Probabilistic Satisfiability: Logic-based Algorithms and Phase Transition*

Keywords: Logic and Probability, Probabilistic Satisfiability, Phase Transition, Normal Form

Abstract

In this paper, we study algorithms for probabilistic satisfiability (PSAT), an NP-complete problem, and their empiric complexity distribution. We define a PSAT normal form, based on which we propose two logic-based algorithms: a reduction of normal form PSAT instances to SAT, and a linear-algebraic algorithm with a logic-based column generation strategy. We conclude that both algorithms present a phase transition behaviour and that the latter has a much better performance. We discuss the role of logic and the normal form in the detection of the phase transition.

1 Introduction

Probabilistic satisfiability (PSAT) is an NP-complete problem that requires the joint application of deductive and probabilistic reasoning. It consists of an assignment of probabilities to a set of propositional formulas, and its solution consists of a decision on whether this assignment is consistent. For example, consider Problem 1.1, in which the goal is to verify the truthfulness of the claims.

> **Problem 1.1** Three friends usually go to a bar, and every night at least two of them are at their exclusive table. However, each of them claims to go to the bar "only" 60% of the nights.
>
> *Question:* Can they be telling the truth? Why?

In this case, the statement of the friends has to be checked against an established fact, namely the presence of two of them every night. Another instance of PSAT is the following.

> **Problem 1.2** Selecting architecture students is notoriously hard. Staff at a university analysed students that received their final degrees and concluded that:
>
> - at least 75% had good grades in their Math entrance test or showed good drawing abilities;
> - at most a third had good Math grades or did not show any drawing abilities.
>
> *Question:* Is this consistent with the low expectation (at most 15%) of students that do well in the Math entrance tests?

This problem involves imprecise probability judgements, widening the scope of application areas. In fact, there is a large number of potential application areas for PSAT, from machine learning to the modelling of biological processes, from hardware and software verification to economics and econometrics. However, there are very few, if any, practical algorithms available, used in limited applications.

This work aims at studying PSAT algorithms to reveal its empiric *complexity distribution* and, in particular, detect the existence of a *phase-transition* behaviour.

1.1 A Brief History of PSAT

The original formulation of PSAT is attributed to George Boole [1854]. The problem has since been independently rediscovered several times (see [Hailperin, 1986; Hansen and Jaumard, 2000]) until it was presented to the Computer Science and Artificial Intelligence community by Nilsson [Nilsson, 1986] and was shown to be an NP-complete problem, even for cases where the corresponding classical satisfiability is known to be in PTIME [Georgakopoulos *et al.*, 1988].

The PSAT problem is formulated in terms of a linear algebraic problem of exponential size. The vast majority of algorithms for PSAT solving in the literature are based on linear programming techniques, such as column generation, enhanced by several types of heuristics [Kavvadias and Papadimitriou, 1990; Hansen *et al.*, 1995].

Boole's original formulation of the PSAT problem did not consider conditional probabilities, but extensions for them have been developed [Hailperin, 1986; Hansen *et al.*, 1995; Hansen and Jaumard, 2000; Walley *et al.*, 2004]; the latter two works also works covered extensions of PSAT with imprecise probabilities. A few tractable fragments of PSAT were presented [Andersen and Pretolani, 2001]. In this work, however, we concentrate on PSAT's original formulation.

With respect to practical implementations, initial attempts to implement a solution led to the belief that the problem appeared to be very hard [Nilsson, 1993]. There are computational results reported in [Jaumard *et al.*, 1991] and [Kavvadias and Papadimitriou, 1990], solving PSAT problems of 140 variables and 300 sentences. However, to the best of our knowledge, these implementations were rarely, if ever,

*This work was supported by Fapesp Thematic Project 2008/03995-5 (LOGPROB).

employed to solve practical instances of PSAT problems or served as a basis to practical applications.

1.2 Phase Transition of NP-complete Problems

Cheeseman *et al.* [1991] presented the phase-transition phenomenon for NP-complete problems and conjectured that it is a property of all problems in that class. Gent and Walsh [1994] studied phase transition for SAT problems parameterised by the rate m/n, where m is the number of clauses in a 3-SAT instance and n is the number of variables, and showed that the harder instances concentrate around a point where $m/n = P_t$, the *phase transition point*. When the rate m/n is small (< 3) almost all instances are satisfiable, and when this rate is high (> 6) instances are unsatisfiable, and the decision time remains low at both cases. At the phase transition point P_t, the number of expected satisfiable instances is 50%, which for 3-SAT is $P_t \approx 4.3$.

To the best of our knowledge, no phase transition has been described for PSAT so far.

Very efficient SAT solvers are now available [Moskewicz *et al.*, 2001; Eén and Sörensson, 2003], but the SAT phase transition behaviour remains independent of implementation and number of atoms. However it relies on presenting the problem in a *normal form*, usually in clausal form with a fixed number of atoms per clause.

Efficient SAT-solvers enabled the solution of other NP-complete problems by translating them to a SAT instance and applying the solvers. Cook's theorem guarantees that there exists a polynomial-time translation from any NP-complete problem into SAT. However, no such translation is found in the literature from PSAT to SAT.

1.3 Goals and Organisation of the Paper

The aim of this work is to study logic-based PSAT algorithms and their empiric complexity distribution, looking for a PSAT phase transition behaviour.

For that, we formally present the PSAT problem and develop a logic-based normal form of it in Section 2, based on which two kinds of algorithms are developed. A (canonical) reduction of PSAT to SAT is developed in Section 3. And an efficient SAT-based modification of usual linear programming PSAT algorithms in Section 4. The empiric complexity distribution on both algorithms is presented and PSAT phase transition behaviour is detected. We end with a discussion on what helped the detection of the PSAT phase transition and its efficiency.

2 The PSAT Problem

A *PSAT instance* is a set $\Sigma = \{P(\alpha_i) \bowtie_i p_i | 1 \leq i \leq k\}$, where $\alpha_1, \ldots, \alpha_k$ are classical propositional formulas defined on n logical variables $\mathcal{P} = \{x_1, \ldots, x_n\}$, which are restricted by probability assignments $P(\alpha_i) \bowtie_i p_i$, $\bowtie_i \in \{=, \leq, \geq\}$ and $1 \leq i \leq k$.

There are 2^n possible propositional valuations v over the logical variables, $v : \mathcal{P} \to \{0, 1\}$; each such valuation is extended, as usual, to all formulas, $v : \mathcal{L} \to \{0, 1\}$. A *probability distribution over propositional valuations* $\pi : V \to [0, 1]$, is a function that maps every valuation to a value in the real interval $[0, 1]$ such that $\sum_{i=1}^{2^n} \pi(v_i) = 1$. The probability of a formula α according to distribution π is given by $P_\pi(\alpha) = \sum\{\pi(v_i) | v_i(\alpha) = 1\}$.

Nilsson [1986]'s formulation of PSAT considers a $k \times 2^n$ matrix $A = [a_{ij}]$ such that $a_{ij} = v_j(\alpha_i)$. The *probabilistic satisfiability problem* is to decide if there is a probability vector π of dimension 2^n that obeys the *PSAT restriction*:

$$\begin{array}{rcl} A\pi & \bowtie & p \\ \sum \pi_i & = & 1 \\ \pi & \geq & 0 \end{array} \quad (1)$$

A *PSAT instance* Σ is *satisfiable* iff its associated PSAT restriction (1) has a solution. If π is a solution to (1) we say that π satisfies Σ. The last two conditions of (1) force π to be a probability distribution. Usually the first two conditions of (1) are joined, A is a $(k + 1) \times 2^n$ matrix with 1's at its first line, $p_1 = 1$ in vector $p_{(k+1)\times 1}$, so $\bowtie_1$-relation is "=".

Example 2.1 Consider Problem 1.1, with friends $\{1, 2, 3\}$. Let x_i represent that person i is at the bar tonight, $i \in \{1, 2, 3\}$. As every night at least two friends are at the bar, no two friends can be absent at the same night, represented by $\neg(\neg x_i \wedge \neg x_j)$ with 100% certainty for $i \neq j$:

$$P(x_1 \vee x_2) = P(x_1 \vee x_3) = P(x_2 \vee x_3) = 1.$$

Furthermore, each claims to be at the bar "only" 60% of the time:

$$P(x_1) = P(x_2) = P(x_3) = 0.6,$$

and the question is if there exists a probability distribution that simultaneously satisfies these 6 probability assignments.

Consider now Problem 1.2. Let x_1 mean that a student has good grades in the Math entrance test and x_2 mean that that student showed good drawing abilities. In this case, we obtain the restrictions Σ:

$$P(x_1 \vee x_2) \geq 0.75 \quad P(x_1 \vee \neg x_2) \leq 1/3 \quad P(x_1) \leq 0.15$$

Consider a probability distribution π and all the possible valuations as follows.

π	x_1	x_2	$x_1 \vee x_2$	$x_1 \vee \neg x_2$
0.20	0	0	0	1
0.05	1	0	1	1
0.70	0	1	1	0
0.05	1	1	1	1
1.00	0.10	0.75	0.80	0.30

which jointly satisfies the assignments above, so Problem 1.2 is satisfiable. We are going to study algorithms to compute one such probability distribution if one exists. □

An important result of [Georgakopoulos *et al.*, 1988] guarantees that a solvable PSAT instance has a "small" witness.

Fact 2.2 *If a PSAT instance* $\Sigma = \{P(\alpha_i) = p_i | 1 \leq i \leq k\}$ *has a solution, then there are* $k + 1$ *columns of* A *such that the system* $A_{(k+1)\times(k+1)}\pi = p_{(k+1)\times 1}$ *has a solution* $\pi \geq 0$.

The solution given by Fact 2.2 serves as an NP-certificate for this instance, so PSAT is in NP. Furthermore, as *propositional satisfiability* (SAT) is a subproblem obtained when all $p_i = 1$, PSAT is NP-hard. So PSAT is NP-complete.

It follows from the Cook-Levin Theorem [Cook, 1971] that there must be a polynomial time reduction from PSAT to SAT. However, finding an efficient such reduction is neither obvious nor easy at all. In the following, we study some logical properties of PSAT instances to study ways in which this kind of reduction can be performed.

First, some notation. If A is an $m \times n$ matrix, A^j represents its j-th column, and if b is an m-dimensional column, $A[j := b]$ represents the matrix obtained by substituting b for A^j; if A is square matrix, $|A|$ is A's determinant. If A is a $\{0,1\}$-matrix, where each line corresponds to a variable, than each A^j can be seen as valuation.

2.1 A PSAT Normal Form

We say that a PSAT instance $\Sigma = \{P(\alpha_i) \bowtie_i p_i | 1 \leq i \leq l\}$ is in *(atomic) normal form* if it can be partitioned in two sets, $\langle \Gamma, \Psi \rangle$, where $\Gamma = \{P(\alpha_i) = 1 | 1 \leq i \leq m\}$ and $\Psi = \{P(y_i) = p_i | y_i \text{ is an atom}, 1 \leq i \leq k\}$, with $0 < p_i < 1$, where $l = m + k$. The partition Γ is the SAT part of the normal form, usually represented only as a set of propositional formulas and Ψ is the *atomic probability assignment* part.

By adding a small number of extra variables, any PSAT instance can be brought to normal form.

Theorem 2.3 (Normal Form) *For every PSAT instance Σ there exists an atomic normal form instance $\langle \Gamma, \Psi \rangle$ such that Σ is a satisfiable iff $\langle \Gamma, \Psi \rangle$ is; the atomic instance can be built from Σ in polynomial time.* □

Example 2.4 The presentation of Problem 1.1 in Example 2.1 is already in normal form, with $\Gamma = \{x_1 \vee x_2, x_1 \vee x_3, x_2 \vee x_3\}$ and $\Psi = \{P(x_1) = P(x_2) = P(x_3) = 0.6\}$. This indicates that the normal form is a "natural" form in many cases, such as when one wants to confront a theory Γ with the evidence Ψ.

For the formulation of Problem 1.2, we start by rewriting the problem in the form $P(\alpha) \leq p$ as

$$\Sigma = \{P(\neg(x_1 \vee x_2)) \leq 0.25, P(x_1 \vee \neg x_2) \leq \tfrac{1}{3}, P(x_1) \leq 0.15\}$$

Add three new variables, y_1, y_2, y_3 and make

$$\begin{aligned} \Gamma &= \{\ \neg(x_1 \vee x_2) \to y_1,\ (x_1 \vee \neg x_2) \to y_2,\ x_1 \to y_3\ \} \\ &\equiv \{\ x_1 \vee x_2 \vee y_1, \neg x_1 \vee y_2, x_2 \vee y_2, x_1 \to y_3\ \} \end{aligned}$$

and $\Psi = \{P(y_1) = 0.25, P(y_2) = \frac{1}{3}, P(y_3) = 0.15\}$. □

The normal form allows us to see a PSAT instance $\langle \Gamma, \Psi \rangle$ as an interaction between a probability problem Ψ and a SAT instance Γ. Solutions to the instance can be seen as solutions to Ψ constrained by the SAT instance Γ.

This is formalised as follows. A valuation v over $y_1, \ldots, y_k$ is Γ-*consistent* if there is an extension of v over $y_1, \ldots, y_k, x_1, \ldots, x_n$ such that $v(\Gamma) = 1$.

Lemma 2.5 *A normal form instance $\langle \Gamma, \Psi \rangle$ is satisfiable iff there is a $(k+1) \times (k+1)$-matrix A_Ψ, such that all of its columns are Γ-consistent and $A_\Psi \pi = p$ has a solution $\pi \geq 0$.*

Lemma 2.5 is the basis for the PSAT solving algorithms that we present next.

3 Reduction of PSAT to SAT

A *(canonical) reduction* is a polynomial time translation of an instance of PSAT into an instance of SAT, such that the PSAT instance is satisfiable iff the SAT instance is. As both PSAT and SAT are NP-complete, the existence of a reduction is guaranteed.

We present a reduction that inputs a normal form PSAT instance $\langle \Gamma, \Psi \rangle$ and outputs a SAT formula that encodes Lemma 2.5. Given p extracted from Ψ, we search for a matrix $A_{(k+1)\times(k+1)}$ whose columns are Γ-consistent and a probability distribution $\pi \geq 0$ such that $A\pi = p$. We assume that the elements of p and π are represented by a fixed point number in binary positional system with precision b_π bits, where each bit is a boolean variable, the higher the bit's index the higher its significance. As $a_{ij}, \pi^{i_1}_{j_1}, p^{i_2}_{j_2} \in \{0,1\}$, each such element is represented by a boolean variable. This is represented by the following schema.

$$\begin{bmatrix} 1 & \cdots & 1 \\ a_{1,1} & \cdots & a_{1,k+1} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,k+1} \end{bmatrix} \cdot \begin{bmatrix} \langle 0.\pi^1_{b_\pi} \cdots \pi^1_1 \rangle \\ \\ \vdots \\ \\ \langle 0.\pi^{k+1}_{b_\pi} \cdots \pi^{k+1}_1 \rangle \end{bmatrix} = \begin{bmatrix} \langle 1.0 \ldots 0 \rangle \\ \langle 0.p^1_{b_\pi} \cdots p^1_1 \rangle \\ \vdots \\ \langle 0.p^k_{b_\pi} \cdots p^k_1 \rangle \end{bmatrix}$$

3.1 Determining the Precision

The hard part of this encoding is to find a large enough precision b_π that guarantees the correctness and polynomial size of the encoding, assuming a fixed number of bits in the input.

To guarantee a polynomial time reduction, the number of bits b_π in which the components of vector π are encoded has to be bounded by a polynomial on the size of input of the PSAT instance. By Cramer's rule, $\pi_j = \frac{|A[j:=p]|}{|A|}$ so we estimate the maximum precision b_π as twice the number of bits of the maximum possible determinant of a $0,1$-matrix A. For that, we use a result due to J. Hadamard who solved the Maximum Determinant Problem; see [Garling, 2007].

Fact 3.1 (Hadamard, 1893) *Let $A = [a_{ij}]_{n\times n}$, $a_{ij} \in \{0,1\}$. Then $|A| \leq \dfrac{(n+1)^{(n+1)/2}}{2^n}$.* □

Thus $b_\pi = 2\left(\left\lceil \frac{k+2}{2} \log(k+2) \right\rceil - (k+1)\right) = O(k \log k)$, and the reduction is polynomial-time. This bound presupposes rational arithmetic; correctness is kept in the presence of truncation errors due to the convexity of the solution space.

3.2 The SAT Instance

Multiplication of $a_{ij} \cdot \pi_j$ is encoded as a simple conjunction of each bit-variable of π_j with the variable representing a_{ij}. Sums will be directly encoded as bitwise sums for each position, which demands a carry-bit and a result-bit for each position. Equality is encoded as bitwise equivalence. Due to space reasons we do not detail the reduction formulas here.

We only point that the number of variables in the clausal SAT instance is $O(k^3 \log k)$ and that the number of clauses is also $O(k^3 \log k)$.

This implies that reduction makes PSAT solving a lot *less efficient* than SAT solving, both if $P = NP$ or if $P \neq NP$!

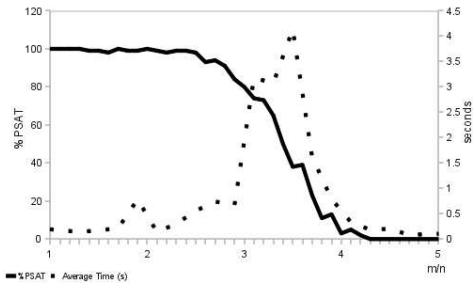

Figure 1: PSAT Phase Transition, for $n = 40$ and $k = 4$.

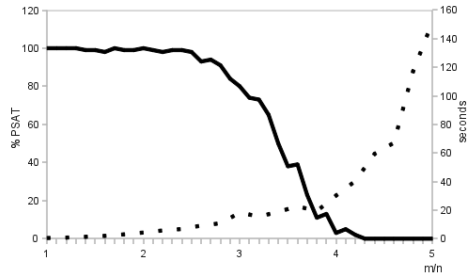

Figure 2: Complexity distribution, implementation of (2)

3.3 Results: A PSAT Phase Transition

A translation of PSAT instances into SAT was implemented in C++, which invoked the zchaff SAT solver [Moskewicz *et al.*, 2001]. We generate uniformly distributed random PSAT instances in normal form with k probability assignments, n variables $y_1, \ldots, y_k, x_{k+1}, \ldots, x_n$ and m 3-SAT clauses. To restrict complexity, we fixed $|\Psi| = k = 4$ which leads to $b_\pi = 2(\lceil (6 \log 6)/2 \rceil - 5) = 6$. We also fixed $n = 40$ and increased the rate m/n in steps of 0.2. For each value of m, we generated 100 instances of PSAT, and computed the percentage of satisfiable instances (%SAT) and the average time of computation in seconds. We then obtained the graphic illustrated in Figure 1.

Figure 1 clearly illustrates a phase transition behaviour, with the transition point around 4. We know that when $k = 0$, PSAT becomes SAT, with phase transition point around 4.3. As k is increased, it is expected that the rate of satisfiable formulas decreases for a given m/n, so the phase transition point is supposed to dislocate to the left, as observed. With more experiments, we could increase k and expect the phase transition point to move even further to the left. However, due to the high complexity of the translation, experiments with $k = 6$ were almost impractical around the transition point. This is why we turn to other algorithms for PSAT solving.

4 A Logic-Algebraic PSAT Solver

Traditional algebraic PSAT solvers extend an exponentially large linear program with $m + k + 1$ new variables (since m clauses in Γ have probability 1) of the form

$$\begin{array}{l}\text{minimize } \langle \textit{objective function of the form } c'\pi \rangle \\ \text{subject to } A\pi = p \text{ and } \pi \geq 0\end{array} \qquad (2)$$

which is solved iteratively by the simplex algorithm; at each iteration step i, Fact 2.2 allows for storing $A_{(i)}$ with $m+k+1$ columns and a *column generation* method is employed in which an *auxiliary problem* generates a column that replaces some column in $A_{(i)}$ and decreases the objective function. Vector c selects only the new variables in π, so $c'\pi = 0$ iff the PSAT problem is satisfiable. Figure 2 presents the results of an implementation of this method for *exactly the same instances* of Figure 1, showing no phase transition and much greater decision times, mainly explained by each formula $\alpha \in \Gamma$ having to be dealt with a restriction $P(\alpha) = 1$.

We propose the following logic-based modifications of that method that inputs a normal form PSAT instance, with

(a) a much smaller basis $A_{(i)}$ of order $k + 1$, by Lemma 2.5;

(b) a different initial feasible solution at step 0;

(c) a new objective function;

(d) a new SAT-based auxiliary problem.

We now detail (b), (c) and (d) and present our algorithm. We first establish some terminology. A matrix A that satisfy conditions (3) is a *feasible solution* for Ψ.

$$\begin{bmatrix} 1 & \cdots & 1 \\ a_{1,1} & \cdots & a_{1,k+1} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,k+1} \end{bmatrix} \cdot \begin{bmatrix} \pi_1 \\ \pi_2 \\ \vdots \\ \pi_{k+1} \end{bmatrix} = \begin{bmatrix} 1 \\ p_1 \\ \vdots \\ p_k \end{bmatrix} \qquad (3)$$

$$a_{i,j} \in \{0,1\}, \qquad A \text{ is non-singular}, \qquad \pi_j \geq 0$$

We will further assume that the input probabilities $p_1, \ldots, p_k$ in (3) are in decreasing order. By Lemma 2.5, $\langle \Gamma, \Psi \rangle$ has a solution iff there is a partial solution A satisfying (3) such that if $\pi_j > 0$ then $a_{1,j}, \ldots, a_{k,j}$ are Γ-consistent valuations for $1 \leq i \leq k, 1 \leq j \leq k + 1$. We usually abuse terminology calling A^j a Γ-consistent column.

4.1 Initial and Further Feasible Solutions

Given $\langle \Gamma, \Psi \rangle$, consider $\langle \varnothing, \Psi \rangle$ obtained by ignoring Γ. As the elements of p are in decreasing order, consider the $\{0,1\}$-matrix $I^* = [a_{i,j}]_{1 \leq i,j \leq k+1}$ where $a_{i,j} = 1$ iff $i \leq j$, that is, I^* is all 1's in and above the diagonal, 0's elsewhere. As p is in decreasing order, I^* satisfies $\langle \varnothing, \Psi \rangle$ and is called a *relaxed solution* for $\langle \Gamma, \Psi \rangle$. Clearly, I^* is a feasible for Ψ.

Example 4.1 The relaxed solution to normal form of Problem 1.2 in Example 2.4 is such that all columns are Γ-consistent, this decides positively Problem 1.2 by Lemma 2.5.

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.67 \\ 0.08 \\ 0.10 \\ 0.15 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.33 \\ 0.25 \\ 0.15 \end{bmatrix} \begin{matrix} \\ y_2 \\ y_1 \\ y_3 \end{matrix} \qquad \square$$

The relaxed solution is the initial feasible solution of our method. Further feasible solutions are obtained by generating new $\{0,1\}$-columns and substituting them into a feasible solution, as shown by the following.

Lemma 4.2 *Let A be a feasible solution satisfying (3) and let $b = [1\ b_1\ \cdots\ b_k]'$ be a $\{0,1\}$-column. Then there always exists a column j such that $A[j := b]$ is a feasible solution.* $\square$

Lemma 4.2 is a well-known fact that comes from the pivoting in the simplex algorithm. Its proof actually gives us an algorithm to compute a new feasible solution from a previous one, so let $merge(A, b)$ be a function that computes it.

Our method moves through feasible solutions, at each step generating a column b that decreases the value of the objective function.

4.2 The Objective Function

In a feasible solution A such that $A\pi = p$ and $\pi \geq 0$, some columns may not be Γ-consistent. Let $J = \{j | A^j \text{ is } \Gamma\text{-inconsistent and } \pi_j > 0\}$; J is the set of column indexes in A corresponding to Γ-inconsistent columns with non-null associated probability; clearly $|J| \leq k+1$. If $J = \varnothing$, we call A a *(total) solution*.

By Lemma 2.5, a positive decision for the PSAT instance $\langle \Gamma, \Psi \rangle$ is reached iff $J = \varnothing$. Thus, a candidate objective function is simply $|J|$. Lemma 4.2 guarantees that if we only generate Γ-consistent columns, $|J|$ never increases. However, it is not guaranteed that, if a solution exists, we can find a path in which $|J|$ decreases at every step s. If this were true, we would have a solution in at most $k+1$ steps, which is unfortunately not the case.

A second candidate objective function is the sum of probabilities of Γ-inconsistent columns, $f = \sum_{j \in J} \pi_j$. Note that f and $|J|$ become 0 at the same time, which occurs iff a positive decision is reached. The simplex algorithm with appropriate column generation ensures that, if there is a solution, it can be obtained with finitely (but possibly exponentially) many steps of non-increasing f-values. We propose a combined objective function $\langle |J|, f \rangle$ ordered lexicographically. In minimising this combined objective function, we first try to minimise the number of Γ-inconsistent columns; if this is not possible, then minimise f, keeping J constant. We thus have the following program

$$\begin{array}{ll} \min & \langle |J|, f \rangle \\ \text{subject to} & A\pi = p, \pi \geq 0, f = \sum_{j \in J} \pi_j \text{ and} \\ & J = \{j | A^j \text{ is } \Gamma\text{-inconsistent}, \pi_j > 0\} \end{array} \quad (4)$$

So the PSAT instance $\langle \Gamma, \Psi \rangle$ associated to program (4) is satisfiable iff the objective function is minimal at $\langle 0, 0 \rangle$.

Assume there is a function $GenerateColumn(A, p, \Gamma)$, presented at Section 4.3, that generates a Γ-consistent column that decreases the objective function, if one exists; otherwise it returns an illegal column of the form $[-1 \ \cdots]$. Algorithm 4.1 presents a method that decides a PSAT instance by solving problem (4).

Algorithm 4.1 starts with a relaxed solution for $\langle \Gamma, \Psi \rangle$ (line 2), and via column generation (line 4) generate another feasible solution (line 6), decreasing the objective function, until either the search fails (line 5) or a solution is found; the latter only occurs with the termination of the loop in lines 3–8, when the objective function reaches $\langle 0, 0 \rangle$.

4.3 SAT-Based Column Generation

A Γ-satisfiable column b that never increases the value of the objective function is obtained by solving at most $k+1$ SAT problems as follows. Consider $x_1, \ldots, x_k$ over $\{0, 1\}$ and

$$a_1 \cdot x_1 + \cdots a_k \cdot x_k \ \ op \ \ c \qquad op \in \{<, \leq, >, \geq, =, \neq\} \quad (5)$$

Algorithm 4.1 Logic-algebraic PSAT solver

Input: A normal form PSAT instance $\langle \Gamma, \Psi \rangle$.
Output: Total solution A; or "No", if unsatisfiable.

1: $p := sortDecrescent(\{1\} \cup \{p_i | P(y_i) = p_i \in \Psi\})$;
2: $A_{(0)} := I^*$; $s := 0$; compute $\langle |J_{(s)}|, f_{(s)} \rangle$;
3: **while** $\langle |J_{(s)}|, f_{(s)} \rangle \neq \langle 0, 0 \rangle$ **do**
4: $\quad b^{(s)} = GenerateColumn(A_{(s)}, p, \Gamma)$;
5: $\quad$ **return** "No" **if** $b_1^{(s)} < 0$; /* instance is unsat */
6: $\quad A_{(s+1)} = merge(A_{(s)}, b^{(s)})$;
7: $\quad$ increment s; compute $\langle |J_{(s)}|, f_{(s)} \rangle$;
8: **end while**
9: **return** $A_{(s)}$; /* PSAT instance is satisfiable */

where $a_1, \ldots, a_k, c \in \mathbb{Q}$; (5) can be seen as a propositional formula LR, in the sense that a valuation $v : x_i \mapsto \{0, 1\}$ *satisfies* LR iff v makes (5) a true condition. Such a formula Δ_{LR} can be obtained from LR in time $O(k)$ [Warners, 1998].

Suppose $1, \ldots, q \leq k+1$ are the Γ-inconsistent columns of feasible solution A. First, we try to eliminate a Γ-inconsistent column A^j associated to $\pi_j > 0$. By linear algebraic manipulation, a new column $b = [1 \ y_1 \ldots y_k]'$ to substitute A^j must satisfy the set of linear inequalities:

$$(LR_{ij}) \quad (A_j^{-1}\pi_i - A_i^{-1}\pi_j)[1 \ y_1 \ldots y_k]' \geq 0, \quad 1 \leq i \leq k+1$$

A substitution b for A^j is obtained by a valuation that satisfies the formula $\Theta_j = \Gamma \wedge \bigwedge_{i=1}^{k+1} \Delta_{LR_{ij}}$. Clearly, if Θ_j is satisfiable, $A[j := b]$ is Γ-consistent and decreases $|J|$.

Suppose every Θ_j is unsatisfiable, $1 \leq j \leq q$, and let $f_\pi = \sum_{j=1}^{q} \pi_j$. A column $b = [1 \ y_1 \ldots y_k]'$ that substitutes some A^j and enforces a non-increasing f_π similarly satisfies

$$(LR_f) \quad \textstyle\sum_{j=1}^{q} A_j^{-1} \cdot [1 \ y_1 \ \cdots \ y_k]' \geq 0$$

A substitution b for A^j is obtained by a valuation that satisfies the formula $\Theta_f = \Gamma \wedge \Delta_{LR_f}$. If Θ_f is satisfiable, $A[j := b]$ is Γ-consistent and f_π never increases.

The algorithm that computes $GenerateColumn(A, p, \Gamma)$ faster in practice is as follows. First, send $\Theta_f(y_1, \ldots, y_k)$ to a SAT-solver. If it is unsatisfiable, return invalid column $[-1 \ 0 \ \ldots \ 0]'$; in this way we have an early failure. Otherwise there is a v_f, $v_f(\Theta_f) = 1$. If $b_f = [1 \ v_f(y_1) \ldots v_f(y_k)]'$ substitutes some Γ-inconsistent column, return b_f, which decreases $|J|$ and maybe f as well. Otherwise store b_f. For each Γ-inconsistent A^j submit $\Theta_j(y_1, \ldots, y_k)$ to a SAT-solver; if it is satisfiable with valuation v, return $b = [1 \ v(y_1) \ldots v(y_k)]'$, decreasing $|J|$. If all Θ_j are unsatisfiable, return b_f, so f does not increase. As the SAT solver is deterministic and generates answers in a fixed order, termination of the whole process is guaranteed.

Example 4.3 Apply Algorithm 4.1 to the presentation of Problem 1.2 in Example 2.4. At step 0, the relaxed solution

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.4 \\ 0 \\ 0 \\ 0.6 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.6 \\ 0.6 \\ 0.6 \end{bmatrix} \begin{matrix} y_1 \\ y_2 \\ y_3 \end{matrix}$$

is such that the two columns on the left are Γ-inconsistent. Column generation tries to substitute the column 1, generating the inequalities $-y_1 + y_2 \geq 0, y_2 - y_3 \leq 0, 3y_1 + 2y_3 \leq 3, y_1 \leq 0$ which, when translated to logic and input to a SAT solver jointly with Γ produces the valuation $y_1 = 0, y_2 = 1, y_3 = 1$. At step 1, we substitute column 1 by $[1\ 0\ 1\ 1]'$,

$$\begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0.4 \\ 0.4 \\ 0 \\ 0.2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.6 \\ 0.6 \\ 0.6 \end{bmatrix}$$

Now only column 2 is Γ-inconsistent; Θ_f produced by column generation is unsatisfiable Θ_f and so is the problem. That is, the drunks are lying. □

4.4 Results: Another PSAT Phase Transition

We implemented Algorithm 4.1 in C++, also employing the zchaff SAT solver, thus obtaining a much faster PSAT solver. We generated uniformly distributed normalised random PSAT instances in normal form with k probability assignments, n variables $y_1, \ldots, y_k, x_{k+1}, \ldots, x_n$ and m 3-SAT clauses. We obtained the graphic illustrated in Figure 3 by fixing $k = 12$, $n = 150$, m/n increases in steps of 0.1 and 200 PSAT instances for each value of m.

Again, Figure 3 displays an unequivocal phase transition behaviour. As predicted, with a larger k the phase transition point has moved to the left at $P_t \approx 3.9$. The phase transition behaviour can be credited to the use of PSAT normal form and to the fact that, unlike in program (2), only Γ-consistent columns were generated in a small basis of size $k+1$. The increase in efficiency is mainly due to the small basis, and also to the use of an efficient SAT solver to and to the smaller size of formulas submitted to the SAT solver than in the reduction algorithm.

By generating large instances over the "flat zones" of Figure 3, we were able to obtain a satisfiable instance with $n = 400, m = 400, k = 50$ and decision time 0.76s; and an unsatisfiable instance with $n = 200, m = 800, k = 100$ and decision time 29.4s. This gives an idea of the capabilities of the method.

5 Conclusion

We have shown that a combination of logic and algebraic techniques can be very fruitful to the study of algorithms of PSAT solvers. The logic part appear to be crucial to reveal a phase transition behaviour in this case.

Further work involves the exploration of efficient PSAT fragments using the algorithm and techniques developed here, and the study of PSAT-based methods and applications that employ the algorithms presented here.

References

[Andersen and Pretolani, 2001] K.A. Andersen and D. Pretolani. Easy cases of probabilistic satisfiability. *AMAI*, 33(1):69–91, 2001.

[Boole, 1854] G. Boole. *An Investigation on the Laws of Thought*. Macmillan, London, 1854. Available at `http://www.gutenberg.org/etext/15114`.

[Cheeseman *et al.*, 1991] P. Cheeseman, R. Kanefsky, and W. M. Taylor. Where the really hard problems are. In *12th IJCAI*, pages 331–337, 1991.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In *3rd ACM Symposium on Theory of Computing (STOC)*, pages 151–158. ACM, 1971.

[Eén and Sörensson, 2003] N. Eén and N. Sörensson. An extensible SAT-solver. In *SAT 2003*, volume 2919 of *LNCS*, pages 502–518. Springer, 2003.

[Garling, 2007] D. J. H. Garling. *Inequalities: A Journey into Linear Analysis*. Cambridge University Press, 2007.

[Gent and Walsh, 1994] I. P. Gent and T. Walsh. The sat phase transition. In *11th European Conference on Artificial Intelligence*, pages 105–109, 1994.

[Georgakopoulos *et al.*, 1988] G. Georgakopoulos, D. Kavvadias, and C. H. Papadimitriou. Probabilistic satisfiability. *J. of Complexity*, 4(1):1–11, 1988.

[Hailperin, 1986] T. Hailperin. *Boole's Logic and Probability*. North-Holland, second edition, 1986.

[Hansen and Jaumard, 2000] P. Hansen and B. Jaumard. Probabilistic satisfiability. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems, vol.5*. Springer, 2000.

[Hansen *et al.*, 1995] P. Hansen, B. Jaumard, G.-B. Nguetsé, and M. P. Aragão. Models and algorithms for probabilistic and bayesian logic. In *IJCAI*, pages 1862–1868, 1995.

[Jaumard *et al.*, 1991] B. Jaumard, P. Hansen, and M.P. Aragão. Column generation methods for probabilistic logic. *INFORMS J. on Computing*, 3(2):135–148, 1991.

[Kavvadias and Papadimitriou, 1990] D. Kavvadias and C. H. Papadimitriou. A linear programming approach to reasoning about probabilities. *AMAI*, 1:189–205, 1990.

[Moskewicz *et al.*, 2001] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an Efficient SAT Solver. In *38th Design Automation Conference*, pages 530–535, 2001.

[Nilsson, 1986] Nils Nilsson. Probabilistic logic. *Artificial Intelligence*, 28(1):71–87, 1986.

[Nilsson, 1993] Nils Nilsson. Probabilistic logic revisited. *Artificial Intelligence*, 59(1–2):39–42, 1993.

[Walley *et al.*, 2004] P. Walley, R. Pelessoni, and P. Vicig. Direct algorithms for checking consistency and making inferences from conditional probability assessments. *J. of Statistical Planning and Inference*, 126(1):119–151, 2004.

[Warners, 1998] J. P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Inf. Process. Lett.*, 68(2):63–69, 1998.

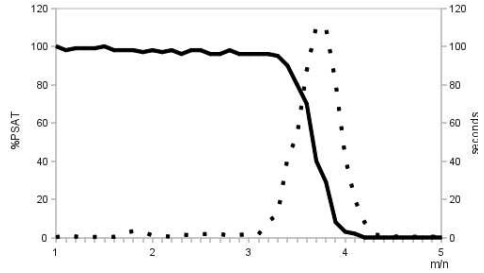

Figure 3: PSAT Phase Transition for Algorithm 4.1