DISTANCE-SAT: Complexity and Algorithms

Olivier Bailleux and **Pierre Marquis**
CRIL/Université d'Artois
rue de l'Université - S.P. 16
62307 Lens Cedex - FRANCE
e-mail: {bailleux, marquis}@cril.univ-artois.fr

Abstract

In many AI fields, the problem of finding out a solution which is as close as possible to a given configuration has to be faced. This paper addresses this problem in a propositional framework. The decision problem DISTANCE-SAT that consists in determining whether a propositional CNF formula admits a model that disagrees with a given partial interpretation on at most d variables, is introduced. The complexity of DISTANCE-SAT and of several restrictions of it are identified. Two algorithms based on the well-known Davis/Putnam search procedure are presented so as to solve DISTANCE-SAT. Their empirical evaluation enables deriving firm conclusions about their respective performances, and to relate the difficulty of DISTANCE-SAT with the difficulty of SAT from the practical side.

Introduction

In many AI fields, the problem of finding out a solution which is close as possible to a given configuration must be faced. Such a configuration typically encodes some form of preference knowledge (e.g., an expected state, or a normal state) that conflicts with the hard constraints of the problem, represented as a knowledge base. For instance, in the consistency-based diagnosis framework (Reiter 1987), the expected state of the components of a device is the one where none of them is faulty. Whenever a failure occurs, such a diagnosis no longer is possible: assuming that every component behaves as its model of correct behaviour requires it conflicts with the observations that have been made. In this situation, this assumption must be revised (some components are to be assumed faulty) so as to restore consistency. Since many fault assumptions can typically be made in order to achieve this goal, a principle of parsimony is often adopted: among the possible diagnoses, the selected ones are those including a minimal set (w.r.t. cardinality or set-inclusion) of faulty assumptions. Thus, the diagnoses that are "not so far" from the expected one are preferred to the remaining ones: one-fault diagnoses are first considered, then two-faults diagnoses, and so on.

In this paper, this problem is addressed within a propositional framework. The knowledge base is represented as a propositional CNF formula Σ, the expected configuration as a partial interpretation PI, and we are interested in finding out a model of Σ that disagrees with PI on at most d variables. We call DISTANCE-SAT the corresponding decision problem.

In the following, the complexity of DISTANCE-SAT is identified in the general case and in some restricted cases. Like the well-known SAT problem (which can be viewed as a restriction of it), DISTANCE-SAT is NP-complete. However, DISTANCE-SAT is somewhat more difficult than SAT, in the sense that the tractable restrictions for SAT do not typically give rise to tractable restrictions for DISTANCE-SAT.

Then, two algorithms for solving DISTANCE-SAT are presented. The first one, $DP_{distance}$, is a straightforward adaptation of the Davis/Putnam search procedure. To every node of the search tree is associated a value that measures the disagreement between the given configuration and the partial interpretation that corresponds to the node (and can be read off directly by picking up the literals from the branch that ends up to the node under consideration). Whenever this value exceeds the given maximal bound d, the algorithm backtracks. Our second algorithm, $DP_{distance+lasso}$, is a variant of $DP_{distance}$. The only difference between them lies in the branching rule. While the branching rule used in $DP_{distance}$ is a standard, "efficient", branching rule for SAT, the branching rule used in $DP_{distance+lasso}$ is much more oriented towards the satisfaction of the distance constraint. The objective is to lasso in priority a model that is close to the given configuration. Thus, among the clauses that are completely falsified by the given configuration, those of minimal length are considered. Among the variables of these clauses, one of those that maximize the standard branching rule heuristic (used in $DP_{distance}$) is selected as the branching variable.

Both algorithms are empirically assessed on many random 3-CNF instances (generated using the now classical "fixed-length clauses model" (Chvátal & Szemerédi 1988)), for several values of the ratio number of clauses/number of variables, for several sizes of the given configuration, and several values of the maximal disagreement number d. When d is small, $DP_{distance+lasso}$ performs much better then $DP_{distance}$. Contrastingly, when d is large, $DP_{distance}$ is the best performer.

The rest of this paper is organized as follows. Section 2 gives some formal preliminaries. Section 3 presents DISTANCE-SAT and its computational complexity. Two algorithms for solving DISTANCE-SAT are given in Section 4. Section 5 presents an empirical evaluation of these algorithms. Section 6 concludes this paper.

Formal Preliminaries

Let $PROP_{PS}$ denote the propositional language built up from a denumerable set PS of propositional symbols (also called variables) and the connectives in the standard way. The elements of $PROP_{PS}$ are called formulas. The size of a formula Σ, noted $|\Sigma|$ is the number of signs (symbols and connectives) used to write it. $Var(\Sigma)$ is the set of propositional variables occurring in Σ.

Formulas are interpreted in the classical way. An interpretation of a formula Σ is a mapping I that associates *every* propositional variable of $Var(\Sigma)$ to one of the two truth values of $BOOL = \{true, false\}$. A partial interpretation of Σ is a mapping PI that associates *some* propositional variables of $Var(\Sigma)$ to one of the two truth values of $BOOL$. $Dom(PI) \subseteq Var(\Sigma)$ denotes the domain of PI. A complete partial interpretation is just an interpretation. In the following, (partial) interpretations are represented as sets of literals. A positive literal x (resp. a negative literal $\neg x$) appears in PI iff $PI(x) = true$ (resp. $PI(x) = false$). An interpretation I is an extension of a partial interpretation PI iff $PI \subseteq I$ holds. A clause is said completely falsified by a partial interpretation whenever every literal of the clause appears in the partial interpretation with the opposite sign.

A k-CNF of a formula is a CNF formula in which every clause contains at most k literals. A formula is Horn CNF (resp. reverse Horn CNF) iff it is a CNF formula s.t. every clause in it contains at most one positive (resp. negative) literal. A Krom formula is a 2-CNF formula, i.e., every clause in it contains at most two literals.

We assume that the reader is familiar with some basic notions of computational complexity (see e.g., (Garey & Johnson 1979)).

Definition and Complexity

Before defining DISTANCE-SAT in a formal way, we first need the definition of disagreement between two partial interpretations:

Definition 1 (disagreement)
A partial interpretation PI_1 is said to disagree *with a partial interpretation PI_2 on at most d variables iff the number of variables x of $Dom(PI_1) \cap Dom(PI_2)$ s.t. $PI_1(x) \neq PI_2(x)$ is less than or equal to d.*

We are now ready to define DISTANCE-SAT.

Definition 2 (DISTANCE-SAT)
DISTANCE-SAT *is the following decision problem:*

- **Input:** *A CNF formula Σ, a partial interpretation PI, and a non-negative integer d.*
- **Question:** *Does there exist a model I of Σ s.t. I disagrees with PI on at most d variables?*

For every instance of DISTANCE-SAT, we call the constraint "I disagrees with PI on at most d variables" its *distance constraint*.

DISTANCE-SAT is closely related to the problem of repairing a supermodel (Ginsberg, Parkes, & Roy 1998). A (S_1^a, S_2^b)-supermodel of a propositional formula Σ is a model I of Σ s.t. if the variables of any subset of S_1^a of size at most a are flipped in I, a model of Σ can be obtained by flipping in I the variables of a disjoint subset of S_2^b of size at most b. Let L_1^a be a set of literals of size at most a s.t. every literal l from it is built up from a variable from S_1^a and $l \notin I$. Repairing I when it is modified as indicated by L_1^a consists in finding out a model of Σ simplified by L_1^a that disagrees on at most b variables with the restriction of I to the variables of S_2^b. Accordingly, our algorithms for DISTANCE-SAT can directly be used to determine such repairs.

Proposition 1 (complexity of DISTANCE-SAT**)**
The complexity of DISTANCE-SAT *and of several restrictions of it obtained by considering:*

- *a knowledge base Σ for which* SAT *is tractable,*
- *a fixed maximal distance d*

are reported in the following table.

KB	any d	a fixed d
any Σ	NP-complete	NP-complete
Σ Horn	NP-complete	in P
Σ reverse Horn	NP-complete	in P
Σ Krom	NP-complete	in P

Clearly enough, SAT, the satisfiability problem of a CNF formula is a restriction of DISTANCE-SAT (taking $PI = \emptyset$ (or $d = |Var(\Sigma)|$) so as to reduce SAT to DISTANCE-SAT is sufficient to prove the NP-hardness of DISTANCE-SAT). Hence, it is not surprising that DISTANCE-SAT is intractable in the general case, i.e., there is no known polynomial algorithm to solve it (and there can be no such algorithm unless P = NP). Nevertheless, DISTANCE-SAT is not much more difficult than SAT since it belongs to NP. Indeed, verifying that a guessed interpretation disagrees with PI on at most d variables can easily be achieved in polynomial time.

Contrastingly, focusing on the standard fragments of propositional logic where SAT is known as tractable is not sufficient to ensure the polynomiality of DISTANCE-SAT in the general case. Both NP-hardness of the restrictions where Σ is Horn, reverse Horn or Krom are consequences of the NP-hardness of DISTANCE-SAT under the restriction where Σ is a 2-CNF monotone formula, i.e., every literal of Σ has only either positive occurrences or negative occurrences in Σ. The NP-hardness of this last problem is a consequence of the fact that the well-known HITTING SET problem – that is NP-complete (Karp 1972)– can be polynomially many-one reduced to it. Thus, DISTANCE-SAT can be considered at least as difficult as SAT.

As Proposition 1 illustrates it, focusing on tractable KBs is sufficient to obtain tractable restrictions of DISTANCE-SAT as long as d is considered as a fixed constant. Some other restrictions can be considered so as to achieve tractability. Thus, while imposing PI to be a complete interpretation does not lower the complexity of DISTANCE-SAT in the general case (even when Σ is s.t. SAT is tractable), determining whether Σ has a model that disagrees with a complete interpretation I on at most d variables, where d is a constant, is in P. To be more precise, if k is the maximal length of clauses of Σ, there exists a $\mathcal{O}(|\Sigma| * k^d)$ time algorithm that solves this last problem (cf. Section 4).

Interestingly, as a by-product of Proposition 1, some new results about the complexity of the satisfiability issue for some extended propositional languages can be derived. Given a propositional language $PROP_{PS}$, let a *cardinality constraint* be an ordered pair $\langle\{l_1, ..., l_k\}, m\rangle$, where each l_i ($i \in 1 .. k$) is a literal of $PROP_{PS}$ and m is a non-negative integer that is less than or equal to k. Given an interpretation I, the semantics of such a cardinality constraint in I is *true* iff *at least* m literals from $\{l_1, ..., l_k\}$ belong to I (i.e., are interpreted as *true* in I as well). A *cardinality formula* is a (finite) conjunction of cardinality constraints (Benhamou, Saïs, & Siegel 1994) (Van Henteryck & Deville 1991).

Clearly enough, expressing that we are looking for a model of Σ that disagrees with $PI = \{l_1, ..., l_k\}$ on at most d variables amounts to look for a model of the formula obtained by adding to the clauses of Σ the single cardinality constraint $\langle\{l_1, ..., l_k\}, k-d\rangle$. Many more clauses, but a polynomial number of it, are required to reduce DISTANCE-SAT to SAT in the general case [2]. Thus, as a direct consequence of Proposition 1, checking whether a cardinality formula Σ is satisfiable is NP-complete, even when Σ contains only (classical) clauses (i.e., with $m = 1$) that form a Horn CNF formula (or a reverse Horn CNF formula or a Krom one), plus one cardinality constraint with $m \neq 1$.

Two Algorithms for DISTANCE-SAT

In this section, two algorithms for DISTANCE-SAT are presented. These algorithms are based on the standard Davis / Putnam search procedure for SAT (Davis, Logemann, & Loveland 1962). This choice is motivated by the two following facts:

- A naive approach that would consist in enumerating in a successive way the interpretations that do *not* disagree with PI on at most d variables is not computationally feasible in the general case, even for quite small values of n, the number of variables of Σ, and d. For instance, with $n = 100$, $d = 10$ and PI is any complete interpretation, more than 10^{13} interpretations should be considered, which makes such a naive enumerative technique far from being practical.
- The best complete algorithms for SAT that we can find in the literature are based on the Davis / Putnam search procedure, and SAT is a restriction of DISTANCE-SAT. Especially, if $\Sigma \notin$ SAT, then $\forall PI \forall d, \langle\Sigma, PI, d\rangle \notin$ DISTANCE-SAT.

Our first algorithm, $DP_{distance}$, mainly is the standard Davis/Putnam search procedure, equipped with a counter that indicates for every node of the search tree the number of variables on which the partial interpretation associated to that node disagrees with the given configuration. As soon as the value of the counter exceeds d, the algorithm backtracks.

Procedure $DP_{distance}$: $BOOLEAN$
Input : an instance $\langle\Sigma, PI, d\rangle$ of DISTANCE-SAT.
Output : *true* iff $\langle\Sigma, PI, d\rangle \in$ DISTANCE-SAT.
Begin
    unit_propagate(Σ);
    if disagree(PI_C, PI) > d **then return** (*false*);
    if the empty clause is generated **then return** (*false*);
    else if all clauses are satisfied **then return** (*true*)
        else begin
            $x := branching(\Sigma, PI_C)$;
            return ($DP_{distance}(\Sigma \wedge x)$ or
                $DP_{distance}(\Sigma \wedge \neg x)$);
        end;
End

In this algorithm, PI_C is the current partial interpretation, i.e., the one associated to the current node of the search tree. PI_C gathers all the variables that have been fixed from the root of the tree to the current node. *unit_propagate* is a function that performs unit-propagation through Σ. PI_C is updated by *unit_propagate*.

It is well-known that the design of a branching rule is a critical factor in the performance of any Davis/Putnam-like algorithm for SAT. Our *branching* function implements the branching rule given in (Dubois *et al.* 1996), that is one of the best performer for SAT. To be more precise, the weight of a literal l of a formula Ω is given by $w(l) = \sum_{\forall c \in \Omega, l \in c} -ln(1 - 1/(2^{|c|}-1)^2)$ and the score of a variable x by $s(x) = w(x) + w(\neg x) + 1.5min(w(x), w(\neg x))$. A variable maximizing s is elected as the branching variable.

Clearly enough, $DP_{distance}$ is very close to the standard Davis / Putnam procedure. Actually, the unique difference between them is the additional backtrack instruction that is triggered as soon as the current partial interpretation PI_C disagrees with PI on more than d variables. Accordingly, the design of $DP_{distance}$ is mainly guided by the purpose of taking advantage of

[2] This comes from the fact that SAT is NP-complete: every problem in NP can be polynomially reduced to it.

a state-of-the-art algorithm for SAT. The distance constraint is not exploited in an aggressive way, but only in a passive way.

Our second algorithm $DP_{distance+lasso}$ is a variant of $DP_{distance}$ in which the *branching* function that is used does not correspond to a standard branching rule for SAT but has been especially tailored for DISTANCE-SAT. The purpose is to take advantage of both the best branching rules that are available for SAT but also to exploit the distance constraint much more aggressively than in $DP_{distance}$. Unlike the *branching* function, all the variables occurring in Σ (simplified by PI_C) are not considered by the $branching_{lasso}$ function. Only the variables that appear in the set S_{PI_C} of the clauses of Σ simplified by PI_C that are completely falsified by PI, and are of minimal size, are taken into account. Then, the weights of these variables are computed using the same weight function as in *branching*, and a variable with a maximal weight is elected. Remark that the idea of choosing the branching variable among the variables which occur in clauses that are falsified by a reference interpretation appears in SCORE(FD/B), a local search-based complete algorithm for SAT (Chabrier, Juliard, & Chabrier 1995).

Let γ be a clause of S_{PI_C} and x a variable of γ. When x will be assigned the sign it has in γ, γ will become satisfied by the updated partial interpretation PI_C. When x will be given the opposite sign, the resulting simplified clause (i.e., γ in which the literal corresponding to x has been removed) is still completely falsified by PI, and necessarily is of minimal size. This will force the remaining variables of γ to be among the candidate variables for branching at the next choice node. Interestingly, whenever PI is a complete interpretation and the size of the longest clause of Σ is bounded by a constant k, only $\mathcal{O}(k^d)$ choice nodes are to be generated by $DP_{distance+lasso}$, provided that a variable of S_{PI_C} is always elected as the branching variable. We call such a property the *lasso effect*. When the lasso effect works, $DP_{distance+lasso}$ runs in $\mathcal{O}(|\Sigma| * k^d)$ time, i.e., the number of clauses occurring in Σ influences the computational performance of $DP_{distance+lasso}$ by only a *linear* factor. This is far from being expected for $DP_{distance}$. Though the lasso effect is not guaranteed when PI is not a complete interpretation, we will show in the following section that $DP_{distance+lasso}$ nevertheless proves "efficient" in many situations where PI is not complete.

The design of the $branching_{lasso}$ function is done so as to take advantage of both the lasso effect (considering only the variables of S_{PI_C}), and the best branching rules for SAT (the variables are ordered so as to select one that maximizes a standard weight function). In particular, since the lasso technique simply consists in filtering out some candidate variables before applying to them any *branching* function, several *branching* functions for SAT can be considered, giving rise to several $branching_{lasso}$ functions. Let us also note that the distance constraint can be exploited in a more integrated way within the *branching* function, especially for the propagation-based ones, like the one used in SATZ (Li & Anbulagan 1997). Propagating a literal through a CNF formula results in a partial interpretation (encoding the literals that have been fixed) and a corresponding simplified CNF formula. The value of the disagreement between such a partial interpretation and the reference one, and the tightness of the associated simplified formula are two parameters that can be used to evaluate heuristically whether propagating a literal is promising for DISTANCE-SAT.

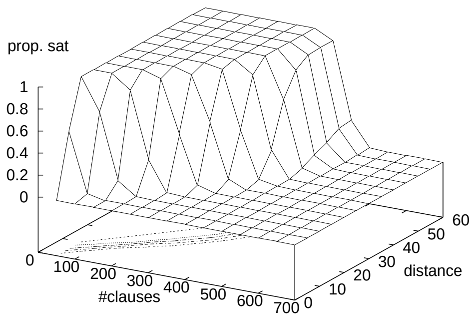

Figure 1: Proportion of satisfiable 100 variables 3-CNF instances of DISTANCE-SAT, as a function of both the number of clauses and the bound d, given a complete reference interpretation.

Finally, it is worth noting that both $DP_{distance}$ and $DP_{distance+lasso}$ can be easily modified to address the function problem associated to DISTANCE-SAT, i.e., to return a model of Σ that disagrees with PI on at most d variables whenever such a model exists. Instead of returning *true* when an implicant of Σ is found, it is sufficient to return any extension of the current partial interpretation PI_C.

Empirical Evaluation

All the results presented hereafter concern random 3-CNF formulas Σ generated under the "fixed-length clauses" model (Chvátal & Szemerédi 1988): literals are drawn under uniform conditions and clauses with redundant variables are rejected. Without loss of generality, the variables of $Dom(PI)$ are the first ones w.r.t. the lexicographic order, and they are assigned to $false$. Every DISTANCE-SAT instance can be turned into an instance for which this assumption is satisfied, through a simple renaming of its literals.

The computational difficulty of a DISTANCE-SAT instance w.r.t. any of our two algorithms is quantified as the size of the corresponding search tree, where both unary and binary nodes are taken into account; in other words, it is evaluated as the number of variable assignments that are required to solve the instance. This difficulty measure does depend neither on the implementation of the algorithms nor on the computer used to perform the experiments.

Figure 1 gives the proportion of satisfiable 100 vari-

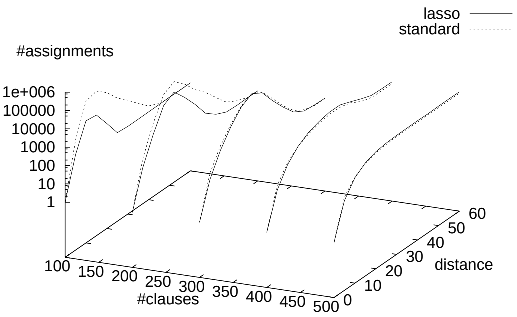

Figure 2: Average number of assignments required by $DP_{distance}$ and $DP_{distance+lasso}$ to solve random 100 variables 3-CNF instances of DISTANCE-SAT with complete reference interpretations.

ables instances, as a function of both the number of clauses of Σ and the bound d, given a complete reference interpretation ($|Dom(PI)| = 100$). A sharp transition appears between the satisfiable and the unsatisfiable regions. When d is large, the transition appears at the well-known satisfiability threshold for SAT, i.e., when number of variables / number of clauses = 4.25 (Cheeseman, Kanefsky, & Taylor 1991) (Crawford & Auton 1996). When d decreases, less clauses are required to produce unsatisfiable instances.

Figure 2 compares the average number of variable assignments required by $DP_{distance}$ and $DP_{distance+lasso}$ to solve random 100 variables instances, for several number of clauses and several d. PI is a fixed complete interpretation ($|Dom(PI)| = 100$). Clearly enough, the lasso branching rule outperforms the standard one for many instances among the most difficult ones (100 and 200 clauses, d between 10 and 30).

This is confirmed by Figure 3, where the ratios between the number of variable assignments required by $DP_{distance}$ to the number of variable assignments required by $DP_{distance+lasso}$ are reported for the same instances as those considered in Figures 1 and 2. Contrastingly, for the largest values of d and the number of clauses, the standard branching rule is slightly better than the lasso one. As additional interesting information, Figures 1 and 2 show that the most difficult instances are near the transition from satisfiable to unsatisfiable for DISTANCE-SAT, and are much more difficult than the corresponding SAT instances (i.e., those obtained by ruling out the distance constraint).

Figure 4 gives the proportion of satisfiable 100 variables instances of DISTANCE-SAT for a fixed $d = 20$, as a function of the number of clauses and $|Dom(PI)|$. Figure 5 compares the average numbers of variable assignments needed by $DP_{distance}$ and $DP_{distance+lasso}$ to solve the instances that have been considered in Figure 4. Figure 6 gives the ratio of the number of variable assignments required by $DP_{distance}$ to the number of variable assignments required by $DP_{distance+lasso}$ for solving the instances considered in Figures 4 and 5.

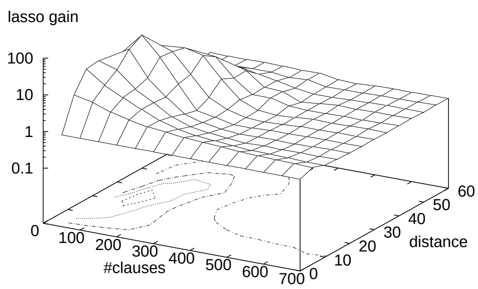

Figure 3: Ratio between the number of assignments required by $DP_{distance}$ to the number of assignments required by $DP_{distance+lasso}$, as a function of both the number of clauses and the bound d. 100 variables 3-CNF formulas, complete reference interpretation.

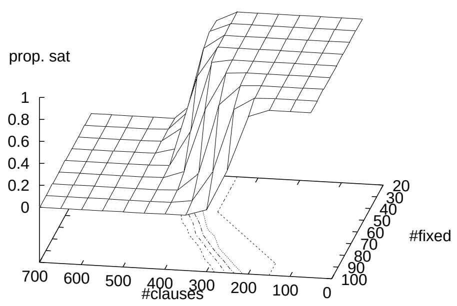

Figure 4: Proportion of satisfiable 100 variables 3-CNF instances of DISTANCE-SAT, as a function of both the number of clauses and the number of fixed variables in the reference interpretation, given $d = 20$.

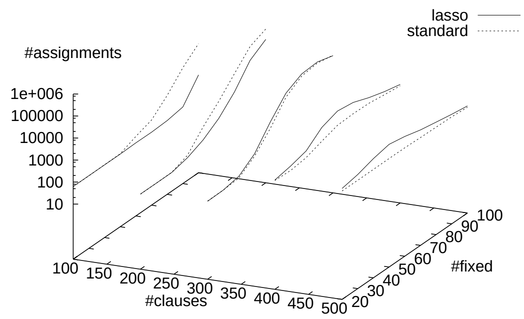

Figure 5: Average number of assignments required by $DP_{distance}$ and $DP_{distance+lasso}$ to solve 100 variables 3-CNF instances with partial reference interpretations, with $d = 20$.

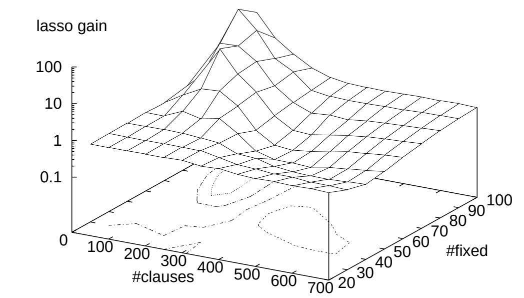

Figure 6: Ratio between the number of assignments required by $DP_{distance}$ to the number of assignments required by $DP_{distance+lasso}$, as a function of both the number of clauses and the number of fixed variables in the reference interpretation. 100 variables 3-CNF formulas, $d = 20$.

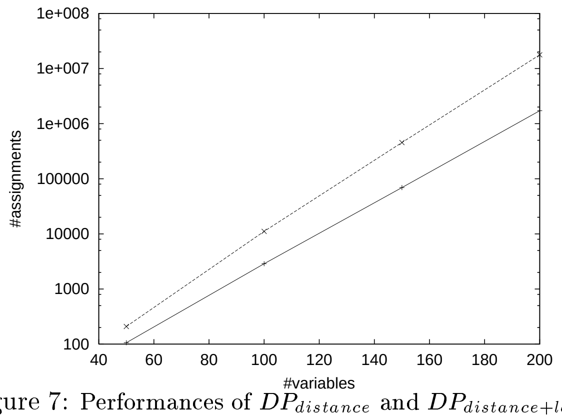

Figure 7: Performances of $DP_{distance}$ and $DP_{distance+lasso}$ in solving 3-CNF instances of DISTANCE-SAT, as a function of the number n of variables. d is fixed to $n/10$ and the number of clauses to $2n$.

At the light of these three figures, it appears that the most difficult instances for DISTANCE-SAT are near its transition from satisfiable to unsatisfiable. The lasso branching rule outperforms the standard one when the number of clauses is small, even when the partial interpretation PI is far from being complete (e.g., $d = 150$ and $|Dom(PI)| = 60$).

Finally, Figure 7 compares the performances of $DP_{distance}$ and $DP_{distance+lasso}$ in solving instances of DISTANCE-SAT, as a function of the number n of variables occurring in them. d is fixed to $n/10$, and the number of clauses to $2n$. These instances of DISTANCE-SAT appear as extremely difficult; in particular, they are much more difficult than their corresponding SAT instances. Clearly enough, the lasso branching rule pushed back the intractability of these instances.

Conclusion

The main contribution of this paper is the identification of the complexity of DISTANCE-SAT and of several restrictions of it, as well as two algorithms for solving it. An empirical evaluation of these two algorithms has been conducted, allowing some conclusions about their respective applicability to be drawn.

Because instances of DISTANCE-SAT can be easily encoded as instances of the satisfiability problem for propositional cardinality formulas, it would be interesting to extend our algorithms so as to make them able to take simultaneously several distance constraints into account. This is an issue for further research.

Acknowledgements

This work has been supported in part by the Ganymède II project of the "Contrat de Plan Etat/Région Nord Pas-de-Calais" and by the IUT of Lens.

References

Benhamou, B.; Saïs, L.; and Siegel, P. 1994. Two proof procedures for a cardinality based langage in propositional calculus. In *Proc. of STACS'94*, 71–84.

Chabrier, J.; Juliard, V.; and Chabrier, J. 1995. SCORE(FD/B) an efficient complete local-based search method for satisiability problems. In *Proc. of the CP'95 Workshop on Solving Really Hard Problems*, 25–30.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. 1991. Where the really hard problems are. In *Proc. of IJCAI'91*, 331–337.

Chvátal, V., and Szemerédi, E. 1988. Many hard examples for resolution. *JACM* 35(4):759–768.

Crawford, J., and Auton, L. 1996. Experimental results on the crossover point in random 3SAT. *Artificial Intelligence* 81:31–57.

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem proving. *CACM* 5:394–397.

Dubois, O.; André, P.; Boufkhad, Y.; and Carlier, J. 1996. *SAT versus UNSAT*. Trick and Johnson. 415–436.

Garey, M., and Johnson, D. 1979. *Computers and intractability: a guide to the theory of NP-completeness*. Freeman.

Ginsberg, M.; Parkes, A.; and Roy, A. 1998. Supermodels and robustness. In *Proc. of AAAI'98*, 334–339.

Karp, R. 1972. *Reducibility among combinatorial problems*. New York: Plenum Press. chapter Complexity of Computer Computations, 85–103.

Li, C., and Anbulagan. 1997. Heuristics based on unit propagation for satisfiability problems. In *Proc. of IJCAI'97*, 366–371.

Reiter, R. 1987. A theory of diagnosis from first principles. *Artificial Intelligence* 32:57–95.

Van Henteryck, P., and Deville, Y. 1991. The cardinality operator: A new logical connective for constraint logic programming. In *Proc. of ICLP'91*, 745–749.