Reactive Local Search Techniques for the Maximum k–Conjunctive Constraint Satisfaction Problem (MAX–k–$CCSP$)

Roberto Battiti [a] Marco Protasi [b,1]

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

[b] *Dipartimento di Matematica, Univ. di Roma "Tor Vergata"*

In this paper the performance of the Hamming-based Reactive Tabu Search algorithm (H-RTS) previously proposed for the Maximum Satisfiability problem is studied for the different Maximum k–Conjunctive Constraint Satisfaction problem. In addition, the use of *non-oblivious* functions recently proposed in the framework of approximation algorithms is investigated.
In particular, two relevant special cases of the Maximum k–Conjunctive Constraint Satisfaction problem are considered: Maximum Directed Cut and Maximum Independent Set in cubic graphs.
The preliminary *diversification–bias* analysis of the basic components shows a remarkable difference between the two problems, and the derived predictions are then validated by extensive experiments with the complete H-RTS algorithm. The performance of H-RTS is compared with that of Simulated Annealing and simple Repeated Local Search.

Key words: Heuristic algorithms, constraint satisfaction, satisfiability, history-sensitive heuristics

⋆ 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 "Algoritmica Sperimentale" Università di Trento

[1] Marco died on February 1st, 1998.

1 Introduction

Many important discrete optimization problems are known to be computationally intractable; formally this means that the associated decision problem is *NP*–complete. Because finding an exact solution in polynomial time is impossible (unless *P*=*NP*), these problems have been studied from an approximation point of view. Actually there are maximization problems, such as Maximum–$\{0,1\}$–Knapsack, for which there are approximation algorithms such that the *performance ratio* (ratio between the optimal value m^* and the approximate one, in the case of maximization problems) can be upper bounded by ρ for any ρ greater than 1. This means that one can approximate the problem for every possible level of error. On the other hand, there are problems, such as Maximum Clique, for which it is possible to prove that no polynomial time approximation algorithm exists that achieves a performance ratio better that $n^{1-\epsilon}$ for any $\epsilon > 0$, where n is the number of nodes of the graph, unless *co–RP*= *NP* (21). An updated compendium of optimization problems with their main complexity properties can be found in (12).

In other words, from a theoretical point of view, the approximation of problems like Maximum Clique is completely intractable. In these cases, the use of heuristics is a possible option. In this paper by heuristic algorithm we mean an algorithm for which no good theoretical approximation behavior has been proved. Of course, one is interested in algorithms that performs effectively in practical situations. Moreover, heuristics play an important role even for problems, such as Maximum Satisfiability (*MAX–SAT*) in which approximation guarantees can be proven (17; 18; 4; 31). In many cases heuristics have been shown to perform much better than the best approximation algorithms when tested on real-world and randomly generated benchmarks considered in the literature.

In this paper we concentrate our attention onto a recently proposed family of heuristics (Reactive Search ,or RS) that seems particularly suitable for dealing with important classes of *NP*–hard optimization problems. Reactive Search advocates the use of simple feedback (sub-symbolic machine learning) schemes in Local Search (LS) algorithms for the adaptation of internal parameters while the algorithm runs on a given instance. In this way the heuristic algorithm maintains the flexibility that is needed to effectively and efficiently solve a set of related problems but the explicit tuning of parameters by the user and the related possible difficulties in reproducing experimental results are avoided. In particular, the "Hamming distance" Reactive Tabu Search algorithm (H-RTS) for the *MAX–SAT* problem (i.e., with disjunctive clauses) obtained state-of-the-art heuristic results on a set of difficult instances in (7).

The purpose of this work is to investigate whether the *same* algorithm is

successful for the Maximum Constraint Satisfaction problems in which each constraint is a *k–conjunction* over a set of Boolean variables. These problems are of particular interest for many reasons. First, their generalization known as Maximum Generalized Satisfiability problem (*GSAT(B)*) plays an important role in the syntactically defined class *MAX–NP* (28). In addition, they model a wide range of relevant problems. For example, this paper considers two important cases: Maximum Cut in Directed Graphs (*MAX–DICUT*) and Maximum Independent Set (*MAX–IND–SET*) in cubic graphs, see Sec. 3.

A secondary issue of this work is the heuristic use of *non-oblivious* modifications to the objective functions, proposed in the framework of approximation algorithms (1; 2; 26) to obtain better approximation ratios for Local Search, and studied for *MAX–SAT* heuristics in (8).

In the following sections, first the problems, the context (non-oblivious Local Search and Reactive Search) and the needed definitions are introduced in Sec. 2. Then the benchmark tasks and the results obtained by Local Search are described in Sec. 3. The preliminary investigation about the diversification–bias properties of different basic schemes is presented in Sec. 4, and, finally, the experimental results obtained by the H-RTS algorithm are compared with those obtained by alternative heuristics in Sec. 5.

2 Context and definitions

2.1 MAX–k–CCSP and special cases

The *MAX–SAT* problem is defined by a set of Boolean variables and a set of clauses, where each clause is the *disjunction* of literals. A literal is either a Boolean variable or its negation.

The Maximum k–Constraint Satisfaction problem (*MAX–k–CSP*) generalizes *MAX–SAT*: it is defined by a set of Boolean variables and a set of *binary constraints* that determine the legal assignment of values to variables.

Definition 1 *Let $\mathcal{X} = \{x_1, ..., x_n\}$ be a set of Boolean variables. A* k–constraint *on $\mathcal{X}$ is $C = (V, P)$, where V is a size k subset of $\mathcal{X}$ and $P : \{True, False\}^k \longrightarrow \{True, False\}$ is a k-ary Boolean predicate.*

Definition 2 *Given a collection $C = \{c_1, ..., c_m\}$ of k–constraints over a set $\mathcal{X}$ of Boolean variables, the MAX–k–CSP problem consists of finding the truth assignment for $\mathcal{X}$ that satisfies the greatest number m^* of constraints.*

The Maximum Conjunctive k–Constraint Satisfaction problem (*MAX–k–CCSP*)

is a proper subclass of *MAX–k–CSP*. A k–conjunctive constraint consists of the conjunction of up to k literals, each literal being either a variable x_j or its negation $\overline{x_j}$.

Definition 3 *Given a collection $C = \{c_1, ..., c_m\}$ of k–conjunctive constraints over a set $\mathcal{X}$ of Boolean variables, the* Maximum k–Conjunctive Constraint Satisfaction (*MAX–k–CCSP*) *problem consists of finding the truth assignment for $\mathcal{X}$ that satisfies the greatest number m^* of constraints.*

Recent results about the approximability of the Maximum Constraint Satisfaction problem, for the case of satisfiable or arbitrary instances, are presented in (29; 30; 25). For instance, in (29), one obtains a performance ratio of 2^{k-1} for the general *MAX–k–CSP*, in (30) one obtains a ratio of $2^k/(k+1)$ for satisfiable instances.

2.2 Non-oblivious Local Search

Because of the intrinsic computational complexity of *NP*-hard optimization problems one cannot efficiently find optimal solutions and one is therefore interested in finding polynomial–time approximations with guaranteed quality, i.e., such that the performance ratio is upper bounded by a constant. Let us now consider Local Search, with the polynomial time assumptions described in (6; 23). When a feasible solution can be represented as a set of items or, equivalently, as a binary string of a given length (e.g., when the solution is a truth assignment) a sensible neighborhood is defined as the set of solutions with bounded Hamming distance from the current one [2].

The *non-oblivious* Local Search technique (1; 2; 26) uses a 1-bounded neighborhood structure (the neighbors are obtained by changing a single bit) like the standard *oblivious* Local Search but uses an auxiliary objective function to guide the search for optimality. Similar modifications of the objective function have been used in different contexts to develop heuristics (see for example (22) for an application to the *TSP* and the recent review in (20) for applications in *MAX–SAT*). The recent results in the area of approximation algorithms are that by means of specific auxiliary functions it is possible: i) to achieve better performance ratios for some problems approximable by the oblivious technique and, ii) to approximate problems non approximable with the oblivious technique.

In particular *MAX–k–CCSP* is not approximable with the oblivious 1-bounded search while it can be approximated with ratio $\rho = (2^k - 1)$ by the non-

[2] Given two binary strings X and Y, the Hamming distance $H(X,Y)$ is defined as the number of corresponding bits that have different values in the two strings.

oblivious search (1). Additional improvements for *MAX–2–CCSP* are possible if the neighborhood is relaxed by considering also the complement of the current solution, leading to a 5/2 approximation ratio. Let us note that these guaranteed ratios are not satisfactory in practice, because, in many cases, state-of-the-art heuristics reach values within a few percent of the optimal one, and therefore an experimental evaluation of the technique is required.

Given a truth assignment X for an instance of *MAX–k–CCSP*, the set of clauses can be partitioned into sets F_j, $j = 0, ..., k$, where each F_j contains the clauses with j false literals. With the above definition, the standard (oblivious) function is simply $f = |F_0|$: one maximizes the cardinality of the set of clauses with all literals matched (zero false literals).

Definition 4 *The* non-oblivious *(*NOB*) function for MAX–k–CCSP is given by the linear combination:*

$$f_{\mathrm{NOB}} = \sum_{j=0}^{k} L_j \cdot |F_j| \tag{1}$$

where the weighting coefficients are defined as follows:

$$L_j = \begin{cases} \frac{1+k \cdot L_{j+1} - (k-j-1) \cdot L_{j+2}}{j+1}, & for \;\; 0 \leq j \leq k-1 \\ L_j = 0, \;\; for \;\; j \geq k \end{cases} \tag{2}$$

To derive the above coefficients one considers the quality of the local optima that can be obtained by considering a particular linear combination. One uses the definition of local optimum (individual variable changes do not increase the number of satisfied clauses) and one sums the non-positive changes over all variables. Finally, one chooses the coefficients that guarantee the above cited approximation ratio. The technical details of the calculation are present in (2).

2.3 Reactive Search and the H-RTS *algorithm*

Reactive Search (RS) is a *history sensitive* heuristic based on Local Search with a simple *machine learning* ("reactive") mechanism for the on-line determination of free parameters (9; 10). In the reactive algorithm considered in this paper the reactive scheme determines the value of a single *prohibition* parameter in *prohibition–based diversification* methods (16; 10). The algorithm is introduced in (7) with the term H-RTS (Hamming-distance based Reactive Tabu Search) and is briefly summarized in the following, together with the needed notation.

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: $\mu_i\ (x_1, x_2, ..., x_i, ..., x_n) = (x_1, x_2, ..., 1 - x_i, ..., x_n)$.

Local search (LS) starts from a random initial configuration $X^{(0)} \in \mathcal{X}$ and generates a search trajectory where, at each iteration, a neighbor with a better f value is selected. In particular, we consider the version where a *best* neighbor is selected and we define as BEST-NEIGHBOR the routine that decides the neighbor. Clearly, Local Search stops as soon as the first local optimum is encountered, when no improving moves are available. LS$^+(max)$ is defined as 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.

Prohibition-based diversification methods have been proposed by Glover with the term *Tabu Search* (TS) (16) and, independently, by Hansen and Jaumard, with the term SAMD (24) ("steepest ascent mildest descent"). 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 FIXED-TS algorithm is obtained by introducing a prohibition parameter T that remains fixed throughout the search (16). 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 LASTUSED(μ) is the last usage time of move μ (LASTUSED$(\mu) = -\infty$ at the beginning):

$$N_A(X^{(t)}) = \{X = \mu\ X^{(t)} \ \text{ such that } \ \textsc{LastUsed}(\mu) < (t - T)\} \qquad (3)$$

Finally, when a *reactive* method (10; 9) is used to tune and therefore change the prohibition period T during the search (the notation is $T^{(t)}$, t being the iteration), 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. 4 below and Fig. 1. The dynamical system becomes:

$$T^{(t)} = \textsc{React}(T^{(t-1)}, X^{(0)}, ..., X^{(t)}) \qquad (4)$$

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

$$X^{(t+1)} = \textsc{Best-Neighbor}\ (N_A(X^{(t)})) \qquad (6)$$

2.4 The H-RTS *algorithm*

The H-RTS algorithm has been introduced in (7) for the *MAX–SAT* problem. The purpose of this paper is that of investigating the capability of the *same* algorithm to deal with different (*conjunctive*) problems. More precisely, the purpose is that of testing that the internal reactive scheme that determines the appropriate balance of diversification versus intensification does indeed endow the algorithm with the flexibility to deal with problems with different characteristics, in an efficient and effective way. By diversification one means the possibility to visit different portions of the search space after encountering a locally optimal point, while by intensification one means the concentration of the search in the the same portion, looking for better local optima.

While we refer to (7) for the detailed introduction of the H-RTS algorithm, let us only mention the two main motivations for considering a *reactive* version of the TS algorithm: automated (and therefore fully reproducible) parameter tuning process and possibility to use different parameter setting in different regions of the search space of a given instance.

The single parameter modified by the procedure REACT is the prohibition T, a parameter related to the diversification. In fact, after a change, the value of a variable remains unchanged for the next T iterations, and therefore i) the Hamming distance H between a starting point and successive points along the trajectory is strictly increasing for $T+1$ steps, ii) if the same configuration is encountered twice along the search trajectory, the separation R must be of at least $2(T+1)$ steps.

Larger T values imply that larger Hamming distances from a given configuration have to be explored before possibly coming back to the starting configuration, but larger T values also 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. It is therefore appropriate to set T to the smallest value that guarantees adequate diversification. 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 such that $T^{(t)}$ remains in the range $[1, n-2]$

A simple *active learning* feedback mechanism is used to adapt the T value. 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 at the subsequent time $t_s + T + 1$. Only in the subsequent steps the Hamming distance *can* decrease. There are two possibilities during the additional iterations: i) H does non increase and the trajectory remains close or is attracted again toward X_s, ii) H keeps increasing. 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. Finally, an upper bound of $T_f = 1/4$, delimits the range where the feedback loop is active.

REACT(T_f, X_F, X_I)
{ *Returns the updated prohibition* T, T_f *is the current fractional prohibition, i.e.,* $T = \lfloor T_f n \rfloor$}
1 $deriv \leftarrow \frac{H(X_F,X_I) \; - \; (T+1)}{T+1}$
2 **if** $deriv \leq 0$ **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** $\max\{T = \lfloor T_f n \rfloor, 4\}$

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

The pseudo-code of the REACT feedback scheme is illustrated in Fig. 1. Let us note that the only part of the *history* actually used is given by the two configurations X_I and X_F defined below. First the rate of increase of the Hamming distance in the last (T+1) steps is calculated in line 1. If $deriv$ is computed at iteration t, X_I is the configuration at the beginning of the TS search phase ($X_I = X^{(t-2(T+1))}$), X_F the configuration at the end of $2(T+1)$ steps ($X_F = X^{(t)}$). The quantity $(T+1)$ subtracted in line 1 corresponds to the distance traveled in the first $(T+1)$ steps. 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 assures that a suitable number of iterations are executed before calculating $deriv$.

The complete H-RTS algorithm is summarized in Fig. 2. 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. By definition, at the local optimum, all possible changes Δf_{NOB} between the current value of f_{NOB} and the values of f_{NOB} in the neighborhood are zero or negative. The BEST-MOVE function is a specific realization of the general BEST-NEIGHBOR function, returning the chosen neighbor with the specified guiding functions, oblivious or non-oblivious. The search proceeds by iterating phases of Local Search followed by phases of TS (lines 9–20 in Fig. 2), 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.

```
H-RTS
1   repeat
2       t_r ← t
3       X ← random truth assignment
4       T ← ⌊T_f n⌋
5       repeat                                   { NOB Local Search }
6           X ← BEST-MOVE (LS, f_NOB)
7           t ← t + 1
8       until largest Δf_NOB ≤ 0
9       repeat
10          repeat                               { Local Search }
11              X ← BEST-MOVE (LS, f)
12              t ← t + 1
13          until largest Δf ≤ 0
14          X_I ← X
15          for 2(T + 1) iterations              { reactive tabu search }
16              X ← BEST-MOVE (TS, f)
17              t ← t + 1
18          X_F ← X
19          T ← REACT(T_f, X_F, X_I)
20      until (t − t_r) > 10 n
21  until solution is acceptable or maximum number of iterations reached
```

Fig. 2. The H-RTS algorithm.

During each combined phase, first the local optimum of f is reached (lines 10–14), then $2(T+1)$ iterations of Tabu Search using the f function are executed (line 15–18). In this case BEST-MOVE depends also on the prohibition parameter T. In our experimental tests, both the oblivious and non-oblivious functions are tried in line 11,13,and 16 depending on the case. 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 1. The fractional prohibition (initialized at the beginning of the run with the value 0.1) 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 (see line 20 and lines 2,3).

Being an heuristic algorithm, there is not a natural termination criterion. The algorithm is therefore run until either the solution is acceptable, or a maximum number of iterations (and therefore CPU time) has elapsed.

3 Benchmark tasks and Local Search results

The performance of Local Search, in the oblivious and non-oblivious versions, is analyzed by running the different versions on a set of randomly-generated instances. The results obtained for the *MAX–DICUT* problem are presented in Sec. 3.1, while the results obtained for *MAX–IND–SET* in cubic graphs are presented in Sec. 3.2.

3.1 The MAX–DICUT problem in graphs

The first series of tasks consists of instances of the *MAX–DICUT* problem.

Definition 5 *Given a directed graph $D = (V, A)$, V being the vertex set and A the arc set, the* Maximum Directed Cut *problem consists of finding a set $S \subseteq V$ such that the cardinality of the directed cut $\delta^+(S) = |\{(i,j) \in A | i \in S, j \notin S\}|$ is maximized.*

MAX–DICUT is *NP*-hard, approximable within 1.165 (14) but *APX*-complete (28), and therefore it is impossible to find an approximation ratio ρ for any $\rho > 1$, unless $P = NP$.

An instance of *MAX–DICUT* is transformed into an instance of Maximum Binary Conjunctive Constraint Satisfaction (*MAX–2–CCSP*) in the following way: to each arc (i,j) one associates a clause $x_i \wedge \bar{x}_j$. The solutions are transformed by including vertex j in S if and only if the variable x_j is *true* in the corresponding *MAX–2–CCSP* solution.

Sets of random instances with different graph densities are generated. In each instance, for all possible pairs of nodes (i,j) such that $i \neq j$ a directed arc (i,j) is generated with probability equal to *density*. This corresponds to the

classic definition of random graphs, whose theory was introduced in (13). For each value of the number of variables n, and for each value of the *density*, 50 random instances are generated 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 $(n, density)$ couple.

variables	100	100	100	100	100
density	0.1	0.3	0.5	0.7	0.9
avg. clauses	999.1 (7.2)	2992.4 (9.3)	4952.3 (13.7)	6928.3 (15.0)	8900.1 (9.4)
LS-OB	361.3 (12.7)	918.6 (15.9)	1431.8 (16.7)	1911.0 (15.5)	2343.6 (10.9)
LS-NOB	364.6 (11.7)	921.5 (15.2)	1434.2 (14.6)	1914.6 (13.5)	2350.7 (9.6)
NOB & OB	367.0 (11.6)	926.0 (14.6)	1438.9 (14.6)	1919.5 (13.6)	2353.0 (9.5)
NOB & OB& $10 \times n$ iter.	370.6 (11.0)	929.9 (13.7)	1443.7 (14.4)	1924.6 (13.0)	2359.8 (8.9)

Table 1
MAX–DICUT: Mean number of satisfied clauses with standard deviation. OB is the oblivious search, NOB the non–oblivious one, see text for details.

Table 1 summarizes the mean number of satisfied clauses obtained by simple Local Search algorithms, using either the oblivious (LS-OB) or the non-oblivious (LS-NOB) function. The main result is that the NOB Local Search does lead to local optima of a better average quality with respect to OB. This result confirms what has been found in (8) for the case of the usual disjunctive SAT problem. Better local optima are found if OB Local Search starts from a local optimum of NOB (line NOB & OB in Table 1), and still better ones if $10 \times n$ additional iterations of LS^+ are allowed (the best move is accepted even if it leads to worse function values).

By considering the dependence on the density, let us note that the relative improvement (clauses satisfied by LS-NOB minus clauses satisfied by LS-OB, divided by clauses satisfied by LS-OB) of the average number of clauses satisfied by LS-OB and LS-NOB decreases for larger densities, ranging from approximately 0.9 % for density 0.1 to 0.3 % for density 0.9. Although not large in relative terms, the performance difference is significant when one considers that the results found by the more complex H-RTS heuristic algorithm running for very large number of iterations are better only by about 2 % for density 0.1 and 0.5 % for density 0.9, see Section 5.

The mean number of iterations of the different LS-based components is illustrated in Table 2. Let us note that LS-NOB requires between 22 % (density 0.1) and 38 % (density 0.9) more iterations than LS-OB. If OB is started

variables	100	100	100	100	100
density	0.1	0.3	0.5	0.7	0.9
LS-OB	37.7 (5.4)	39.7 (5.8)	39.4 (5.8)	37.4 (5.8)	29.7 (5.1)
LS-NOB	46.1 (5.6)	46.9 (5.4)	45.9 (5.6)	45.5 (5.4)	41.1 (5.5)
NOB & OB	48.4 (5.8)	50.4 (5.6)	49.3 (5.9)	48.9 (5.9)	42.8 (5.6)

Table 2
MAX–DICUT: Mean number of iterations (flips) with standard deviation. OB is the oblivious search, NOB the non–oblivious one, see text for details.

from a NOB local optimum, the local optimum of the OB function is found in a very small number of additional iterations (line NOB & OB in Table 2).

3.2 The MAX–IND–SET problem in cubic graphs

Definition 6 *Given a graph $G = (V, E)$, V being the vertex set and E the edge set, the* Maximum Independent Set *(MAX–IND–SET) problem consists of finding a subset $I \subseteq V$ such that no two vertices in I are joined by an edge in E ($i \in I$ and $j \in I \Longrightarrow (i,j) \notin E$) and whose cardinality is maximized.*

MAX–IND–SET is *NP*-hard, it is the same problem as Maximum Clique in the complementary graph, which is not approximable within $|V|^{1-\epsilon}$ for any $\epsilon > 0$ unless *co–RP*= *NP* (21). Additional approximability properties for special cases are collected in (12).

In particular, the *MAX–IND–SET* problem for *cubic* graphs (graphs where each vertex has degree equal to three) is considered in this paper. The motivation is that cubic graphs are "at the boundary" between the difficult and the solvable problems in graphs. In fact several *NP*-hard problems remain *NP*-hard if restricted to cubic graphs, but become polynomial time solvable for graphs of degree two (19). In particular, the APX-hardness of *MAX–IND–SET* on cubic graphs has been recently demonstrated in (3).

An instance of *MAX–IND–SET* is represented as a logic formula and transformed into an instance of *MAX–4–CCSP* in the following way. To each vertex i in the graph one associates a Boolean variable x_i and a clause that describes the connections of the node with its neighbors. If node i is connected to nodes j_1, j_2, and j_3, the corresponding conjunctive clause is $\overline{x_i} \wedge x_{j_1} \wedge x_{j_2} \wedge x_{j_3}$. Let us note that variable x_i appears in exactly four clauses, is negated in one clause (the clause above cited) and positive in the other three clauses.

The solution is transformed by inserting into the independent set the vertices such that the corresponding variable is negated and the clause containing the

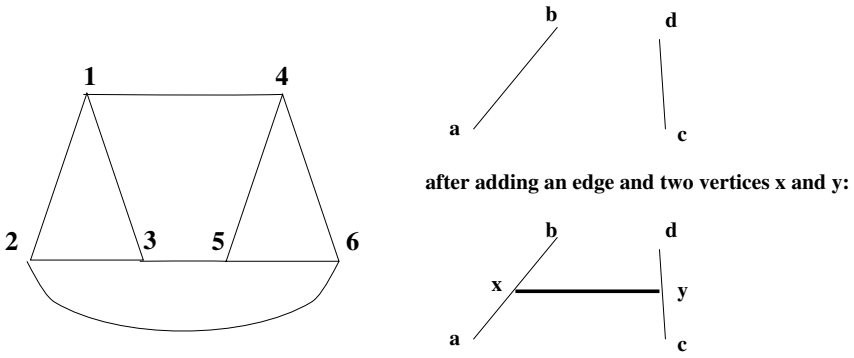

Fig. 3. Generation of random cubic graphs. Initial graph (left), addition of two new vertices x and y (right).

negated variable is satisfied. Clearly, the clauses represent the fact that, if vertex i is in the independent set (and $\overline{x_i}$ is true), all neighbors must not be in the independent set.

Structured instances of cubic graphs are generated in the following way: one starts from an initial cubic graphs with six edges (see Fig. 3, left) and repeatedly extracts two random edges not incident onto the same vertex and substitutes them with the "H" pattern illustrated in Fig. 3 (right), until the desired number of vertices is reached. Let us note that, in this way, the probability distribution obtained is not uniform on all possible cubic graphs, in addition some graphs (for example graphs with self-loops) are not generated at all. More general random cubic graphs were tested before choosing our benchmark instances. In particular, graphs such that the required number of edges is generated sequentially, picking at each iteration with uniform probability among all admissible edges (edges connecting nodes with degree less than three). These graphs can contain *self-loops* and can therefore generate conjunctions containing both a variable and its negation, conjunctions that are eliminated before applying our algorithms. Because the performance of the different algorithms on these more general graphs maintains the same ranking obtained on the more structured graphs (see below for the Local Search results) and because an uneven distribution of the probability is closer to most real-world situations we chose the more structured graphs for our benchmarks.

As it was the case for the *MAX–DICUT* problem, 50 instances are generated by varying the random number generator and the different algorithms are run 10 times for each instance.

Table 3 reports the average results of the different versions of Local-Search based techniques. Again, better local optima (by approximately 5 %) are found by LS-NOB, but now local optima of the NOB function are also local optima

variables	100
LS-OB	38.0 (1.9)
LS-NOB	40.1 (1.3)
NOB & OB	40.1 (1.3)
NOB & OB& $10 \times n$ iter.	43.2 (0.7)

Table 3
MAX–IND–SET in cubic graphs: Mean number of satisfied clauses with standard deviation. OB is the oblivious search, NOB the non–oblivious one.

of the standard (oblivious) function and no additional improvement is found by the combined NOB & OB technique. Furthermore, the performance increases if additional $10n$ iterations are executed. By using our previous work for the *MAX–CLIQUE* problem (9), the average size of the maximum clique found in the complement graphs is of 44.4, a result that is better than that obtained by LS-NOB by about 10 %. This result is in very close agreement with those found in Section 5 by the H-RTS heuristic.

variables	100
LS-OB	27.5 (3.1)
LS-NOB	29.6 (3.3)

Table 4
MAX–IND–SET in cubic graphs: Mean number of iterations (flips) with standard deviation. OB is the oblivious search, NOB the non–oblivious one.

The number of iterations required by the two techniques is shown in Table 4: LS-NOB requires about 7% more iterations.

If the more general random graphs previously mentioned are used for the above tests, LS-OB, LS-NOB, and NOB & OB obtain the following average numbers of satisfied clauses (with std. dev.): 37.0 (2.1), 39.4 (1.4), and 39.4 (1.4), respectively. If $10 \times n$ additional LS^+ iterations are allowed one obtains 40.3 (2.2). These results show that a small improvement, of 0.9 additional satisfied clauses, is obtained with the additional iterations. The average number of flips are 26.7 (3.2) for LS-OB, 30.0 (3.4) for LS-NOB. The performance ranking is the same as that obtained on the more structured graphs, and the advantage of additional LS^+ iterations is stronger for the more structured graphs. The fact that more structured graphs (not picked with a uniform distribution) tend to be harder to solve and therefore to require more complex heuristics beyond the search of the first local optimum motivated us to pick them for the subsequent tests.

Motivated by the experimental findings, it is easy to demonstrate the following proposition:

Proposition 7 *For the problem MAX–4–CCSP a local optimum found by the non-oblivious function is also a local optimum for the oblivious function.*

Proof.
The proof is by contradiction: one assumes that the assignment is *not* an OB optimum and concludes that the assumption that it was a NOB optimum is not true.

Because we consider problems with $k = 4$, it is convenient to multiply the function defined in eqn. 1 by 12 to obtain: $f_{\mathrm{NOB}} = 64\ |F_0| + 19\ |F_1| + 8\ |F_2| + 3\ |F_3|$. Let us assume that one starts from a NOB local optimum. If it is *not* an OB local optimum, there is a variable x such that, by flipping its truth value one has $\Delta_x f_{\mathrm{OB}} > 0$, where $\Delta_x f$ is the change in the value of f caused by a change of the truth value of variable x .

If the OB function is considered, each variable change can satisfy a limited number of additional clauses: three if the variable becomes true, one if it becomes false (trivial, because each variable appears positive in three clauses and negative in exactly one clause).

Now, if x is true in the current assignment, $\Delta_x f_{\mathrm{OB}} > 0$ implies that the unique clause containing $\bar{x}$ becomes true after flipping the variable and f_{NOB} gains at least 45 (64-19). But f_{NOB} loses at most 11×3 when the three clauses containing literal x lose one matched literal. Therefore $\Delta_x f_{\mathrm{NOB}}$ is greater than zero, which contradicts the assumption.

If x is false in the current assignment, $\Delta_x f_{\mathrm{OB}} > 0$ implies that f_{NOB} gains more than 45 when the three clauses containing x gain one additional matched literal. But f_{NOB} loses at most 45 by considering the clause that contains $\bar{x}$. Again, $\Delta_x f_{\mathrm{NOB}}$ is greater than zero, which contradicts the assumption.

□

4 Preliminary investigation: DB plots and relation between prohibition and diversification

Our previous work advocates the use of a preliminary study of the *diversification* and *bias* of individual algorithms in the process of constructing more complex heuristics (7). In (7) the individual algorithms were based on Local

Search and were obtained from the same generic structure by fixing in different ways the value of some parameters.

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 and the proper balance of the two is crucial to the effectiveness and efficacy of the heuristic.

The "falsifiable" assumption that is made is that a simple *metric* (given by the average Hamming distance and average f values at the end of a *short* run of each component) is predictive of the final success of a component as part of the more complex H-RTS scheme described in Section 2.4, when the best solution found during the entire run is considered.

The conjecture proposed is that the components producing (in the average) the best f values during the entire run of the H-RTS algorithm are the *maximal* (i.e. non dominated) components in the diversification-bias (D-B) plane. A component A is dominated by component B if B has *both* a larger diversification *and* a better bias, where the diversification is measured by the Hamming distance and the bias is measured by the number of satisfied clauses, see Sec. 4.1 for the details.

In addition to the interest of the D-B study in the algorithm design process, the results obtained provide an empirical description of the trade-off between bias and diversification when algorithm parameters are changed.

4.1 DB plots

When a Local Search component is started, new configurations are obtained at each iteration until the first local optimum is encountered, because the f value increases (here f represents both f_{OB} and f_{NOB}). 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 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.

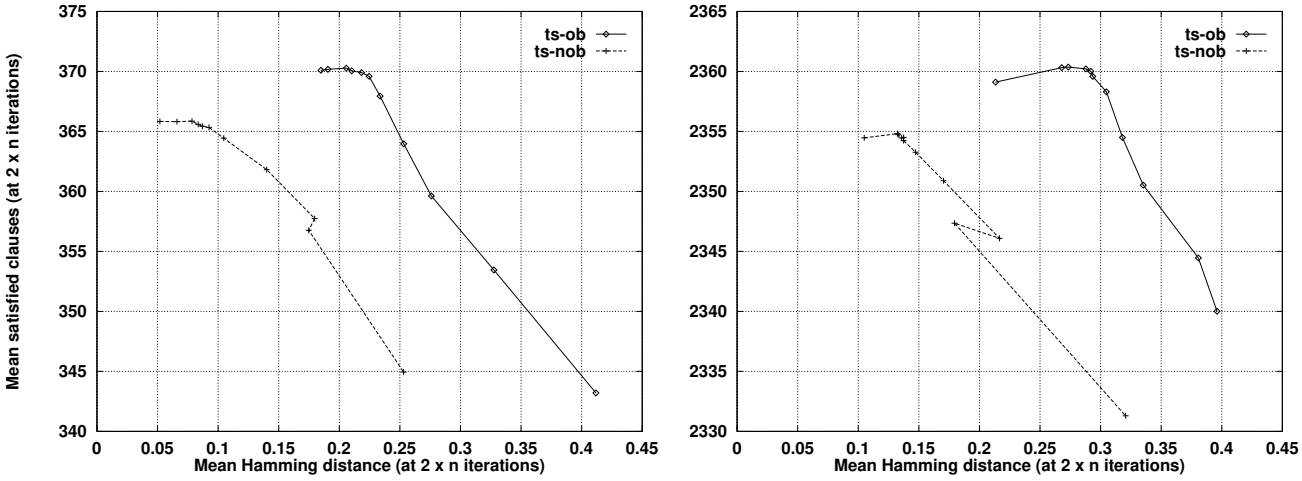

Fig. 4. *MAX–DICUT*: diversification–bias plot. Each run starts from LS local optimum. Graphs with 100 nodes, of density 0.1 (left) and 0.9 (right). $T_f = 0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5$. Prohibition is increasing along the curve when one moves rightwards.

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 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 (OB or NOB depending on the component tested), it is stored and the selected component is then run for $2n$ additional iterations. The final Hamming distance H from the stored local optimum and the final value of the number of satisfied clauses c are collected (let us call "check point" the situation at $2n$ iterations after the local minimum).

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 (*MAX–DICUT*) and in Fig. 5 (*MAX–IND–SET* in cubic graphs). Two basic algorithms are considered: FIXED-TS-OB(T_f) and FIXED-TS-NOB(T_f), using a fixed fractional prohibition parameter T_f. One obtains two parametric curves, along which the T_f parameter takes the values 0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5. Each point gives the D-B coordinates $(\widehat{H_n}, \widehat{c})$, i.e., average Hamming distance divided by n and average number of satisfied clauses, for a specific setting of T_f. Each curve starts with a point for zero prohibition (the leftmost point) and then connects the points with larger prohibitions, situated to the right.

Let us consider the *MAX–DICUT* results. Fig. 4 shows a remarkable *rank inversion* between the OB and NOB results when Local Search is continued

with no prohibitions: while the local optima reached by LS-NOB are of better quality, the average number of satisfied clauses after $2n$ additional iterations is worse. For example, for density 0.1, LS-OB satisfies 361.3 clauses at the local optimum, and **370.1** at the "check point", while LS-NOB satisfies 364.6 at the local optimum, and only **365.8** at the check point. This inversion is related to the very poor diversification properties of LS-NOB: the average Hamming distance is only 5.1 bits at the check point, while the Hamming distance reached by LS-OB is much larger: 18.4 bits. In other words, LS-NOB reaches a good initial local optimum but then spends too many iterations in a ball around the local optimum with a very limited Hamming radius, while LS-OB starts from a worse local optimum but then visits points that are at much greater distances, finding eventually much better solutions.

The effect of prohibitions larger than zero on the diversification is very clear: for small prohibitions up to a "knee point" for $T_f \approx 0.05$ one obtains larger Hamming distances without obtaining worse average f values. The component algorithms corresponding to these point therefore dominate points of zero or very small prohibitions. Then, after the "knee point", larger Hamming distances are reached but the f values worsen in a rapid way. The explanation is that too many moves are prohibited and the trajectory is forced to pick very poor neighbors.

Let us finally note the slight "irregularity" in the NOB DB plots for $T_f = 0.4$ (the point just before the last one to the right): the Hamming distance is less than what could be expected by extrapolating from the points at lower T_f values. The irregularity is significant (larger than the experimental errors), it disappears if the check point is at a larger number of iterations (it is hardly visible after $4n$ iterations) and is probably related to *damped oscillations* in the Hamming distance as a function of the number of iterations (11). While it is not the purpose of this paper to investigate that behavior, we preferred to show it rather than show the much smoother plots at $4n$ iterations.

Qualitatively similar results holds for all other densities, see also the case of density 0.9 illustrated in the right plots of Fig. 4. In particular, the *rank inversion* for zero prohibition happens for all densities, and the knee point is in all cases close to the point corresponding to $T_f = 0.05$. The conclusion derived from the DB plots is that TS with the *oblivious* function dominates TS with the non-oblivious one, even if LS-NOB reaches better local optima. The conjecture that TS-OB will also dominate if the component is used in the more complex H-RTS algorithm and the best value found throughout the search is considered (and not the average f value at a checkpoint) will be tested in Section 5.

Very different results are shown in Fig. 5 for the *MAX–IND–SET* problem. Here the *rank inversion* for zero prohibition is not present (see the leftmost

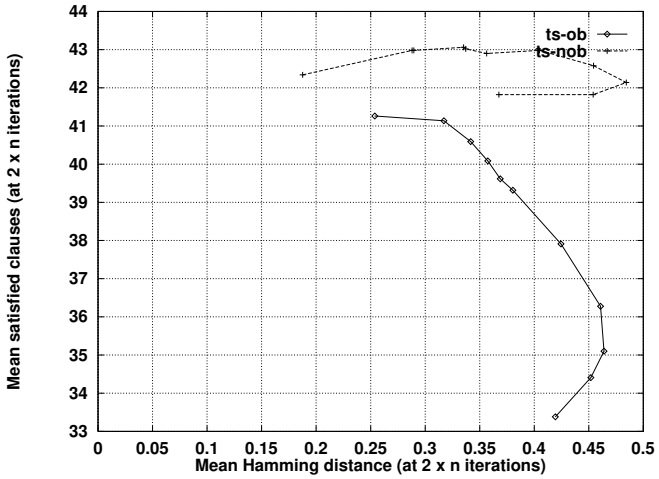

Fig. 5. *MAX–IND–SET* in cubic graphs: diversification–bias plot. Each run starts from LS local optimum. Graphs with 100 nodes. $T_f = 0.0, 0.01, 0.02, 0.03, 0.04, 0.05, 0.1, 0.2, 0.3, 0.4, 0.5$.

points on the DB plots of Fig. 5): LS-OB satisfies 38.0 clauses at the local optimum, **41.2** after $2n$ iterations, while LS-NOB satisfies 40.1 clauses at the local optimum, **42.3** after $2n$ iterations. The performance difference is reduced but the rank is maintained. This result is consistent with the fact that the diversification of NOB (18.7 bits at the checkpoint) is not much less than that of OB (25.3 bits).

The advantage of the NOB function is maintained when the prohibition is larger than zero: the points on the curve corresponding to FIXED-TS-NOB(T_f) dominate the points on the FIXED-TS-OB(T_f). In addition, let us note that FIXED-TS-NOB with very large prohibitions ($T_f = 0.4, 0.5$) is dominated by FIXED-TS-NOB with smaller prohibitions (where one obtains both larger diversification and larger number of satisfied clauses). The plateau at a value of about 43 satisfied clauses is reached by FIXED-TS-NOB for prohibitions between 0.01 and 0.1. Again, we have a prediction to be tested when the components are inserted into the H-RTS algorithm: in this case the predicted winner is TS with the *non-oblivious* function.

In passing, let us note that the above studies are very affordable: the CPU time to derive the above shown DB plots is only of a couple of minutes on a current personal computer.

4.2 Prohibition and diversification

The *reactive* (feedback) scheme in the H-RTS algorithm is added to the basic "Local Search with prohibition-based diversification" component in order to satisfy two requirements: i) automated tuning of the prohibition parameter ii) possibility of adapting the prohibition to the *local* characteristics of a given

instance. The algorithm monitors the behavior of the Hamming distance along the search trajectory (distance between a point and its successors) and determines a minimal value of T, in an heuristic manner, such that a sufficient degree of diversification is obtained.

The basic H-RTS cycle (see Fig. 2) is as follows: a local optimum is reached by LS (with no prohibitions), then $2(T+1)$ steps of TS are executed. In the first $(T+1)$ steps the Hamming distance from the local optimum increases up to $(T+1)$, what happens next is used to modify the T value, see the routine REACT in Fig. 1. In detail, the rate of increase of the Hamming distance in the last $(T+1)$ steps (called *deriv*) is used in the tuning process.

The underlying hypothesis is that there is a positive correlation between T_f and *deriv* for small T_f values, such that, if the diversification is not sufficient, a larger *deriv* can be obtained by increasing T_f.

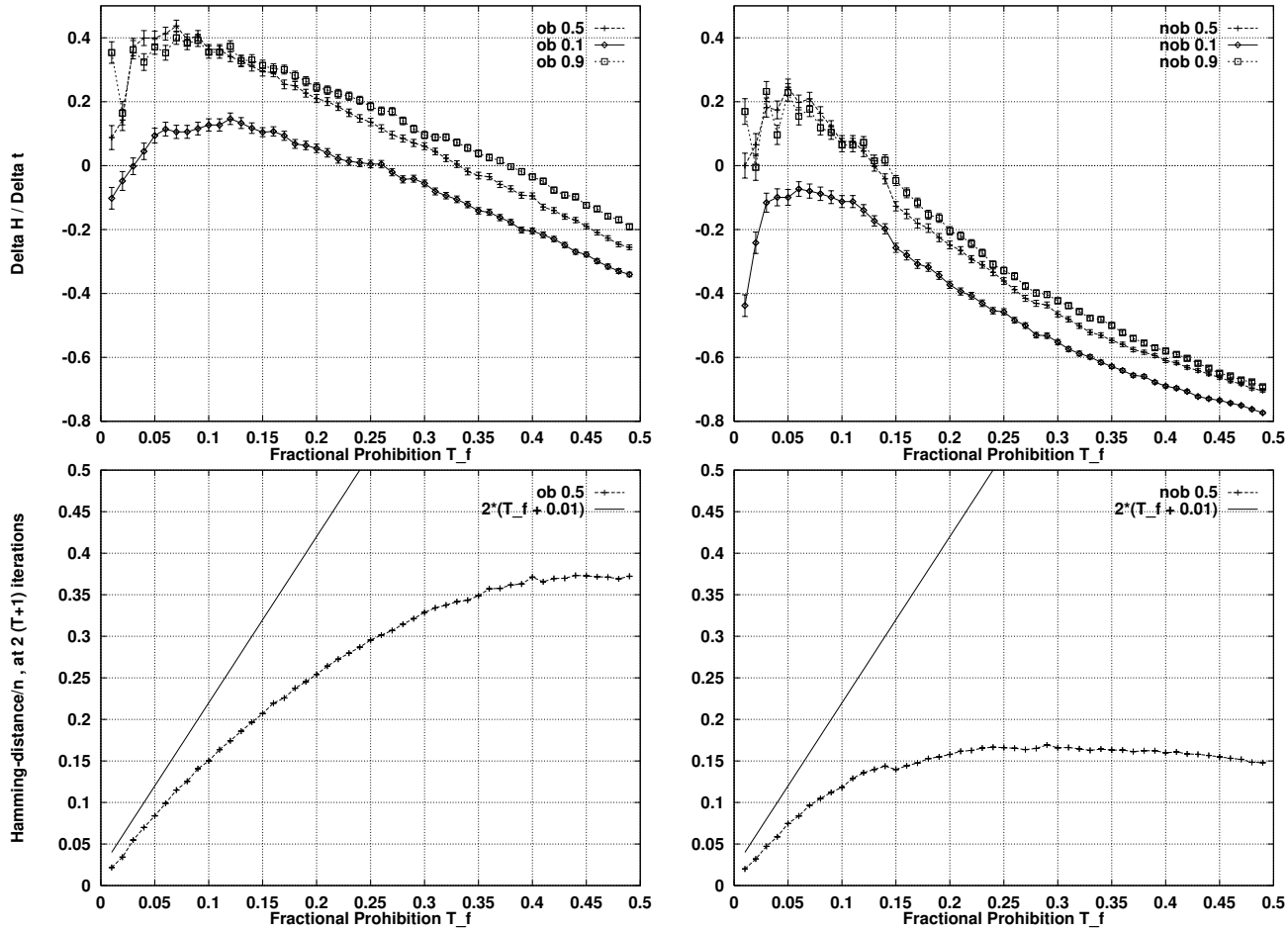

Fig. 6. *MAX–DICUT*: correlation between T_f and *deriv*: (top graphs). Hamming distance (normalized with respect to n) reached at $2(T+1)$ iterations versus T_f (bottom graphs). Oblivious search results (left), non-oblivious search results (right). Graphs with 100 nodes, densities $0.1, 0.5, 0.9$ (above), only 0.5 (below).

A systematic analysis of the average relationship between T_f and *deriv* is

executed in the following tests. The first local optimum found by LS after $10n$ LS$^+$ iterations is stored (a large number of iterations is executed to avoid "transient" effects in the initial part of the search). Then FIXED-TS(T_f) is run, and the *deriv* value is calculated. As usual, the data are averages of 10 runs for each of 50 different tasks.

Fig. 6 (top) reports the results for the *MAX–DICUT* problem, for three values of the graph densities (0.1, 0.5, and 0.9), and considering both the OB (left) and NOB (right) functions. The positive correlation region where *deriv* grows as a function of T_f extends up to $T_f \approx 0.1$ for the OB function, $T_f \approx 0.05$ for the NOB one. After that point a plateau follows and then gradually larger T_f values produce smaller and eventually negative *deriv* values. As expected from the diversification-bias results, NOB suffers from poor diversification: *deriv* does not reach values significantly larger than 0.2 and, for low-density graphs, it never reaches positive average values, see the curve for density 0.1 in the right part of Fig. 6 (top).

In order to have a clearer picture, and to consider also the contribution of the first $(T + 1)$ iterations, it is useful to analyze the data corresponding to the Hamming distance H reached at the end of the $2(T + 1)$ iterations. Given the definition of *deriv*, H is equal to $(T + 1)(1 + deriv)$, a value that is upper bounded by $2(T + 1)$. Fig. 6 (bottom) reports the final Hamming distance normalized with respect to n (H/n) in the two cases. For readability, only the results for graphs with density 0.5 are shown and the upper bound $2(T_f+0.01)$ is also reported with a continuous line. Again, the radical difference between the OB and NOB behavior can be noted: while the Hamming distance increases up to $0.37n$ with the OB function, it reaches a plateau of value close to $0.17n$ with the NOB function. The NOB plateau is reached when $T_f \approx 0.25$.

The results for the *MAX–IND–SET* problem shown in Fig. 7 are consistent with the better diversification expected for the NOB function from the DB results of Section 4.1. While the OB results show an initial positive correlation between *deriv* and T_f (Fig. 7 (top), left), up to a value of $T_f \approx 0.25$, the NOB results show a maximal value of *deriv* for the smallest possible prohibition ($T_f = 0.01$, corresponding to $T = 1$ for the case $n = 100$) and larger prohibitions tend to decrease the *deriv* values (Fig. 7 top, right). Furthermore, the final Hamming distance for the NOB function (Fig. 7 bottom right) is close to its upper bound, especially for small T_f values, and reaches values much larger than those reached for the *MAX–DICUT* problem.

The conclusion of this series of tests is that the region where the *deriv* value grows as a function of the prohibition is not always present. In fact, it is always present for the *MAX–DICUT* problem (with both the OB and NOB functions), see Fig. 6, it is present for the *MAX–IND–SET* problem with the OB function, but not when the NOB function is used, see Fig. 7. In addition,

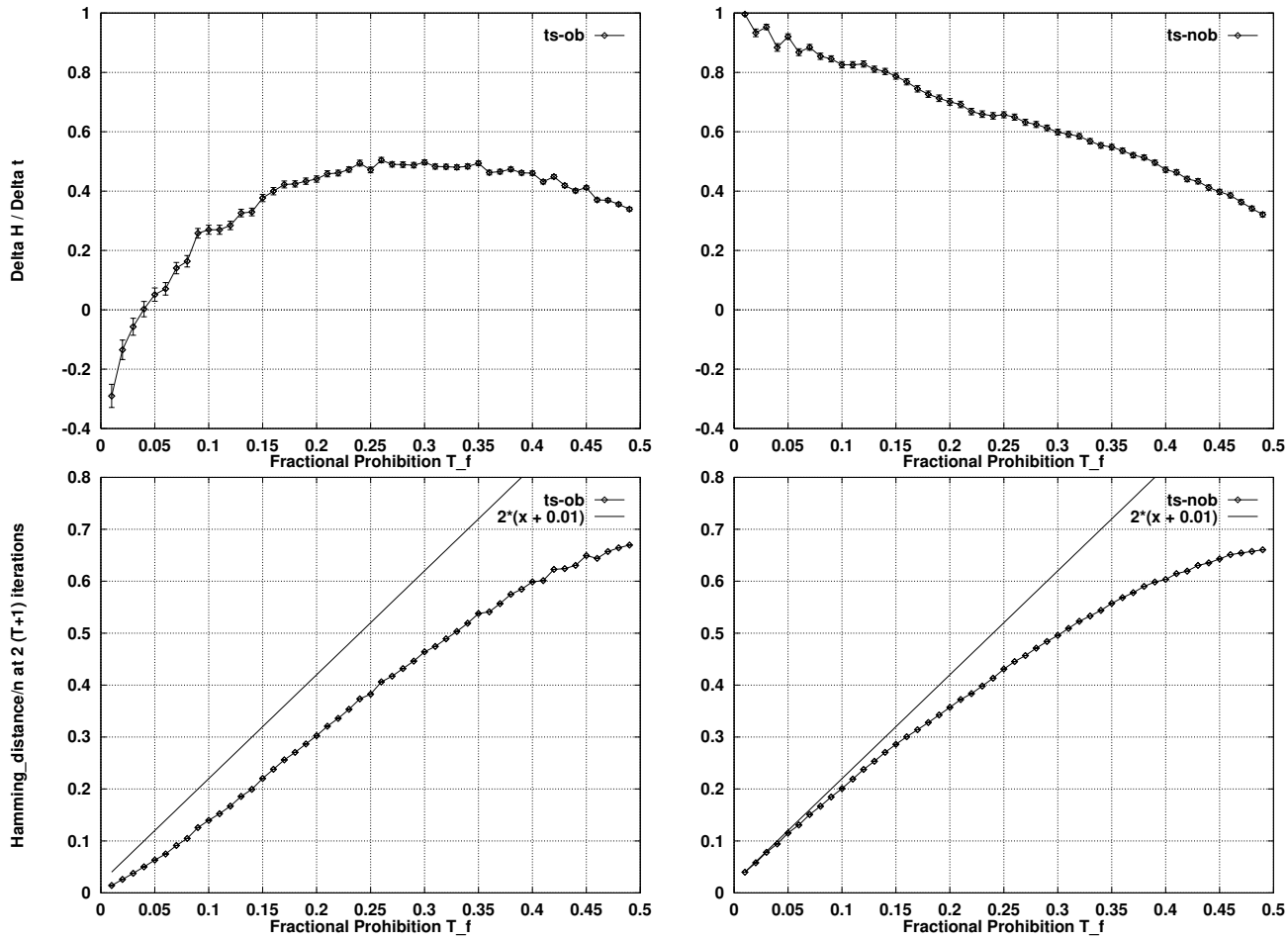

Fig. 7. *MAX–IND–SET* in cubic graphs: correlation between T_f and *deriv* (top graphs). Hamming distance (normalized with respect to n) reached at $2(T+1)$ iterations versus T_f (bottom graphs). Oblivious search results (left), non-oblivious search results (right).

for the *MAX–DICUT* problem, the upper bound of 1/4 for the fractional prohibition used in the reactive scheme (see Fig. 1) overestimates the range of the "positive correlation" region. A value of 1/10 for f_{OB} and 1/20 for f_{NOB} is a more appropriate limit that can be derived from the empirical evidence illustrated in Fig. 6.

5 Final experiments

From the preliminary investigation in Sec. 4.1 one derives the prediction that the oblivious function should reach better results for the *MAX–DICUT* problem, while the non-oblivious function should dominate for the *MAX–IND–SET* in cubic graphs problem. In this section the above prediction is validated by experiments with the H-RTS algorithm, where the performance is judged by

the best solution found during a long H-RTS search (up to $100n$ iterations).

In addition to the previously described algorithms, two versions of the Simulated Annealing (SA) algorithm (27) are considered in the experiments, where OB or NOB functions are used for the acceptance test. The initial "temperature" (27) of the SA runs is equal to twice the maximum $|\Delta f|$ between the current point and the neighbors (so that most moves are accepted in the initial phase). The temperature is then multiplied by 0.9995 after each iteration. Other parameter values have been tested (multiplicative factors between 0.999 and 0.9999, more iterations for a given temperature), without obtaining significantly better results.

As usual, 10 runs for each of the 50 tasks are executed, and the best value found is checked at regular intervals during the runs, and then averaged over all tasks and all runs. Because the CPU time is proportional to the number of iterations with a similar coefficient for all considered algorithms, all results are described in terms of iterations. Furthermore, the number of iterations is clearly independent of the particular implementation, machine, and operating system.

5.1 MAX–DICUT

Table 5 lists the results obtained on the *MAX–DICUT* tasks of dimension 100. Six algorithms are compared: REP-LS-OB is the repeated version of LS-OB (as soon as a local optimum is found, the search is restarted with a new random configuration), and REP-LS-NOB is the repeated version of LS-NOB. H-RTS-OB and H-RTS-NOB are two versions of the H-RTS algorithm illustrated in Fig. 2, obtained by using the oblivious or non-oblivious function during the TS phase (f_{OB} or f_{NOB} substitutes f in lines 11 and 16 of Fig. 2).

In addition, SA-OB and SA-NOB are two versions of the Simulated Annealing algorithm, where the OB or NOB functions are used for the acceptance test.

The same runs are then illustrated graphically in Fig. 8, that shows the average value of the "best so far solution" as a function of the number of elapsed iterations (for graphs of density 0.5).

If one compares the OB versus the NOB results (for example: REP-LS-OB versus REP-LS-NOB, or H-RTS-OB versus H-RTS-NOB, SA-OB versus SA-NOB) a clear superiority of the oblivious function can be observed. Furthermore, H-RTS-OB reaches a large number of satisfied clauses after a very short phase. For example, 1443.2 satisfied clauses are reached at 200 itera-

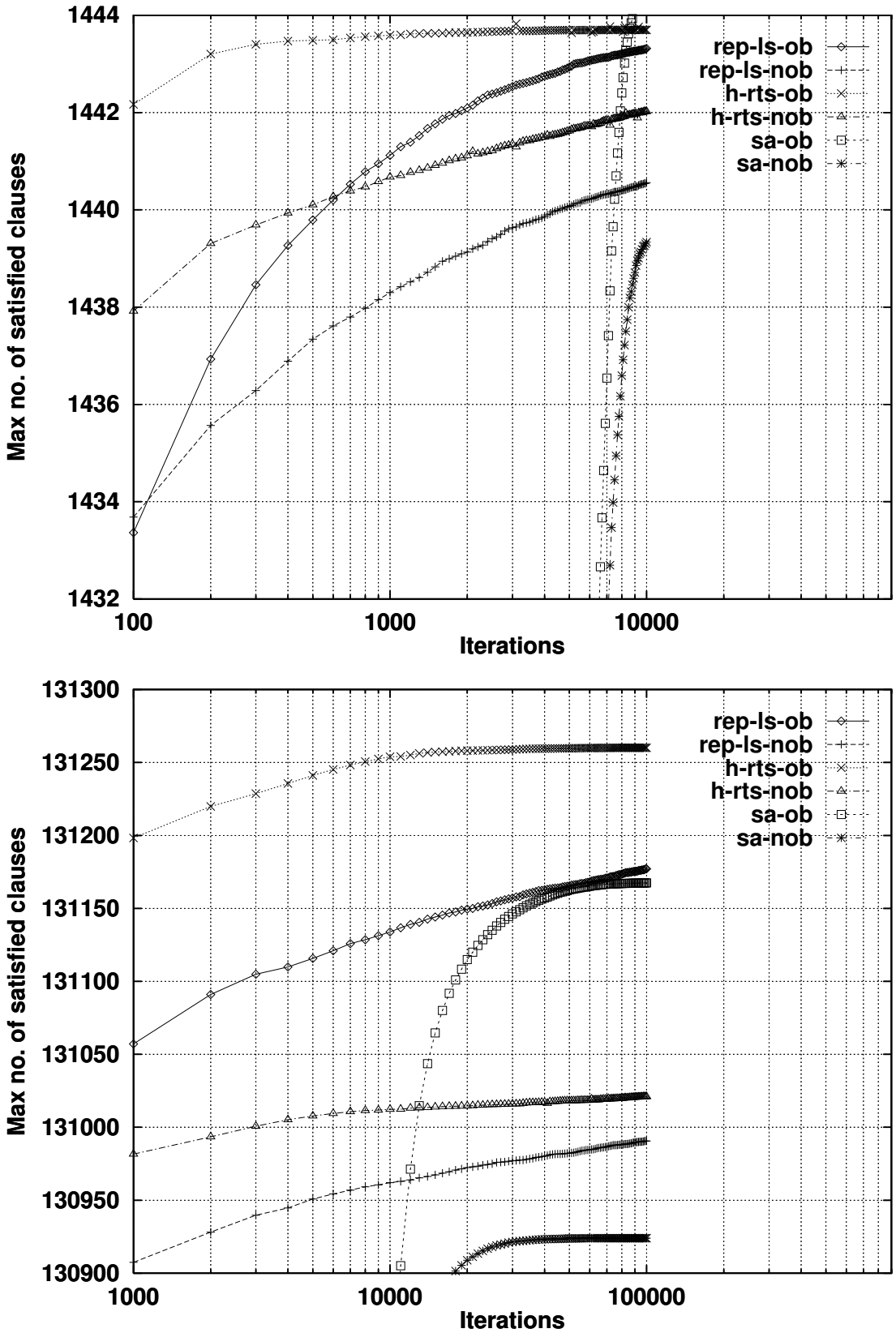

Fig. 8. *MAX–DICUT*. Density 0.5, graphs with 100 nodes (top) and 1000 nodes (bottom).

variables	100	100	100	100	10
density	0.1	0.3	0.5	0.7	0.9
REP-LS-OB ($10n$)	371.1 (11.0)	929.1 (12.6)	1441.1 (17.1)	1924.2 (16.8)	2355.7428
REP-LS-NOB ($10n$)	370.5 (11.4)	925.8 (12.9)	1438.2 (17.0)	1920.9 (16.9)	2356.9
H-RTS-OB ($10n$)	**372.9** (10.7)	**930.8** (12.4)	**1443.6** (16.8)	**1927.2** (17.0)	**2361.8**
H-RTS-NOB ($10n$)	371.5 (11.2)	927.7 (12.8)	1440.6 (14.1)	1923.2 (17.1)	2359.4
SA-OB ($10n$)	295.3 (10.7)	816.7 (15.8)	1320.8 (16.7)	1811.9 (16.1)	2286.4
SA-NOB ($10n$)	295.5 (11.3)	817.6 (15.7)	1322.2 (16.7)	1812.0 (15.5)	2288.5
REP-LS-OB ($100n$)	372.6 (10.8)	930.6 (12.3)	1443.3 (16.7)	1926.5 (17.0)	2359.0
REP-LS-NOB ($100n$)	371.5 (11.2)	927.6 (12.5)	1440.5 (16.8)	1923.2 (16.8)	2358.8
H-RTS-OB ($100n$)	**373.0** (10.7)	**930.9** (12.4)	1443.7 (16.7)	**1927.3** (17.0)	**2362.0**
H-RTS-NOB ($100n$)	372.1 (11.1)	928.7 (12.7)	1442.0 (16.6)	1925.0 (16.9)	2360.3
SA-OB ($100n$)	370.5 (11.0)	930.2 (13.4)	**1444.6** (14.4)	1925.7 (12.7)	2359.7
SA-NOB ($100n$)	367.9 (11.5)	926.1 (14.2)	1439.3 (14.4)	1920.7 (12.2)	2356.2

Table 5
MAX–DICUT: Mean number of satisfied clauses ("best so far value") with standard deviation after $10n$ and $100n$ iterations.

tions, while REP-LS-OB reaches 1443.0 clauses only at iteration 5300. For that level of performance, H-RTS-OB is therefore about 26.5 faster than REP-LS-OB. The Simulated Annealing algorithm reaches very poor results in the initial part of the search, although SA-OB does eventually (at 100 n iterations) reach a performance that is not too far from that of H-RTS-OB (only in the case of density 0.5 SA performs slightly better).

Finally, the performance of the same algorithm is tested in the same experimental conditions for larger graphs (of dimension 1000 and density 0.5), see Fig. 8 (bottom). The same relative ranking of the different algorithms is obtained. The performance of SA-OB is now much worse and it is beaten even by the simple REP-LS-OB scheme.

5.2 MAX–IND–SET

Table 6 lists the results obtained on the *MAX–IND–SET* tasks of dimension 100. As it was the case for the *MAX–DICUT* tasks, the runs are then illustrated graphically in Fig. 9, that shows the average behavior of the "best so far solution."

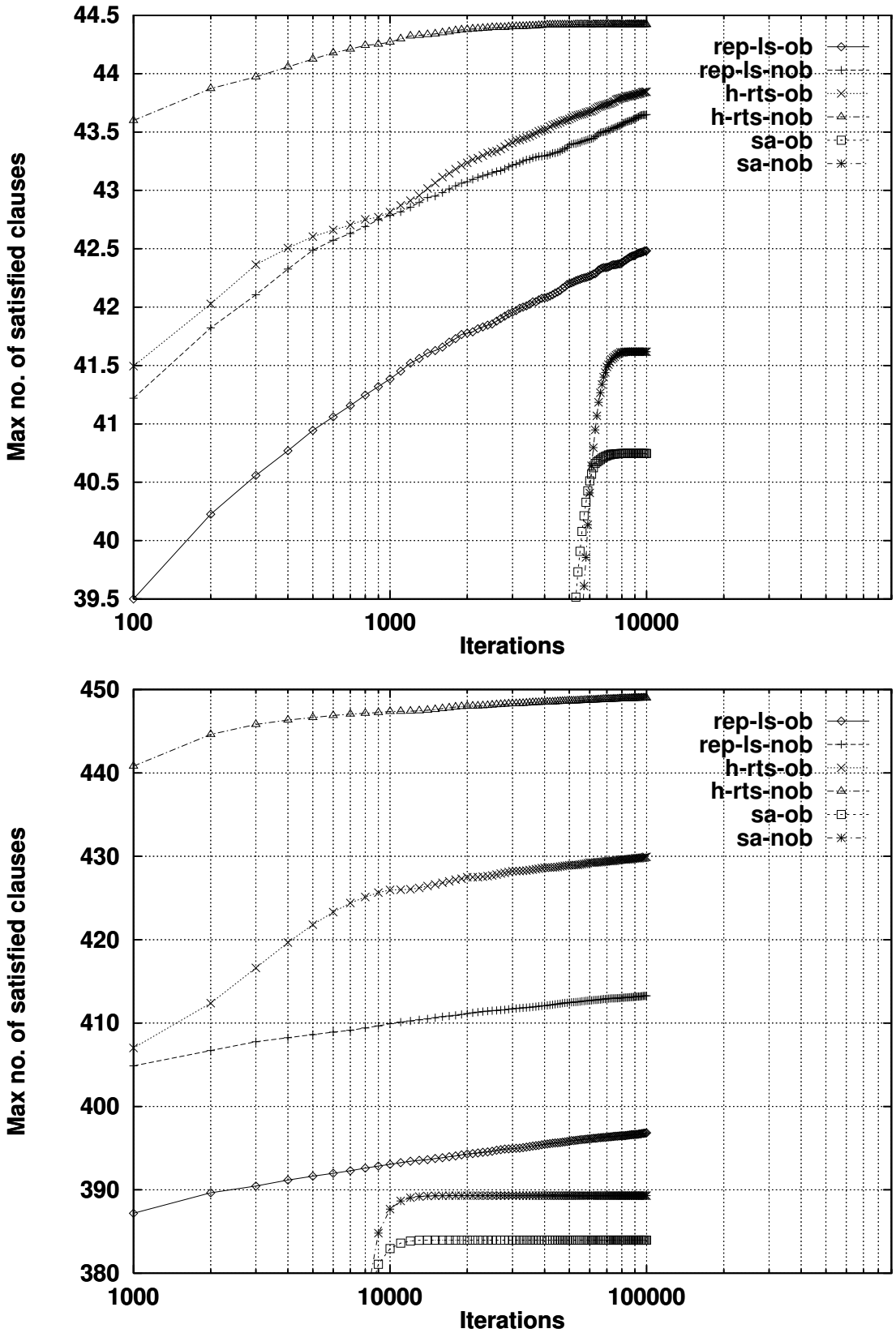

Fig. 9. *MAX–IND–SET* in cubic graphs. Graphs with 100 nodes (top) and 1000 nodes (bottom).

variables	100
REP-LS-OB ($10n$)	41.3 (0.7)
REP-LS-NOB ($10n$)	42.7 (0.6)
H-RTS-OB ($10n$)	42.8 (0.8)
H-RTS-NOB ($10n$)	**44.2** (0.5)
SA-OB ($10n$)	17.7 (2.1)
SA-NOB ($10n$)	18.0 (2.2)
REP-LS-OB ($100n$)	42.4 (0.6)
REP-LS-NOB ($100n$)	43.6 (0.6)
H-RTS-OB ($100n$)	43.8 (0.6)
H-RTS-NOB ($100n$)	**44.4** (0.5)
SA-OB ($100n$)	40.7 (1.3)
SA-NOB ($100n$)	41.6 (1.1)

Table 6
MAX–IND–SET in cubic graphs: Mean number of satisfied clauses ("best so far value") with standard deviation after $10n$ and $100n$ iterations.

As expected from the DB results, the situation is now different and a clear superiority of the *non-oblivious* function can be observed, and the H-RTS-NOB algorithm clearly appears as the most effective and efficient algorithm: very good solutions are found in the early phase of the search, and the alternative algorithms do not reach a comparable performance even if they are run for a much larger number of iterations (for example 10,000 iterations are needed by REP-LS-NOB to reach the same number of satisfied clauses reached by H-RTS-NOB after 100 iterations, Fig. 9 top).

The tests for larger graphs of dimension 1000 are shown in Fig. 9 (bottom) and confirm the results obtained on the smaller graphs. In particular, H-RTS-NOB reaches 440 satisfied clauses after 1000 iterations and almost 450 clauses after 100,000, while the OB version H-RTS-OB reaches only 430 at the end of the run and the other competitors do not reach more than 413 clauses (obtained by REP-LS-NOB). It is also to be noted that Simulated Annealing is beaten by the simple repeated Local Search algorithms, confirming the theoretical predictions (15).

6 Conclusions

The initial purpose of our paper was to test the H-RTS algorithm proposed for *MAX–SAT* on different problems where the clauses are *conjunctions* of variables.

In the preliminary investigation about the diversification–bias characteristics of simple components based on Local Search with prohibitions a rich variety of behaviors has been observed. In particular, the most remarkable difference with respect to the disjunctive *MAX–SAT* case has been found for the *MAX–IND–SET* problem in cubic graphs. For this problem the *non-oblivious* function, while confirming the superior average quality of its local minima, is also superior when used for longer runs with prohibition-based diversification. This result is caused by the coupling of better bias *and* larger diversification obtained by NOB Local Search with reactive prohibition.

Our results confirm the effectiveness of the H-RTS algorithm with the proper (oblivious or non-oblivious) function to find in a very fast way approximations that are close to the results obtained at the end of long searches, and superior to those reachable by repeated Local Search algorithms and Markov processes (SA). In addition, the detailed findings about the behavior of the Hamming distance when the prohibition is varied suggests more precise reactive schemes, that are not being investigated in the current paper in order to limit its size.

Acknowledgements

We wish to thank P. Alimonti and G. P. Leonardi for their useful comments on a preliminary version of this paper.

References

[1] P. Alimonti, "Non-oblivious local search for graph and hypergraph coloring problems," *21st International Workshop on Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science 1017, Springer Verlag, 1995.

[2] P. Alimonti, " New local search approximation techniques for maximum generalized satisfiability problems," *Information Processing Letters* **57** (1996), 151–158.

[3] P. Alimonti and V. Kann, "Hardness of Approximating Problems on Cubic Graphs," *Proc. 3rd Italian Conference on Algorithms and Com-*

plexity (CIAC-97), Lecture Notes in Computer Science. Springer-Verlag 1997.

[4] T. Asano, "Approximation algorithms for MAX-SAT: Yannakakis vs. Goemans-Williamson," *Proc. 3rd Israel Symp. on the Theory of Computing and Systems, Ramat Gan, Israel*, 1997, 24–37.

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

[6] G.Ausiello and M.Protasi, "Local search, reducibility and approximability of NP optimization problems," *Information Processing Letters* **54** (1995), 73–79.

[7] R. Battiti and M. Protasi, "Reactive Search, a history-sensitive heuristic for MAX-SAT," *ACM Journal of Experimental Algorithmics*, **2** (1997), Paper 2 (electronic journal at `http://www.jea.acm.org/`).

[8] R. Battiti and M. Protasi, "Solving MAX–SAT with non-oblivious functions and history-based heuristics," In: Satisfiability Problem: Theory and Applications, D. Du, J. Gu and P. M. Pardalos, Eds., DIMACS: Series in Discrete Mathematics and Theoretical Computer Science no. 35, AMS/ACM 1997, in press.

[9] R. Battiti and M. Protasi, "Reactive local search for Maximum Clique," *Proc. of the Workshop on Algorithm Engineering (WAE'97), Venice*, G. F. Italiano and S. Orlando, Eds., 1997, 74–82.

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

[11] R. Battiti and G. Tecchiolli, "Tabu Random Walk: Transient and Asymptotic Behavior". Conference of the Institute for Operations Research and the Management Sciences (INFORMS), New Orleans, 1995.

[12] P. Crescenzi and V. Kann, "A compendium of NP optimization problems," available at `http://www.nada.kth.se/~viggo/problemlist/compendium.html` .

[13] P. Erdos and A. Renyi, "On random graph I," *Publ. Math. Debrecen* **6** (1959), 290–297.

[14] U. Feige and M. X. Goemans, "Approximating the Value of Two Prover Proof Systems, with Applications to MAX 2SAT and MAX DICUT," *Proc. of 3rd Israel Symp. on Theory of Computing and Systems*, IEEE Computer Society, 1995.

[15] A.G. Ferreira and J. Zerovnik, "Bounding the probability of success of stochastic methods for global optimization," *Computers and Mathematics with Applications*, **25** (1993), 1–8.

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

[17] M. X. Goemans and D.P. Williamson, "New $\frac{3}{4}$-approximation algorithms for the maximum satisfiability problem," *SIAM Journal on Discrete Mathematics*, **7(4)** (1994), 656–666.

[18] M.X. Goemans and D.P. Williamson, "Improved approximation algo-

rithms for maximum cut and satisfiability problems using semidefinite programming," *Journal of the ACM*, **42(6)** (1995), 1115–1145.

[19] R. Greenlaw and R. Petreschi, "Cubic graphs," *ACM Computing Surveys* **27** (1995), 471–495.

[20] J. Gu, P. W. Purdom, J. Franco, and B.W. Wah, "Algorithms for the Satisfiability (SAT) Problem: a Survey," Preliminary version, 1996. In: *Satisfiability Problem: Theory and Applications*, D.-Z. Du, J. Gu and P.M. Pardalos (Eds.), DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 35, AMS/ACM 1997.

[21] J. Håstad, "Clique is hard to approximate within $n^{1-\epsilon}$," *Proc. 37th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1996, 627–636.

[22] J. J. Hopfield and D. W. Tank, "Neural computation of decisions in optimization problems," *Biological Cybernetics* **52** (1985), 141–152.

[23] D.S. Johnson, C. H. Papadimitriou, and M. Yannakakis, "How easy is local search ?," *Journal of Computer and System Sciences* **37** (1988), 79–100.

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

[25] H. Karloff and U. Zwick, "A 7/8-Approximation Algorithm for MAX 3SAT?," *Proc. of the 38th Annual Symposium on Foundations of Computer Science*, IEEE Computer Society, 1997.

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

[27] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science* **220** (1983), 671–680.

[28] C.H. Papadimitriou and M. Yannakakis, "Optimization, approximation and complexity classes," *Journal of Computer and System Sciences* **43** (1991) 425–440.

[29] L. Trevisan, "Positive linear programming, parallel approximation, and PCP's" *Proc. 4th European Symposium on Algorithms*, 1996, 62–75.

[30] L. Trevisan, "Approximating Satisfiable Satisfiability Problems," *Proc. 5th European Symposium on Algorithms*, 1997, 472–485.

[31] M. Yannakakis, "On the approximation of maximum satisfiability," *Journal of Algorithms*, **17** (1994), 475–502.