, 1\}, \quad \forall i : \big((\lambda_i = 0) \vee (\tau_i = 1)\big). \tag{19}$$

To prove that failure of g-coherence is equivalent to feasibility of Expressions (18) and (19), reason as follows.[4] First, suppose there are $\{\lambda_i\}_{i=1}^q$ satisfying Expressions (18) and (19). Then the assessments fail to be g-coherent: we have for all ω, $\sum_{i:\lambda_i>0} \lambda_i(\mathbb{G}_i(\omega) + (1/\lambda_i)\mathbb{I}_{B_i}(\omega)) \leq 0$ and then Expression (16) has a solution with $\epsilon = \min(1/\lambda_i : \lambda_i > 0)$. Conversely, suppose the assessments fail to be g-coherent; that is, we have $\sum_i \lambda_i \mathbb{G}_i(\omega) + \epsilon \sum_i \mathbb{I}_{B_i}(\omega) \leq 0$ for all ω, for some $\lambda_i \geq 0$ such that $\sum_i \lambda_i > 0$, and some $\epsilon > 0$. Note that we can assume that $\epsilon < 1$; if there is a solution with $\epsilon \geq 1$, there is always a solution with smaller ϵ. Now if $\lambda_i > 0$, impose $\tau_i = 1$; if $\lambda_i = 0$, impose $\tau_i = 0$; then $\sum_i \lambda_i' \mathbb{G}_i(\omega) + \sum_i \tau_i \mathbb{I}_{B_i}(\omega) \leq 0$ for all ω, where $\lambda_i' = \lambda_i / (\epsilon \min(\lambda_i : \lambda_i > 0))$. Note that as we can take $\epsilon < 1$, this guarantees that $\sum_i \lambda_i' \geq 1$. Hence Expressions (18) and (19) are satisfied.

The feasibility problem encoded by Expressions (18) and (19) can be solved with currently available linear solvers as their solution uses techniques from integer linear programming [59,60] (an example is the CPLEX package mentioned later).

The solutions presented so far in this section consider up to 2^n constraints in linear programs, hence they can only be used with relatively small problems. We wish to move towards compact programs as built by the PSAT-IP algorithm; to do so, we follow the approach by Walley et al. [58]. They build a procedure that iteratively enforces the constraints $(\tau_i = 0) \to (\lambda_i = 0)$, by solving the program

$$\max \sum_i \tau_i \quad \text{s.t.} \quad \forall i : \lambda_i \geq 0, \quad \forall i : \tau_i \in [0, 1],$$
$$\forall j : \sum_i \lambda_i \mathbb{G}_i(\omega_j) + \sum_i \tau_i \mathbb{I}_{B_i}(\omega_j) \leq 0. \tag{20}$$

[3] This result can be found in the work of Williams [61,62] or Walley [57]; it can also be constructed from a combination of results by Coletti [20, Theorem 2.3] or Biazzo and Gilio [8] and by Holzer [39, Theorem 4.4]; for the propositional language we deal with, the latter results are basically settled already by Lehman [44, Theorem 4]. The matter is also discussed by Coletti and Scozzafava [19, Chapter 11].

[4] This reasoning uses an argument that is due to Walley, Pelessoni and Vicig in the discussion of their Algorithm 2 [58].

Algorithm 3 Coherence checking by Walley et al. [58, Algorithm 2].

1. Start with assessments $\{\mathbb{P}(A_i|B_i) \geq \alpha_i\}_{i=1}^{q}$, and solve program (20).
2. If $\tau_i = 1$ for all assessments, then g-coherence fails. Otherwise, discard the assessments $\mathbb{P}(A_i|B_i) \geq \alpha_i$ such that $\tau_i = 0$. If the remaining set of assessments is empty, then g-coherence holds; otherwise, renumber the remaining assessments, and return to the first step.

Algorithm 4 Coherence checking by sequence of PSAT problems.

1. Given assessments $\{\mathbb{P}(A_i|B_i) \geq \alpha_i\}_{i=1}^{q}$, verify feasibility of

$$\forall j : p_j \geq 0, \quad \sum_j p_j = 1, \quad \forall i : \sum_j \mathbb{I}_{B_i}(\omega_j)\big(\mathbb{I}_{A_i}(\omega_j) - \alpha_i\big) \geq 0,$$

   where j is restricted to values such that $\omega_j \models \bigvee_i B_i$. If this conditional PSAT problem is unsatisfiable, assessments fail to be g-coherent (announce it and stop).
2. For each B_i, determine whether there is $\{p_j\}_j$ such that $\sum_j \mathbb{I}_{B_i}(\omega_j)p_j$ is strictly larger than zero. Discard each such B_i. If all assessments are discarded, then original assessments are g-coherent (announce it and stop).
3. Collect the remaining assessments, renumber them, and return to the first step.

Their algorithm is presented in Algorithm 3. As shown by Walley et al. [58], Algorithm 3 works because if there is any $\tau_i > 0$ that satisfies the constraints of program (20), then the optimal solution of this program has $\tau_i = 1$. Intuitively, this is true because we can always increase the values of λ_i, so if it is possible to have $\tau_i > 0$, it is possible to have $\tau_i = 1$.

Some additional insight can be gained by examining the dual of program (20):

$$\min \sum_i q_i \quad \text{s.t.} \quad \forall j : p_j \geq 0, \quad \forall i : q_i \geq 0,$$
$$\forall i : \sum_j \mathbb{G}_i(\omega_j)p_j \geq 0, \quad \forall i : \sum_j \mathbb{I}_{B_i}(\omega_j)p_j \geq 1 - q_i. \tag{21}$$

By the complementary slackness property [14, Section 4.5], we have $\tau_i < 1$ implies $q_i = 0$. But note that if $\tau_i < 1$, then $\tau_i = 0$ [58]. Also, if $q_i \geq 1$, then it must be that $q_i = 1$; and because $\sum_i \tau_i = \sum_i q_i$ by strong duality, we obtain that $\tau_i = 1$ if and only if $q_i = 1$. Hence we can interpret the algorithm by Walley et al. [58] as a method that verifies whether assessments satisfy a PSAT problem *and* whether $\mathbb{P}(B_i)$ can be positive for each B_i; if $\mathbb{P}(B_i) > 0$ for some B_i, then this B_i is discarded, and the analysis iterates only with the remaining B_i.

Hence we have reached, by a different route, an existing algorithm for coherence checking [8,10] that is presented in Algorithm 4. The second step of Algorithm 4 asks for an "inference" where the maximum of $\sum_j \mathbb{I}_{B_i}(\omega_j)p_j$ is computed subject to the constraints in the PSAT problem described in the first step (this can be computed using the method at the end of Section 4). If this maximum is strictly larger than zero, B_i is discarded. So, the whole coherence checking procedure is a sequence of PSAT-like problems, and each one of them can be dealt with by the PSAT-IP algorithm. To save computation, after each PSAT problem is solved within an iteration of the algorithm, we should discard every B_i that is associated with a positive p_j (that is, $p_j > 0$ and $\omega_j \in B_i$). Obviously, in practice one need not build a series of linear programs from scratch, as most linear solvers have facilities to change objective functions and even to add/remove constraints.

Algorithm 4 appeared in the work of Biazzo and Gilio [8], as an upgrade on previous algorithms for de Finettian coherence checking [32]. Later the relationship with PSAT was explored, together with important results on complexity [12] and on connections with default reasoning [11]. Variants on this algorithm emphasize other calculations [20,55]. For instance, it is possible to exploit the fact that several values of p_j can be assigned zero at any given iteration so as to simplify computations locally [16,17]. Local rules of inference that help in computations have been explored intensely for coherence checking [4,6,9]. Finally, it should be noted that coherence checking algorithms, in particular Walley et al.'s algorithms [58], have been extended to structures that generalize full conditional probabilities [22,52,53].

To summarize, we can check coherence by

- verifying the feasibility of disjunctive linear constraints (18) and (19); or by
- running Algorithm 4, using the PSAT-IP algorithm whenever necessary.

6. 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 4 GB of memory. The PSAT-IP algorithm is very compact, using only 45 lines of code (basically a direct translation of Algorithm 1 into CPLEX calls). We then produced an extended version that directly handles probability assessments over clauses as in Expression (6), using Expressions (7), (8), and (9). It might be interesting to consider conditional or expectation assessments, but to keep the number of controlled parameters manageable, we decided to focus on Normal Form PSAT problems, with a few variants.

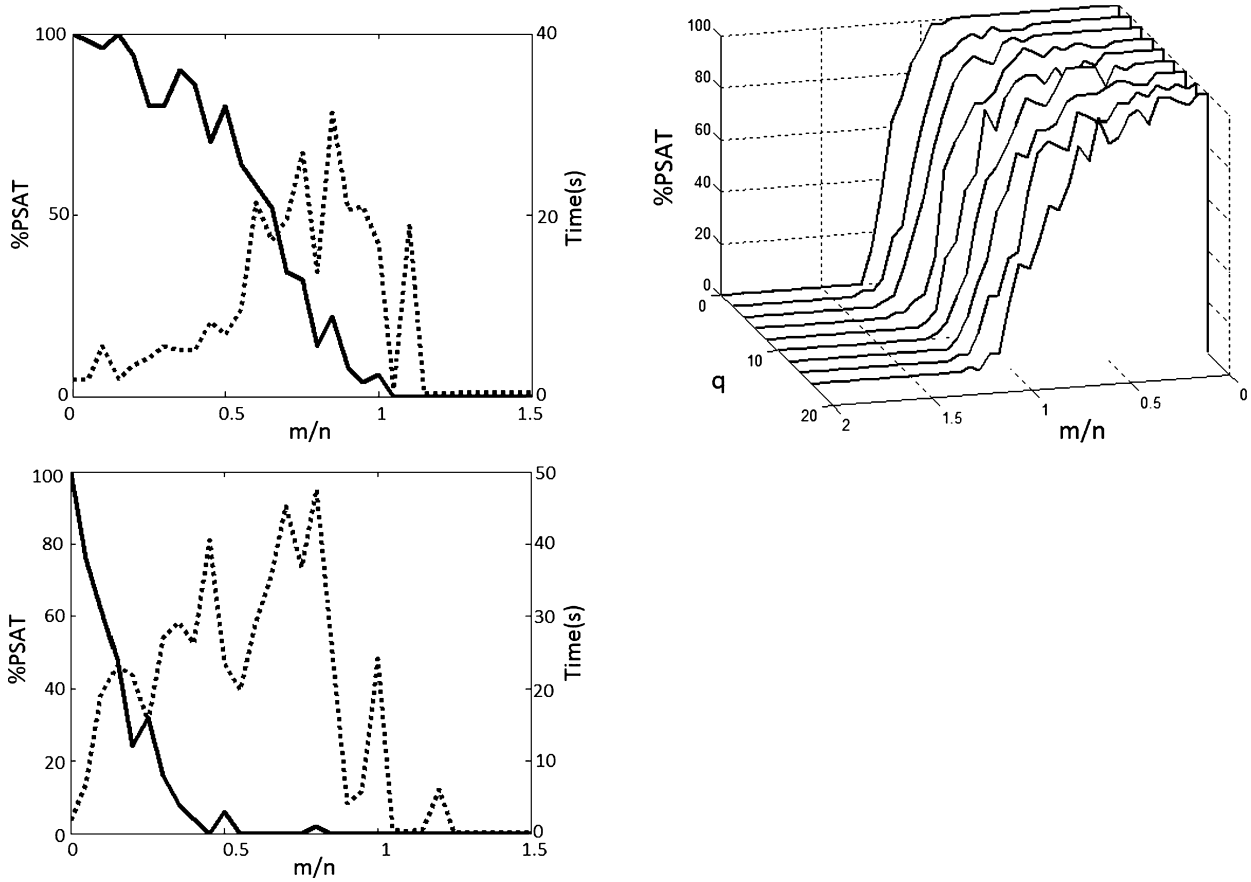

Fig. 1. Experiments with random 2-PSAT in Normal Form (each point averages over 50 randomly generated problems). Top left graph shows phase transition for $q = 30$, $n = 1000$, and m varying from 0 to 1500; the solid line shows the percentage of satisfiable problems (note that for small m/n we have close to 100% satisfiable problems, while for large m/n we have close to 0% satisfiable problems), while the dotted line shows the mean time spent in solving each problem (note that computational effort grows around the transition region). Bottom left graphs display the same information for $n = 100$ (note the change in the transition). Right: as q varies, the transition region changes (here $n = 1000$ and m goes from 0 to 2000).

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 [26].

Additionally, we were particularly interested in investigating phase transition phenomena. Until the work of Baioletti et al. [5], and Finger and De Bona [26], there was no evidence of phase transition in the literature; their work indicates that at fixed values of q, variation of the ratio m/n does lead to a qualitative change in the percentage of satisfiable problems (apparently, for small and large values of m/n most problems are respectively satisfiable and unsatisfiable). Moreover, at the transition between satisfiable/unsatisfiable regions, the computational effort to determine satisfiability markedly grows. 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. For the data in all of our graphs, we set a time limit of 10 min per problem; some of the more difficult problems did not finish within this time limit, but it should be noted that all such problems occur within the transition region; consequently, the average computational effort during the transition is even larger than shown in the graphs.

We start by describing experiments with randomly generated PSAT problems in Normal Form.

Fig. 1 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 $[0, 1]$ (always equality; that is, assessments are $\mathbb{P}(A_i) = \alpha_i$). Each point in each graph conveys mean values for 50 different random PSAT problems.

The left graphs in Fig. 1 show typical behavior for random 2-PSAT: the solid line indicates the percentage of satisfiable problems, and the dotted 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, keeping $q = 30$; 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

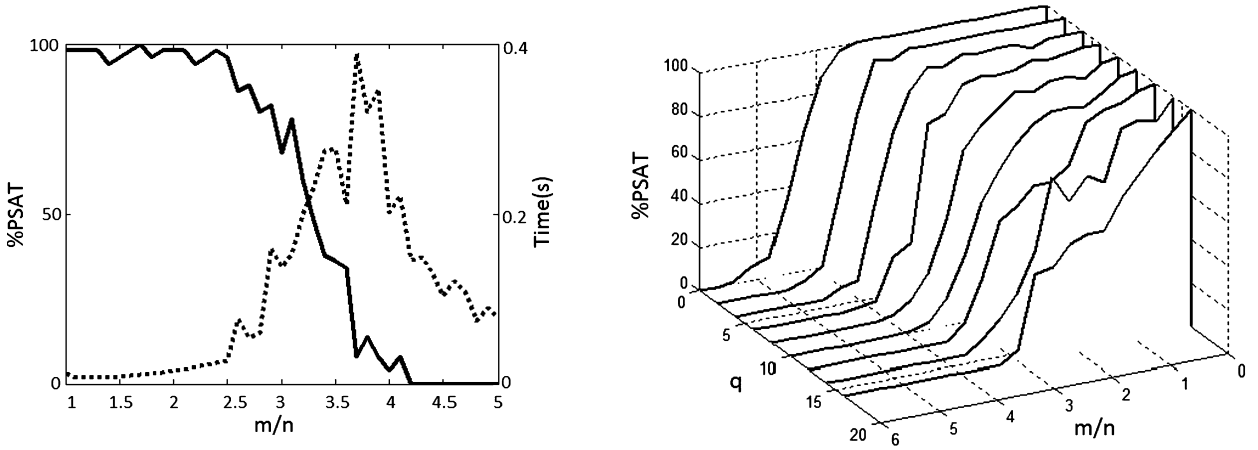

Fig. 2. Experiments with random 3-PSAT in Normal Form (each point averages over 50 randomly generated problems). Left graph shows phase transition for $q = 10$, $n = 40$, and m varying from 0 to 200; solid and dotted lines as in Fig. 1. Right: as q varies, the transition region changes (here $n = 50$ and m goes from 0 to 300).

Table 1
Experiments with 2-PSAT and 3-PSAT, first part.

n	q	m	2-PSAT, mean time sec. (standard deviation)	3-PSAT, mean time sec. (standard deviation)
500	25	500	3.02 (8.25)	1.68 (1.07)
500	25	750	0.13 (0.01)	2.15 (1.19)
500	25	1000	0.16 (0.01)	13.49 (5.09)
500	50	500	1.72 (8.67)	20.12 (9.96)
500	50	750	0.29 (0.02)	30.09 (13.86)
500	50	1000	0.35 (0.03)	191.37 (52.45)
750	25	500	4.2 (6.24)	1.78 (1.07)
750	25	750	5.09 (12.19)	1.47 (0.65)
750	25	1000	0.18 (0.01)	1.85 (0.96)
750	50	500	1.72 (10.15)	20.12 (10.76)

Table 2
Experiments with 2-PSAT and 3-PSAT, second part.

n	q	m	2-PSAT, mean time sec. (standard deviation)	3-PSAT, mean time sec. (standard deviation)
750	50	750	0.39 (0.14)	16.95 (7.14)
750	50	1000	0.39 (0.04)	21.49 (11.88)
1000	25	500	1.93 (1.46)	1.93 (1.13)
1000	25	750	1.40 (1.39)	1.80 (0.90)
1000	25	1000	0.62 (1.04)	1.66 (0.85)
1000	25	1500	0.27 (0.06)	2.22 (1.88)
1000	25	2000	0.35 (0.03)	27.33 (6.31)
1000	50	500	2.92 (3.31)	19.68 (9.10)
1000	50	750	1.44 (2.50)	20.18 (7.59)
1000	50	1000	0.44 (0.12)	22.04 (11.27)

investigation by Finger and De Bona [26]). The lower graph conveys the same information, but now for $q = 30$ and $n = 100$. 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 among n, q and m in ways that are not observed in 2-SAT (where the phase transition occurs at $m/n = 1$). 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 as m varies and n is fixed at 1000.

Similar results are displayed in Fig. 2 for 3-PSAT. In the left graph we see typical phase transition behavior, now centered around $m/n \approx 3$. The reason we show this particular graph (with $n = 40$, $q = 4$) is that the same experiment is reported by Finger and De Bona [26]. The right graph shows the change in the location of phase transition as q varies, similarly to what happens with 2-PSAT, for $n = 50$.

To give a better feel of the times involved in solving PSAT problems with our method, Tables 1 and 2 summarize a large variety of tests; each entry is the mean of 50 distinct PSAT problems. Time was measured using the CPLEX facility (in nanoseconds). Note that we are only generating PSAT problems in Normal Form. Also note that it is not correct to expect

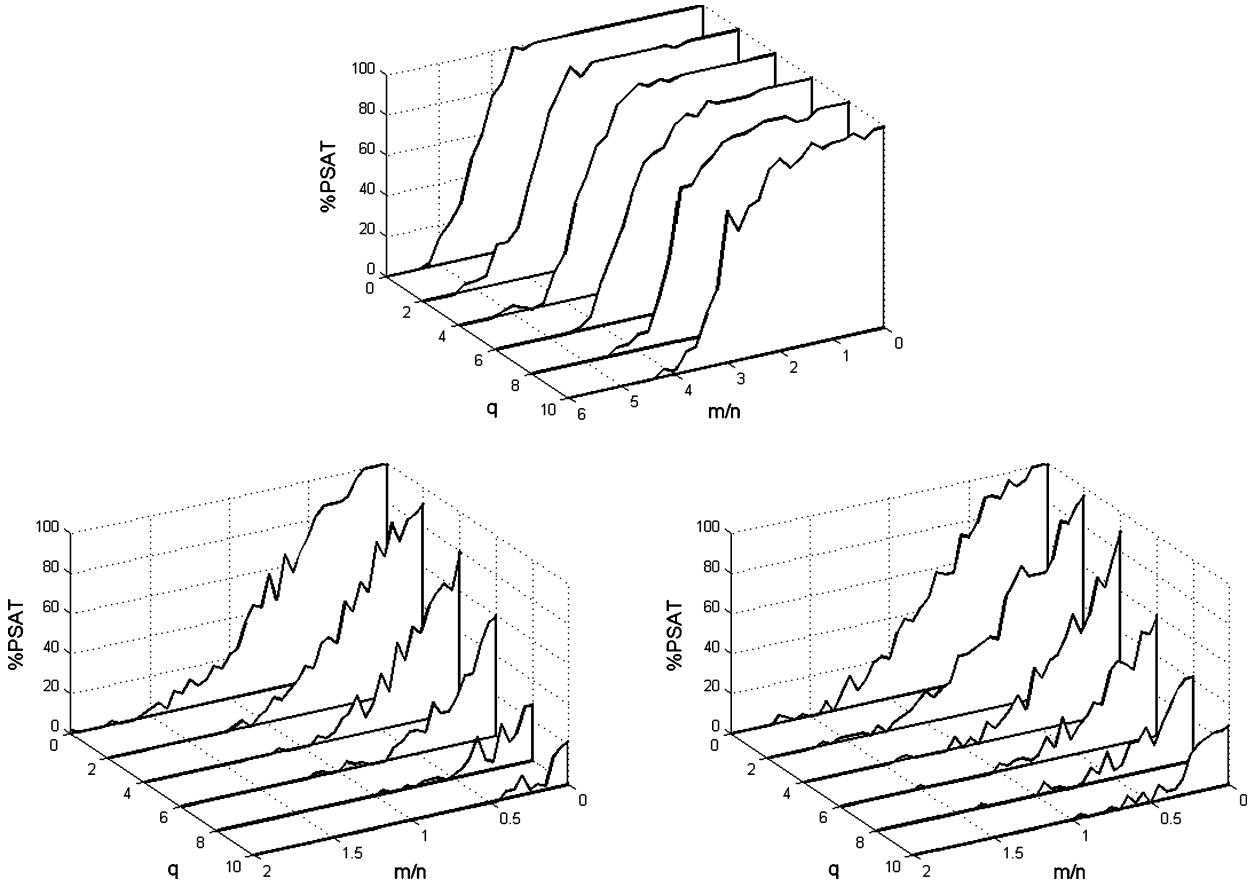

Fig. 3. Experiments with variations on Normal Form (each point averages over 50 randomly generated problems). Top graph shows phase transition for random problems generated as before but with assessments $\mathbb{P}(A_i) \geq \alpha_i$, for $n = 40$. Bottom graphs show phase transition for assessments over clauses, both assessments with equality (left) and inequality (right), for $n = 40$.

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. For 2-PSAT problems, we have observed phase transition at values of m/n close to one, decreasing with increasing q. For 3-PSAT, the phase transition region tends to be more spread out, as m/n ranges from about 2 to 4. Note that standard deviations usually grow with mean time; that is, within the transition regions we do not find that all problems become harder; rather, some problems are easy while others are much harder than the mean time suggests.

We now summarize a number of experiments conducted with variants of Normal Form. First, we considered problems where assessments are of the form $\mathbb{P}(A_i) \geq \alpha_i$ (instead of $\mathbb{P}(A_i) = \alpha_i$). The top graph in Fig. 3 shows the phase transition for such a situation; note that overall there is no change in behavior as we move from equalities to inequalities. We then generated PSAT problems with probabilities over clauses; that is, assessments as in Expression (6). Following the procedure used by Jaumard et al. [40], we uniformly sampled a number of literals between 1 and k for each clause. In our experiments we used $k = 4$. For each generated clause, a probability value was randomly selected. The two graphs in the bottom of Fig. 3 show phase transitions with such PSAT problems; the left graph was produced with equality assessments, while the right graph was produced with inequality assessments.

7. Conclusion

In this paper we have applied integer programming techniques to probabilistic satisfiability and coherence checking. Given current linear solver technology, our algorithms are relatively easy to code and inherit any parallelization and numerical stability techniques already implemented in solvers. Our method is quite effective for small to medium problems, and the focus on integer programming allows one to consider other techniques such as logical constraints and disjunctive programming. We have presented a number of variants that can benefit from integer programming, including the problem of coherence checking. Finally, we have presented a preliminary analysis of phase transition in PSAT that refines previous results in the literature.

Future work should extend the ideas in this paper to inference problems, particularly inference in the coherence setting. It is also important to further study the phase transition phenomenon, in particular to verify the relationship between the transition region and the value of q.

Acknowledgements

Both authors received support by CNPq grant 305395/2010-6. An initial version of this paper appeared at the ECSQARU 2013 conference; we thank the reviewers of that version for very useful suggestions, in particular for pointing us to Refs. [5] and [6]. Thanks also to the reviewers of this version, whose comments led to several improvements to the text; one reviewer pointed us to Refs. [28] and [29] and suggested a simplified way to compute inferences.

In our empirical analysis we employed the CPLEX Optimizer package, available for academic use; thanks to the CPLEX developer (IBM Corporation) for its generosity (we used resources located at the web site http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/index.html).

References

[1] Ernest W. Adams, A Primer of Probability Logic, CSLI Publications, Stanford, CA, 2002.
[2] Kim Allan Andersen, Daniele Pretolani, Easy cases of probabilistic satisfiability, Ann. Math. Artif. Intell. 33 (1) (2001) 69–91.
[3] Thomas Augustin, Frank P.A. Coolen, Gert de Cooman, Matthias C.M. Troffaes, Introduction to Imprecise Probabilities, Wiley, 2014.
[4] M. Baioletti, A. Capotorti, S. Tulipani, B. Vantaggi, Elimination of Boolean variables for probabilistic coherence, Soft Comput. 4 (2000) 81–88.
[5] Marco Baioletti, Andrea Capotorti, Sauro Tulipani, An empirical complexity study for a 2CPA solver, in: B. Bouchon-Meunier, G. Coletti, R.R. Yager (Eds.), Modern Information Processing: From Theory to Applications, Elsevier Science, 2005, pp. 73–84.
[6] Marco Baioletti, Andrea Capotorti, Sauro Tulipani, Barbara Vantaggi, Simplification rules for the coherent probability assessment problem, Ann. Math. Artif. Intell. 35 (2002) 11–28.
[7] Dimitris Bertsimas, John N. Tsitsiklis, Introduction to Linear Optimization, Athena Scientific, Belmont, Massachusetts, 1997.
[8] Veronica Biazzo, Angelo Gilio, A generalization of the fundamental theorem of de Finetti for imprecise conditional probability assessments, Int. J. Approx. Reason. 24 (2–3) (2000) 251–272.
[9] Veronica Biazzo, Angelo Gilio, On the linear structure of betting criterion and the checking of coherence, Ann. Math. Artif. Intell. 35 (2002) 83–106.
[10] Veronica Biazzo, Angelo Gilio, Thomas Lukasiewicz, Giuseppe Sanfilippo, Probabilistic logic under coherence: complexity and algorithms, in: Gert de Cooman, Terrence L. Fine, Teddy Seidenfeld (Eds.), International Symposium on Imprecise Probabilities and Their Applications, The Netherlands, Shaker, 2001, pp. 51–61.
[11] Veronica Biazzo, Angelo Gilio, Thomas Lukasiewicz, Giuseppe Sanfilippo, Probabilistic logic under coherence, model-theoretic probabilistic logic, and default reasoning in system P, J. Appl. Non-Class. Log. 12 (2) (2002) 189–213.
[12] Veronica Biazzo, Angelo Gilio, Thomas Lukasiewicz, Giuseppe Sanfilippo, Probabilistic logic under coherence: complexity and algorithms, Ann. Math. Artif. Intell. 45 (1–2) (2005) 35–81.
[13] George Boole, The Laws of Thought, Dover Edition, 1958.
[14] Stephen P. Bradley, Arnoldo C. Hax, Thomas L. Magnanti, Applied Mathematical Programming, Addison–Wesley, 1977.
[15] G. Bruno, Angelo Gilio, Applicazione del metodo del simplesso al teorema fondamentale per le probabilità nella concezione soggettivistica, Statistica 40 (1980) 337–344.
[16] Andrea Capotorti, L. Galli, Barbara Vantaggi, Locally strong coherence and inference with lower–upper probabilities, Soft Comput. 7 (5) (2003) 280–287.
[17] Andrea Capotorti, Barbara Vantaggi, Locally strong coherence in inference processes, Ann. Math. Artif. Intell. 35 (2002) 125–149.
[18] Vijay Chandru, John Hooker, Optimization Methods for Logical Inference, John Wiley & Sons Inc., 1999.
[19] G. Coletti, R. Scozzafava, Probabilistic Logic in a Coherent Setting, Trends Log., vol. 15, Kluwer, Dordrecht, 2002.
[20] Giulianella Coletti, Coherent numerical and ordinal probabilistic assessments, IEEE Trans. Syst. Man Cybern. 24 (12) (1994) 1747–1753.
[21] Giulianella Coletti, Romano Scozzafava, The role of coherence in eliciting and handling imprecise probabilities and its application to medical diagnosis, Inf. Sci. 130 (2000) 41–65.
[22] Inés Couso, Serafín Moral, Sets of desirable gambles: conditioning, representation, and precise probabilities, Int. J. Approx. Reason. 52 (2011) 1034–1055.
[23] Bruno de Finetti, Theory of Probability, vols. 1–2, Wiley, New York, 1974.
[24] Lester E. Dubins, Finitely additive conditional probability, conglomerability and disintegrations, Ann. Stat. 3 (1) (1975) 89–99.
[25] R. Fagin, J.Y. Halpern, N. Megiddo, A logic for reasoning about probabilities, Inf. Comput. 87 (1990) 78–128.
[26] Marcelo Finger, Glauber De Bona, Probabilistic satisfiability: logic-based algorithms and phase transition, in: International Joint Conference on Artificial Intelligence, 2011, pp. 528–533.
[27] A.M. Frisch, P. Haddawy, Anytime deduction for probabilistic logic, Artif. Intell. 69 (1994) 93–122.
[28] Linda van der Gaag, On probability intervals and their updating, Technical report RUU-CS-90-22, Department of Computer Science, Utrecht University, 1990.
[29] Linda van der Gaag, Computing probability intervals under independency constraints, in: Conference on Uncertainty in Artificial Intelligence, 1990, pp. 457–466.
[30] P. Gent, T. Walsh, The SAT phase transition, in: European Conference on Artificial Intelligence, 1994, pp. 105–109.
[31] G. Georgakopoulos, D. Kavvadias, C.H. Papadimitriou, Probabilistic satisfiability, J. Complex. 4 (1988) 1–11.
[32] Angelo Gilio, Algorithms for precise and imprecise conditional probability assessments, in: G. Coletti, D. Dubois, R. Scozzafava (Eds.), Mathematical Models for Handling Partial Knowledge in Artificial Intelligence, 1995, pp. 231–254, New York.
[33] T. Hailperin, Best possible inequalities for the probability of a logical function of events, Am. Math. Mon. 72 (1965) 343–359.
[34] Theodore Hailperin, Boole's Logic and Probability: A Critical Exposition from the Standpoint of Contemporary Algebra, Logic, and Probability Theory, North-Holland, Amsterdam, 1976.
[35] Joseph Y. Halpern, Reasoning About Uncertainty, MIT Press, Cambridge, Massachusetts, 2003.
[36] Peter J. Hammond, Elementary non-Archimedean representations of probability for decision theory and games, in: P. Humphreys (Ed.), Patrick Suppes: Scientific Philosopher, vol. 1, Kluwer, Dordrecht, The Netherlands, 1994, pp. 25–59.
[37] Pierre Hansen, Brigitte Jaumard, Probabilistic satisfiability, Technical report G-96-31, Les Cahiers du GERAD, École Polytechique de Montréal, 1996.
[38] Pierre Hansen, Sylvain Perron, Merging the local and global approaches to probabilistic satisfiability, Int. J. Approx. Reason. 47 (2008) 125–140.
[39] S. Holzer, On coherence and conditional prevision, Boll. Unione Mat. Ital., Ser. VI, Anal. Funz. Appl. IV-C (1) (1985) 441–460.

[40] B. Jaumard, P. Hansen, M.P. de Aragão, Column generation methods for probabilistic logic, ORSA J. Comput. 3 (2) (1991) 135–148.
[41] Pavel Klinov, Bijan Parsia, A hybrid method for probabilistic satisfiability, in: N. Bjorner, V. Sofronie-Stokkermans (Eds.), International Conference on Automated Deduction, Springer-Verlag, 2011, pp. 354–368.
[42] Peter Krauss, Representation of conditional probability measures on Boolean algebras, Acta Math. Acad. Sci. Hung. 19 (3–4) (1968) 229–241.
[43] H.E. Kyburg Jr., H.E. Smokler, Studies in Subjective Probability, John Wiley & Sons Inc., New York, 1964.
[44] R. Sherman Lehman, On confirmation and rational betting, J. Symb. Log. 20 (3) (1955) 251–262.
[45] Isaac Levi, The Enterprise of Knowledge, MIT Press, Cambridge, Massachusetts, 1980.
[46] Thomas Lukasiewicz, Expressive probabilistic description logics, Artif. Intell. 172 (6–7) (April 2008) 852–883.
[47] V. McGee, Learning the impossible, in: E. Bells, B. Skyrms (Eds.), Probability and Conditionals, Cambridge University Press, 1994, pp. 179–199.
[48] Enrique Miranda, A survey of the theory of coherent lower previsions, Int. J. Approx. Reason. 48 (2) (2008) 628–658.
[49] R. Myerson, Game Theory: Analysis of Conflict, Harvard University Press, Cambridge, MA, 1991.
[50] Raymond Ng, V.S. Subrahmanian, Probabilistic logic programming, Inf. Comput. 101 (2) (1992) 150–201.
[51] N.J. Nilsson, Probabilistic logic, Artif. Intell. 28 (1986) 71–87.
[52] E. Quaeghebeur, The CONEstrip algorithm, in: Advances in Intelligent Systems and Computing, vol. 190, 2013, pp. 45–54.
[53] E. Quaeghebeur, A propositional CONEstrip algorithm, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 2014, pp. 466–475.
[54] E. Regazzini, Finitely additive conditional probability, Rend. Semin. Mat. Fis. 55 (1985) 69–89.
[55] Paolo Vicig, An algorithm for imprecise conditional probability assessments in expert systems, in: International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, 1996, pp. 61–66.
[56] Carl G. Wagner, The Smith–Walley interpretation of subjective probability: an appreciation, Stud. Log. 86 (2007) 343–350.
[57] P. Walley, Statistical Reasoning with Imprecise Probabilities, Chapman and Hall, London, 1991.
[58] Peter Walley, Renato Pelessoni, Paolo Vicig, Direct algorithms for checking consistency and making inferences from conditional probability assessments, J. Stat. Plan. Inference 126 (1) (2004) 119–151.
[59] H. Paul Williams, The formulation and solution of discrete optimisation problems, in: G.M. Appa, L.S. Pitsoulis, H.P. Williams (Eds.), Handbook on Modelling for Discrete Optimization, Springer, 2006, pp. 3–38.
[60] H. Paul Williams, Logic and Integer Programming, Springer, London, 2009.
[61] P.M. Williams, Coherence, strict coherence and zero probabilities, in: International Congress of Logic, Methodology and Philos. Sci., vol. VI, 1975, pp. 29–30.
[62] P.M. Williams, Notes on conditional previsions, Int. J. Approx. Reason. 44 (2007) 366–383.