Three Truth Values for the SAT and MAX-SAT Problems

Frédéric Lardeux and **Frédéric Saubion** and **Jin-Kao Hao**
LERIA, University of Angers,
2 Bd Lavoisier,
F-49045 Angers Cedex 01
{lardeux,saubion,hao}@info.univ-angers.fr

Abstract

The aim of this paper is to propose a new resolution framework for the SAT and MAX-SAT problems which introduces a third truth value *undefined* in order to improve the resolution efficiency. Using this framework, we have adapted the classic algorithms Tabu Search and Walksat. Promising results are obtained and show the interest of our approach.

1 Introduction

The satisfiability problem (SAT) [Garey and Johnson, 1979] 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) $\mathcal{X} = \{x_1, ..., x_n\}$ and a Boolean formula $\phi: \mathbb{B}^n \rightarrow \mathbb{B}$. As usual, the formula ϕ is supposed to be in conjunctive normal form (CNF) (i.e., it is a conjunction of clauses where a clause is a disjunction of literals[1]). The formula is said to be satisfiable if there exists an assignment (i.e., a function $A: \mathcal{X} \rightarrow \mathbb{B}^n$) satisfying ϕ and unsatisfiable otherwise. The search space $\mathcal{S}$ corresponds to the set of all possible assignments.ϕ is obviously satisfied if all of its clauses are satisfied. In this context, the maximum satisfiability problem (MAX-SAT) corresponds to the minimization of the number of false clauses.

Two classes of methods are used to solve SAT and MAX-SAT problems, namely 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-Logemann-Loveland procedure [Davis *et al.*, 1962] which explores a binary search tree, building incrementally truth assignments. They provide very good results but are not suitable for the MAX-SAT problem. Other exact methods, generally based on Branch and Bound (B&B) algorithms [Borchers and Furman, 1999] have been designed to handle MAX-SAT, but their performances are often limited for large instances.

Approximate methods are mainly based on local search and evolutionary algorithms. We focus here on local search algorithms which have been widely studied in the SAT community [Selman *et al.*, 1992; 1994; Spears, 1996; Mazure *et al.*, 1997; Hirsch and Kojevnikov, 2001]. As usual, these algorithms rely on the fundamental characteristics of local search: the exploration of the search space is achieved by moving from an element to one of its neighbors, these moves being performed according to specific heuristics. Therefore, they expect to minimize, on the assignments space, a function corresponding to the number of false clauses. They are thus naturally designed for the MAX-SAT problem and allow one to handle large instances.

The efficiency of such algorithms is strongly related to their capacities to finely exploit specific areas of the search space when needed (i.e., intensify the search) and to widely explore the space by moving to different promising areas (i.e., diversify the search). These aspects are controlled through parameters to ensure a good tradeoff between diversification and intensification. But, actually these two notions are not really well defined and therefore, parameters tuning remains a key point when using such algorithms.

We should remark that exact and approximate methods do not use the same representations for their search space. Their hybridizations [Mazure *et al.*, 1998; Prestwich, 2000; Habet *et al.*, 2002; Prestwich, 2004] are not trivial since exact methods work on partial assignments incrementally completed by the resolution process while approximate algorithms explore the set of all possible complete assignments. A partial assignment is an assignment where some variables are not yet valued and can be thus considered as the representation of a set of complete assignments. From a local search point of view, the use of such partial assignments could lead to new diversification and intensification processes. Indeed, the diversification of the search could be achieved by moving over large sets of complete assignments induced by given partial assignments, while the valuation of some variables of such partial assignments could be considered as a way to focus on a more restricted part of the search space.

The purpose of this paper is to define a uniform model for these two types of assignments in order to precisely define and study the fundamental mechanisms of local search for SAT and MAX-SAT resolutions. This framework allows us to propose a new local search scheme which provides a uniform control of the search and which can be introduced in well known local search procedures: Tabu Search and Walksat.

[1] A literal is a variable or its negation.

2 A 3-Valued Framework

We propose a new resolution framework for the SAT and the MAX-SAT problems which introduces a third truth value *undefined* in order to improve the resolution efficiency. Within this framework, local search methods will be extended in order to take into account partial assignments and, therefore, several notions and logical rules have to be redefined.

2.1 Standard Local Searches

In the classic SAT and MAX-SAT context, the search space $\mathcal{S}$ is the set of all possible complete truth assignments. The objective function to maximize corresponds to the number of satisfied clauses given by the $standard\_eval$ function:

$$\begin{array}{rccl} standard\_eval: & \mathcal{S} & \rightarrow & I\!N \\ & A & \mapsto & |\{c|sat(A,c) \wedge c \in \phi\}| \end{array}$$

where $sat(A, c)$ means that the clause c is satisfied by the assignment $A \in \mathcal{S}$ and $|E|$ is the cardinality of the set E.

The moves are clearly possible flips of the values of a given assignment. The flip of a variable i in an assignment A is the swap of its truth value (*T (true)* to *F (false)* or *F* to *T*). The best flip is selected thanks to a $standard\_choose$ function which returns the variable whose flip provides the best improvement (i.e., maximizes the number of false clauses which become true after the flip minus the number of satisfied clauses which become false).

A naive local search algorithm for SAT and MAX-SAT consists in selecting the best move at each step (using $standard\_choose$). It stops either when a solution is found or when a maximum number of steps has been reached. Then, it returns the best assignment found (w.r.t. $standard\_eval$). Specific control heuristics can be added to improve the efficiency of the search, such as random walks and other noise strategies [Selman *et al.*, 1994].

2.2 Introducing an *undefined* Value

Our purpose is to provide a uniform framework to represent complete and partial assignments. Concerning complete assignments, with classical $true$ and $false$ values, the logical interpretation rules are well-known but, with partial assignments, non valued variables appear and we decide to introduce a third truth value to represent them.

We add a truth value *U(undefined)* and consider the set of truth values $\mathcal{T} = \{U, T, F\}$ in our *3-valued framework*. This new value induces some changes in the standard logical interpretation rules. Two approaches can be used to define these rules: an optimistic approach and a pessimistic approach.

- The *optimistic approach* is the most commonly used. Indeed, all the exact methods exploring a search tree are based on this approach. A clause is considered undefined if none of its literals is true and at least one of its literals is undefined. This approach corresponds to the following simplification rules:

$$\begin{array}{l} T \vee U \rightarrow T \\ F \vee U \rightarrow U \\ U \vee U \rightarrow U \end{array}$$

- We propose here an alternative *pessimistic approach* which relies on the fact that as soon as a clause has a false literal and no true literal, it is considered as false. An undefined clause must have all its literals set to undefined. This approach corresponds to the following rules:

$$\begin{array}{l} T \vee U \rightarrow T \\ F \vee U \rightarrow F \\ U \vee U \rightarrow U \end{array}$$

Our search structure is now defined and we may study the precise behavior of local search algorithms on this structure with three truth values.

2.3 Local Search Transitions

We define a partial order $\sqsupset$ on the set $\mathcal{T}$ such as $U \sqsupset T$ and $U \sqsupset F$. In this context, given a SAT problem with n variables, our search space will be the set $\mathcal{T}^n$. The ordering relation $\sqsupset$ can be naturally extended to $\mathcal{T}^n$: $(x_1, \cdots, x_n) \sqsupset (y_1, \cdots, y_n)$ if and only if $\exists i, x_i \sqsupset y_i$ and $\nexists j,\ y_j \sqsupset x_j$. We consider the partial ordering $(\mathcal{T}^n, \sqsupset)$ with the greatest element $\top = (U, \cdots, U)$. This structure will now be used to precisely describe the behavior of local search algorithms. We consider a basic local search move[2] as a transition $A \rightarrow A'$ where $A, A' \in \mathcal{T}^n$. Since we only consider here a classical neighborhood we require that transitions $(x_1, \cdots, x_n) \rightarrow (y_1, \cdots, y_n)$ satisfy $\exists i, x_i \neq y_i$ and $\forall j \neq i, x_j = y_j$. This neighborhood corresponds to a hamming distance equal to 1 between two neighbors, which is related to the $flip$ function.

On the one hand, we first consider the assignment landscape point of view. The 3-valued framework allows us to handle both partial and complete assignments and, as mentioned above, a partial assignments is seen as a representation of a set of smaller assignments (w.r.t. $\sqsupset$). Therefore, diversification consists in considering larger sets of elements of the search space while intensification focus on fewer elements. The transition corresponds to a layer exploration when the number of undefined variables remains the same.

- **Diversification:** $A \rightarrow A'$ with $A' \sqsupset A$
- **Intensification:** $A \rightarrow A'$ with $A \sqsupset A'$
- **Layer Exploration:** $A \rightarrow A'$ with $A \not\sqsupset A'$ and $A' \not\sqsupset A$

On the other hand, we have to take into account the evaluation of the assignments. The introduction of the third truth value requires us to redefine the $standard\_eval$ function and the $standard\_choose$ function of section 2.1, 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 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.

$$\begin{array}{rccl} eval: & \mathcal{S} & \rightarrow & (I\!N, I\!N) \\ & A & \mapsto & (|\{c|sat(A,c) \wedge c \in \phi\}|, \\ & & & |\{c|undefined(A,c) \wedge c \in \phi\}|) \end{array}$$

[2] With three truth values, the concepts of flip is not valid anymore. We prefer to use the concept of move, step or transition.

where $undefined(A, c)$ means that the clause c is undefined by the assignment $A \in \mathcal{S}$.

Using this function, we may define Sol_ϕ, the set of all the solutions for a formula ϕ: $Sol_\phi = \{A | eval(A) = (k, 0), A \in \mathcal{T}^n\}$ with $k = |\{c | c \in \phi\}|$.

We define the order $>_{eval}$ as the lexicographic extension $(>, >)$ of the order $>$ on the pair constructed by the $eval$ function. We may now consider the transitions from this evaluation point of view.

- **Improve:** $A \rightarrow A'$ with $A' >_{eval} A$
- **Deteriorate:** $A \rightarrow A'$ with $A >_{eval} A'$
- **Preserve:** $A \rightarrow A'$ with $eval(A') = eval(A)$

Using these concepts, a move is now characterized from each point of view: for instance, a given move may correspond to a diversification and improve transition.

These notions are illustrated in example 1. Indeed, in our 3-valued framework, both exact and approximate methods can be modeled. The search trees used by Davis-Putnam-Logemann-Loveland like procedures are sets of intensification and diversification transitions whereas classic local search algorithms perform sets of improving, deteriorating and preserving transitions at the lowest layer (with no undefined value).

Example 1: $(a \vee b) \wedge (\neg a \vee b) \wedge (a \vee \neg b)$

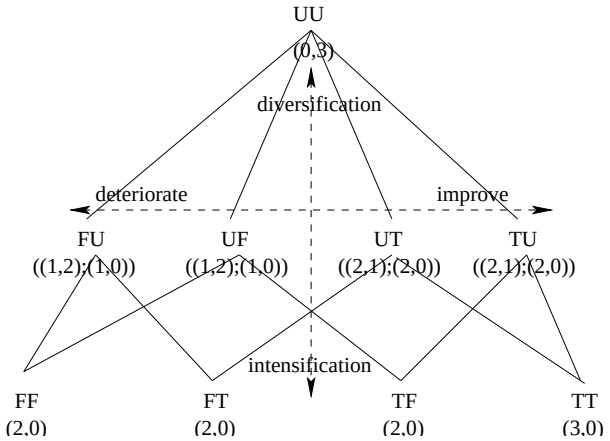

Figure 1: The 3-valued framework. All the elements of $\mathcal{T}^n$ are represented w.r.t. the ordering relation $\sqsupset$. A diversification transition increases the number of undefined variables whereas an intensification transition decreases this number. In this scheme, complete assignments are at the bottom and $\top = (UU)$ is on top. When the number of undefined variables remains stable, the elements are sorted by the order $>_{eval}$. The valuations of the $eval$ function are given as a couple (v_1;v_2) where v_1 (resp. v_2) corresponds to the evaluation of the current assignment under the optimistic (resp. pessimistic) approach. This couple is reduced to a single evaluation if $v_1 = v_2$. The best element is on the right side and the worst on the left side. Therefore, in this figure, best solutions are on rightmost bottom positions.

At this stage, one has to consider possible combinations of these transitions, issued from these two points of view, to build a general local search process.

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 new $choose$ function which selects a variable and a change of its value which maximizes the number of true clauses and minimizes the number of false clauses (increasing the number of undefined clauses if necessary).

At this step, the main components of a local search algorithm have been described. We will now study their behaviour and combination.

2.4 Evaluation of the Logical Rules

We want to measure the effect of the logical interpretation rules, presented in section 2.2, on the search process. As described before, a 3-valued local search (3LS) algorithm may return now partial assignments which represent indeed sets of assignments. To evaluate the quality of these partial assignments, it is necessary to explore exhaustively the corresponding subset in order to find the best complete assignment that it contains. This can be achieved thanks to a Branch and Bound procedure (B&B) which performs an exhaustive search by exploring a restricted search tree whose root is the partial assignment provided by the 3LS algorithm.

Table 1 compares the effects of the different logical rules. Tests are realized on standard random 3-sat instances with 500, 1000 and 2000 variables (F500.cnf, F1000.cnf and F2000.cnf). The 3LS algorithm used for these tests is a 3-valued Tabu Search (3TS-BB, see next section). The algorithm runs until 5000 iterations are performed. The number of undefined variables ($nb.\ var.\ U$), the number of clauses with one, two and three undefined variables ((U, UU, UUU)) and the quality ($(T, U, F)\ after\ 3TS\text{-}BB$) are memorized for the last partial assignment found. To estimate the potential quality of the partial assignment found by 3TS-BB, a B&B algorithm is executed and provides the best complete assignment ($(T, F)\ after\ B\&B$) by assigning the undefined variables.

Inst.	Observations	Optimistic	Pessimistic
500 var.	(T,U,F) after 3TS-BB	(2139,1,10)	(2145,0,5)
	nb. var. U	20	18
	(U,UU,UUU)	(188,2,0)	(177,4,0)
	(T,F) after B&B	(2140,10)	**(2145,5)**
1000 var.	(T,U,F) after 3TS-BB	(4227,5,18)	(4233,0,17)
	nb. var. U	33	40
	(U,UU,UUU)	(356,5,0)	(394,16,0)
	(T,F) after B&B	(4232,18)	**(4234,16)**
2000 var.	(T,U,F) after 3TS-BB	(8455,1,44)	(8465,0,35)
	nb. var. U	69	78
	(U,UU,UUU)	(662,16,0)	(774,20,0)
	(T,F) after B&B	(8456,44)	**(8465,35)**

Table 1: Comparisons between the two approaches.

Table 1 shows a clear dominance of the pessimistic interpretation. The quality of the partial assignment found during

the 3TS-BB is better since the number of true clauses found after B&B is greater than the corresponding number obtained with the optimistic interpretation.

2.5 Impact of Transitions in the Search Process

The optimistic and the pessimistic interpretations do not involve the same behaviour w.r.t. the different combinations of transitions characterizing a move, as described in section 2.3. Table 2 presents the evolution of the number of true, undefined and false clauses (T,U,F) for the optimistic (opt.) and the pessimistic (pes.) approaches and all the combinations of transitions excepted for the "preserve" transition since no change is performed in this case.

	Diversification			Intensification			Layer exploration		
	T	U	F	T	U	F	T	U	F
Imp. (opt.)	$=$	$>$	$<$	$>$	$\leq$	$\geq$	$\geq$	$\sim$	$\sim$
(pes.)	$=$	$>$	$<$	$>$	$\leq$	$\geq$	$>$	$=$	$<$
Det. (opt.)	$<$	$\geq$	$\leq$	$=$	$<$	$>$	$\leq$	$\sim$	$\sim$
(pes.)	$<$	$\geq$	$\sim$	$=$	$<$	$>$	$<$	$=$	$>$

Table 2: Possible evolutions of the number of true, undefined and false clauses (T,U,F) due to the different combinations of transitions. ($>$: increase, $\geq$: increase or remain stable, $=$: remain stable, $\leq$: decrease or remain stable, $<$: decrease and $\sim$: increase, decrease or remain stable)

We may assess common properties for the optimistic and pessimistic approaches. When applying an intensification transition, it is neither possible to create undefined clauses nor to delete true clauses. Concerning the diversification transition, true clauses cannot be created and undefined clauses cannot be deleted. Specific properties related to the optimistic and pessimistic interpretations can also be depicted: using the optimistic approach, false clauses cannot be generated during a diversification transition while in the pessimistic case, undefined clauses cannot be created nor deleted during the "Layer exploration".

3 3-valued Framework used in Well-known Algorithms

Tabu Search (TS) [Glover and Laguna, 1997; Mazure *et al.*, 1997] and Walksat [Selman *et al.*, 1994] are two well-known local search algorithms for SAT and MAX-SAT. To show the interest of the 3-valued framework, we have adapted these two algorithms in order to obtain 3TS-BB and 3Walksat-BB. The $choose$ and the $eval$ functions (2.3) were used to transform the two algorithms in their 3-valued versions.

3.1 3TS-BB

The 3-valued Tabu Search (3TS-BB) is a local search metaheuristics based on standard Tabu Search. Its aim is to maximize the number of true clauses and minimize the number of false clauses. The exploration of the search space is achieved by moving from an assignment to an other selected by the $choose$ function. These moves are guided by the $eval$ function which evaluates their benefit. In order to avoid problems induced by local optima, 3TS-BB uses a tabu list containing informations on previously visited configurations to forbid some possible loops in the search process. The length of the tabu list is set to 10% of the number of variables (empirical result inspired by [Mazure *et al.*, 1997]). One expects 3TS-BB to return a complete assignment to answer to the MAX-SAT problem. Then, we decide to regularly complete the partial assignments with a B&B procedure in order to generate a complete assignment. This completion is realized every s moves (empirically fixed to 5000). To reduce the execution time, B&B must not be used when there are too many undefined variables in the partial assignments. We impose that the maximum number of unassigned variables should not to be greater than a bound b (empirically fixed to 100). The search stops when a solution is found or when 10^6 moves are executed.

3.2 3Walksat-BB

The general process of the 3-valued Walksat (3Walksat-BB) uses similar principles. Like the standard Walksat, 3Walksat-BB is a randomized local search algorithm, which tries to determine the best move by randomly choosing an unsatisfied clause and selecting one of its variable thanks to the $choose$ function. We use as basis the version v41 of Walksat. The search is composed of 10 tries. Each try starts with a random assignment and after 10^5 moves, the best assignment found is completed with a classic B&B, if it is not a complete assignment. When a solution is found, the search stops. Like Walksat, 3Walksat-BB uses the "novelty" heuristic [Selman *et al.*, 1994] with a noise set to 0.5 (its default value).

Remark: It is important to note that, contrary to the standard local search algorithms, the 3-valued algorithms may start their searches with the $\top$ assignment $(U, \cdots, U)$. The influence of the initial assignment in the results is then avoided.

4 Experimental Results

Due to the approximate and non-deterministic nature of TS, 3TS-BB, Walksat and 3Walksat-BB, each algorithm runs 20 times on each benchmark. Tests are realized on a cluster with Linux and Alinka (5 nodes each of them with 2 CPU Pentium IV 2.2 GHz and 1 Gb of RAM) used sequentially. The maximum number of allowed local search steps is 10^6. For our algorithms, we consider that a backtrack performed during the B&B procedure has the same cost than a local search step.

All studied instances (Table 3) have been used for SAT competition 2003 [Simon *et al.*, 2003] and contain satisfiable and unsatisfiable instances. They come from different families of instances.

To study the four algorithms we present two tables (Tables 4 and 5): the first one is a comparison between TS and 3TS-BB and the second one between Walksat and 3Walksat-BB. Two criterions are used: the number of false clauses (fc.) and the number of steps (including local search moves and B&B backtracks) to obtain the best result (steps). For these two criterions we present the average (avg.) and its standard deviation (s.d.). Another interesting value is proposed in the last column: the percentage of improvement of the 3-version v.s the standard version, w.r.t. the average number of false clauses.

Families	Instances	sat03	var.	cl.
comb	comb1	419	5910	16804
	comb3	421	4774	16331
des-encryption	cnf-r4-b1-k1.1-comp	416	2424	14812
	cnf-r4-b1-k1.2-comp	417	2424	14812
ezfact-64	ezfact64_2	1518	3073	19785
	ezfact64_3	1519	3073	19785
	ezfact64_6	1522	3073	19785
	ezfact64_7	1523	3073	19785
kukula	am_6_6	362	2269	7814
	am_7_7	363	4264	14751
markstrom	mm-1x10-10-10-s.1	1488	1120	7220
	mm-2x2-5-5-s.1	1497	324	2064
okgen	okgen-c1935-v450-s569787048	1704	450	1935
	okgen-c2025-v450-s1380514806	1705	450	2025
Parity-32	par32-5-c	1539	1339	5350
	par32-5	1540	3176	10325
purdom	10142772393204023fw	1661	9366	37225
	10142772393204023nc	1662	8372	33248
	10142772393204023nw	1663	8589	34117
pyhala-braun	pyhala-braun-unsat-35-4-03	1543	7383	24320
	pyhala-braun-unsat-35-4-04	1544	7383	24320
	pyhala-braun-unsat-40-4-02	1546	9638	31795
	pyhala-braun-unsat-40-4-03	1547	9638	31795
qwh	qwh.40.544	1653	2843	22558
	qwh.40.560	1654	3100	26345
unif	unif-r4.25-v500-c2125-02-S1567634734	1111	500	2125
	unif-r4.25-v500-c2125-03-S948115330	1112	500	2125

Table 3: Instances selected from the SAT2003 competition. (sat03→ SAT2003 instance number, var.→variables, cl.→clauses)

Tables 4 and 5 show that the 3-valued framework permits to boost classic local search methods like TS and Walksat. Indeed, 3TS-BB provides an improvement which can be sometimes weak and exceptionally negative (-1.92% for `ezfact64_3`) but often really more significant for most of the instances ($+93.71\%$ for `par32-5`) and all the families. For Walksat, the improvement is less important with the 3-valued framework. However, 3Walksat-BB provides interesting results because 8 families are improved (4 more than 15%) and only 2 families are deteriorated (never more than -5%).

It is interesting to remark that the 3TS-BB and 3Walksat-BB results are better while the search power represented by the number of steps is in the same order of magnitude. Execution time is not mentioned in the tables but remains approximatively the same with and without the 3-valued framework.

5 Discussion and Conclusion

Our 3-valued framework allows us to handle partial assignments and, thanks to these partial assignments, some characteristics of the formula, like the backbone [Dubois and Dequen, 2001], can be easily defined. The backbone of a satisfiable formula ϕ is the set of entailed literals. A literal l is entailed by ϕ if and only if $\phi \wedge (\neg l)$ is unsatisfiable. In the 3-valued framework, the backbone is the element $A \in Sol_\phi$ such that $\nexists A' \sqsupset A, A' \in Sol_\phi$.

In this paper, we have proposed a new resolution framework for SAT and MAX-SAT problems which includes a third truth value $undefined$ in order to handle both partial and complete assignments. We have precisely studied the extension and behavior of local search algorithms using this third truth value. Two possible logical interpretations have been considered and it appears that the pessimistic approach provides better results than the optimistic approach.

We have then introduced this 3-valued framework inside Tabu Search and Walksat algorithms and the results are very promising. Our future works will consist in developing new specific heuristics using this 3-valued framework and in extending other well known algorithms in a 3-valued version. This framework could also serve as basis for full hybridizations of complete and incomplete resolution techniques.

Acknowledgments

The work presented in this paper is partially supported by the CPER COM program. We would like to thank the anonymous referees for their useful comments.

References

[Borchers and Furman, 1999] 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.

[Davis *et al.*, 1962] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Com. of the ACM*, 5(7):394–397, 1962.

[Dubois and Dequen, 2001] Olivier Dubois and Gilles Dequen. A backbone-search heuristic for efficient solving of hard 3-SAT formulae. In *Proc. of IJCAI'01*, pages 248–253, M. Kaufmann, 2001.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability , A Guide to the Theory of NP-Completeness*. Freeman & Co, 1979.

[Glover and Laguna, 1997] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.

[Habet *et al.*, 2002] Djamal Habet, Chu Min Li, Laure Devendeville, and Michel Vasquez. A Hybrid Approach for SAT. In *Proc. of CP'02, LNCS*, pages 172–184. Springer, vol. 2470, 2002.

[Hirsch and Kojevnikov, 2001] Edward A. Hirsch and Arist Kojevnikov. UnitWalk: A new SAT solver that uses local search guided by unit clause elimination. Steklov Institute of Mathematics at St.Petersburg, 2001.

[Mazure *et al.*, 1997] Bertrand Mazure, Lakhdar Sais, and Eric Grégoire. Tabu search for SAT. In *Proc. of AAAI-97/IAAI-97*, pages 281–285, AAAI Press, 1997.

[Mazure *et al.*, 1998] 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.

[Prestwich, 2000] Steve Prestwich. A hybrid search architecture applied to hard random 3-sat and low-autocorrelation binary sequences. In *Proc. of CP'00, LNCS*, pages 337–352. Springer, vol. 1894, 2000.

[Prestwich, 2004] Steve Prestwich. Exploiting relaxation in local search. In *Proc. of the 1rst Int. Workshop on Local Search Techniques in Constraint Satisfaction*, 2004.

[Selman *et al.*, 1992] Bart Selman, Hector J. Levesque, and David G. Mitchell. A new method for solving hard satisfiability problems. In *Proc. of AAAI'92*, pages 440–446, AAAI Press, 1992.

Families	sat03	3TS-BB fc. avg.	3TS-BB fc. s.d.	3TS-BB steps($\times 10^3$) avg.	3TS-BB steps($\times 10^3$) s.d.	TS fc. avg.	TS fc. s.d.	TS steps($\times 10^3$) avg.	TS steps($\times 10^3$) s.d.	Imp. fc. %	
purdom	1661	304.00	4.00	665.05	23.72	497.50	16.50	273.81	172.52	+38.89	**+33.39**
	1662	256.33	9.09	462.10	300.75	350.00	3.00	401.52	259.26	+26.76	
	1663	247.50	1.50	663.28	122.31	378.00	10.00	669.85	60.97	+34.52	
kukula	362	22.58	2.36	346.57	291.14	48.00	8.72	803.18	225.66	+52.96	**+52.88**
	363	53.33	2.93	353.74	282.04	113.00	16.20	720.32	178.27	+52.81	
des-encryption	416	35.75	2.83	372.87	197.30	53.33	2.29	1.32	0.27	+32.96	**+30.54**
	417	36.42	2.38	202.87	258.25	50.67	4.38	66.50	216.62	+28.12	
comb	419	123.90	5.79	143.10	145.53	160.20	8.70	328.83	233.63	+22.66	**+25.41**
	421	79.33	5.01	243.34	190.96	110.42	4.83	156.43	87.09	+28.16	
ezfact-64	1518	105.00	3.37	463.88	299.16	104.58	18.72	631.44	333.80	−0.40	+1.05
	1519	102.00	4.47	388.21	209.20	100.08	11.21	669.38	315.35	−1.92	
	1522	108.83	5.26	383.12	319.05	112.33	10.92	672.35	249.31	+3.12	
	1523	110.00	4.64	488.94	287.57	113.42	15.67	623.18	276.93	+3.02	
markstrom	1488	9.92	0.82	382.26	255.69	16.08	5.12	304.65	303.09	+38.31	**+29.93**
	1497	6.08	0.78	359.18	266.21	7.75	1.59	102.11	198.37	+21.55	
okgen	1704	4.33	1.04	306.77	249.03	4.58	1.20	340.69	283.10	+5.46	**+5.73**
	1705	5.17	0.52	315.29	173.85	5.50	0.87	361.38	250.97	+6.00	
Parity-32	1539	9.67	0.81	488.20	290.15	65.58	1.23	173.14	213.85	+85.25	**+89.48**
	1540	9.75	1.09	509.59	225.89	155.00	1.68	192.88	262.24	+93.71	
pyhala-braun	1543	239.00	3.00	92.11	20.37	377.00	0.00	269.85	30.10	+36.60	**+34.17**
	1544	197.00	3.74	355.73	304.16	347.00	3.00	617.31	373.33	+43.23	
	1546	342.50	10.50	25.80	4.98	464.00	0.00	571.45	0.03	+26.19	
	1547	330.00	4.00	53.95	7.58	476.00	0.00	294.21	0.04	+30.67	
qwh	1653	125.67	9.67	52.49	63.89	130.08	10.03	33.86	60.82	+3.39	**+0.95**
	1654	132.00	9.05	56.67	69.93	130.08	10.84	48.06	119.94	−1.48	
unif	1111	2.75	0.44	283.00	310.58	3.58	1.26	289.46	343.48	+23.18	**+45.44**
	1112	1.75	0.60	379.46	353.53	5.42	1.43	135.37	81.25	+67.71	

Table 4: Comparisons between TS and 3TS-BB.

Instances	sat03	3Walksat-BB fc. avg.	3Walksat-BB fc. s.d.	3Walksat-BB steps($\times 10^3$) avg.	3Walksat-BB steps($\times 10^3$) s.d.	Walksat fc. avg.	Walksat fc. s.d.	Walksat steps($\times 10^3$) avg.	Walksat steps($\times 10^3$) s.d.	Imp. fc. %	
purdom	1661	372.45	12.65	635.00	270.69	412.05	12.77	487.16	248.84	+9.61	**+21.61**
	1662	327.85	14.54	455.00	257.83	374.55	8.45	508.58	295.37	+12.47	
	1663	353.00	15.70	540.00	281.78	393.70	9.07	656.23	247.86	+10.34	
kukula	362	1.10	0.44	333.41	265.67	1.80	2.38	409.08	263.57	+38.89	**+23.42**
	363	71.80	14.85	519.66	292.22	78.00	14.95	555.39	304.03	+7.95	
des-encryption	416	24.55	2.42	337.15	288.45	23.70	1.76	419.57	257.28	−3.59	−4.87
	417	23.95	1.66	551.11	322.33	23.35	1.68	458.96	295.70	−2.57	
comb	419	69.35	5.96	548.77	303.13	70.45	7.07	569.64	273.53	+1.56	**+0.33**
	421	39.60	4.09	452.94	309.93	39.25	5.62	527.25	338.57	−0.89	
ezfact-64	1518	70.95	3.92	544.84	279.54	86.95	4.69	393.75	280.57	+18.40	**+18.43**
	1519	72.70	4.56	629.37	294.40	90.40	3.57	520.31	276.70	+19.58	
	1522	73.95	4.24	518.53	284.14	88.60	2.62	308.05	267.34	+16.53	
	1523	72.55	3.93	548.05	280.23	89.80	3.67	587.71	287.36	+19.21	
markstrom	1488	10.70	1.49	570.15	239.46	11.45	1.63	400.54	269.03	+6.55	**+2.75**
	1497	4.85	0.57	423.88	302.64	4.80	0.51	214.27	240.11	−1.04	
okgen	1704	3.00	0.00	51.10	31.28	3.00	0.00	48.65	36.04	0.00	0
	1705	4.00	0.00	147.20	140.04	4.00	0.00	191.90	168.66	0.00	
Parity-32	1539	13.80	1.89	500.11	276.64	13.85	1.68	486.16	278.46	+0.36	**+0.1**
	1540	15.10	2.07	349.38	267.51	15.05	2.18	493.77	284.38	−0.33	
pyhala-braun	1543	175.85	7.27	452.92	302.98	216.70	11.33	505.76	300.51	+18.85	**+19.48**
	1544	172.10	7.63	555.63	293.43	215.90	10.19	542.80	313.90	+20.29	
	1546	238.90	9.32	660.00	259.62	293.25	9.15	551.42	280.12	+18.53	
	1547	239.70	10.85	625.00	289.61	300.55	8.69	552.30	263.27	+20.25	
qwh	1653	138.40	4.42	509.55	267.21	138.75	3.24	402.36	239.36	+0.25	**+0.63**
	1654	135.70	4.75	527.37	281.81	137.10	4.58	535.02	270.60	+1.02	
unif	1111	1.55	0.50	240.75	292.05	1.45	0.50	252.81	255.21	−6.90	−3.45
	1112	1.00	0.00	21.61	19.49	1.00	0.00	21.20	17.01	0.00	

Table 5: Comparisons between Walksat and 3Walksat-BB.

[Selman *et al.*, 1994] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *Proc. of AAAI'94*, pages 337–343, AAAI Press, 1994.

[Simon *et al.*, 2003] 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, 2003.

[Spears, 1996] 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, vol. 26, 1996.