

System Description: CRIL Platform for SAT ^{*}

Bertrand MAZURE, Lakhdar SAÏS, and Éric GRÉGOIRE

CRIL – Université d'Artois,
rue de l'Université SP 16
F-62307 Lens Cedex,
FRANCE.

{mazure,sais,gregoire}@cril.univ-artois.fr

Abstract. The CRIL multi-strategy platform for SAT includes a whole family of local search techniques and some of the best Davis and Putnam strategies for checking propositional satisfiability. Most notably, it features an optimized tabu-based local search method and includes a powerful logically complete approach that combines the respective strengths of local and systematic search techniques. This platform is a comprehensive toolkit provided with most current SAT instances generators and with various user-friendly tools.

1 Introduction

CRIL multi-strategy platform for SAT provides the user with a selection of both local and systematic search techniques. More precisely, it includes many variants of Selman et al. local search [6, 7] techniques that prove efficient in showing the consistency of SAT instances. In particular, it features an optimized tabu-based local search method called TSAT [4]. It also includes a powerful logically complete approach that combines the power of local search methods with the completeness of Davis and Putnam procedure (DP), allowing both large satisfiable and unsatisfiable instances to be proved [5]. It thus shows that local search methods can play a key role with respect to proving unsatisfiability.

In this system description paper, the focus is laid on the main two salient features of the CRIL platform: an optimized tabu-based local search method and its use as an heuristic to guide the branching strategy of DP.

The platform is available by anonymous ftp at `ftp.lifl.fr` in directory: `/pub/projects/SAT`. It has been developed in standard ANSI C and should run on most UNIX systems. Our implementation and experimentations have been conducted on 133, 166 and 200 Pentium under Linux 2.0.30. All the available search techniques have been implemented by us (when these techniques have been previously proposed by other authors, our implementation is as efficient as the original code).

^{*} The development of the CRIL platform for SAT was supported in part by the Ganymède II project of the *Contrat de plan État/Région Nord-Pas-de-Calais*.

The other various software tools are described in an interactive help module. Thanks to its open-ended architecture, the platform can easily be augmented with other techniques, heuristics, options and generators.

2 TSAT: a brief description

TSAT departs from basic GSAT by making a systematic use of a tabu list of variables in order to avoid recurrent flips and thus escape from local minima. This allows a better and more uniform coverage of the search space. More precisely, TSAT keeps a fixed length - chronologically-ordered FIFO - list of flipped variables and prevents any of the variables in the list from being flipped again during a given amount of time. The length of the tabu list has been experimentally optimized (for K-SAT instances) with respect to the size of the problems [4]. TSAT proves very competitive w.r.t. most families of SAT instances.

3 DP + local search: a brief description

Let us now describe a basic algorithm using Selman et al. local search GSAT-like procedures to guide the branching strategy of logically-complete algorithms based on DP [1].

```

Procedure DP+TSAT ;
Input : a set of clauses  $S$ ;
Output : a satisfying truth assignment of  $S$  if found,
          or a definitive statement that  $S$  is inconsistent;
Begin
  Unit_propagate( $S$ );
  if the empty clause is generated
  then return (false);
  else
    if all variables are assigned
    then return (true);
    else
      begin
        if TSAT( $S$ ) succeeds
        then return (true);
        else
          begin
             $p$  := the most often falsified literal during TSAT search;
            return (DP+TSAT( $S \wedge p$ )  $\vee$  DP+TSAT( $S \wedge \neg p$ ));
          end;
        end;
      end;
  End.

```

Fig. 1. DP + TSAT: basic version

TSAT (or another local search procedure) is run to deliver the next literal to be assigned by DP¹. This literal is selected as the one with the highest score according to the following counting. A trace of TSAT is recorded: for each literal, taking each flip as a step of time, we count the number of times the literal appears in the falsified clauses. (Intuitively, it seemed to us that the most often falsified clauses should normally belong to an inconsistent kernel of the SAT instance if this instance is actually inconsistent. This hypothesis proves most often correct).

Such an approach can be seen as using the trace of TSAT as an heuristic for selecting the next literal to be assigned by DP, and a way to extend the partial assignment made by DP towards a model of the SAT instance when this instance is satisfiable. Each time DP needs to select the next variable to be considered, such a call to TSAT can be performed w.r.t. the remaining part of the SAT instance. This algorithm is given in Figure 1. Let us stress that this combination schema was simply designed to show its feasibility: in this respect, it is a very primitive one that can be optimized in several ways [5].

4 Experimental results

In Fig. 2, TSAT [4], GSAT with Random Walk Strategy [6, 7] and WSAT [3] are compared on hard random 3-SAT instances using 500 instances for each problem size at the threshold. ($\#clauses/\#variables = 4.25$).

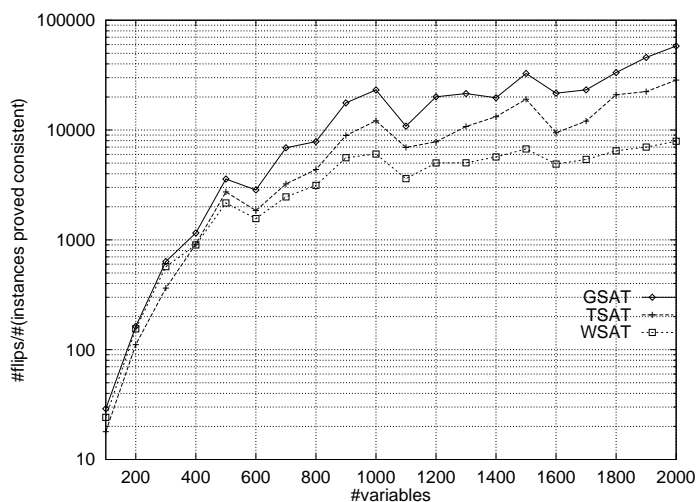


Fig. 2. Results for hard random 3-SAT instances

In Table 1, a significant sample of our extensive experimentations is given, showing the obtained dramatical performance improvement, in particular for classes of inconsistent SAT instances. Also, very good results are obtained for consistent instances; indeed, DP+TSAT is as efficient as local search techniques since DP+TSAT begins with a call to them.

¹ Independly, a close approach has been proposed in [2].

Table 1. DIMACS problems²

Instances	Sat	Size		Inc. Ker. ³		DP+FFIS			DP+TSAT		
		Var.	Cla.	Var.	Cla.	assign.	choices	time	assign.	choices	time
<u>AIM series:</u>											
1.6-no-3	No	100	160	51	57	3E+07	2E+06	214 s71	178	16	0s26
1.6-yes1-2	Yes	100	160	***	***	495858	30052	4s3 9	77	6	0s08
2.0-no-1	No	100	200	18	19	4E+07	2E+06	349 s52	46	5	0s10
2.0-yes1-1	Yes	100	200	***	***	706388	31274	7s4 1	81	8	0s15
1.6-no-1	No	200	320	52	55	***	***	> 8h	240	16	0s58
1.6-yes1-3	Yes	200	320	***	***	***	***	> 9h	232	11	0s32
2.0-no-3	No	200	400	35	37	***	***	> 15h	120	10	0s42
2.0-yes1-1	Yes	200	400	***	***	2E+09	7E+07	218 59s45	291	27	1s21
1.6-no-1	No	50	80	20	22	12072	895	0s 09	72	8	0s06
1.6-yes1-1	Yes	50	80	***	***	1540	84	0s 01	37	6	0s05
2.0-no-1	No	50	100	21	22	54014	2759	0s 43	52	5	0s05
2.0-yes1-1	Yes	50	100	***	***	2878	176	0s 03	11	3	0s03
<u>BF series:</u>											
0432-007	No	1040	3668	674	1252	9E+08	6E+06	19553s44	115766	870	85s25
1355-075	No	2180	6778	82	185	317628	2047	18s88	4602	28	26s23
1355-638	No	2177	4768	83	154	***	***	>17h	6192	32	32s57
2670-001	No	1393	3434	79	139	***	***	>25h	490692	4822	519s40
<u>SSA series:</u>											
0432-003	No	435	1027	306	320	133794	1570	1s79	1338	16	0s80
2670-130	No	1359	3321	552	669	***	***	>33h	2E+07	79426	8040s64
2670-141	No	986	2315	579	1247	3E+08	2E+06	6350s77	1E+07	92421	6639s44
7552-038	Yes	1501	3575	***	***	***	***	>13h	29	1	0s34
7552-158	Yes	1363	3034	***	***	1639	78	0s19	12	1	0s29
7552-159	Yes	1363	3032	***	***	1557	84	0s21	12	1	0s25
7552-160	Yes	1391	3126	***	***	1457	76	0s18	1	1	0s30

References

1. Davis, M., Putnam, H.: A Computing Procedure for Quantification Theory. Journal of the Association for Computing Machinery, **7**, pp. 201–215.
2. Crawford, J.: Solving Satisfiability Problems Using a Combination of Systematic and Local Search. Working notes of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability (1993).
3. McAllester, D., Selman, B., Kautz, H.A.: Evidence for Invariants in Local Search. Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97) (1997) pp. 321–326.
4. Mazure, B., Saïs, L., Grégoire, É.: Tabu Search for SAT, Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97), (1997) pp. 281–285.
5. Mazure, B., Saïs, L., Grégoire, É.: Detecting Logical Inconsistencies. Proceedings of Mathematics and Artificial Intelligence Symposium (1996) pp. 116-121, extended version in Annals of Mathematics and Artificial Intelligence (1998).
6. Selman, B., Levesque, H., Mitchell, D.: A New Method for Solving Hard Satisfiability Problems. Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI'92) (1992) pp. 440–446.
7. Selman, B., Kautz, H.A., Cohen, B.: Local Search Strategies for Satisfiability Testing. Proceedings of the DIMACS Workshop on Maximum Clique, Graph Coloring, and Satisfiability (1993).

² In the table, “> n H” means that we gave up after the problem had not been solved within n hours of CPU time.

³ “Inc. Ker.” in the table means “Inconsistent Kernel”.