Reactive Tabu Search for MAX-SAT, an Analysis of Sampling Strategies

Roberto Battiti and Mauro Brunato(✉)

Università di Trento, Via Sommarive 9, 38123 Trento, Italy
{roberto.battiti,mauro.brunato}@unitn.it

Abstract. We revisit a reactive algorithm for MAX-SAT that was proposed almost 30 years ago [12]. In the meantime, a large number of heuristics have been developed, in some cases with many free parameters to be carefully chosen. Some of the winners of the recent MAX-SAT competitions combined a variety of different principles (building blocks) to beat the other candidates in a "horse-race" spirit.

We modify the original Hamming-Reactive-TS by replacing the "best improvement" strategy in local (perturbative) search with one based on "first improvement" or on a sample of bit changes to evaluate. After studying the diversification-bias effect of the new choices we show an impressive acceleration. The measure of effort considered in the experiments is primarily the number of function evaluations (of tentative bit-flips tried before the Local Search heuristic applies a move) because it is related to the amount of *information* obtained on the function to be optimized, it is independent of the particular language and compiler, and roughly proportional to the CPU time.

The experiments are run on a selection of many randomized instances intended to simulate hard problems (MAX-3SAT instances close to the boundary between solvable and unsolvable problems) and problems with different degrees of internal structure. A preliminary evaluation considering a current state-of-the-art algorithm concludes the work.

1 Introduction

In our previous work [11] a component based on local (perturbative) 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 (Hamming-Reactive-TS, from now on H-RTS). In addition, the non-oblivious functions introduced in the framework of approximation algorithms [24] are used to discover a better local optimum in the initial part of the search. In addition to the usage for achieving a guaranteed approximation ratio, a non-oblivious guiding function can be used heuristically to encourage a *redundant* satisfaction of clauses, so that eventually some of the variables satisfying clauses in a redundant manner can be negated - without losing the satisfaction - to satisfy additional clauses.

C. Blum et al. (Eds.): WBO 2025, SBD 186, pp. 33–47, 2026.
https://doi.org/10.1007/978-3-032-15455-2_3

The work [9] demonstrates that penalties used in clause-weighting schemes can have dangerous collateral effects: good-quality local minima can be hidden by the transformation of the fitness landscape. On the other hand, recent work (like [16]) uses precisely dynamic clause-weighting to diversify the search by encouraging the satisfaction of clauses that are currently not satisfied. Because clause-weighting is used by many recent competitive algorithms the topic is worth reconsidering. Learning (RL) applied to online parameter tuning in Stochastic Local Search (SLS) methods is considered in the Reactive Tabu Search (RTS) method [8, 10]

Given the limited scope of this contribution, there is no space for a comprehensive presentation of the large variety of heuristics for maximum satisfiability. As some representative contributions, [29] proposes Iterated Robust Tabu Search for MAX-SAT, [22] investigates search space features underlying the performance of stochastic local search. [15] proposes heuristic checks of "photos" of local configurations (neighboring bits, appearing in the same clauses) to avoid cycles with the motivation that a check of repeated configurations is too expensive (which is not necessarily true if hashing schemes are used). An example of the combination of many successful building blocks is considered in [25]. A recent implementation of a competitive clause-weighting scheme is [16]. Changes of two bits per iteration - instead of one - are considered in [30]. The latest MAX-SAT evaluation 2024 (solver and benchmark descriptions) [14] cites a list of competing algorithms with their results.

The fact that prohibition in Reactive Tabu Search are a simple and direct scheme to ensure diversification motivates us to reconsider the H-RTS algorithm in light of the current state-of-the-art. In many cases, the pursuit of "novelty" is not directly related to significant improvements with respect to old algorithms. And in any case, understanding the reasons for the efficacy of building block (or lack of it) is as important as demonstrating superior results in benchmarks or competitions.

2 Satisfiability and Maximum Satisfiability: Definitions

In the Maximum Satisfiability (MAX-SAT) problem one has a Boolean formula in conjunctive normal form, i.e., as a conjunction of clauses, each clause being a disjunction. The objective is to find an assignment of truth values to the variables that satisfies the maximum number of clauses. SAT is the decision version of the problem, i.e., to assess if *all* clauses can be satisfied or not.

In our notation, n is the number of variables and m the number of clauses. A formula has the form:

$$\bigwedge_{1\le i\le m} \Big(\bigvee_{1\le k\le |C_i|} l_{ik} \Big)$$

where $|C_i|$ is the number of literals in clause C_i and l_{ik} is a literal, i.e., a propositional variable u_j or its negation $\overline{u_j}$, for $1 \le j \le n$. The set of clauses in the formula is $\mathbf{C}$. If one associates a *weight* w_i to each clause C_i one obtains the weighted

MAX-SAT problem, denoted as MAX-WSAT: one is to determine the assignment of truth values to the n variables that maximizes the sum of the weights of the satisfied clauses. In some benchmarks one considers problems with different numbers k of literals per clause, defined as MAX-kSAT. Our MAX-kSAT instances contain *exactly* k literals per clause.

MAX-SAT is of interest not only for theory but also for practical applications. The decision version SAT was the first example of an $\mathcal{NP}$-complete problem. Many issues in mathematical logic and symbolic **artificial intelligence** can be formulated as satisfiability or some of its variants, like **constraint satisfaction**. Relevant applications are for example consistency in expert system knowledge bases [26], integrity constraints in databases [1], inductive inference [21,23], digital circuit synthesis [27].

2.1 Local Search for SAT and History-Sensitive Heuristics

Perturbative local search (LS) has been very effective for MAX-SAT. Different variations of LS with randomness techniques have been proposed for SAT and MAX-SAT starting from the late eighties [19,28].

In LS one generates an initial point in the set of admissible solutions and tries to improve it through the application of simple *basic* moves. If a move ("trial") is successful one accepts it, otherwise one keeps the current point.

The use of a guiding function different from the original one can in some cases guarantee local optima of better quality [24].

A truth assignment is described by a binary string. The elementary changes to the current assignment are obtained by changing a single value.

Let $\mathcal{U}$ be the discrete search space: $\mathcal{U} = \{0,1\}^n$, and let $f : \mathcal{U} \longrightarrow R$ (R are the real numbers) be the function to be maximized, i.e., the number of satisfied clauses. Let $U^{(t)} \in \mathcal{U}$ be the current configuration along the *search trajectory* at iteration t, and $N(U^{(t)})$ the neighborhood of point $U^{(t)}$, obtained by applying a set of basic moves μ_i ($1 \leq i \leq n$), where μ_i complements the i-th bit u_i of the string: $\mu_i\ (u_1, u_2, ..., u_i, ..., u_n) = (u_1, u_2, ..., 1 - u_i, ..., u_n)$. These moves are idempotent ($\mu_i^{-1} = \mu_i$).

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

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

$$V = \textsc{Best-Neighbor}(\ N(U^{(t)})\ ) \tag{1}$$

$$U^{(t+1)} = \begin{cases} V & \textbf{if } \ f(V) > f(U^{(t)}) \\ U^{(t)} & \textbf{if } \ f(V) \leq f(U^{(t)}) \end{cases} \tag{2}$$

where Best-Neighbor selects $V \in N(U^{(t)})$ with the best f value and ties are broken randomly. V in turn becomes the new current configuration if f improves. LS stops as soon as the first local optimum point is encountered.

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

Some algorithm parameters can be modified and tuned in an *on-line* manner, to reflect the characteristics of the *task* to be solved and the *local* properties of the configuration space in the neighborhood of the current point [2,5]. This tuning has to be contrasted with the *off-line* tuning of an algorithm, where some parameters or choices are determined for a given problem in a preliminary phase.

2.2 The Hamming-Reactive Tabu Search (H-RTS) Algorithm

Tabu Search (TS) is a *history-sensitive* heuristic proposed by F. Glover [18] and applied it to the MAX-SAT problem in [20]. The 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 $N_A(U^{(t)}) \subset N(U^{(t)})$ of them is *allowed*. The search trajectory is generated as follows:

$$N_A(U^{(t)}) = \textsc{Allow}(N(U^{(t)}), U^{(0)}, ..., U^{(t)}) \tag{3}$$

$$U^{(t+1)} = \textsc{Best-Neighbor}(\ N_A(U^{(t)})\ ) \tag{4}$$

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

The diversification effect of prohibiting moves on the search trajectory has been clarified by the **fundamental relationships between prohibition and diversification** a theorem demonstrated by Battiti and Bertossi in [6]. Let $H(X, Y)$ be the Hamming distance between two strings X and Y, defined as the number of corresponding bits that are different in the two strings. Now, if only allowed moves are executed, and T satisfies $T \leq (n-2)$, which guarantees that at least two moves are allowed at each iteration, one obtains the following.

– The Hamming distance H between a starting point and successive points 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 relationships clearly show how **the prohibition is related to the amount of diversification**: the larger T, the larger is the distance H that the search trajectory must travel before it is allowed to come back to a previously visited point. But T cannot be too large, otherwise a shrinking number of moves will be allowed after an initial phase, leading to less freedom of movement.

The proper value of the prohibition T can be adapted in an online manner to the local characteristics of a specific instance through the Reactive Search Optimization (RSO) principles of "learning while searching" discussed in [7]. An algorithm that combines local search (oblivious and non-oblivious), the use of prohibitions (see TS), and a reactive scheme to determine the prohibition parameter is presented in [3]. The H-RTS algorithm is illustrated in Fig. 2.

```
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 $deriv \leq 0$ then      $T_f \leftarrow T_f + \Delta T_f$
3        else if $deriv > \frac{1}{2}$ then      $T_f \leftarrow T_f - \Delta T_f$

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

Fig. 1. REACT: feedback scheme to adjust the prohibition T_f.

The initial truth assignment is generated randomly, and non-oblivious local search (LS-NOB) is applied until the first local optimum of f_{NOB}. 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 [4] and by the poor diversification properties of NOB alone, see [3].

The search proceeds by repeating phases of local search followed by phases of TS (lines 8–17 in Fig. 2), for a suitable number of iterations (see line 17 in Fig. 2). A single elementary move is applied at each iteration. The variable t, initialized to zero, identifies the current iteration and increases after a local move is applied, while t_r identifies the iteration when the last random assignment was generated.

During each combined phase, first the local optimum of f is reached, then $2(T+1)$ moves 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. The fractional prohibition T_f is changed by the function REACT (see Fig. 1) to obtain a proper balance of diversification and bias [3]; the amount by which T_f is changed is set as $\Delta T_f = 0.01$. The random restart executed after 10 n moves guarantees that the search trajectory is not confined in a localized portion of the search space.

The paper [3] demonstrates that H-RTS achieves much better average results with respect to competitive algorithms (GSAT and GSAT-WITH-WALK).

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

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

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

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

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

Fig. 2. The *H-RTS* algorithm.

Table 1. The four experimental scenarios

Name	vars	clauses	vars/cls	var distribution	satisfiable	description
3CNF	300	2000	3	uniform	unlikely	Classical 3CNF
NU-S	300	2000	$3 \dots 5$	non-uniform	likely	Non-uniform, small
NU-M	500	5000	$2 \dots 20$	non-uniform	likely	Non-uniform, medium
NU-L	500	20000	$2 \dots 10$	non-uniform	unlikely	Non-uniform, large

3 Experimental Evaluation

3.1 Experimental Setup

Test Scenarios. In order to cover a reasonably wide range of scenarios, we focus our experiments on four classes of randomly generated CNF formulas shown in Table 1. Scenario 3CNF is composed by classical 3CNF formulas, while scenarios NU-S, NU-M and NU-L (Non-Uniform Small, Medium and Large respectively) introduce more “structured” formulas by allowing for clauses of different sizes and with non-uniform variable distribution (different variables have different frequencies; the most frequently appearing variable appears twice as much as the least frequent one).

Experimental Setup. To allow for rapid prototyping and testing, the H-RTS algorithm and the variants discussed below have been implemented in Python 3. Therefore, direct CPU time comparison with other techniques is impossible at this moment. In the following Sections, we assume that the number of Boolean CNF formula evaluations is roughly proportional to CPU time in a compiled language such as C; in particular, we consider that, in a Local Search setting, subsequent formula evaluations, barring sporadic restarts whose complexity is easily amortized, happen by single bit flips and allow for fast evaluation by maintaining incremental data structures:

- for every disjunctive clause C_i:
  - list of literals l_{ij} appearing in it, each composed of a variable index and a boolean negation flag;
  - number of literals currently true in the clause: this allows for fast recomputation of the clause's truth value in case of a single variable flip;
- for every variable:
  - list of clauses it appears in.
- (optionally) a data structure allowing for sequential access to variables in a given order, e.g., by decreasing number of unsatisfied clauses they appear in.

Observe that incremental evaluation after a bit flip and maintenance of the described data structures is proportional to the number of clauses the affected variable appears in; the optional ordering structure can be implemented as described in [13,17], achieving a $O(1)$ amortized time per affected clause, therefore not contributing to the overall asymptotic complexity. As shown in Sect. 3.2, however, sorting variables is not beneficial and will not be used in later experiments.

To ensure a robust evaluation, every combination of a scenario and an algorithmic configuration described in this Section used 10 different random seeds for formula generation and 10 random seeds for algorithm execution, for a total of 100 tests per scenario.

3.2 Diversification and Bias

The basic compromise in heuristics is that between *diversification* and *bias*. Given 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. Diversification can be associated with different metrics. We adopt the *Hamming distance* to measure 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.

The *diversification-bias plots* (D-B plots) of different basic components are investigated in [12] and a conjecture is formulated that the best components for a

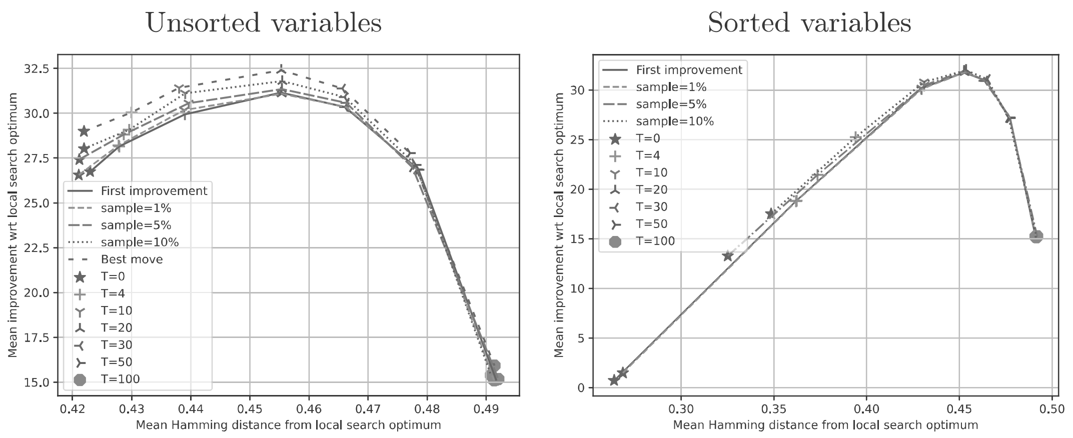

Fig. 3. Diversification-Bias plot for fixed Tabu search on scenario "3CNF". Left: variables are queried in random order; right: variables are queried by decreasing number of unsatisfied clauses they appear in.

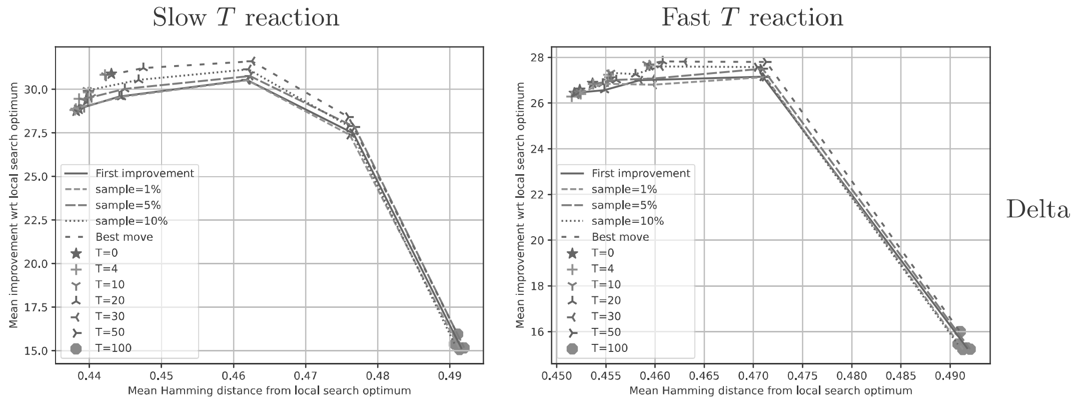

Fig. 4. Diversification-Bias plot for Reactive Tabu search on scenario "3CNF" with different reaction coefficients on the fractional prohibition value T_f. Left: $\Delta T_f = 0.01$; right: $\Delta T_f = 0.1$.

given problem are the Pareto-optimal elements in the diversification-bias plane (configurations that are not dominated by configurations with both a better diversification *and* a better bias).

In Figs. 3 and 4 we plot the number of satisfied clauses versus the Hamming distance, therefore the *maximal* components are in the upper-right corner of the set of $(\widehat{H_n}, \widehat{u})$ points. The points are characterized by the fact that no other point has both a larger diversification and a larger number of satisfied clauses.

The mean bias and diversification depend on the value of the internal parameters of the different components. All runs proceed as follows: as soon as the first local optimum is encountered by a simple Local Search starting from a random configuration, it is stored and the selected component is then run for additional $4n$ iterations. The mean Hamming distance H from the stored local optimum and the mean number of satisfied clauses u are collected. The values are then averaged over different tasks and different random number seeds.

From Fig. 3, left side, we observe that for a given *fixed* prohibition value (no reaction), the neighborhood sample size has little effect on diversification; its effect on improvement is limited. Therefore, a small sample size is likely more efficient if our measure of CPU work is determined by the number of function evaluations, rather than iterations. This observation will further be developed in Sect. 3.3. The right-hand side of Fig. 3, on the other hand, disproves a working hypothesis, namely that when exploring small sample sizes (and in particular when looking for the first improvement), sorting variables by (decreasing) number of unsatisfied clauses should improve efficiency. However, the D-B plot suggests the opposite: in particular, variable sorting yields very poor diversification results when coupled with small prohibition values (due to the same variables being queried again and again); therefore we do not consider this option in the rest of this paper.

Figure 4 shows the diversification vs. bias behavior if the reaction is allowed with different values of the parameter ΔT_f from the reaction function in Fig. 1: the standard, slow-moving prohibition ($\Delta T_F = 0.01$) on the left, a faster-moving prohibition ($\Delta T_f = 0.1$) on the right. We can observe how different initial prohibition values yield results that are closer to the Pareto optimum, as long as extreme values (e.g., $T = 100$) are avoided, therefore motivating the reaction on the prohibition value T.

3.3 Neighborhood Exploration

As suggested by the D-B plot in Fig. 3 (left), and as discussed in Sect. 3.2, the H-RTS heuristic might benefit from spending less time looking for the best local move, to accelerate each iteration. Figure 5 shows a comparison between four neighborhood exploration heuristics:

- Best move: all (non-prohibited) bit flips are tested before determining the one which most improves the objective function (number of satisfied clauses, weighted by appropriate coefficients in the non-oblivious phase); ties are broken randomly;
- First improvement: (non-prohibited) bit flips are tested in random order until an improvement is found;
- Best move in 1% sample: $0.01n$ (non-prohibited) variables are flipped; the best improving flip is selected, with ties broken randomly; if no improvement is found in the sample, revert to First improvement;
- Best move in 5% sample: same as above, but with a larger number of flips tested;

Figure 5 compares these exploration heuristics, with every row (top to bottom) focusing on one of the four test scenarios. The left-hand side chart shows the evolution of the average best-so-far vs. the number of formula evaluations, while the right-hand side boxplot focuses on results at two specific points in the horizontal axis, namely 1,000,000 and 10,000,000 evaluations.

Notice how the best-improvement neighborhood selection heuristic offers two advantages over the others: (i) it ramps up more rapidly in the initial phase

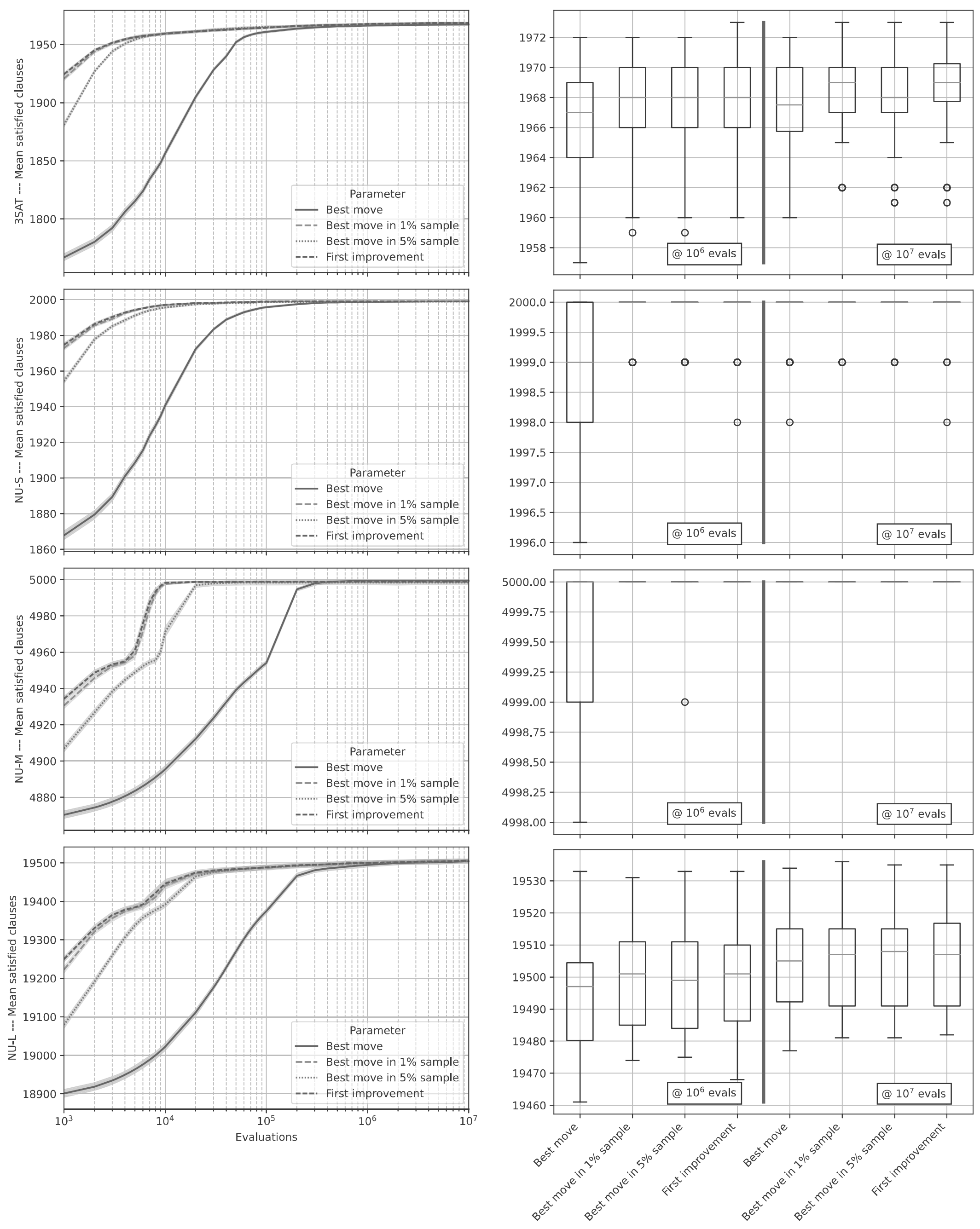

Fig. 5. Mean number of satisfied clauses for the four experimental scenarios vs number of CNF formula evaluations for different local move options.

(initial part of the left-hand charts), while also (ii) not being clearly dominated by the other options when going close to the optimum.

The better performance of the first improvement method is also apparent in Table 2, where the average speedup is reported in terms of the ratio between the number of evaluations required by best-move and first-improvement to achieve specific solution qualities (therefore, the larger the ratio the more effective is

Table 2. Speedup: average ratio (100 samples) between the number of formula evaluations required by best-move vs. first-improvement policies to achieve a given ratio of satisfied clauses.

Scenario	Satisfied clauses		
	90%	95%	97.5%
3CNF	33.9	31.0	17.6
NU-S	85.2	31.0	26.6
NU-M	138.2	138.2	87.4
NU-L	135.0	52.3	9.1

first-improvement wrt best-move). The three thresholds (90%, 95% and 97.5% satisfied clauses) have been chosen in order for every scenario to be fully covered (in particular NU-L, where the optimum tends to have more than 2% unsatisfied clauses).

3.4 Comparison with the NuWLS Algorithm

In addition to the described acceleration, it is of interest to assess whether the original H-RTS algorithm—proposed three decades ago—is still competitive with much more recent MAX-SAT heuristics.

The original version of H-RTS has been tested against NuWLS [16], a recent heuristic which performed well in the MaxSat competition 2022. The algorithm was picked among various candidates because of its performance relative to competing techniques and because its iterative structure allows us to perform direct comparisons in terms of iterations. A hybrid solver that combines NuWLS and an SAT-based solver won all four categories in the incomplete track of the MaxSAT Evaluation 2022.

Because the NuWLS code is written in C and optimized, execution times are not directly comparable. The NuWLS code doesn't output the total number of formula evaluations (tentative bit flips), but the number of completed iterations (moves in the search space). For a rough comparison, in Fig. 6 we plotted H-RTS and NuWLS best-so-far values against the number of search iterations. NuWLS tends to have better values in the starting phase, due to special care in producing the initial solution (that is random in the current version of H-RTS), and it tends to have slightly better results in the long run. In particular, the "easy" NU-M instances are solved by NuWLS within a handful of iterations, while a few thousand local moves are required on average by H-RTS. On the other hand, H-RTS performs better in some intermediate phases.

Let's note that this preliminary comparison does not take into account the substantial acceleration produced by the new modifications studied in this paper and the possible addition of a smarter initialization. Given the "age" of H-RTS (no additional tuning has been executed) and the many free parameters of NuWLS with a setting that has been refined over the years, this preliminary

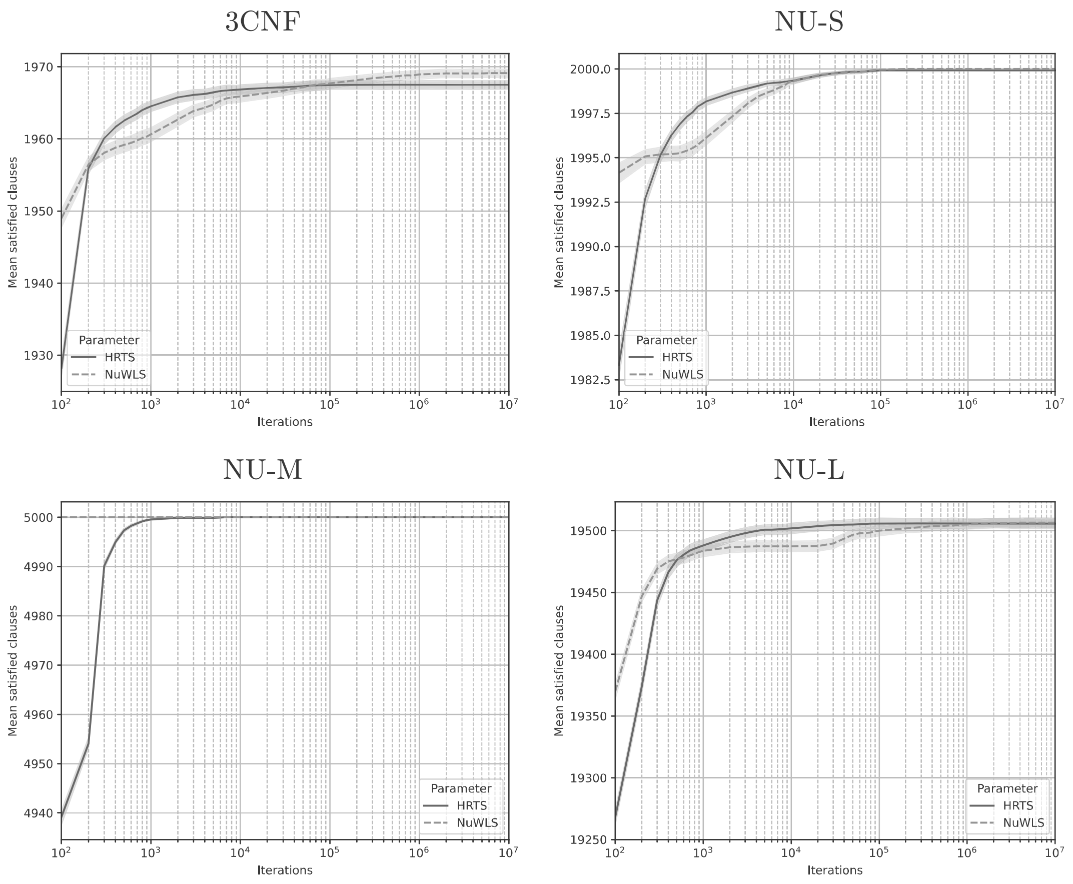

Fig. 6. Mean number of satisfied clauses for the four experimental scenarios vs number of CNF formula evaluations for H-RTS compared with other heuristics.

result is interesting and surprising. A more complete picture will be achieved in the continuation of this research by translating the H-RTS code into C, optimizing it and producing more intermediate results from NuWLS and other competing algorithms.

4 Conclusion

We proposed modifications to the original Hamming-Reactive-TS algorithm dealing with the transition between the "best improvement" strategy (considering all bit changes before picking a move), a sampling version, and the "first improvement" strategy, with and without variable ordering.

The experiments are run on a large selection of randomized instances intended to simulate hard problems (MAX-3SAT close to the boundary between solvable and unsolvable problems) and problems with different degrees of internal structure.

The main results are:

1. We demonstrate that the original Hamming-Reactive-TS algorithm can be accelerated in a substantial manner (by one or two orders of magnitude) by

replacing the best-improvement strategy in the local (perturbative) search with one based on the first improvement or on a sample of bit changes to evaluate. These results highlight the importance of considering simple changes to effective strategies to improve them.
2. We study the diversification-bias effect of the new choices and show that sampling strategies maintain a high diversification level Without losing a lot in terms of function improvement.
3. We perform a preliminary evaluation with current state-of-the-art algorithms sand show that HAMMING-REACTIVE-TS still has the potential to obtain results that are not too far from those of state-of-the-art heuristics, even without considering the acceleration and the possible construction of a better initial solution (not completely random as it is now).

The measure of effort considered in the experiments is primarily the number of function evaluations (of tentative bit-flips tried before the Local Search heuristic applies a move) because it is related to the amount of information obtained on the function to be optimized, it is independent of the particular language and compiler, and roughly proportional to the CPU time.

The continuation of this work will build upon these promising preliminary results to assess whether the new proposals will be at the state of the art; in particular:

– an optimized C++ implementation is under development in order to directly compare CPU times with other algorithms in the recent literature;
– the restart policy is now exceedingly simple; many options can be considered, taking into account the state of the search in an online fashion, or pre-optimizing restart times according to static problem features;
– many other parameters, pertaining the prohibition time reaction mechanism and the neighborhood size, need to be more thoroughly investigated.

References

1. Asirelli, P., de Santis, M., Martelli, A.: Integrity constraints in logic databases. J. Log. Program. **3**, 221–232 (1985)
2. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization, Operations Research/computer Science Interfaces, vol. 45. Springer (2008)
3. Battiti, R., Protasi, M.: Reactive search, a history-sensitive heuristic for MAXSAT. ACM J. Exp. Algorithmics **2**(ARTICLE 2) (1997). http://www.jea.acm.org/
4. Battiti, R., Protasi, M.: Solving MAX-SAT with non-oblivious functions and history-based heuristics. In: Du, D., Gu, J., Pardalos, P.M. (eds.) Satisfiability Problem: Theory and Applications, pp. 649–667. No. 35 in DIMACS: Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, Association for Computing Machinery (1997)
5. Battiti, R., Tecchiolli, G.: The reactive tabu search. ORSA J. Comput. **6**(2), 126–140 (1994)
6. Battiti, R., Bertossi, A.A.: Greedy, prohibition, and reactive heuristics for graph partitioning. IEEE Trans. Comput. **48**(4), 361–385 (1999)

7. Battiti, R., Brunato, M.: The LION way. Machine Learning plus Intelligent Optimization. LIONlab, University of Trento, Italy (2018). http://intelligentoptimization.org/LIONbook/
8. Battiti, R., Brunato, M., Campigotto, P.: Learning while optimizing an unknown fitness surface. In: Maniezzo, V., Battiti, R., Watson, J.-P. (eds.) LION 2007. LNCS, vol. 5313, pp. 25–40. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-92695-5_3
9. Battiti, R., Campigotto, P.: Penalties may have collateral effects. a max-sat analysis. In: Steve Prestwich, C.G.Q., Walsh, T. (eds.) Proceedings of the Workshop on Modeling and Solving Problems with Constraints (in conjunction with the 18th European Conference on Artificial Intelligence, ECAI 2008), Patras, Greece, pp. 8–17 (2008)
10. Battiti, R., Campigotto, P.: Reinforcement learning and reactive search: an adaptive max-sat solver. In: ECAI 2008, pp. 909–910. IOS Press (2008)
11. Battiti, R., Protasi, M.: Reactive search, a history-based heuristic for max-sat. ACM J. Exp. Algorithmics **2**(10.1145), 264216–264220 (1997)
12. Battiti, R., Protasi, M.: Reactive search, a history-sensitive heuristic for max-sat. ACM J. Exp. Algorithmics **2**, 2 (1997). https://doi.org/10.1145/264216.264220
13. Bender, M.A., Cole, R., Demaine, E.D., Farach-Colton, M., Zito, J.: Two simplified algorithms for maintaining order in a list. In: Möhring, R., Raman, R. (eds.) ESA 2002. LNCS, vol. 2461, pp. 152–164. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45749-6_17
14. Berg, J., Järvisalo, M., Martins, R., Niskanen, A., Paxian, T.: Maxsat evaluation 2024: solver and benchmark descriptions. Technical report, Department of Computer Science, University of Helsinki (2024)
15. Cai, S., Su, K., Sattar, A.: Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artif. Intell. **175**(9–10), 1672–1696 (2011)
16. Chu, Y., Cai, S., Luo, C.: Nuwls: improving local search for (weighted) partial maxsat by new weighting techniques. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 3915–3923 (2023)
17. Dietz, P., Sleator, D.: Two algorithms for maintaining order in a list. In: Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 365–372 (1987)
18. Glover, F.: Tabu search - part I. ORSA J. Comput. **1**(3), 190–260 (1989)
19. Gu, J.: Efficient local search for very large-scale satisfiability problem. ACM SIGART Bull. **3**(1), 8–12 (1992)
20. Hansen, P., Jaumard, B.: Algorithms for the maximum satisfiability problem. Computing **44**, 279–303 (1990)
21. Hooker, J.: Resolution vs. cutting plane solution of inference problems: some computational experience. Oper. Res. Lett. **7**(1), 1–7 (1988)
22. Hoos, H.H., Smyth, K., Stützle, T.: Search space features underlying the performance of stochastic local search algorithms for MAX-SAT. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 51–60. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30217-9_6
23. Kamath, A.P., Karmarkar, N., Ramakrishnan, K., Resende, M.: Computational experience with an interior point algorithm on the satisfiability problem. Ann. Oper. Res. **25**, 43–58 (1990)
24. Khanna, S., Motwani, R., Sudan, M., Vazirani, U.: On syntactic versus computational views of approximability. SIAM J. Comput. **28**(1), 164–191 (1998)

25. Luo, C., Cai, S., Wu, W., Jie, Z., Su, K.: CCLS: an efficient local search algorithm for weighted maximum satisfiability. IEEE Trans. Comput. **64**(7), 1830–1843 (2014)
26. Nguyen, T.A., Perkins, W.A., Laffrey, T.J., Pecora, D.: Checking an expert system knowledge base for consistency and completeness. In: Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI 1985), Los Altos, CA, pp. 375–378 (1985) Altos, CA (1985)
27. Puri, R., Gu, J.: A BDD SAT solver for satisfiability testing: an industrial case study. Ann. Math. Artif. Intell. **17**(3–4), 315–337 (1996)
28. Selman, B., Levesque, H., Mitchell, D.: A new method for solving hard satisfiability problems. In: Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI 1992), San Jose, Ca, pp. 440–446 (1992)
29. Smyth, K., Hoos, H.H., Stützle, T.: Iterated robust tabu search for MAX-SAT. In: Xiang, Y., Chaib-draa, B. (eds.) AI 2003. LNCS, vol. 2671, pp. 129–144. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-44886-1_12
30. Zheng, J., He, K., Zhou, J.: Farsighted probabilistic sampling: a general strategy for boosting local search maxsat solvers. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 37, pp. 4132–4139 (2023)