

Local Search with Three Truth Values for SAT Problems

Frédéric Lardeux*

Frédéric Saubion*

Jin-Kao Hao*

*LERIA, University of Angers

2 Boulevard Lavoisier, 49045 Angers Cedex 01, France

Firstname.Name@univ-angers.fr

1 Introduction

The satisfiability problem (SAT) [1] consists in finding a truth assignment that satisfies a well-formed boolean expression. An instance of the SAT problem is defined by a boolean formula ϕ over a set of boolean variables $\mathcal{X} = \{x_1, \dots, x_n\}$. ϕ is in conjunctive normal form (CNF) if it is a conjunction of clauses where a clause is a disjunction of literals (a variable or the negation of a variable). In this paper, ϕ is supposed to be in CNF. A truth assignment for a CNF ϕ is a function $A: \mathcal{X} \rightarrow \{0, 1\}$. ϕ is said to be satisfiable if there exists an assignment A satisfying ϕ , i.e. all its clauses are true under A (a clause is true if at least one of its literals is true). Otherwise, the formula is said unsatisfiable. The maximum satisfiability problem (MAX-SAT) corresponds to finding an assignment that maximizes the number of true clauses (and equivalently minimizes the number of false clauses) for the given formula.

Two classes of methods are used to solve SAT and MAX-SAT problems, namely exact and approximate methods. While exact methods work on partial assignments (containing unassigned variables) incrementally completed by the resolution process, approximate algorithms explore the set of all possible complete assignments (composed only of 0 and 1 truth values). Since a partial assignment contains unassigned variables, such an assignment can be considered as representing a *set* of complete assignments. From a local search perspective, exploring partial assignments could lead to new diversification and intensification processes. Diversification could be achieved by moving over large sets of complete assignments induced by a partial assignment, while the valuation of some variables of a partial assignment could be considered as a way to focus on a more restricted part of the search space (intensification).

In this paper, we introduce a 3-valued logic framework in which a third truth value $U(\text{undefined})$ is added. This 3-valued framework allows us to design local search algorithms handling both complete and partial assignments. It permits the introduction of new mechanisms for diversification and intensification. It offers a natural way to conciliate exact and approximate methods working respectively on complete and partial assignment. As a proof-of-concept implementation, we show how this 3-valued framework can be easily integrated into well-known Local Search algorithms such as Tabu Search and Walksat for SAT and MAX-SAT.

Vienna, Austria, August 22–26, 2005

2 Local Search with Three Truth Values

2.1 Evaluation function

With the 3-valued framework, we introduce, in addition to $\{1,0\}$, a third truth value U (*undefined*) and consider the set of truth values $\mathcal{T} = \{1, 0, U\}$. The main purpose here is to provide a uniform way to represent both complete and partial assignments. Notice first that well-known logical rules are available to evaluate a given boolean formula ϕ for any complete assignment A . This is simply done by an evaluation function (call it *eval*) that counts the number of true clauses of ϕ under the assignment A . This evaluation function is typically used by local search algorithm to guide the search process.

However, when a partial assignment is given, it is no more clear as to how to evaluate the boolean formula ϕ , since the truth value of some clauses may be unknown. Therefore, with the 3-valued framework we must redefine the evaluation function *eval* taking into account unassigned variables.

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. This evaluation function will be used by our 3-valued local search algorithms presented in Section 3.

2.2 Moves and Neighborhood

First notice that within the 3-valued framework, we are working on the space \mathcal{T}^n (n being the number of the variables of the given boolean formula). Consequently, the neighborhood must take into account both partial and complete assignments. Within this context, consider a basic local search move or transition: $s \rightarrow s'$ where $s, s' \in \mathcal{T}^n$ such that the transition $s = (x_1, \dots, x_n) \rightarrow s' = (y_1, \dots, y_n)$ satisfies $\exists i, x_i \neq y_i$ and $\forall j \neq i, x_j = y_j$. This transition corresponds in fact to an extension of the classical "flip" neighborhood used in local search algorithms for SAT that flips the truth value of a variable ($0 \rightarrow 1$ or $1 \rightarrow 0$). Notice that here while the hamming distance equal always to 1 between two neighbors, we have now six move possibilities from any point from \mathcal{T}^n : $0 \rightarrow 1, 0 \rightarrow U, 1 \rightarrow 0, 1 \rightarrow U, U \rightarrow 0, U \rightarrow 1$.

Using the *eval* function defined in Section 2.2, we may define an order $>_{eval}$ as the lexicographic extension ($>, >$) of the order $>$ on the pair constructed by *eval*. From this point of view, there are three possibilities for a move according to the *eval* function.

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

Vienna, Austria, August 22–26, 2005

2.3 Characterizing 3-valued Local Search Algorithms

Now let us characterize local search algorithms with the 3-valued framework. First, we define a partial order \sqsupseteq on the set \mathcal{T} such as $U \sqsupseteq 1$ and $U \sqsupseteq 0$. In this context, given a SAT problem with n variables, our search space will be the set \mathcal{T}^n . Then, we extend the ordering relation \sqsupseteq to \mathcal{T}^n : $(x_1, \dots, x_n) \sqsupseteq (y_1, \dots, y_n)$ if and only if $\exists i, x_i \sqsupseteq y_i$ and $\nexists j, y_j \sqsupseteq x_j$. This partial ordering $(\mathcal{T}^n, \sqsupseteq)$ has $\top = (U, \dots, U)$ as its maximal element. Any complete assignment constitutes a minimal element of the ordering. Finally, this ordering can be viewed as a layered structure with n levels (see Fig. 1).

From this point of view, a local search algorithm will go through the layered structure by a series of moves, each move corresponding to a step of diversification (moving up), intensification (moving down) or layer exploration (side-walk):

- Diversification: $s \rightarrow s'$ with $s' \sqsupseteq s$ (increasing the number of unassigned variables)
- Intensification: $s \rightarrow s'$ with $s \sqsupseteq s'$ (decreasing the number of unassigned variables)
- Layer Exploration: $s \rightarrow s'$ with $s \not\sqsupseteq s'$ and $s' \not\sqsupseteq s$ (the number of unassigned variables remains unchanged)

Therefore, contrary to classical local search algorithms that perform sets of improving, deteriorating and preserving transitions at the lowest layer (only complete assignments), a 3-valued local search algorithm will be able to work at any level of this structure, offering thus more facilities for diversification, intensification as well the switching between these two fundamental phases. It is interesting to notice that a 3-valued local search algorithm may begin its search with any assignment from \mathcal{T}^n , which may be complete or partial.

Finally, notice that using the 3-valued framework, we can also model tree search exact algorithms such as the Davis-Putnam-Loveland and Branch-Bound procedures. This opens new ways to design hybrid algorithms combining exact and approximate methods.

3 Implementing 3-valued Local Search Algorithms

As a proof-of-concept implementation, we present two 3-valued local search algorithms based on Tabu Search (TS) [2] and Walksat [4] for SAT and MAX-SAT (call them respectively 3TS and 3Walksat). The main purpose here is to assess the gain of the 3-valued versions compared with their versions. Both algorithms use the *eval* evaluation function and the neighborhood function defined in Section 2.

3.1 3TS

The 3-valued Tabu Search (3TS) is a standard TS algorithm working in $\{0, 1, U\}^n$. Its aim is to maximize the number of true clauses and minimize the number of false clauses during the search. The exploration of the search space is achieved by moving from an assignment

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

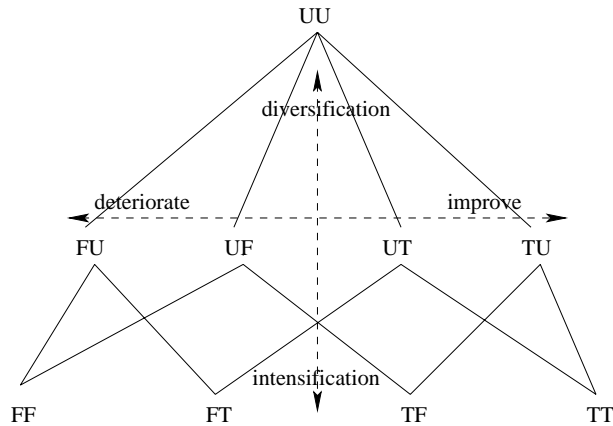


Figure 1: The 3-valued framework. All the elements of \mathcal{T}^n are represented w.r.t. the ordering relation \sqsubset .

to another, following the neighborhood function and guided by the *eval* function. In order to avoid problems induced by local optima, 3TS 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 [3].

Since 3TS alone is not guaranteed to return a complete assignment, 3TS regularly calls a Branch&Bound (B&B) subroutine 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 unassigned 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 a fixed number of iterations (fixed at 10^6 in this paper) are executed.

3.2 3Walksat

The 3-valued Walksat (3Walksat) uses similar principles. Like Walksat, 3Walksat is a randomized local search algorithm, which tries to determine the best move by randomly choosing an unsatisfied clause and selecting then one of its variable to flip. 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 the B&B procedure indicated before, if it is not a complete assignment. The search stops, when a solution is found. Like Walksat, 3Walksat uses the “novelty” heuristic [4] with a noise set to 0.5 (its default value).

4 Experimental Results

The purpose of this section is to assess the potential of gain of two previous algorithms (3TS and 3Walksat) with their classical versions (TS and Walksat). Each algorithm is run 20 times

Vienna, Austria, August 22–26, 2005

on each benchmark of 12 instances (Table 1) taken from SAT competition 2003 [5]. The maximum number of allowed local search steps for each run is fixed at 10^6 .

To study the four algorithms we present two tables (table 2 and table 3): the first one is a comparison between TS and 3TS and the second one between Walksat and 3Walksat. Two criteria are used: the number of false clauses (fc.) (to be minimized) and the number of steps (a backtrack performed by the B&B procedure counts as a local search step) to obtain the best result (steps). For these two criteria we present the average (avg.) and its standard deviation (s.d.). Another 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.

Instances	SAT03	var.	cl.
10142772393204023fw	1661	9366	37225
am_6_6	362	2269	7814
cnf-r4-b1-k1.1-comp	416	2424	14812
comb1	419	5910	16804
ezfact64_3	1519	3073	19785
ezfact64_6	1522	3073	19785
mm-lx10-10-10-s.1	1488	1120	7220
okgen-c2025-v450-s1380514806-1380514806	1705	450	2025
par32-5	1540	3176	10325
pyhala-braun-unsat-35-4-04	1544	7383	24320
qwh.40.544	1653	2843	22558
unif-r4.25-v500-c2125-03-S948115330	1112	500	2125

Table 1: Instances selected from the SAT2003 competition. (var. \rightarrow variables, cl. \rightarrow clauses)

Instances	3TS				TS				Imp. fc. %
	fc.		steps($\times 10^3$)		fc.		steps($\times 10^3$)		
	avg.	s.d.	avg.	s.d.	avg.	s.d.	avg.	s.d.	
1661	304.00	4.00	665.05	23.72	497.50	16.50	273.81	172.52	+38.89
362	22.58	2.36	346.57	291.14	48.00	8.72	803.18	225.66	+52.96
416	35.75	2.83	372.87	197.30	53.33	2.29	1.32	0.27	+32.96
419	123.90	5.79	143.10	145.53	160.20	8.70	328.83	233.63	+22.66
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
1488	9.92	0.82	382.26	255.69	16.08	5.12	304.65	303.09	+38.31
1705	5.17	0.52	315.29	173.85	5.50	0.87	361.38	250.97	+6.00
1540	9.75	1.09	509.59	225.89	155.00	1.68	192.88	262.24	+93.71
1544	197.00	3.74	355.73	304.16	347.00	3.00	617.31	373.33	+43.23
1653	125.67	9.67	52.49	63.89	130.08	10.03	33.86	60.82	+3.39
1112	1.75	0.60	379.46	353.53	5.42	1.43	135.37	81.25	+67.71

Table 2: Comparisons between 3TS and TS.

Table 2 and 3 show that the 3-valued framework permits to boost the classic local search algorithms. Indeed, 3TS provides an improvement which can be sometimes weak and exceptionally negative (-1.92% for 1519 (ezfact64_3)) but often really significant for most of the tested instances ($+93.71\%$ for 1540 (par32-5)). Also, the robustness of the search algorithms seems improved with smaller standard deviation. For Walksat, the improvement is less important with the 3-valued framework. Even so, 3Walksat provides interesting results because 8 instances are improved (6 more than 5%) and only 2 instances are deteriorated (never less than -5%).

Instances	3Walksat				Walksat				Imp. fc. %
	fc.		steps($\times 10^3$)		fc.		steps($\times 10^3$)		
	avg.	s.d.	avg.	s.d.	avg.	s.d.	avg.	s.d.	
1661	372.45	12.65	635.00	270.69	412.05	12.77	487.16	248.84	+ 9.61
362	1.10	0.44	333.41	265.67	1.80	2.38	409.08	263.57	+ 38.89
416	24.55	2.42	337.15	288.45	23.70	1.76	419.57	257.28	-3.59
419	69.35	5.96	548.77	303.13	70.45	7.07	569.64	273.53	+ 1.56
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
1488	10.70	1.49	570.15	239.46	11.45	1.63	400.54	269.03	+ 6.55
1705	4.00	0.00	147.20	140.04	4.00	0.00	191.90	168.66	0.00
1540	15.10	2.07	349.38	267.51	15.05	2.18	493.77	284.38	-0.33
1544	172.10	7.63	555.63	293.43	215.90	10.19	542.80	313.90	+ 20.29
1653	138.40	4.42	509.55	267.21	138.75	3.24	402.36	239.36	+ 0.25
1112	1.00	0.00	21.61	19.49	1.00	0.00	21.20	17.01	0.00

Table 3: Comparisons between 3Walksat and Walksat

5 Conclusion

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 characterized local search algorithms under this 3-valued framework and shown that diversification and intensification can be naturally conciliated via a simple move mechanism. We have also presented two proof-of-concept implementations based on Tabu Search and Walksat. Experimental results on benchmarks instances showed that the 3-valued versions of these algorithms do boost the performances of their initial versions.

In addition to these results, we believe this study opens interesting research perspectives. In particular, the proposed framework could serve as a basis for full hybridizations of exact and approximate methods.

References

- [1] Michael R. Garey and David S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness*. W.H. Freeman & Company, San Francisco, 1979.
- [2] Fred Glover and Manuel Laguna. *Tabu Search*. Kluwer Academic Publishers, 1997.
- [3] 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.
- [4] 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.
- [5] 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.

Vienna, Austria, August 22–26, 2005