Available online at www.sciencedirect.com
SCIENCE DIRECT®
European Journal of Operational Research 166 (2005) 63–76
EUROPEAN JOURNAL OF OPERATIONAL RESEARCH
www.elsevier.com/locate/dsw

Maximum satisfiability: How good are tabu search and plateau moves in the worst-case?

Monaldo Mastrolilli *, Luca Maria Gambardella

IDSIA—Istituto Dalle Molle di Studi sull'Intelligenza Artificiale, Strada Cantonale Galleria 2, 6928 Manno, Switzerland

Received 29 October 2002; accepted 22 January 2003
Available online 11 August 2004

Abstract

Tabu search algorithms are amongst the most successful local search based methods for the maximum satisfiability problem. The practical superiority of tabu search over local search alone has been already shown experimentally several times. A natural question addressed here is to understand if this superiority holds also from the worst-case point of view. Moreover, it is well known that a critical parameter of tabu techniques is the tabu list length. Focussing on MAX-2-SAT problem, the main contribution of this paper is a worst-case analysis of tabu search as a function of the tabu list length. We give the first theoretical evidence of the advantage of a tabu search strategy over the basic local search alone that critically depends on the tabu list length. Our second contribution is about a widespread belief that when a local optimal solution is reached then simply continuing to search by making non-deteriorating, "plateau" moves, dramatically increases the success rate of local search based algorithms. We start by observing that no plateau search strategy can help to improve the approximation ratio of the basic local search. However, for a restricted version of the problem, we prove that plateau moves can enhance the approximation ratio of the basic local search from 1/2 to 2/3.

Keywords: Maximum satisfiability; Tabu search; Local search; Worst-case analysis

1. Introduction

In the maximum satisfiability problem (MAX-SAT) we are given a boolean formula in conjunctive normal form, i.e., as a conjunction of clauses, each clause being a disjunction. More formally, n is the number of variables and m the number of clauses, so that a formula has the following form:

* Corresponding author. Tel.: +41 91 610 86 64; fax: +41 91 610 86 61.
E-mail addresses: monaldo@idsia.ch (M. Mastrolilli), luca@idsia.ch (L.M. Gambardella).

0377-2217/$ - see front matter © 2004 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2003.01.005

$$\bigwedge_{1 \leqslant i \leqslant m} \left(\bigvee_{1 \leqslant h \leqslant |C_i|} l_{ih} \right),$$

where $|C_i|$ is the number of literals in clause C_i and l_{ih} is a literal, i.e., a propositional variable v_j, or its negation $\overline{v}_j$, for $1 \leqslant j \leqslant n$. The set of clauses in the formula is denoted by $\mathbb{C} = \{C_1, C_2, \ldots, C_m\}$. If one associates a positive weight w_i to each clause C_i, one obtains the *weighted* MAX-SAT problem. We are asked to find an assignment of values to the variables that maximizes the (weighted) sum of the satisfied clauses.

The interest for MAX-SAT stems from many reasons. On one hand, the decision version SAT was the first example of an NP-complete problem; moreover, MAX-SAT and related variants play an important role in the characterization of different approximation classes like APX and PTAS [2]. On the other hand, many issues in mathematical logic and artificial intelligence can be expressed in the form of satisfiability or some of its variants, like constraint satisfaction.

Local search strategies. Amongst the many different approaches proposed to deal with this problem (see [4] for a survey), local search based algorithms play a fundamental role. Local search employs the idea that a given solution may be improved by making "small" changes: starting from some initial solution, move from neighbor to neighbor as long as possible while increasing the objective function value. Most local search based algorithms for MAX-SAT use a 1-*flip* neighborhood relation for which two truth value assignments are neighbors if they differ in the truth value of exactly one variable. A basic local search (*LS*) starts with any given assignment, and then repeatedly changes ("flips") the assignment of a variable that leads to the largest decrease in the total number of unsatisfied clauses. A local optimal solution is defined as a state whose local neighborhood does not include a state that is strictly better. Let OPT_{loc} (and OPT) be the number of satisfied clauses at a local (global) optimum of any instance of MAX-SAT with at least k literals per clause. Then it is known [13] that

$$OPT_{\text{loc}} \geqslant \frac{k}{k+1} OPT. \tag{1}$$

Therefore the described basic local search algorithm is (in the general case $k=1$) a 1/2-approximate algorithm for MAX-SAT.

The idea is simple and natural and it is surprising to see how successful the use of this local search has been on a variety of difficult instances. Moreover, the combination of local optimization and other mechanisms for escaping from local optimal solutions would seem to give them some advantage over either alone. MAX-SAT is among the problems for which many local search based heuristics (like GSAT, tabu search, simulated annealing, etc.) have been proved (experimentally) to be very effective (see the review in [4] and [14,18,21,22] for more recent results). These algorithms have been widely used and tested. However, the efficiency and effectiveness have never been analyzed analytically. In particular, nothing is known about their performance guarantees.

Aim of the paper. The aim of this paper is not to improve algorithms that achieve the best known approximation ratios [1] for MAX-SAT (for this see [1,11,12,23]), but to provide a first worst-case analysis of some local search strategies that achieve very good results in 'practice'. The present work was motivated by the current state of the MAX-SAT local search algorithms, where a considerable gap exists between successful applications of heuristics and the theoretical analysis of these various heuristic methods. The focus of this paper is twofold, as explained in the following. We remark that our analysis assumes that the initial starting solution is arbitrary (given by an adversary).

Analysis of tabu search for MAX-2-SAT. A fundamental approach for escaping from local optima is to use aspects of the search history. Tabu search (TS) is a general local search method which systematically

[1] The quality of an approximation algorithm is measured by its so called *performance ratio*. An approximation algorithm for MAX-SAT has a performance ratio of ρ, if for all input instances it finds a solution of value at least ρ times the optimal solution value.

utilizes memory for guiding the search process [8–10]. In the simplest and most widely applied version of TS, a greedy local search algorithm is enhanced with a form of short term memory which enables it to escape from local optima. Overall, TS algorithms are amongst the most successful local search based methods to date. MAX-SAT was one of the first application problems to which TS algorithms were applied. In fact, in one of the papers in which TS algorithms were first proposed (see [13]), the target application was MAX-SAT.

Mazure et al. [17] considered a simple tabu search algorithm, TSAT, for satisfiability problems. TSAT makes a systematic use of a tabu list of variables in order to avoid recurrent flips and thus escape from local optima. The tabu list is updated each time a flip is made. TSAT keeps a fixed length-chronologically-ordered FIFO list of flipped variables and prevents any of the variables in the list from being flipped again during a given amount of time. TSAT was compared with Selman et al. [19] Random Walk Strategy GSAT, an important extension of the basic GSAT and amongst the best performing algorithms. TSAT proves extremely competitive in the resolution of many problems, in particular hard random instances at the critical point of the phase transition. In this empirical study, Mazure et al. [17] found that the optimal length of the tabu list is crucial to the algorithm's performance. Interestingly, they observed that the optimal length of the tabu lists for these random problems proves (experimentally) linear with respect to the number of variables.

In this paper we consider the MAX-2-SAT problem, i.e. the restricted version of MAX-SAT in which each clause contains two literals. This problem is known to be NP-Hard [7]. We analyze a tabu search algorithm, $TS(\ell)$, which has the same flavor of TSAT (and of many other tabu search implementations), and whose tabu list length is bounded by the parameter ℓ (see Section 2). The superiority of TS with respect to the local search alone has already been shown experimentally several times. It is well known that the main critical parameter of tabu techniques is the tabu list length. Here the main goal is to understand if there are values of ℓ for which $TS(\ell)$ is better than the basic local search even from the worst-case point of view. We show that the approximation ratio for $TS(\ell)$ is 2/3 for any tabu list length sublinear in the number of variables (see Section 2.1). This means that, in the worst-case, and for any $\ell = o(n)$, $TS(\ell)$ is not better than the basic local search (see (1) when $k = 2$). However we prove a strong separation in the performance of the basic local search and $TS(\ell)$ when the tabu list length is allowed to increase up to n. Indeed we show that $TS(n)$ achieves an approximation ratio of 3/4, whereas the basic local search ensures an approximation ratio of 2/3 for MAX-2-SAT (see Section 2.2). To some extent, these results confirm the linearity in n of the optimal length of the tabu lists, as it was observed experimentally in [17]. Moreover, we give a first theoretical evidence of the advantage of a tabu search strategy over the basic local search alone even from the worst-case point of view.

Analysis of plateau moves. It is widely believed that the approach of terminating the search when a local optimal solution is reached does not work well for MAX-SAT. Selman et al. [20] showed empirically that simply continuing to search by making non-deteriorating, random "sideways" moves, dramatically increases the success rate of their algorithm. They proposed an amazingly simple and efficient greedy local search procedure, i.e., GSAT, allowing for a breakthrough in the class of computer-solvable SAT instances. When GSAT becomes stuck at a local optimal solution, then a long sequence of sideways moves follows. They refer to each sequence of sideways moves as *plateau steps*, i.e., 1-flip local search steps which do not lead to a change in evaluation function value [2]. The success of GSAT is determined by its ability to move between successively lower plateaus. Although this is now a common feature of the state-of-the-art algorithms for MAX-SAT [4], this "augmented" local search algorithm has never been analyzed theoretically.

[2] The plateau steps of GSAT can be seen as a simple instance of a probabilistic tabu search method with a zero length tabu list, see [9, pp. 200–203].

Moreover no specific strategy to search along plateau regions has been proven to give an advantage over the basic local search alone in the worst-case.

In Section 3 we start observing that no plateau search strategy (based on 1-flip neighborhood, like GSAT [20]) can help to improve the approximation ratio of the basic local search algorithm. However, for a restricted version of the problem (i.e. the unweighted MAX-SAT instances in which there are no copies of the same clause), we prove that plateau moves can enhance the approximation ratio of the basic local search from 1/2 to 2/3.

2. Analysis of tabu search for MAX-2-SAT

In this section we provide a worst-case analysis of a tabu search procedure $TS(\ell)$ (see the description below) whose tabu list length is bounded by the parameter ℓ. We derive an upper bound on the approximation ratio of $TS(\ell)$ as a function of ℓ. In particular we show that the approximation ratio of $TS(\ell)$ is 2/3 for any tabu list length that is sublinear in the number of variables. Our analysis implies that $TS(\ell)$ is not better than the basic local search for $\ell = o(n)$. However, we show that when the tabu list length is allowed to increase up to n the tabu search procedure achieves an approximation ratio of 3/4.

Tabu search description. The considered tabu search algorithm $TS(\ell)$ is as follows (see also Fig. 1). The search starts with any given variable assignment and with the tabu list length TT (i.e., the *tabu tenure*) equal to the parameter ℓ. Each search step corresponds to a single variable flip, which is selected according to the associated change in the number of unsatisfied clauses and its tabu status. More precisely, let $N(\vec{X})$ denote the set of variables that appear in some unsatisfied clauses according to the current variable assignment $\vec{X}$. Let $LastUsed(x_i)$ denote the last time the truth assignment of x_i was reversed ($LastUsed(x_i) = -\infty$ at the beginning). At step k, a variable x_i is considered *admissible* if x_i appears in some unsatisfied clause (i.e., $x_i \in N(\vec{X})$) and either

1: Start with any given truth assignment $\overrightarrow{X}$;
2: $k \leftarrow 0$, $TT \leftarrow \ell$, $LastUsed(x_i) \leftarrow -\infty$, for $i = 1 \ldots n$;
3: **while** a termination condition is not met **do**
4: $AdmissibleSet \leftarrow \{x_i \in N(\overrightarrow{X}) : LastUsed(x_i) < (k - TT)$ or x_i satisfies the *aspiration criterion*$\}$;
5: **if** $AdmissibleSet \neq \emptyset$ **then**
6: $u \leftarrow$ ChooseBestOf($AdmissibleSet$);
7: **else**
8: Determine variable u by using a *dynamic tabu policy*;
9: **end if**
10: $\overrightarrow{X} \leftarrow \overrightarrow{X}$ with the truth assignment of u reversed;
11: $LastUsed(u) \leftarrow k$;
12: $k \leftarrow k + 1$;
13: **end while**.

Fig. 1. Tabu search algorithm $TS(\ell)$ with tabu tenure at most ℓ.

(a) x_i has not been flipped during the last TT steps ($LastUsed(x_i) < (k - TT)$),
(b) or, when flipped, leads to a lower number of unsatisfied clauses than the best assignment found so far (the latter condition defines the so called *aspiration criterion* [9]).

From the set of admissible variables (if not empty), one variable among those that when flipped yield a maximal decrease (or, equivalently, a minimal increase) in the number of unsatisfied clauses, is selected (ties are broken arbitrarily, or even randomly).

If there is no admissible variable, the analysis provided in this paper applies to the following (and possibly to others) alternative *dynamic tabu policies*:

- (*self-adapting tabu list length*) decrease the tabu tenure TT until there is a non-tabu variable that appears in some unsatisfied clause (add this command at line 8 of Fig. 1). Each time a variable is flipped set $TT \leftarrow \max\{TT + 1, \ell\}$ (add this command just after line 11 of Fig. 1);
- (*least recently used*) flip the least recently used variable that appears in some unsatisfied clause (add this command at line 8 of Fig. 1).

Observe that in both strategies the tabu tenure of a variable is at most ℓ, but it can be less than ℓ (dynamic tabu policy). We refer to ℓ as the *maximum tabu tenure*.

The algorithm stops when a termination condition is met. For our purposes it is sufficient to assume that the tabu search procedure stops if for n consecutive steps there are no improvements.

Before providing the analysis of algorithm $TS(\ell)$ we make the following considerations about the dynamic tabu policies. One of the main purposes of a tabu list is to prevent from returning to recently visited solutions, and therefore from endless cycling. A list that is too short may not prevent cycling (see the proof of Lemma 8 for an example of this), but a list that is too long may create excessive restrictions. It is often difficult (or even impossible) to find a value that prevents cycling and does not excessively restrict the search for all instances of a given size. In Section 2.1 we prove that the useful values of ℓ are linear in n, and for these large tabu tenures it may easily happen that all the variables that appear in some unsatisfied clauses are tabu. An effective way of circumventing this difficulty is to have a tabu tenure that can be varied during the search, i.e. a dynamic tabu policy. Evidence has shown that dynamic tabu policies are generally more effective. Interesting examples can be found in [3,5,6,16]. More advanced ways to create dynamic tabu tenure are described in [10].

2.1. An upper bound on the approximation ratio

Let $c = \frac{n}{\ell}$ be the ratio between the number of variables n and the maximum tabu tenure ℓ. In this subsection we study the approximation ratio of $TS(\ell)$ as a function of c. In particular we provide the following upper bound on the approximation ratio of $TS(\ell)$.

Theorem 1. *Starting from any arbitrary initial solution and for $c \geqslant 3$, the approximation ratio of $TS(\ell)$ for MAX-2-SAT is bounded from above by*

$$R(c) = \frac{2c^2}{3c^2 - 2c + 2}. \tag{2}$$

Proof. We prove the theorem by showing that there exists a class of input instances for which $TS(\ell)$ achieves an approximation ratio that asymptotically converges to $R(c)$. With this aim, we borrow and modify an input instance for MAX-2-SAT appeared in [15]. We assume that $n \geqslant 2\ell + 5$. The input S consists of a disjoint union of four sets of clauses, say S_1, S_2, S_3 and S_4, defined as follows:

$$S_1 = \bigcup\nolimits_{1 \leqslant i < j \leqslant n} (x_i \vee \overline{x}_j),$$
$$S_2 = \bigcup\nolimits_{1 \leqslant i < j \leqslant n} (\overline{x}_i \vee x_j),$$
$$S_3 = \left(\bigcup_{0 \leqslant i \leqslant \ell+2} T_{2i+1} \right) - \{(\overline{x}_{2\ell+3} \vee \overline{x}_n)\},$$
$$S_4 = \bigcup\nolimits_{2\ell+6 \leqslant i \leqslant n} T_i,$$
$$T_i = \bigcup\nolimits_{i < j \leqslant n} (\overline{x}_i \vee \overline{x}_j).$$

Note that set S_3 is obtained by eliminating clause $(\overline{x}_{2\ell+3} \vee \overline{x}_n)$. Clearly, $|S_1| = |S_2| = \binom{n}{2}$, and $|S_3| + |S_4| = \binom{n}{2} - n(\ell+2) + (\ell+2)(\ell+3) - 1$. Assume that $\vec{X}$ is the starting assignment for which all variables are set to 1 except for x_{2i-1} $(i = 1, \ldots, \ell+1)$ whose value is 0. The number of clauses from $S_1 \cup S_2$ unsatisfied by the assignment $\vec{X}$ is $(\ell+1)(n-\ell-1)$, whereas the number of satisfied clauses from $S_3 \cup S_4$ is given by $\sum_{i=0}^{\ell} |T_{2i+1}| = (\ell+1)(n-\ell-1)$.

Let $\vec{X}(i)$ denote the current solution after i steps of execution. In Lemma 8 (see Appendix A) we show that $TS(\ell)$ cannot find a solution better than $\vec{X}$ if the starting solution $\vec{X}(0)$ of $TS(\ell)$ is $\vec{X}$. On the other hand, an optimal assignment can satisfy all the clauses by choosing the vector $\vec{X}_{\text{opt}} = (0, 0, \ldots, 0)$. Thus the approximation ratio of $TS(\ell)$, say R_ℓ, is

$$R_\ell = \frac{2\binom{n}{2}}{3\binom{n}{2} - n(\ell+2) + (\ell+2)(\ell+3) - 1}.$$

Let $\ell = n/c$ and assume that $c \geqslant 3$ (which is sufficient to guarantee that $n \geqslant 2\ell+5$ for any $n \geqslant 15$). As soon as the number n of variables increases, the approximation ratio R_ℓ asymptotically converges to

$$\lim_{n\to\infty} R_\ell = \frac{2c^2}{3c^2 - 2c + 2}. \qquad \square$$

Fig. 2 plots function (2). For any $\ell = o(n)$, this ratio asymptotically converges to 2/3. Therefore, $TS(\ell)$ cannot improve the basic local search when the maximum tabu tenure ℓ is sublinear in n (see (1) when $k = 2$).

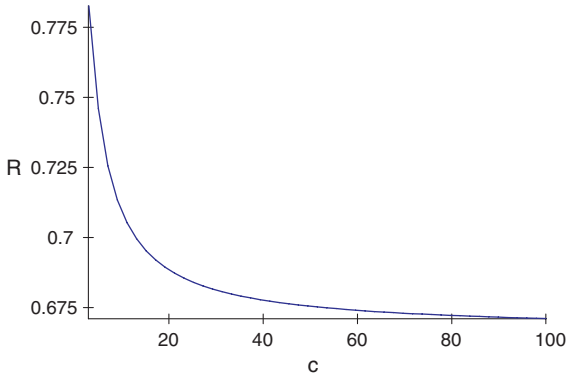

Fig. 2. Approximation ratio upper bound of $TS(\ell)$ as a function of $c = n/\ell$.

Corollary 2. *Starting from any arbitrary initial solution, the approximation ratio of TS(ℓ) for MAX-2-SAT is 2/3 for any $\ell = o(n)$.*

Moreover, already for $\ell \leqslant n/5$ the approximation ratio of $TS(\ell)$ is smaller than 3/4.

2.2. Tabu search with linear tabu tenure

The main question addressed in this subsection is to understand if there is a value of $\ell = \Theta(n)$ such that $TS(\ell)$ exhibits a worst-case superiority over the basic local search alone. Fig. 2 suggests that interesting values can be found when the ratio $c = n/\ell$ is "small". Indeed, we show that $\ell = n$ suffices to our goal. We prove that $TS(n)$ achieves an approximation ratio of 3/4, whereas the basic local search ensures an approximation ratio of 2/3.

Lemma 3. *Assume that for n consecutive steps TS(n) cannot find a better solution. Then the best found solution is a $\frac{3}{4}$-approximate solution.*

Proof. The claim follows by analyzing the execution of $TS(n)$. Remember that $\vec{X}(i)$ denotes the current solution after i steps of execution. Moreover, we denote by $\mathbb{C}_h^i$ the subset of clauses that have exactly h literals satisfied by assignment $\vec{X}(i)$. Without loss of generality, assume that $TS(n)$ has performed a number of steps such that the best solution found so far has been found at step s and that $\vec{X}(s)$ is a local optimum for LS (the presence of the aspiration criterion guarantees that, each time an improved solution is found, then that solution is improved until a local optimum is reached).

Now, in at most n steps from step s either a solution better than $\vec{X}(s)$ is found or there is a positive integer $t \leqslant n$ such that all the variables that appear in some clause from $\mathbb{C}_0^{s+t}$ have been flipped once during the last t steps. The latter can be easily observed by recalling that the flipped variables have been chosen according to their tabu status, and by using one of the described dynamic tabu policies. Therefore no clause that at step s has at least one unsatisfied literal, can be an unsatisfied clause at step $s+t$. It follows that every clause from $\mathbb{C}_0^s$ belongs to the set $\mathbb{C} - \mathbb{C}_0^{s+t}$ of satisfied clauses at iteration $s+t$. Moreover, within t steps all clauses from $\mathbb{C}_1^s$ are still satisfied, i.e. $\mathbb{C}_1^s \in \mathbb{C} - \mathbb{C}_0^{s+t}$. Now there are two possibilities:

1. $|\mathbb{C}_0^s| > |\mathbb{C}_0^{s+t}|$,
2. $|\mathbb{C}_0^s| \leqslant |\mathbb{C}_0^{s+t}|$.

If case 1 happens then after $t \leqslant n$ steps the previous best known solution has been improved by solution $\vec{X}(s+t)$ (the latter has a lower number of unsatisfied clauses). The number of times case 1 may happen is clearly bounded by m.

If case 2 happens then

$$|\mathbb{C}_0^{s+t}| \leqslant |\mathbb{C}| - |\mathbb{C}_0^s| - |\mathbb{C}_1^s|,$$

since, by previous arguments, no clause from $\mathbb{C}_0^s \cup \mathbb{C}_1^s$ belongs to the set $\mathbb{C}_0^{s+t}$ of unsatisfied clauses of solution $\vec{X}(s+t)$, i.e. $\mathbb{C}_0^{s+t} \subseteq \mathbb{C}/(\mathbb{C}_0^s \cup \mathbb{C}_1^s)$. It follows that

$$|\mathbb{C}| = |\mathbb{C}| - |\mathbb{C}_0^s| - |\mathbb{C}_1^s| + |\mathbb{C}_0^s| + |\mathbb{C}_1^s| \geqslant |\mathbb{C}_0^{s+t}| + |\mathbb{C}_0^s| + |\mathbb{C}_1^s| \geqslant 2|\mathbb{C}_0^s| + |\mathbb{C}_1^s| \geqslant 4|\mathbb{C}_0^s|, \tag{3}$$

where we have used $|\mathbb{C}_1^s| \geqslant 2|\mathbb{C}_0^s|$ that holds since $\vec{X}(s)$ is a local optimum (see the proof of Theorem 6 and inequality (11)). We remark that the aspiration criterion suffices to guarantee the latter condition. Inequality (3) immediately implies that the number of unsatisfied clauses at step s is no more than 1/4 the total number of clauses. Thus this algorithm ensures an approximation ratio of 3/4. □

Theorem 4. *Starting from any arbitrary initial assignment, TS(n) achieves an approximation ratio of* 3/4 *for MAX-2-SAT in* O(*mn*) *steps.*

Proof. The claim easily follows from Lemma 3 and by observing that the number of times a better solution is computed is bounded by the number m of clauses. □

We note that the same approximation ratio can be obtained for the weighted case of MAX-2-SAT, although the time complexity might be only pseudopolynomially bounded in the input size (simply take multiple copies of the clauses).

3. Analysis of plateau moves

We start this section by observing that no plateau search strategy (based on 1-flip neighborhood, like GSAT [20]) can help to improve the approximation ratio of the basic local search algorithm.

Theorem 5. *No plateau search strategy* (*based on* 1-*flip neighborhood*) *can improve the* 1/2-*approximation ratio of the basic local search LS.*

Proof. Consider the following instance of the weighted MAX-SAT problem that consists of three clauses:

$$(\overline{x}) \wedge (y) \wedge (x \vee \overline{y}),$$

where the weight of clause $(\overline{x})$ is 1, the weight of (y) is $M-1$ and the weight of $(x \vee \overline{y})$ is M, where M is a large number ($\gg 2$). Starting from assignment ($x=0$, $y=0$), it is easy to observe that plateau moves cannot improve the initial solution, which has value $M+1$. Since the optimal solution ($x=1$, $y=1$) value is $2M-1$, plateau search strategies do not help to improve the 1/2 approximation ratio of the basic local search *LS*. Now, observe that the weighted instance can be transformed into an unweighted instance by producing a number of copies of the same clause that is proportional to its weight (in the previous case 1 copy of $(\overline{x})$, $M-1$ of (y), and M of $(x \vee \overline{y})$, assuming, without loss of generality, M integral). Therefore, the same limitation holds also for the unweighted MAX-SAT problem when multiple copies of the same clause are allowed. □

By the proof of Theorem 5, it turns out that for the weighted MAX-SAT problem (and the unweighted version when multiple copies of the same clause are allowed), there is no plateau search strategy that has an approximation ratio strictly better than 1/2.

A natural question is to understand if the same holds when we restrict our analysis to the unweighted MAX-SAT problem where multiple copies of the same clause are not allowed. Observe that under these assumptions the basic local search algorithm has still an approximation ratio equal to 1/2. Indeed, consider the simple input instance $(\overline{x}) \wedge (x \vee y)$, and assume that the current input assignment is $\vec{X} = (x = 1, y = 0)$. Clearly, $\vec{X}$ is a local optimum for *LS* which achieves a performance ratio of 1/2. In contrast, we show that under these assumptions plateau moves can enhance the approximation ratio of the basic local search from 1/2 to 2/3. In particular, a reduced version of GSAT (in which only variables that appear in some unsatisfied unit clauses [3] can be involved in plateau steps) suffices for our purposes as explained in the following.

Let us start by saying that (x_j) (or $(\overline{x}_j)$) is a *contradicting* unit clause if, both, (x_j) and $(\overline{x}_j)$, are clauses of the input formula. According to any given variable assignment $\vec{X}$, let $U(\vec{X})$ denote the set (possibly empty) of variables such that variable x_j belongs to $U(\vec{X})$ if (x_j) (or $(\overline{x}_j)$) is a non-contradicting unsatisfied unit clause. Moreover, let $\Delta(x_j)$ be the difference in the number of satisfied clauses when variable x_j is flipped.

[3] A unit clause is a clause that consists of a single literal.

1: Apply LS until a local optimal assignment $\overrightarrow{X}$ is reached;
2: **if** there is a variable $u \in U(\overrightarrow{X})$ such that $\Delta(u) = 0$ **then**
3: reverse the truth assignment of u; $\{plateau\ step\}$
4: **goto** 1;
5: **else**
6: **return** $\overrightarrow{X}$;
7: **end if.**

Fig. 3. Algorithm LS^+.

The considered plateau search strategy LS^+ is obtained by enhancing the standard local search LS as described in Fig. 3. The main result of this section can be stated as follows.

Theorem 6. *Starting from any arbitrary initial assignment, algorithm LS^+ returns in $O(m)$ steps a $\max\left\{\frac{2}{3}, \frac{k}{k+1}\right\}$-approximate solution for the unweighted MAX-SAT problem when multiple copies of the same clause are not allowed.*

Proof. Let $z_i = 1$ when clause C_i is satisfied, and $z_i = 0$ otherwise. Our goal is to maximize the number of satisfied clauses, i.e.

$$\max f = \sum_{i=1}^{m} z_i.$$

Let $\vec{X}$ the current input assignment and consider the neighborhood of $\vec{X}$ obtained by flipping the value of one variable. The basic local search algorithm LS selects as next solution the neighbor with the best f value. Algorithm LS^+ can be conveniently analyzed by using a suitable function $f^+(\alpha)$ described as follows. Without loss of generality, assume that the non-contradicting unit clauses are the first $t \leqslant m$ in $\mathbb{C}$, i.e., $C_1, \ldots, C_t$. Now, function $f^+(\alpha)$ is defined as follows:

$$f^+(\alpha) = f + \alpha \sum_{i=1}^{t} z_i \quad 0 < \alpha < 1.$$

Let $\Delta^+(x_j)$ be the variation of the value of $f^+(\alpha)$ if variable x_j is flipped, according to any given assignment $\vec{X}$. By Lemma 7 (see below) the solution quality returned by LS^+ can be obtained by analyzing the local optimum (OPT^+_{loc}) of function $f^+(\alpha)$. Without loss of generality, assume that the solution returned by LS^+ is $\vec{X} = (x_1 = 1, x_2 = 1, \ldots, x_n = 1)$. Let $\mathbb{C}_s$ be the subset of clauses that have exactly s literals satisfied by assignment $\vec{X}$, and let $\mathbb{C}_s(x_i)$ denote the subset of clauses from $\mathbb{C}_s$ that contain literal x_i. By Lemma 7 the solution $\vec{X}$ returned by LS^+ is a local optimum for $f^+(\alpha)$. If we flip the current value of x_i then three cases may occur depending if **(a)** $(x_i) \in \mathbb{C}$ and $(\overline{x}_i) \notin \mathbb{C}$, **(b)** $(\overline{x}_i) \in \mathbb{C}$ and $(x_i) \notin \mathbb{C}$, **(c)** otherwise. According to these three possible scenarios the following three inequalities hold at local optimum, respectively:

(a) $\alpha + |\mathbb{C}_1(x_i)| > |\mathbb{C}_0(\overline{x}_i)|$;
(b) $|\mathbb{C}_1(x_i)| > |\mathbb{C}_0(\overline{x}_i)| + \alpha$;
(c) $|\mathbb{C}_1(x_i)| \geqslant |\mathbb{C}_0(\overline{x}_i)|$.

Cases (a) and (c) imply that

$$|\mathbb{C}_1(x_i)| \geqslant |\mathbb{C}_0(\overline{x}_i)|. \tag{4}$$

In case (b), since $|\mathbb{C}_1(x_i)|$ and $|\mathbb{C}_0(\bar{x}_i)|$ are two non-negative integers and $0<\alpha<1$, then

$$|\mathbb{C}_1(x_i)| \geqslant |\mathbb{C}_0(\bar{x}_i)| + 1. \tag{5}$$

Let $\mathbb{C}_{0,1}$ be the set of unsatisfied unit clauses. Set $\mathbb{C}_{0,1}$ can be seen as $\mathbb{C}_{0,1} = \mathbb{C}^{c}_{0,1} \cup \mathbb{C}^{nc}_{0,1}$, where $\mathbb{C}^{c}_{0,1}$ and $\mathbb{C}^{nc}_{0,1}$ are, respectively, the set of unsatisfied contradicting and non-contradicting unit clauses. Summing inequalities (4) and (5) over all variables we obtain the following inequality:

$$\sum_{i=1}^{n} |\mathbb{C}_1(x_i)| \geqslant \sum_{i=1}^{n} |\mathbb{C}_0(\bar{x}_i)| + |\mathbb{C}^{nc}_{0,1}|. \tag{6}$$

Let k_i denote the number of literals of clause C_i. Observe that

$$\sum_{i=1}^{n} |\mathbb{C}_0(\bar{x}_i)| = \sum_{C_i \in \mathbb{C}_0} k_i, \tag{7}$$

and the equality is demonstrated by counting how many times a clause in $\mathbb{C}_0$ is encountered during the sum: each clause $C_i \in \mathbb{C}_0$ with k_i literal will be encountered k_i times during the sum. Note that

$$\sum_{C_i \in \mathbb{C}_0} k_i = |\mathbb{C}_{0,1}| + \sum_{C_i \in \mathbb{C}_0/\mathbb{C}_{0,1}} k_i \geqslant |\mathbb{C}_{0,1}| + 2|\mathbb{C}_0/\mathbb{C}_{0,1}| \tag{8}$$

and therefore by combining conditions (7) and (8) we have

$$\sum_{i=1}^{n} |\mathbb{C}_0(\bar{x}_i)| + |\mathbb{C}^{nc}_{0,1}| \geqslant 2|\mathbb{C}_0| - |\mathbb{C}^{c}_{0,1}|. \tag{9}$$

By noting that

$$\sum_{i=1}^{n} |\mathbb{C}_0(\bar{x}_i)| \geqslant k|\mathbb{C}_0| \tag{10}$$

and $|\mathbb{C}_1| = \sum_{i=1}^{n} |\mathbb{C}_1(x_i)|$, we have by inequalities (6), (9) and (10) that

$$|\mathbb{C}_1| \geqslant \max(2,k)|\mathbb{C}_0| - |\mathbb{C}^{c}_{0,1}|. \tag{11}$$

Now, since $m = OPT^{+}_{\mathrm{loc}} + |\mathbb{C}_0| \geqslant |\mathbb{C}_1| + |\mathbb{C}_0| \geqslant \max(3,k+1)|\mathbb{C}_0| - |\mathbb{C}^{c}_{0,1}|$, it follows that $|\mathbb{C}_0| \leqslant \frac{m+|\mathbb{C}^{c}_{0,1}|}{\max(2,k)+1}$. Observing that $OPT \leqslant m - |\mathbb{C}^{c}_{0,1}|$, we have

$$OPT^{+}_{\mathrm{loc}} = m - |\mathbb{C}_0| \geqslant m - \frac{m + |\mathbb{C}^{c}_{0,1}|}{\max(3,k+1)} \geqslant \max\left\{\frac{2}{3}, \frac{k}{k+1}\right\} OPT.$$

Finally, we observe that the number of steps of algorithm LS^{+} is bounded by $O(m)$. To prove this, we start observing that LS^{+} keeps flipping variables that cause a net increase of $f^{+}(\alpha)$, and this holds for any $\alpha \in (0,1)$. Indeed, if LS^{+} flips the current value of some variable x_i that does not appear in any unit clause or, alternatively, it appears in some contradicting unit clause, then it is because this move improves the number of satisfied clauses by at least one. Otherwise, if LS^{+} flips the current value of some variable x_i that appear in some unsatisfied (non-contradicting) unit clause, then the net increase of $f^{+}(\alpha)$ is at least α. The last case is when LS^{+} flips the current value of some variable x_i that appear in some currently satisfied (non-contradicting) unit clause. In this case the net increase is at least $1-\alpha$. Now, let $\Delta(\alpha)$ denote the minimum increase of $f^{+}(\alpha)$ caused by one step of algorithm LS^{+}. Since $0 \leqslant f^{+}(\alpha) < m(1+\alpha)$, the number of steps of LS^{+} is bounded by $\frac{m(1+\alpha)}{\Delta(\alpha)}$, for any $\alpha \in (0,1)$. Set $\alpha := 1/2$. Using $\Delta(\alpha) \geqslant \min\{\alpha, 1-\alpha\}$, it follows that LS^{+} ends in at most $\frac{m(1+\alpha)}{\min\{\alpha,1-\alpha\}} = 3m$ steps. □

Lemma 7. *If $\vec{X}$ is the solution returned by LS^+ then $\vec{X}$ is a local optimum for $f^+(\alpha)$, i.e. $\Delta^+(x_j) \leqslant 0$ for every $j = 1, \ldots, n$.*

Proof. By contradiction, assume that solution $\vec{X}$ is not a local optimum for $f^+(\alpha)$ and therefore there is a variable x_j such that $f^+(\alpha)$ increases when x_j is flipped. Since $\vec{X}$ is a local optimum for LS, we cannot increase the value of f by flipping x_j. Moreover, if by flipping variable x_j the value of $\alpha\sum_{i=1}^{t} z_i$ increases (by α), then f decreases at least by 1, otherwise $\vec{X}$ would not be the solution returned by LS^+; it follows that $f^+(\alpha)$ actually decreases by at least $1-\alpha>0$, a contradiction. □

4. Future work

In this paper we have approached the problem of understanding the worst-case behavior of some local search strategies. In particular, we analyzed the worst-case behavior of tabu search as a function of the tabu list length for MAX-2-SAT. Moreover, we showed a simple case where a plateau search strategy enhances the approximation ratio of the basic local search. On both of these fronts there are many interesting lines for future research. It would be interesting to understand if the provided analysis of tabu search can be extended also for the general MAX-SAT problem. An interesting starting case would be MAX-3-SAT problem. The analysis of the plateau search strategy has been done when $k=1$. What about other cases? We believe that results in these directions are necessary steps toward a better understanding of local search strategies for MAX-SAT.

Acknowledgements

We are grateful to Fred Glover and the anonymous referees for many helpful comments. This research has been supported by the Swiss National Science Foundation project 200021-100539/1, "Approximation Algorithms for Machine scheduling Through Theory and Experiments", and by the "Metaheuristics Network", grant HPRN-CT-1999-00106.

Appendix A

Lemma 8. *According to the definitions given in the proof of Theorem* 1, *if $\vec{X}(0) = \vec{X}$ then $TS(\ell)$ cannot find a solution better than $\vec{X}$ for input S when $n \geqslant 2\ell+5$.*

Proof. The claim follows by showing that $TS(\ell)$ cycles around a set of solutions that are not better than $\vec{X}$. More precisely, we start proving that $TS(\ell)$ first flips, in the order, $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \ldots, x_1$, where $\delta = \ell+2$. Therefore, $TS(\ell)$ reaches solution $(1,1,\ldots,1)$ with a tabu list equal to $(x_1, x_3, \ldots, x_{2(\delta-2)-1})$. Then we show that $TS(\ell)$ flips, in the order, $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \ldots, x_1$ and again the starting solution $\vec{X}(0)$ is obtained. Since at this point the tabu list is equal to $(x_1, x_3, \ldots, x_{2(\delta-2)-1})$, $TS(\ell)$ repeats the same moves generating a cycle.

Let $gain(x_j, s)$ (and $loss(x_j, s)$) denote the number of clauses that become satisfied (unsatisfied) if variable x_j is flipped at step s. At step s algorithm $TS(\ell)$ chooses the variable x_j with the highest $\Delta(x_j, s) = gain(x_j, s) - loss(x_j, s)$. In the chosen example and at each step s, we will see that there is no more than one variable x_j that when flipped yields a maximal $\Delta(x_j, s)$. Therefore, random moves introduced to break ties are not used.

Starting from $\vec{X}(0)$, the following analysis reveals that at step $s=\delta-j$ (for $j=\delta-1,\delta-2,\ldots,1$) $TS(\ell)$ flips variable x_{2j-1}. We compute $\Delta(x_j, s=\delta-j)$ for every admissible variable x_j. The claim that $TS(\ell)$ flips, in the order, $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \ldots, x_1$, follows by induction.

1. $\Delta(x_{2w-1}, \delta-j)=w-j$, for $w=1,2,\ldots,j$.

Explanation: by flipping x_{2w-1}, for $w=1,2,\ldots,j$, the number of unsatisfied clauses in $S_1\cup S_2$ reduces from $j(n-j)$ to $(j-1)(n-(j-1))$; except for $(j-w)$ that remains satisfied, all the clauses from T_{2w-1} becomes unsatisfied, and the number of unsatisfied clauses in $S_3\cup S_4$ increases by $n-(2w-1)-(j-w)=n-(w+j)+1$. Therefore, $gain(x_{2w-1}, \delta-j)=n-2j+1$ and $loss(x_{2w-1}, \delta-j)=n-(w+j)+1$.

2. $\Delta(x_{2\delta-1}, \delta-j)=j-\delta$.

Explanation: by flipping $x_{2\delta-1}$ the number of unsatisfied clauses in $S_1\cup S_2$ increases from $j(n-j)$ to $(j+1)(n-(j+1))$, i.e. $loss(x_{2\delta-1}, \delta-j)=n-2j-1$; all the clauses from $T_{2\delta-1}$ becomes true plus one clause from each set T_{2i-1}, for $i=j+1,j+2,\ldots,\delta-1$, i.e., $gain(x_{2\delta-1}, \delta-j)=n-2\delta+(\delta-j-1)=n-\delta-j-1$.

3. $\Delta(x_{2w}, \delta-j)=-(n-2j-1)$, for $w=1,2,\ldots,j$.

Explanation: this move does not affect clauses from $S_3\cup S_4$, whereas the number of unsatisfied clauses in $S_1\cup S_2$ increases from $j(n-j)$ to $(j+1)(n-(j+1))$.

4. $\Delta(x_{2w}, \delta-j)=-n+j+w+1$, for $w=j+1, j+2,\ldots,\delta$.

Explanation: this move does not affect clauses from S_4, whereas it satisfies one clause for each set $T_{2j+1}, T_{2(j+1)+1}, \ldots, T_{2(w-1)+1}$. The number of unsatisfied clauses in $S_1\cup S_2$ increases from $j(n-j)$ to $(j+1)(n-(j+1))$.

5. $\Delta(x_{2\delta+1+w}, \ell-j)\leqslant j-\delta$, for $w=0,1,\ldots,n-(2\delta+1)$.

Explanation: by flipping $x_{2\delta+1+w}$, the number of unsatisfied clauses in $S_1\cup S_2$ increases from $j(n-j)$ to $(j+1)(n-(j+1))$. At most one clause for each set T_{2i+1} becomes true, for $i=j,\ldots,\delta-1$. One clause for each set $T_{2\delta+1+i}$ becomes true, for $i=0,\ldots,w-1$. All the clauses from $T_{2\delta+1+w}$ becomes satisfied. Therefore, $gain(x_{2\ell+1+w}, \ell-j+1)\leqslant \delta-j+w+n-(2\delta+1+w)=n-\delta-j-1$ and $loss(x_{2\ell+1}, 1)=n-2j-1$.

By the previous analysis, it follows that $TS(\ell)$ flips variable x_{2j-1} (case 1) at step $s=\delta-j$. The latter proves that $TS(\ell)$ flips, in the order, $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \ldots, x_1$; therefore, $TS(\ell)$ reaches solution $(1,1,\ldots,1)$ with a tabu list equal to $(x_1, x_3, \ldots, x_{2(\delta-2)-1})$.

In the following we show that $TS(\ell)$ starting from solution $(1,1,\ldots,1)$, and with a tabu list equal to $(x_1, x_3, \ldots, x_{2(\delta-2)-1})$, flips, in the order, $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \ldots, x_1$ and again the initial solution $\vec{X}(0)$ is obtained. At this point, we can easily observe that $TS(\ell)$ gets stuck into a cycle.

The analysis below reveals that at step $s=2\delta-j+1$ (for $j=\delta-1,\delta-2,\ldots,1$) algorithm $TS(\ell)$ flips variable x_{2j-1}. We compute $\Delta(x_j, s=2\delta-j+1)$ for every admissible variable x_j. The claim that $TS(\ell)$ flips, in the order, $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \ldots, x_1$, follows by induction. Let us start observing that every admissible move causes the increase of unsatisfied clauses in $S_1\cup S_2$ from $(\delta-1-j)(n-(\delta-1-j))$ to $(\delta-j)(n-(\delta-j))$, i.e., a loss of $(\delta-j)(n-(\delta-j))-(\delta-1-j)(n-(\delta-1-j))=n+2j-2\delta+1$.

1. $\Delta(x_{2j-1}, 2\delta-j+1)=\delta-2j$.

Explanation: by flipping x_{2j-1}, one clause for each set T_{2i-1} becomes true, for $i=1,\ldots,j-1$. All the clauses from T_{2j-1} becomes satisfied, except for $\delta-j-1$ that are already satisfied. Therefore, $gain(x_{2j-1}, 2\delta-j+1)=j-1+n-(2j-1)-(\delta-j-1)=n-\delta+1$ and $loss(x_{2j-1}, 2\delta-j+1)=n+2j-2\delta+1$.

2. $\Delta(x_{2\delta-1}, 2\delta-j+1)=-1-j$.

Explanation: by flipping $x_{2\delta-1}$ all the clauses from $T_{2\delta-1}$ becomes satisfied plus one clause from each set T_{2i-1}, for $i=1,2,\ldots,j$, i.e., $gain(x_{2\delta-1}, 2\delta-j+1)=n-2\delta+j$ and $loss(x_{2\delta-1}, 2\delta-j+1)=n+2j-2\delta+1$.

3. $\Delta(x_{2w}, 2\delta-j+1)=w-(n+2j-2\delta+1)$, for $w=1,2,\ldots,j$.

Explanation: one clause for each set T_{2i-1} becomes true, for $i=1,\ldots,w$.

4. $\Delta(x_{2w}, 2\delta-j+1) \leqslant -j-n+2\delta$, for $w=j+1,\ldots,\delta$.

Explanation: one clause for each set T_{2i-1} becomes true, for $i=1,\ldots,j$, plus one clause from $T_{2\delta-1}$ if $w=\delta$.

5. $\Delta(x_{2\delta+1}, 2\delta-j+1) \leqslant -j-1$.

Explanation: by flipping $x_{2\delta+1}$, one clause for each set T_{2i-1} becomes true, for $i=1,\ldots,j$, plus at most one clause from $T_{2\delta-1}$. All the clauses from $T_{2\delta+1}$ becomes satisfied. Therefore, $gain\,(x_{2\delta+1}, 2\delta-j+1) \leqslant j+1+n-(2\delta+1)=j+n-2\delta$ and $loss(x_{2\delta-1}, 2\delta-j+1)=n+2j-2\delta+1$.

6. $\Delta(x_{2\delta+2+w}, 2\delta-j+1) \leqslant -j-1$, for $w=0,1,\ldots,n-(2\delta+2)$.

Explanation: by flipping $x_{2\delta+2+w}$, one clause for each set T_{2i+1} becomes true, for $i=0,1,\ldots,j$, plus at most one clause from $T_{2\delta-1}$. One clause for each set $T_{2\delta+2+i}$ becomes true, for $i=0,1,\ldots,w-1$. All the clauses from $T_{2\delta+2+w}$ becomes satisfied. Therefore, $gain(x_{2w-1}, 2\delta-j+1) \leqslant j+2+w+n-(2\delta+2+w)=j+n-2\delta$ and $loss(x_{2w-1}, 2\delta-j+1)=n+2j-2\delta+1$. □

References

[1] T. Asano, D.P. Williamson, Improved approximation algorithms for MAX SAT, in: Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, ACM Press, 2000, pp. 96–105.

[2] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, Complexity and Approximation, Springer, 1999.

[3] R. Battiti, M. Protasi, Reactive search, a history-sensitive heuristic for MAX-SAT, ACM Journal of Experimental Algorithms 2 (Article 2) (1997).

[4] R. Battiti, M. Protasi, Approximate algorithms and heuristics for MaxSat, in: D.-Z. Du, P.M. Pardalos (Eds.), Handbook of Combinatorial Optimization, vol. 1, Kluwer Academic Publishers, Dordrecht, 1998, pp. 77–148.

[5] R. Battiti, G. Tecchiolli, The reactive tabu search, ORSA Journal on Computing 6 (1994) 126–140.

[6] M. Dell'Amico, M. Trubian, Applying tabu-search to the job shop scheduling problem, Annals of Operations Research 22 (1993) 15–24.

[7] M.R. Garey, D.S. Johnson, Computers and Intractability; A Guide to the Theory of NP-Completeness, W.H. Freeman, 1979.

[8] F. Glover, Future paths for integer programming and links to artificial intelligence, Computers and Operations Research 13 (1986) 533–549.

[9] F. Glover, Tabu search—Part I, ORSA Journal on Computing 1 (3) (1989) 190–206.

[10] F. Glover, Tabu search—Part II, ORSA Journal on Computing 2 (1) (1990) 4–32.

[11] M.X. Goemans, D.P. Williamson, New $\frac{3}{4}$-approximation algorithms for MAX SAT, SIAM Journal on Discrete Mathematics 7 (1994) 656–666.

[12] M.X. Goemans, D.P. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming, Journal of the Association for Computing Machinery 42 (1995) 1115–1145.

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

[14] H.H. Hoos, On the run-time behaviour of stochastic local search algorithms for SAT, in: Proceedings of the 6th National Conference on Artificial Intelligence (AAAI-99); Proceedings of the 11th Conference on Innovative Applications of Artificial Intelligence, AAAI/MIT Press, 1999, pp. 661–666.

[15] S. Khanna, R. Motwani, M. Sudan, U. Vazirani, On syntactic versus computational views of approximability, in: Proceedings: 35th Annual Symposium on Foundations of Computer Science, 1994, pp. 819–830.

[16] M. Mastrolilli, L.M. Gambardella, Effective neighbourhood functions for the flexible job shop problem, Journal of Scheduling 3 (2000) 3–20.

[17] B. Mazure, L. Saïs, É. Grégoire, Tabu search for SAT, in: Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference (AAAI-97/IAAI-97), AAAI Press, 1997, pp. 281–285.

[18] P. Mills, E. Tsang, Guided local search for solving SAT and weighted MAX-SAT problems, JAR: Journal of Automated Reasoning 24 (2000) 205–223.

[19] B. Selman, H.A. Kautz, B. Cohen, Local search strategies for satisfiability testing, in: M. Trick, D.S. Johnson (Eds.), Proceedings of the Second DIMACS, Challenge on Cliques, Coloring, and Satisfiability, 1993.

[20] B. Selman, H.J. Levesque, D. Mitchell, A new method for solving hard satisfiability problems, in: P. Rosenbloom, P. Szolovits (Eds.), Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI Press, 1992, pp. 440–446.
[21] K. Smyth, H.H. Hoos, T. Stuetzle, Iterated robust tabu search for MAX-SAT, in: Proceedings of the 16th Canadian Conference on Artificial Intelligence (AI'2003), LNCS 2671, 2003, pp. 661–666.
[22] Z. Wu, B.W. Wah, An efficient global-search strategy in discrete lagrangian methods for solving hard satisfiability problems, in: Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the 12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00), AAAI Press, 2000, pp. 310–315.
[23] M. Yannakakis, On the approximation of maximum satisfiability, Journal of Algorithms 17 (3) (1994) 475–502.