DIMACS Series in Discrete Mathematics
and Theoretical Computer Science
Volume 00, 0000

Presented at DIMACS Workshop on Satisfiability Problem: Theory and Applications, March 1996, Rutgers University

Solving MAX-SAT with non-oblivious functions and history-based heuristics

R. BATTITI AND M. PROTASI

ABSTRACT. The MAX-Satisfiability problem has been thoroughly investigated from theoretical and practical points of view. In the first area, new *approximation* algorithms have been recently proposed that achieve in the worst-case a solution whose value is slightly better than 3/4 of the optimal one. In the second area a wide selection of *heuristics* have been designed with a performance that is of interest for many practical applications.

The research in this paper analyzes the use of *non-oblivious* local search, an approximation technique with better theoretical properties with respect to the standard local search, in the framework of heuristics. First the average approximation performance is studied through a statistical analysis. Then a combined two-phase local search using both *non-oblivious* and *oblivious* functions is investigated.

The satisfactory results obtained in these simple tests motivate a more complex integration of the basic methods. In particular a "reactive" algorithm based on Tabu Search (Non-Oblivious Reactive Tabu Search, NRTS for short) is designed with periodic use of non-oblivious functions and an automated diversification scheme. Extensive experimental results as a function of the allotted number of iterations show a better performance in comparison with competitive heuristics like GSAT-with-walk, on a benchmark of random MAX–3–SAT problems.

1. Introduction

The interest for the MAX–Satisfiability problem (MAX–SAT) stems from many reasons. The starting point was that the related satisfiability problem (SAT), as it is well known, was the first example of an NP–complete problem and therefore it opened the long list that marks the border between computational tractability and intractability. On the other hand many issues in mathematical logic and artificial intelligence can be expressed in the form of satisfiability or some of its variants, like constraint satisfaction. There has recently been a renaissance in the study of effective heuristic algorithms based on local search and random diversification techniques for SAT and MAX-SAT. Starting from the late eighties

1991 *Mathematics Subject Classification.* Primary 90C09; Secondary 90C27, 68P10, 68R05, 68T99, 94C10, 94C12.

Key words and phrases. Satisfiability, logic, MAX-SAT, greedy, artificial intelligence, local search, reactive search, history-based heuristics, non-oblivious functions, Tabu Search, computer implementation.

both discrete and continuous-based greedy algorithms have been proposed, see for example [**13, 14, 7, 24**] and the reviews in [**15**] and [**22**].

The focus of this paper is twofold: first we examine the average performance of the recently proposed *non-oblivious* functions for local search and demonstrate that, beyond possessing a better worst-case behavior [**19**], these functions lead to local optima of better average quality with respect to standard local search, for a benchmark of random MAX–3–SAT problems. Then we investigate novel integrations of standard and non–oblivious functions in simple local search schemes and in complex heuristics and propose a new heuristic (N-RTS) that integrates Tabu Search with the periodic activation of non–oblivious functions and the "reactive" triggering of a strong diversification phase. This integration achieves better average performance than existing algorithms.

In the following sections, first we summarize some theoretical results on MAX-SAT (Sec. 2), then we investigate the use of *non–oblivious* functions in basic local search schemes (Sec. 3) and in complex heuristics (Sec. 4). A comparative analysis of the different algorithms is executed by analyzing the average approximation behavior as a function of the number of elapsed iterations, on a benchmark of random MAX-3-SAT tasks used in [**17**] and [**25**].

2. Some theoretical results on MAX–SAT

While SAT is a decision problem, MAX–SAT is its optimization version, in which one looks for an assignment that maximizes the number of satisfied clauses of a boolean formula in conjunctive normal form.

Now, finding the optimal solution for optimization problems with an NP–complete decision version is too computationally expensive (the existing exact algorithms require exponential time). Therefore an important research area starting between the end of the seventies and the beginning of the eighties was devoted to solve these problems in an "approximate way," for a survey, see, for example, [**2**]. However, this approach is effective if two conditions hold: a) the approximation algorithm must be efficient; b) the value of the solution found by the algorithm must be "close" to the optimal one.

Condition a) can be formalized by asking that the algorithm runs in polynomial time. For condition b) one has to introduce some measure of the goodness of the approximation. One of the most common approaches is based on the notion of *performance ratio.*

Let us now summarize some results in this area. In the following, by approximation algorithm, we mean an algorithm that finds a feasible solution $\mathbf{x}$.

Definition 2.1. *Let P be a maximization problem whose decision version is NP–complete. Let A be an approximation algorithm for P. Moreover, given any instance I of P, $m(A(I))$ denotes the value achieved by algorithm A on I and $m^*(I)$ denotes the optimal value on I. The* performance ratio *of A with respect to I is the ratio*

$$R(I, A(I)) = \frac{m(A(I))}{m^*(I)}$$

When the worst-case ratio is considered one has the following definition:

DEFINITION 2.2. *Let P be a maximization problem whose decision version is NP–complete. P is solved by an approximation algorithm A with* performance ratio R *if, for any instance $I \in P$*

$$R(I, A(I)) \geq R$$

Similar definitions can be given for minimization problems. The performance ratio ranges between 0 and 1. Of course, when R is close to 1, algorithm A achieves feasible solutions whose values are very good.

Many polynomial–time approximation algorithms have been introduced for the MAX–SAT problem. The first important algorithm [**18**] based on a greedy approach achieves a performance ratio 1/2. Recently this bound has been considerably improved . In [**28**] and in [**10**] two different algorithms achieve a performance ratio 3/4 for any MAX–SAT formula, while further small improvements are given in [**11**] and [**6**]. These results can be ameliorated if we restrict ourselves to consider special cases. Up to now, the "record" for MAX–2–SAT is 0.931, see [**6**], while that for MAX–3–SAT is 0.801, see [**27**].

In practice, for many instances, such algorithms find a feasible solution whose value is considerably better than that guaranteed by the worst–case performance ratio.

On the other hand, negative results for MAX–SAT have been proven by showing that this problem cannot be approximated beyond a certain level. In particular the following results are obtained in [**5**]:

THEOREM 2.1. *If $P \neq NP$ then no polynomial time approximation algorithm for MAX–SAT can exist with performance ratio $\frac{77}{80} + \epsilon$ for every $\epsilon > 0$.*

while, for the special case of MAX–2–SAT one has:

THEOREM 2.2. *If $P \neq NP$ then no polynomial time approximation algorithm for MAX–2–SAT can exist with performance ratio $\frac{217}{220} + \epsilon$ for every $\epsilon > 0$.*

From a theoretical point of view, the approximation complexity of MAX–SAT is therefore not completely determined because there is still a gap between the bound that cannot be achieved and the bound found by the best approximation algorithms.

2.1. Non-oblivious functions. In the design of efficient approximation algorithms for MAX–SAT a recent approach of interest is based on the use of *non-oblivious functions* independently introduced in [**19**] and in [**1**].

Let us consider a classical local search algorithm for MAX–SAT, denoted as *oblivious* local search (OB). Let n be the number of variables and m be the number of clauses. The algorithm starts from an arbitrary feasible solution $\mathbf{x} \in \{0,1\}^n$ (that is, a truth assignment) and looks for a better solution in a neighborhood N of $\mathbf{x}$. The most common neighborhoods considered in this framework are the d–neighborhoods. By definition, a d–neighborhood consists of all truth assignments that differ from $\mathbf{x}$ in at most d points. In particular, a 1–neighborhood is obtained by applying to $\mathbf{x}$ the elementary moves that change the value of a single variable x_j. At each iteration the algorithm evaluates the points in the neighborhood by calculating the number of satisfied clauses. Then a point in the

neighborhood becomes the new current solution if more clauses are satisfied. The procedure stops when the achieved best solution cannot be improved by moving to a neighbor. Clearly, the feasible solution found by OB local search typically is only a *local* and not a *global* optimum.

Now, a different type of local search can be obtained by using a *different* objective function to direct the search, i.e., to select the best move at each iteration. Local optima of the "standard" objective function are not necessarily local optima of the different objective function. In this event, the second function causes an *escape* from a given local optimum. Clearly, in order to be useful, the second function has to improve the performance of the algorithm. Interestingly enough, this is indeed the case for suitable functions if one considers both the worst-case performance ratio and, at it will be shown in this paper, the actual results obtained on benchmark instances. With a clever choice of the second *non-oblivious* function the search trajectory can be steered in other directions and the performance of the local search can be improved.

Let us start with MAX–2–SAT. Given an assignment $\mathbf{x}$, let S_i denote the set of clauses in the given task in which exactly i literals are true and let $w(S_i)$ denote the total number of clauses in S_i. Then the theoretically-derived non–oblivious function for MAX–2–SAT is:

$$\frac{3}{2}w(S_1) + 2w(S_2)$$

The following result ([**19**]) holds:

THEOREM 2.3. *The performance ratio for any oblivious local search algorithm with a d-neighborhood for MAX–2–SAT is* $2/3$ *for any* $d = o(n)$. *Non-oblivious local search with an 1-neighborhood achieves a performance ratio* $3/4$ *for MAX–2–SAT.*

Therefore non-oblivious local search (NOB) improves considerably the performance ratio even if the search is restricted to a much smaller neighborhood. With a suitable generalization of the non-oblivious function the above theorem can be extended:

THEOREM 2.4. *Non–oblivious local search achieves a performance ratio* $1-\frac{1}{2^k}$ *for MAX–k–SAT.*

In detail, the oblivious function for MAX–3–SAT is:

$$w(S_1) + \frac{9}{7}w(S_2) + \frac{10}{7}w(S_3)$$

while that for MAX–4–SAT is:

$$\frac{45}{12}w(S_1) + \frac{56}{12}w(S_2) + \frac{61}{12}w(S_3) + \frac{64}{12}w(S_4)$$

It is important to stress that these results fall in the worst-case approach and that they cannot be immediately translated into competitive heuristics. However, in the following sections, we will see that the use of non–oblivious functions can be fruitfully applied to the design of heuristics for MAX–SAT, therefore establishing a remarkable connection between approximation algorithms and heuristics.

For the following discussion, let us denote with f_{OB} or , simply, f the "standard" oblivious function given by the number of satisfied clauses, and by f_{NOB} the above introduced non–oblivious function.

2.2. Example. Let us consider the following task with number of variables $n = 5$, and clauses $m = m^* = 4$:

$$(\overline{x}_1 \vee \overline{x}_2 \vee x_3), (\overline{x}_1 \vee \overline{x}_2 \vee x_4), (\overline{x}_1 \vee \overline{x}_2 \vee x_5), (\overline{x}_3 \vee \overline{x}_4 \vee \overline{x}_5)$$

Let us assume that the assignment $\mathbf{x} = (11111)$ is reached by OB local search. It is immediate to check that $\mathbf{x} = (11111)$ is an oblivious local optimum with one unsatisfied clause (clause-4). While OB stops here, a possible sequence to reach the global optimum starting from $\mathbf{x}$ is the following: i) x_1 is set to false ii) x_3 is set to false. Now, the first move does not change the number of satisfied clauses, but it changes the "amount of redundancy" (in clause-1 two literals are now satisfied, i.e., clause-1 enters S_2) and the move *is* a possible choice for a selection based on the *non-oblivious* function. The *oblivious* plateau has been eliminated and the search can continue toward the globally optimal point $\mathbf{x} = (01011)$.

3. Non-oblivious functions in basic local search

The above cited theoretical results tell us that local search with *non-oblivious* functions obtains a better worst-case result with respect to OB. Our first series of tests aims at assessing the relative *average-case* performance of the two possibilities, both in terms of the average quality of the local optima discovered and in terms of the number of iterations executed. The considered neighborhood is the 1-neighborhood, whose n members are obtained by flipping a single variable.

Our tests are dedicated to MAX-3-SAT instances. The same benchmark suite as that of [**17**] and [**25**] is considered and the generator of random 3-SAT instances was obtained from B. Selman[1]. For each couple of variables and clauses (n, m), 50 instances were randomly generated (seeds 1,2,...,50).

The mean results found by different techniques are reported in Table 1. The first five lines in the Table are derived from [**17**] that proposed the Steepest Ascent – Mildest Descent method (SAMD), a prohibition-based heuristic similar to Tabu Search [**9**], with superior performance in comparison with Johnson's approximation algorithms [**18**] (JOHN1 and JOHN2) and Simulated Annealing (SA). The line about GSAT+ refer to the GSAT-with-walk algorithm proposed in [**23, 25**]. Additional entries refer to our experiments and will be described in the following sections.

3.1. Average performance for local search (OB, NOB). Local search stopping at the first local optimum is run 10 times for each instance after using different random initial assignments, for a total of 500 tests for each (n, m) couple. Ties between moves causing the same Δf are broken randomly, with uniform probability for all candidates.

Table 1 lists the average number of unsatisfied clauses at the local optimum (with its statistical error in parenthesis), while Table 2 reports the average number of iterations applied after starting from a random assignment. The lines

[1] The code is available at an ftp archive, contact selman@research.att.com for details.

TABLE 1. MAX-3–SAT: Mean number of unsatisfied clauses (statistical error in parenthesis when available). The first five lines are from Hansen-Jaumard, GSAT+ is from Selman et. al. OB is the oblivious search, NOB the non–oblivious one, see text for details.

variables clauses	100 500	100 700	300 1500	300 2000	500 5000
LOC	15.6	26.6	43.5	73.5	249.1
JOHN1	14.9	28.2	45.3	74.1	268.8
JOHN2	13.5	26.9	44.1	76.5	257.4
SA	8.2	18.1	30.0	58.0	226.4
SAMD	5.1	14.7	15.3	39.0	182.8
GSAT+	2.9	12.9	8.1	34.9	163.6
OB	15.1 (0.3)	27.7 (0.6)	43.9 (0.9)	74.4 (0.9)	244.1 (1.8)
NOB	10.9 (0.3)	23.6 (0.3)	31.0 (0.6)	60.7 (0.9)	220.8 (1.2)
NOB&OB	8.9 (0.3)	20.8 (0.3)	25.7 (0.6)	53.1 (0.6)	202.7 (1.2)
NOB&OB & $10 \times n$ iter.	4.5 (0.3)	15.7 (0.3)	11.8 (0.3)	36.6 (0.6)	172.3 (0.9)
NOB&OB & $50 \times n$ iter.	4.2 (0.3)	15.6 (0.3)	10.8 (0.3)	36.1 (0.6)	171.8 (0.9)

variables clauses	100 200	300 600	300 800
LOC	1.9	5.9	12.2
JOHN1	2.3	4.7	10.6
JOHN2	1.4	2.7	9.0
SA	0.2	2.7	6.1
SAMD	0.3	2.4	4.3
GSAT+	0	0	0
OB	1.9 (0.3)	5.7 (0.3)	11.6 (0.3)
NOB	0.3 (0.1)	1.0 (0.1)	4.0 (0.3)
NOB&OB	0.2 (0.1)	0.6 (0.1)	2.7 (0.3)
NOB&OB & $10 \times n$ iter.	0 (0)	0 (0)	0 (0)
NOB&OB & $50 \times n$ iter.	0 (0)	0 (0)	0 (0)

denoted with OB (NOB) corresponds to the oblivious and non-oblivious search, respectively.

The first result to be noted is that the NOB local search does lead to local optima of a better average quality with respect to OB. In some cases the relative improvement is large. As an example from Table 1, for the $(300, 600)$ problems one passes from 5.7 unsatisfied clauses to 1.0. The number of iterations increases by about a factor of two but it is in all cases upper bounded by $1/2 \times n$.

Table 3 reports the measured CPU time for the different tests, including the initialization phase and input/output operations. Our software has been optimized for flexibility and not for speed: therefore the CPU times are not to be considered as "best effort" estimates. Even with this research version of the software, the CPU time needed by NOB is very small (less that 0.5 sec for the largest tasks).

TABLE 2. MAX-3–SAT: Average number of iterations to reach the local optimum. OB = oblivious, NOB = non–oblivious local search, NOB&OB = oblivious search following NOB.

variables	100	100	300	300	500
clauses	500	700	1500	2000	5000
OB	23.4 (0.6)	26.0 (0.6)	70.0 (0.9)	76 .8 (0.9)	143.8 (1.5)
NOB	42.6 (0.9)	44.2 (0.9)	131.8 (1.5)	134.1 (1.5)	230.5 (2.1)
NOB&OB	44.6 (0.9)	46.9 (0.9)	137.0 (1.5)	141.2 (1.5)	246.2 (2.1)

variables	100	300	300
clauses	200	600	800
OB	14.6 (0.3)	44.7 (0.6)	52.7 (0.6)
NOB	41.1 (0.6)	125.5 (1.2)	125.4 (1.2)
NOB&OB	41.2 (0.6)	125.9 (1.2)	126.7 (1.2)

TABLE 3. MAX–3–SAT: Average CPU time (seconds, Pentium PC at 120 MHz) for different algorithms, see text.

variables	100	100	100	300	300	300	300	500
clauses	200	500	700	600	800	1500	2000	5000
OB	0.08	0.09	0.10	0.10	0.12	0.14	0.16	0.37
NOB	0.08	0.10	0.10	0.13	0.16	0.20	0.22	0.48
NOB&OB	0.08	0.10	0.10	0.13	0.17	0.20	0.23	0.52
NOB&OB & $10 \times n$ iterations	0.13	0.19	0.20	0.51	0.57	0.64	0.72	2.07
NOB&OB & $50 \times n$ iterations	0.39	0.48	0.54	2.06	2.16	2.34	2.64	8.34

3.2. Two-phase local search (NOB&OB). Let us remind that the local-optimality criteria corresponding to f and f_{NOB} are not equivalent: in particular a local optimum for the non-oblivious function is not necessarily a local optimum for the oblivious one. The reason is intuitively clear: approximations obtained through f_{NOB} tend to be characterized by a "redundant" satisfaction of many clauses (let us remember that positive terms in the function f_{NOB} proportional to $w(S_2)$ and $w(S_3)$ favor this result). Now, it may be the case that *more* clauses can be covered by reducing the amount of redundancy.

Let us therefore consider the following two-phase local search algorithm: after a random start the *non-oblivious* function guides the search until a local optimum is encountered (NOB&OB). As soon as this happens a second phase is started where the move evaluation is based on the *oblivious* function. This phase is in turn stopped as soon as its first local optimum is found (it is therefore stopped immediately if the assignment obtained by NOB cannot be modified to increase the number of satisfied clauses).

The intuition is confirmed by the experimental evidence (see Table 1–2): the second phase produces a non-trivial reduction in the number of unsatisfied clauses at the price of a limited number of additional iterations. Clearly, each additional iteration of OB decreases the number of unsatisfied clauses by at least one.

A further reduction in the number of unsatisfied clauses can be obtained by a "plateau search" phase following NOB&OB: the search is continued for a certain number of iterations after the local optimum of OB is encountered, by using f as guiding function. In this phase the best move is always applied, even if it leads to an equal or worse f value. In particular, the results obtained with an additional phase of $10 \times n$ and $50 \times n$ iterations are listed in the last two lines of Table 1, while the CPU time is listed in Table 3.

From the above results it is apparent that the instances corresponding to some ratios of clauses to variables are satisfiable with a very large probability. In fact, so large that no unsatisfiable instance was found for (n, m) equal to $(100, 200), (300, 600), (300, 800)$. This result is clearly explained by the analysis of [**21**]. For this reason, and in order to focus onto the most relevant data for MAX-SAT, only the non-trivial cases (100,500), (100,700), (300,1500), (300,2000), and (500, 5000) are considered in the following experimental part.

4. Non-oblivious functions in complex heuristics

From the above results it is apparent that NOB obtains better approximations than OB, on the average. In addition, better approximations are obtained if OB follows NOB and, finally, still better results are obtained if the search is not stopped at the "oblivious" plateau but continued for a certain number of iterations. While the results about non-oblivious functions are novel, the fact that MAX-SAT is characterized by many local optima and flat plateaus and that exploring the vicinity of a local optimum for a certain period before restarting is appropriate has been observed by many authors [**14, 24, 7, 8**].

It is remarkable that the above combination (NOB&OB) when followed by a short number of iterations (e.g., equal to $10 \times n$), reaches results that are comparable to, and in some cases better than those obtained by Hansen and Jaumard [**17**]. In particular, NOB&OB plus $10 \times n$ iterations solves all satisfiable instances, while SAMD does not, see the columns corresponding to the (100,200), (300,600), and (300,800) tasks in Table 1. For the unsatisfiable instances a sizable performance difference is observed, particularly for large-size tasks. As an example, for (500,5000) tasks SAMD reaches 182.8 unsatisfied clauses, while NOB&OB plus $10 \times n$ iterations reaches 172.3.

Nonetheless, in spite of the above encouraging evidence, the final solution quality is not competitive with recently-proposed heuristics like "GSAT-with-walk" of Selman et al. [**25**], see the line marked GSAT+ in Table 1. The GSAT-with-walk algorithm will be summarized in the following section. Now, while both SAMD and NOB&OB require a very limited CPU time (at most some seconds on the largest tasks) GSAT-with-walk has been given ample time in [**25**] (up to 10,000 seconds on the largest instances). Clearly, the larger the allotted number of iterations t, the better the solution. In fact, given the presence of random restarts, if t tends to infinity GSAT+ will encounter the optimal solution with probability tending to one (eventually the optimal solution will be generated as starting point!).

At this point it is therefore appropriate to consider the average quality of results as a function of the elapsed iterations when more complex algorithms are

considered. In this manner the performance of different algorithms can be judged in a fair manner and the appropriate compromise between computation time and solution quality can be decided in a given application.

In a first series of tests the "relevant" MAX–3–SAT benchmark suite described above is used to test the different algorithms up to a number of iterations equal to 1,000 $\times n$.

4.1. GSAT with "random noise" strategies. Different variations of local search with randomness techniques have been proposed for SAT and MAX-SAT starting from the late eighties, for some examples see [**13**], [**26**], and the updated review of [**15**], and motivated by previous applications of "min-conflicts" heuristics in the area of Artificial Intelligence [**12**], [**20**].

A popular algorithm like GSAT [**26**] consists of multiple runs of OB local search followed by a certain number of iterations, for a total of a fixed number of iterations for each run. Different "noise" strategies to escape from attraction basins are added to GSAT in [**25**]. In particular, the "GSAT-with-walk" algorithm has been tested in [**25**] on the Hansen-Jaumard benchmark of [**17**], where a better performance with respect to SAMD is demonstrated (although requiring much longer CPU times).

```
GSAT-WITH-WALK
1       for i ← 1 to MAX-TRIES
2           x ← random truth assignment
3           for j ← 1 to MAX-FLIPS
4               if RANDOM < p then
5                   var ← any variable occurring in some unsatisfied clause
6               else
7                   var ← any variable with largest Δf
8               FLIP (var)
```

FIGURE 1. The "GSAT-with-walk" algorithm. RANDOM generates random numbers in the range $[0, 1]$

The algorithm is briefly summarized in Fig. 1. A certain number of tries (*MAX-TRIES*) is executed, where each try consists of a number of iterations (*MAX-FLIPS*). At each iteration a variable is chosen by two possible criteria and then flipped. One criterion (active with "noise" probability p) selects a variable occurring in some unsatisfied clause with uniform probability over such variables, the other one is the "standard" method based on the OB function. The best assignment is saved and reported at the end, unless all clauses are satisfied and the run is terminated before. The use of more than one try is not crucial given the better diversification properties in comparison with GSAT.

The software corresponding to the algorithm has been developed in C++ and the algorithm has been tested on the above benchmark. The data in [**25**] are not sufficient because we are interested in analyzing the approximation behavior as a function of the number of iterations. The original seeds were not available and therefore the tasks are different from those used in [**25**], although, clearly, extracted from the same distribution. The parameters used are *MAX-FLIPS*

$= 5 \times n$, $p = 0.4$ and *MAX-TRIES* corresponding to the desired total number of iterations.

The approximation behavior of GSAT-with-walk is shown in Fig. 2. Each graph corresponds to a different problem dimension. Each curve reports the results for different number of clauses by showing the average number of unsatisfied clauses (the best-so-far value) as a function of the elapsed iterations. The superiority of this algorithms with respect to simple GSAT is confirmed in our tests, that are not reported for brevity. In addition, results that are statistically comparable to those of [**25**] are reproduced in runs of up to three hours of CPU time. On the contrary, the maximum CPU time for the tests of Fig. 2 ranges from 10 sec for $(100, 500)$ tasks to 4 minutes for $(500, 5000)$ tasks.

4.2. GSAT-with-walk using NOB. Given the superior approximation results obtained by non-oblivious functions and the encouraging performance of GSAT-with-walk one is led to consider the adoption of the non-oblivious function in the given algorithm.

The results obtained when f_{NOB} is used to select variables in GSAT-with-walk are shown in Fig. 3. The approach is disappointing: in fact the obtained results are much worse that those obtained with the standard f. A similar result has been obtained after applying the same substitution to GSAT: GSAT with f_{NOB} produces much worse results with respect to standard GSAT.

Apparently the use of f_{NOB} is "misleading" if a very good approximation is searched for: the search trajectory is biased towards assignments with redundant coverings, i.e., large values of $w(S_2)$ and $w(S_3)$, but the price to be paid is that more clauses tend to remain unsatisfied. In addition, many moves that are judged as equivalent when using f obtain different scores with f_{NOB}. In other words, f_{NOB} tends to produce finer-grained rankings of the moves and therefore to reduce the need of a random breaking of ties. The implied reduction in diversification (caused by the reduction in the number of "best" moves at each iteration) could be acceptable if the search is directly focussed onto assignments with high values of f (not only of f_{NOB} !) but the obtained results are negative in this respect.

4.3. History-based search with NOB&OB: the N-RTS algorithm. Given the unsatisfactory performance obtained through the simple substitution of f with f_{NOB} it is useful to reconsider the results obtained in Sec. 3. In that case better results were obtained when f_{NOB} was used in a first phase, until a local optimum of f_{NOB} was encountered, then f was used during a second phase. In addition, the results summarized in Table 1 demonstrate that *diversification* is indeed needed after a certain number of iterations on the plateau. In fact, a very limited gain was obtained in passing from $10 \times n$ to $50 \times n$ iterations.

History-based heuristics like SAMD ("steepest ascent mildest descent") [**17**], clause–weighting [**24**], and HSAT [**8**] have been proposed to increase the amount of diversification without compromising the average performance, to continue local search schemes beyond local optimality and, at the same time, to avoid cycles in the search directory. These schemes and related ones aim at diversifying the search into uncharted territories by using the information collected from the previous phase (the "history") of the search. As an example clause–weighting where the weights are dynamically modified during problem solving can be con-

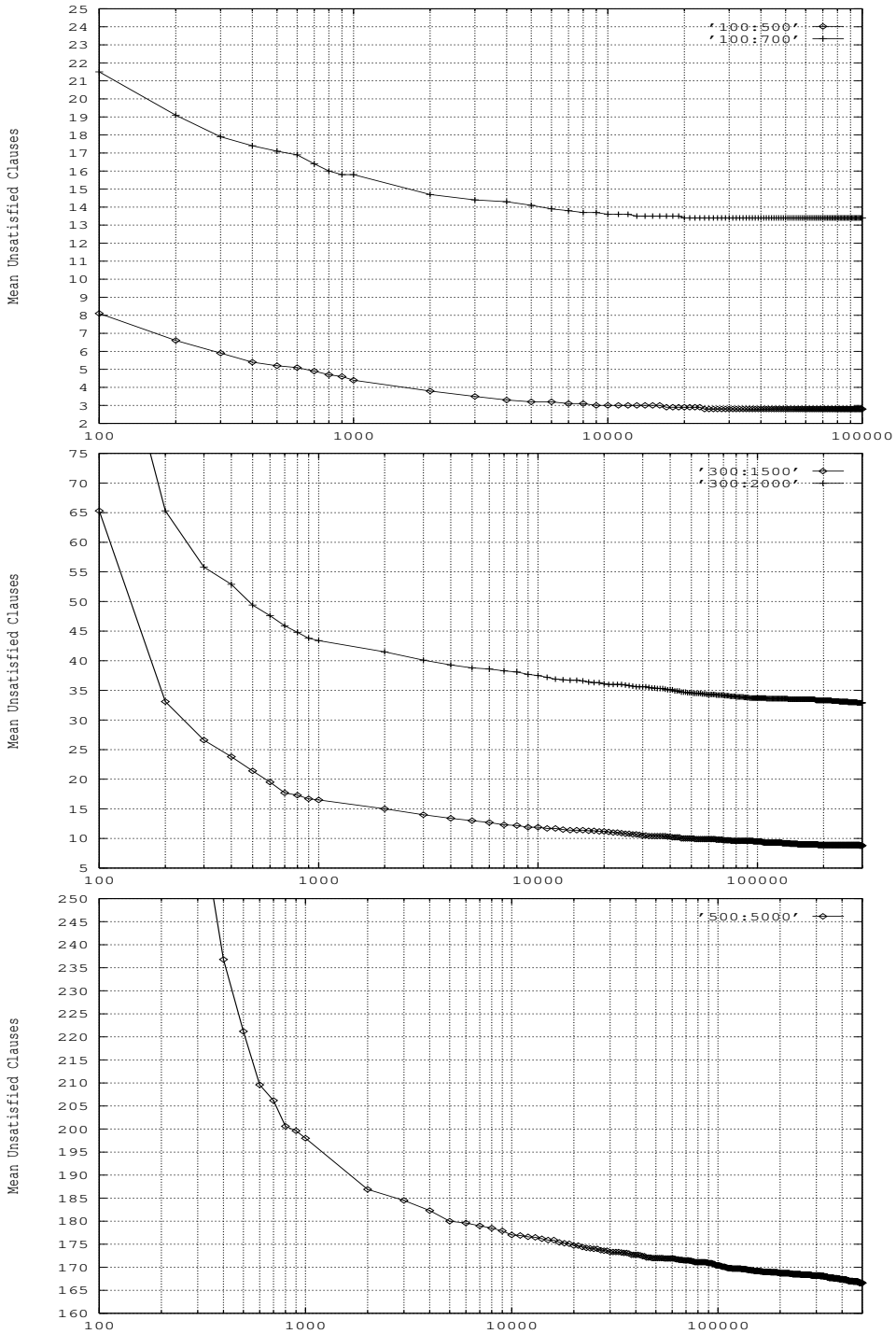

FIGURE 2. GSAT-with-walk: results up to 1,000 $\times n$ iterations.

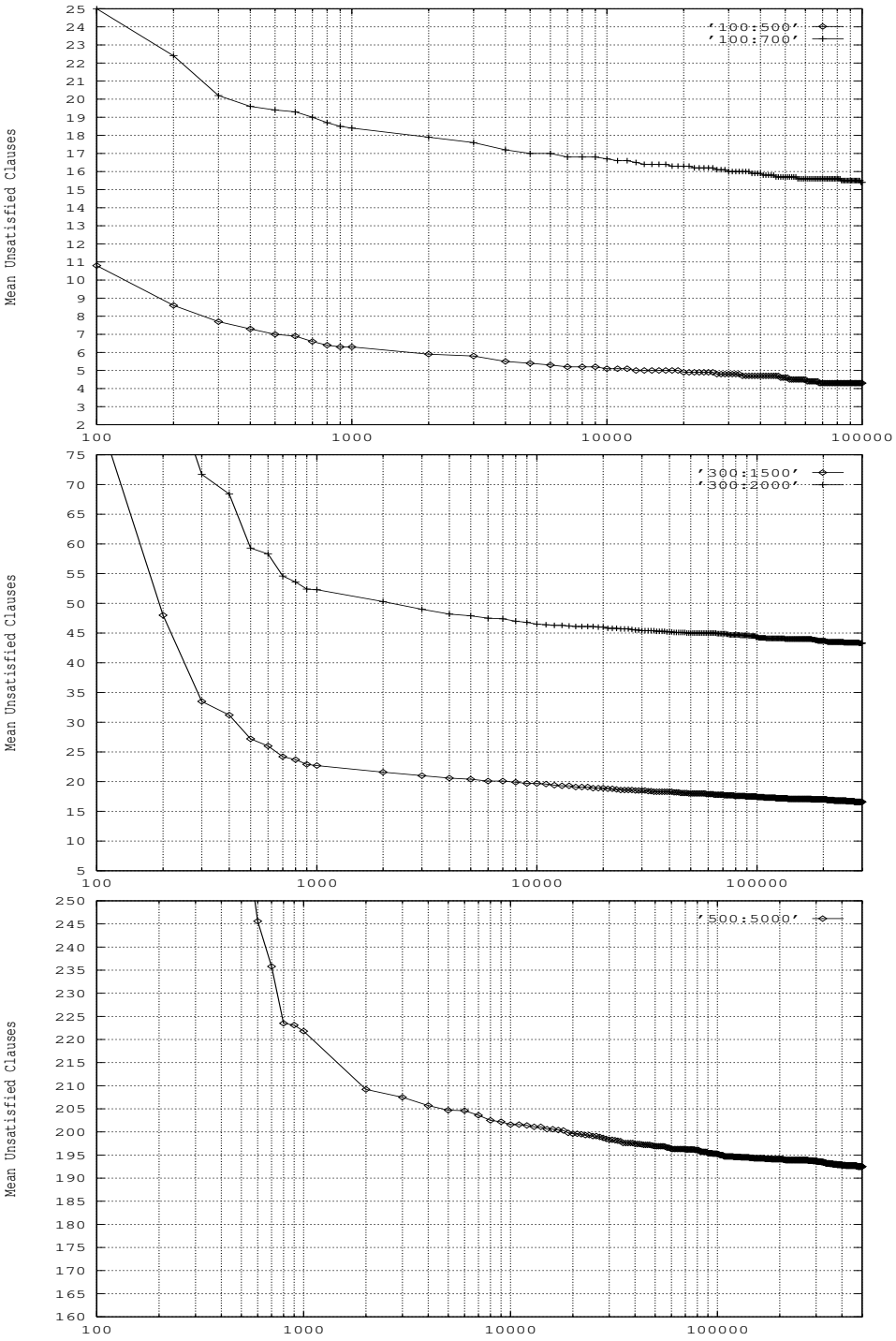

FIGURE 3. GSAT-with-walk using NOB function up to 1,000 $\times n$ iterations.

sidered as a "reactive" technique where a repulsion from a given local optimum is generated in order to induce an escape [**3**].

These methods are to be compared with simpler diversification schemes (like random restarts or simple versions of Simulated Annealing, SA for short) that do not use the previous history. As an example, the choice of the next move in the Markov chain generated by SA is influenced only by the current point $\mathbf{x}$: there is no way for the SA algorithm to determine that the search is trapped in a local attraction basin close to a locally optimal point. Therefore SA may spend huge amounts of computing time in a confined and suboptimal search region. The asymptotic results for SA are not relevant for its practical application (a simpler scheme like random search is asymptotically faster than SA, while exhaustive search terminates in finite time!).

History-based schemes, like other heuristics, can be characterized by some free parameters whose values have to be tuned for specific problems by the user or, preferably, by some automated tuning mechanism. In particular, the Reactive Tabu Search [**4**] (RTS) proposes a simple but effective mechanism to tune the *prohibition* parameter of TS. The feedback scheme is based on the repetition of points along the search trajectory: an evidence of cycling or of a "chaotic" trapping of the trajectory. In general, the Reactive Search framework [**3**] proposes to use *feedback* mechanisms based on the previous history (machine–learning) for two purposes: i) to tune the internal parameters in an adaptive way and ii) to generate an automated balance of diversification and intensification.

Preliminary tests of RTS for MAX–SAT showed that the use of repetitions as triggering mechanism tends to be fragile in the presence of large plateaus with the same f values: for large n the repetitions can be very rare even if the trajectory is confined.

Therefore we decided to base the "reaction" on a more robust measure of diversification: the Hamming distance H. After a local optimum $\mathbf{x}_L$ is reached, the corresponding assignment is stored into memory, and the Hamming distance between the current point and the stored point $\mathbf{x}_L$ is monitored during the subsequent iterations. Now, it is well known that, for large n values, most binary strings are at distance of approximately $n/2$ from a given string. In detail, the Hamming distances are distributed with a binomial distribution with the same probability of success and failure ($p = q = 1/2$), in fact the fraction of strings at distance H is equal to

$$\left(\begin{array}{c} n \\ H \end{array} \right) \times \frac{1}{2^n}$$

It is well known that the mean is $n/2$ and the variance is $n/4$. The above coefficients increase up to the mean $n/2$ and then decrease. The mean Hamming distance is therefore $n/2$ and a large fraction of the strings are clustered at nearby distances. As an example, one can use the Chernoff bound [**16**]:

$$Pr[H \leq (1-\theta)pn] \leq e^{-\theta^2 np/2}$$

the probability to find a point at a distance less than $np = n/2$ decreases in the above exponential way ($\theta \geq 0$).

This fact can be used to trigger a radical diversification if the maximum Hamming distance reached from the stored local optimum is less than $n/2$ after a suitable search period, otherwise only a negligible fraction of the possible strings, all located close to $\mathbf{x}$, will be visited. The time constant with which pure random walk, after starting from $\mathbf{x}$ converges to its asymptotic value of mean distance $H = n/2$ from $\mathbf{x}$ is proportional to n. Therefore it is reasonable to wait for a number of iterations equal to a small multiple of n before testing for a possible strong diversification. This prescription agrees with the experimental prescriptions for using GSAT, where a restart is activated every $k \times n$ iterations, k being a small integer [**7**].

```
NON-OBLIVIOUS-REACTIVE-TABU-SEARCH
1   x ← random truth assignment
2   repeat
3       repeat                                   { NOB local search }
4           var ← any variable with largest Δf_NOB
5           FLIP (var)
6       until largest Δf_NOB = 0
7       x_L ← current x
8       H_max ← 0
9       for 10 × n iterations                    { tabu search }
10          var ← any allowed variable with largest Δf
11                if none allowed, any variable with largest Δf
12          FLIP (var)
13          H_max ← max{H_max, H(x_L, x)}
14      if H_max < n/2 then                      { diversification test }
15          randomly pick n/2 variables and change their values
16  until solution is acceptable or maximum number of iterations reached
```

FIGURE 4. The N-RTS algorithm.

The proposed N-RTS algorithm (Non-oblivious Reactive Tabu Search) is illustrated in Fig. 4. It consists of phases of Tabu Search starting at the local optima produced by NOB local search. After $10 \times n$ iterations of TS the *diversification test* is executed: if the diversification is not sufficient, $n/2$ variables are randomly chosen (without replacement) and complemented. In this way a distance $H = n/2$ is enforced. The effect is similar to that obtained with a random restart: in this case an average of $n/2$ variables will have a different truth value. After the check and the (possible) strong diversification, the search continues with OB local search. The prohibition parameter of TS is equal to $n/10$. Additional implementation details for TS include the use of an aspiration criterion (the prohibition of a move is relaxed if a value of f larger than the best-so-far can be obtained by using the move) and a tie-breaking scheme: if more allowed moves would cause the same Δf, the ones satisfying more new clauses (not satisfied in the current $\mathbf{x}$) are preferred. This criterion increases the diversification without changing the f values obtained. If more that one move survives this second criterion, a random choice is executed.

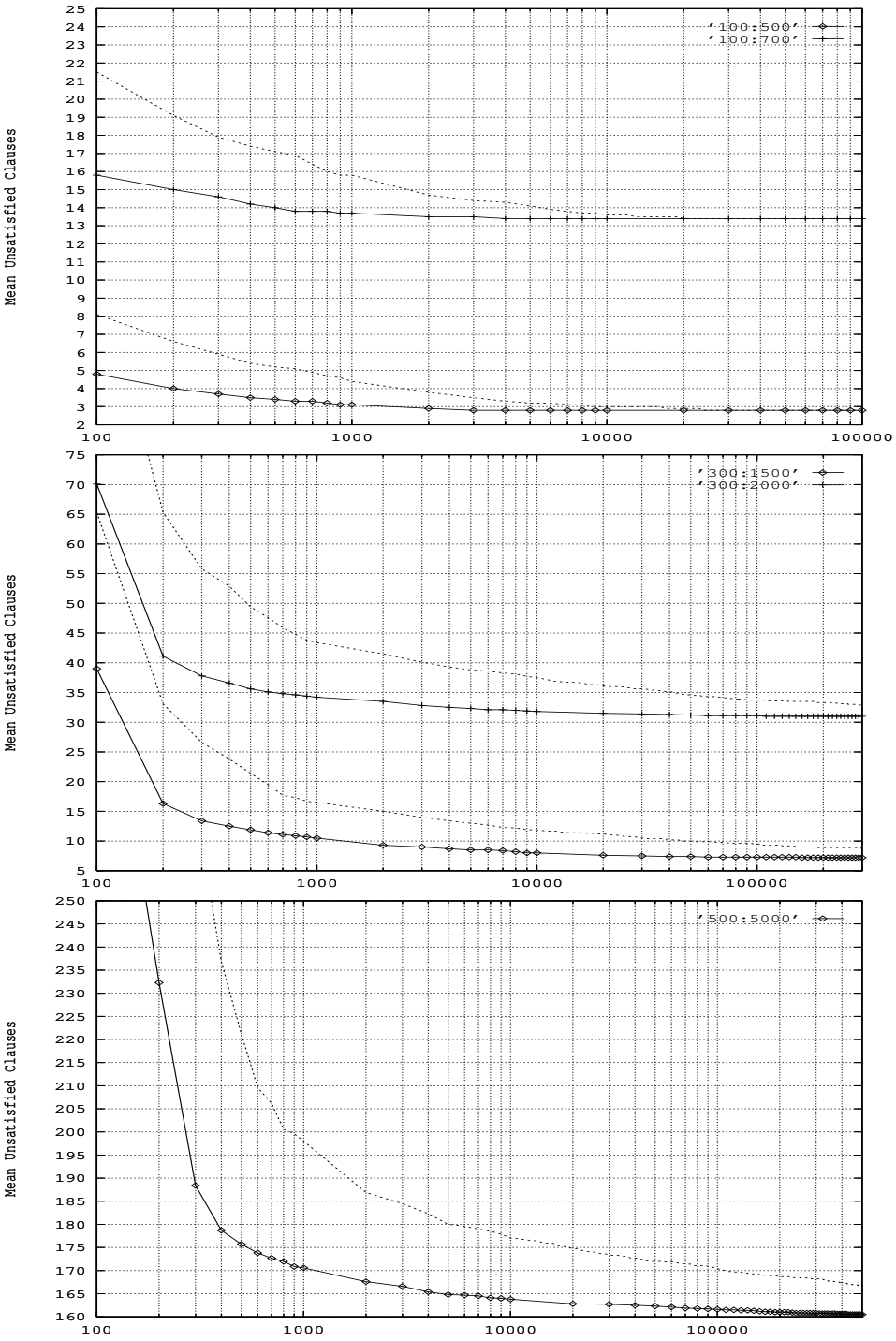

FIGURE 5. The N-RTS algorithm (Non-Oblivious Reactive Tabu Search) up to 1,000 $\times n$. The results for GSAT-with-walk are reported with dashed lines for a better comparison.

The obtained results are illustrated in Fig. 5, where the results of Fig 2 are reported with dashed lines for a better comparison. It can be observed that the approximation results of N-RTS are much better than those of GSAT-with-walk if the results are compared at the same number of elapsed iterations. The performance advantage starts in the very early phase, given the better quality of the first local optimum produced when NOB local search is followed by tabu search with the standard oblivious function. The initial advantage is maintained in the following phases of the search, in particular N-RTS obtains results close to the final ones after a limited number of iterations. The performance advantage is limited for the small-size tasks: both GSAT-with-walk and N-RTS reach the same value at the end of 100,000 iterations for $n = 100$, and it increases for the larger tasks as it is evidenced by the growing performance gap at the end of the run when one considers tasks with $n = 300$ and, particularly, $n = 500$. For a more quantitative comparison of the final results, the average performance of the considered heuristics at the end of different numbers of allotted iterations is collected in Table 4.

TABLE 4. Average performance at the end of different numbers of iterations ($1K = 1,000$).

variables	100	100	300	300	500
clauses	500	700	1500	2000	5000
GSAT+ at $1K \times n$	2.8 (0.6)	13.4 (0.6)	8.8 (1.2)	32.9 (1.8)	166.6 (3.0)
GSAT+ at $5K \times n$	2.8 (0.6)	13.4 (0.6)	7.8 (1.1)	32.5 (1.5)	165.5 (2.8)
GSAT+ & NOB at $1K \times n$	4.2 (0.7)	15.4 (0.6)	16.6 (1.2)	43.3 (1.8)	192.5 (2.9)
N-RTS at $1K \times n$	2.8 (0.5)	13.4 (0.6)	7.2 (0.7)	31.0 (0.9)	160.5 (1.8)
N-RTS at $5K \times n$	2.8 (0.5)	13.4 (0.6)	7.1 (0.7)	31.0 (0.9)	160.0 (1.2)

5. Conclusion

Motivated by recent theoretical results about non-oblivious functions for local search [**19, 1**], the average performance of NOB local search was investigated on a benchmark of random MAX–3–SAT problems. The first experimental result obtained is that NOB reaches local optima with a number of unsatisfied clauses lower than the number obtained through standard (OB) local search, on the average.

Then a novel combination (NOB&OB) is investigated: standard local search takes off from the local optimum found by NOB and reaches its own local optimum. When NOB&OB is followed by a limited number of iterations, approximations better than those reached by SAMD [**17**] are obtained.

Because the results are not competitive with more recent heuristics, the integration of non-oblivious functions in more complex algorithms is studied through a statistical analysis of the approximation behavior as a function of the elapsed iterations. Here a negative result is obtained about the simple substitution of the function guiding the search in the neighborhood: if f_{NOB} is used instead of the standard f in GSAT or GSAT-with-walk, the performance worsens.

Finally, a novel algorithm in which phases of NOB local search are alternated with phases of Tabu Search is proposed (N-RTS). In this algorithm a "reactive"

activation of a stronger form of diversification (beyond that provided by the basic TS mechanism) is triggered by monitoring the Hamming distance from a stored local optimum. In this case the approximation obtained in the initial phase of the search is much better than that provided by GSAT-with-walk and the performance gap is maintained up to large number of iterations, especially for larger dimensional tasks.

Acknowledgment

We wish to thank P. Alimonti for discussions about non-oblivious functions and for suggesting the example, P. Crescenzi and L. Trevisan for discussion about the latest approximability results, and B. Selman and collaborators for making available their software. This research was partially supported by CNR Contract "Strutture Informative e Teoria degli Algoritmi", MURST 40% Project "Efficienza di Algoritmi e Progetto di Strutture Informative", ESPRIT-LTR Project ALCOM-IT(20244), Progetto Speciale 95-96 Università di Trento

References

1. P. Alimonti. New local search approximation techniques for maximum generalized satisfiability problems. In *Proc. Second Italian Conf. on Algorithms and Complexity*, pages 40–53, 1994.
2. G. Ausiello, P. Crescenzi, and M. Protasi. Approximate solution of np optimization problems. *Theoretical Computer Science*, 150:1–55, 1995.
3. R. Battiti. Reactive search: Toward self-tuning heuristics. In V. J. Rayward-Smith, editor, *Modern Heuristic Search Methods*. John Wiley and Sons Ltd, 1996, to appear.
4. R. Battiti and G. Tecchiolli. The reactive tabu search. *ORSA Journal on Computing*, 6(2):126–140, 1994.
5. M. Bellare, O. Goldreich, and M. Sudan. Free bits, PCP and non-approximability — towards tight results. Technical Report TR 95–24, Electronic Colloquium on Computational Complexity, December 1995.
6. U. Feige and M.X. Goemans. Approximating the value of two proper proof systems, with applications to MAX-2-SAT and MAX-DICUT. In *Proceeding of the Third Israel Symposium on Theory of Computing and Systems*, pages 182–189, 1995.
7. I.P. Gent and T. Walsh. An empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, 1:47–59, 1993.
8. I.P. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. unpublished report, 1993.
9. F. Glover. Tabu search - part i. *ORSA Journal on Computing*, 1(3):190–260, 1989.
10. M.X. Goemans and D.P. Williamson. A new $\frac{3}{4}$ approximation algorithm for the maximum satisfiability problem. *SIAM Journal on Discrete Mathematics*, 7(4):656–666, 1994.
11. M.X. Goemans and D.P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of Association for Computing Machinery*, 42(6):1115–1145, 1995.
12. J. Gu. *Parallel Algorithms and Architectures for very fast AI Search*. PhD thesis, University of Utah, 1989.
13. J. Gu. Efficient local search for very large-scale satisfiability problem. *ACM SIGART Bulletin*, 3(1):8–12, 1992.
14. J. Gu. Local search for the satisfiability (SAT) problem. *IEEE Transactions on Systems, Man, and Cybernetics*, 23(4):1108–1129, 1993. A number of typographical errors in the paper were corrected in IEEE Trans. on Systems, Man, and Cybernetics, 24(4):709, 1994.
15. J. Gu, P.W. Purdom, J. Franco, and B.W. Wah. Algorithms for the Satisfiability (SAT) Problem: a survey. In D.-Z. Du, J. Gu, and P.M. Pardalos, editors, *Satisfiability Prob-*

lem: Theory and Applications, DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1996.
16. T. Hagerup and C. Rüb. A guided tour of chernoff bounds. *Information Processing Letters*, 33:305–308, 1989/90.
17. P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44:279–303, 1990.
18. D.S. Johnson. Approximation algorithms for combinatorial problems. *Journal of Computer and System Sciences*, 9:256–278, 1974.
19. S. Khanna, R. Motwani, M. Sudan, and U. Vazirani. On syntactic versus computational views of approximability. In *Proc. 35th Ann. IEEE Symp. on Foundations of Computer Science*, pages 819–836, 1994.
20. S. Minton, M. D. Johnston, A. B. Philips, and P. Laird. Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method. In *Proceedings AAAI-90*, pages 17–24, 1990.
21. D. Mitchell, B. Selman, and H. Levesque. Hard and easy distributions of sat problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 459–465, San Jose, Ca, July 1992.
22. P. Pardalos. Continuous approaches to discrete optimization problems. In G. Di Pillo and F. Giannessi, editors, *Nonlinear optimization and applications*. Plenum Publishing, 1995.
23. B. Selman and H. Kautz. Noise strategies for improving local search. In *Proceedings of AAAI-94*, 1994.
24. B. Selman and H.A. Kautz. An empirical study of greedy local search for satisfiability testing. In *Proceedings of the eleventh national Conference on Artificial Intelligence (AAAI-93)*, Washington, D. C., 1993.
25. B. Selman, H.A. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In M. Trick and D. S. Johson, editors, *Proceedings of the Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability*, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, 1995.
26. B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 440–446, San Jose, Ca, July 1992.
27. G. Sorkin, M. Sudan, L. Trevisan, and D.P. Williamson. Gadgets, approximation and linear programming. Technical report, 1996.
28. M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms*, 17:475–502, 1994.

(R. Battiti) Dipartimento di Matematica, Università di Trento, Via Sommarive 14, 38050 Povo (Trento) Italy
E-mail address, R. Battiti: battiti@science.unitn.it

(M. Protasi) Dipartimento di Matematica, Università di Roma "Tor Vergata" Via della ricerca scientifica, 00133 Roma Italy
E-mail address, M. Protasi: protasi@mat.utovrm.it