Improving Satisfiability Algorithms with Dominance and Partitioning

João P. Marques Silva and Arlindo L. Oliveira
Cadence European Laboratories / IST / INESC
Lisbon, Portugal
{jpms,aml}@inesc.pt

Abstract

In this paper we describe how several search pruning concepts, commonly used in algorithms for solving covering problems, can be incorporated in algorithms for propositional satisfiability (SAT). In particular, we show that the concepts of row dominance and matrix partitioning, commonly used for solving unate and binate covering problems, can be naturally applied to SAT algorithms. Experimental results, conducted on a large number of benchmarks, indicate that these techniques lead to significant performance improvements for an existing SAT algorithm.

1. Introduction

Propositional satisfiability (SAT) finds application in different Electronic Design Automation (EDA) tasks, for example combinational equivalence checking, automatic test pattern generation, delay-fault testing and timing analysis. In addition, SAT algorithms can be used as the back-end search engines of algorithms for solving other EDA problems.

In recent years there has been extensive research work on algorithms for solving covering problems with very promising experimental results [1, 2, 5, 6]. Similarly, new algorithms for SAT have been proposed [7], also with promising experimental results. The purpose of this paper is twofold. First, we note that some of the pruning techniques used for solving covering problems and for solving satisfiability are equivalent. Next, we show how some of the techniques used for solving covering problems can be naturally applied to SAT. In particular, this is the case with row dominance and matrix partitioning. The experiments performed show that the integration of these techniques in a SAT solver yields significant performance improvements on a large number of benchmark examples [4].

2. The GRASP SAT Algorithm

The GRASP [7] SAT algorithm is a search-based SAT algorithm for Conjunctive Normal Form (CNF) formulas, which incorporates several powerful search pruning techniques. These techniques are based on the analysis of conflicts, which denote situations when one or more clauses have all literals assigned value 0, and include:

- *Non-chronological backtracking*, that denotes the ability for jumping over portions of the decision tree which cannot contain a solution to the given instance of SAT.
- *Conflict-based equivalence*, that is associated with creating clauses representing the causes of conflicts, i.e. *conflict-induced clauses*. These clauses are later used for avoiding entering portions of the search space where a solution cannot be found.
- *Failure-driven assertions*, denoting the ability for identifying necessary assignments, which introduce additional constraints to the search.

GRASP is characterized by three main engines, a *decision engine* for electing decision assignments, a *deduction engine* for identifying necessary assignments, and a *diagnosis engine* for analyzing the causes of conflicts. Different realizations of each of these engines allow for different degrees of pruning ability for the SAT algorithm.

3. Pruning Techniques from Set Covering

Some pruning techniques used in algorithms for solving set covering problems already find equivalent in SAT algorithms. This is the case, as observed in [5], with derivation of necessary assignments using the unit-clause rule [7], which is equivalent to the identification of essential columns. As we show next, other pruning techniques can also be easily adapted to the SAT framework.

3.1 Row and Column Dominance

For instances of SAT row dominance does not seem to be particularly useful, since the original CNF formulation in most cases does not contain clauses that dominate other clauses. Nevertheless, dominance relations can be relevant for the GRASP SAT algorithm, since during the search new conflict-induced clauses are created each time a conflict is identified and analyzed. Some of the larger clauses may eventually become irrelevant and dominated by other clauses that may be subsequently identified. Thus, the identification of dominance relations, specifically with respect to these conflict-induced clauses, will reduce the computational overhead and the memory requirements of GRASP.

We should also note that dominance relations can be used to yield additional partitions of the covering matrix and that this technique can also be naturally incorporated in GRASP. Consider for example the following two clauses $\omega_1 = \left(x_1 + \overline{x_2} + \overline{x_3} + \overline{x_4}\right)$ and $\omega_2 = \left(x_1 + \overline{x_4}\right)$. In terms of a binate covering matrix, the row associated with ω_2 dominates the row associated with ω_1. Hence, ω_1 can be removed from the CNF formula. Moreover, both clauses could possibly have been created as the result of conflict analysis.

With respect to column dominance, this technique is commonly used for solving unate covering problems. However, for binate covering problems (or equivalently, for CNF formulas) it is not applicable in general. One exception is when a variable is unate, since all clauses containing that variable can be removed. Notice, however, that this is equivalent to the pure-literal rule of the Davis-Putnam SAT algorithm [3].

3.2 Dynamic Partitioning

The identification of partitions in covering matrices can be particularly helpful for solving set covering problems [1, 6]. Similarly, the identification of disjoint sub-formulas in a CNF formula can be used for simplifying the computation of the solution of a given instance of SAT, since the solution of each sub-formula can then be computed separately. It is straightforward to devise a linear-time procedure for identifying disjoint sub-formulas in a CNF formula, or partitions in a covering matrix. In GRASP we developed instead a new decision making procedure which finds disjoint sub-formulas *implicitly*, by focusing in subsets of variables that can be used for decision making purposes at each step in the search process. Hence, dynamic partitions are automatically maintained by the new decision making engine of GRASP. Basically, the new decision making procedure is restricted to making the next decision assignment with respect to variables which share clauses with variables assigned at the previous level of the search. Consequently, partitions will provably, but implicitly, be solved separately. Notice that the proposed decision making procedure can incorporate the same heuristic knowledge of other decision making procedures.

Dynamic partitions offer one key significant advantage in GRASP. By focusing the sets of variables in which decisions can be made, non-chronological behavior is enhanced and expected to be observed more often. The experimental results given in Section 4 clearly support this intuition.

In order to illustrate dynamic partitioning and dominance relations, we consider the following covering matrix,

$$\begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & -1 \\ 0 & 0 & 0 & -1 & 0 & 1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix} \tag{1}$$

which can be viewed as representing the following CNF formula:

$$\begin{aligned} &(x_1' + x_3) \cdot (x_1' + x_2) \cdot (x_1 + x_2' + x_3') \cdot \\ &(x_4 + x_5 + x_6') \cdot (x_4' + x_6) \cdot (x_5' + x_6) \end{aligned} \tag{2}$$

The set of clauses involving variables in the set $\{x_1, x_2, x_3\}$ is disjoint from the set of clauses involving variables in the set $\{x_4, x_5, x_6\}$. Hence, we can consider two sub-formulas and solve them independently. It is interesting to note that dynamic partitions can also result from assigning values to variables. Moreover, clause dominance can trigger the identification of additional partitions. For example, for the following CNF formula:

$$\begin{aligned} &(x_1' + x_3) \cdot (x_1' + x_2) \cdot (x_1 + x_2' + x_3') \cdot \\ &(x_4 + x_5 + x_6') \cdot (x_4' + x_6) \cdot (x_5' + x_6) \cdot \\ &(x_2 + x_3 + x_6 + x_7) \cdot (x_6 + x_7) \cdot \end{aligned} \tag{3}$$

no partitions can be identified. However, observe that clause $(x_2 + x_3 + x_6 + x_7)$ is dominated by clause $(x_6 + x_7)$. As a result, we can ignore $(x_2 + x_3 + x_6 + x_7)$ and split the satisfiability problem into two new disjoint satisfiability problems.

4. Experimental Results

In this section we include preliminary experimental results of a new version of GRASP. For this purpose we use some of the DIMACS benchmarks [4]. The experimental results, obtained on a SUN 5/85 machine with 64 MByte of physical memory are shown in Table 1. The total CPU times for different classes of benchmarks (each with a different number of instances #M) are shown for each of the four possible configurations of GRASP, with

Benchmark Class	#M	GRASP w/o partitioning w/o dominance	GRASP w/o partitioning w/ dominance	GRASP w/ partitioning w/o dominance	GRASP w/ partitioning w/ dominance
AIM-100	24	1.9	2.1	1.1	1.3
AIM-200	24	26.9	23.6	4.5	7.0
BF	4	10.6	12.6	11.3	8.1
DUBOIS	13	48.2	28.4	3.0	3.0
II-32	17	11.3	11.3	185.7	178.1
PRET	8	20.0	18.6	9.8	8.5
SSA	8	17.8	19.5	5.2	5.1
AIM-50	24	0.5	0.5	0.4	0.4
II-8	14	24.5	24.6	6.0	6.2
JNH	50	35.1	37.7	41.4	42.4
PAR-8	10	0.5	0.4	0.6	0.6
II-16	10	10,508	10,571	127.8	124.5
BF-0432	21	74.7	76.9	53.1	52.5
BF-1355	149	205.1	203.0	76.5	76.6
BF-2670	53	152.9	160.6	92.1	72.4
SSA-0432	7	1.1	1.2	0.9	0.8
SSA-2670	12	113.3	129.6	109.7	82.4
SSA-6288	3	0.0	0.0	0.0	0.0
SSA-7552	80	34.2	33.9	21.1	21.2

Table 1: Experimental results on DIMACS benchmarks

and without partitioning and with and without dominance. As can be concluded, and with a few exceptions, significant improvements are achieved when row dominance and dynamic partitioning are incorporated into the SAT algorithm. It is important to note that dynamic partitioning plays a key role in reducing the amount of search. In general, the usefulness of row dominance depends on the benchmark class considered.

5. Conclusions and Ongoing Work

This paper illustrates how the decision making engine of a search-based SAT algorithm can be used for effectively pruning the search. This is accomplished by using techniques from set covering problems. The experimental results of Section 4 indicate that these pruning techniques can be particularly useful in solving different classes of instances of SAT. Future research work involves pursuing these topics further. For example, we are currently studying how the identification of more clause dominance relations can yield additional partitions of the CNF formula. In addition, we intend to study how the inclusion of large conflict-induced clauses [7] can negatively affect the ability for identifying dynamic partitions of the CNF formula.

References

[1] O. Coudert, "On Solving Covering Problems," in *Proceedings of the Design Automation Conference*, June 1996.

[2] O. Coudert and J.C. Madre, "New Ideas for Solving Covering Problems," in *Proceedings of the Design Automation Conference*, June 1995.

[3] M. Davis and H. Putnam, "A Computing Procedure for Quantification Theory," *Journal of the Association for Computing Machinery*, vol. 7, pp. 201-215, 1960.

[4] D. S. Johnson and M. A. Trick (eds.), *Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, 1993. DIMACS benchmarks available in ftp://Dimacs.Rutgers.EDU/pub/challenge/sat/benchmarks/cnf. UCSC benchmarks available in ftp://Dimacs.Rutgers.EDU/pub/challenge/sat/benchmarks/contributed/UCSC.

[5] S. Liao and S. Devadas, "Solving Covering Problems Using LPR-Based Lower Bounds," MIT Technical Report, December 1996.

[6] R. L. Rudell, Logic Synthesis in VLSI Design, Ph.D. Dissertation, UCB/ERL M89/49. University of California, Berkeley, 1989.

[7] J. P. M. Silva and K. A. Sakallah, "GRASP—A New Search Algorithm for Satisfiability," in *Proceedings of the International Conference on Computer-Aided Design*, November 1996.