DEPARTMENT OF OPERATIONS RESEARCH
UNIVERSITY OF AARHUS

Publication no. 99/1

Easy Cases of Probabilistic Satisfiability

Kim Allan Andersen
Daniele Pretolani

ISSN 1398-8964

Departments of Mathematical Sciences
Telephone: +45 8942 3188

Building 530, Ny Munkegade
DK-8000 Aarhus C, Denmark

Easy Cases of Probabilistic Satisfiability

K. A. Andersen[*] D. Pretolani[†]

March 1999

Abstract

The Probabilistic Satisfiability problem (PSAT) can be considered as a probabilistic counterpart of the classical SAT problem. In a PSAT instance, each clause in a CNF formula is assigned a probability of being true; the problem consists in checking the consistency of the assigned probabilities. Actually, PSAT turns out to be computationally much harder than SAT, e.g. it remains difficult for some classes of formulas where SAT can be solved in polynomial time. A *column generation* approach has been proposed in the literature, where the pricing sub-problem reduces to a Weighted Max-SAT problem on the original formula.

Here we consider some easy cases of PSAT, where it is possible to give a compact representation of the set of consistent probability assignments. We follow two different approaches, based on two different representations of CNF formulas.

First we consider a representation based on *directed hypergraphs*. By extending a well-known integer programming formulation of the Max-SAT problem, we solve the case in which the hypergraph does not contain cycles.

Then we consider the co-occurrence graph associated with a formula. We provide a solution method for the case in which the co-occurrence graph is a partial 2-tree.

Keywords: *probabilistic satisfiability, CNF formulas; directed hypergraphs; partial* 2-*trees; balanced matrices.*

[*]Afdeling for Operationsanalyse, Institut for Matematiske Fag, Aarhus Universitet, Bygning 530, Ny Munkegade, DK-8000 Århus C (Danmark). *e-mail:* `kima@imf.au.dk`

[†]Dipartimento di Matematica e Fisica, Università di Camerino, via Madonna delle Carceri, I-62032 Camerino (Italia). *e-mail:* `pretola@campus.unicam.it`

1 Introduction

Two of the main problems in *artificial intelligence, logic* and *computational complexity* are the *satisfiability problem* (SAT) and the *probabilistic satisfiability problem* (PSAT). These two problems can be formulated as follows:

- Satisfiability problem - SAT:

  Suppose we have a collection $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of m clauses on n propositional variables $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$. Determine whether or not there exists a truth assignment for the n propositional variables in $\mathcal{X}$ such that all m clauses in $\mathcal{C}$ are simultaneously satisfied.

- Probabilistic satisfiability problem - PSAT:

  Suppose we have a collection $\mathcal{C} = \{C_1, C_2, \ldots, C_m\}$ of m clauses on n propositional variables $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$. Assign probabilities $\pi_1, \pi_2, \ldots, \pi_m$ to the m clauses in $\mathcal{C}$. Determine whether or not this assignment of probabilities to the clauses is consistent.

Notice that PSAT is a generalization of SAT. We obtain SAT from PSAT by assigning the probabilities $(\pi_1, \pi_2, \ldots, \pi_m) = (1, 1, \ldots, 1)$ to the m clauses. It is well known that SAT is an $\mathcal{NP}$-complete problem [8]. It immediately follows that also PSAT is $\mathcal{NP}$-complete.

Probabilistic satisfiability is far from a new problem. In fact it dates back to George Boole [12] in the nineteenth century. These days the problem is formulated in terms of a linear programming model [12, 18], which were not known by George Boole at the time; nonetheless, he was almost able to formulate it in today's terms. The problem continues to be just as relevant as ever. It is well known that SAT has a wide range of applications. One example is checking the consistency of clauses in expert system knowledge bases. To the best of our knowledge there has been no direct application of PSAT as such, but one possible application might be checking the consistency of probabilities assigned to clauses in knowledge bases. It may very well be that one does not know that a particular clause is true with certainty but one is able to assign a probability that the clause is true. In this context, a significant tractable fragment of PSAT may be useful for supporting specific applications, in analogy with what happens for *Horn clauses* in traditional knowledge bases.

Since its introduction, PSAT has been considered to be computationally hard [18]; solution methods for PSAT still require substantial improvements, comparable to those obtained for SAT in the last few years. As SAT is a special case of PSAT one may hope that the methods to solve SAT can be modified to take care of the generalized problem PSAT. This, however, is certainly not the case: SAT is by nature a purely combinatorial problem, whereas PSAT is an inherently continuous problem. Therefore PSAT cannot take advantage of most of the techniques developed for SAT. The solution methods for PSAT proposed in the literature are primarily based on *column generation schemes* [13, 14, 16]. In this context, it has been shown that the *pricing* problem can be reduced to an instance of *weighted Max-SAT*. This implies that PSAT can be solved in polynomial time (via the ellipsoid method) for those classes of formulas where weighted Max-SAT can be solved in polynomial time. It follows that every advance in the research on Max-SAT has a direct impact on the solution of PSAT problems. However, it is not likely that computational methods for Max-SAT (which is a purely combinatorial problem, as SAT is) may be adapted to the solution of PSAT.

Nevertheless, the research on PSAT can utilize several tools that have been originally developed within, or succesfully applied to, the research on satisfiability problems and related areas. In fact, the techniques we develop in this paper are based on several such tools.

For example, *directed hypergraphs* have been widely applied, both as a modelling and an algorithmic tool, to many classes of satisfiability problems. Besides the general SAT problem, these applications include (but are not limited to): polynomially solvable cases and variants [19]; the Max-SAT problem for Horn formulas [10]; Horn formulas with uncertainty [4]; solvable fragments of first order logic [20].

Furthermore, *balanced matrices* have been used [6] to define a class of formulas where several logical problems, including SAT and Max-SAT, can be solved by linear programming.

Finally, *co-occurence graphs* are commonly adopted for representing logical formulas as well as *pseudo–Boolean functions* [7]. Based on co-occurrence partial k-trees, classes of polynomially solvable instances have been defined, e.g. for Max-SAT and PSAT [11] and for pseudo-Boolean programming [7].

The outline of the paper is as follows. In Section 2 we introduce our notation. In Section 3 we consider the case where the clauses are represented by a directed hypergraph. If the hypergraph does not contain any cycles, we solve the PSAT problem by extending a well-known integer programming formulation of the Max-SAT problem. In Section 4 we consider the co-occurrence graph associated with a set of clauses. In the particular case where the co-occurrence graph is a partial 2-tree we provide a solution method for PSAT. The method is based on a reduction scheme for the partial 2-tree. Section 5 contains the conclusions.

2 The Probabilistic Satisfiability Problem

Let $\mathcal{X}$ be a set of *propositional variables*, or *propositions*; a *positive literal* is a variable $x \in \mathcal{X}$, and a *negative literal* is a negated variable $\neg x$, $x \in \mathcal{X}$. A *clause* is a disjunction of literals; we assume that a clause contains at least one literal, and does not contain both literals x and $\neg x$. A *formula* in *Conjunctive Normal Form* (CNF) is a conjunction of clauses, and is denoted by the pair $\sigma = (\mathcal{X}, \mathcal{C})$, where $\mathcal{C}$ is a set of clauses.

We write $x \in C$ if clause C contains literal x; $\mathcal{X}(C)$ denotes the set of variables that appear in C, either as positive or negative literals. The length of a clause is the number of literals it contains; a unit (binary) clause has length one (two). We say that σ is a *kSAT* formula if the length of its clauses is at most k. We denote by L the *length* of σ, that is the sum of the lengths of the clauses.

Given a formula $\sigma = (\mathcal{X}, \mathcal{C})$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ and $\mathcal{C} = \{C_1, \ldots, C_m\}$ the *clause-*

variable incidence matrix A of σ is an $m \times n$ $\{0, \pm 1\}$ matrix in which:

$$A_{ij} = \begin{cases} -1 & \text{if } \neg x_j \text{ appears in } C_i \\ +1 & \text{if } x_j \text{ appears in } C_i \\ 0 & \text{otherwise.} \end{cases}$$

Denote by $n(E)$ the vector that gives the number of -1 in each row of an $m \times n$ $\{0, \pm 1\}$ matrix E, and denote by $\mathbf{1}$ the vector of dimension m with entries $+1$. It is well known that the *maximum satisfiability* problem (Max-SAT) can be formulated as an Integer Linear Programming (ILP) problem as follows:

$$(P) = \begin{cases} \min \mathbf{1}\mu \\ A\lambda + \mu \quad \geq \quad \mathbf{1} - n(A) \\ \lambda \in \{0,1\}^n, \mu \in \{0,1\}^m \end{cases}$$

The intuition behind problem (P) is that variable x_i is set to *true* (*false*) if $\lambda_i = 1$ ($\lambda_i = 0$), and $\mu_i = 1$ if clause C_i is not satisfied. Note that the *satisfiability* problem (SAT) corresponds to finding a solution of (P) with $\mu = \mathbf{0}$.

A *truth assignment* or, equivalently, a *possible world*, is an assignment of values {true,false} to the variables in $\mathcal{X}$. If $|\mathcal{X}| = n$ we denote by $\mathcal{W}^n$ the set of possible worlds. Notice that the number of possible worlds is $N = 2^n$.

For a given truth assignment (possible world) the *truth value* of a clause C is 1 if C is satisfied by the truth assignment, and 0 otherwise.

A vector $y \in \{0,1\}^m$ is a *(feasible) truth vector* for σ if, for at least one possible world, y_i is the truth value of clause C_i for $1 \leq i \leq m$. We denote by $Y = \{y_1, y_2, \ldots, y_N\}$ the set of N truth vectors corresponding to the set of possible worlds $\mathcal{W}^n$ (some of the truth vectors might be identical). Denote by M the $m \times N$ matrix with columns $\{y_1, y_2, \ldots, y_N\}$.

A *probability assignment* to the possible worlds $\mathcal{W}^n$ is a non-negative N-vector p which contains the probabilities of the possible worlds. The sum of the entries in p is equal to 1. Note that p gives a convex combination of the vertices of the unit hypercube in $\mathbb{R}^n$, i.e. it defines a vector $r(p)$ in the hypercube. In particular, $r(p)_i$ gives the probability that variable x_i is true under the probability assignment p.

Let p be a particular probability assignment to the possible worlds $\mathcal{W}^n$. Then the probability π_i of a clause C_i is the sum of the probabilities of the possible worlds in which the clause is true.

We denote by $\Pi \subset \mathbb{R}^m$ the polyhedron containing the feasible probability assignments π to clauses in σ. Clearly $\Pi = \text{conv}(Y) = \{\pi \in R^m : Mp = \pi,\ \mathbf{1}p = 1,\ p \geq 0\}$.

Suppose that $\pi \in [0,1]^m$ is an assignment of probabilities to the clauses in σ. The *probabilistic satisfiability* problem (PSAT) is to determine if $\pi \in \Pi$. If this is the case we say that the assignment of probabilities π to the clauses in $\mathcal{C}$ is *consistent*, i.e. there exists a probability assignment p to the possible worlds $\mathcal{W}^n$ such that the probability of clauses C_i is equal to π_i, for $1 \leq i \leq m$.

Now let $\sigma = (\mathcal{X}, \mathcal{C})$ be a CNF formula. Assume we consistently assign a probability vector π to the formulas in $\mathcal{C}$. Then we might ask the following question:

> What is the minimum (maximum) probability we can assign to a propositional variable $x \in \mathcal{X}$, given that the clauses in $\mathcal{C}$ have the probabilities π?

This question can be answered by mimimizing (maximizing) a linear programming problem [3]. The minimum probability we can assign to $x_i \in \mathcal{X}$ is found by solving the problem:

$$\begin{array}{lll} \min & a \cdot p & \\ \text{s.t.} & & \\ & Mp & = \pi \\ & \mathbf{1}p & = 1 \\ & p & \geq 0 \end{array} \tag{1}$$

where a is an N-vector with the j'th entry equal to 1 if propositional variable x_i is true in the j'th possible world and 0 otherwise. In practice, problem (1) asks to find the assignment p that yields the maximum value $r(p)_i$.

In the same way, we can find the minimum (maximum) probability that we can (consistently) assign to a particular clause $C_i \in \mathcal{C}$, given that the probabilities of the other clauses in $\mathcal{C}$ are fixed to the values given by π. For example, suppose we want to find the minimum probability for clause C_1 while keeping the probabilities of clauses $C_2, C_3, \ldots, C_m$ equal to the values $\pi_2, \pi_3, \ldots, \pi_m$. Let m_1 be the first row in M. We minimize the linear function $m_1 \cdot p$ subject to all the constraints in (1) except the first one (which has become the objective function instead). This is usually referred to as the *entailment* problem [18].

3 Directed Hypergraphs

A *(directed) hypergraph* $\mathcal{H}$ is a pair $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of *nodes* and $\mathcal{E}$ is the set of *(directed) hyperarcs*; a hyperarc is a pair $e = (T(e), H(e))$, where $T(e) \subseteq \mathcal{V}$ is the *tail* of e, and $H(e) \subseteq \mathcal{V} \setminus T(e)$ is its *head.* Only the basic definitions used in the paper are reported here; the interested reader is referred to the paper by Gallo *et al.* [9] for a detailed introduction to directed hypergraphs.

We write $u \in e$ if hyperarc e contains node u, i.e. $u \in T(u) \cup H(e)$. We denote the cardinality of hyperarc e by:

$$|e| = |T(e)| + |H(e)|;$$

we assume that $|e| \geq 1$, i.e. either $T(e)$ or $H(e)$ may be empty, but not both. We define the size of $\mathcal{H}$ as the sum of the cardinalities of its hyperarcs:

$$\text{size}(\mathcal{H}) = \sum_{e \in \mathcal{E}} |e|.$$

Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_n\}$ and $\mathcal{E} = \{e_1, \ldots, e_m\}$ the *(hyperarc-node) incidence matrix* $I(\mathcal{H})$ of $\mathcal{H}$ is an $m \times n$ $\{0, \pm 1\}$ matrix where:

$$I_{ij} = \begin{cases} -1 & \text{if } v_j \in T(e_i); \\ +1 & \text{if } v_j \in H(e_i); \\ 0 & \text{otherwise.} \end{cases}$$

An *undirected path* (or, simply, a path) P_{st}, of length q, in the hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ is a sequence:

$$P_{st} = (v_1 = s, e_1, v_2, e_2, \ldots, v_{q+1} = t)$$

where, for $1 \leq i \leq q$, $e_i \in \mathcal{E}$, $v_i \in e_i$ and $v_{i+1} \in e_i$. Nodes t and s are *connected* by P_{st}. A path P_{st} is a *cycle* if $s = t$. A hypergraph is *acyclic* if it does not contain any cycle; an acyclic hypergraph is a *hypertree* if each pair of nodes is connected by a path. Note that $|\mathcal{V}| < |\mathcal{E}|$ in a hypertree.

A cycle is *simple* if the following conditions hold:

1. the nodes $\{v_1, v_2, \ldots, v_{q+1}\}$ are distinct, except for $v_1 = v_{q+1}$;
2. the hyperarcs $\{e_1, e_2, \ldots, e_q\}$ are distinct;
3. $v_i \in e_j \;\Rightarrow\; i \in \{j, j+1\}$.

The last condition means that a hyperarc e_j does not contain any node in the cycle except v_j and v_{j+1}. A hyperarc e_j is called a *bridge* if either $v_j, v_{j+1} \in T(e_j)$ or $v_j, v_{j+1} \in H(e_j)$. A simple cycle is *odd* if the total number of bridges is odd.

Hypergraphs provide a simple way of representing CNF formulas. Given a formula $\sigma = (\mathcal{X}, \mathcal{C})$, we associate a node u_j with each variable $x_j \in \mathcal{X}$. For each clause $C_i \in \mathcal{C}$, we define a hyperarc e_i whose tail and head contain the nodes associated with negative and positive literals in C_i respectively. In other words, $u_j \in H(e_i)$ if $x_j \in C_i$, and $u_j \in T(e_i)$ if $\neg x_j \in C_i$. The hypergraph $\mathcal{H}_\sigma = (\mathcal{V}_\sigma, \mathcal{E}_\sigma)$ *representing* σ is defined as follows:

- $\mathcal{V}_\sigma = \{\ u_i :\ x_i \in \mathcal{X}\ \}$;
- $\mathcal{E}_\sigma = \{\ e_i :\ C_i \in \mathcal{C}\ \}$.

Note that the incidence matrix $I(\mathcal{H}_\sigma)$ is in fact the clause-variable incidence matrix A of formula σ. Remark that a different representation is often used in the literature, see e.g. [9, 19].

3.1 Balanced Matrices and Hypergraphs

Let us briefly recall some basic results about *balanced matrices* (see e.g. [21, 6]). We say that a $\{0, \pm 1\}$ matrix is a *cycle matrix* if it contains exactly two non-zeroes in each row and in each column. Clearly, in a cycle matrix the sum of the non-zeroes modulo four is either zero or two. A matrix is *balanced* if, in each cycle sub-matrix, the sum of the non-zeroes modulo four is zero. Given a balanced $m \times n$ matrix B, define the polyhedron:

$$Q(B) = \{x \in \mathbb{R}^n :\ Bx \geq \mathbf{1} - n(B),\ \mathbf{0} \leq x \leq \mathbf{1}\}.$$

Recall that a non-empty polyhedron is *integral* if all its vertices are integral. The following result holds true:

Theorem 3.1 *([6] Th. 2.4) The polytope Q(B) is integral.*

We say that a hypergraph $\mathcal{H}$ is *balanced* if its incidence matrix $I(\mathcal{H})$ is balanced. In the following, we give a characterization of balanced hypergraphs in terms of simple cycles.

Theorem 3.2 *There is a one-to-one correspondence between simple cycles in $\mathcal{H}$ and cycle submatrices of $I(\mathcal{H})$.*

Proof: ($\Rightarrow$) Given a simple cycle $(v_1 = s, e_1, v_2, e_2, \ldots, v_{q+1} = s)$ in $\mathcal{H}$, consider the square submatrix of $I(\mathcal{H})$ whose rows and columns correspond to hyperarcs $e_1, e_2, \ldots, e_q$ and nodes $v_1, v_2, \ldots, v_q$ respectively; since the cycle is simple, each row and column contains exactly two non-zeroes.
($\Leftarrow$) consider a cycle submatrix C of $I(\mathcal{H})$; assume w.l.o.g. that the rows of C correspond to hyperarcs $e_1, e_2, \ldots, e_q$ in this order, and that the columns of C correspond to nodes $v_1, v_2, \ldots, v_q$ in this order, moreover:

- row i, $1 \leq i < q$, contains the non-zero entries C_{ii} and C_{ij}, $j = i + 1$;
- row q contains the non-zero entries C_{1q} and C_{qq}.

It is easy to see that $(v_1, e_1, v_2, e_2, \ldots, v_{q+1} = v_1)$ is a simple cycle in $\mathcal{H}$. □

Theorem 3.3 *A hypergraph $\mathcal{H}$ is balanced if and only if it does not contain odd simple cycles.*

Proof: It follows from Theorem 3.2, observing that in a cycle sub-matrix of $I(\mathcal{H})$ the sum of the non-zeroes modulo four is two if and only if the corresponding simple cycle is odd. □

It follows immediately from Theorem 3.3 that every hypertree is balanced. In the following, we define a particular class of balanced hypergraphs, obtained by suitably extending hypertrees. Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, we obtain the *extended hypergraph* $\mathcal{H}^E = (\mathcal{V}^E, \mathcal{E}^E)$ as follows. For each hyperarc $e \in \mathcal{E}$ we add a new node $u(e)$. Then, for each node $u \in T(e)$ we add a hyperarc:

$$e^T(u) = \Big(\emptyset, \{u, u(e)\}\Big),$$

while for each node $u \in H(e)$ we add a hyperarc:

$$e^H(u) = \Big(\{u\}, \{u(e)\}\Big).$$

Finally, we add node $u(e)$ to the tail of e, obtaining a new hyperarc

$$e^E = \Big(T(e) \cup \{e(u)\}, H(e)\Big).$$

Note that for each hyperarc $e \in \mathcal{E}$ there are $|e| + 1$ hyperarcs in $\mathcal{E}^E$; an example is given in figure 1. In formal terms, we have:

$\mathcal{V}^E = \mathcal{V} \cup \{u(e) : \ e \in \mathcal{E}\}$;

$\mathcal{E}^E = \{e^E : \ e \in \mathcal{E}\} \ \cup \ \{e^T(u) : \ e \in \mathcal{E}, u \in T(e)\} \ \cup \ \{e^H(u) : \ e \in \mathcal{E}, u \in H(e)\}$.

Theorem 3.4 *The extended hypergraph $\mathcal{H}^E$ obtained from a hypertree $\mathcal{H}$ is balanced.*

Proof: In light of Theorem 3.3, we shall show that $\mathcal{H}^E$ does not contain odd simple cycles. Let C be a cycle in $\mathcal{H}^E$. Since $\mathcal{H}$ is acyclic, all the nodes in C must belong to the same hyperarc $e^E \in \mathcal{E}^E$, moreover, C must contain hyperarc e^E (see figure 1). Therefore, in order to be simple, C must contain at most two nodes, and consequently it has one of the following forms:

$$C = \Big(u(e), e^E, u, e^T(u), u(e)\Big) \qquad u \in T(a)$$

$$C = \Big(u(e), e^E, u, e^H(u), u(e)\Big) \qquad u \in H(a)$$

In both cases, C is not odd, and the theorem follows. □

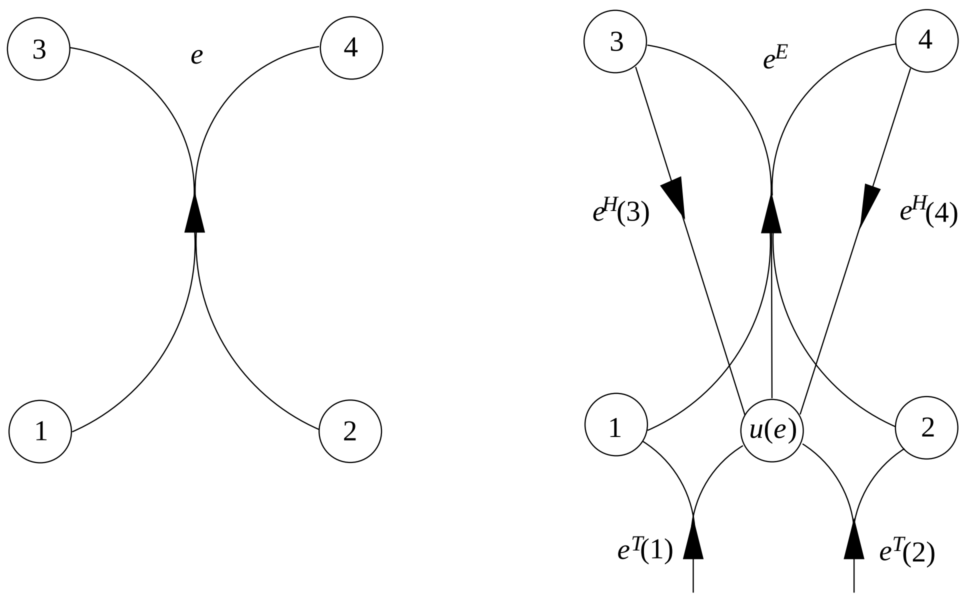

Figure 1: Hyperarcs in $\mathcal{E}^E$

3.2 A PSAT Formulation for Hypertrees

We consider formulas represented by hypertrees, and we propose a formulation that reduces the PSAT problem to the solution of a set of linear inequalities. Our approach is similar to the one proposed in [1, 2], i.e. we give a description of the polyhedron $\Pi \subset \mathbb{R}^m$ containing the feasible probability assignments. However, in our formulation we obtain Π as the projection onto $\mathbb{R}^m$ of a polyedron in $\mathbb{R}^{m+n}$. In this way we obtain a compact representation of Π, where the number of variables and constraints is linear in n.

Consider a formula $\sigma = (\mathcal{X}, \mathcal{C})$, where $\mathcal{X} = \{x_1, \dots, x_n\}$ and $\mathcal{C} = \{C_1, \dots, C_m\}$. We introduce two vectors of binary variables, $x \in \{0,1\}^n$, and $y \in \{0,1\}^m$. The x vector represents a truth assignment for the propositional variables, i.e. $x_j \in \mathcal{X}$ is set to true (false) if $x_j = 1$ ($x_j = 0$). Moreover, y represents a truth vector, i.e. $y_i = 1$ if clause C_i is satisfied, and $y_i = 0$ otherwise. The variables x and y are related by means of the following sets of constraints:

$$\begin{array}{rrcll} \text{i)} & Ax & \geq & y - n(A) & \\ \text{ii)} & y_i & \geq & x_j & \forall C_i \in \mathcal{C},\ x_j \in C_i; \\ \text{iii)} & y_i & \geq & (1 - x_j) & \forall C_i \in \mathcal{C},\ \neg x_j \in C_i; \end{array} \tag{2}$$

where A is the clause-variable incidence matrix of σ. Constraints (2.i) mean that if $y_i = 1$ then clause C_i must be satisfied by the truth assignment given by x. Indeed, by setting $\lambda = x$ and $\mu = \mathbf{1} - y$ we obtain the constraints of the ILP formulation (P) of Max-SAT. Constraints (2.ii) and (2.iii) mean that if $y_i = 0$ then clause C_i must *not* be satisfied, hence it must be $x_j = 0$ for each positive literal $x_j \in C_i$, and $x_j = 1$ for each negative literal $\neg x_j \in C_i$. Note that the constraints (2) can be rewritten as follows:

$$\begin{array}{rrcll} \text{i)} & Ax - y & \geq & \mathbf{1} - (n(A) + \mathbf{1}) & \\ \text{ii)} & -x_j + y_i & \geq & 0 & \forall C_i \in \mathcal{C},\ x_j \in C_i; \\ \text{iii)} & x_j + y_i & \geq & 1 & \forall C_i \in \mathcal{C},\ \neg x_j \in C_i; \end{array} \tag{3}$$

it is easy to see that constraints (3) can be written as:

$$B[x,y] \geq \mathbf{1} - n(B) \tag{4}$$

where B is an $(m+L) \times (n+m)$ $\{0, \pm 1\}$ matrix.

As discussed above, A is the incidence matrix of the hypergraph $\mathcal{H}_\sigma$ representing the formula σ: row i represents the hyperarc e_i associated with clause $C_i \in \mathcal{C}$, and column j represents node u_j. In fact, B is the incidence matrix of the expanded hypergraph $\mathcal{H}_\sigma^E$ obtained from $\mathcal{H}_\sigma$. To see why this is true, assume that each variable y_i represents the node $u(e_i)$ added to the tail of e_i in $\mathcal{H}_\sigma^E$. Then the (row of B corresponding to) constraint (3.i) for clause C_i represents the hyperarc $e_i^E \in \mathcal{E}^E$. Moreover, for each positive literal x_j in clause C_i, the constraint (3.ii) represents the hyperarc $e_i^H(u_j) \in \mathcal{E}^E$. Finally, for each negative literal $\neg x_j$ in clause C_i, the constraint (3.iii) represents the hyperarc $e_i^T(u_j) \in \mathcal{E}^E$.

In light of the above observations and of Theorem 3.4, we can state the following:

Lemma 3.1 *If formula σ is represented by a hypertree, the matrix B in constraints (4) is balanced.*

Denote by S the set of vectors $[x,y] \in \{0,1\}^{n+m}$ satisfying the constraints (2):

$$S = \Big\{[x,y] \in \{0,1\}^{n+m} \,:\, B[x,y] \geq \mathbf{1} - n(B)\Big\}.$$

Let us define the projection of S onto the y variables space:

$$S_y = \Big\{y \in \{0,1\}^m \,:\, \exists x \in \{0,1\}^n,\, [x,y] \in S\Big\}.$$

Clearly, S_y is the set of feasible truth vectors for σ, i.e. $Y = S_y$. Note also that $\mathrm{conv}(S_y)$ is the projection onto the y space of $\mathrm{conv}(S)$:

$$\Pi = \mathrm{conv}(S_y) = \mathrm{conv}(S)_y = \Big\{y \in [0,1]^m \,:\, \exists x \in [0,1]^n,\, [x,y] \in \mathrm{conv}(S)\Big\}$$

Consider now the solutions of the *linear relaxation* of constraints (2), i.e. the set $\overline{S}$ of vectors $[x,y] \in [0,1]^{n+m}$ satisfying (2):

$$\overline{S} = \Big\{[x,y] \in [0,1]^{n+m} \,:\, B[x,y] \geq \mathbf{1} - n(B)\Big\}.$$

Define the projection of $\overline{S}$ onto the y variable space, i.e. the polyhedron:

$$\overline{S}_y = \Big\{y \in [0,1]^m \,:\, \exists x \in [0,1]^n,\, [x,y] \in \overline{S}\Big\}$$

It is well known that $\mathrm{conv}(S) \subseteq \overline{S}$, and therefore $\mathrm{conv}(S)_y \subseteq \overline{S}_y$, but $\mathrm{conv}(S)_y \neq \overline{S}_y$ in general. However, if the polyhedron $\overline{S}$ is integral, we have $\mathrm{conv}(S) = \overline{S}$ and $\Pi = \mathrm{conv}(S)_y = \overline{S}_y$. This is the case, in particular, if the matrix B is balanced. In light of Lemma 3.1, we can state the following theorem:

Theorem 3.5 *Given a formula σ represented by a hypertree, a vector $y \in [0,1]^m$ is a feasible probability assignment if and only if there exists an $x \in [0,1]^n$ such that $[x,y] \in \overline{S}$.*

In order to check whether a given vector $\pi \in [0,1]^m$ is a feasible probability assignment, we must find a feasible solution $x \in \mathbb{R}^n$ for the system $B[x, \bar{y}] \geq \mathbf{1} - n(B)$, where $\bar{y} = \pi$. In practice, x must satisfy the following system of inequalities:

$$\begin{cases} Ax \geq \bar{y} - n(A) & \\ -x_j \geq -\bar{y}_i & \forall C_i \in \mathcal{C},\ x_j \in C_i; \\ x_j \geq (1 - \bar{y}_i) & \forall C_i \in \mathcal{C},\ \neg x_j \in C_i; \end{cases} \tag{5}$$

It is not difficult to see that (5) can be reduced to a system of at most $2n + m < 3n$ inequalities:

$$\begin{cases} Ax \geq \bar{y} - n(A) \\ l \leq x \leq u \end{cases} \tag{6}$$

where :

$$l_j = \max\{0, \max_{\neg x_j \in C_i} (1 - \bar{y}_i)\}, \qquad u_j = \min\{1, \min_{x_j \in C_i} \bar{y}_i\}.$$

Moreover, we can find the maximum (minimum) probability for a proposition x_j by solving an LP problem where the feasible region is defined by (6), and x_j is the objective function to be maximized (minimized).

Consider now the problem of finding the maximum (minimum) probability of a clause C_i. Since we have to maximize or minimize the value of y_i, the inequalities (2) involving y_i cannot be rewritten as in (5). In practice, we have to add these constraints to a system (6) obtained from the inequalities (2) not involving y_i. Since y_i appears in at most $n + 1$ inequalities, we obtain an LP problem with $n + 1$ variables and at most $4n$ constraints.

Result 1: for a CNF formula $\sigma = (\mathcal{X}, \mathcal{C})$ represented by a hypertree, with $|\mathcal{X}| = n$:

- the PSAT problem can be reduced to solving a system of at most $3n$ inequalities in n variables;
- the maximum (minimum) probability for a propositional variable can be obtained by solving an LP problem with n variables and at most $3n$ constraints;
- the maximum (minimum) probability for a clause can be obtained by solving an LP problem with $n + 1$ variables and at most $4n$ constraints.

4 Partial 2-trees

In this section, we consider a representation of CNF formulas based on *co-occurrence graphs.* In particular, we investigate the case where the graph representing a formula σ is a *partial* 2-*tree.* We reduce PSAT to the solution of a system of equations (in non-negative variables) whose size is linear in the number of variables of σ. The definitions and properties concerning partial 2-trees given below, are taken from the tutorial paper by Bodlaender [5], where the reader can find a more detailed survey of several related topics.

4.1 Partial 2-trees and co-occurrence graphs

An undirected graph $G = (N, A)$ is a *partial* 2-*tree* if and only if it can be reduced to an empty graph by means of the following *graph reduction* operations:

series operation *series*(i, k, j): given three nodes i, j such that $(i, k) \in A$, $(k, j) \in A$, and k has degree two, delete arcs $(i, k), (k, j)$ and node k; if $(i, j) \notin A$, add (i, j) to A;

tail operation *tail*(i, j): given two nodes i, j such that $(i, j) \in A$ and j has degree one, delete arc (i, j) and node j;

node operation *node*(i): delete the isolated node i.

¿From now on, we assume that G is connected. Let $|N| = n$ and $|A| = m$. Since each series and tail operation deletes exactly one node, and returns a connected graph, a *reduction sequence* $R = \{r_1, r_2, \ldots, r_{n-1}\}$ of exactly $(n-1)$ series and tail operations reduces G to a single node. Note that r_{n-1} is a tail operation, therefore we have at most $(n-2)$ series operations in R. As a consequence, it is $m \le m_D \le 2n - 3$, where m_D is the total number of arcs deleted by reduction operations.

In figure 2 we show a partial 2-tree (actually a 2-tree) and the graphs obtained by the reduction sequence:

$$R = \Big\{ series(u, y, v),\ series(u, w, v),\ tail(u, v) \Big\}.$$

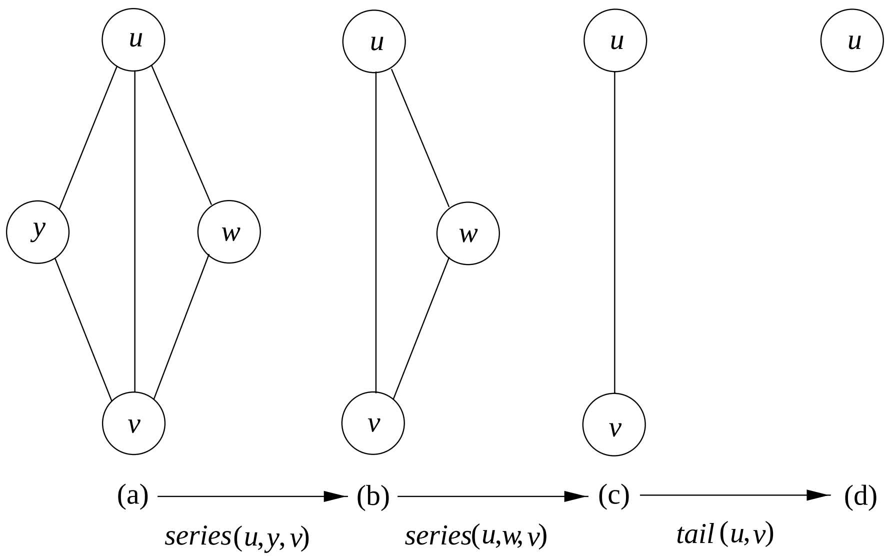

Figure 2: Reduction of a partial 2-tree.

Observe that no reduction operation can be applied to a complete graph of four nodes (K_4); thus if G contains K_4 as a subgraph, then it is not a partial 2-tree. In fact, G is a partial 2-tree if and only if it does not contain K_4 as a *minor* ([5] Th. 17(ii)).

Given a CNF formula $\sigma = (\mathcal{X}, \mathcal{C})$, the corresponding *co-occurrence graph* $G_\sigma = (N_\sigma, A_\sigma)$ contains one node for each variable $x \in \mathcal{X}$, and one arc (x, x') for each pair of variables that appear (either as positive or negative literals) in the same clause. In formal terms:

- $N_\sigma = \mathcal{X}$;
- $A_\sigma = \Big\{(x, x') : \exists C \in \mathcal{C} : \{x, x'\} \subseteq \mathcal{X}(C)\Big\}$.

Observe that G_σ does not provide a complete representation of σ, indeed positive and negative literals are not distinguished. Moreover, unit clauses are not represented at all. In fact, the same co-ocurrence graph can represent several different formulas. As an example, the partial 2-tree in figure 2 represents the 2SAT formula:

$$\sigma = \Big\{(u \vee y) \wedge (\neg u \vee w) \wedge (u \vee \neg v) \wedge (w \vee v) \wedge (\neg y \vee v) \wedge (y)\Big\}$$

as well as the 3SAT formula:

$$\sigma' = \Big\{(y \vee \neg u \vee v) \ \wedge \ (\neg w \vee u \vee \neg v)\Big\}.$$

A clause of length four induces a complete subgraph K_4 in G_σ. Therefore the co-occurrence graph of σ is a partial 2-tree only if the maximum clause length is three. In the following, we denote by 2T-SAT the class of formulas represented by a co-occurrence partial 2-tree.

4.2 A PSAT Formulation for 2T-SAT

Let $V \subseteq \mathcal{X}$ a set of propositions, and W a set containing exactly one literal, either x or $\neg x$, for each $x \in V$. Let $|V| = |W| = k$. The set W represents a *partial truth assignment* on the propositions V, where for each $x \in V$:

$$x = \begin{cases} \text{true} & \text{if } x \in W \\ \text{false} & \text{if } \neg x \in W. \end{cases}$$

In practice, W represents the 2^{n-k} possible worlds where the truth values of the variables in V are fixed as above. For a generic W, the *probability variable* p_W denotes the sum of the probability values assigned to the possible worlds represented by W. In other words, p_W is the probability that the variables in V are assigned truth values as specified by W. Note that there are 2^k possible truth assignments for V; the corresponding set of 2^k probability variables will be denoted by $\{p_V\}$.

It must be remarked that p_W is not a probability assignment for a clause, however, we can write the probability assignment for a clause in terms of probability variables. Consider the case where $V = \{u, v\}$; we have $2^k = 4$ possible partial truth assignments, represented by the sets $\{u, v\}$, $\{\neg u, v\}$, $\{u, \neg v\}$ and $\{\neg u, \neg v\}$. Accordingly, the set $\{p_{uv}\}$ contains four probability variables, denoted as:

$$p_{uv},\ p_{\bar{u}v},\ p_{u\bar{v}},\ p_{\bar{u}\bar{v}}$$

where, for $x \in \mathcal{X}$, $\bar{x}$ denotes $\neg x$. The probability assignment $\pi_{\bar{u}v}$ for clause $C = (\neg u \vee v)$ is the total probability value assigned to possible worlds where C is true, and is given by:

$$\pi_{\bar{u}v} = p_{uv} + p_{\bar{u}v} + p_{\bar{u}\bar{v}} = 1 - p_{u\bar{v}}.$$

In general, the probability assignment of a clause C can be written as

$$\pi_C = 1 - p_{\overline{C}}, \tag{7}$$

where $p_{\overline{C}} \in \{p_{\mathcal{X}(C)}\}$, and $\overline{C}$ denotes the unique truth assignment to variables in $\mathcal{X}(C)$ that falsifies C.

In principle, the number of partial truth assignments W is very large, namely $O(3^n)$. However, the underlying idea of our solution method for 2T-SAT is that only a small number of probability variables p_W need being introduced. Some of these variables are associated with nodes and arcs of the graph G_σ; the probability assignment of each clause can be written in terms of these variables, according to (7). Further variables and equations are introduced in order to relate probability variables to each other. This is done according to a reduction sequence for G_σ: the probability variables associated with deleted nodes and arcs are related to variables associated with elements remaining in the graph. Our approach is described in the following example.

We have the following set of clauses:

$$\sigma = \Big\{(u \vee y) \wedge (\neg u \vee w) \wedge (u \vee \neg v) \wedge (w \vee v) \wedge (\neg y \vee v) \wedge (y)\Big\}$$

represented by the co-occurrence graph G_σ shown in figure 2. Assume that the reduction sequence R for G_σ is given by $R = \{series(u, y, v),\ series(u, w, v),\ tail(u, v)\}$. First of all we need equations expressing the probability π_C of each of the six clauses $C \in \sigma$ in terms of the probability variables $p_{\bar{C}}$. The equations are:

$$\begin{aligned}
\pi_{yu} &= 1 - p_{\bar{y}\bar{u}} \\
\pi_{w\bar{u}} &= 1 - p_{\bar{w}u} \\
\pi_{\bar{v}u} &= 1 - p_{v\bar{u}} \\
\pi_{wv} &= 1 - p_{\bar{w}\bar{v}} \\
\pi_{\bar{y}v} &= 1 - p_{y\bar{v}} \\
\pi_{y} &= 1 - p_{\bar{y}}
\end{aligned}$$

Next we need to express the fact that the probability p_x of a propositional variable $x \in \{y, w, v, u\}$ plus the probability of its negation $p_{\bar{x}}$ must be equal to one. This gives rise to the set of four equations:

$$\begin{aligned}
p_y + p_{\bar{y}} &= 1 \\
p_w + p_{\bar{w}} &= 1 \\
p_v + p_{\bar{v}} &= 1 \\
p_u + p_{\bar{u}} &= 1
\end{aligned}$$

The five arcs in G_σ give rise to a set of equations. Let (x, x') be an arc in G_σ. We need to express the fact that the sum of the probabilities of the four possible combinations of propositions x and x' sum to one. We therefore get the equations:

$$\begin{aligned}
p_{yu} + p_{\bar{y}u} + p_{y\bar{u}} + p_{\bar{y}\bar{u}} &= 1 \\
p_{vu} + p_{\bar{v}u} + p_{v\bar{u}} + p_{\bar{v}\bar{u}} &= 1 \\
p_{wu} + p_{\bar{w}u} + p_{w\bar{u}} + p_{\bar{w}\bar{u}} &= 1 \\
p_{yv} + p_{\bar{y}v} + p_{y\bar{v}} + p_{\bar{y}\bar{v}} &= 1 \\
p_{wv} + p_{\bar{w}v} + p_{w\bar{v}} + p_{\bar{w}\bar{v}} &= 1
\end{aligned}$$

$\underline{series(u,y,v)}$

For the series operation (u, y, v), it is necessary to relate probability variables for arcs (y, u), (y, v) and (v, u) to each other. To this aim we introduce the probability variables $\{p_{yvu}\}$, whose probability must sum to one:

$$p_{yvu} + p_{yv\bar{u}} + p_{\bar{y}vu} + p_{\bar{y}v\bar{u}} + p_{y\bar{v}u} + p_{y\bar{v}\bar{u}} + p_{\bar{y}\bar{v}u} + p_{\bar{y}\bar{v}\bar{u}} = 1$$

$$\begin{array}{lllll}
p_{yv} & = & p_{yvu} & + & p_{yv\bar{u}} \\
p_{\bar{y}v} & = & p_{\bar{y}vu} & + & p_{\bar{y}v\bar{u}} \\
p_{y\bar{v}} & = & p_{y\bar{v}u} & + & p_{y\bar{v}\bar{u}} \\
p_{\bar{y}\bar{v}} & = & p_{\bar{y}\bar{v}u} & + & p_{\bar{y}\bar{v}\bar{u}} \\
p_{yu} & = & p_{yvu} & + & p_{y\bar{v}u} \\
p_{\bar{y}u} & = & p_{\bar{y}vu} & + & p_{\bar{y}\bar{v}u} \\
p_{y\bar{u}} & = & p_{yv\bar{u}} & + & p_{y\bar{v}\bar{u}} \\
p_{\bar{y}\bar{u}} & = & p_{\bar{y}v\bar{u}} & + & p_{\bar{y}\bar{v}\bar{u}} \\
p_{vu} & = & p_{yvu} & + & p_{\bar{y}vu} \\
p_{\bar{v}u} & = & p_{y\bar{v}u} & + & p_{\bar{y}\bar{v}u} \\
p_{v\bar{u}} & = & p_{yv\bar{u}} & + & p_{\bar{y}v\bar{u}} \\
p_{\bar{v}\bar{u}} & = & p_{y\bar{v}\bar{u}} & + & p_{\bar{y}\bar{v}\bar{u}}
\end{array}$$

The fact that the proposition y disappears under the series operation gives rise to the following two equations:

$$\begin{array}{lllllllll}
p_{y} & = & p_{yvu} & + & p_{y\bar{v}u} & + & p_{yv\bar{u}} & + & p_{y\bar{v}\bar{u}} \\
p_{\bar{y}} & = & p_{\bar{y}vu} & + & p_{\bar{y}\bar{v}u} & + & p_{\bar{y}v\bar{u}} & + & p_{\bar{y}\bar{v}\bar{u}}
\end{array}$$

$\underline{series(u,w,v)}$

We get a set of constraints similar to the ones obtained for operation $series(u, y, v)$.

$$p_{wvu} + p_{wv\bar{u}} + p_{\bar{w}vu} + p_{\bar{w}v\bar{u}} + p_{w\bar{v}u} + p_{w\bar{v}\bar{u}} + p_{\bar{w}\bar{v}u} + p_{\bar{w}\bar{v}\bar{u}} = 1$$

$$\begin{array}{lllll}
p_{wv} & = & p_{wvu} & + & p_{wv\bar{u}} \\
p_{\bar{w}v} & = & p_{\bar{w}vu} & + & p_{\bar{w}v\bar{u}} \\
p_{w\bar{v}} & = & p_{w\bar{v}u} & + & p_{w\bar{v}\bar{u}} \\
p_{\bar{w}\bar{v}} & = & p_{\bar{w}\bar{v}u} & + & p_{\bar{w}\bar{v}\bar{u}} \\
p_{wu} & = & p_{wvu} & + & p_{w\bar{v}u} \\
p_{\bar{w}u} & = & p_{\bar{w}vu} & + & p_{\bar{w}\bar{v}u} \\
p_{w\bar{u}} & = & p_{wv\bar{u}} & + & p_{w\bar{v}\bar{u}} \\
p_{\bar{w}\bar{u}} & = & p_{\bar{w}v\bar{u}} & + & p_{\bar{w}\bar{v}\bar{u}} \\
p_{vu} & = & p_{wvu} & + & p_{\bar{w}vu} \\
p_{\bar{v}u} & = & p_{w\bar{v}u} & + & p_{\bar{w}\bar{v}u} \\
p_{v\bar{u}} & = & p_{wv\bar{u}} & + & p_{\bar{w}v\bar{u}} \\
p_{\bar{v}\bar{u}} & = & p_{w\bar{v}\bar{u}} & + & p_{\bar{w}\bar{v}\bar{u}}
\end{array}$$

$$\begin{array}{lllllllll} p_w & = & p_{wvu} & + & p_{w\bar{v}u} & + & p_{wv\bar{u}} & + & p_{w\bar{v}\bar{u}} \\ p_{\bar{w}} & = & p_{\bar{w}vu} & + & p_{\bar{w}\bar{v}u} & + & p_{\bar{w}v\bar{u}} & + & p_{\bar{w}\bar{v}\bar{u}} \end{array}$$

tail(u,v)

In this case we just need to relate the probabilities of propositions u and v to the probability variable for arc (u,v) to each other in this way:

$$\begin{array}{lllll} p_u & = & p_{vu} & + & p_{\bar{v}u} \\ p_{\bar{u}} & = & p_{v\bar{u}} & + & p_{\bar{v}\bar{u}} \\ p_v & = & p_{vu} & + & p_{v\bar{u}} \\ p_{\bar{v}} & = & p_{\bar{v}u} & + & p_{\bar{v}\bar{u}} \end{array}$$

All together we have introduced 44 probability variables and 49 equations; some of them are, however, redundant. Consider for example the equations stating that the sum of the variables in a set $\{p_V\}$ is one: the four ones for $V = \{x\}$, $x \in \{y, w, v, u\}$, as well as the two ones for $V = \{y, v, u\}$ and $V = \{w, v, u\}$, are implied by the remaining equations. Moreover, some of the variables may be rewritten in terms of other variables.

What follows is a formal description of our method for the case where the clause length is at most two; the extension to 3-SAT is given later. Given a 2T-SAT formula $\sigma = (\mathcal{X}, \mathcal{C})$, represented by the co-occurence graph G_σ, let R be a reduction sequence for G_σ. We define the set $P(\sigma, R)$ of non-negative probability variables containing:

- for each node $u \in G_\sigma$, two variables p_u and $p_{\bar{u}}$;
- for each arc $(u,v) \in G_\sigma$, and for each arc $(u,v) \notin G_\sigma$ added by a series operation, the set $\{p_{uv}\}$, i.e. four variables p_{uv}, $p_{\bar{u}v}$, $p_{u\bar{v}}$ and $p_{\bar{u}\bar{v}}$;
- for each series reduction *series*(u,z,v), the set $\{p_{uvz}\}$, i.e. eight variables p_{uvz}, $p_{uv\bar{z}}$, $p_{\bar{u}vz}$, $p_{\bar{u}v\bar{z}}$, $p_{u\bar{v}z}$, $p_{u\bar{v}\bar{z}}$, $p_{\bar{u}\bar{v}z}$ and $p_{\bar{u}\bar{v}\bar{z}}$.

Recall that R contains at most $(n-2)$ series operations, and that $m_D \leq 2n-3$. Thus the total number of variables is at most $2n + 4(2n-3) + 8(n-2) = (18n - 28)$. We define the following sets of equality constraints:

i) for each clause $C \in \mathcal{C}$, the corresponding equation (7);

ii) for each node $u \in G_\sigma$, an equation

$$p_u + p_{\bar{u}} = 1;$$

for each arc $(u,v) \in G_\sigma$, and for each arc $(u,v) \notin G_\sigma$ added by a series operation, an equation:

$$p_{uv} + p_{\bar{u}v} + p_{u\bar{v}} + p_{\bar{u}\bar{v}} = 1;$$

for each series reduction *series*(u,z,v), an equation:

$$p_{uvz} + p_{uv\bar{z}} + p_{\bar{u}vz} + p_{\bar{u}v\bar{z}} + p_{u\bar{v}z} + p_{u\bar{v}\bar{z}} + p_{\bar{u}\bar{v}z} + p_{\bar{u}\bar{v}\bar{z}} = 1;$$

iii) for each tail reduction $tail(u,v)$, two pairs of equations:

$$\begin{array}{ll} \text{a)} & p_u = p_{uv} + p_{u\bar{v}}, \quad p_{\bar{u}} = p_{\bar{u}v} + p_{\bar{u}\bar{v}}; \\ \text{b)} & p_v = p_{uv} + p_{\bar{u}v}, \quad p_{\bar{v}} = p_{u\bar{v}} + p_{\bar{u}\bar{v}}; \end{array}$$

iv) for each series reduction $series(u,z,v)$, three sets of four equations:

$$\begin{array}{lllll} \text{a)} & p_{uv} = p_{uvz} + p_{uv\bar{z}}, & p_{\bar{u}v} = p_{\bar{u}vz} + p_{\bar{u}v\bar{z}}, & p_{u\bar{v}} = p_{u\bar{v}z} + p_{u\bar{v}\bar{z}}, & p_{\bar{u}\bar{v}} = p_{\bar{u}\bar{v}z} + p_{\bar{u}\bar{v}\bar{z}}; \\ \text{b)} & p_{uz} = p_{uvz} + p_{u\bar{v}z}, & p_{\bar{u}z} = p_{\bar{u}vz} + p_{\bar{u}\bar{v}z}, & p_{u\bar{z}} = p_{uv\bar{z}} + p_{u\bar{v}\bar{z}}, & p_{\bar{u}\bar{z}} = p_{\bar{u}v\bar{z}} + p_{\bar{u}\bar{v}\bar{z}}; \\ \text{c)} & p_{vz} = p_{uvz} + p_{\bar{u}vz}, & p_{\bar{v}z} = p_{u\bar{v}z} + p_{\bar{u}\bar{v}z}, & p_{v\bar{z}} = p_{uv\bar{z}} + p_{\bar{u}v\bar{z}}, & p_{\bar{u}\bar{z}} = p_{u\bar{v}\bar{z}} + p_{\bar{u}\bar{v}\bar{z}}; \end{array}$$

and a set of two equations:

$$\text{d)} \quad p_z = p_{uvz} + p_{\bar{u}vz} + p_{u\bar{v}z} + p_{\bar{u}\bar{v}z}, \qquad p_{\bar{z}} = p_{uv\bar{z}} + p_{\bar{u}v\bar{z}} + p_{u\bar{v}\bar{z}} + p_{\bar{u}\bar{v}\bar{z}}.$$

Constraints (i) express probability assignments in term of probability variables, while constraints (ii) require that probabilities sum up to one. The constraints (iii) and (iv) are related to reduction operations. For a tail reduction $tail(u,v)$, constraints (iii) relate the variables for node v and arc (u,v) to variables for node u. For a series reduction $series(u,z,v)$, constraints (iv) relate the variables for node z and arcs (u,z), (v,z) to variables for arc (u,v). Observe that these relations are written in terms of the probability variables $\{p_{uvz}\}$.

It is not difficult to see that for each arc (node) of G_σ we have at most four different binary clauses (two unit clauses respectively). Thus we have at most $4(2n-3)+2n = (10n-12)$ constraints (i), and at most $n+(2n-3)+(n-2) = (4n-5)$ constraints (ii). Moreover, the number of constraints (iii) and (iv) is at most $4+14(n-2) = (14n-24)$. Overall, we have at most $(28n-41)$ constraints.

As shown by the previous example, some of the variables and constraints we introduced are redundant, and might be eliminated. However, we believe that a certain degree of redundancy could make the presentation more compact and clear.

Extension to 3SAT formulas We can easily extend our method to 2T-SAT formulas containing clauses of length three. In practice, it suffices to add a constraint (i), i.e. an equation (7), for each such clause C. Indeed, we can show that it is not necessary to add new probability variables in this case.

Consider a clause C with $\mathcal{X}(C) = \{u,v,z\}$; this clause induces a clique of cardinality three in G_σ. Clearly, we need at least one series operation to reduce the clique. In particular, any reduction sequence R contains a series operation $series(x_1,x_2,x_3)$, where $\{x_1,x_2,x_3\} = \{u,z,v\}$. It follows that the probability variables $\{p_{uvz}\}$ belong to $P(\sigma,R)$. Recall that constraint (i) for clause C is $\pi_C = 1 - p_{\overline{C}}$, and $p_{\overline{C}} \in \{p_{uvz}\}$.

Since we have at most $(n-2)$ series operations, and we can build eight different clauses with variables $\{u,v,z\}$, a 2T-SAT formula contains at most $(8n-16)$ clauses of length three. In conclusion, for a general 2T-SAT instance we generate at most $(36n-57)$ constraints.

We denote by $S(\sigma,R)$ the linear system of equations (i)–(iv) on the non-negative variables $P(\sigma,R)$. Next we show that solving $S(\sigma,R)$ is equivalent to solving PSAT.

Definition 4.1 *Given a solution of the system $S(\sigma, R)$, we say that a probability assignment to possible worlds in $\mathcal{W}^n$ is consistent with the probability variable $p_W \in P(\sigma, R)$ if the value of p_W gives the sum of the probabilities assigned to the possible worlds represented by W.*

Theorem 4.1 *For every solution of the linear system $S(\sigma, R)$, there exists a probability assignment to $\mathcal{W}^n$ consistent with each probability variable in $P(\sigma, R)$.*

Proof:

We use induction on the number of nodes n in G_σ.

For $n = 2$ we have four probabilities: p_{uv}, $p_{u\bar{v}}$, $p_{\bar{u}v}$, $p_{\bar{u}\bar{v}}$. This gives a consistent assignment of probabilities to the four possible worlds. The theorem clearly holds true in this case.

Now assume the theorem is valid if the number of nodes is $n-1$. We prove it to be valid if the number of nodes is equal to n. We only consider the series operation, the proof for the tail operation is similar.

Let u be the first node deleted by the reduction sequence R. Assume that the neighbours of node u are v and z. Recall that there are four equations expressing relations between v, z and u, namely:

$$p_{vz} = p_{uvz} + p_{\bar{u}vz}, \qquad p_{\bar{v}z} = p_{u\bar{v}z} + p_{\bar{u}\bar{v}z}, \qquad p_{v\bar{z}} = p_{uv\bar{z}} + p_{\bar{u}v\bar{z}}, \qquad p_{\bar{u}\bar{z}} = p_{u\bar{v}\bar{z}} + p_{\bar{u}\bar{v}\bar{z}}.$$

Denote by $\mathcal{W}$ the set of $2^{(n-1)}$ possible worlds for the propositions $\mathcal{X} \setminus \{u\}$. ¿From each $W \in \mathcal{W}$ we obtain two possible worlds W_u and $W_{\bar{u}}$ in $\mathcal{W}^n$, where u is set to true and false respectively.

By the induction hypothesis, there exists a probability assignment for $\mathcal{W}$ which is consistent with all the probability variables not involving proposition u. Consider the possible worlds in $\mathcal{W}$ represented by $\{v, z\}$, i.e. where v and z are set to true. The overall probability assigned to these worlds (i.e. p_{vz}) must be split between the possible worlds in $\mathcal{W}^n$ represented by $\{u, v, z\}$ and $\{\bar{u}, v, z\}$, according to equation $p_{vz} = p_{uvz} + p_{\bar{u}vz}$. This can be obtained as follows.

Let α_W be the probability assigned to a possible world $W \in \mathcal{W}$ represented by $\{v, z\}$. We assign to W_u a probability:

$$\alpha_W \frac{p_{uvz}}{p_{vz}},$$

and we assign to $W_{\bar{u}}$ a probability:

$$\alpha_W \frac{p_{\bar{u}vz}}{p_{vz}}.$$

A similar process can be repeated for the possible worlds in $\mathcal{W}$ represented by $\{v, \neg z\}$, $\{\neg v, z\}$, and $\{\neg v, \neg z\}$. In this way we obtain a probability assignment to $\mathcal{W}^n$ which is consistent with the variables $\{p_{uvz}\}$ and with the variables not involving proposition u.

Since the value of any probability variable involving the proposition u is written in terms of the variables $\{p_{uvz}\}$ (see constraints (iv)) this completes the proof.

□

Corollary 4.1 *A probability assignment π to the clauses of a 2T-SAT formula σ is feasible if and only if the system $S(\sigma, R)$ has a solution.*

Result 2: for a 2T-SAT formula $\sigma = (\mathcal{X}, \mathcal{C})$, with $|\mathcal{X}| = n$:

- the PSAT problem can be reduced to solving a system of $O(n)$ equations in $O(n)$ non-negative variables;
- the maximum (minimum) probability for a proposition or a clause can be obtained by solving an LP problem with $O(n)$ equality constraints and $O(n)$ non-negative variables.

5 Conclusions, and further work

In this paper we identified some easy cases of the probabilistic satisfiability problem. In particular, for two classes of formulas, we provided a compact representation of the set of feasible solutions, i.e. the set of consistent probability assignments. This allows to reduce PSAT to solving a system of linear equations or inequalities, whose size is linear in the number of propositional variables. Moreover, finding the maximum (minimum) probability for single clauses or propositions can be reduced to an LP problem with the same compact set of constraints.

It must be remarked that the Max-SAT problem can be solved efficiently for the classes of formulas we consider here (see [17, 7]). Therefore, for these classes a polynomial algorithm for PSAT is available [11]; this algorithm is, however, mainly of theoretical interest. For practical purposes, standard column generation algorithms can be used [16]. By contrast, our methods reduce PSAT to checking the consistency of a (small) set of equations and inequalities. This can be done in polynomial time, e.g. by a polynomial algorithm for linear programming. Still, we could not provide a *purely combinatorial* polynomial algorithm for at least one class of PSAT instances. Actually, such an algorithm has to our knowledge not yet been proposed for any reasonable class of formulas. This will be the subject of further research.

In our opinion, our methods can be extended (up to a certain extent) to wider classes of formulas. One possible direction is the use of partial k-trees with $k > 2$. As discussed in [7], both SAT and Max-SAT can be solved in polynomial time (in n and 2^k) if the co-occurrence graph of the given formula is a partial k-tree. Moreover, besides the co-occurrence graphs, one may consider other representations of formulas, such as the *clause linked graphs* [17].

On the long term, we may consider the following (admittedly ambitious) goal: provide a compact representation of the feasible set for each class where a purely combinatorial, polynomial Max-SAT algorithm is known. A relevant and challenging example are the formulas whose associated hypergraph is a directed graph; for these formulas, Max-SAT can be reduced to a Max-Flow problem [15].

On the other hand, we cannot expect to extend our results to all efficiently solvable Max-SAT classes. For example, consider the formulas represented by balanced hypergraphs; for this class, Max-SAT can be solved by linear programming [6]. However, it is easy to find formulas represented by balanced hypergraphs whose extended graph is not balanced.

In conclusion, probabilistic satisfiability is an interesting area for further research, both from a theoretical and a computational point of view. The results described in this paper seem to confirm that effective solution methods for PSAT require substantially new ideas and ad-hoc techniques that, however, can utilize several tools succesfully applied to satisfiability problems in the past.

References

[1] K. A. Andersen, "Characterizing consistency in probabilistic logic for a class of Horn clauses", *Mathematical Programming* **66** (1994) 257–271.

[2] K. A. Andersen, "On consistency in probabilistic logic for logical formulas represented by B-hypertrees", *European Journal of Operational Research* **108** (1998) 696–709.

[3] K. A. Andersen and J. N. Hooker, "Determining lower and upper bounds on probabilities of atomic propositions in sets of logical formulas represented by digraphs", *Annals of Operations Research* **65** (1996) 1–20.

[4] G. Ausiello and R. Giaccio, "On-line algorithms for satisfiability formulae with uncertainty", *Theoretical Computer Science* **171** (1997) 3–24.

[5] H. L. Bodlaender, "A partial k-arboretum of graphs with bounded treewidth", *Theoretical Computer Science* **209** (1998) 1–46.

[6] M. Conforti and G. Cornuéjols, "A class of logic problems solvable by Linear Programming", *Journal of the Association for Computing Machinery* **42** (1995) 1107–1113.

[7] Y. Crama, P. Hansen and B. Jaumard, "The Basic Algorithm for pseudo-Boolean programming revisited", *Discrete Applied Mathematics* **29** (1990) 171–185.

[8] M. R. Garey, and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman (1979).

[9] G. Gallo, G. Longo, S. Nguyen and S. Pallottino, "Directed Hypergraphs and Applications", *Discrete Applied Mathematics* **42** (1993) 177–201.

[10] G. Gallo, C. Gentile, D. Pretolani and G. Rago, "Max Horn SAT and the Minimum Cut Problem in Directed Hypergraphs", *Mathematical Programming* **80** (1998) 213–237.

[11] G. Georgakopoulos, D. Kavvadias and C. H. Papadimitriou, "Probabilistic Satisfiability", *Journal of Complexity* **4** (1988) 1–11.

[12] T. Hailperin, *Boole's Logic and Probability*, 2nd ed., Studies in Logic and the Foundations of Mathematics vol. 85, North-Holland, Amsterdam (1986).

[13] J. Hooker, "A Mathematical Programming Model for Probabilistic Logic", working paper 05-88-89, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, PA 15213 (1988).

[14] B. Jaumard, P. Hansen and M. P. de Aragão, "Column Generation Methods for Probabilistic Logic", *ORSA Journal on Computing* **3** (2) (1991) 135-148.

[15] B. Jaumard and B. Simeone, "On the complexity of the Maximum Satisfiability Problem for Horn formulas", *Information Processing Letters* **26** (1987/88) 1–4.

[16] D. Kavvadias and C. H. Papadimitriou, "A Linear Programming approach to reasoning about probabilities" *Annals of Mathematics and Artificial Intelligence* **1** (1990) 189–205.

[17] J. Kratochvíl and M. Křivánek, "Satisfiability of co-nested formulas", *Acta Informatica* **30** (1993) 397–403.

[18] N. J. Nilsson, "Probabilistic Logic", *Artificial Intelligence* **28** (1986) 71–87.

[19] D. Pretolani, "Satisfiability and Hypergraphs", PhD. Thesis, TD-12/93, Dipartimento di Informatica, University of Pisa, Italy, 1993.

[20] G. Rago, "Optimization Hypergraphs and Logical Inference", PhD. Thesis, TD-4/94, Dipartimento di Informatica, University of Pisa, Italy, 1993.

[21] K. Truemper, "Alpha-balanced graphs and matrices and GF(3)-representability of matroids", *Journal of Combinatorial Theory B* **32** (1982) 112–139.