Computing Horn Strong Backdoor Sets Thanks to Local Search

Lionel Paris, Richard Ostrowski, Pierre Siegel
LSIS, Université de Provence,
Marseille, France,
{Lionel.Paris, Richard.Ostrowski, Pierre.Siegel}@cmi.univ-mrs.fr

Lakhdar Saïs
CRIL, Université d'Artois
Lens, France
Lakhdar.Sais@cril.univ-artois.fr

Abstract

In this paper a new approach for computing Strong Backdoor sets of boolean formula in conjunctive normal form (CNF) is proposed. It makes an original use of local search techniques for finding an assignment leading to a largest renamable Horn sub-formula of a given CNF. More precisely, at each step, preference is given to variables such that when assigned to the opposite value lead to the smallest number of remaining non-Horn clauses. Consequently, if no positive or non Horn clauses remain in the formula, our approach answer the satisfiability of the original formula; otherwise, a smallest non-Horn sub-formula is used to extract the set of variables (Strong Backdoor) such that when assigned leads to a tractable sub-formula. Branching on the variables of the Strong Backdoor set leads to significant improvements of Zchaff SAT solver with respect to many real worlds SAT instances.

1. Introduction

Propositional satisfiability (SAT) is the problem of deciding whether a boolean formula in conjunctive normal form (CNF) is satisfiable. SAT is one of the most studied NP-Complete problems because of its theoretical and practical importance. Encouraged by the impressive progress in practical solving of SAT, various applications ranging from formal verification to planning are encoded and solved using SAT. Such improvements in SAT solving are obtained using two complementary approach : stochastic local search and systematic search based techniques. Most of the best complete solvers are based on the backtrack search algorithm called Davis Logemann Loveland (DPLL) procedure [1]. Such basic algorithm is enhanced with many important pruning techniques such as learning, extended use of boolean constraint propagation, preprocessing, symetries, etc. The second important class of satisfiability algorithms concerns the local search based methods. For these incomplete techniques, search space is explored in a non systematic way. An initial complete instantiation of the boolean variables is randomly generated, at each step a new instantiation is obtained by inverting (" flipping ") the value of a chosen variable. Such process is repeated until a model is found or a preset number of steps is reached. Several variants of this basic scheme have been proposed for SAT (*e.g.* [11, 6]). These techniques have shown their impressive performance, particularly on hard and large satisfiable instances (*e.g.* difficult random instances).

Another important factor for the efficiency of SAT solvers concerns the exploitation of the problem structure. Recently, interesting kinds of structure called (Strong) Backdoor were proposed in [14]. Computing Backdoor sets is an active research topic because of its connection to problem hardness. A set of variables forms a Backdoor for a given formula if there exists an assignment to these variables such that the simplified formula can be solved in polynomial time. Such a set of variables is called a Strong Backdoor if any assignment to these variables leads to a tractable sub-formula. Let us remind that computing the smallest Strong Backdoor is a NP-hard problem. In practice, approximating (in polynomial time) a Strong Backdoor set of "reasonable" size is an interesting and important issue. Previous works have addressed this issue [2]. Other approaches have been proposed using different techniques such as adapted systematic search algorithm [13, 4]). Recently, in [5] an enhanced concept of sub-optimal reverse Horn fraction of CNF formula was introduced and an interesting correlation is observed with the satisfiability and the performances of SAT solvers on fixed

density random 3-SAT instances.

The main goal of this paper is to design a polynomial approach that leads to Strong Backdoor sets of reasonable size. To this end, our approach first consider a given CNF formula as a conjunction of two different sub-formulas, where the first one is tractable and the second one is made of the remaining clauses. A Strong Backdoor set can be obtained from the variables appearing in the second sub-formula. However such Backdoor set might be in some cases very large *i.e.* most of the variables of the second formula appear in the tractable part of the formula. To avoid this main drawback, our approach tries to reduce the number of variables in common between these two sub-formulas.

Considering the tractable part as the sub-formula made of Horn clauses, to reduce the size of the Strong Backdoor set, our approach is based on the two following problems :

1. find a "best" renaming of the variables that maximize (resp. minimize) the size of the (resp. non) Horn part of the formula with respect to the number of clauses.

2. compute from the reduced non-Horn sub formula (obtained in 1.) a minimal Horn Strong Backdoor set *i.e.* a minimal set of variables B such that any truth assignment of B leads to a simplified formula which belong to Horn class.

Unfortunately, in the general case the above two problems are NP-hard. Indeed, computing the maximal Horn sub-formulas of a given CNF is equivalent to the MAX2SAT problem [8, 3] *i.e.* given a set of binary clauses, find a model which satisfy the maximum number of clauses. The decision version of the second problem is defined and shown to be NP-complete by Nishimura et al in [7]. First, it belongs to NP, because one can verify in polynomial time that a set B is a Horn Strong Backdoor set. The NP-hardness is shown using a reduction of Vertex-Cover to Horn Strong Backdoor's problem. Consequently, finding a Horn Strong Backdoor of minimal size is also NP-hard.

To circumvent the difficulty of the above two hard problems, our approach makes an original use of local search based techniques to approximate the maximum Horn sub-formula (first problem). Secondly, an efficient heuristic based approach for calculating a Strong Backdoor set (second problem) is proposed. Such heuristic is restricted to the set of variables occurring in the non-Horn part with positive polarity.

The paper is organized as follows. First we introduce some technical backgrounds. Then we present our local search based approach for approximating the maximum Horn sub-formula. Next a heuristic based approach for computing Horn Strong Backdoor set is shown and its exploitation for satisfiability checking is presented. Experiments of our approach on a large class of SAT instances are reported. Finally, the scope of these results is discussed and further promising paths for future research are motivated.

2. Preliminary definitions

Let $\mathcal{B}$ be a Boolean (*i.e.* propositional) language of formulas built in the standard way, using usual connectives ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) and a set of propositional variables. A *CNF formula* Σ is a set (interpreted as a conjunction) of *clauses*, where a clause is a set (interpreted as a disjunction) of *literals*. A literal is a positive or negated propositional variable. Let us recall that any boolean formula can be translated to CNF using linear Tseitin encodings [12]. The size of CNF Σ is defined by $\sum_{c\in\Sigma}|c|$ where $|c|$ is the number of literals in c. A *unit* (resp. *binary*) clause is a clause of size 1 (resp. 2). A *unit literal* is the unique literal of a unit clause. We note $nbVar(\Sigma)$ (resp. $nbCla(\Sigma)$) the number of variables (resp. clauses) of Σ. $\mathcal{V}(\Sigma)$ (resp. $\mathcal{L}(\Sigma)$) is the set of variables (resp. literals) occurring in Σ. The set $\mathcal{L}(\Sigma)$ is the union of positive literals $\mathcal{L}^+(\Sigma)$ and negative literals $\mathcal{L}^-(\Sigma)$. A set of literals $S \subset \mathcal{L}(\Sigma)$ is consistent iff $\forall l \in S, \neg l \notin S$. A literal l is called monotone if $\neg l \notin \mathcal{L}^-(\Sigma)$.

An *truth assignment* of a Boolean formula is an assignment of truth values $\{true, false\}$ to its variables. A variable x is satisfied (resp. falsified) under I if $I[x] = true$ (resp. $I[x] = false$). A *model* of a formula is a truth assignment that satisfies the formula. Accordingly, SAT consists in determining if the formula admits a model.

3. Approximating MRH with local search

Our approach in this paper for approximating the Maximum Horn sub-formula is based on the exploitation of the well known algorithm *WalkSat* [11].

To present our approach, we need to introduce some necessary definitions.

Definition 1 *Let Σ be a CNF formula, $x \in Var(\Sigma)$, I a truth assignment and I' the truth assignment obtained from I by inverting the truth value of x. We define $breakCount(x, I) = |\{c | I[c] = true, I'[c] = false\}|$ and $makeCount(x, I) = |\{c | I[c] = false, I'[c] = true\}|$. We define $score(x, I) = makeCount(x, I) - breakCount(x, I)$ as the score of x under I*

The next variable to flip is chosen in a falsified clause with a maximum score or chosen randomly according to a certain fixed probability. Then the chosen variable is flipped and the counters *makeCount* and *breakCount* are updated.

As a *renaming* of a set of variables V can be defined as an application :

Definition 2 *Let Σ be a CNF formula. We define a renaming R of $Var(\Sigma)$ as the application of V to $\{false, true\}$.*

We can remark that a renaming R can be identified to a classical truth assignment.

Definition 3 *Let Σ be a CNF formula and R a renaming of $Var(\Sigma)$. We define Σ_R as the formula obtained by substituting, for all variables x st. $R[x] = true$, every occurrence of x (resp. $\neg x$) by $\neg x$ (resp. x). x is said to be renamed in Σ. When Σ_R is a Horn formula, R is called a Horn-renaming of Σ.*

To compute the maximum Horn sub-formula, our local search based approach is slightly different. First the *truth assignment* returned by our algorithm represents either a *Horn renaming* of the formula or a best *renaming* with respect to the size of the Horn sub-formula. In the first case, the formula belong to a renamable Horn class and its satisfiability can be checked in linear time. In the second case we obtain an approximation of the maximum Horn sub-formula.

Definition 4 *Let Σ be a CNF formula, I a truth assignment, and c a clause of Σ. We define the number of literals satisfying (resp. falsifying) c under I by $nbLS(c, I)$ (resp. $nbLU(c, I)$).*

Definition 5 *Let Σ be a CNF formula and I a truth assignment. A clause $c \in \Sigma$ is called Horn-satisfied by I (in short $h\_sat(c, I)$) if $nbLU(c, I) \leq 1$ i.e c is satisfied by at most one literal; otherwise it is called Horn-unsatisfied by I (in short $h\_unsat(c, I)$). We define $nbHorn(\Sigma, I)$ as the number of clauses of Σ Horn-satisfied by I.*

Definition 6 *Let Σ be a CNF formula, $x \in Var(\Sigma)$, I a truth assignment and I' the truth assignement obtained from I by inverting the truth value of x. We define $h\_breakCount(x, I) = |\{c | h\_sat(c, I), h\_unsat(c, I')|$ and $h\_makeCount(x, I) = |\{c | h\_unsat(c, I), h\_sat(c, I')\}|$. We define $h\_score(x, I) = h\_makeCount(x, I) - h\_breakCount(x, I)$*

In traditional local search techniques, a clause is considered as satisfied if *at least one of its variables* is true; otherwise it is called unsatisfied. If no falsified clause remains in the formula, these algorithms return a model.

In our Horn local search approach, we only need to maintain, for each clause, the number of literals assigned to true. Also, unsatisfied clauses in traditional local search techniques can be identified to *h_unsat* clauses in our approach. Consequently, we can obtain in a very simple way a Horn local search variant using any local search techniques (*e.g.* tabu, novelty, rnovelty...). The Algorithm 1 describes the Horn Walksat version for computing maximum Horn sub-formula (MRH in short).

Algorithm 1 WalkHorn algorithm

Function WalkHorn
Require: A CNF formula Σ
Ensure: The best *renaming* found for MRH of Σ
1: Initialize R_{max} with all variables of Σ set to true
2: **for** $i = 1$ to *MAX_TRIES* **do**
3: $R =$ a randomly generated *truth assignment*
4: **for** $j = 1$ to *MAX_FLIPS* **do**
5: **if** $nbHorn(\Sigma, R) = nbCla(\Sigma)$ **then**
6: **return** R {Σ *is renamable Horn*}
7: **end if**
8: **if** $(\forall c \in \Sigma, nbLU(c, R) > 0)$ **or** $(\forall c \in \Sigma, nbLS(c, R) > 0)$ **then**
9: **return** R {Σ *is satisfiable*}
10: **end if**
11: {*Horn Random Walk Strategy*}
12: With probability p **do**
13: Select randomly a non Horn clause c and a literal l in c
14: $R = R - \{l\} \cup \{\neg l\}$
15: **done**
16: With probability $1 - p$ **do**
17: Let $l \in R$ st. $\forall l' \in R$ with $l \neq l'$, $h\_score(l, R) > h\_score(l', R)$
18: $R = R - \{l\} \cup \{\neg l\}$
19: **done**
20: **if** $nbHorn(\Sigma, R) > nbHorn(\Sigma, R_{max})$ **then**
21: $R_{max} = R$
22: **end if**
23: **end for**
24: **end for**
25: **return** R_{max}

Besides, the ending conditions of our algorithm are slightly different from classical local search ones. $WalkHorn$ terminates in three different cases :

1. the maximum number of tries (*MAX_TRIES*) is exceeded. The best renaming found is returned (line 25) or,
2. a Horn renaming is found (line 6). The formula Σ is renamable Horn or,
3. a renaming R such that for each clause c in Σ, $nbLU(c, R) > 0$, or such that for each clause c in Σ, $nbLS(c, R) > 0$. Consequently, each clause of Σ_R contains a least one positive literal, or each clause of Σ_R contains a least one negative literal (line 9).

In the third case, the formula is not renamable Horn, but a model exists for Σ. All clauses in Σ_R contain at least one positive literal, then Σ_R is satisfiable. As Σ and Σ_R are equivalent for SAT, then Σ is also satisfiable. In this case, the *Strong Backdoor* of Σ is empty.

4. Computing Strong Backdoor sets

Even though SAT is NP-complete in general, instances encoding real world problems often contain hidden structures that can be used for an efficient SAT solving. In this section, we describe how to compute Strong Backdoor sets using the non-Horn sub-formula obtained by WalkHorn in 3. Backdoor sets of variables were introduced by Williams, Gomes and Selman in [13].

Definition 7 *Let Σ be a CNF. $B \subseteq \mathcal{L}(\Sigma)$ is a Backdoor set if it exists a truth assignment of B such that the simplified formula belongs to a tractable class.*

Definition 8 *Let Σ be a CNF. $B \subseteq \mathcal{L}(\Sigma)$ is a Strong Backdoor set if for all truth assignments of B the simplified formula belongs to a tractable class.*

In the sequel, we consider the non-Horn sub-formula for computing a Strong Backdoor set. The problem of finding the best Strong Backdoor consists in finding a set of positive literals such that when removed from this sub-formula it becomes a Horn formula.

Finding the smallest Horn Strong Backdoor set of a given formula Σ is an NP-hard problem [7]., That is why we propose a greedy method to compute a good subset of variables. The algorithm consists in choosing the variable appearing mostly in the sub-formula and in removing all its positive occurrences until the sub-formula become a Horn formula.

Algorithm 2 shows how to compute a Strong Backdoor set using this greedy method.

Algorithm 2 Computing Strong Backdoor

Function Backdoor
Require: NHF : non-Horn clauses
Ensure: B : Strong Backdoor
1: init $B = \emptyset$
2: **repeat**
3: v=chooseVariable()
4: remove from NHF all positive occurrences of v
5: add v in B
6: **until** NHF become Horn
7: **return** B

5 Experiments

The first experimentations were carried out over many classes of randomly generated 3-SAT instances, containing from 150 to 450 clauses. Each class of instances were generated at the threshold *i.e.* the ratio between the number of clauses and the number of variables was equal to 4,25. For each class, we randomly generated 400 instances. In average over all the classes, the Strong Backdoor set contains 54,64% of all the variables, and the number of non-Horn clauses (after the renaming) represents 30,2% of the total number of clauses. As the size of the Strong Backdoor set (about 50% of the set of original variables) is not significantly reduced, we can not expect significant improvements in exploiting such sets as branching sets of variables for SAT solvers.

Then experimentations were carried out over more than 8000 instances issued from the last sat competitions (industrial, crafted)[9, 10]. After having computed the Horn Strong Backdoor set for each instance, we integrated it in the SAT solver Zchaff as the set of branching variables. This means that the solver is forced to branch on the variables contained in the Strong Backdoor set. We compared the performances of our method with the original Zchaff solver (2003 version) without restricting its branching to the Strong Backdoor set. We give some results of these experiments in table 1. For each instance, Zchaff runs on the original (resp. renamed) instance without (resp. with) restricting its branching to the variables of the Strong Backdoor set. The column *Zchaff* (resp. *Zchaff+Backdoor*) shows their respective behavior. For each instance, the first three columns give the name of the instance, its number of variables (# V) and its number of clauses (# C). The following column indicates if the instance is satisfiable or not ('S' for Satisfiable and 'U' for Unsatisfiable). The following two (*S. Backdoor*) give the number of variables contained in the Strong Backdoor (|B|) and the proportion of non-Horn clauses in the renamed problem (% NH). Then we give the the maximum tree depth reached during the search (MxD.), the number of nodes visited during the search (# Node) and the cpu time in seconds (time) needed to determine the satisfiability of the instances. These preliminary experiments clearly show that exploiting Strong Backdoor sets achieve significant improvements on many real world instances. Besides, we can see that the size of the Horn Strong Backdoor set is linked to the difficulty of the instances, as shown with the vmp* instances.

6. Conclusions and perspectives

In this paper, we have proposed a new approach for approximating Strong Backdoor sets of CNF formula. It originally uses an adaptation of local search techniques for computing a best approximation of the maximum Horn sub-formula of a given CNF. A Strong Backdoor set is then extracted from the remaining set

Instances	# V	# C	S/U	S. Backdoor		Zchaff			Zchaff + Backdoor		
				|B|	% NH	MxD.	# Node	time	MxD.	# Node	time
pret150_60	150	400	U	117	27,5	51	4643957	218,6	59	3354	0,0
pret150_75	150	400	U	117	25,5	51	4639531	219,3	56	2716	0,0
pret150_4	150	400	U	116	27,5	51	4364817	204,3	58	3805	0,1
pret150_2	150	400	U	116	27,5	51	4359886	202,7	56	2815	0,0
pret60_25	60	160	U	46	26,2	21	131394	4,7	25	654	0,0
pret60_75	60	160	U	46	26,2	21	131394	4,6	22	719	0,0
pret60_60	60	160	U	46	26,2	21	131391	4,6	25	729	0,0
dp10s10	8372	23004	S	2635	21,2	51	449994	377,1	48	50664	27,3
vda_gr_rcs_w9	6498	130997	S	4809	< 0,1	875	3316	0.1	571	2997	0,5
urquhart2_25	60	160	U	43	26,2	21	270323	10,6	21	47455	1,7
lisa20_2_a	1201	6563	S	820	43,5	31	307127	184,3	37	91990	37,8
lisa20_0_a	1201	6563	S	824	43,6	32	214046	120,9	42	117961	55
unif-c1075-v250-s640	250	1075	U	139	30	34	533501	338,6	30	396039	164
unif-c1075-v250-s550	250	1075	U	133	30	29	532667	334,5	37	409268	194,4
rand_net40-25-10	2000	5921	U	811	19,5	38	442508	235,2	38	330984	153,5
c499_gr_rcs_w5	2070	22470	S	885	0,01	237	194966	1,6	212	60051	2,5
apex7_gr_rcs_w5	1500	11695	S	740	0,01	426	701	0,0	281	561	0,0
bart11	162	684	S	132	0,1	68	183047	85,4	69	84702	22,5
vmpc_21	441	45339	?	439	<0,01	–	–	–	–	–	–
vmpc_25	625	76755	?	603	<0,01	–	–	–	–	–	–
vmpc_29	841	120147	?	815	<0,01	–	–	–	–	–	–

Table 1. Zchaff on industrials instances.

of clauses. Preliminary results are very encouraging. On some instances, local search found a good Horn renaming formula. Then a good Strong Backdoor can be set using this greedy algorithm. Moreover we believe that the study of Strong Backdoor can be essential in the understanding of the difficulty of hard SAT instances. For example, vmpc* instances, despite their small non Horn sub-formula, have a Strong Backdoor set of the size of the instance and are for all solvers difficult to solve. Finally we show that the integration of our approach in Zchaff SAT solver using the computed Strong Backdoor set as the branching set of variables achieve significant improvements wrt. some real world instances. The work presented in this paper open interesting perspectives. We think that, non-Horn clauses can be used to define a new objective function that can be grafted in local search techniques as a new strategy to escape from local minima. Finally, considering other tractable formulas is another path for future research.

References

[1] M. Davis, G. Logemann, and D. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[2] E. Grégoire, B. Mazure, R. Ostrowski, and L. Saïs. Automatic extraction of functional dependencies. In *proc. of SAT*, volume 3542 of *LNCS*, pages 122–132, 2005.

[3] E. A. Hirsch. A new algorithm for MAX-2-SAT. *Lecture Notes in Computer Science*, 1770:65–73, 2000.

[4] P. Kilby, J. Slaney, S. Thiebaux, and T. Walsh. Backbones and backdoors in satisfiability. In *AAAI'2005*, 2005.

[5] H. V. Maaren and L. V. Norden. Correlations between horn fractions, satisfiability and solver performance for fixed density random 3-cnf instances. *Annals of Mathematics and Artificial Intelligence*, 44:157– 177, 2005.

[6] B. Mazure, L. Saïs, and É. Grégoire. Tabu search for SAT. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 281–285. AAAI Press, July 27–31 1997.

[7] N. Nishimura, P. Ragde, and S. Szieder. Detecting backdoor sets with respect to horn and binary clauses. In *7th International Conference on theory and Application of Satisfiability Testing*, 2004.

[8] C. H. Papadimitriou. *Computational complexity*. Addison–Wesley, 1994.

[9] Sat 2002 : Fifth international symposium on theory and applications of satisfiability testing, May 2002. http://gauss.ececs.uc.edu/Conferences/SAT2002/.

[10] Sat 2003 : Sixth international symposium on theory and applications of satisfiability testing, May 2003. http://www.mrg.dist.unige.it/events/sat03/.

[11] B. Selman and H. A. Kautz. An empirical study of greedy local search for satisfiability testing. 1993.

[12] G. Tseitin. On the complexity of derivations in the propositional calculus. In H. Slesenko, editor, *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pages 115–125, 1968.

[13] R. Williams, C. Gomes, and bart Selman. Backdoors to typical case complexity. In *Procceding of Intenrational Joint Conference on Artificial Intelligence (IJCAI'2003)*, 2003.

[14] R. Williams, C. Gomes, and B. Selman. On the connections between heavy-tails, backdoors, and restarts in combinatorial search. In *International Conference on Theory and Applications of Satisfiability Testing (SAT'2003)*, 2003.