

# A Resolution Framework to Combine Exact and Approximate Methods for SAT and MAX-SAT Problems

Frédéric Lardeux, Frédéric Saubion and Jin-Kao Hao

LERIA, University of Angers,

2 Bd Lavoisier, F-49045 Angers Cedex 01

email: {lardeux,saubion,hao}@info.univ-angers.fr

## 1 Introduction

The satisfiability problem (SAT) [4] consists in finding a truth assignment that satisfies a well-formed boolean expression. An instance of the SAT problem is then defined by a set of boolean variables (also called atoms)  $\mathcal{X} = \{x_1, \dots, x_n\}$  and a boolean formula  $\phi: \mathcal{B}^n \rightarrow \mathcal{B}$ . A literal is a variable or its negation; a clause is a disjunction of literals. A truth assignment is a function  $v: \mathcal{X} \rightarrow \mathcal{B}$ . The formula is said to be satisfiable if there exists an assignment satisfying  $\phi$  and unsatisfiable otherwise. The maximum satisfiability problem (MAX-SAT) corresponds to the minimization of the number of false clauses. The formula  $\phi$  is in conjunctive normal form (CNF) if it is a conjunction of clauses where a clause is a disjunction of literals. In this paper,  $\phi$  is supposed to be in CNF.

Two classes of methods can be used to solve SAT and MAX-SAT problems: exact and approximate methods.

- Exact methods are able to find all the solutions of an instance or, if there is no solution, to prove its unsatisfiability. These methods are generally based on the Davis-Putnam-Loveland procedure (DPL) [3] which explores a binary search tree corresponding to the different possible assignments. They provide very good results but are not suitable for the MAX-SAT problem. Other exact methods [1, 14] based on a Branch and Bound (B&B) algorithm have been designed to handle the MAX-SAT problem but their performances are very limited for large instances. In these algorithms, B&B is roughly an extension of the DPL procedure.
- Approximate methods are mainly based on local search (LS) [10, 13, 8] and evolutionary algorithms [5, 7]. They expect to minimize, on the assignments space, a function corresponding to the number of false clauses. Therefore, they are naturally designed for the MAX-SAT problem and allows one to handle large instances.

Hybridizations of these two approaches have been proposed [15, 9, 2, 6] and are mainly based on a *master-slave* combination of a local search process and an exhaustive search. Indeed, in these algorithms, LS is used as a sub-routine in a DPL-like algorithm except for [15] which uses a DPL-like algorithm after a hill-climbing process. In any case, there is no interaction between both processes.

Concerning such hybridizations, we have to remark that, while exact methods work on partial assignments incrementally built by the resolution process, approximate algorithms explore the set of all possible complete assignments. Moreover, approximate methods are originally designed for MAX-SAT while exact methods are more often used for SAT. We obviously consider here SAT as a subcase of MAX-SAT where the number of false clauses is 0. Therefore, our purpose is to provide a uniform framework in order to design hybrid solvers which allows the methods to share a common search space and then to interact.

This is achieved by introducing a tri-valued logical framework. We have proposed an implementation of it within a classical hybridization for MAX-SAT problem between a standard Tabu Search (TS) and a classical Branch and Bound (B&B) process. This new framework allows us to integrate more homogeneously diversification and intensification processes in the TS and in particular an intensification strategy by means of the B&B algorithm.

## 2 TTS: a Tri-valued Tabu Search

As mentioned before, exact methods construct their solutions along the search process and therefore generate successive partial assignments to reach a solution. To unify exact and approximate methods, a possible approach consists in extending approximate methods in order to use partial assignments. To introduce such partial assignments in a local search process, we add a truth value *undefined* denoted by  $U$  to the classic *true* and *false* values.

This third truth value will allow TS to diversify its search without the addition of an external heuristic: whereas a classic TS with two truth values explores the search space  $\mathcal{S}$  moving from an assignment to another, TS with three truth values (TTS) introduces moves across areas of the search space  $\mathcal{S}$  thanks to partial assignments, since a partial assignment can be actually considered as a set of complete assignments. For instance  $T T F U T U$  represents  $\{ T T F F T F, T T F F T T, T T F T T F, T T F T T T \}$ .

The general TS and TTS processes share common principles. The main difference is that the valuation of a variable to *undefined* induces a diversification step whereas the valuation to *true* or to *false* of an undefined variable induces an intensification step (Fig.1).

The introduction of the third truth value requires to redefine the evaluation function and the choose function <sup>1</sup> of TS since the number of undefined clauses must be taken into account. The new evaluation function *eval* returns the number of true clauses as well as the number of undefined clauses. In this context, an assignment is better than another if it satisfies more clauses. If

---

<sup>1</sup>This function selects the variable which value will be changed.

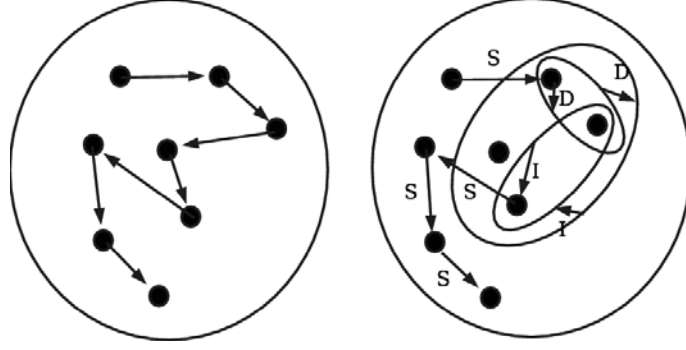


Figure 1: TS (left) and TTS (right). S represents a standard move with two truth values. A diversification (D) step occurs when a variable is valued to *undefined* whereas an intensification step (I) occurs when a variable undefined is valued to *true* or *false*.

the number of true clauses is equal for two assignments, the one generating the greatest number of undefined clauses will be considered as the best assignment.

$$\begin{aligned}
 eval: \quad \mathcal{S} &\rightarrow (IN, IN) \\
 X &\mapsto (|\{c | sat(X, c) \wedge c \in \mathcal{F}\}|, \\
 &\quad |\{c | c \in \mathcal{F}\}| - |\{c | sat(X, c) \wedge c \in \mathcal{F}\}| - |\{c | \neg sat(X, c) \wedge c \in \mathcal{F}\}|)
 \end{aligned}$$

where  $\mathcal{F}$  represents the set of the formula's clauses and  $sat(X, c)$  means that the clause  $c$  is satisfied by the assignment  $X$ . We define the order  $>_{eval}$  as the lexicographic extension  $(>, >)$  of the order  $>$  on the pair constructed by the  $eval$  function.

The purpose of the search process is to increase the number of true clauses. When this is not possible, it is then preferable to increase the number of undefined clauses and thus to reduce the number of false clauses. We propose a choose function which maximizes the number of true clauses and minimizes the number of false clauses (increasing the number of undefined clauses if necessary).

### 3 TTS-BB: a Hybrid Algorithm

The algorithms [1, 2] based on a B&B method obtain good results on instances with few variables but, as the number of variables grows up, the performance of these algorithms decrease dramatically. The interest of the hybridization between TTS and B&B relies on the fact that TTS selects subsets of the search space and B&B explores them exhaustively. Then, TTS takes as input the best assignment found during the B&B phase and a tabu list which contains the variables assigned in the B&B step. It avoids to get stuck in an already studied space. In our algorithm, this interaction continues until a solution is found (0

false clause for SAT instances) or a maximum number of visited assignments is reached. To control the allowed search power of each method, it is necessary to impose that at least a certain number  $s$  of steps (empirically fixed to 5000) are performed by TTS process between two calls to B&B. On the other hand, B&B is executed only if the number of unassigned variables is not greater than a bound  $b$  (empirically fixed to 100). The general process is sketched in algorithm 1.

- 1) TTS executes at least  $s$  steps (except if a solution is found) until a partial assignment with at maximum  $b$  undefined variables is found. (Intensification/Diversification)
- 2) B&B returns the best complete assignment contained in the partial assignment obtained in step 1. (Intensification)
- 3) The undefined variables fixed during the B&B process are added to the tabu list.
- 4) Return to 1.

**Algorithm 1:** General TTS-BB process

## 4 Experimental Results

In this section, we compare the performances of TTS-BB with a classic TS algorithm [8]. No comparison is done with B&B alone since the studied instances have too many variables. The benchmarks used are well-known instances presented during the SAT competitions [11, 12]. Experimentations are carried out for the proof-of-concept purpose only. Comparisons with the state of the art solvers are the subject of an ongoing work.

Due to the approximate and non-deterministic nature of TS and TTS-BB, each algorithm runs 20 times on each benchmark. The maximum number of allowed local search steps is  $10^6$  and the value  $s$  is set to  $5 \times 10^3$ . For our hybridization, we consider that a backtrack has the same cost as a local search step. The length of the tabu list is fixed to 10% of the number of variables of the studied problem. To compare the two algorithms, four criterions are used: the average number of false clauses (avg.) and its standard deviation (s.d.), the average number of steps (including LS steps and B&B backtracks) to obtain the best result (steps) and the percentage of improvement (Imp.) of TTS-BB with respect to the classic TS.

Table 1 shows that the TTS-BB algorithm improves the number of false clauses provided by the classic TS algorithm. This improvement can be weak for some instances (+1.9% for `color-15-4`) but really more significant for other instances (+73.3% for `Mat25.shuffled`). All the TS results are improved by TTS-BB. Moreover, the small standard deviations of the results show that the hybridization provides reliability to the algorithm. The greater number of steps

Benchmarks				TS			TTS-BB			Imp.
instances	var	cls	SAT	false clauses		steps	false clauses		steps	%
				avg.	s.d.		avg.	s.d.		
color-10-3	300	6475	Y	2.60	0.50	219359	2.45	0.76	256962	+5.8
color-15-4	900	45675	Y	5.30	0.52	263991	5.20	0.57	341175	+1.9
f1000	1000	4250	Y	5.58	2.35	365002	3.58	0.79	497987	+35.8
f2000	2000	8500	Y	14.58	4.17	684653	8.67	2.01	543274	+40.5
Mat25.shuffled	588	1968	N	21.83	14.80	43198	5.83	0.57	371227	+73.3
Mat26.shuffled	744	2464	N	8.00	0.00	170516	6.83	0.39	513994	+14.6
par32-5-c	1339	5350	Y	14.70	11.38	303836	11.70	6.66	219720	+20.4
par32-5	3176	10325	Y	23.70	30.04	361385	10.20	1.32	473116	+57.0
ssa7552-038	1501	3575	Y	10.00	13.06	407185	8.55	1.88	553387	+14.5
term1mul.miter	3504	22229	N	154.92	10.47	190158	63.67	4.38	444574	+58.9

Table 1: Comparisons between classic TS and TTS-BB

performed by TTS-BB compared with TS is related to the quality of the solution found. TS stops very quickly its improvement while TTS-BB continues its search, generating more steps. The good performance of TTS is due to the new tri-valued framework which allows the algorithm to benefit from a unified combination of the two resolution techniques.

The current implementation of the proposed hybridization framework does not compete well yet with the most powerful and highly optimized SAT solvers. However, the first experiments showed interesting results. Also the proposed framework allows the integration of some existing SAT solvers like Walksat as a basic local search component of a hybrid algorithm.

## References

- [1] Teresa Alsinet, Felip Many, and Jordi Planes. Improved branch and bound algorithms for MAX-SAT. In *Proc of the 6th International Conference on the Theory and Applications of Satisfiability Testing. SAT2003*, pages 408–415, 2003.
- [2] Brian Borchers and Judith Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 2:299–306, 1999.
- [3] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, Jul 1962.
- [4] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, San Francisco, 1978.
- [5] Jens Gottlieb, Elena Marchiori, and Claudio Rossi. Evolutionary algorithms for the satisfiability problem. *Evolutionary Computation*, 10(1):35–50, 2002.

- [6] Djamel Habet, Chu Min Li, Laure Devendeville, and Michel Vasquez. A Hybrid Approach for SAT. In *International Conference on Constraint Programming (CP), LNCS*, 2002.
- [7] Jin-Kao Hao, Frédéric Lardeux, and Frédéric Saubion. Evolutionary computing for the satisfiability problem. In *Applications of Evolutionary Computing*, volume 2611 of *LNCS*, pages 259–268, University of Essex, England, UK, 14-16 April 2003.
- [8] Bertrand Mazure, Lakhdar Sais, and Eric Grégoire. Tabu search for SAT. In *Proc. of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97)*, pages 281–285, Providence, Rhode Island, 1997.
- [9] Bertrand Mazure, Lakhdar Sais, and Eric Grégoire. Boosting complete techniques thanks to local search methods. *Annals of Mathematics and Artificial Intelligence*, 22:319–331, 1998.
- [10] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proc. of the AAAI, Vol. 1*, pages 337–343, 1994.
- [11] Laurent Simon, Daniel Le Berre, and Edward A. Hirsch. The SAT2002 competition. Technical report, Fifth International Symposium on the Theory and Applications of Satisfiability Testing, May 2002.
- [12] Laurent Simon, Daniel Le Berre, and Edward A. Hirsch. The SAT2003 competition. Technical report, Sixth International Conference on the Theory and Applications of Satisfiability Testing, May 2003.
- [13] William M. Spears. Simulated annealing for hard satisfiability problems. In *Second DIMACS implementation challenge : cliques, coloring and satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 533–558, 1996.
- [14] Hantao Zhang, Haiou Shen, and Felip Many. Exact algorithms for MAX-SAT. In Ingo Dahn and Laurent Vigneron, editors, *Electronic Notes in Theoretical Computer Science*, volume 86. Elsevier, 2003.
- [15] Jian Zhang and Hantao Zhang. Combining local search and backtracking techniques for constraint satisfaction. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pages 369–374, Menlo Park, August 4–8 1996. AAAI Press / MIT Press.