Reactive Search, a history-based heuristic for MAX–SAT *

Roberto Battiti[1] and Marco Protasi[2]

[1]Dipartimento di Matematica, Univ. of Trento
Via Sommarive 14, 38050 Povo (Trento) - Italy
battiti@science.unitn.it

[2]Dipartimento di Matematica, Univ. di Roma "Tor Vergata"
Via della ricerca scientifica, 00133 Roma - Italy
protasi@mat.utovrm.it

Workshop on the Satisfiability Problem, Siena, Italy, May, 1996.
J. Franco, G. Gallo, H. K. Büning, E. Speckenmeyer, and C. Spera (eds.)

Abstract

The Reactive Search (RS) method proposes the integration of a simple *history-based feedback scheme* into local search for the on-line determination of free parameters. In this paper a new RS algorithm is proposed for the approximated solution of the Maximum Satisfiability problem: a component based on local search with temporary prohibitions is complemented with a reactive scheme that determines ("learns") the appropriate value of the prohibition parameter by monitoring the Hamming distance along the search trajectory (algorithm H-RTS). In addition, the *non-oblivious* functions recently introduced in the framework of approximation algorithms are used to discover a better local optimum in the initial part of the search.

The algorithm is developed in two phases. First the bias-diversification properties of individual candidate components are analyzed by extensive empirical evaluation, then a reactive scheme is added to the winning component, based on Tabu Search.

The final tests on a benchmark of random MAX–3–SAT and MAX–4–SAT problems demonstrate the superiority of H-RTS with respect to alternative heuristics.

1 Introduction

In the Maximum Satisfiability (MAX–SAT) problem one looks for an assignment of truth values that maximizes the number of satisfied clauses of a boolean formula in conjunctive normal form. In our work, n is the number of variables and m the number of clauses, so that the formula has the following form:

$$\bigwedge_{1\le c\le m} \Big(\bigvee_{1\le j\le l_c} p_{cj} \Big)$$

*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 where l_c is the number of literals in clause c and p_{cj} is a literal, i.e., a propositional variable x_k or its negation $\overline{x_k}$, for $1 \leq k \leq n$. MAX–SAT is of considerable interest not only from the theoretical side but also from the practical one. On one hand, the decision version SAT was the first example of an NP–complete problem; moreover MAX–SAT and related variants play an important role in the characterization of different approximation classes like APX and PTAS [4]. 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.

The present work was motivated by the current state of the MAX–SAT algorithmics, where a considerable gap exists between worst-case theoretical results and successful applications of heuristics. From the theoretical point of view, 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. On the other hand, negative theoretical results for MAX–SAT have been proven by showing that this problem cannot be approximated beyond a certain level, see [7] for a summary and references to papers that present a complete treatment of the subject. From the practical point of view there has recently been a renaissance in the study of effective heuristic algorithms based on local search and random diversification techniques. Starting from the late eighties both discrete and continuous-based greedy algorithms have been proposed, see for example [10, 15, 28] and the reviews in [16] and [27]. MAX–SAT is therefore a paradigmatic problem for the "algorithmic engineering" and scientific testing and tuning effort advocated for example by [2], [5], and [22]. The core of this paper consists of the design of a new heuristic algorithm guided by a series of focussed experiments, where individual factors are isolated and studied through a statistical analysis. Because of this design process the algorithm achieves average results that are significantly better than the existing heuristics, to the best of our knowledge.

This paper is organized in the following way. First a summary of the Reactive Search framework (Sec. 2) and of existing heuristic approaches that are relevant for the work is presented (Sec. 3). Then the investigation is subdivided in two phases: in the first one the candidate individual components based on local search are evaluated by studying their diversification-bias trade-off (Sec. 4). In the second one the best component selected in the first phase is complemented with a reactive scheme and the novel algorithm H-RTS is designed (Sec. 5). Finally, extensive experimental results on a benchmark of MAX–3–SAT and MAX–4–SAT tasks demonstrate the superiority of the H-RTS algorithm with respect to alternative heuristics (Sec. 6).

2 Reactive Search: reinforcement learning for heuristics

State-of-the-art heuristics tend to be characterized by the composition of basic building blocks in complex algorithms, which present many possible structural choices and free parameters. Examples are competitive Simulated Annealing (SA) schemes [18], Genetic Algorithms (GA) [21], Tabu Search (TS) [12]. Now, the pitfalls of *competitive testing* where the participants select options and tune parameters in a non–scientific way (i.e., poorly documented and hardly reproducible) are being recognized in the heuristics research community, see for example [22], where a "more scientific approach of controlled experimentation" is advocated.

It is often the case that the user is a crucial *learning* component of an heuristic algorithm, whose eventual success on a problem should be credited more to the human smartness than to the algorithm intrinsic potentiality. The long–term goal of the Reactive Search framework is that of completely *eliminating the user intervention* in the tuning process, while maintaining the internal flexibility needed to cover in an effective way a wide selection of problems. The implied fact is that the *developer* must endow the heuristic algorithm with an *internal learning* *loop* in order to present to the users (and to the research community) a self-consistent and autonomous algorithm that can be judged in isolation, independently from the user ability.

The analogy with *machine learning* is apparent: in both cases a system is developed for a desired functionality through an automated training process. As an example in the area of sub-symbolic machine learning, hypotheses can be learned from a set of example instances [32], “reactive agents” can be trained though *reinforcement learning* [24]. In the area of empirical algorithm desing the use of machine learning is advocated for example in [13].

The Reactive Search (RS) method proposes the integration of simple *history-based feedback schemes* in local search for the on-line determination of free parameters. Reinforcement learning acts through an internal trial-and-error mechanism while the algorithm runs on a specific instance and “monitors” its past behavior. A particular realization of the RS framework is the Reactive Tabu Search (RTS) algorithm, where a simple feedback scheme determines the value of the prohibition parameter of Tabu Search (TS), so that a balance of exploration versus exploitation is obtained that is appropriate for the local characteristics of the task [9]. Let us now briefly define the main concepts and the notation related to Local Search, Tabu Search, and Reactive Tabu Search.

2.1 Local Search and Tabu Search

Let $\mathcal{X}$ be the discrete search space: $\mathcal{X} = \{0,1\}^n$, and let $f : \mathcal{X} \longrightarrow R$ be the function to be maximized. In addition, let $X^{(t)} \in \mathcal{X}$ be the current configuration along the *search trajectory* at iteration t, and $N(X^{(t)})$ the neighborhood of point $X^{(t)}$, obtained by applying a set of basic moves μ_i $(1 \leq i \leq n)$, where μ_i complements the i-th bit x_i of the string: $\mu_i\,(x_1, x_2, ..., x_i, ..., x_n) = (x_1, x_2, ..., 1-x_i, ..., x_n)$. Clearly, these moves are idempotent ($\mu_i^{-1} = \mu_i$).

$$N(X^{(t)}) = \{X \in \mathcal{X} \ \text{ such that } \ X = \mu_i\, X^{(t)}, i = 1, ...n\}$$

Local search (LS) starts from a random initial configuration $X^{(0)} \in \mathcal{X}$ and generates a search trajectory as follows:

$$\begin{aligned} Y &= \textsc{Best-Neighbor}\ (\ N(X^{(t)})\ ) && (1) \\ X^{(t+1)} &= \begin{cases} Y & \textbf{if} \quad f(Y) > f(X^{(t)}) \\ X^{(t)} & \textbf{if} \quad f(Y) \leq f(X^{(t)}) \end{cases} && (2) \end{aligned}$$

where Best-Neighbor selects $Y \in N(X^{(t)})$ with the best f value and ties are broken randomly. Y in turn becomes the new current configuration if f improves. Clearly, local search stops as soon as the first local optimum point is encountered, when no improving moves are available, see eqn. 2. Let us define as LS$^+(max)$ a modification of LS where max iterations are executed and the candidate move obtained by Best-Neighbor is always accepted even if the f value remains equal or worsens.

Tabu Search (TS) is a *history-based* heuristic proposed by F. Glover [12] and, independently, by Hansen and Jaumard, that used the term SAMD (“steepest ascent mildest descent”) and applied it to the MAX–SAT problem in [20]. The main mechanism by which the history influences the search in TS is that, at a given iteration, some neighbors are *prohibited*, only a non-empty subset $N_A(X^{(t)}) \subset N(X^{(t)})$ of them is *allowed*. The general way of generating the search trajectory that we consider is given by:

$$\begin{aligned} N_A(X^{(t)}) &= \textsc{Allow}(N(X^{(t)}), X^{(0)}, ..., X^{(t)}) && (3) \\ X^{(t+1)} &= \textsc{Best-Neighbor}\ (\ N_A(X^{(t)})\ ) && (4) \end{aligned}$$

The set-valued function ALLOW selects a non-empty subset of $N(X^{(t)})$ in a manner that depends on the entire previous history of the search $X^{(0)}, ..., X^{(t)}$. Let us note that worsening moves *can* be produced by eqn. 4, as it must be in order to exit local optima.

2.2 Reactive Tabu Search

The ALLOW function can be specified by introducing a *prohibition parameter* T (also called *list size*) that determines how long a move will remain prohibited after its execution. The Fixed-TS algorithm is obtained by fixing T throughout the search [12]. A neighbor is allowed if and only if it is obtained from the current point by applying a move that has not been used during the last T iterations. In detail, if $\textsc{LastUsed}(\mu)$ is the last usage time of move μ ($\textsc{LastUsed}(\mu) = -\infty$ at the beginning):

$$N_A(X^{(t)}) = \{X = \mu\, X^{(t)} \text{ such that } \textsc{LastUsed}(\mu) < (t - T)\} \tag{5}$$

The Reactive Tabu Search algorithm [9], RTS for short, defines simple rules to determine the prohibition parameter by reacting to the repetition of previously-visited configurations. One has a repetition if $X^{(t+R)} = X^{(t)}$, for $R \geq 1$. The prohibition period T depends on the iteration t (therefore the notation is $T^{(t)}$), and the discrete dynamical system that generates the search trajectory comprises an additional evolution equation for $T^{(t)}$, that is specified through the function REACT, see eqn. 6 below. The dynamical system becomes:

$$\begin{aligned} T^{(t)} &= \textsc{React}(T^{(t-1)}, X^{(0)}, ..., X^{(t)}) && (6)\\ N_A(X^{(t)}) &= \{X = \mu\, X^{(t)} \text{ such that } \textsc{LastUsed}(\mu) < (t - T^{(t)})\} && (7)\\ X^{(t+1)} &= \textsc{Best-Neighbor}\ (N_A(X^{(t)})) && (8) \end{aligned}$$

While the reader is referred to [9] for the details, the design principles of RTS are that $T^{(t)}$ (in the range $1 \leq T^{(t)} \leq n-2$) increases when repetitions happen, and decreases when repetitions disappear for a sufficiently long search period. For convenience, let us introduce a "fractional prohibition" T_f, such that the prohibition is obtained by setting $T = \lfloor T_f\ n \rfloor$. T_f ranges between zero and one, with bounds inherited from those on T.

RTS has been applied to various problems with competitive performance with respect to alternative heuristics like Fixed-TS, Simulated Annealing, Neural Networks, and Genetic Algorithms, see the review in [6]. Recently, a variation of RTS reached the best heuristic results on a large benchmark of maximum-clique tasks [8].

3 Existing heuristics for MAX–SAT

To make this paper reasonably self-contained, let us now summarize some heuristics for MAX–SAT that are relevant for the following discussion. Clearly, although some of the cited algorithm are widely used and competitive, the list is not exhaustive.

3.1 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 [15], [31], and the updated review of [16]. These techniques were in part motivated by previous applications of "min-conflicts" heuristics in the area of Artificial Intelligence [14], [25].

A popular algorithm like GSAT [31] consists of multiple runs of LS^+ local search, each one consisting of a number of iterations that is typically proportional to the problem dimension n.

Different "noise" strategies to escape from attraction basins are added to GSAT in [30]. In particular, the GSAT-with-walk algorithm has been tested in [30] on the Hansen-Jaumard benchmark of [20], where a better performance with respect to SAMD is demonstrated, although requiring much longer CPU times.

```
GSAT-WITH-WALK
1   for $i \leftarrow 1$ to $MAX\text{-}TRIES$
2       $X \leftarrow$ random truth assignment
3       for $j \leftarrow 1$ to $MAX\text{-}FLIPS$
4           if RANDOM $< p$ then
5               $var \leftarrow$ any variable occurring in some unsatisfied clause
6           else
7               $var \leftarrow$ any variable with largest $\Delta f$
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 by the function Flip , i.e., x_i becomes equal to $(1 - x_i)$. 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 function f given by the number of satisfied clauses. For a generic move μ, the quantity $\Delta_\mu f$ (or Δf for short) is defined as $f(\mu\ X^{(t)}) - f(X^{(t)})$. The straightforward book-keeping part of the algorithm is not shown. In particular, the best assignment found during all trials is saved and reported at the end of the run. In addition, the run is terminated immediately if an assignment is found that satisfies all clauses.

3.2 History-based heuristics

Different history-based heuristics have been proposed to continue local search schemes beyond local optimality. These schemes aim at intensifying the search in promising regions and at diversifying the search into uncharted territories by using the information collected from the previous phase (the *history*) of the search. The *history* at iteration t is formally defined as the set of ordered couples (X, s) such that $0 \leq s \leq t$ and $X = X^{(s)}$.

In addition to the already cited SAMD ("steepest ascent mildest descent") [20] heuristic that uses the temporary prohibitions of recently executed moves, let us mention two variations of GSAT that make use of the previous history.

HSAT [11] introduces a tie-breaking rule into GSAT: if more moves produce the same (best) Δf, the preferred move is the one that has not been applied for the longest span. HSAT can be seen as a "soft" version of Tabu Search: while TS prohibits recently-applied moves, HSAT discourages recent moves if the same Δf can be obtained with moves that have been "inactive" for a longer time. It is remarkable to see how this innocent variation of GSAT can increase its performance on some SAT benchmark tasks [11].

Clause–weighting has been proposed in [28] in order to increase the effectiveness of GSAT for problems characterized by strong asymmetries. In this algorithm a positive weight is associated to each clause to determine how often the clause should be counted when determining which variable to flip. The weights are dynamically modified during problem solving and the qualitative effect is that of "filling in" local optima while the search proceeds. Clause–

weighting can be considered as a "reactive" technique where a repulsion from a given local optimum is generated in order to induce an escape from a given attraction basin.

History-based heuristics are to be contrasted with simpler diversification schemes (like GSAT or simple versions of Simulated Annealing) 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 $X^{(t)}$: there is no mechanism for the SA algorithm to determine that the search is trapped in a local attraction basin close to a locally optimal point. Unfortunately, the asymptotic results for SA are not very relevant for its practical application [1].

3.3 Non-oblivious local search

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 [3] and in [23].

Let us consider the classical local search algorithm LS for MAX–SAT, here redefined as *oblivious* local search (LS-OB). Clearly, the feasible solution found by LS-OB 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 neighbor at each iteration. Local optima of the standard objective function f are not necessarily local optima of the different objective function. In this event, the second function causes an *escape* from a given local optimum. Interestingly enough, suitable *non-oblivious* functions f_{NOB} improve the performance of LS if one considers both the worst-case performance ratio and, as it has been shown in [7], the actual average results obtained on benchmark instances.

Let us introduce the notation and mention a theoretical result for MAX–2–SAT. Given an assignment 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 cardinality of S_i. In addition, a d-neighborhood of a given truth assignment is defined as the set of all assignment where the value of at most d variables is changed. The theoretically-derived non–oblivious function for MAX–2–SAT is:

$$f_{NOB}(X) = \frac{3}{2} w(S_1) + 2w(S_2)$$

Theorems 7-8 of [23] state that the performance ratio for any oblivious local search algorithm with a d-neighborhood for MAX–2–SAT is 2/3 for any $d = o(n)$, while non-oblivious local search with an 1-neighborhood achieves a performance ratio 3/4. Therefore LS-NOB improves considerably the performance ratio even if the search is restricted to a much smaller neighborhood. With a suitable generalization the above result can be extended: LS-NOB achieves a performance ratio $1 - \frac{1}{2^k}$ for MAX–k–SAT. The oblivious function for MAX–k–SAT is of the form:

$$f_{NOB}(X) = \sum_{i=1}^{k} c_i w(S_i)$$

and the above given performance ratio is obtained if the quantities $\Delta_i = c_{i+1} - c_i$ satisfy:

$$\Delta_i = \frac{1}{(k-i+1)\begin{pmatrix} k \\ i-1 \end{pmatrix}} \left[\sum_{j=0}^{k-i} \begin{pmatrix} k \\ j \end{pmatrix} \right]$$

Because the positive factors c_i that multiply $w(S_i)$ in the function f_{NOB} are strictly increasing with i, the approximations obtained through f_{NOB} tend to be characterized by a "redundant" satisfaction of many clauses. Better approximations, at the price of a limited number of additional iterations, can be obtained by a two-phase local search algorithm (NOB&OB): after a random start f_{NOB} guides the search until a local optimum is encountered. As soon as this happens a second phase of LS is started where the move evaluation is based on f. 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 LS^+, with f as guiding function.

3.4 Benchmark problems

Here we describe the benchmark set that will be used for the tests of the current paper, and summarize the results obtained by the above cited algorithms.

Our tests are dedicated to MAX-3-SAT and MAX-4-SAT instances. The same benchmark suite as that of [20] and [30] is considered and the generator of random instances was obtained from B. Selman[1]. In addition, larger tasks than those used in the cited benchmark have been generated to provide an additional suite of tests. These tasks have 1,000 variables and 10,000 clauses.

For each couple of variables and clauses $n : m$, 50 instances are randomly generated (seeds 1,2,...,50) and the different algorithms are run 10 times for each instance, after using different random seeds, and therefore also different random initial assignments. The total number of tests is therefore 500 for each $n : m$ couple. Unless specified in a different way, the mean results are averages of the 500 runs. If $\widehat{r}$ is a measured average quantity for a task obtained by the 10 runs with different random seeds, the default standard deviation is that of $\widehat{r}$ with respect to task variation.

variables	100	100	100	300	300	300	300	500
clauses	200	500	700	600	800	1500	2000	5000
LOC	1.9	15.6	26.6	5.9	12.2	43.5	73.5	249.1
JOHN1	2.3	14.9	28.2	4.7	10.6	45.3	74.1	268.8
JOHN2	1.4	13.5	26.9	2.7	9.0	44.1	76.5	257.4
SA	0.2	8.2	18.1	2.7	6.1	30.0	58.0	226.4
SAMD	0.3	5.1	14.7	2.4	4.3	15.3	39.0	182.8
GSAT+	0	2.9	12.9	0	0	8.1	34.9	163.6
OB	1.9 (0.3)	15.1 (0.3)	27.7 (0.6)	5.7 (0.3)	11.6 (0.3)	43.9 (0.9)	74.4 (0.9)	244.1 (1.8)
NOB	0.3 (0.1)	10.9 (0.3)	23.6 (0.3)	1.0 (0.1)	4.0 (0.3)	31.0 (0.6)	60.7 (0.9)	220.8 (1.2)
NOB&OB	0.2 (0.1)	8.9 (0.3)	20.8 (0.3)	0.6 (0.1)	2.7 (0.3)	25.7 (0.6)	53.1 (0.6)	202.7 (1.2)
NOB&OB & $10 \times n$ iter.	0 (0)	4.5 (0.3)	15.7 (0.3)	0 (0)	0 (0)	11.8 (0.3)	36.6 (0.6)	172.3 (0.9)

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.

Table 1 summarizes a previous comparison executed in [7] for MAX-3-SAT tasks. The average number of unsatisfied clauses at the local optimum is listed with its statistical error in parenthesis. The first five lines in the Table are derived from [20] that proposed the Steepest Ascent – Mildest Descent method (SAMD), with a better performance in comparison with Johnson's approximation algorithms [17] (JOHN1 and JOHN2) and Simulated Annealing (SA). The line about $GSAT^+$ refer to the GSAT-with-walk algorithm proposed in [29, 30]. Additional entries refer to our experiments in [7]. The lines denoted with OB (NOB) corresponds to the

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

oblivious and non-oblivious search, respectively. The main result is that the NOB local search does lead to local optima of a better average quality with respect to OB [7]. The mean number of iterations increases by about a factor of two but it is in all cases upper bounded by $n/2$, and the CPU time needed by NOB is very small (less that 0.5 sec for the largest tasks). The results obtained with an additional phase of 10 n iterations are listed in the last line of Table 1.

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 MAX-3-SAT instance was found for $n:m$ equal to 100:200, 300:600, 300:800. This result is clearly explained by the analysis of [26]. For this reason, and in order to focus onto the most relevant data for MAX-3-SAT, only the non-trivial cases 100:500, 100:700, 300:1500, 300:2000, 500:5000, and 1,000:10,000 are considered in the following experimental part.

For MAX-4-SAT, the instances considered are 100:700, 300:1500, 300:2500, 300:3000, and 1000:10000. Apart from the largest 1000:10000 instances, the MAX-4-SAT benchmark coincides with that used by [20] and [30].

4 Selecting components based on diversification and bias

Basic algorithms like LS-OB or LS-NOB reach local optima in a short number of iterations, but the average f values obtained can be improved in a significant way by using more complex heuristics obtained by starting from basic building blocks and, for example, by continuing the search on a given *plateau* (LS^+), by restarting LS^+ from different initial points (GSAT), by adding stochasticity (GSAT-with-walk) or by adding diversification methods to avoid cycles in the search (TS).

Now, the typical process to develop complex heuristics is based on a trial-and-error loop where alternatives are tested until acceptable results are obtained. While this process cannot be entirely automated and, in fact, is part of the research activity in the heuristics area, it is worth considering the option of guiding the choice of alternatives by simple "metrics" of the individual components. Let us focus onto local-search based heuristics: it is well known that the basic compromise to be reached is that between *diversification* and *bias*. Given the obvious fact that only a negligible fraction of the admissible points can be visited for a non-trivial task, the search trajectory $X^{(t)}$ should be generated to visit preferentially points with large f values (*bias*) and to avoid the confinement of the search in a limited and localized portion of the search space (*diversification*). The two requirements are conflicting: as an extreme example, random search is optimal for diversification but not for bias.

The investigation follows this scheme:

- After selecting the metric (diversification is measured with the Hamming distance and bias with mean f values visited), the diversification of simple *random walk* is analyzed to provide a basic system against which more complex components are evaluated (Sec. 4.1)

- The diversification-bias metrics (D-B plots) of different basic components are investigated and a conjecture is formulated that the best components for a given problem are those on the Pareto-optimal frontier in the diversification-bias (D-B) plane (Sec. 4.2).

- The conjecture is validated by a competitive analysis of the components on a benchmark suite (Sec. 4.3).

The purpose of this analysis is that of explaining in a quantitative way the success (or lack of it) of the individual components.

4.1 Diversification: a comparison with Random Walk

Diversification can be associated with different metrics. In this paper we adopt the *Hamming distance* as a measure of the distance between points along the search trajectory. The Hamming distance $H(X,Y)$ between two binary strings X and $Y$is given by the number of bits that are different, and it is a natural choice in the absence of more specific information.

Let us now consider the diversification properties of Random Walk (RW). Random Walk generates a Markov chain by selecting at each iteration a random move, with uniform probability:

$$X^{(t+1)} = \mu_{r(t)} X^{(t+1)} \quad \text{where} \quad r(t) = \text{Random } \{1, n\}$$

Without loss of generality, let us assume that the search starts from the zero string: $X^{(0)} = (0, 0, ..., 0)$. In this case the Hamming distance at iteration t is:

$$H(X^{(t)}, X^{(0)}) = \sum_{i=1,n} x_i^{(t)}$$

and therefore the expected value $\widehat{H}^{(t)} = \widehat{H}(X^{(t)}, X^{(0)})$ at time t is:

$$\widehat{H}^{(t)} = \sum_{i=1}^{n} \widehat{x_i}^{(t)} = n\, \widehat{x}^{(t)} \tag{9}$$

To make explicit the dependence of $\widehat{H}^{(t)}$ on the parameter p the notation $\widehat{H}_p^{(t)}$ will be used. The equation for $\widehat{x}^{(t)}$, the probability that a bit is equal to 1 at iteration t, is immediately derived by considering the two possible events that i) the bit remains equal to 1 and ii) the bit is set to 1. In detail, after defining as $p = 1/n$ the probability that a given bit is changed at iteration t, one obtains:

$$\widehat{x}^{(t+1)} = \widehat{x}^{(t)}\,(1-p) \;+\; (1-\widehat{x}^{(t)})\, p = \widehat{x}^{(t)} \;+\; p\,(1-2\widehat{x}^{(t)}) \tag{10}$$

It is straightforward to derive the following theorem:

Theorem 4.1 *If $n > 2$ (and therefore $0 < p < \frac{1}{2}$) the difference equation 10 for the evolution of the probability $\widehat{x}^{(t)}$ that a bit is equal to one at iteration t, with initial value $\widehat{x}^{(0)} = 0$, is solved for t integer, $t \geq 0$ by:*

$$\widehat{x}^{(t)} \;=\; p \sum_{h=0}^{t-1} (1-2p)^h \;=\; \frac{1-(1-2p)^t}{2} \tag{11}$$

Proof. The proof is by induction. First, eqn. 11 satisfies the initial condition at $t = 0$. In addition, if one assumes that eqn. 11 coincides with the solution at iteration t, after using eqn. 10, it will coincide with the solution at iteration $t+1$. $\square$

Some conclusions are immediate: after considering the incremental ratio

$$\pi^{(t+1)} \equiv \widehat{x}^{(t+1)} - \widehat{x}^{(t)} = p\,(1-2p)^t$$

one has that $\pi^{(1)} = p$, $\pi^{(t)} > 0$, and that $\pi^{(t)}$ decreases toward zero for t tending to infinity. In particular, this means that the $\widehat{x}^{(t)}$ function is always increasing and concave.

For the purpose of the current paper one is interested in studying the behavior of eqn. 10 when the number of variables n is large. In this case it is useful to associate to the discrete eqn. 10 the differential equation:

$$y' = p\,(1-2y) \tag{12}$$

whose solution for the initial value $y(0) = 0$ is:

$$y(t) = \frac{1}{2}\,(1 - e^{-2pt}) \tag{13}$$

For large n the above solution of the associated differential equation provides a very good approximation to the solution of the original discrete equation. The point-wise convergence is immediate to check: given a point $t \geq 0$, $\lim_{p\to 0} y(t) = \widehat{x}^{(t)}$. In fact, one obtains the stronger result:

Theorem 4.2 *Let $\widehat{H}_p^{(t)}$ be the average Hamming distance at iteration $t \geq 0$ derived from eqn. 9, with $p = (1/n) \in (0, 1/2)$. In addition, let $\alpha = 2p$, and, for a generic function $z(t)$ with domain on the integer numbers $t \geq 0$, let the norm be defined as $\|z\|_\infty = \max_{t\geq 0} |z(t)|$. One obtains:*

$$\|\widehat{H}_p^{(t)} - \frac{n}{2}(1 - \; e^{-\frac{2t}{n}})\|_\infty \leq n\,\frac{e^{\alpha-1}}{2}\frac{e^{-\alpha} - (1-\alpha)}{\alpha} \tag{14}$$

and therefore:

$$\lim_{p\to 0} \|\widehat{H}_p^{(t)} - \frac{n}{2}(1 - \; e^{-\frac{2t}{n}})\|_\infty = \lim_{\alpha\to 0^+} n\,\frac{e^{\alpha-1}}{2}\frac{e^{-\alpha} - (1-\alpha)}{\alpha} = 0 \tag{15}$$

The proof is given in Appendix I.

The qualitative behavior of the average Hamming distance can be derived from Theorem 4.2. At the beginning $\widehat{H}^{(t)}$ remains close to its initial value of zero and, as a first order approximation, it has a linear growth in time:

$$\widehat{H}^{(t)} \approx t \tag{16}$$

For large t the expected Hamming distance $\widehat{H}^{(t)}$ tends to its asymptotic value of $n/2$ in an exponential way, with a "time constant" $\tau = n/2$

Let us now compare the evolution of the mean Hamming distance for different algorithms. The analysis is started as soon as the first local optimum is encountered by LS, when diversification becomes crucial. LS$^+$, and Fixed-TS with fractional prohibition T_f equal to 0.1, denoted as TS(0.1), are then run for 10 n additional iterations. Fig. 2 shows the average Hamming distance as a function of the additional iterations after reaching the LS optimum on the 500:5000 benchmark tasks. The evolution of Random Walk is also reported for reference (on this graph the difference between the discrete evolution of eqn. 10 and the continuous one of eqn. 12 is not visible).

Some conclusions can be derived: although the initial linear growth is similar to that of RW, the Hamming distance does not reach the asymptotic value $n/2$ and a remarkable difference is present for the two algorithms. In detail, the average Hamming distance reached by LS$^+$ after $10n$ iterations is 0.27, while that of TS(0.1) is 0.40. The limited standard deviation is related to the fact that the actual distance (and not only the average) often does not reach the $n/2$ value. The fact that the asymptotic value is not reached even for large iteration numbers implies that all visited strings tend to lie in a confined region of the search space, with bounded Hamming distance from the starting point.

In fact, 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 ($p = q = 1/2$): the fraction of strings at distance H is equal to

$$\begin{pmatrix} n \\ H \end{pmatrix} \times \frac{1}{2^n} \tag{17}$$

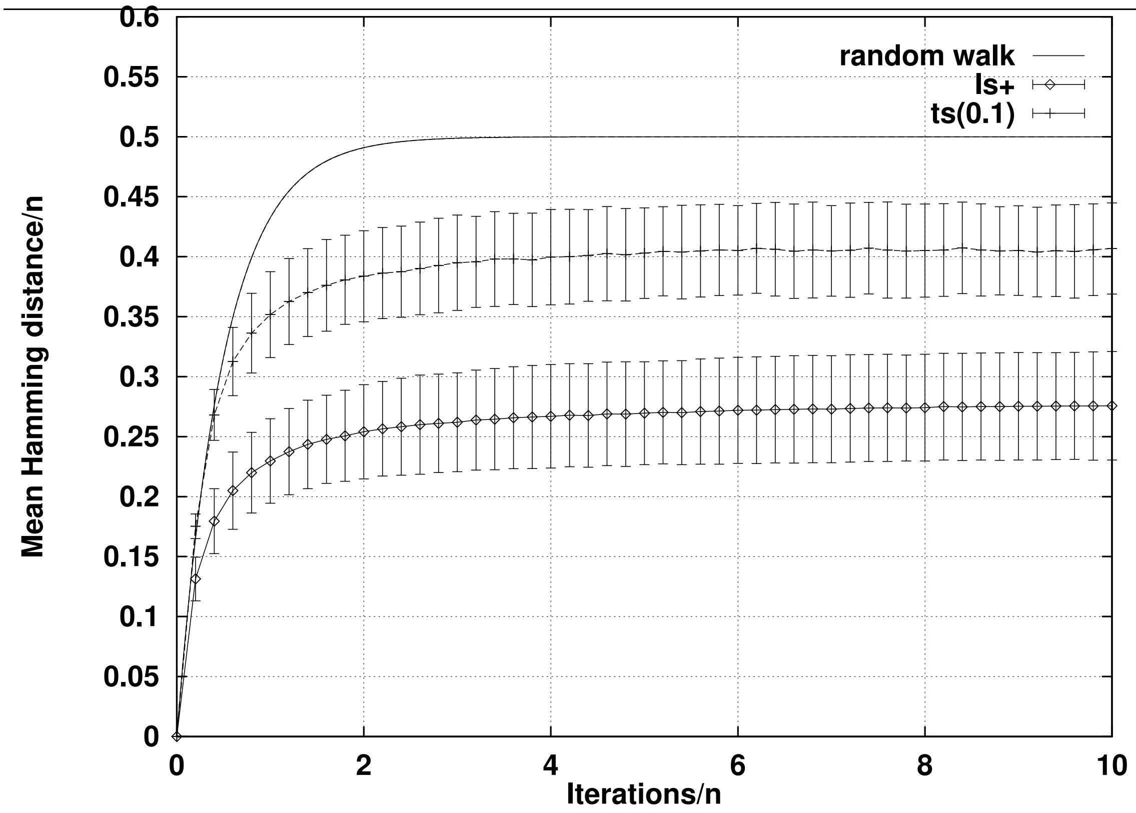


Figure 2: Average Hamming distance reached by RW, LS$^+$ and TS(0.1) from the first local optimum of LS, with standard deviation (500:5000 MAX-3-SAT tasks).

It is well known that the mean is $n/2$ and the standard deviation is $\sigma = \sqrt{n}/2$. The above coefficients increase up to the mean $n/2$ and then decrease. Because the ratio σ/n tends to zero for n tending to infinity, for large n values most strings are clustered in a narrow peak at Hamming distance $H = n/2$. As an example, one can use the Chernoff bound [19]:

$$Pr[H \leq (1-\theta)pn] \leq e^{-\theta^2 np/2} \tag{18}$$

the probability to find a point at a distance less than $np = n/2$ decreases in the above exponential way ($\theta \geq 0$). The distribution of Hamming distances for $n = 500$ is shown in Fig 3.

Clearly, if better local optima are located in a cluster that is not reached by the trajectory, they will never be found. In other words, a robust algorithm demands that some stronger diversification action is executed. For example, an option is to activate a restart after a number of iterations that is a small multiple of the time constant $n/2$. For the GSAT algorithm, the approximate scaling of *MAX-FLIPS* with n has been verified experimentally by [11, 31].

Now, depending on the structure of the different tasks, this form of radical diversification can be unnecessary. Nonetheless, its cost in terms of additional iterations is very limited provided that it is executed at intervals much larger than n, and the increased robustness obtained by inserting the periodic restart more than compensates for the (possible) limited waste of computing cycles. For example, if a new local optimum is reached in less than $n/2$ iterations, and the restart is executed after 10 n iterations, its additional iteration cost is less than 5%, on the average.

As already mentioned, *diversification* is only one of the relevant metrics, the other being the *bias*. The compromise between the two competing requirements is the subject of the next subsection.

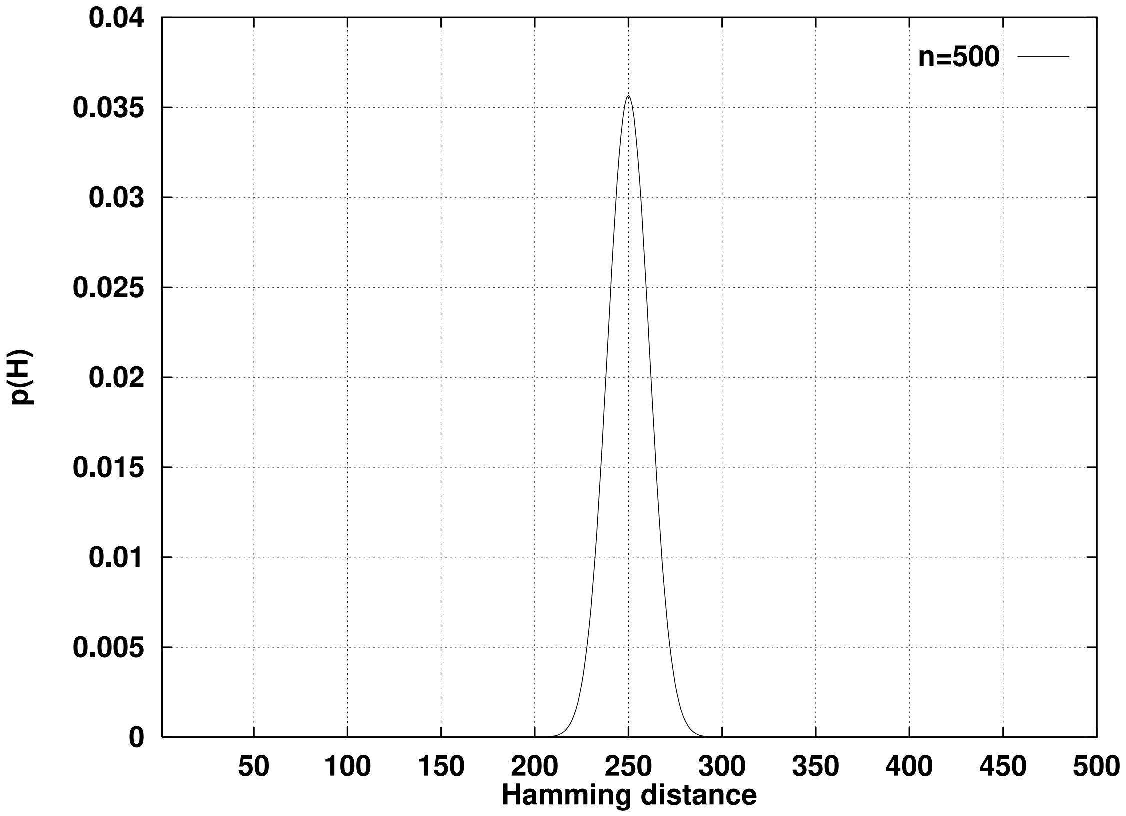


Figure 3: Probability of different Hamming distances for $n = 500$.

4.2 The diversification-bias compromise (D-B plots)

Let us now analyze how individual components behave when bias (mean function values) and diversification (mean Hamming distance) are considered. For convenience, let us define as u the number of unsatisfied clauses for a given truth assignment (clearly, $u = n - f$).

When a local search component is started, new configurations are obtained at each iteration until the first local optimum is encountered, because the number of satisfied clauses increases by at least one. During this phase additional diversification schemes are not necessary and potentially dangerous, because they could lead the trajectory astray, away from the local optimum.

The compromise between bias and diversification becomes critical after the first local optimum is encountered. In fact, if the local optimum is strict, the application of a move will worsen the f value, and an additional move could be selected to bring the trajectory back to the starting local optimum. Even if the local optimum is not strict (*plateau* regions are typical for MAX–SAT) there is no guarantee that a simple local search component will not produce a localized trajectory, for example such that its maximum Hamming distance from the first local optimum encountered is bounded by a value much less than n.

The mean bias and diversification depend on the value of the internal parameters of the different components. In order to isolate the effect of these parameters, a series of tests is executed where all other experimental conditions are unchanged and only a single parameter is changed. In particular, all tests of the different components on the 500:5000 benchmark suite use the same sequence of random numbers. In addition, all runs proceed as follows: as soon as the first local optimum is encountered by LS, it is stored and the selected component is then run for additional $4n$ iterations (from the results of Sec. 4.1 a small multiple of n is sufficient to reach an empirical asymptotic value of $\widehat{H}$). The final Hamming distance H

from the stored local optimum and the final value of the number of unsatisfied clauses u are collected. As usual, these values are then averaged by considering 50 different tasks and 10 runs with different random number seeds for each task.

Different diversification-bias (D-B) plots are shown in Fig. 4 (top). Each point gives the D-B coordinates $(\widehat{H_n}, \widehat{u})$, i.e., average Hamming distance divided by n and average number of unsatisfied clauses, for a specific parameter setting in the different algorithms. The Hamming distance is normalized with respect to the problem dimension n, i.e., $\widehat{H_n} \equiv \widehat{H}/n$. Three basic algorithms are considered: GSAT-with-walk, Fixed-TS, and HSAT. For each of these, two options about the guiding functions are studied: one adopts the "standard" oblivious function, the other the non-oblivious f_{NOB} introduced in Sec. 3.3. Finally, for GSAT-with-walk one can change the probability parameter p, while for Fixed-TS one can change the fractional prohibition T_f: parametric curves as a function of a single parameter are therefore obtained. For both p and T_f, the values tested are 0 , 0.01 , 0.02 , 0.03, 0.04 , 0.05 , 0.1 , 0.15 , 0.2 , 0.25 , 0.3 , 0.35 , 0.4, 0.45 , and 0.5. In addition, the values 0.55 and 0.6 are tested only for the parameter p. For visibility, only a subset of the parameter values are written next to the curves.

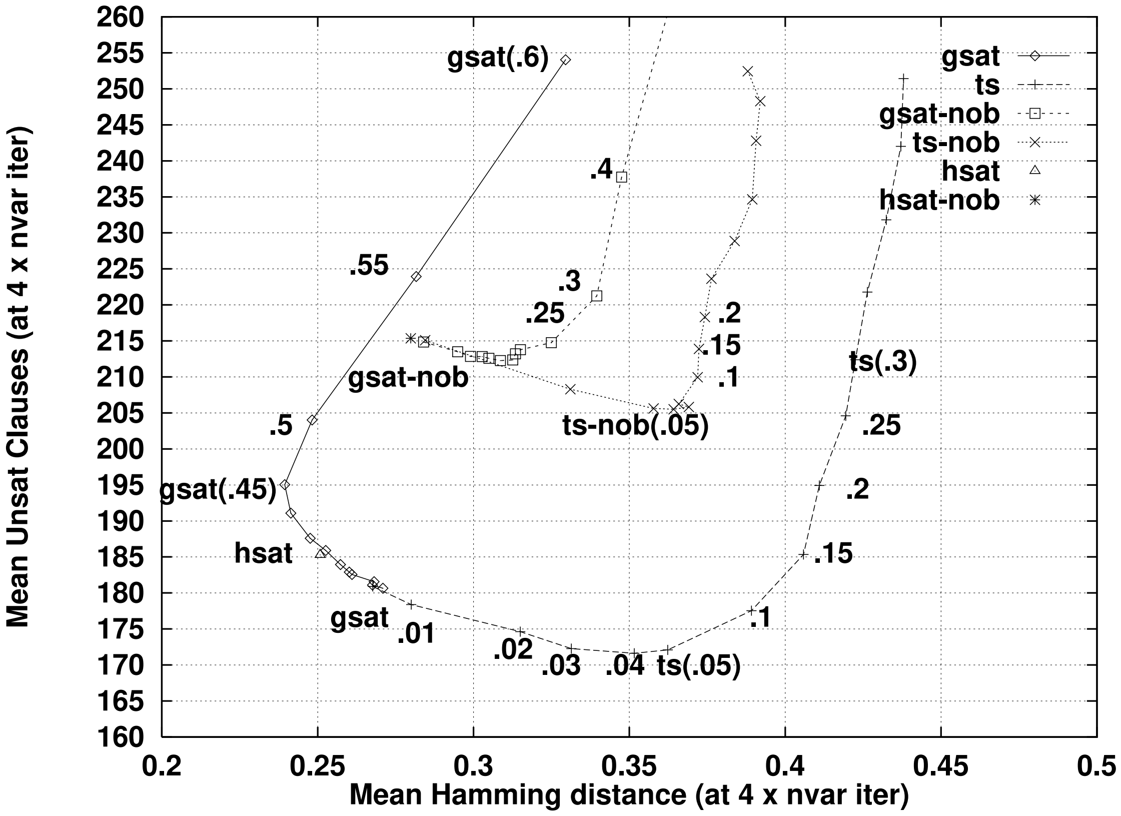
Let us draw some conclusions. First, let us note that GSAT, Fixed-TS(0.0), and GSAT-with-walk(0.0) coincide: no prohibitions are present in TS and no stochastic choice is present in GSAT-with-walk.

By considering the parametric curve for GSAT-with-walk(p) (label "gsat" in Fig 4, top) one observes a gradual increase of $\widehat{u}$ for increasing p, while the mean Hamming distance reached at first decreases (down to$\widehat{H_n} = 0.239$ for $p = 0.45$) and then increases up to $\widehat{H_n} = 0.329$ for $p = 0.6$. The initial decrease in $\widehat{H_n}$ is unexpected because it contradicts the intuitive argument that more stochasticity implies more diversification. The reason for the above result is that there are two sources of "randomness" in the GSAT-with-walk algorithm (see Fig. 1), one deriving from the random choice among variables in unsatisfied clauses, active with probability p, the other one deriving from the random breaking of ties if more variables achieve the largest Δf.

Because the first randomness source increases with p, the decrease in $\widehat{H_n}$ could be explained if the second source decreases. This conjecture has been tested and the results are collected in Fig. 5, where the average number of ties (number of moves achieving the largest Δf) is plotted as a function of p, with statistical error bars. To avoid transient effects during the initial part of the run the number of ties has been measured at iteration $4n$, the largest iteration considered in the D-B studies. A radical decrease in the mean number of ties can be observed for the oblivious function: one passes from 10.7 ties for GSAT to 4.8 ties for GSAT-with-walk(0.45), to 3.0 ties for GSAT-with-walk(0.6). In pictorial terms, the larger amount of stochasticity implied by a larger p keeps the trajectory on a rough terrain at higher values of f, where flat portions tend to be rare. *Vice versa*, almost no tie is present when the non-oblivious function is used.

The parametric curve for Fixed-TS(T_f) (label "ts" in the figure) shows an opposite behavior at the beginning: the average u values decrease *and* the diversification increases. The lowest values of $\widehat{u}$ are reached for $T_f = .04$, then the diversification gradually increases but $\widehat{u}$ worsens in a rapid way, after the "corner" point with $T_f = 0.1$, where $\widehat{u} = 177.55$ and $\widehat{H_n} = 0.389$.

The parametric curves obtained with the non-oblivious function start at the GSAT-NOB point with coordinates $\widehat{H_n} = 0.283$, $\widehat{u} = 214.85$ (i.e., the diversification is a little larger than that of GSAT but the function values are much worse). A curve qualitatively similar to that of Fixed-TS(T_f) is obtained for Fixed-TS-NOB(T_f), although with worse function values and less diversification, while the initial behavior of the curve for GSAT-with-walk-NOB(p) is in contrast with that of GSAT-with-walk(p): now the initial derivative is negative. The result



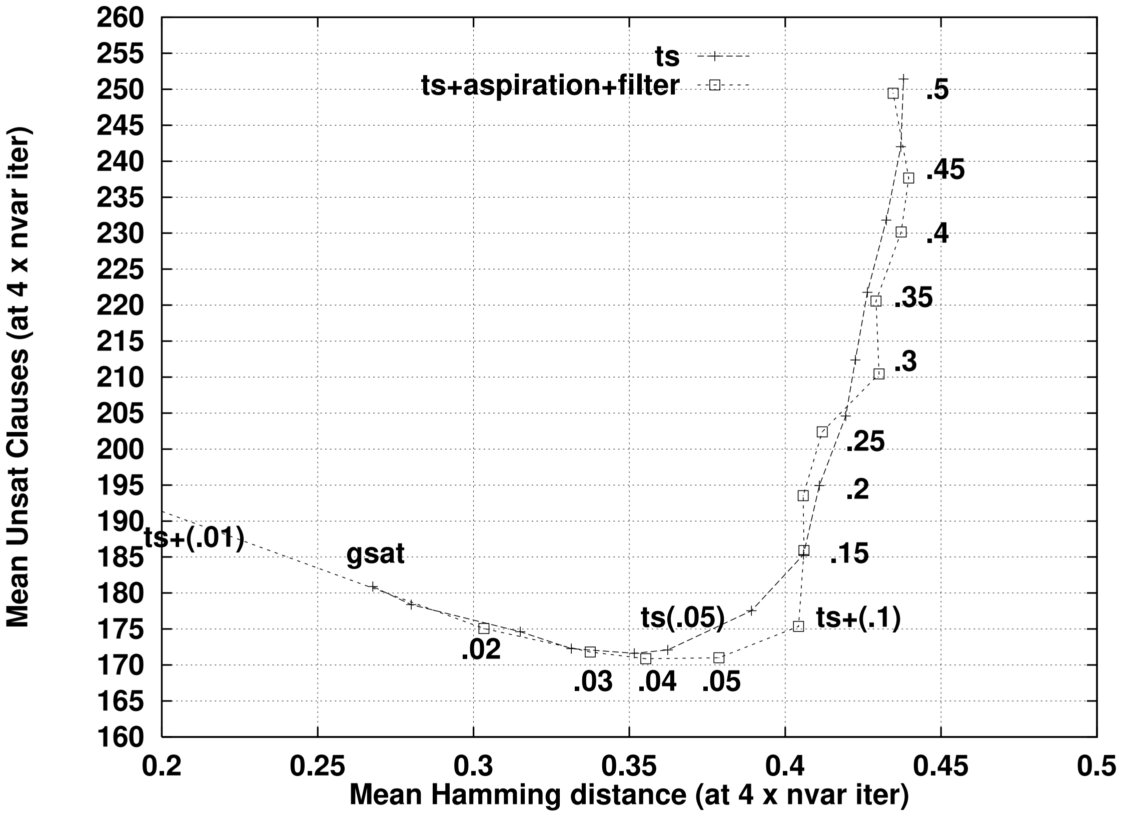


Figure 4: Diversification-bias plane. Mean number of unsatisfied clauses after 4 n iterations versus mean Hamming distance. Each run starts from GSAT local optimum. See text for explanation.

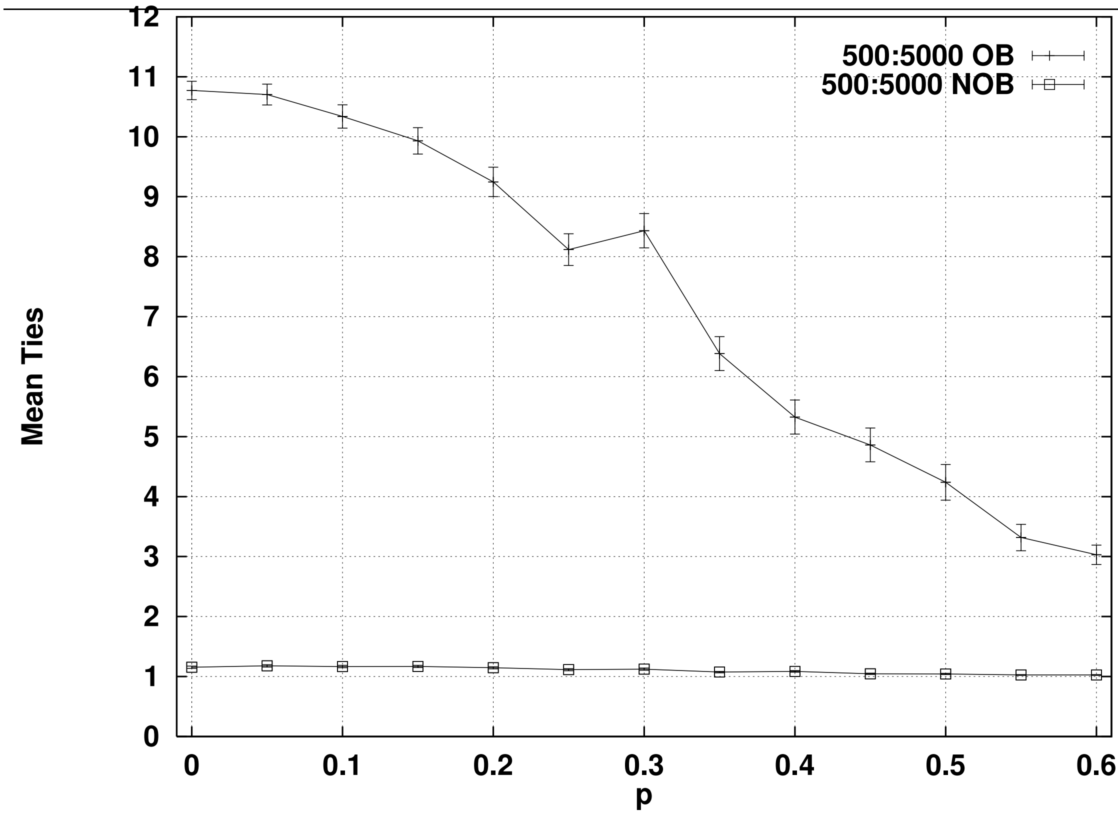


Figure 5: Mean number of ties as a function of the probability p in GSAT-with-walk(p), for the oblivious (OB) and non-oblivious (NOB) f. Values at 4 n iterations.

is related to the fact that the random breaking of ties acts almost always for GSAT, see Fig. 5, but very rarely for GSAT-NOB, where the mean number of ties is 1.15. The decrease of the number of ties with p is not very relevant: in fact it becomes 1.02 for GSAT-with-walk-NOB(0.6). Therefore the second randomness source related to p is dominating and it immediately increases the diversification.

Finally, the two points obtained with HSAT and HSAT-NOB are close to the curves for GSAT-with-walk: in particular HSAT-NOB has D-B coordinates similar to those of GSAT-NOB, while HSAT has coordinates similar to those of GSAT-with-walk(0.3).

Let us now formulate a conjecture about the relevance of the diversification-bias metric for the application of a component as a heuristic technique. By definition, component A is Pareto-optimal with respect to a set of components if it maximizes the quantity $\widehat{f} + \lambda \widehat{H_n}$ for a $\lambda > 0$. In other words, if A is Pareto-optimal, the other components in the set *cannot* possess both a higher diversification, and a better bias. In the graph one plots the number of *un*satisfied clauses versus the Hamming distance, therefore the optimal components are in the lower–right part of the set of $(\widehat{H_n}, \widehat{u})$ points.

Conjecture

If local-search based components are used in heuristic algorithms for optimization, the components producing the best f values during a run, on the average, are those corresponding to the Pareto-optimal points in the diversification-bias plane.

The conjecture produces some "falsifiable" predictions that can be tested experimentally.

In particular, a partial ordering of the different components is introduced: component A is better than component B if $\widehat{H_{nA}} \geq \widehat{H_{nB}}$ *and* $\widehat{u}_A \leq \widehat{u}_B$. The ordering is partial because no conclusions can be reached if, for example, A has better diversification but worse bias when compared with B.

From the above results, the Pareto-optimal frontier is dominated by Fixed-TS(T_f) for values of $T_f \geq 0.04$. In particular:

1. For every value of T_f, Fixed-TS-NOB(T_f) is inferior to some Fixed-TS(T'_f) point
2. For every value of T_f, GSAT-with-walk (oblivious or non-oblivious) is inferior to some Fixed-TS(T'_f) point
3. HSAT (oblivious or non-oblivious) is inferior to some Fixed-TS(T'_f) point.

Let us now consider Fixed-TS(T_f), the algorithm on the optimal frontier of Fig 4 (top), and study the effect of a simple *aspiration* criterion [12], and a tie-breaking rule.

The aspiration criterion cancels the prohibition of a move μ dictated by eqn. 5 if and only if an f value better than the best value found during the run is obtained after applying μ. The motivation for allowing μ is that the danger of repeating previously visited configurations does not exist in this case.

The tie-breaking rule is activated if more allowed moves cause the same best Δf. In this case one selects among the winning moves those that break the largest number of clauses satisfied by the current configuration ("break" means "change their value from true to false"). If more than a single move is selected in this phase, a random choice is then executed to determine the move to be applied. The particular tie-breaking rule was suggested by the implementation of WALKSAT [29] in the C code distributed by B. Selman and collaborators.

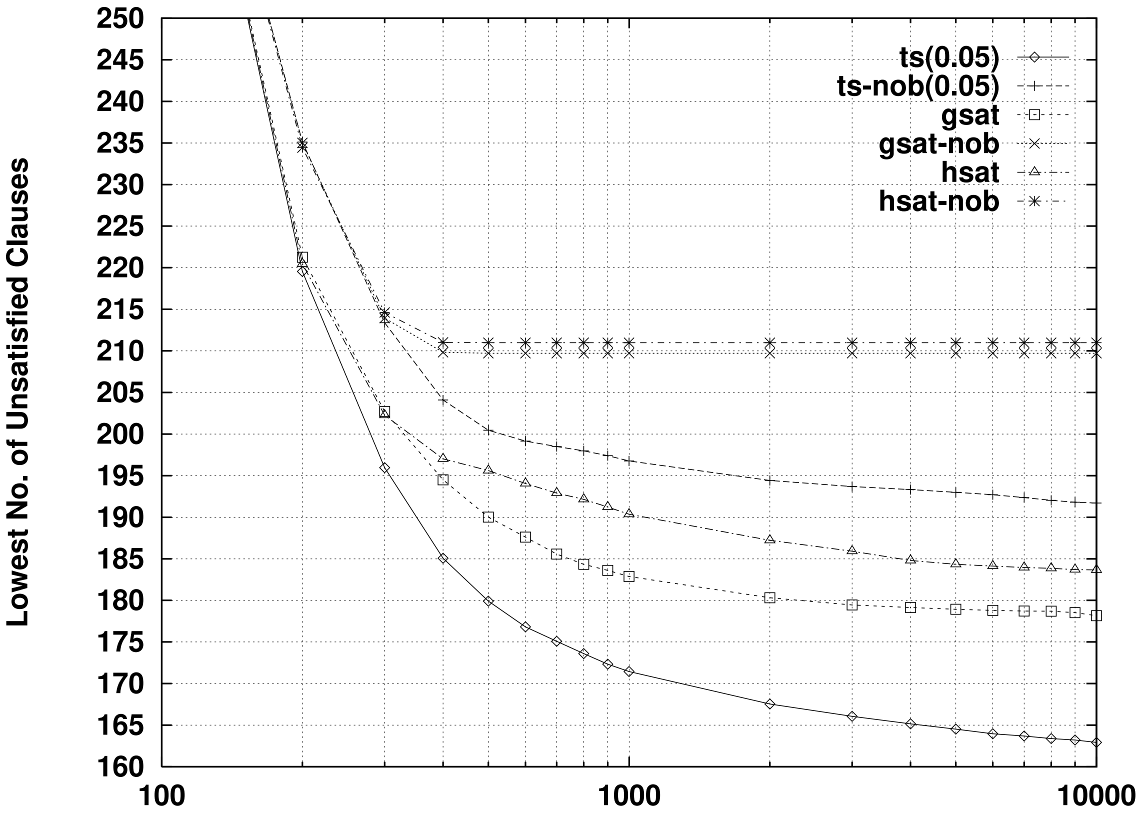
Fig. 4 (bottom) compares the average D-B coordinates of this derived algorithm (label "ts+aspiration+filter") with those of the previous Fixed-TS(T_f) scheme. It can be observed that the addition of the aspiration criterion and, in particular, of the tie-breaking rule greatly reduces the diversification for small T_f values. In particular, the D-B coordinated ($\widehat{H_n}, \widehat{u}$) are (.031 , 235.70) for $T_f = 0$, and (.199 , 191.48) for $T_f = 0.1$. These values are not visible in the bottom plots of Fig. 4, that maintains the same scale as the top one, to facilitate the comparison. The important results is that, for larger T_f values, the D-B coordinates are better that those of the simpler algorithm, in the "corner" Pareto-optimal region. In particular, the coordinates for $T_f =, 0.05$ and $T_f = 0.1$ are better. The comparison is less clear for larger T_f values, when no version dominates the other one.

Given the advantage of the version "TS plus aspiration plus tie-breaking," in the "corner" region, this version will be used in the final algorithm.

4.3 Validation of Pareto-optimal components in D-B plane

Clearly, when one applies a technique for optimization, one wants to maximize the best value found during the run. This value is affected by both the *bias* and the *diversification*. The search trajectory must visit preferentially points with large f values but, as soon as one of this point is visited, the search must proceed to visit new regions.

Let us now test experimentally the above conjecture by measuring the effect of different *diversification - bias* coordinates on the best value found during a run by the different techniques. In a given run each technique starts from a random initial configuration, reaches the first local optimum by using LS and then proceeds for additional $20n$ iterations. The number of iteration is larger than the $4n$ used in the D-B tests executed in the previous section to see



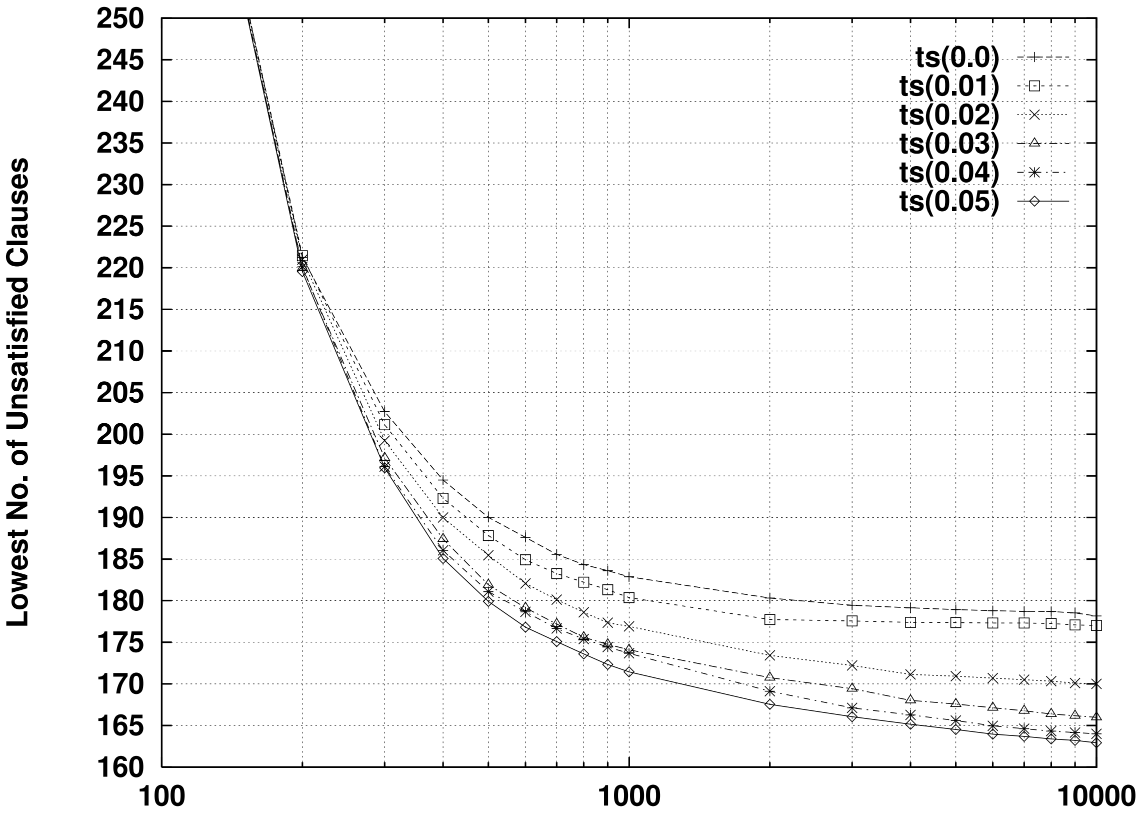


Figure 6: Lowest number of unsatisfied clauses for different techniques (above). TS with different amounts of prohibition (below).

whether the predictions are valid also for larger numbers of iterations. The best value during the run is collected at different numbers of iterations and averaged over tasks and runs.

The first tests (see Fig. 6, above) compare one Pareto-optimal point TS(0.05) of Fig. 4 (top) with a series of non optimal techniques: TS-NOB, GSAT, GSAT-NOB, HSAT, HSAT-NOB. While all methods start in the same statistical conditions (from a local optimum of LS) the difference for large number of iterations is striking: TS(0.05) continues to reach lower marks, while the other techniques show a spectrum of inferior performances. From the D-B plot, other predictions of the above conjecture are satisfied: HSAT is inferior to GSAT in the D-B plot and shows worse results in Fig. 4, HSAT_NOB and GSAT_NOB are suboptimal with respect to TS_NOB(0.05) in the D-B plot and the prediction is again confirmed.

Let us note that no prediction can be derived from the conjecture about the relative performance of TS-NOB(0.05) versus GSAT (GSAT has better bias *but* lower diversification). The fact that GSAT has a better performance means that the increase in bias more than compensates for the decrease in diversification, for this particular comparison.

The worst performance is obtained for non-oblivious functions: in particular the "low mark" does not change after the first $n = 500$ iterations for GSAT-NOB and HSAT-NOB: the diversification is insufficient. The explanation is related to the results described in Fig. 5: while the oblivious function produces many ties, the non-oblivious one does not. Most plateaus with f are eliminated with f_{NOB}, the region around a local optimum tends to have slopes leading to the optimum and escaping from its attraction basin becomes much more difficult.

Fig. 6 (below) compares the mean best values obtained by Fixed-TS(T_f) for values of T_f corresponding to the lower-right portion of the D-B plot shown in Fig. 6 (top), up to the corner point with $T_f = 0.05$. Here again the prediction from the conjecture is correct: Fixed-TS(0.05), the Pareto-optimal with respect to the points with $T_f \leq 0.05$, is also the best option when the best value found during the search is considered.

The conclusion that can be reached from this preliminary series of tests dedicated to the basic local-search based algorithms is that Fixed-TS(T_f), the "standard" oblivious variant of Tabu Search, manages to obtain better diversification properties with respect to the other local-search components, without sacrificing the mean f value along the trajectory. Let us restate the explanation of the success of TS based on the previously formulated conjecture: Fixed-TS(T_f) is better because the component possesses *both* larger diversification *and* larger bias with respect to alternative components.

In passing, let us note that parametric D-B plots can be obtained in a fast way (each component is run only for a small number of iterations, e.g. $4n$ steps) and are therefore an analysis tool that can be used effectively in the construction of complex algorithm based on local-search components.

5 The Hamming-Reactive Tabu Search (H-RTS) algorithm

From the previous investigation Tabu Search with a fixed prohibition parameter – TS(0.05) – obtained the best results on the 500:5000 MAX-3-SAT tasks, when only 20 n iterations are executed.

There are two main motivations for considering a *reactive* version of the TS algorithm. First, tuning the appropriate value of the prohibition for different MAX-SAT problems is time consuming and, unless the tuning process process is fully documented, it is difficult to reproduce for different users, possibly with different degrees of knowledge and smartness. In addition, different values of T may be appropriate for different tasks and, on a specific task, different values may be appropriate in different regions of the search space.

Larger T values imply larger diversification, in particular the relationship between T and the diversification is as follows:

- The Hamming distance H between a starting point and successive point along the trajectory is strictly increasing for $T+1$ steps.

$$H(X^{(t+\tau)}, X^{(t)}) = \tau \quad \text{for} \quad \tau \leq T+1$$

- The minimum repetition interval R along the trajectory is $2(T+1)$.

$$X^{(t+R)} = X^{(t)} \;\Rightarrow\; R \geq 2(T+1)$$

The above relations are immediately demonstrated if one considers that, as soon as the value of a variable is flipped, this value remains unchanged for the next T iterations. In order to come back to the starting configuration, all $T+1$ variables changed in the first phase must be changed back to their original values.

But large values of T imply that only a limited subset of the possible moves can be applied to the current configuration. In particular, T must be less than or equal to $n-2$ to assure that at least two moves can be applied (clearly, if only one move can be applied the choice is not influenced by the f values in the neighborhood). It is therefore appropriate to set T to the smallest value that guarantees diversification.

If the trajectory is at a point X_s at time t_s, a Hamming distance $H(X^{(t)}, X_s)$ equal to $T+1$ will be reached in the time t_s+T+1. Only in the following steps the Hamming distance can decrease.

The actual behavior as a function of different T_f values is illustrated in Fig. 7 that reports the average values obtained from the following experiment. The first local optimum found by LS after $10n$ LS$^+$ iterations is stored. Then Fixed-TS(T_f) is run, with aspiration and tie-breaking, and the Hamming distance from the stored point is measured up to $2(T+1)$ iterations. The average $\widehat{H}_n$ and its standard deviation is reported. To identify the curves for the different T_f values in Fig. 7, let us note that the initial phase is the same, while the curves for larger T_f values produce larger "cones" in the figure. It can be seen that the "incremental ratio" $\Delta H/\Delta t$ during the first $T+1$ steps is 1, as expected, while in the second part larger T_f values are correlated with a higher incremental ratio. In particular, the average incremental ratio obtained with $T_f = 0.01$ (left "cone" in the figure) is negative, while it increases for $T_f = 0.05$ (middle "cone") and $T_f = 0.1$ (right "cone"). The standard deviation bars illustrate the statistical variation of the incremental ratio.

A more systematic analysis is executed in Fig 8, that reports the values of *deriv* (the incremental ratio $\Delta H/\Delta t$ in the last $T+1$ steps, corresponding to the "cone" region in Fig 7, see also Fig. 9) obtained on the 500 tests for each T_f value ranging from 0.01 to 0.5. In addition, average values and standard deviations are reported. As expected, there is a positive correlation between T_f and *deriv* for small T_f values (*deriv* is negative for $T_f = 0.01$ and increases up to $deriv = 0.48$ for $T_f = 0.1$. Then a plateau is present and finally the correlation becomes negative for T_f values larger than about 0.2. In particular, *deriv* values close to zero are obtained for T_f close to 0.5. This result can be explained after considering that when T_f tends to 0.5, T tends to $n/2$ and, therefore, the trajectory reaches the region where the peak of probability is located (see the discussion in Sec. 4.1, in particular eqn. 17-18 and Fig. 3) after the first phase of $T+1$ iterations. After this happens, the results of the tests indicate that the probability of exiting that region at Hamming distance around $n/2$ is very small.

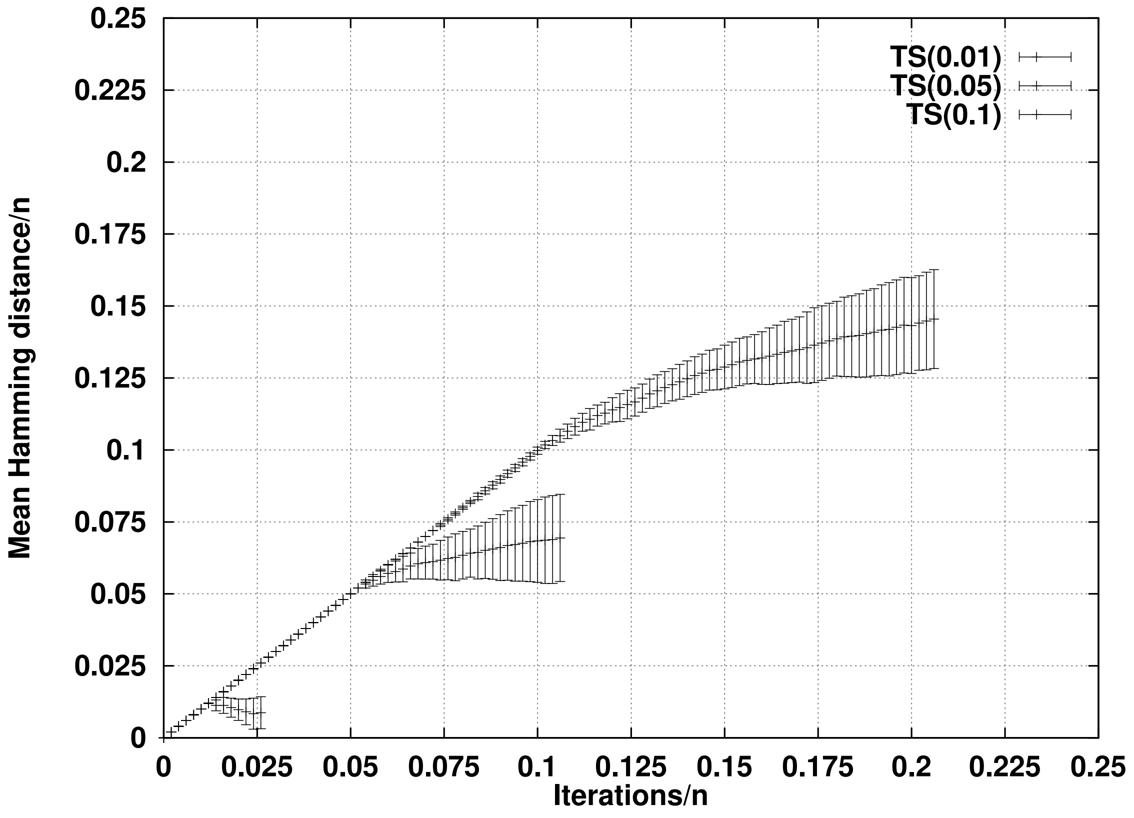


Figure 7: Mean Hamming distance $\widehat{H}/n$ from a given point with st. dev. TS with different amounts of prohibition ($T_f = 0.01, 0.05, 0.1$) starting at LS local optimum, see text for details.

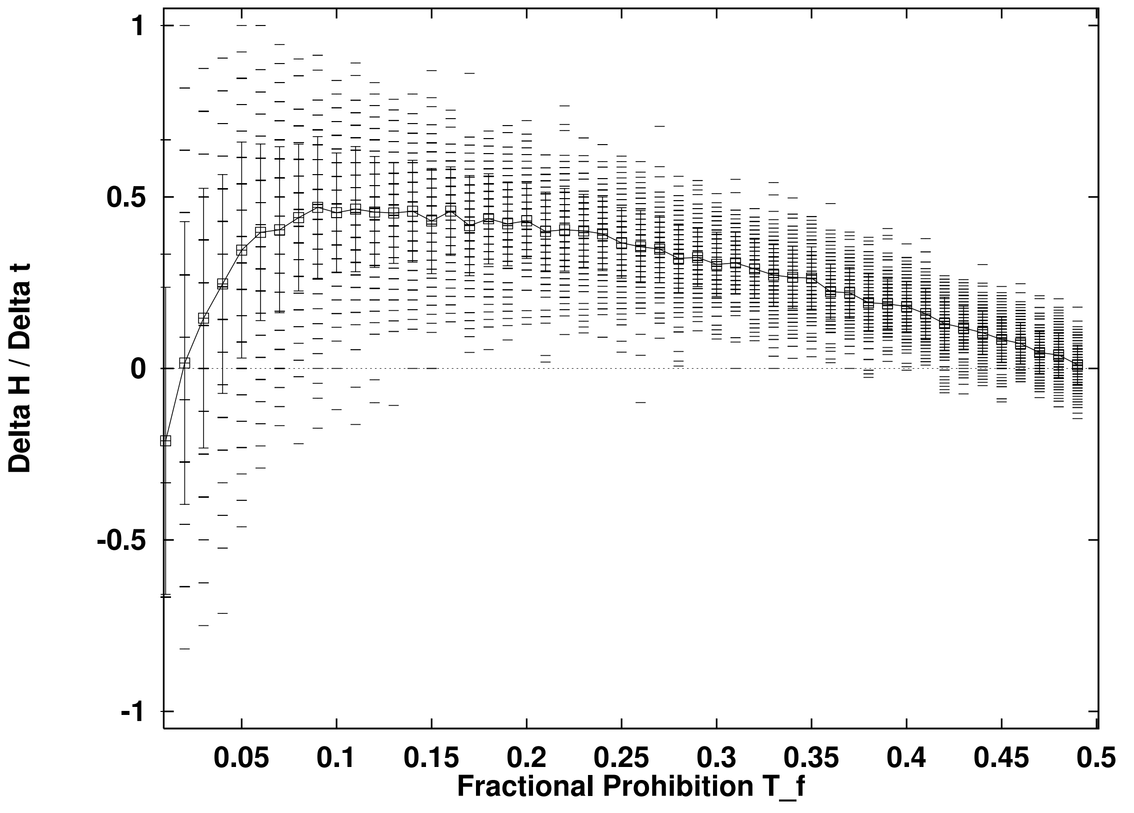


Figure 8: Correlation between T_f and $\Delta H/\Delta t$, actual values, average with st. dev.

```
REACT($T_f, X^{(t)}, X^{(t-2(T+1))}$)
{ Returns the updated prohibition $T$, $T_f$ is the reference to the current
fractional prohibition}
1    $deriv \leftarrow \frac{H(X^{(t)},X^{(t-2(T+1))})-(T+1)}{T+1}$
2    if $deriv \le 0$ then    $T_f \leftarrow T_f + \frac{1}{100}$
3    else if $deriv > \frac{1}{2}$ then    $T_f \leftarrow T_f - \frac{1}{100}$

4    if $T_f > \frac{1}{4}$ then    $T_f \leftarrow \frac{1}{4}$
5    else if $T_f < \frac{1}{40}$ then    $T_f \leftarrow \frac{1}{40}$

7    return  $\max\{\lfloor T_f n \rfloor, 4\}$
```

Figure 9: React: feedback scheme to adjust the prohibition T_f.

5.1 Reactive determination of the prohibition

Let us see how, after considering the last analysis, a simple *reinforcement learning* feedback mechanism can be used to adapt the T value during the search.

There are two possibilities after the trajectory reaches a Hamming distance $H(X^{(t)}, X_s)$ equal to $T+1$ at iteration t_s+T+1 : i) H does non increase and the trajectory remains close or is attracted again toward X_s, ii) H keeps increasing. Now, the fact that the incremental ratio is less than or equal to zero, event i), is considered as evidence that T_f is not sufficient to diversify and must be increased. Vice-versa, if ii) happens and the increase in Hamming distance is very fast, T_f is decreased, to test whether a smaller value is sufficient to diversify.

In addition, an upper bound has to be set for T_f so that the region with a strong negative correlation is not entered. From the previous results, the value of $T_f = 0.25$ is chosen to delimit this region: after this value, the *deriv* values decrease rapidly toward zero in Fig. 8.

The pseudo-code of the React feedback scheme is illustrated in Fig. 9. First the rate of increase of the Hamming distance in the last (T+1) steps is calculated in line 1. Then the reaction on T_f according to the above given directives is executed (lines 2, 3). Finally, the obtained T_f is adjusted so that it belongs to the range $[1/40, 1/4]$ and the updated prohibition T is returned. The lower bound of 4 on T_f assures that a suitable number of iterations (at least $T+1=5$) are executed before calculating *deriv*. Crearly, a sufficiently large number of variables (at least 6) is assumed.

Clearly, we are not claiming that the above React scheme is optimal, other options are possible for a reactive scheme to determine T_f. What is remarkable is that this simple scheme, realized by observing the results of 500:5000 tasks, is in fact appropriate to obtain competitive results on all considered benchmark instances.

5.2 The complete algorithm

Let us now describe the complete Hamming-Reactive-Tabu-Search algorithm, whose structure takes into account the results of the previous analysis.

Fig. 10 describes the core of the algorithm. The initial truth assignment is generated in a random way, and NOB local search is applied until the first local optimum of f_{NOB} is encountered. LS-NOB obtains local minima of better average quality than LS-OB, but then the guiding function becomes the standard oblivious one. This choice was motivated by the success of the NOB & OB combination [7] and by the poor diversification properties of NOB alone, see Sec. 4.2– 4.3.

```
HAMMING-REACTIVE-TABU-SEARCH
1   repeat
2       $t_r \leftarrow t$
3       $X \leftarrow$ random truth assignment
4       $T \leftarrow \lfloor T_f \, n \rfloor$

5       repeat                                   { NOB local search }
6           $X \leftarrow$ BEST-MOVE $(LS, f_{NOB})$
7       until largest $\Delta f_{NOB} = 0$

8       repeat
9           repeat                               { local search }
10              $X \leftarrow$ BEST-MOVE $(LS, f_{OB})$
11          until largest $\Delta f_{OB} = 0$
12          $X_I \leftarrow X$

13          for $2(T+1)$ iterations              { reactive tabu search }
14              $X \leftarrow$ BEST-MOVE $(TS, f_{OB})$
15          $X_F \leftarrow X$

16          $T \leftarrow$ REACT$(T_f, X_F, X_I)$
17      until $(t - t_r) > 10\, n$
18  until solution is acceptable or maximum number of iterations reached
```

Figure 10: The H-RTS algorithm.

The search proceeds by iterating phases of local search followed by phases of TS (lines 8–17 in Fig. 10), until 10 n iterations are accumulated after starting from the random initial truth assignment. The variable t, initialized to zero, contains the current iteration and increases after a local move is applied, while t_r contains the iteration when the last random assignment was generated.

During each combined phase, first the local optimum of f is reached, then $2(T+1)$ iterations of Tabu Search are executed. The design principle underlying this choice is that prohibitions are necessary for diversifying the search only after LS reaches a local optimum. Finally, the "incremental ratio" test is executed and a possible modification of T_f is applied, depending on the *deriv* value calculated, see Fig 9. The fractional prohibition (initialized at the beginning of the run with the value 0.1, see also Sec 6.1) is therefore changed during the run to obtain a proper balance of diversification and bias.

The random restart executed after 10 n iterations guarantees that the search trajectory is not confined in a localized portion of the search space.

Being an heuristic algorithm, there is not a natural termination criterion. In its practical application, the algorithm is therefore run until either the solution is acceptable, or a maximum number of iterations (and therefore CPU time) has elapsed. What we demonstrate in the computational experiments in the next section is that, given a fixed number of iterations, HAMMING-REACTIVE-TABU-SEARCH achieves better average results with respect to competitive algorithms (GSAT and GSAT-with-walk). Because, to a good approximation, the actual running time is proportional to the number of iterations, HAMMING-REACTIVE-TABU-SEARCH should therefore be used to obtain better approximations in a given allotted number of iterations, or equivalent approximations in a much smaller number of iterations.

6 Final experimental tests

The software corresponding to the Hamming-Reactive-Tabu-Search algorithm has been developed in C++ and the algorithm has been tested on the benchmark described in Sec. 3.4. For GSAT and GSAT-with-walk, the data in [30] are not sufficient because we are interested in analyzing the approximation behavior as a function of the number of iterations. Therefore both algorithms are developed in C++ and new tests are run on the same benchmark. The original seeds were not available and therefore the tasks are different from those used in [30], although, clearly, extracted from the same distribution. The parameter *MAX-FLIPS* used is $5\,n$, the probability p for GSAT-with-walk is 0.5 and *MAX-TRIES* corresponds to the desired total number of iterations.

All algorithms are run for a total of $1,000\,n$ iterations, a number such that a "plateau" in the performance of Hamming-Reactive-Tabu-Search is reached and additional iterations do not produce significantly better results (up to $10,000\,n$ iterations have been allowed).

Fig. 11 compares the average results of H-RTS (with initial value $T_f = 0.1$) with those of GSAT and GSAT-with-walk(0.5) on the MAX–3–SAT tasks, with dimensions n from 100 to 500. It is apparent that, for a given number of allotted iterations, H-RTS reaches a much lower number of unsatisfied clauses, especially in the initial part of the search. In fact, H-RTS reaches in the initial phase a number of unsatisfied clauses that is not too far from the value obtained after a very long search. For example, for the 500:5000 tasks, 170 unsatisfied clauses (on the average) are reached by H-RTS at 1,000 iterations, while GSAT and GSAT-with-walk(0.5) need more than 50,000. 160 unsatisfied clauses are reached at 20,000 iterations, while the two competing algorithms are still far from that value after 500,000 iterations.

For the larger-dimensional 1000:10000 MAX-3-SAT tasks, 340 unsatisfied clauses are reached after 1,500 iterations by H-RTS, after 48,000 by GSAT, after 817,000 by GSAT-with-walk(0.5). 330 unsatisfied clauses are reached after 3,200 iterations by H-RTS. GSAT comes close to that value (330.16) at 1,000,000 iterations, while GSAT-with-walk(0.5) is still stuck at 339.38. Only H-RTS reaches a value lower than 320 unsatisfied clauses, at 44,000 iterations.

A similar remarkable performance difference is obtained on the MAX–4–SAT tasks. For example, on the 1000:10000 tasks, 20 unsatisfied clauses are reached at 4,000 iterations by H-RTS, while GSAT and GSAT-with-walk are still stuck at 21.72 and 23.66 unsatisfied clauses, respectively, after 1,000,000 iterations.

The average values obtained for all problem dimensions with the complete runs of $1000\,n$ iterations are collected in Table 2 and Table 3.

Let us note that the CPU times for the runs are affordable. As an example, in our implementation (C++, GNU g++ compiler, Pentium PC with Linux at 120 MHz) the CPU time needed to execute 10,000 iterations is of 4.0 seconds on a 500:5000 MAX-3-SAT task, 7.6 sec on a 1000:10000 MAX-3-SAT task, 9.2 sec on a 1000:10000 MAX-4-SAT task. A detailed analysis of the CPU times is not presented in this paper for brevity reasons. On the other hand, the times on different machines can be easily obtained by the readers by compiling and running the C++ software that is an integral part of this paper.

6.1 Robustness of H-RTS

The main purpose of our *reactive search* scheme has been that of endowing the algorithm with an internal tuning mechanism acting during the run so that the user intervention is avoided.

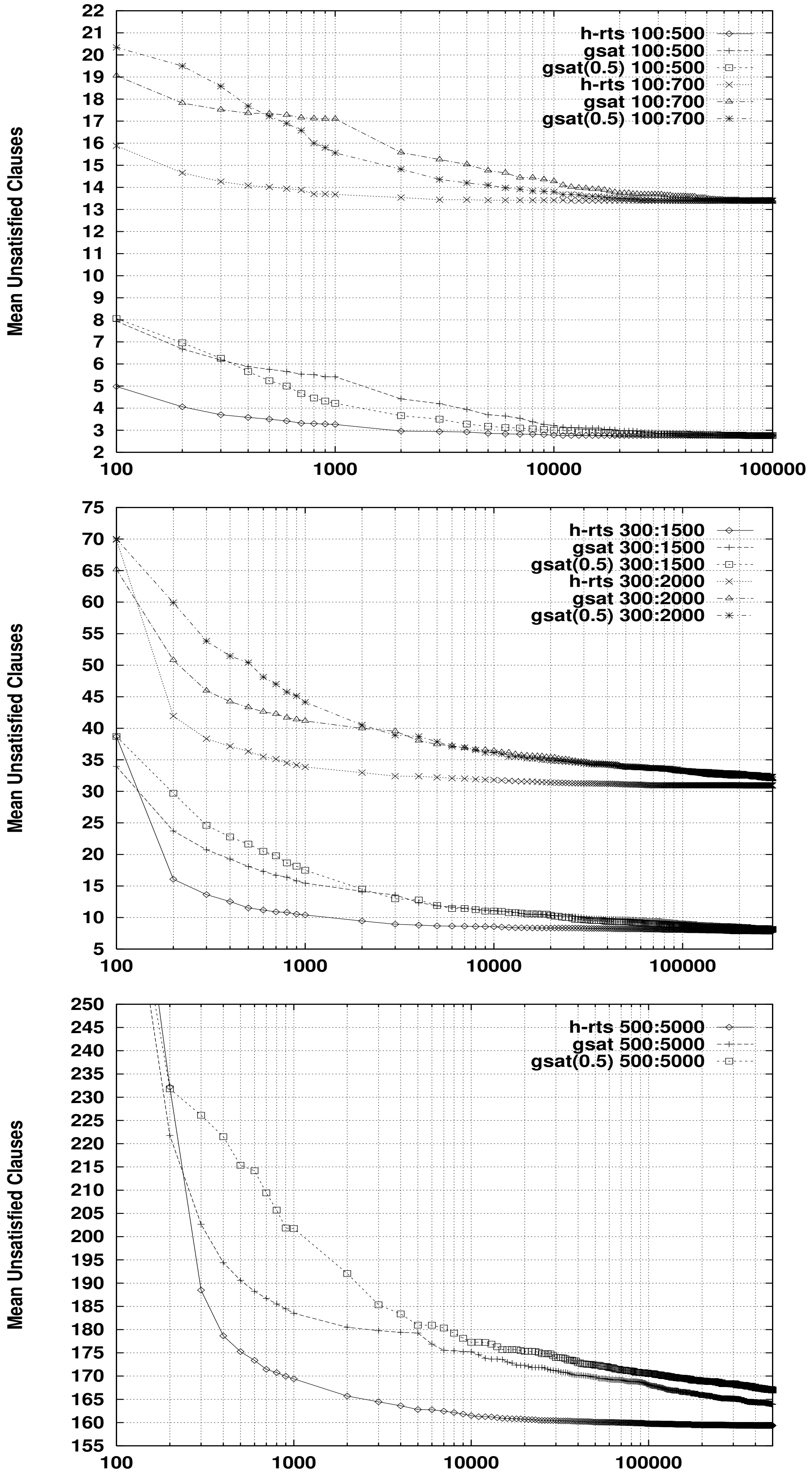


Figure 11: MAX–3–SAT: the H-RTS algorithm (Hamming-Reactive Tabu Search) up to 1,000 n iterations. The results for GSAT and GSAT-with-walk(0.5) are reported with dashed lines for a comparison.

Let us now test that the chosen mechanism is in fact capable of determining in a dynamic way a value of the prohibition parameter such that competitive results are obtained.

The purpose of the following series of tests is twofold: i) to show the robustness of H-RTS when the initial value of T_f at the beginning of the run is changed over a wide range of values and ii) to demonstrate the superiority of the reactive approach with respect to the Fixed-TS and GSAT algorithms.

The average performance of the considered heuristics at the end of different numbers of allotted iterations is collected in Table 2 and Table 3.

variables clauses	100 500	100 700	300 1500	300 2000	500 5000	1000 10000
H-RTS, init. $T_f = 0.02$	2.76 (1.0)	13.40 (1.1)	7.40 (1.5)	30.94 (2.4)	159.32 (4.1)	316.64 (6.63)
H-RTS, init. $T_f = 0.1$	2.76 (1.0)	13.40 (1.1)	7.34 (1.5)	30.96 (2.3)	159.34 (4.1)	316.84 (6.91)
H-RTS, init. $T_f = 0.2$	2.76 (1.0)	13.40 (1.1)	7.38 (1.5)	30.98 (2.4)	159.34 (4.0)	316.53 (6.86)
Fixed-TS(0.02)	5.50 (2.2)	17.26 (3.5)	12.20 (3.2)	38.18 (5.4)	168.50 (9.0)	328.66 (9.92)
Fixed-TS(0.1)	2.76 (1.0)	13.40 (1.1)	7.18 (1.5)	30.98 (2.4)	160.12 (4.2)	321.27 (6.98)
Fixed-TS(0.2)	2.76 (1.0)	13.40 (1.1)	8.32 (1.8)	32.70 (2.7)	167.30 (4.8)	340.66 (7.96)
GSAT	2.78 (1.07)	13.40 (1.19)	8.26 (1.88)	31.94 (2.80)	163.94 (4.97)	330.16 (7.90)
GSAT-with-walk(0.5)	2.76 (1.06)	13.40 (1.19)	8.12 (1.89)	32.28 (2.77)	167.00 (4.88)	339.38 (8.02)

Table 2: MAX–3–SAT: Average performance at the end of $1,000n$ iterations. H-RTS with different initial values of T_f (0.02, 0.1, 0.2) is compared with Fixed-TS with the same (fixed) prohibition values.

variables clauses	100 700	300 1500	300 2500	300 3000	1000 10000
H-RTS, init. $T_f = 0.02$	0 (0)	0 (0)	0 (0)	1.8 (0.9)	9.97 (1.53)
H-RTS, init. $T_f = 0.1$	0 (0)	0 (0)	0 (0)	1.7 (1.0)	9.95 (1.48)
H-RTS, init. $T_f = 0.2$	0 (0)	0 (0)	0 (0)	1.7 (0.9)	10.11 (1.45)
Fixed-TS(0.02)	0.06 (0.23)	0 (0)	1.32 (1.55)	8.90 (2.65)	25.70 (5.66)
Fixed-TS(0.1)	0 (0)	0 (0)	0 (0)	1.94 (0.99)	11.32 (1.64)
Fixed-TS(0.2)	0 (0)	0 (0)	0.10 (0.30)	5.72 (1.22)	28.26 (2.64)
GSAT	0 (0)	0 (0)	0.04 (0.19)	5.06 (1.23)	21.72 (2.49)
GSAT-with-walk(0.5)	0 (0)	0 (0)	0 (0)	4.72 (1.12)	23.66 (2.23)

Table 3: MAX–4–SAT: Average performance at the end of $1,000n$ iterations. H-RTS with different initial values of T_f (0.02, 0.1, 0.2) is compared with Fixed-TS with the same (fixed) prohibition values.

The notable stability of Hamming-Reactive-Tabu-Search when the initial value of T_f is changed can be compared with the changes in the performance of Fixed-TS(T_f) when its T_f value is changed in the same way. Let us note that the tables reports the standard deviation, so that the statistical error on the average is obtained by dividing the standard deviation by $\sqrt{50} \approx 7$, 50 being the number of different tasks. Therefore, the performance variations are much larger than the statistical error.

7 Conclusions

A new heuristic algorithm (H-RTS) has been proposed for the approximated solution of the MAX-SAT problem. The scheme builds upon local search by adding a novel *reactive* scheme to determine in an automated way the appropriate amount of diversification during the run.

The algorithm design has been based on a stepwise analysis complemented by focussed experiments, firstly to study the simple random walk system, secondly to select the most effective building block among different options based on local search, and lastly to engineer the appropriate feedback loop. In detail, the Pareto-optimal component in the first phase is local search with temporary prohibitions (Tabu Search), while the feedback mechanism acts to determine the prohibition period of TS.

In the design phase, only experiments about MAX-3-SAT tasks with 500 variables and 5000 clauses have been used. The obtained algorithm has then been tested without changes on all other cases, including larger dimensional 1000:10000 tasks and MAX-4-SAT tasks. To the best of our knowledge, the best heuristics result on this class of problems have been obtained. Furthermore the results are remarkably robust for changes in the initial values of the prohibition, therefore providing an experimental evidence of the effectiveness of the reactive ("sub-symbolic machine learning") paradigm applied to heuristics.

Acknowledgment

We wish to thank Prof. I. Tamanini, P. Alimonti P. Crescenzi and L. Trevisan for useful discussions, and B. Selman and collaborators for making available their software.

References

[1] E. H. L. Aarts, J.H.M. Korst, and P.J. Zwietering, " Deterministic and randomized local search," in *Mathematical Perspectives on Neural Networks*, edited by M. Mozer P. Smolensky and D. Rumelhart (Hillsdale,NJ:Lawrence Erlbaum Publishers, to appear).

[2] A. V. Aho, D. S. Johnson, R. M. Karp, S. R. Kosaraju, C. C. McGeoch, C. H. Papadimitriou, and P. Pevzner, "Theory of Computing: Goals and Directions," University of Washington Tech. Rep. CSE-93-03, to appear in *Computing Surveys*.

[3] P.Alimonti, " New local search approximation techniques for maximum generalized satisfiability problems," *Proceedings Second Italian Conf. on Algorithms and Complexity*, (1994), 40–53.

[4] G.Ausiello, P.Crescenzi, and M.Protasi, "Approximate solution of NP optimization problems," *Theoretical Computer Science* **150** (1995), 1–55.

[5] R. S. Barr, B. L. Golden, J. P. Kelly, M. G. C. Resende, and W. Stewart, "Designing and Reporting on Computational Experiments with Heuristic Methods," *Journal of Heuristics* **1(1)** (1996), 9–32.

[6] R. Battiti, "Reactive Search: Toward Self-Tuning Heuristics," in *Modern Heuristic Search Methods*, edited by V. J. Rayward-Smith, (John Wiley and Sons Ltd, 1996, to appear).

[7] R. Battiti and M. Protasi, "Solving MAX–SAT with non-oblivious functions and history-based heuristics," DIMACS Workshop on Satisfiability, March 11–13, 1996, Rutgers University.

[8] R. Battiti and M. Protasi, "Reactive local search for the Maximum Clique problem," Tech. Rept. TR-95-052, International Computer Science Institute, Berkeley, CA, 1995.

[9] R. Battiti and G. Tecchiolli, "The reactive tabu search," *ORSA Journal on Computing* **6(2)** (1994), 126–140.

[10] I.P. Gent and T. Walsh, "An empirical analysis of search in GSAT," *Journal of Artificial Intelligence Research* **1** (1993), 47–59.

[11] I.P. Gent and T. Walsh, "Towards an understanding of hill-climbing procedures for SAT," Preprint, 1993.

[12] F. Glover, "Tabu search - part I," *ORSA Journal on Computing* **1(3)** (1989), 190–260.

[13] M. K. Goldberg and D. L. Hollinger, "Learning Better Algorithms for the Maximum Clique Problem," Tech. Report, Rensselaer Polytechnic Institute, Jan 1996.

[14] J. Gu, "Parallel Algorithms and Architectures for very fast AI Search," PhD thesis, University of Utah.

[15] J. Gu, "Efficient local search for very large-scale satisfiability problem," *ACM SIGART Bulletin* **3(1)** (1992), 8–12.

[16] J. Gu, P. W. Purdom, J. Franco, and B.W. Wah, "Algorithms for the Satisfiability (SAT) Problem: a Survey," Preliminary version, 1996.

[17] D.S. Johnson, "Approximation algorithms for combinatorial problems," *J. of Computer and System Sciences* **9** (1974), 256–278.

[18] D.S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning," *Operations Research* **39** (1991), 378–406.

[19] T. Hagerup and C. Rüb, "A guided tour of Chernoff bounds," *Information Processing Letters* **33** (1989/90), 305–308.

[20] P. Hansen and B. Jaumard, "Algorithms for the maximum satisfiability problem," *Computing* **44** (1990), 279–303.

[21] J. Holland, *Adaptation in Natural and Artificial Systems.* (Ann Arbor, MI: University of Michigan Press, 1975).

[22] J. Hooker, "Testing Heuristics: We Have it All Wrong," *Journal of Heuristics* **1(1)** (1996), 33–42.

[23] S.Khanna, R.Motwani, M.Sudan, and U.Vazirani, "On syntactic versus computational views of approximability," *Proceedings 35th Ann. IEEE Symp. on Foundations of Computer Science*, (1994), 819–836,.

[24] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," *Machine Learning Journal, Special Issue on Reinforcement Learning* **8** (1992), 293-322.

[25] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, "Solving large-scale constraint satisfaction and scheduling problems using a heuristic repair method," *Proceedings AAAI-90*, (1990), 17–24.

[26] D. Mitchell, B. Selman, and H. Levesque, "Hard and easy distributions of sat problems," *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), San Jose, Ca* (1992), 459–465.

[27] P. Pardalos, "Continuous approaches to discrete optimization problems," in *Nonlinear optimization and applications*, edited by G. Di Pillo and F. Giannessi, (Plenum Publishing, 1995).

[28] B. Selman and H.A. Kautz, "An empirical study of greedy local search for satisfiability testing," *Proceedings of the eleventh national Conference on Artificial Intelligence (AAAI-93), Washington, D. C.*, 1993.

[29] B. Selman and H. Kautz, "Noise strategies for improving local search," *Proceedings of AAAI-94*, 1994.

[30] B. Selman, H.A. Kautz, and B. Cohen, "Local search strategies for satisfiability testing," *Proceedings of the Second DIMACS Algorithm Implementation Challenge on Cliques, Coloring and Satisfiability, Oct. 1993*, edited by .M. Trick and D. S. Johnson, DIMACS Series on Discrete Mathematics and Theoretical Computer Science, to appear.

[31] B. Selman, H. Levesque, and D. Mitchell, "A new method for solving hard satisfiability problems," *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), San Jose, Ca*, (9992), 440–446.

[32] L. G. Valiant, "A theory of the learnable," *Communications of the ACM* **27(11)** (1984), 1134-1142.

Appendix I: proof of theorem 4.2

Let $\widehat{H}_p^{(t)}$ be the average Hamming distance at iteration $t \geq 0$ derived from eqn. 9, with $p = (1/n) \in (0, 1/2)$. In addition, let $\alpha = 2p$, and, for a generic function $z(t)$ with domain on the integer numbers $t \geq 0$, let the norm be defined as $||z||_\infty = \max_{t\geq 0} |z(t)|$. One must demonstrate that:

$$||\widehat{H}_p^{(t)} - \frac{n}{2}(1 - e^{-\frac{2t}{n}})||_\infty \leq n \, \frac{e^{\alpha-1}}{2} \frac{e^{-\alpha} - (1-\alpha)}{\alpha}$$

Proof. Let us give the main steps in the demonstration. First, for $t \geq 1$ and $\alpha \in (0, 1)$ one has:

$$\begin{aligned} e^{-\alpha t} - (1-\alpha)^t &= (e^{-\alpha})^t - (1-\alpha)^t && (19)\\ &= [e^{-\alpha} - (1-\alpha)] \sum_{h=1}^{t} (e^{-\alpha})^{t-h} (1-\alpha)^{h-1} && (20)\\ &< [e^{-\alpha} - (1-\alpha)] \sum_{h=1}^{t} (e^{-\alpha})^{t-1} && (21)\\ &= te^{-\alpha(t-1)}[e^{-\alpha} - (1-\alpha)] && (22) \end{aligned}$$

where the inequality $e^{-\alpha} > 1 - \alpha$, $\forall \alpha \neq 0$ has been used. Let us now consider the function $\gamma(t) \equiv te^{-\alpha(t-1)}$. The function achieves the maximum in the range $t \geq 1$ at $t = 1/\alpha$. In fact:

$$\begin{aligned} \gamma(1) &= 1 \,,\ \gamma(+\infty) = 0 && (23)\\ \gamma'(t) &= e^{-\alpha(t-1)} - t\alpha e^{-\alpha(t-1)} = [1 - t\alpha]e^{-\alpha(t-1)} && (24)\\ \gamma'(1) &= 1 - \alpha > 0 && (25)\\ \gamma'(t) &= 0 \iff t = \frac{1}{\alpha} && (26)\\ \gamma''(t) &= [t\alpha - 2]\alpha e^{-\alpha(t-1)} = 0 \iff t = \frac{2}{\alpha} && (27) \end{aligned}$$

Therefore one has:

$$\max_{t\geq 0} \gamma(t) = \gamma(\frac{1}{\alpha}) = \frac{e^{\alpha-1}}{\alpha} \tag{28}$$

After starting from eqn. 11 and by using the above inequalities one obtains $\forall t \geq 1$, $\forall \alpha \in (0, 1)$:

$$\begin{aligned} \widehat{H}^{(t)} - \frac{n}{2}(1 - e^{-\frac{2t}{n}}) &= \frac{n}{2}[e^{-\alpha t} - (1-\alpha)^t] \leq && (29)\\ &\leq \frac{n}{2} te^{-\alpha(t-1)}[e^{-\alpha} - (1-\alpha)] && (30)\\ &\leq \frac{ne^{\alpha-1}}{2\alpha}[e^{-\alpha} - (1-\alpha)] && (31) \end{aligned}$$

The theorem immediately follows from the implied inequality:

$$\max_{t\geq 0} |\widehat{H}_p^{(t)} - \frac{n}{2}(1 - e^{-\alpha t})| \leq \frac{ne^{\alpha-1}}{2} \frac{e^{-\alpha} - (1-\alpha)}{\alpha} \tag{32}$$

□