Probably Half True: Probabilistic Satisfiability over Łukasiewicz Infinitely-valued Logic

Marcelo Finger* and Sandro Preto

Department of Computer Science, University of São Paulo, São Paulo, Brazil
{mfinger,spreto}@ime.usp.br

Abstract. We study probabilistic-logic reasoning in a context that allows for "partial truths", focusing on computational and algorithmic properties of non-classical Łukasiewicz Infinitely-valued Probabilistic Logic. In particular, we study the satisfiability of joint probabilistic assignments, which we call LIPSAT. Although the search space is initially infinite, we provide linear algebraic methods that guarantee polynomial size witnesses, placing LIPSAT complexity in the NP-complete class. An exact satisfiability decision algorithm is presented which employs, as a subroutine, the decision problem for Łukasiewicz Infinitely-valued (non probabilistic) logic, that is also an NP-complete problem. We develop implementations of the algorithms described and discuss the empirical presence of a phase transition behavior for those implementations.

1 Introduction

This paper deals with the problem of determining the consistency of probabilistic assertions allowing for "partial truths" considerations. This means that we depart from the classical probabilistic setting and instead employ a many-valued underlying logic. In this way we enlarge our capacity to model situations in which a gradation of truth may be closer to the perceptions of agents involved. We employ Łukasiewicz Infinitely-valued logic as it is one of the best studied many-valued logics, having interesting properties which lead to amenable computational treatment. Notably, it has been shown that foundational properties of probabilistic theory such as de Finetti coherence criteria also applies to Łukasiewicz Infinitely-valued probabilistic theories [27].

We provide theoretical presentation leading to algorithms that decide the satisfiability of probabilistic assertions in which the underlying logic is Łukasiewicz logic with infinity truth values in the interval $[0, 1]$. For that, we employ techniques from linear programming and many-valued logics. In the latter case we need to solve several instances of the satisfiability problem in Łukasiewicz Infinitely-valued logic. This problem has been shown to be NP-complete [25] and there are some implementations discussed in the literature [3], but there are

* Partially supported by Fapesp projs. 2015/21880-4 and 2014/12236-1 and CNPq grant 306582/2014-7.

many implementation options with considerable efficiency differences which we also analyze in this work.

To understand the kind of situation in which our techniques can be applicable consider the following example.

Example 1. Three friends have the habit of going to a bar to watch their soccer team's matches. Staff at the bar claims that at every such match at least two of the friends come to the premises, but if you ask them, they will say that each of them comes to watch at most 60% of the games.

In classical terms, the claims of the staff and of the three friends are in contradiction. In fact, if there are always two of the three friends present at matches, someone must attend to least two-thirds of the team's matches.

However, one may allow someone to arrive for the second half of the match, and consider his attendance only "partially true", say, a truth value of 0.5 in that case. Then it may well be the case that staff and customers are both telling the truth, that is, their claims are jointly satisfiable. □

It turns out that the example above is unsatisfiable in classical probabilistic logic, but it is satisfiable in Łukasiewicz Infinitely-valued Probabilistic logic. In this work we are going to formalize such problems and present techniques and algorithms to solve them.

1.1 Classical and Non-classical Probabilistic Logic

Classical probabilistic logic combines classical propositional inference with classical (discrete) probability theory. The original formulation of such a mix of logic and probability is due to George Boole who, in his seminal work introducing what is now known as Boolean Algebras, already discussed the problem [4]. Among the foundational works on classical probabilistic theory we highlight that provided by de Finetti's notion of coherent probabilities [9, 11].

The decision problem over classical probabilistic logic is called Probabilistic Satisfiability (PSAT). PSAT has been extensively discussed in the literature [18, 20, 28], and has recently received a lot of attention due to the improvements in SAT solving and linear programming techniques, having generated a variety of algorithms, for which the empirical phenomenon of phase-transition is by now established [14, 15].

Łukasiewicz Infinitely-valued Logic is widely used in the literature to model situations that require the notion of "partial truth", seen as a many-valued logic and algebra [8]. A probability theory over such a many-valued context, including a notion of coherent probabilities in line with de Finetti's original work, was developed as a sound basis for non-classical probability theory [27]. The problem of deciding whether a set of probabilistic assignments over Łukasiewicz Infinitely-valued Logic is coherent was shown to be NP-complete by [6]. It is the goal of this paper to explore equivalent formulations and algorithmic ways to solve this problem and study the existence of a phase transition in its empirical behavior.

The rest of this paper is organized as follows. In Section 2 we describe the notions pertaining Łukasiewicz Infinitely-valued Logic and Łukasiewicz Infinitely-valued Probabilistic Logic and the notion of coherent probability over such logic. In Section 3 we study the theoretical relationship between linear algebraic methods and the solution of the LIPSAT problem. In Section 4 we develop a column generation algorithm for LIPSAT solving and show its correctness. Finally, we discuss implementation issues and the phase transition behavior of the solvers in Section 5.

Due to space restrictions, proofs of some results have been omitted. Source code of the solvers developed are publicly available.

2 Preliminaries

Łukasiewicz Infinitely-valued Logic ($Ł_\infty$) is arguably one of the best studied many-valued logics [8]. It has several interesting properties, such as a truth-functional semantics that is continuous, having classical logic as a limit case and possessing well developed proof-theoretical and algebraic presentations. The semantics of $Ł_\infty$-formulas represent all piecewise linear functions and only those [23, 26].

The basic $Ł_\infty$-language is built from a countable set of propositional symbols $\mathbb{P}$, and disjunction ($\oplus$) and negation ($\neg$) operators. For the semantics, define a $Ł_\infty$-valuation $v : \mathbb{P} \to [0, 1]$, which maps propositional symbols to a value in the rational interval $[0, 1]$. Then v is extended to all $Ł_\infty$-formulas as follows

$$v(\varphi \oplus \psi) = \min(1, v(\varphi) + v(\psi))$$
$$v(\neg\varphi) = 1 - v(\varphi)$$

From those operations one usually derives the following:

Conjunction: $\varphi \odot \psi =_{\text{def}} \neg(\neg\varphi \oplus \neg\psi)$	$v(\varphi \odot \psi) = \max(0, v(\varphi) + v(\psi) - 1)$
Implication: $\varphi \to \psi =_{\text{def}} \neg\varphi \oplus \psi$	$v(\varphi \to \psi) = \min(1, 1 - v(\varphi) + v(\psi))$
Maximum: $\varphi \vee \psi =_{\text{def}} \neg(\neg\varphi \oplus \psi) \oplus \psi$	$v(\varphi \vee \psi) = \max(v(\varphi), v(\psi))$
Minimum: $\varphi \wedge \psi =_{\text{def}} \neg(\neg\varphi \vee \neg\psi)$	$v(\varphi \wedge \psi) = \min(v(\varphi), v(\psi))$
Bi-implication: $\varphi \leftrightarrow \psi =_{\text{def}} (\varphi \to \psi) \wedge (\psi \to \varphi)$	$v(\varphi \leftrightarrow \psi) = 1 - \lvert v(\varphi) - v(\psi)\rvert$

A formula φ is $Ł_\infty$-*valid* if $v(\varphi) = 1$ for every valuation v. A formula φ is $Ł_\infty$-*satisfiable* if there exists a v such that $v(\varphi) = 1$; otherwise it is $Ł_\infty$-*unsatisfiable*. A set of formulas Φ is satisfiable if there exists a v such that $v(\varphi) = 1$ for all $\varphi \in \Phi$. Note that $v(\varphi \to \psi) = 1$ iff $v(\varphi) \leq v(\psi)$; similarly, $v(\varphi \leftrightarrow \psi) = 1$ iff $v(\varphi) = v(\psi)$.

$Ł_\infty$ also serves as a basis for a well-founded non-classical probability theory [24]. Define a *convex combination* over a finite set of valuations $v_1, \cdots, v_m$ as a function on formulas into $[0, 1]$ such that

$$C(\varphi) = \lambda_1 v_1(\varphi) + \cdots + \lambda_m v_m(\varphi) \tag{1}$$

where $\lambda_i \geq 0$ and $\sum_{i=1}^{m} \lambda_i = 1$. So a $Ł_\infty$-probability distribution $\lambda = [\lambda_1, \cdots, \lambda_m]$ is a set of coefficients that form the convex combination of $Ł_\infty$-valuations. To distinguish $Ł_\infty$-probabilities from classical ones, we use the notation $C(\cdot)$, following [24]; it is important to note that C is defined over *any* ***finite*** set of valuations[1]. Note that classical discrete probabilities are also convex combinations of $\{0,1\}$-valuations.

This notion of probability associates non-zero values only to a finite number of $Ł_\infty$-valuations; thus the notion of $Ł_\infty$-probability is intrinsically discrete. As there are infinitely many possible $Ł_\infty$-valuations, the remaining ones are assumed to be zero. In this work we are interested in deciding the existence of convex combinations of the form (1) given a set of constraints. So, in theory, the search space is infinite.

It follows immediately from this definition that $C(\alpha) = 1$ if there is a convex combination over $v_1, \cdots, v_m$ where $v_i(\alpha) = 1, 1 \leq i \leq m$.

Lemma 1. *$C(\alpha \to \beta) = 1$ iff $C(\alpha) \leq C(\beta)$.* □

Lemma 1 is a direct consequence from the fact that $v(\varphi \to \psi) = 1$ iff $v(\varphi) \leq v(\psi)$.

We define a Łukasiewicz Infinitely-valued Probabilistic (LIP) assignment as an expression of the form

$$\Sigma = \left\{ C(\alpha_i) = q_i \mid q_i \in [0,1], 1 \leq i \leq k \right\}.$$

As a foundational view of probabilities, it is possible to define a coherence criterion over LIP-assignments, in analogy to the de Finetti classical notion of coherent assignment of probabilities [10,11]. Thus, define the $Ł_\infty$-coherence of a LIP-assignment $\{C(\alpha_i) = q_i \mid 1 \leq i \leq k\}$ in terms of a bet between two players, Alice the bookmaker and Bob the bettor. The outcome on which the players bet is a $Ł_\infty$-valuation describing an actual "possible world". For each formula α_i, Alice states her betting odd $C(\alpha_i) = q_i \in [0,1]$ and Bob chooses a "stake" $\sigma_i \in \mathbb{Q}$; Bob pays Alice $\sum_{i=1}^{k} \sigma_i \cdot C(\alpha_i)$ with the promise that Alice will pay back $\sum_{i=1}^{k} \sigma_i \cdot v(\alpha_i)$ if the outcome is the possible world (or valuation) v. As in the classical case, the chosen stake σ_i is allowed to be negative, in which case Alice pays Bob $|\sigma_i| \cdot C(\alpha_i)$ and gets back $|\sigma_i| \cdot v(\alpha_i)$ if the world turns out to be v. Alice's total balance in the bet is

$$\sum_{i=1}^{k} \sigma_i (C(\alpha_i) - v(\alpha_i)).$$

We say that there is a *LIP-Dutch Book* against Alice's LIP-assignment if there is a choice of stakes σ_i such that, for every possible outcome v, Alice's total balance is always negative, indicating a bad choice of betting odds made by Alice.

[1] Thus C is more restrictive than the full class of states of an MV-algebra, in the sense of [24], which will not be discussed here.

Definition 1. Given a probability assignment to propositional formulas $\{C(\alpha_i) = q_i \mid 1 \leq i \leq k\}$, the LIP-assignment is *coherent* if there are no Dutch Books against it.

While the coherence of an assignment provides a foundational view to deal with Ł_∞-probabilities, a more computational view is possible, based on the satisfiability of assignments. Such a view will allow a more operational way of dealing with Ł_∞-probabilistic assignments.

Definition 2. A LIP-assignment is *satisfiable* if there exists a convex combination C and a set of valuations that jointly verifies all restrictions in it.

Example 2. Consider again Example 1, let x_1, x_2, x_3 be variables representing the presence at the bar of each of the three friends. An Ł_∞-valuation assigns to each variable a value in $[0, 1]$. The probabilistic constraint expressing that each friend comes at most 60% of the games can be expressed as

$$C(x_1) = C(x_2) = C(x_3) \leq 0.6, \tag{*}$$

and the fact that at least two of them are present is expressed by the constraints

$$C(x_1 \oplus x_2) = C(x_1 \oplus x_3) = C(x_2 \oplus x_3) = 1 \tag{**}$$

which means that no two of them are simultaneously absent. There are infinitely many ways of obtaining a convex combination of Ł_∞-valuations that satisfy all six conditions, the simplest of which is achieved with a single Ł_∞-valuation v, $v(x_1) = v(x_2) = v(x_3) = 0.6$; in fact, $v(x_1 \oplus x_2) = v(x_1 \oplus x_3) = v(x_2 \oplus x_3) = \min(1, 0.6 + 0.6) = 1$, so we can attribute 100% of probability mass to v.

A similar result can be obtained with three "classical" valuations $v_i(x_i) = 0, v_i(x_j) = v_i(x_k) = 1$, for pair-wise distinct $i, j, k \in \{1, 2, 3\}$ and a fourth valuation $v_4(x_1) = v_4(x_2) = v_4(x_3) = 0.5$. Note all four valuation satisfy the formulas in (**). The convex valuation assigns probability 0.2 to v_1, v_2, v_3 and 0.4 to v_4, satisfying all constraints (*) and (**). □

The following result is the characterization of coherence for Łukasiewicz Infinitely-valued Probabilistic Logic.

Proposition 1 (Mundici [27]). *Given a LIP-assignment $\Sigma = \{C(\alpha_i) = q_i \mid 1 \leq i \leq k\}$, the following are equivalent:*

(a) Σ is a coherent LIP-assignment.
(b) Σ is a satisfiable LIP-assignment.

Proposition 1 asserts that deciding LIP coherence is the same as determining LIP-assignment satisfiability, which we call *LIPSAT*. This result is the Ł_∞ analogous to de Finetti's characterization of coherence of classical probabilistic assignment as equivalent to the *probabilistic satisfiability* (PSAT) of the assignment, which was shown to be an NP-complete problem that can be solved using linear algebraic methods [18,28]. It has also been shown by Bova and Flaminio [6] that deciding the coherence of a LIP-assignment is also an NP-complete problem.

Our goal here is to explore efficient ways to decide the coherence of LIP-assignments. In analogy to the algorithms used for deciding PSAT [14, 15], we explore a linear algebraic formulation of the problem.

3 Algebraic Formulation of LIPSAT

We consider an *extended* version of LIP-assignments of the form

$$\Sigma = \left\{ C(\alpha_i) \bowtie_i q_i \mid q_i \in [0,1], \bowtie_i \in \{=, \leq, \geq\}, 1 \leq i \leq k \right\}. \tag{2}$$

Extended LIP-assignments may have both inequalities and equalities. Such an assignment is satisfiable if there is a $\mathcal{L}_\infty$-probability distribution λ that verify all inequalities and equalities in it.

Given an extended LIP-assignment $\Sigma = \{C(\alpha_i) \bowtie_i q_i\}$, let $q = (q_1, \ldots, q_k)'$ be the vector of probabilities in Σ, $\bowtie$ the "vector" of (in)equality symbols. Suppose we are given $\mathcal{L}_\infty$-valuations $v_1, \ldots, v_m$ and let $\lambda = (\lambda_1, \ldots, \lambda_m)'$ be a vector of convex weights. Consider the $k \times m$ matrix $A = [a_{ij}]$ where $a_{ij} = v_j(\alpha_i)$. Then an extended LIP-assignment of the form (2) is satisfiable if there are $v_1, \ldots, v_m$ and λ such that the set of algebraic constrains (3) has a solution:

$$\begin{aligned} A \cdot \lambda &\bowtie q \\ \textstyle\sum \lambda_j &= 1 \\ \lambda &\geq 0 \end{aligned} \tag{3}$$

The condition $\sum \lambda_j = 1$ can be incorporated as an all-1 row $k+1$ in matrix A, $q = (q_1, \ldots, q_k, 1)'$ and $\bowtie_{k+1}$ is "=". Note that the number m of columns in A is in principle unbounded, but the following consequence of Carathéodory's Theorem [13] yields that a if (3) has a solution, than it has a "small" solution.

Proposition 2 (Carathéodory's Theorem for LIP). *If a set of restrictions of the form (3) has a solution, then it has a solution in which at most $k+1$ elements of λ are non-zero.* □

Given the algebraic formulation in (3), NP-completeness of LIP satisfiability, originally shown by Bova and Flaminio [6], can be seen as a direct corollary of Proposition 2. In fact, that LIPSAT is NP-hard comes from the fact that when all $q_i = 1$, the problem becomes $\mathcal{L}_\infty$-satisfiability, which is NP-complete [25]; and Proposition 2 asserts the existence of a polynomial size witness for LIPSAT, hence is in NP; so LIPSAT is NP-complete. See Corollary 1.

However, to apply linear algebraic methods to efficiently solve LIPSAT, first we need to provide a normal form for it.

3.1 A Normal Form for LIP-Assignments

An extended assignment may seem more expressive than regular LIP-assignments, but we show that no expressivity is gained by this extension. In fact, we define a *normal form* LIP assignment as a pair $\langle \Gamma, \Theta \rangle$, where Γ is a set of $\mathcal{L}_\infty$-formulas and Θ is a set of LIP restrictions over propositional symbols of the form

$$\Theta = \left\{ C(p_i) = q_i \mid q_i \in [0,1], p_i \in \mathbb{P}, 1 \leq i \leq k \right\}. \tag{4}$$

The formulas $\gamma \in \Gamma$ represent LIP-assignments of the form $C(\gamma) = 1$, that is, a set of hard constrains in the form of $Ł_\infty$-formulas which must be satisfied by all valuations in the convex combination that compose a $Ł_\infty$-probability distribution.

A normal form assignment $\langle \Gamma, \Theta \rangle$ is satisfiable if there are $Ł_\infty$-valuations $v_1, \cdots, v_m$ such that $v_i(\gamma) = 1$ for every $\gamma \in \Gamma$ and there is a $Ł_\infty$-probability distribution $\lambda_1, \cdots, \lambda_m$, such that for each assignment $C(p_i) = q_i \in \Theta$, $\sum_{j=1}^{m} \lambda_j \cdot v_j(p_i) = q_i$.

The satisfiability of extended LIP-assignments reduces to that of normal form ones, as follows.

Theorem 1 (Atomic Normal Form). *For every extended LIP-assignments Σ there exists a normal form LIP-assignment $\langle \Gamma, \Theta \rangle$ such that Σ is a satisfiable iff $\langle \Gamma, \Theta \rangle$ is; the normal form assignment can be built from Σ in polynomial time.*

Proof. Start with $\Gamma = \Theta = \varnothing$. Given Σ, first transform it into Σ' in which all assignments are of the form $C(\alpha) \leq p$; for that, if Σ contains a constraint of the form $C(\alpha) \bowtie 1$, $\bowtie \in \{=, \geq\}$ (resp. $C(\alpha) = 0, C(\alpha) \leq 0$) we insert α (resp. $\neg\alpha$) in Γ and do not insert the constraint in Σ'. If $C(\alpha) = q \in \Sigma$ we insert $C(\alpha) \leq q$ and $C(\alpha) \geq q$ in Σ'. Then all assignments of the latter form are transformed into $C(\neg\alpha) \leq 1 - q$. All transformation steps preserve satisfiability and can be made in linear time, so $\Gamma \cup \Sigma'$ is satisfiable iff Σ is.

For every $C(\alpha_i) \leq q_i \in \Sigma', 0 < q_i < 1$, consider a *new* symbol y_i; insert $\alpha_i \to y_i$ in Γ and $C(y_i) = q_i$ in Θ. Clearly $\langle \Gamma, \Theta \rangle$ is in normal form and is obtained in linear time. The fact that Σ is satisfiable iff $\langle \Gamma, \Theta \rangle$ is follows from Lemma 1. □

Example 3. Note that the formalization presented in Example 2 is already in normal form, witnessing that this format is quite a natural one to formulate LIP-assignments. □

3.2 Algebraic Methods for Normal Form LIP-Assignments

For the rest of this paper we assume that LIP-assignments are in normal form. Here we explore their algebraic structure as it allows for the interaction between a LIP problem Θ and a $Ł_\infty$-SAT instance Γ, such that solutions satisfying the normal form assignment can be seen as probabilistic solutions to Θ constrained by the SAT instance Γ.

Furthermore, to construct a convex combination of the form (1) we will *only* consider Γ-satisfiable valuations. Given a LIP-assignment $\langle \Gamma, \Theta = \{C(p_i) = q_i\} \rangle$, a partial assignment v over $p_i, \ldots, p_k$ is *Γ-satisfiable* if it can be extended to a full assignment that satisfies all formulas in Γ. Let q be a $k+1$ dimensional vector $(q_1, \ldots, q_k, 1)'$. The following is a direct consequence of Theorem 1.

Lemma 2. *A normal form instance $\langle \Gamma, \Theta \rangle$ is satisfiable iff there is a $(k+1) \times (k+1)$-matrix A_Θ, such that all of its columns are Γ-satisfiable, A_Θ last row is all 1's, and $A_\Theta \lambda = q$ has a solution $\lambda \geq 0$.*

Lemma 2 leads to a linear algebraic PSAT solving method as follows. Let V be the set of partial valuations over the symbols in Θ; consider a $|V|$-dimensional vector c such that

$$c_j = \begin{cases} 0, \, v_j \in V \text{ is } \Gamma\text{-satisfiable} \\ 1, \text{ otherwise} \end{cases} \tag{5}$$

The vector c is a boolean "cost" associated to each partial valuation $v_j \in V$, such that the cost is 1 iff v_j is Γ-unsatisfiable. Consider a matrix A whose columns are the valuations in V. Now consider linear program (6) which aims at minimizing that cost, weighted by the corresponding probability value λ_j.

$$\begin{array}{ll} \min & c' \cdot \lambda \\ \text{subject to} & A \cdot \lambda = q \\ & \sum \lambda_i = 1 \\ & \lambda \geq 0 \\ & A\text{'s columns are partial valuations in } V \end{array} \tag{6}$$

Theorem 2. *A normal form instance $\langle \Gamma, \Theta = \{C(p_i) = q_i \mid 1 \leq i \leq k\}\rangle$ is satisfiable iff linear program (6) reaches a minimal solution $c' \cdot \lambda = 0$. Furthermore, if there is a solution, then there is a solution in which at most $k+1$ values of λ are not null.*

Proof. If linear program (6) reaches 0, we obtain $v_1, \ldots, v_m$ by selecting only the Γ-satisfiable columns A_j for which $\lambda_j > 0$, obtaining a convex combination satisfying Θ. So $\langle \Gamma, \Theta \rangle$ is satisfiable. Conversely, if $\langle \Gamma, \Theta \rangle$ is satisfiable, by Lemma 2 there exists a matrix A_Θ such that all of its columns are Γ-satisfiable partial valuations and $A_\Theta \cdot \lambda = q$; clearly A_Θ is a submatrix of A; make $\lambda_j = 0$ when A_j is a A_Θ column and thus $c' \cdot \lambda = 0$. Again by Lemma 2, A_Θ has at most $k+1$ columns so at most $k+1$ values of λ are not null. □

The following consequence of Theorem 2 was originally proven by Bova and Flaminio [6] as the decision of LIP-assignment coherence, which is equivalent to LIP satisfiability by Proposition 1.

Corollary 1 (LIPSAT Complexity). *The problem of deciding the satisfiability of a LIP-assignment is NP-complete.*

Despite the fact that solvable linear programs of the form (6) always have polynomial size solutions, with respect to the size of the corresponding normal form LIP-assignment, the elements of linear program itself (6) may be exponentially large, rendering the explicit representation of matrix A impractical. In the following, we present an algorithmic technique that avoids that exponential explosion.

4 A LIPSAT-solving Algorithm

Based on the results of the previous section we are going to present an algorithm employing a linear programming technique called *column generation* [21, 22],

to obtain a decision procedure for Łukasiewicz Infinitely-valued Probabilistic Logic, which we call *LIPSAT solving*. This algorithm solves the potentially large linear program (6) without explicitly representing all columns and making use of an extended solver for $\L_\infty$-satisfiability as an auxiliary procedure to generate columns.

To avoid the exponential blow of the size of matrix in (6), the algorithm basic idea is to employ the simplex algorithm [2,29] over a normal form LIP-assignment $\langle \Gamma, \Theta \rangle$, coupled with a strategy that generates cost decreasing columns without explicitly representing the full matrix A. In this process, we start with a *feasible solution*, which may contain several $\L_\infty$ Γ-unsatisfiable columns. We minimize the cost function consisting of the sum of the probabilities associated to Γ-unsatisfiable columns, such that when it reaches zero, we know that the problem is satisfiable; if no column can be generated and the minimum achieved is bigger than zero, a negative decision is reached.

The general strategy employed here is similar to that employed to PSAT solving [14, 15], but the column generation algorithm is considerably distinct and requires an extension of $\L_\infty$ decision procedure.

From the input $\langle \Gamma, \Theta \rangle$, we implicitly obtain an unbounded matrix A and explicit obtain the vector of probabilities q mentioned in (6). The basic idea of the simplex algorithm is to move from one feasible solution to another one with a decreasing cost. The feasible solution consists of a square matrix B, called the basis, whose columns are extracted from the unbounded matrix A. The pair $\langle B, \lambda \rangle$ consisting of the basis B and a LIP probability distribution λ is a *feasible solution* if $B \cdot \lambda = q$ and $\lambda \geq 0$. We assume that $q_{k+1} = 1$ such that the last line of B we will force $\sum_G \lambda_j = 1$, where G is the set of B columns that are Γ-satisfiable. Each step of the algorithm replaces one column of the feasible solution $\langle B^{(s-1)}, \lambda^{(s-1)} \rangle$ at step $s-1$ obtaining a new feasible solution $\langle B^{(s)}, \lambda^{(s)} \rangle$. The cost vector $c^{(s)}$ is a $\{0,1\}$ vector such that $c_j^{(s)} = 1$ iff B_j is Γ-unsatisfiable. The column generation and substitution is designed such that the total cost is never increasing, that is $c^{(s)\prime} \cdot \lambda^{(s)} \leq c^{(s-1)\prime} \cdot \lambda^{(s-1)}$.

Algorithm 4.1 presents the top level LIPSAT decision procedure. Lines 1–3 present the initialization of the algorithm. We assume the vector q is in ascending order. Let the D_{k+1} be a $k+1$ square matrix in which the elements on the diagonal and below are 1 and all the others are 0. At the initial step we make $B^{(0)} = D_{k+1}$, this forces $\lambda_1^{(0)} = q_1 \geq 0$, $\lambda_{j+1}^{(0)} = q_{j+1} - q_j \geq 0, 1 \leq j \leq k$; and $c^{(0)} = [c_1 \cdots c_{k+1}]'$, where $c_k = 0$ if column j in $B^{(0)}$ is Γ-satisfiable; otherwise $c_j = 1$. Thus the initial state $s = 0$ is a feasible solution.

Algorithm 4.1 main loop covers lines 5–12 which contains the column generation strategy described above. Column generation occurs at beginning of the loop (line 5) which we are going to detail bellow. If column generation fails the process ends with failure in line 7. Otherwise a column is removed and the generated column is inserted in a process we called *merge* at line 9. The loop ends successfully when the objective function (total cost) $c^{(s)\prime} \cdot \lambda^{(s)}$ reaches zero and the algorithm outputs a probability distribution λ and the set of Γ-satisfiable columns in B, at line 13.

Algorithm 4.1 LIPSAT-CG: a LIPSAT solver via Column Generation

Input: A normal form LIPSAT instance $\langle \Gamma, \Theta \rangle$.
Output: No, if $\langle \Gamma, \Theta \rangle$ is unsatisfiable. Or a solution $\langle B, \lambda \rangle$ that minimizes (6).

$q := [\{q_i \mid C(p_i) = q_i \in \Theta, 1 \leq i \leq k\} \cup \{1\}]$ in ascending order;
$B^{(0)} := D_{k+1}$;
$s := 0$, $\lambda^{(s)} = (B^{(0)})^{-1} \cdot q$ and $c^{(s)} = [c_1 \cdots c_{k+1}]'$;
while $c^{(s)\prime} \cdot \lambda^{(s)} \neq 0$ **do**
  $y^{(s)} = GenerateColumn(B^{(s)}, \Gamma, c^{(s)})$;
  if $y^{(s)}$ column generation failed **then**
    return No; \\ LIPSAT instance is unsatisfiable
  else
    $B^{(s+1)} = merge(B^{(s)}, b^{(s)})$
    $s{+}{+}$, recompute $\lambda^{(s)}$ and $c^{(s)}$;
  end if
end while
return $\langle B^{(s)}, \lambda(s) \rangle$; \\ LIPSAT instance is satisfiable

The procedure *merge* is part of the simplex method which guarantees that given a $k+1$ column y and a feasible solution $\langle B, \lambda \rangle$ there always exists a column j in B such that if $B[j := y]$ is obtained from B by replacing column j with y, then there is λ' such that $\langle B[j := y], \lambda' \rangle$ is a feasible solution.

Lemma 3. *Let $\langle B, \lambda \rangle$ be a feasible solution of (6), such that B is non-singular, and let y be a column. Then there always exists a column j such that $\langle B[j := y], \lambda' \rangle$ is a non-singular feasible solution.*

Lemma 3 guarantees the existence of a column which may not be unique and further selection heuristic is necessary; in our implementation we give priority to remove columns which are associated to probability zero on a left-to-right order.

We now describe the column generation method, which takes as input the current basis B, the current cost c, and the L_∞ restrictions Γ; the output is a column y, if it exists, otherwise it signals **No**. The basic idea for column generation is the property of the simplex algorithm called the *reduced cost* of inserting a column y with cost c_y in the basis. The reduced cost is given by equation

$$r_y = c_y - c' B^{-1} y \tag{7}$$

and the simplex method guarantees that the objective function is non increasing if $r_y \leq 0$. Furthermore the generation method is such that the column y is Γ-satisfiable so that $c_y = 0$. We thus obtain

$$c' B^{-1} y \geq 0 \tag{8}$$

which is an inequality on the elements of y. To force λ to be a probability distribution, we make $y_{k+1} = 1$, the remaining elements y_i are valuations of the variables in Θ, so that we are searching for solution to (8) such that $0 \leq$

$y_i \leq 1, 1 \leq i \leq k$. To finally obtain column y we must extend a $Ł_\infty$-solver that generates valuations satisfying Γ so that it also respects the linear restriction (8). In fact this is not an expressive extension of $Ł_\infty$ as the McNaughton property guarantees that (8) is equivalent to some $Ł_\infty$-formula on variables $y_1, \ldots, y_k$ [8]. In practice, we tested two ways of obtaining a joint solver for Γ and (8):

- Employ an SMT (SAT modulo theories) solver that can handle linear algebraic equations such as (8) and the linear inequalities generated by the $Ł_\infty$-semantics. $Ł_\infty$-solvers based on SMT can be found in the literature, see [3];
- Use a MIP (mixed integer programming) solver that encodes $Ł_\infty$-semantics. Equation (8) is simply a new linear restriction to be dealt by the MIP solver. $Ł_\infty$-solvers based on MIP solvers have been proposed by [19].

In both cases, the restrictions posed by Γ-formulas and (8) are jointly handled by the semantics of the underlying solver. Note that both MIP solving and SMT(linear algebra) are NP-complete problems. We have thus the following result.

Lemma 4. *There are algorithmic solutions to the problem of jointly satisfying $Ł_\infty$-formulas and inequalities with common variables.*

We now deal with the problem of termination. Column generation as above guarantees that the cost is never increasing. The simplex method ensures that a solvable problem always terminates if the costs always decrease, we are left with the problem of guaranteeing that the objective function does not become stationary. This is guaranteed in the implementation by a column selection strategy that respects *Bland's Rule* and also by plateau escaping strategies such as *Tabu search* [2, 29].

Lemma 5. *There are column selection strategies that guarantee that the Algorithm 4.1 always terminates.*

We know that there are no column selection heuristics that guarantee that the simplex method terminates in a polynomial number of steps. However, the simplex method performs very well in most practical cases and its average complexity is known to be polynomial [5].

By placing all the results above together we can state the correction of Algorithm 4.1.

Theorem 3. *Consider the output of Algorithm 4.1 with normal form input $\langle \Gamma, \Theta \rangle$. If the algorithm succeeds with solution $\langle B, \lambda \rangle$, then the input problem is satisfiable with distribution λ over the valuations which are columns of B. If the program outputs no, then the input problem is unsatisfiable. Furthermore, there are column selection strategies that guarantee termination.*

Proof. Lemma 3 guarantees that all steps $\langle B^{(s)}, \lambda(s) \rangle$ is a feasible solution to the problem. If Algorithm 4.1 terminates with success, than cost zero has been

reached, so by Theorem 2 the input problem is satisfiable. On the other hand, if column generation fails, this fails with a positive cost, this means there are no Γ-satisfiable columns that can reduce the cost. So, the problem in unsatisfiable. Finally, a suitable column selection strategy by Lemma 5 guarantees termination.

Example 4. We show the steps for the solution of Example 2. Initially, we have

$$q = \begin{bmatrix} 0.6 \\ 0.6 \\ 0.6 \\ 1 \end{bmatrix}, B^{(0)} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 \end{bmatrix}, \lambda^{(0)} = (B^{(0)})^{-1} \cdot q = \begin{bmatrix} 0.6 \\ 0 \\ 0 \\ 0.4 \end{bmatrix}, c^{(0)} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}.$$

$c^{(0)}$ expresses that the first two columns of $B^{(0)}$ are Γ-satisfiable. The total cost $\text{cost}^{(0)} = c^{(0)\prime} \cdot \lambda^{(0)} = 0.4$. At this point, column $y^{(1)}$ is generated substituting $B^{(0)}$'s column 3 in the *merge* procedure:

$$y^{(1)} = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, B^{(1)} = \begin{bmatrix} 1 & 0 & \mathbf{1} & 0 \\ 1 & 1 & \mathbf{0} & 0 \\ 1 & 1 & \mathbf{1} & 0 \\ 1 & 1 & \mathbf{1} & 1 \end{bmatrix}, \lambda^{(1)} = \begin{bmatrix} 0.6 \\ 0 \\ 0 \\ 0.4 \end{bmatrix}, c^{(1)} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{0} \\ 1 \end{bmatrix}.$$

$\text{cost}^{(1)} = 0.4$. Again, column generation provides $y^{(2)}$ in place of column 1:

$$y^{(2)} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, B^{(2)} = \begin{bmatrix} \mathbf{1} & 0 & \mathit{1} & 0 \\ \mathbf{1} & 1 & \mathit{0} & 0 \\ \mathbf{0} & 1 & \mathit{1} & 0 \\ \mathbf{1} & 1 & \mathit{1} & 1 \end{bmatrix}, \lambda^{(2)} = \begin{bmatrix} 0.3 \\ 0.3 \\ 0.3 \\ 0.1 \end{bmatrix}, c^{(2)} = \begin{bmatrix} \mathbf{0} \\ 0 \\ \mathit{0} \\ 1 \end{bmatrix}.$$

$\text{cost}^{(2)} = 0.1$. Finally, column generation provides $y^{(3)}$ in place of column 4:

$$y^{(3)} = \begin{bmatrix} 0.5 \\ 0.5 \\ 0.5 \\ 1 \end{bmatrix}, B^{(3)} = \begin{bmatrix} \mathit{1} & 0 & \mathit{1} & \mathbf{0.5} \\ \mathit{1} & 1 & \mathit{0} & \mathbf{0.5} \\ \mathit{0} & 1 & \mathit{1} & \mathbf{0.5} \\ \mathit{1} & 1 & \mathit{1} & \mathbf{1} \end{bmatrix}, \lambda^{(3)} = \begin{bmatrix} 0.2 \\ 0.2 \\ 0.2 \\ 0.4 \end{bmatrix}, c^{(3)} = \begin{bmatrix} \mathit{0} \\ 0 \\ \mathit{0} \\ \mathbf{0} \end{bmatrix}.$$

$\text{cost}^{(3)} = 0$, so that the problem is satisfiable with solution $\langle B^{(3)}, \lambda^{(3)} \rangle$. □

5 Implementation and Results

The mere development of a solver over a handful of tests is, in our opinion, an insufficient way to assess the quality of an implementation. In this section we explore a qualitative behavior of solvers, called phase transition, over a large class of randomly generated formulas.

A decision problem displays a *phase transition* when there is an ordering of classes of problems that presents a transition from predominantly satisfiable instances (answer "yes") to predominantly unsatisfiable instances (answer "no"), which is called *a first order phase transition*. Furthermore the decision problem displays a peak in average execution time around the middle of that transition in which fifty percent of answers "yes" and fifty percent of answers "no", which is called a second order phase transition, following the terminology of mechanical statistics [7].

It is conjectured that there is a (second order) phase transition for every NP-complete decision problem [7]. Empirical phase transition behavior are well

established for classical SAT [17] and PSAT [14], among many others. In fact, the empirical verification of phase transition for solvers of an NP-complete problem can be perceived as a quality test for its implementation. In the following we present our empirical results, searching for a phase transition behavior, for $Ł_\infty$-solvers and LIPSAT solver.

5.1 Phase Transition for $Ł_\infty$-Solvers

In a classical setting one usually employs 3-SAT format to obtain a phase transition diagram. The randomly generated formulas are clauses with three literals each, the number of symbols n is fixed and the rate between the number n of clauses and the rate $\frac{m}{n}$ is used as the control parameter, where m is the number of clauses. In classical 3-SAT, the shape of the curve and the phase transition point is maintained when n is changed. Unfortunately for $Ł_\infty$ Logic there is no clausal normal form. So instead we employ a set of formulas which are used by [3] consisting of

$$l_1 \oplus l_2 \oplus l_3 \tag{9}$$

$$\neg(l_4 \oplus l_5) \oplus l_6 \tag{10}$$

where l_i are literals (negated or non-negated symbols). The generation of the formulas is parametrized by the number n of propositional symbols and the number m of formulas, which define the class of randomly generated formulas. Following [3], formulas are generated as follows: 70% of formulas are of format (9) and 30% of the format (10). Each literal is randomly chosen from the n possible symbols with equal probability, then there is a 50% chance of being a positive or negative literal.

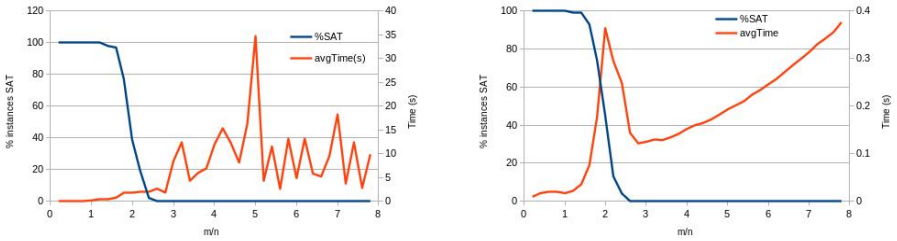
Two implementations were developed using publicly available open source software[2]:

- a C++-implementation using the C++ interface to the YICES SMT(LA) solver [12];
- a C++-implementation using the C++ interface to the SCIP MIP solver [1].

For each implementation, the experiment proceed as follows: with a fixed $n = 100$ we varied the value of m such that the rate $\frac{m}{n}$ varies from 0.2 to 8 in 0.2 steps. For each pair $\langle n, m\rangle$ we construct a set of 100 randomly generated formulas as described above. And for each set we compute the percentage of $Ł_\infty$-satisfiable formulas and the average decision time (user time).

All the experiments in this section were run on a UNIX machine with a i7-6900K CPU @ 3.20GHz with 16 processors. The results of the experiments using two $Ł_\infty$-solvers are shown in Fig. 1. In Fig. 1a we see the results of an SMT(LA) $Ł_\infty$-solver using YICES which presents a first-order phase transition from SAT to unSAT with a middle point occurring at rate $\frac{m}{n} \approx 2$; however the average

[2] The source code for all experiments under license GPLv3 are publicly available at http://lipsat.sourceforge.net.



(a) Based on SMT(LA) using YICES (b) Based on MIP solver using SCIP

Fig. 1: $\mathcal{L}_\infty$-solvers performance, randomly gen. instances: $n = 100$, $m = 20$ to 780

decision time peak occurs at $\frac{m}{n} \approx 5$, unlike what is expected. Furthermore, the peak time for solving a $\mathcal{L}_\infty$ problem is about 35 seconds. This unexpected behavior may be credited to the fact that YICES converts internally all floating point numbers to pair of integers, which impacts the efficiency of problems whose formulation involves a lot of floating point numbers as is the case of $\mathcal{L}_\infty$ decision.

Figure 1b presents an $\mathcal{L}_\infty$-solver build with using MIP solver SCIP, in which we can see a phase transition from SAT to unSAT also at $\frac{m}{n} \approx 2$, with an average time peak also around $\frac{m}{n} \approx 2$, as expected. Furthermore, the peak time is 0.35 seconds, two orders of magnitude more efficient than the YICES solver. Observing the average time, we note an always increasing right tail, which can be credited to the fact that MIP solvers are not implemented with a "fail early" strategy commonly used in logic based solvers, which normally employ what is called restriction learning strategies; furthermore, the size of the matrices used by the MIP solver increases with m. Another possibility to explain such a behavior is the fact that the choice of the family of formulas may be inappropriate, however no such increasing tail was observed in the SMT based method, which reinforces the hypothesis that this behavior is due to the MIP solver. Due to its superior efficiency we only use the SCIP solver as an auxiliary procedure for the LIPSAT solver described next.

5.2 Phase Transition for LIPSAT

The input for the LIPSAT solver is a normal form $\langle \Gamma, \Theta \rangle$. We developed a C++-implementation for Algorithm 4.1 using the C++ interface of the SoPlex linear algebra solver which is part of the SCIP suite of optimizers. We used the $\mathcal{L}_\infty$-solver based on SCIP MIP.

The experiments were obtained as follows. The input $\mathcal{L}_\infty$-formula Γ was generated in the form we describe above, with a fixed number of symbols n and a varying number of clauses of format (9) and (10) as described above. The probabilistic Θ-restrictions of the form $\{C(y_i) = q_i \mid i \leq i \leq k\}$ were generated fixing $k \leq n$ and randomly choosing the probabilities q_i uniformly over the interval $[0, 1]$.

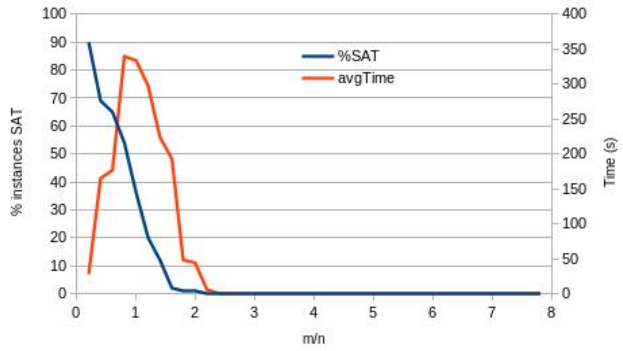


Fig. 2: Phase transition for LIPSAT solver: $k = 20$, $n = 100$ and $m = 20$ to 780

The results of the experiment can be seen in Fig. 2. We clearly see a second order phase transition with a peak average time execution that overlaps the decreasing part of the percentage SAT curve. Note that no increasing tail is observed, so that the "fail early" mechanism is achieved in the combination of logic and linear algebra. The peak is near but does not coincide with the fifty percent point of the first order phase transition which may be credited to the increasing shape of the right tail in the L_∞-solver presented in Fig. 2. Also, there is a left shift of the phase transition point $\frac{m}{n} \approx 1$, similar to the shift of PSAT phase transition point with respect to SAT [16]. Overall the phase transition format can be considered satisfactory.

6 Conclusion and the Future

We provided the theoretical basis for the development and implementation of probabilistic reasoning over "partial truth" that respect Łukasiewicz Infinitely-valued Logic restricts. A phase transition behavior could be empirically observed. For the future we hope to develop better solvers for the logics employed having the analysis of the phase transition as a qualitative guideline; and hope to employ the mechanisms developed here to linearly approximate generic functions.

References

1. Achterberg, T.: Scip: solving constraint integer programs. Mathematical Programming Computation **1**(1) (2009) 1–41 See http://scip.zib.de/.
2. Bertsimas, D., Tsitsiklis, J.N.: Introduction to linear optimization. Athena Scientific (1997)
3. Bofill, M., Manya, F., Vidal, A., Villaret, M.: Finding hard instances of satisfiability in Łukasiewicz logics. In: ISMVL, IEEE (2015) 30–35
4. Boole, G.: An Investigation on the Laws of Thought. Macmillan, London (1854) Available on project Gutemberg at `http://www.gutenberg.org/etext/15114`.
5. Borgward, K.H.: The Simplex Method: A Probabilistic Analysis. Algorithms and Combinatorics 1. Springer (1986)

6. Bova, S., Flaminio, T.: The coherence of Łukasiewicz assessments is NP-complete. International Journal of Approximate Reasoning **51**(3) (2010) 294–304
7. Cheeseman, P., Kanefsky, B., Taylor, W.M.: Where the really hard problems are. In: 12th IJCAI, Morgan Kaufmann (1991) 331–337
8. Cignoli, R., d'Ottaviano, I., Mundici, D.: Algebraic Foundations of Many-Valued Reasoning. Trends in Logic. Springer Netherlands (2000)
9. de Finetti, B.: Sul significato soggettivo della probabilità. Fundamenta Mathematicae **17**(1) (1931) 298–329
10. de Finetti, B.: La prévision: Ses lois logiques, ses sources subjectives (1937)
11. de Finetti, B.: Theory of probability: A critical introductory treatment. Translated by Antonio Machí and Adrian Smith. John Wiley & Sons (2017)
12. Dutertre, B.: Yices 2.2. In Biere, A., Bloem, R., eds.: Computer-Aided Verification (CAV'2014). Volume 8559 of Lecture Notes in Computer Science., Springer (July 2014) 737–744
13. Eckhoff, J.: Helly, Radon, and Caratheodory type theorems. In: Handbook of Convex Geometry. Elsevier Science Publishers (1993) 389–448
14. Finger, M., Bona, G.D.: Probabilistic satisfiability: Logic-based algorithms and phase transition. In Walsh, T., ed.: IJCAI, IJCAI/AAAI (2011) 528–533
15. Finger, M., De Bona, G.: Probabilistic satisfiability: algorithms with the presence and absence of a phase transition. AMAI **75**(3) (2015) 351–379
16. Finger, M., De Bona, G.: Probabilistic satisfiability: algorithms with the presence and absence of a phase transition. Annals of Mathematics and Artificial Intelligence **75**(3) (2015) 351–379
17. Gent, I.P., Walsh, T.: The SAT phase transition. In: ECAI94 – Proceedings of the Eleventh European Conference on Artificial Intelligence, John Wiley & Sons (1994) 105–109
18. Georgakopoulos, G., Kavvadias, D., Papadimitriou, C.H.: Probabilistic satisfiability. Journal of Complexity **4**(1) (1988) 1–11
19. Hähnle, R.: Towards an efficient tableau proof procedure for multiple-valued logics. In: 4th Workshop, CSL. Springer (1991) 248–260
20. Hansen, P., Jaumard, B.: Probabilistic satisfiability. In: Handbook of Defeasible Reasoning and Uncertainty Management Systems. Vol.5. Springer Netherlands (2000) 321
21. Hansen, P., Jaumard, B.: Algorithms for the maximum satisfiability problem. Computing **44** (1990) 279–303 10.1007/BF02241270.
22. Kavvadias, D., Papadimitriou, C.H.: A linear programming approach to reasoning about probabilities. AMAI **1** (1990) 189–205
23. McNaughton, R.: A theorem about infinite-valued sentential logic. Journal of Symbolic Logic **16** (1951) 1–13
24. Mundici, D.: Advanced Łukasiewicz calculus and MV-algebras. Trends in Logic. Springer Netherlands (2011)
25. Mundici, D.: Satisfiability in many-valued sentential logic is NP-complete. Theoretical Computer Science **52**(1-2) (1987) 145–153
26. Mundici, D.: A constructive proof of McNaughton's theorem in infinite-valued logic. The Journal of Symbolic Logic **59**(2) (1994) 596–602
27. Mundici, D.: Bookmaking over infinite-valued events. International Journal of Approximate Reasoning **43**(3) (2006) 223–240
28. Nilsson, N.: Probabilistic logic. Artificial Intelligence **28**(1) (1986) 71–87
29. Papadimitriou, C., Steiglitz, K.: Combinatorial Optimization: Algorithms and Complexity. Dover (1998)