

# MAX-2-SAT: How Good is Tabu Search in the Worst-Case?

**Monaldo Mastrolilli**

IDSIA  
Galleria 2, 6928 Manno, Switzerland  
monaldo@idsia.ch

**Luca Maria Gambardella**

IDSIA  
Galleria 2, 6928 Manno, Switzerland  
luca@idsia.ch

## 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 the 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 the main 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 a 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.

## 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:

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

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  $\bar{v}_j$ , for  $1 \leq j \leq n$ . The set of clauses in the formula is denoted by  $\mathbb{C} = \{C_1, C_2, \dots, C_m\}$ . 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 (Ausiello *et al.* 1999). 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 (Battiti & Protasi 1998) 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 (Hansen & Jaumard 1990) that

$$OPT_{loc} \geq \frac{k}{k+1} OPT. \quad (1)$$

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 (Battiti & Protasi 1998) and (Hoos 1999; Mills & Tsang 2000; Wu & Wah 2000; Smyth, Hoos, & Stuetzle 2003) 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<sup>1</sup> for MAX-SAT (for this see (Asano & Williamson

<sup>1</sup>The quality of an approximation algorithm is measured by its so called *performance ratio*. An approximation algorithm for

2000; Goemans & Williamson 1994; 1995; Yannakakis 1994)), but to provide a first worst-case analysis of some tabu search procedures 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. 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 utilizes memory for guiding the search process (Glover 1989; 1990). 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 (Hansen & Jaumard 1990), the target application was MAX-SAT.

Mazure, Saïs, & Grégoire (1997) 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 the Random Walk Strategy GSAT (Selman, Kautz, & Cohen 1993), 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, Saïs, & Grégoire 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 (Garey & Johnson 1979). We analyze a tabu search algorithm,  $TS(\ell)$  (see Figure 1), which has the same flavor of TSAT (and of many other tabu search implementations), and whose tabu list length is bounded by the parameter  $\ell$ . 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  $\ell$  for which  $TS(\ell)$  is better

MAX-SAT has a performance ratio of  $\rho$ , if for all input instances it finds a solution of value at least  $\rho$  times the optimal solution value.

```

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

```

Figure 1: Tabu Search algorithm  $TS(\ell)$  with tabu tenure at most  $\ell$ .

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. 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. To some extent, these results confirm the linearity in  $n$  of the optimal length of the tabu lists, as it was observed experimentally in (Mazure, Saïs, & Grégoire 1997). 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.

The paper is organized as follows. In the next section, we give a detailed description of the  $TS(\ell)$  procedure. We then derive an upper bound on the approximation ratio of  $TS(\ell)$  as a function of  $\ell$ . Finally, 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 Figure 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  $\ell$ . 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 each step, 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

- (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* (Glover 1989)).

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 9 of Figure 1). Each time a variable is flipped set  $TT \leftarrow \max\{TT + 1, \ell\}$  (add this command just after line 11 of Figure 1);
- (*least recently used*) flip the least recently used variable that appears in some unsatisfied clause (add this command at line 9 of Figure 1).

Observe that in both strategies the tabu tenure of a variable is at most  $\ell$ , but it can be less than  $\ell$  (dynamic tabu policy). We refer to  $\ell$  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 5 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 the next section we prove that the useful values of  $\ell$  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 (Battiti & Protasi 1997; Battiti & Tecchiolli 1994; Dell'Amico & Trubian 1993; Mastrolilli & Gambardella 2000). Advanced ways to create dynamic tabu tenure are described in (Glover 1990).

## 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  $\ell$ . 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  $c \geq 3$ , the approximation ratio of  $TS(\ell)$  for MAX-2-SAT is bounded from above by*

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

**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 (Khanna *et al.* 1994). We assume that  $n \geq 2\ell + 5$ . 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.

$$\begin{aligned} S_1 &= \bigcup_{1 \leq i < j \leq n} (x_i \vee \bar{x}_j), \\ S_2 &= \bigcup_{1 \leq i < j \leq n} (\bar{x}_i \vee x_j), \\ S_3 &= \left( \bigcup_{0 \leq i \leq \ell+2} T_{2i+1} \right) - \{(\bar{x}_{2\ell+3} \vee \bar{x}_n)\}, \\ S_4 &= \bigcup_{2\ell+6 \leq i \leq n} T_i, \\ T_i &= \bigcup_{i < j \leq n} (\bar{x}_i \vee \bar{x}_j). \end{aligned}$$

Note that set  $S_3$  is obtained by eliminating clause  $(\bar{x}_{2\ell+3} \vee \bar{x}_n)$ . 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  $x_{2i-1}$  ( $i = 1, \dots, \ell + 1$ ) 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 5 (see appendix) 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  $\vec{X}_{opt} = (0, 0, \dots, 0)$ . Thus the approximation ratio of  $TS(\ell)$ , say  $R_\ell$ , 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  $c \geq 3$  (which is sufficient to guarantee that  $n \geq 2\ell + 5$  for any  $n \geq 15$ ). As soon as the

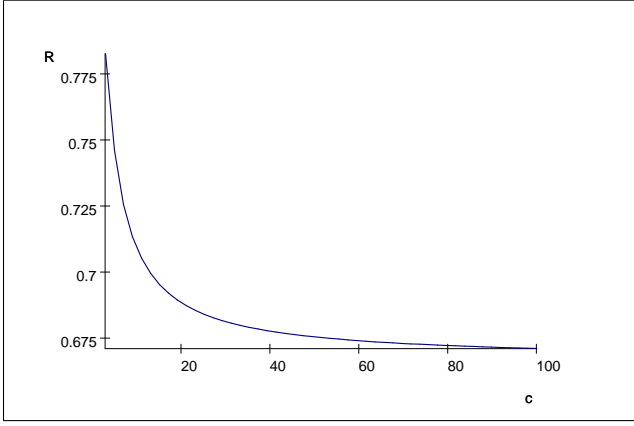


Figure 2: Approximation ratio upper bound of  $TS(\ell)$  as a function of  $c = n/\ell$ .

number  $n$  of variables increases, the approximation ratio  $R_\ell$  asymptotically converges to

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

Figure 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  $\ell$  is sublinear in  $n$  (see (1) when  $k = 2$ ).

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

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

### 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. Figure 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  $t \leq n$  such that all the variables that appear in some clause from  $\mathbb{C}_0^{s+t}$  have been flipped just 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.  $|\mathbb{C}_0^s| \leq |\mathbb{C}_0^{s+t}|$

If case 1 happens then after  $t \leq n$  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

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

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

$$\begin{aligned} |\mathbb{C}| &= |\mathbb{C}| - |\mathbb{C}_0^s| - |\mathbb{C}_1^s| + |\mathbb{C}_0^s| + |\mathbb{C}_1^s| \\ &\geq |\mathbb{C}_0^{s+t}| + |\mathbb{C}_0^s| + |\mathbb{C}_1^s| \geq 2|\mathbb{C}_0^s| + |\mathbb{C}_1^s| \\ &\geq 4|\mathbb{C}_0^s|, \end{aligned} \quad (3)$$

where we have used  $|\mathbb{C}_1^s| \geq 2|\mathbb{C}_0^s|$  that holds since  $\vec{X}(s)$  is a local optimum (see (Hansen & Jaumard 1990) or pp. 120-121 in (Battiti & Protasi 1998)). 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).

## Future Work

In this paper we have approached the problem of understanding the worst-case behavior of tabu search. In particular, we analyzed the worst-case behavior of tabu search as a function of the tabu list length for MAX-2-SAT. Certainly there are many interesting lines for future research. It would be interesting to understand if a similar analysis can be provided for the general MAX-SAT problem. An interesting starting case would be MAX-3-SAT. We believe that results in these directions are necessary steps toward a better understanding of local search strategies for MAX-SAT.

## Acknowledgments

We are grateful to Fred Glover 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

**Lemma 5** *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  $n \geq 2\ell + 5$ .*

**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,  $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \dots, x_1$ , where  $\delta = \ell + 2$ . Therefore,  $TS(\ell)$  reaches solution  $(1, 1, \dots, 1)$  with a tabu list equal to  $(x_1, x_3, \dots, x_{2(\delta-2)-1})$ . Then we show that  $TS(\ell)$  flips, in the order,  $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \dots, x_1$  and again the starting solution  $\vec{X}(0)$  is obtained. Since at this point the tabu list is equal to  $(x_1, x_3, \dots, x_{2(\delta-2)-1})$ ,  $TS(\ell)$  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  $j = \delta - 1, \delta - 2, \dots, 1$ )  $TS(\ell)$  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,  $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \dots, x_1$ , follows by induction.

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

Explanation: by flipping  $x_{2w-1}$ , for  $w = 1, 2, \dots, j$ , 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  $i = j + 1, j + 2, \dots, \delta - 1$ , 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  $w = 1, 2, \dots, j$ .

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  $w = j + 1, j + 2, \dots, \delta$ .

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

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

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  $i = j, \dots, \delta - 1$ . One clause for each set  $T_{2\delta+1+i}$  becomes true, for  $i = 0, \dots, w - 1$ . All the clauses from  $T_{2\delta+1+w}$  becomes satisfied. Therefore,  $gain(x_{2\delta+1+w}, \ell - j + 1) \leq \delta - j + w + n - (2\delta + 1 + w) = n - \delta - j - 1$  and  $loss(x_{2\delta+1+w}, \ell - j + 1) = n - 2j - 1$ .

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,  $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \dots, x_1$ ; therefore,  $TS(\ell)$  reaches solution  $(1, 1, \dots, 1)$  with a tabu list equal to  $(x_1, x_3, \dots, x_{2(\delta-2)-1})$ .

In the following we show that  $TS(\ell)$  starting from solution  $(1, 1, \dots, 1)$ , and with a tabu list equal to  $(x_1, x_3, \dots, x_{2(\delta-2)-1})$ , flips, in the order,  $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \dots, x_1$  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  $j = \delta - 1, \delta - 2, \dots, 1$ ) 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,  $x_{2(\delta-1)-1}, x_{2(\delta-2)-1}, \dots, x_1$ , 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  $i = 1, \dots, j - 1$ . 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  $i = 1, 2, \dots, j$ , 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  $w = 1, 2, \dots, j$ .  
Explanation: one clause for each set  $T_{2i-1}$  becomes true, for  $i = 1, \dots, w$ .
4.  $\Delta(x_{2w}, 2\delta - j + 1) \leq -j - n + 2\delta$ , for  $w = j + 1, \dots, \delta$ .  
Explanation: one clause for each set  $T_{2i-1}$  becomes true, for  $i = 1, \dots, j$ , plus one clause from  $T_{2\delta-1}$  if  $w = \delta$ .
5.  $\Delta(x_{2\delta+1}, 2\delta - j + 1) \leq -j - 1$ .  
Explanation: by flipping  $x_{2\delta+1}$ , one clause for each set  $T_{2i-1}$  becomes true, for  $i = 1, \dots, j$ , plus at most one clause from  $T_{2\delta-1}$ . All the clauses from  $T_{2\delta+1}$  becomes satisfied. Therefore,  $gain(x_{2\delta+1}, 2\delta - j + 1) \leq j + 1 + n - (2\delta + 1) = j + n - 2\delta$  and  $loss(x_{2\delta-1}, 2\delta - j + 1) = n + 2j - 2\delta + 1$ .
6.  $\Delta(x_{2\delta+2+w}, 2\delta - j + 1) \leq -j - 1$ , for  $w = 0, 1, \dots, n - (2\delta + 2)$ .  
Explanation: by flipping  $x_{2\delta+2+w}$ , one clause for each set  $T_{2i+1}$  becomes true, for  $i = 0, 1, \dots, j$ , plus at most one clause from  $T_{2\delta-1}$ . One clause for each set  $T_{2\delta+2+i}$  becomes true, for  $i = 0, 1, \dots, w - 1$ . All the clauses from  $T_{2\delta+2+w}$  becomes satisfied. Therefore,  $gain(x_{2\delta+2+w}, 2\delta - j + 1) \leq j + 2 + w + n - (2\delta + 2 + w) = j + n - 2\delta$  and  $loss(x_{2w-1}, 2\delta - j + 1) = n + 2j - 2\delta + 1$ .

■

## References

- Asano, T., and Williamson, D. P. 2000. Improved approximation algorithms for MAX SAT. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms*, 96–105. N.Y.: ACM Press.
- Ausiello, G.; Crescenzi, P.; Gambosi, G.; Kann, V.; Marchetti-Spaccamela, A.; and Protasi, M. 1999. *Complexity and Approximation*. Springer.
- Battiti, R., and Protasi, M. 1997. Reactive Search, a History-Sensitive Heuristic for MAX-SAT. *ACM Journal of Experimental Algorithms* 2:Article 2.
- Battiti, R., and Protasi, M. 1998. Approximate Algorithms and Heuristics for MaxSat. In Du, D.-Z., and Pardalos, P. M., eds., *Handbook of Combinatorial Optimization*, volume 1. Kluwer Academic Publishers. 77–148.
- Battiti, R., and Tecchiolli, G. 1994. The reactive tabu search. *ORSA Journal on Computing* 6:126–140.
- Dell’Amico, M., and Trubian, M. 1993. Applying tabu-search to the job shop scheduling problem. *Annals of Operations Research* 22:15–24.
- Garey, M. R., and Johnson, D. S. 1979. *Computers and intractability; a guide to the theory of NP-completeness*. W.H. Freeman.
- Glover, F. 1989. Tabu search - Part I. *ORSA Journal on Computing* 1(3):190–206.
- Glover, F. 1990. Tabu search - Part II. *ORSA Journal on Computing* 2(1):4–32.
- Goemans, M. X., and Williamson, D. P. 1994. New  $\frac{3}{4}$ -approximation algorithms for MAX SAT. *SIAM Journal on Discrete Mathematics* 7:656–666.
- Goemans, M. X., and Williamson, D. P. 1995. Improved Approximation Algorithms for Maximum Cut and Satisfiability Problems Using Semidefinite Programming. *J. Assoc. Comput. Mach.* 42:1115–1145.
- Hansen, P., and Jaumard, B. 1990. Algorithms for the maximum satisfiability problem. *Computing* 44:279–303.
- Hoos, H. H. 1999. On the run-time behaviour of stochastic local search algorithms for SAT. In *Proceedings of the 16th National Conference on Artificial Intelligence (AAAI-99) and of the 11th Conference on Innovative Applications of Artificial Intelligence (IAAI-99)*, 661–666. AAAI Press.
- Khanna, S.; Motwani, R.; Sudan, M.; and Vazirani, U. 1994. On syntactic versus computational views of approximability. In *Proceedings: 35th Annual Symposium on Foundations of Computer Science*, 819–830.
- Mastrolilli, M., and Gambardella, L. M. 2000. Effective neighbourhood functions for the flexible job shop problem. *Journal of Scheduling* 3:3–20.
- Mazure, B.; Saïs, L.; and Grégoire, É. 1997. Tabu search for SAT. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI-97) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI-97)*, 281–285. AAAI Press.
- Mills, P., and Tsang, E. 2000. Guided local search for solving SAT and weighted MAX-SAT problems. *JAR: Journal of Automated Reasoning* 24:205–223.
- Selman, B.; Kautz, H. A.; and Cohen, B. 1993. Local search strategies for satisfiability testing. In Trick, M., and Johnson, D. S., eds., *Proceedings of the Second DIMACS Challenge on Cliques, Coloring, and Satisfiability*.
- Smyth, K.; Hoos, H. H.; and Stuetzle, T. 2003. Iterated robust tabu search for MAX-SAT. In *Proceedings of the 16th Canadian Conference on Artificial Intelligence (AI’2003)*, 661–666. LNCS 2671.
- Wu, Z., and Wah, B. W. 2000. 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)*, 310–315. AAAI Press.
- Yannakakis, M. 1994. On the approximation of maximum satisfiability. *J. Algorithms* 17(3):475–502.