On the Relationship between Probabilistic Logic and π-CMS

P. HANSEN
GERAD and Dept. Methodes Quantitatives de Gestion
Ecole des Hautes Etudes Commercials
pierreh@crt.umontreal.ca

B. JAUMARD
GERAD and Dept. Math, and hid. Eng.
Ecole Polytechnique de Montreal
brigittOcrt.umontreal.ca

A. D. PARHEIRA
Dept. Math, and Ind. Eng.
Ecole Polytechnique de Montreal
anderson@crt.umontreal.ca

Abstract

We discuss the relationship between probabilistic logic and π-CMS. Given a set of logical sentences and their probabilities of being true, the outcome of a probabilistic logic system consists of lower and upper bounds on the probability of an additional sentence to be true. These bounds are computed using a linear programming formulation. In π-CMS systems, the outcome is defined by the probabilities of the support and the plausibility (with the assumptions being independent) after a first phase which consists of computing the prime implicants depending only on the variables of the assumptions. We propose to reformulate a π-CMS system without independence conditions on the assumptions, using the linear programming framework of probabilistic logic, and show how to exploit its particular structure to solve it efficiently. When an independence condition is imposed on the assumptions the two systems give different results. Comparisons are made on small problems using the assumption-based evidential language program (ABEL) of [Anrig *et al.*, 1998] and the PSAT program of [Jaumard *et al.*, 1991].

1 Introduction

Many different models have been proposed for reasoning under uncertainty in expert systems. Among those based on the use of logic and probability theory which require a moderate amount of input data, two important families are the probabilistic Clause Maintenance Systems (TT-CMS) [Reiter and de Kleer, 1987; deKleer and Williams, 1987], and the probabilistic logic models (Nilsson, 1986; Jaumard et ol., 1991; Hansen *et al.*, 1995]. While these families have been developed separately, they have much in common, with however the difference that models of the former family usually suppose independence of their assumptions, while those of the latter do not. In this paper, we explore π-CMS systems, with and without independence of the assumptions, from the probabilistic logic viewpoint. The main results are the following: (i) when relaxing the independence condition, two formulations can be proposed for π-CMS systems; (ii) the second formulation suggests an algorithm which exploits the particular structure of π-CMS and is, on large instances, 7000 times faster than the column generation algorithm for probabilistic logic of (Jaumard *et al.*, 1991] applied to the first formulation; (iii) with the independence condition, probability intervals of an additional proposition obtained by π-CMS may overlap with those of the corresponding probabilistic logic model, which suggests that π-CMS systems do not exploit the available information to its fullest degree.

The paper is organized as follows: probabilistic logic models and π-CMS systems are briefly reviewed in the next two sections. The two probabilistic logic models for π-CMS systems are presented in Section 4. Algorithms are discussed in Section 5 and computational results in Section 6. Brief conclusions are drawn in the last section.

2 Probabilistic Logic

Nilsson [Nilsson, 1986] has presented a semantical generalization of propositional logic in which the truth values of sentences are probability values *(probabilistic logic,* also called *probabilistic satisfiability* or PSAT for short).

Let $X = \{x_1, x_2, \ldots, x_n\}$ denote a set of propositional variables and $S = \{S_1, S_2, \ldots, S_m\}$ a set of propositional sentences over X defined with the usual operators $\vee$ (disjunction), $\wedge$ (conjunction), and -. (negation). A literal y_k is a propositional variable $(y_k = x_k)$ or its negation $(y_k = \neg x_k)$. Propositional sentences are assumed to be written using a DNF (Disjunctive Normal Form) expression in this paragraph.

Let $w = (w_1, \ldots, w_m)$ be a valuation for 5, where w_i is equal to 1 if S_i has value true, and to 0 otherwise, w is a *possible world* if there exists a truth assignment over X which leads to w over S and it is an *impossible world* otherwise. Observe that two different truth assignments on X may lead to the same possible world. Let W denote the set of possible worlds and set k to $|W|$ (note that $k \leq$

2^n). To illustrate, if $X = \{x_1, x_2\}$ and $S = \{x_1, x_1 \vee x_2\}$, we have $W = \{w^1 = (0,0); w^2 = (0,1); w^3 = (1,1)\}$, $k = 3 < 2^2$, and $w^4 = (1,0)$ is an impossible world.

Let $p = (p_1, p_2, \ldots, p_k)$ be a probability distribution on W, with $0 \leq p_j \leq 1$ $(j = 1, \ldots, k)$ and $\sum_j p_j = 1$. Let $\pi = (\pi_1, \ldots, \pi_m)$ be a probability vector such that π_i denotes the probability of sentence S_i in S, $i = 1, \ldots, m$.

The probability distribution p is *consistent* if it satis* fies the set of logical sentences together with their probabilities, i.e., if for each sentence S_i, the sum of p_j's over all truth assignment w^j that satisfy S_i (i.e., $w_i^j = 1$) equals π_i.

For a given set S of sentences, let A^S be an $m \times k$ matrix such that a_{ij}^S is equal to 1 if S_i is true for w^j, and equal to 0 otherwise. This leads to the following linear programming formulation of the PSAT problem (in decision form):

$$\begin{aligned} \mathbb{1}\, p &= 1 \\ A^S p &= \pi \\ p &\geq 0 \end{aligned} \qquad \text{(i)}$$

where 1 is a unit row k-vector. The problem PSAT thus reduces to determining whether there exists or not a vector p satisfying (1). Observe that (PSAT) is completely determined by the pair (S, π).

Let us assume that the PSAT problem defined by (1) is consistent. Let S_m+1 denote an additional logical sentence, with an unknown probability π_{m+1}. The optimization form of PSAT (also known as *probabilistic* en*tailmenty* see [Nilsson, 1986]), is to determine the range $[\underline{\pi}_{m+1}, \overline{\pi}_{m+1}]$ of values of the probability π_{m+1} such that $(S \cup \{S_{m+1}\}, (\pi, \pi_{m+1}))$ is consistent. Consider the objective function $A_{m+1}p$ (with $A_{m+1} = (a_{m+1,j})$, where $a_{m+1,j}$ is equal to 1 if S_{m+1} is true for the possible word w^j and equal to 0 otherwise). PSAT corresponds to determining $\underline{\pi}_{m+1} = \min \{A_{m+1}p : \text{constraints (1)}\}$ and $\overline{\pi}_{m+1} = \max \{A_{m+1}p : \text{constraints (1)}\}$.

2.1 Numerical solution of PSAT

The linear program which expresses the PSAT problem in its decision or optimization form has an exponential number of variables in the size of the input; for this reason, in the sequel we will speak of Generalized Linear Programming (GLP for short) formulation. So writing explicitly PSAT already requires exponential time. This led Nilsson [Nilsson, 1986] to suggest looking for heuristic solution methods only.

Such a view is overly pessimistic since, although writing large PSAT problems explicitly is impossible, they can be solved quite efficiently by keeping them implicit. This can be done using the so-called column generation technique [Lasdon, 1970]. It extends the revised simplex method, in which only a small number of columns are kept explicitly. This technique makes use of a master problem which implies an objective and constraints related to the probabilities of the possible worlds, and a subproblem which determines the entering column for the master problem. This last subproblem depends on the structure of the original problem and has quite often a combinatorial nature. For more details see [Jaumard *et al.y* 1991].

3 CMS and π-CMS models

The Assumption-based Truth Maintenance Systems (or ATMS for short) [de Kleer, 1986] and later the CMS (Clause Maintenance Systems) [Reiter and de Kleer, 1987] can be viewed as symbolic algebra systems for producing a set of statements (Boolean expressions) in which one can believe. A brief description of the CMS is given below; details can be found in [Reiter and de Kleer, 1987].

Given a set of propositional variables X, a CMS consists of a set of *assumptions* $A = \{a_1, a_2, \ldots, a_m\}$ supposed to be such that $A \subset X$ and a set of propositional sentences $= \{h_1, h_2, \ldots, h_r\}$ (also called *justifications,* see [de Kleer, 1986J) such that the h_i are clauses. Propositional sentences are assumed to be written using a CNF (Conjunctive Normal Form) expression in this paragraph. Therefore, a clause is a finite disjunction of literals with no literal repeated, also represented as a set of literals.

A clause $\mathcal{L}(h)$ is a support for a clause h with respect to H iff $H \not\models \mathcal{L}(h)$ (i.e., $H \cup \{\neg\mathcal{L}(h)\}$ is satisfiable) and $H \models \mathcal{L}(h) \cup h$ (i.e., $H \models \neg\mathcal{L}(h) \supset h$). $\mathcal{L}(h)$ is a minimal support for h with respect to H iff no proper subset of $\mathcal{L}(h)$ is a support for h with respect to H. The CMS is a database management system which, given a set H of propositional sentences computes a minimal support clause, $\mathcal{L}(h)$, for a clause h. $\mathcal{L}(h)$ summarizes a list of sets of nonredundant assumptions (in terms of a Boolean formula), each of which is sufficient to support a proof of h. The $\mathcal{L}(h)$ for a clause h is related with a prime implicant for H. This relationship is given as follows [Reiter and de Kleer, 1987]: $\mathcal{L}(h)$ is a minimal support clause for h with respect to H iff there is a prime implicant σ for H such that $h \cap \sigma \neq \varnothing$ and $\mathcal{L}(h) = \sigma \setminus h$. Conversely, if σ is a prime implicant for H such that $h \subseteq \sigma$ then $\sigma \setminus h$ is a minimal support clause for h with respect to H.

3.1 CMS Algorithms

The prime implicants can be used to implement CMSs [Reiter and de Kleer, 1987]. The prime implicants of a set of clauses can be computed by repeatedly resolving pairs of clauses, adding the resulting resolvents to the set and removing subsumed clauses. This method is known to be a brute-force approach and performs far more resolution steps than necessary. The running time of this algorithm depends on the number and expense of the subsumption checks required. De Kleer [de Kleer, 1992] describes an improved algorithm for generating the prime implicants of a set of clauses.

3.2 Adding Numerical Uncertainties

From a logical viewpoint, the $\mathcal{L}(h)$ that the ATMS/CMS attaches to a clause yield only three possible truth values for h: believed, disbelieved, and unknown. This

three-value logic cannot rate the degree of imcertainty associated with unknown clauses, and thus may lead to a stalemate whenever a decision is to be made whose outcome depends critically on the truth of these propositions. For this and others reasons, several attempts have been made to augment ATMS/CMS with a numerical measure of uncertainty. Several authors use probability theory to deal with uncertainty associated with assumptions for ATMS (De Kleer and Williams [de Kleer and Williams, 1987] and Liu *et al.* *[Liu* a/., 1993]) or CMS (Kohlas *et al.* [Kohlas and Haenni, 1996]). In all extensions the probabilities assigned to assumptions must be independent (see [Hansen *et al*., 1999] for details) in order to calculate the degree of belief.

The quantitative judgment of belief can be measured by the degree of support and plausibility. The degree of support of *h (dsp* for short) is the probability of arguments $(\mathcal{L}(h))$ in favor of it. Similarly, the degree of plausibility *(dpl* for short) of h is 1 minus the probability of arguments $(\mathcal{L}(\neg h))$against it.

As next shown, π-CMS without independence condition can be reformulated using the linear programming expression of the PSAT problem.

4 Using PSAT Linear Programming expression to reformulate a π-CMS model

Consider the PSAT problem $(S,\ \pi')$ defined as follows: $S = A \cup H$ where $S_i = a_i\ (i = 1, 2, \ldots, m)$ and $S_{i+m} = h_i\ (i = 1, 2, \ldots, r)$; $\pi' = (\pi, \mathbb{1})$, where π is a probability m-vector associated with the set A of assumptions and 1 is a unit r-vector (all the probabilities are equal to 1 for the set H of clauses). Note that $|S| = m'$ with $m' = m + r$. The clause h with unknown probability is associated with $S_{m'+1}$. Denote by $Y_i \subseteq Y$ the set of literals of clause h_i, i.e., $h_i = \bigvee_{y_j \in Y_i} y_j$.

First Formulation (F1)

The π-CMS problem can be expressed using the GLP formulation of the PSAT problem;

Master problem:

$$\min/\max\ A_{m'+1}p \tag{2}$$

subject to:

$$\mathbb{1}\,p = 1 \tag{3}$$
$$A^A p = \pi \tag{4}$$
$$A^H p = \mathbb{1} \tag{5}$$
$$p \geq 0 \tag{6}$$

Subproblem:

$$\min/\max_j \bar{c}_j = c_j - \left(1, (A^A)^j, (A^H)^j\right)(u_0, u_A, u_H) \tag{7}$$

where u_0, u^A and u^H are the dual variables associated with the constraints (3)-(5), respectively.

This reformulation results in a PSAT problem with a subproblem corresponding to an unconstrained nonlinear program in 0-1 variables.

Note that the set of prepositional sentences of if has a specific structure, in that its probabilities are all equal to 1, i.e, these propositiona! sentences are always true. Exploring this structure leads to a second formulation.

Second Formulation (Fa)

There is a natural partitioning of the constraints of the GLP defined by formulation *F1* Observe that satisfying the constraint (5) is equivalent to satisfying the equation:

$$\bigwedge_{i=1,\ldots,r} h_i = 1,$$

i.e., to solving a satisfiability problem (SAT).

Hence, constraints (5) can be transferred to the subproblem, which then leads to consider in the master problem only the solutions of (5) which satisfy (3), (4), (6) and minimize or maximize (2).

We thus obtain the following GLP reformulation.

Master problem:

$$\min/\max\ A_{m'+1}p \tag{8}$$

subject to:

$$\mathbb{1}\,p = 1 \tag{9}$$
$$A^A p = \pi \tag{10}$$
$$p \geq 0 \tag{11}$$

Subproblem:

$$\begin{aligned} \min/\max_j \bar{c}_j &= c_j - (1, (A^A)^j)(u_0, u^A) \\ \text{subject to:} & \\ \sum_{y_j \in Y_i} y_j &\geq 1 \qquad i = 1, 2, \ldots, r \\ x_i + \bar{x}_i &= 1 \qquad x_i \in X \end{aligned} \tag{12}$$

The subproblem is now a linear program in the 2n 0-1 variables x_i, $\bar{x}_i$ (as the y_j are either variables x_l or $\bar{x}_l$) with linear constraints.

Each column of (10) corresponds to a possible world w^j, or equivalently a subset of assumptions with true/false values, which may imply $S_{m'+1}$, imply $\neg S_{m'+1}$ or neither. In the first two cases there is one corresponding column in (8)-(II), and in the latter two. When minimizing $A_{m'+1}p$, columns which have $a_{m'+1,j} = 1$ will have probability 0 if there are twin columns with $a_{m'+1,j'} = 0$. So π_{m+1} will correspond to the sum of probabilities of those clauses which imply $S_{m'+1}$, i.e., to the degree of support of $S_{m'+1}$. When maximizing $A_{m'+1}p$, columns which have $a_{m'+1,j} = 0$ will have probabilities 0 if there are twin columns with $a_{m'+1,j'} = 1$. So $\bar{\pi}_{m+1}$ will correspond to 1 minus the sum of probabilities of those clauses which imply $\neg S_{m'+1}$, i.e., to the degree of plausibility of $S_{m'+1}$

5 Solution Method

The method is an iterative one (in fact, the revised simplex algorithm [Chvatal, 1983]) where at each iteration, we determine the column with the minimum (maximum) reduced cost by solving (7) in the case of F_1, ie,

$$\min/\max_j \bar{c}_j = a_{m'+1,j} - u_0 - \sum_{i=1}^{m'} u_i a_{ij} \qquad (13)$$

where the u_0 and ui are the current dual variables associated with the tautology *So* and with the propositional sentences S_i, $i = 1, \ldots, m'$, respectively. By associating the logical values true with 1 and false with 0 in each logical proposition, one can rewrite (13) as:

$$\min/\max_j \bar{c}_j = S_{m'+1} - u_0 - \sum_{i=1}^{m'} u_i S_i.$$

This optimization problem can be reformulated using arithmetic expressions of the propositional variables of *X.* This can easily be done with the convention 1 = true and 0 = false and the relations $x_i \vee x_j \equiv x_i + x_j - x_i \times x_j$, $x_i \wedge x_j \equiv x_i \times x_j$ and $\neg x_i \equiv 1 - x_i$, where the left-hand side variables are logical ones, and the right-hand side variables are integer ones. The choice of the entering column is thus reduced to a problem of minimizing (maximizing) an unconstrained nonlinear function in 0-1 variables (PNL-0/1). In formulation F2 the subproblem is a constrained linear program in 0-1 variables (CPL-0/1).

5.1 Basic Algorithm

We assume that the optimization problem is a minimization one. Algorithm 1 provides a description of the column generation method for Formulation *F* or F2.

Phase-1 is not detailed as it proceeds in a similar way than Phase-2, i.e., through STEP 3 to STEP 7.

5.2 Solution of the Subproblems

Again assume that the optimization problem is a minimization one. At each iteration a column with a negative reduced cost must be found by a heuristic or an exact algorithm. Exact solution of the subproblem is not necessary at each iteration to guarantee convergence. As long as a negative reduced cost is found by the heuristic an iteration of the revised simplex algorithm may be done. If a feasible solution is obtained in that way, the decision version of the satisfiability problem is solved, but finding none when choosing the entering column in a heuristic way cannot guarantee that none exists. So, when no more negative reduced cost is obtained by the heuristic it is necessary to turn to an exact algorithm to prove that there is no feasible solution for the decision version of PSAT, nor feasible solution giving a better bound than the incumbent one for the optimization version. We use a Steepest Ascent Mildest Descent (SAMD) or Tabu Search (TS) heuristic ((Hansen and Jaumard, 1990; Glover and Laguna, 1997]) for PNL-0/1, and a variant of Tabu Search for CPL-0/1 with constraints to find an approximately optimal solution. We also use a method based on linearization for PNL-0/1, and branch-and-bound for CPL-0/1, to find an exact optimal solution.

Algorithm 1 Column Generation Method

STEP 1. Using phase-I of the simplex algorithm (see, e.g. [Chvatal, 1983] for details), build an initial matrix B_0 associated with a feasible solution.

STEP 2. If there is no such matrix Bo then the problem is inconsistent: Stop.

STEP 3. Solve the master problem to compute the dual variables (phase-2 of the simplex algorithm).

STEP 4. Solve the subproblem to compute a column with negative reduced cost c_t using a heuristic;

STEP 5. If $\bar{c}_t < 0$ then add the corresponding column to the master problem, reoptimize it, and go to STEP 3.

STEP 6. Solve the subproblem to compute the column with the most negative reduced cost using an exact algorithm.

STEP 7. If $\bar{c}_t < 0$ then add the corresponding column to the master problem, reoptimize it, and go to STEP 3.

STEP 8. Stop: the optimal solution has been reached.

PNL-0/1 without constraints

The Tabu Search (TS) heuristic proceeds to a local optimum by moving at each iteration from one feasible solution to another in its neighborhood (here the vectors at Hamming distance 1 from the current one). We then pick the solution in the neighborhood that produces the best improvement in the objective function. If there is no improving solution (a local optimum was reached assuming we use aspiration, see page 50 in [Glover and Laguna, 1997]) then we choose that one which degrades the objective function least. In order to avoid returning to the local optimum just visited, or cycling, the reverse move is forbidden for a given number of iterations *(SizeMaxTabu).*

The steps of TS are presented in Algorithm 2. The parameters T_j denote the remaining number of iterations during which a local change in direction j is forbidden. When a change in an ascent direction j is done T_j is set at the value *tabu.* That value is chosen by random selection among integers in the range (1, *SizeMaxTabu].* The stopping condition may be, e.g., maximum computing time allowed, maximum number of iterations, or maximum number of iterations between two improvements.

The linearization method works by replacing in a standard way each product of variables by a new 0-1 variable and adding constraints to ensure that the values agree in 0-1 variables.

The size of the resulting linear 0-1 variables increases rapidly with m, n and the number of nonzero dual variables u_i.

Algorithm 2 Tabu Search

STEP 1. Select a feasible solution x and let $z(x)$ be the value associated with x. Set $x_{opt} := x$, $z_{opt} := z(x)$, and T empty, the directions where changes are forbidden. Choose a stopping condition.

STEP 2. Transform x into the solution x^j modified in the j^{th} direction and evaluate $\delta_j := z(x_j) - z(x)$ with $T_j = 0$. Let $\delta_k := min\{\delta_j : T_j = 0\}$. If $\delta_k \leq 0$, $T_k := tabu$. Set $x := x_k$.

STEP 3. If $z(x_k) < z_{opt}$ then update $x_{opt} := x_k$ and $z_{opt} := z(x_k)$.

STEP 4. If the stopping condition is met then STOP. Otherwise, update T and return to STEP 2.

Algorithm 3 Tabu Search-Constrained

STEP 1. Select a solution x (not necessarily feasible) and let $z(x)$ denote the value associated with x. Set $x_{opt} := x$, $z_{opt} := z(x)$, $iter := 1$ and T empty. Choose the values of $UnsatMax$, REP and a stopping condition.

STEP 2. Transform x into the solution x^j modified in the j^{th} direction and evaluate $\delta_j := z(x^j) - z(x)$ with $T_j = 0$. Let $\delta_k := \min\{\delta_j : T_j = 0 \text{ and } unsat < UnsatMax\}$. If $\delta_k \leq 0$, $T_k := tabu$. Set $x := x^k$.

STEP 3. If $z(x^k) < z_{opt}$ and x^k feasible then update $x_{opt} := x^k$ and $z_{opt} := z(x^k)$.

STEP 4. If the stopping condition is met then Stop. Otherwise, update T and return to STEP 2.

STEP 5. If x^k is feasible then $UnsatMax := UnsatMax - \beta * UnsatMax$. Otherwise $UnsatMax = \alpha * UnsatMax$.

STEP 6. $iter := iter + 1$.

STEP 7. If $iter < REP$ then return to STEP 2. Otherwise Stop.

CPL-0/1 with constraints

Heuristic solution of CPL-0/1 is again done by a SAMD or TS algorithm. The idea behind this algorithm is to allow infeasibility. However it must be limited, therefore for each sequence of iterations a move is allowed only if the resulting number of unsatisfied clauses (or violated constraints) is smaller than a given threshold *UnsatMax*. After a number of iterations we verify if the current solution is feasible. In the affirmative we reduce the value of *UnsatMax* by β and another sequence of iterations is performed. In the negative the value of *UnsatMax* is increased by a, we apply a transformation to get a feasible solution and another sequence of iterations is performed. The overall algorithm has a maximum number of *REP* iterations. Steps of the TS-Constrained heuristic are presented on Algorithm 3.

The constrained linear 0-1 program (12) has the form of a combiped set covering and set partitioning problem. Exact solution is based on the algorithm of [Fisher and Kedia, 1990]. The main feature of this algorithm is the use of dual variables in the linear programming relaxation to provide lower bounds for a branch-and-bound algorithm.

6 Computational Experiments

The algorithms for π-CMS/PSAT described in the previous sections have been coded in C and tested on a Ultra-2 SUN SPARC computer. The resulting program uses the CPLEX 6.0 package for linear programming. The problems for π-CMS/PSAT were randomly generated in the following way: (i) the numbers m of assumptions, r of justifications, which here are clauses, n of variables (assumptions and propositional symbols) are parameters; (ii) the clauses contain at most 4 literals. There is an uniform distribution of clauses with 1, 2, 3 and 4 literals as well as of uncomplemented and complemented variables. Probabilities $\pi_i(A)$ corresponding to feasible problems were obtained by generating randomly max(2^{10};2^n) Boolean vectors x, constructing the corresponding possible worlds, associating with them positive numbers summing up to 1 (i.e., probabilities for these worlds to occur), and then summing for each clause the probabilities of the worlds in which it is true.

Results of the comparison between the two π-CMS/PSAT models are given in Table 1 and 2, which detail problem sizes, total cpu time, number of columns generated, cpu time for the tabu search and the exact algorithms for the subproblems. Each line corresponds to averages over five problems. Note than the number of propositions m' in the first formulation is equal to the number of assumptions (propositions in the master problem) plus the number of justifications (considered only in the subprobiems) in the second formulation. Clearly the second formulation is by far superior to the first one: computation times for the larger instances are 7000 times smaller.

If an independence condition is imposed on the assumptions, only much smaller instances of π-CMS are solvable in reasonable time. If columns corresponding to feasible worlds are considered explicitly, each has a positive probability and their number increases exponentially. This contrast with the linear programming model of PSAT, where the number of columns with positive probability in a basic, and hence in an optimal, solution is at most equal to the number of lines. Using Boolean simplifications we may reduce somewhat the computational burden but it often remains too high to solve large instances.

Intervals $[\pi_{m+1}, \pi_{m+1}]$ π-CMS/PSAT and

Problems		tcpu		Column		TS Heuristic			Exact Algorithm		
n	m'	tcpu μ (s)	σ	μ	σ	tcpu μ (s)	σ	Iter	tcpu μ (s)	σ	Iter
Maximization+Minimization											
100	100	17.278	6.376	712.80	140.599	1.96	0.41	714.80	0.01	0.01	3.60
100	150	3307.274	743.309	3863.00	255.948	12.66	0.95	3865.00	0.01	0.03	3.20
150	200	10890.038	3779.051	4464.00	208.784	22.51	1.70	4466.00	0.01	0.02	3.40

Table 1: Formulation-1

Problems			tcpu		Column		TS Heuristic			Exact Algorithm		
n	m	r	tcpu μ (s)	σ	μ	σ	tcpu μ (s)	σ	Iter	tcpu μ (s)	σ	Iter
Maximization+Minimization												
100	50	50	0.532	0.043	20.20	1.939	0.41	0.03	22.20	0.05	0.15	2.20
100	50	100	0.984	0.183	26.80	4.308	0.82	0.12	28.80	0.08	0.21	2.80
150	100	100	1.532	0.136	32.80	3.370	1.24	0.11	34.80	0.09	0.22	2.20

Table 2: Formulation-2

[dap, dpi) for π-CMS with independence condition (obtained with the ABEL program [Anrig *et al,* 1998]) are given in Table 3. It appears that in some cases the intervals of π-CMS with the independence conditions overlap in part with those of π-CMS/PSAT instead of being included in them, as one would have expected. This indicates π-CMS does not exploit the available information to its fullest degree. Reasons why this is so will be explored in future work.

Problems			PSAT	ABEL
n	m	r	$\underline{\pi}_{m+1}$-$\overline{\pi}_{m+1}$	*dsp-dpl*
15	10	20	0.0000-0.5513	0.000-0.313
15	10	20	0.1046-1.0000	0.037-1.000
15	10	20	0.3666-1.0000	0.473-1.000
15	10	20	0.4301-1.0000	0.601-1.000

Table 3: Comparison of bounds

7 Conclusions

π-CMS systems can be expressed in two ways as a PSAT problem, when the independence condition on the assumptions is removed. Then the lower and upper bounds on the probability of the additional propositional sentence coincide with its support and plausibility. The second formulation, which keeps the justifications implicit (or transfer them to the subproblem when using column generation as solution method), is much more efficient than the first, direct one. If the independence condition is kept in π-CMS, solution becomes more cumbersome and probability intervals for the objective function clause provided by both methods may overlap.

Acknowledgments

Research of the authors was supported by DREV-Valcartier contract. Work of the first and second authors was supported by FCAR grants 92-EQ-1048 and 95-ER-1048. Work of the first has also been supported by NSERC grant to HEC and NSERC grant GP0105574. Work of the second author was also supported by NSERC grants GP0036426 and EQP0157431. Work of the third author has also been supported by CNPq (Conselho Nacional de Desevolvimento Cientifico e TecnoWgico), Brazil, grant 200722/95-6.

References

[Anrig *et al,* 1998] Bernhard Anrig, Roman Bissig, Itolf Haenni, Jurg Kohlas, and Nobert Lehmann. Probabilistic Argumentation Systems: Introduction to Assumption-Based Modeling using ABEL. Technical report, Institute of Informatics, University of Fribourg, September 1998.

[Chvdtal, 1983] Vasek Chvatal. *Linear Programming.* Freeman, New York, NY, 1983.

[de Kleer and Williams, 1987] Johan de Kleer and Brian C. Williams. Diagnosing Multiple Faults. *Artificial Intelligence,* 32(1):97-130,1987.

[de Kleer, 1986] Johan de Kleer. An Assumption-based TMS. *Artificial Intelligence,* 28:127-162, 1986.

[de Kleer, 1992] Johan de Kleer. An Improved Incremental Algorithm for Generating Prime Implicates. In *Proc. of the 10th Nat. Conf. on Artificial Intelligence,* pages 780-785, San Jose, California, July 1992. AAAI.

[Fisher and Kedia, 1990] Marchall L. Fisher and Pradeep Kedia. Optimal Solution of Set Covering/Partitioning Problems Using Dual Heuristics. *Management Science,* 36(6):674-688, June 1990.

[Glover and Laguna, 1997] Fred Glover and Manuel Laguna. *Tabu Search.* Kluwer, 1997.

[Hansen and Jaumard, 1990] Pierre Hansen and Brigitte Jaumard. Algorithms for the Maximum Satisfiability Problem. *Computing,* 44:279-303,1990.

[Hansen *et al.,* 1995] Pierre Hansen, Brigitte Jaumard, and Marcus V. Poggi de Aragao. Boole's Conditions

of Possible Experience and Reasoning Under Uncertainty. *Discrete Applied Mathematics,* 60:181-193, 1995.

[Hansen et al., 19991 Pierre Hansen, Brigitte Jaumard, Anderson D. Parreira, and Marcus V. Poggi de Aragao. Difficulties of Conditioning and Conditional Independence in Probabilistic Satisfiability. Technical Report G-99-16, Cahiers du GERAD, February 1999.

[Jaumard *et al,* 1991] Brigitte Jaumard, Pierre Hansen, and Marcus V. Poggi de Aragao. Column Generation Methods for Probabilistic Logic. *ORSA Journal on Computing,* 3:135-148,1991.

[Kohlas and Haenni, 1996] Jurg Kohlas and Rolf Haenni. Assumption-based Reasoning and Probabilistic Argumentation Systems. In J. Kohlas and S. Moral, editors, *Defeasible Reasoning and Uncertainty Management Systems: Algorithms.* Oxford University Press, 1996.

[Lasdon, 1970] Leon S. Lasdon. *Optimization Theory for Large Systems,* MacMillan, New York, 1970.

[Liu *et ai,* 1993] Weiru Liu, Alan Bundy, and Dave Robertson. On the Relations between Incidence Calculus and ATMS. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty (Granada, 1993),* pages 249-256, Springer-Verlag, Berlin, 1993.

[Nilsson, 1986] Nils J. Nilsson. Probabilistic Logic. *Artificial Intelligence,* 28:71-87, 1986.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of Assumption-Based Truth Maintenance Systems: Preliminary Report. In *Proc. of the 6th Nat. Conf. on Artificial Intelligence,* volume 1, pages 183-188, Seatle, Washington, July 1987.