Efficient 2 and 3-Flip Neighborhood Search Algorithms for the MAX SAT: Experimental Evaluation

M. Yagiura and T. Ibaraki

Department of Applied Mathematics and Physics
Graduate School of Informatics, Kyoto University
Kyoto 606-8501, Japan
e-mail: {yagiura, ibaraki}@i.kyoto-u.ac.jp

Abstract: For problems SAT and MAX SAT, local search algorithms are widely acknowledged as one of the most effective approaches. Most of the local search algorithms are based on the 1-flip neighborhood, which is the set of solutions obtainable by flipping the truth assignment of one variable. In this paper, we consider r-flip neighborhoods for $r = 2, 3$, and examine their effectiveness by computational experiments. In the accompanying paper, we proposed new implementations of these neighborhoods, and showed that the expected size of 2-flip neighborhood is $O(n+m)$ and that of 3-flip neighborhood is $O(m+t^2n)$, compared to their original size $O(n^2)$ and $O(n^3)$, respectively, where n is the number of variables, m is the number of clauses and t is the maximum number of appearances of one variable. These are used in this paper under the framework of tabu search and other metaheuristic methods, and compared with other existing algorithms with 1-flip neighborhood. The results exhibit good prospects of larger neighborhoods.

Keywords: weighted MAX SAT, SAT, local search, metaheuristics, neighborhood.

1 Introduction

Given n 0-1 variables x_j, $j \in N$, m clauses C_i, $i \in M$, and weights w_i (≥ 0), $i \in M$, where $N = \{1, 2, \ldots, n\}$ and $M = \{1, 2, \ldots, m\}$, the MAX SAT problem asks to determine a 0-1 assignment that maximizes the sum of the weights of satisfied clauses. Denoting $\overline{x}_j = 1 - x_j$ and $L = \cup_{j \in N}\{x_j, \overline{x}_j\}$ (the set of literals), clauses are defined by $C_i \subseteq L$ for $i \in M$ (e.g., $C_i = \{x_1, \overline{x}_3, x_8\}$). Without loss of generality, we assume that at most one of x_j and $\overline{x}_j$ is included in each clause. For a $v \in \{0,1\}^n$, let

$$P_i(v) = \{j \in N \mid x_j \in C_i \text{ and } v_j = 1, \text{ or } \overline{x}_j \in C_i \text{ and } v_j = 0\} \tag{1}$$

and

$$\theta_i(v) = \begin{cases} 1, & \text{if } |P_i(v)| \geq 1 \\ 0, & \text{otherwise.} \end{cases}$$

Then the objective function is given by:

$$\text{maximize} \quad f(v) = \sum_{i \in M} w_i \theta_i(v). \tag{2}$$

The problem SAT is a decision problem asking whether there exists a $v \in \{0,1\}^n$ that attains $f(v) = \sum_{i \in M} w_i$; such a v is called a satisfying assignment. The problems SAT and MAX SAT are known to be NP-hard [4].

Many local search (abbreviated as LS) algorithms for SAT and MAX SAT problems have been proposed [5, 6, 15, 16, 18]. The local search starts from an initial solution v and repeats replacing v with a better solution v' in its *neighborhood* $NB(v)$ until no better solution is found in $NB(v)$, where $NB(v)$ is the set of solutions obtainable from v by slight perturbations. A solution v is called *locally optimal* (with respect to the neighborhood), if no better solution exists in $NB(v)$. We call the replacement of the current solution v by another solution a *move*. One of the following two move strategies are commonly used: *First admissible* move strategy (abbreviated as FA) and *best admissible* move strategy (abbreviated as BA). FA scans the neighborhood $NB(v)$ according to a prespecified random order and moves to the first improved solution. BA scans the entire neighborhood and move to the best solution in $NB(v)$. The local search is often applied to a number of randomly generated initial solutions, and the best among the obtained locally optimal solutions is output. This is called the *random multi-start local search* (abbreviated as MLS).

Let

$$D(v, v') = \{j \in N |\ v_j \neq v'_j\}$$

and

$$NB_r(v) = \{v' \in \{0,1\}^n |\ |D(v, v')| \leq r\}$$

(i.e., $v' \in NB_r(v)$ is obtainable from v by flipping at most r variables). We call $NB_r(v)$ *r-flip* neighborhood. To the best of authors' knowledge, most of the existing local search algorithms for SAT and MAX SAT are based on 1-flip neighborhood [1, 3, 5, 6, 11, 14, 15, 16, 18, 19, 20, 21, 22].

In this paper, we consider r-flip neighborhood for $r = 2$ and 3, and examine their effectiveness by computational experiments. Usually the quality of locally optimal solutions improves if larger neighborhood is used; however, the computational time to search NB_r also increases exponentially with r, since $|NB_r| = O(n^r)$ holds in general. To overcome this, we propose, in [23], restricted 2 and 3-flip neighborhoods NB'_r ($r = 2, 3$) and their efficient implementations by making use of the memory structure, which are reviewed in Section 2.

Computational experiments to evaluate the effectiveness of these neighborhoods are conducted on five types of instances with up to $n = 1000$: (1) random unweighted instances (denoted RNDU), (2) random weighted instances (denoted RNDW), (3) graph coloring instances (denoted GCP), (4)

set covering instances (denoted SCP), and (5) a time tabling instance (denoted TTP). We tested three metaheuristic frameworks: (i) *random multi-start local search* (denoted MLS), (ii) *iterated local search* (denoted ILS), and (iii) *tabu search* (denoted TS). It is observed, for types GCP, SCP and TTP instances, that the proposed 2 and 3-flip neighborhood search algorithms with restricted neighborhoods are more effective than 1-flip neighborhood under the frameworks of MLS, ILS and TS, if about the same computational time is allowed. However, for types RNDU and RNDW instances, the overall best performance is obtained by TS with 1-flip neighborhood, although 2 and 3-flip neighborhoods are more effective under the frameworks of MLS and ILS. As a special case of SCP, we tested the set covering problems based on Steiner triple systems (denoted STS), and found the best solutions reported in [12, 13]. The proposed algorithms are also compared with other existing algorithms. The results exhibited good prospects of the 2 and 3-flip neighborhoods, especially for problems with structures such as SCP and TTP.

2 Efficient LS Implementation

In this section, we outline the idea of the efficient implementation of 2 and 3-flip neighborhoods, and review the results of [23]. Let us call the computation needed for determining one move in LS as *one-round*. Let t denote the maximum number of appearances of a variable in m clauses C_i and l denote the maximum number of literals in a clause. If we evaluate the objective values (2) for all the solutions in the neighborhood from scratch, taking $O(ml)$ time for each evaluation, one-round time is $O(ml|NB_r|) = O(mln^r)$ for r-flip neighborhood. To reduce this computational effort, we use two ideas: (1) a clever memory structure to evaluate each solution in the neighborhood in $O(1)$ time, and (2) restricted neighborhoods without sacrificing the solution quality.

First we explain the idea of the memory structure. For a subset $S \subseteq N$ and a vector $v \in \{0,1\}^n$, let $v \updownarrow S$ denote the vector obtained from v by flipping the 0-1 assignment of variables in S (i.e., $D(v, v\updownarrow S) = S$), and call the following increase in the objective function as the *flip gain*,

$$\begin{aligned} \Delta f(v,S) &= f(v\updownarrow S) - f(v) \\ &= \sum_{i\in M} \Delta f_i(v,S), \end{aligned} \tag{3}$$

where

$$\Delta f_i(v,S) = w_i\{\theta_i(v\updownarrow S) - \theta_i(v)\}.$$

If we store the flip gains for all candidates in NB_r, we can evaluate each solution in NB_r in $O(1)$ time; however, $\Omega(n^r)$ space is required and $\Omega(n^{r-1})$ time is needed to update the memory for each move. To overcome this, we first prepare flip gains of all candidates in 1-flip neighborhood. For the

candidates in 2-flip neighborhood, instead of storing the flip gains directly, we evaluate the flip gain by

$$\Delta f(v, \{j_1, j_2\}) = \Delta f(v, \{j_1\}) + \Delta f(v, \{j_2\}) + g(v, \{j_1, j_2\}),$$

whose precise definition is omitted here (see [23]). For example, consider a clause C_i with $x_{j_1}, x_{j_2} \in C_i$ and $P_i(v) = \emptyset$. In this case, $\Delta f_i(v, \{j_1\}) = \Delta f_i(v, \{j_2\}) = \Delta f_i(v, \{j_1, j_2\}) = w_i$ holds, and we need the adjustment term $g_i(v, \{j_1, j_2\}) = -w_i$ so that we obtain $\Delta f_i(v, \{j_1, j_2\}) = \Delta f_i(v, \{j_1\}) + \Delta f_i(v, \{j_2\}) + g_i(v, \{j_1, j_2\})$. The value $g(v, \{j_1, j_2\}) = \sum_{i \in M} g_i(v, \{j_1, j_2\})$ represents the sum of such adjustments for all clauses C_i. By storing $g(v, \{j_1, j_2\})$ in memory, $\Delta f(v, \{j_1, j_2\})$ of each solution $v \updownarrow \{j_1, j_2\}$ in the 2-flip neighborhood can be evaluated in $O(1)$ time. Important point is that $g(v, \{j_1, j_2\})$ takes nonzero value only if x_{j_1} or $\overline{x}_{j_1} \in C_i$ and x_{j_2} or $\overline{x}_{j_2} \in C_i$ hold for some clause C_i. As l is usually small, such cases occur rather rarely. For the same reason, the update of such $g(v, \{j_1, j_2\})$ values at each move will not be expensive. This idea can be generalized to the case of $r \geq 3$ by using the principle of inclusion and exclusion.

With this memory structure, assuming that r is a fixed constant, the expected time needed for each update of the memory structure becomes $O(t)$, and the expected memory requirement becomes $O(n + m)$ [23].

Then we restrict the search to the candidates having the possibility of increasing the objective value (2). This is done by using the information of the above memory structure.

Lemma 2.1 [23] *If* $\Delta f(v, \{j_1\}) \leq 0$ *and* $\Delta f(v, \{j_2\}) \leq 0$ *hold for the current solution* v*, then* $\Delta f(v, \{j_1, j_2\}) > 0$ *is possible only if* $g(v, \{j_1, j_2\}) > 0$ *holds.*

Define

$$NB'_2(v) = \{v \updownarrow S \mid g(v, S) > 0 \text{ and } |S| \leq 2\}.$$

Then, by this lemma, we have only to check $NB'_2(v)$ to find a better solution in $NB_2(v)$ or to conclude local optimality.

Based on these restricted 2 and 3-flip neighborhoods (without sacrificing the solution quality), the expected size of 2-flip neighborhood is $O(n + m)$, and that of 3-flip neighborhood is $O(m + t^2 n)$ [23]. Therefore, the expected one-round time with the restricted neighborhood becomes $O(n + m)$ if $r = 2$, and $O(m + t^2 n)$ if $r = 3$. It is also noted that, for $r = 1$ and move strategy FA, the worst case one-round time of this algorithm is $O(tl)$, and its expected one-round time is $O(t)$. For the 1-flip neighborhood, similar result has already been reported in [3].

3 Frameworks of ILS and TS

Based on the results in [23], the LS with the restricted 2 and 3-flip neighborhoods can be implemented so that they become much faster and more practical than those with the ordinary 2 and 3-flip neighborhoods. We implemented random multi-start local search (MLS), iterated local search (ILS), and tabu search (TS), using the ordinary 1-flip neighborhood and the restricted 2 and 3-flip neighborhoods. Note that 1-flip neighborhood is also implemented by using a similar memory structure. Since the main target of our experiment is to compare the effect of neighborhoods, simple implementations are employed for the above three search algorithms, rather than pursuing algorithmic perfection (e.g., long-term memory is not incorporated in TS). The ILS and TS with the restricted r-flip neighborhood NB'_r in our experiment are described as follows.

Algorithm ILS

1. Randomly generate a solution v. Set $v^* := v$.
2. Improve v by applying LS with NB'_r, and let v' be the obtained locally optimal solution.
3. If $f(v') > f(v^*)$, set $v^* := v'$.
4. If the computational time exceeds the given bound, output v^* and stop; otherwise, randomly choose a solution v'' in $NB_\kappa(v^*)$ (κ is a prespecified integer), let $v := v''$, and return to Step 2.

Algorithm TS

1. Randomly generate a solution v. Set $v^* := v$, and $TL := \emptyset$.
2. Find $v' \in NB'_r(v) \backslash TL$ with $f(v') > f(v)$ or $v' \in NB'_r(v)$ with $f(v') > f(v^*)$ using FA strategy. If no such solution exists, let v' be the best solution in $NB_1(v) \backslash TL$.
3. Set $v := v'$, and $TL := \{v\} \cup$ {solutions obtainable from v by flipping at least one variable flipped in the last τ moves} (τ is a prespecified integer). If $f(v') > f(v^*)$, set $v^* := v'$.
4. If the computational time exceeds the given bound, output v^* and stop; otherwise, return to Step 2.

In Step 2 of TS, we restrict the degrading moves to 1-flip neighborhood. The reason is explained as follows. If $f(v') \leq f(v)$ for all $v' \in NB_r(v) \backslash TL$, the best solution in $NB_r(v) \backslash TL$ might be in $NB_r(v) \backslash NB'_r(v)$. In other words, the restricted neighborhood is valid only for improving moves. Therefore, we avoid the use of restricted neighborhood NB'_r ($r = 2,3$) to find the best degrading moves in Step 2 of TS.

4 Experimental Results

The algorithms were coded in C language and run on a workstation Sun Ultra 2 Model 2300 (300 MHz, 1 GB memory), where the computation was restricted on a single processor. For comparison purposes, we also show the results of two local search algorithms called GSAT [18] and WALKSAT (abbreviated as WSAT) [21], tabu search for constraint satisfaction problem (abbreviated as TS-CSP) [12], and an exact algorithm for SAT problem called POSIT [2]. The codes of GSAT, WSAT and POSIT are taken from the web sites of the respective authors `http://www.research.att.com/~selman/` and `ftp://ftp.cis.upenn.edu/pub/freeman/`, and the code of TS-CSP was obtained from the authors.

4.1 Problem Instances

We tested five types of problem instances RNDU, RNDW, GCP, SCP and TTP. The RNDU instances were generated according to [7]. That is, each clause C_i is generated by repeating the following rule until C_i has at least two literals: For each variable x_j, include x_j in C_i with probability $5/n$, and negate it (i.e., $\overline{x}_j \in C_i$) with probability $1/2$. The clauses for RNDW instances were similarly generated, and the integer weights were randomly chosen from [1, 1000] as described in [16]. For each of $n = 1000$ and $m = 7700, 11050$, we generated 10 instances, which are available at our web site `http://www.kuamp.kyoto-u.ac.jp/labs/or/members/yagiura/sat/`. It is known (e.g., [7]) that the expected number of satisfying assignments for this type of instances can be computed easily. In our case, this expectation becomes 1.0 for $m \simeq 11050$. It is observed in literature that no satisfying assignments usually exists for such an instance, which is true for all of our instances with $n = 1000$ and $m = 11050$. On the other hand, satisfying assignments were found for all the instances of $n = 1000$ and $m = 7700$. We also tested 17 RNDW instances of $n = 100$ and $m = 850$, which are available in `http://www.research.att.com/~mgcr/data/maxsat.tar.gz` with the name `jnh1` $\sim$ `jnh19`.

As the second type of problems, we considered the following minimum cost graph coloring problem (GCP). Given a graph with weights on edges and an integer K, we are asked to determine a K-coloring of vertices with the minimum cost, where the cost is the weighted sum of edges having different colors on their end vertices. We transform the GCP instance to the MAX SAT instance as follows. For each edge $\{j_1, j_2\}$ and color k, we create a clause $\{\overline{x}_{j_1,k}, \overline{x}_{j_2,k}\}$ whose weight is the weight of the edge. Let $wsum_j$ be the sum of the weights of the edges adjacent to vertex j, and $wmax$ be the maximum of $wsum_j$ among all vertices j. Then, for each vertex j, we create a clause $\{x_{j,1}, \ldots, x_{j,K}\}$ whose weight is $wmax + 1$. It is easy to see that, for any locally optimal solution v of r-flip neighborhood, exactly one of $v_{j,1}, \ldots, v_{j,K}$ takes value 1 for each vertex

j. Therefore, a feasible solution for the original GCP instance is obtained by coloring each vertex j with color k if $v_{j,k} = 1$. We generated 10 GCP instances with 200 vertices and 2000 edges, where K was set to 4 and edge weights were randomly chosen from [1, 1000]. The size of the resulting MAX SAT formulation is $n = 800$ and $m = 8200$. These instances are available at our web site `http://www.kuamp.kyoto-u.ac.jp/labs/or/members/yagiura/sat/`.

The SCP instances were transformed from 10 set cover instances `scp41` $\sim$ `scp410`, which were taken from the web site of OR-Library (`http://mscmga.ms.ic.ac.uk/jeb/orlib/scpinfo.html`). As a special case, we also tested STS instances $A_{135}, A_{243}, A_{405}, A_{729}$, which are available at `http://www.kuamp.kyoto-u.ac.jp/labs/or/members/yagiura/scp/stcp/` (A_{135} and A_{243} are also available at OR-Library). For details on STS, see [8, 10, 13]. The set cover problem is defined as follows:

$$\begin{aligned} \text{minimize} \quad & \sum_{j=1}^{n'} d_j y_j \\ \text{subject to} \quad & \sum_{j=1}^{n'} a_{ij} y_j \geq 1, \ \forall i = 1, \ldots, m' \\ & y_j \in \{0,1\}, \ \forall j = 1, \ldots, n', \end{aligned} \tag{4}$$

where $d_j \in Z_+$ (the set of positive integers) are the cost coefficients and $a_{ij} \in \{0,1\}$. Let $d_{\max} = \max\{d_j \mid j = 1, \ldots, n'\}$ be the maximum cost coefficient. We transform the SCP instance into MAX SAT formulation by setting $C_i = \{x_j \mid a_{ij} = 1\}$ and $w_i = d_{\max} + 1$, for each $i = 1, \ldots, m'$, and $C_{m'+j} = \{\overline{x}_j\}$ and $w_{m'+j} = d_j$, for each $j = 1, \ldots, n'$, and finally $n = n'$ and $m = m' + n'$. Then, for any locally optimal solution v of r-flip neighborhood, a feasible solution for the original SCP instance is obtained by setting $y = v$. The program to convert the SCP instances to the MAX SAT formulation is available at our web site mentioned above.

The TTP instance is a real world time tabling problem of Kyushu University in Japan, whose description is given in [1]. The size of this instance is rather large ($n = 900$ and $m = 236,539$) in MAX SAT formulation; however, much more compact formulation is possible if it is formulated as a constraint satisfaction problem (CSP) [12].

4.2 Effect of Parameters

In this subsection, we examine the effect of parameters κ, τ and p of ILS, TS and WSAT, respectively. (The parameters κ and τ are explained in Section 3, and p is the parameter called noise in the program code, which is also written as p in [21].) Figures 1 $\sim$ 12 show the average error in % from

the trivial bound $\sum_{i \in M} w_i$, i.e.,

$$\text{error of a solution } v = \left(1 - \frac{f(v)}{\sum_{i \in M} w_i}\right) \times 100\ (\%), \tag{5}$$

against the above parameters for 10 RNDU instances of $n = 1000$ and $m = 11050$, for 10 RNDW instances of $n = 1000$ and $m = 11050$, for 10 GCP instances of $n = 800$ and $m = 8200$, and for 10 SCP instances of $n = 1000$ and $m = 1200$. The algorithms are terminated after 300 seconds for each instance. Figures 13 $\sim$ 15 show the best cost of five runs, allowing 300 seconds for each run, for the TTP instance of $n = 900$ and $m = 236539$. It is observed that ILS with $r = 1, 2, 3$ and TS with $r = 3$ are robust with these parameters; on the other hand, TS with $r = 1, 2$ and WSAT are quite sensitive. Especially, for TS with $r = 1$, the best performance was observed at $\tau = 80$ for RNDU instances (Figure 2); however, $\tau = 120$ was the best for RNDW instances (Figure 5). Note that the only differences between these two instances are the weights w_i. Therefore, to get good results by TS, careful parameter tuning is necessary whenever an instance of different type or size is encountered.

4.3 Comparisons

We examine the performance of the restricted 2 and 3-flip neighborhoods with that of 1-flip neighborhood for MLS, ILS and TS. For comparison purposes, we also show the results of GSAT, WSAT, TS-CSP and POSIT. All of GSAT, WSAT and TS-CSP are based on 1-flip neighborhood. ILS, TS and WSAT include parameters κ, τ and p respectively, which are carefully tuned. Their values are also shown in the tables.

Table 1 shows the average computational time of the tested six algorithms (of the cases in which satisfying assignments are found) for problem SAT of RNDU instances with $n = 1000$ and $m = 7700$, where all instances have satisfying assignments. MLS with $r = 1, 2, 3$ and TS with $r = 3$ could not find any satisfying assignment for all of the 10 instances; and hence, the results are omitted. The results indicate that 1-flip neighborhood is more effective than the proposed 2 and 3-flip neighborhoods, and WSAT and POSIT are much more effective than other tested algorithms.

Tables 2, 3, 5 and 6 then show the average errors in % from the trivial bound (5) for RNDU, RNDW, GCP and SCP instances, respectively. Table 8 shows the number of unsatisfied clauses for the TTP instance. For RNDW instances of $n = 100$ and $m = 850$, optimal solutions are known (e.g., [16]), and average error from the optimal values are shown in Table 4. It is known that no satisfying assignment exists for most of these instances. For such instances, the algorithm POSIT just answers 'no' and does not seek for an assignment with maximum weights; and hence, the results are omitted. Here we note that algorithms GSAT and WSAT are originally developed for unweighted instances,

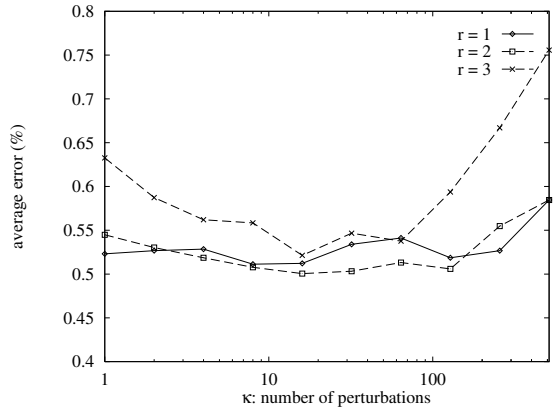

Figure 1: Effect of parameter κ of ILS for RNDU instances of $n = 1000$ and $m = 11050$.

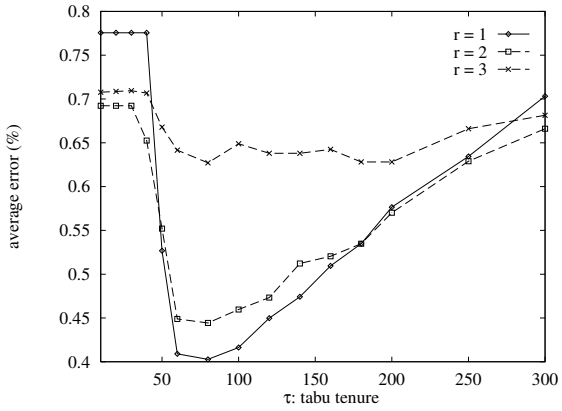

Figure 2: Effect of parameter τ of TS for RNDU instances of $n = 1000$ and $m = 11050$.

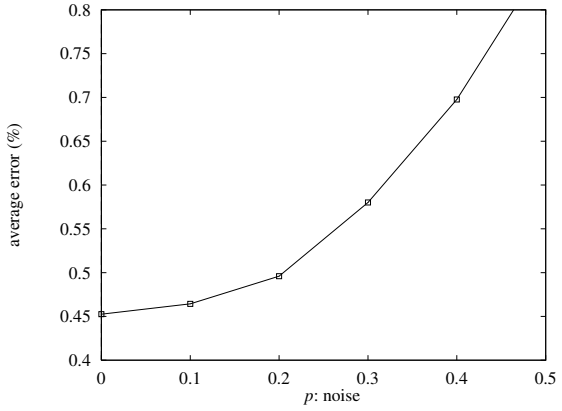

Figure 3: Effect of parameter p of WSAT for RNDU instances of $n = 1000$ and $m = 11050$.

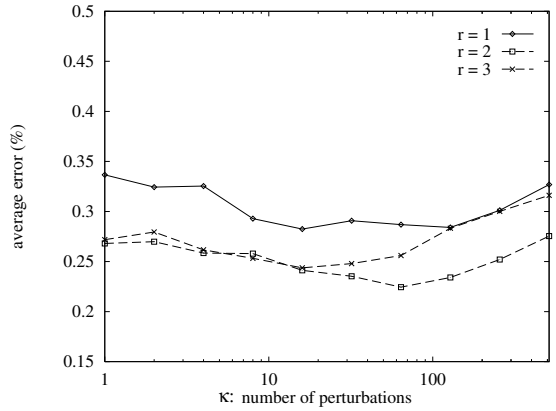

Figure 4: Effect of parameter κ of ILS for RNDW instances of $n = 1000$ and $m = 11050$.

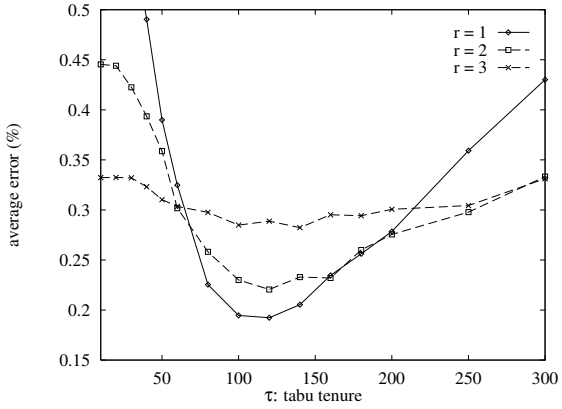

Figure 5: Effect of parameter τ of TS for RNDW instances of $n = 1000$ and $m = 11050$.

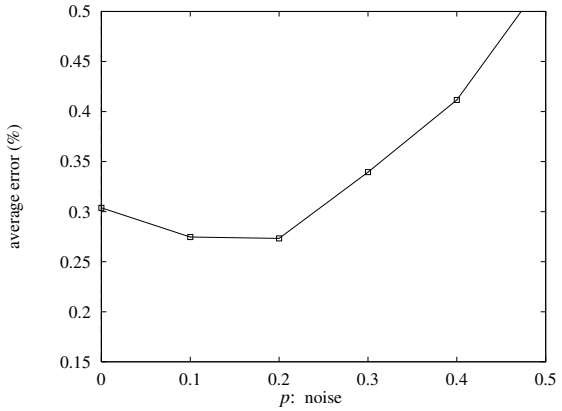

Figure 6: Effect of parameter p of WSAT for RNDW instances of $n = 1000$ and $m = 11050$.

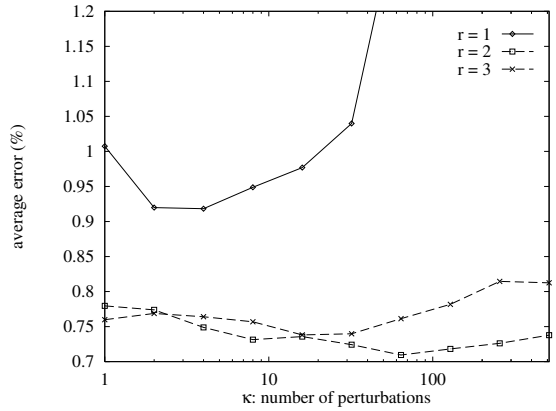

Figure 7: Effect of parameter κ of ILS for GCP instances of $n = 800$ and $m = 8200$.

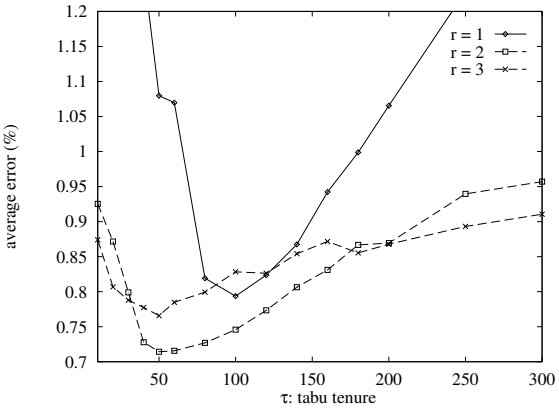

Figure 8: Effect of parameter τ of TS for GCP instances of $n = 800$ and $m = 8200$.

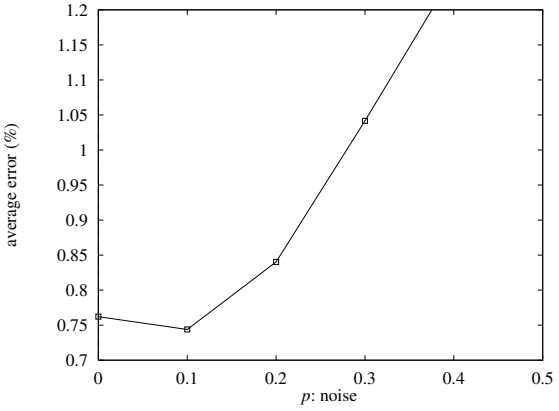

Figure 9: Effect of parameter p of WSAT for GCP instances of $n = 800$ and $m = 8200$.

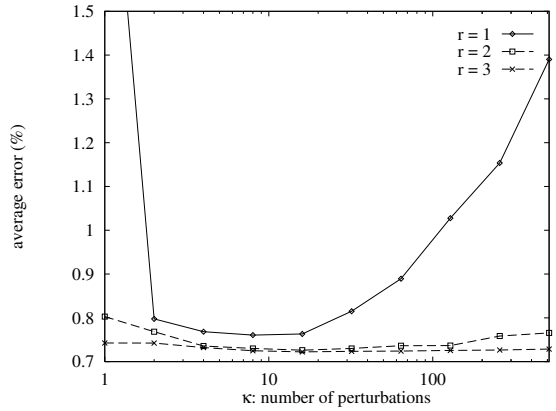

Figure 10: Effect of parameter κ of ILS for SCP instances of $n = 1000$ and $m = 1200$.

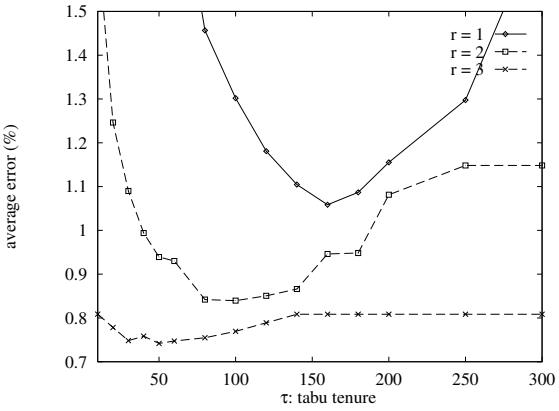

Figure 11: Effect of parameter τ of TS for SCP instances of $n = 1000$ and $m = 1200$.

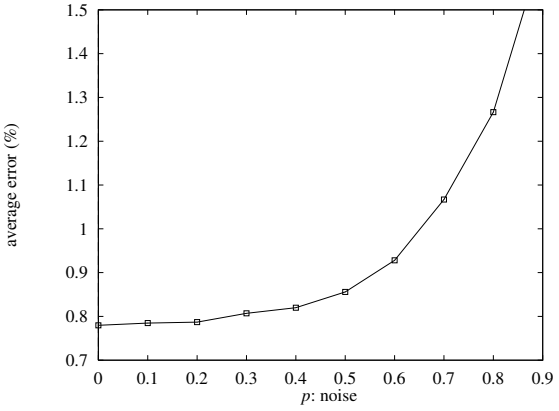

Figure 12: Effect of parameter p of WSAT for SCP instances of $n = 1000$ and $m = 1200$.

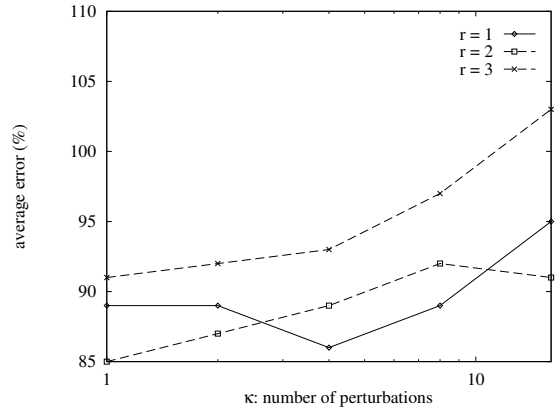

Figure 13: Effect of parameter κ of ILS for TTP instances of $n = 900$ and $m = 236539$.

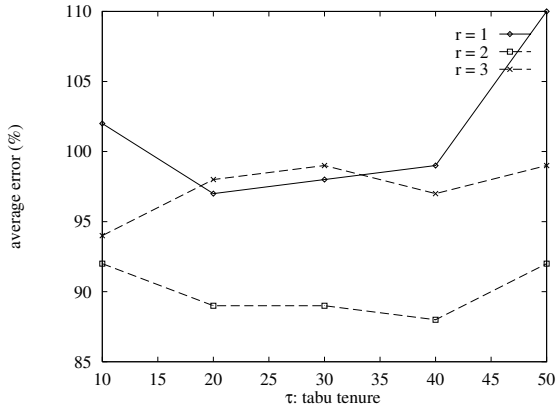

Figure 14: Effect of parameter τ of TS for TTP instances of $n = 900$ and $m = 236539$.

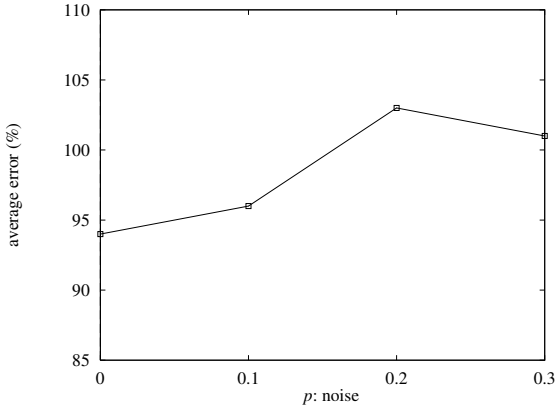

Figure 15: Effect of parameter p of WSAT for TTP instances of $n = 900$ and $m = 236539$.

Table 1: Average computational time in seconds for SAT with 10 RNDU instances ($n = 1000$ and $m = 7700$). The numbers in parentheses denote how many instances were given satisfying assignments within 300 seconds.

ILS			TS					
$r = 1$	$r = 2$	$r = 3$	$r = 1$	$r = 2$	GSAT	WSAT	TS-CSP	POSIT
$\kappa = 16$	$\kappa = 16$	$\kappa = 16$	$\tau = 160$	$\tau = 180$		$p = 0.5$		
49.5 (8)	113.6 (8)	134.4 (4)	22.0 (9)	82.5 (8)	37.7 (8)	0.13 (10)	37.5 (9)	3.7 (10)

and weights are represented by multiplexing each clause C_i w_i times.

Now Table 2 shows the results for 10 RNDU instances of $n = 1000$ and $m = 11050$. We can observe that 3-flip neighborhood gives the best results with respect to the error for MLS, and 2-flip gives the best results for ILS; however, 1-flip is the best choice for TS. It is also observed that the performance of TS with $r = 1,2$ is better than GSAT, WSAT and TS-CSP. Similar results are observed in Table 3 for RNDW instances of $n = 1000$ and $m = 11050$.

Table 2: Results for 10 RNDU instances of $n = 1000$ and $m = 11050$.

algorithm	r	error (%)	#moves	#restarts	#moves/trial	CPU secs.
MLS	1	1.2606	2079353.8	7441.0	279.4	300.0
MLS	2	0.8154	297311.8	906.8	327.9	300.2
MLS	3	0.7231	15367.3	42.9	358.2	303.7
ILS ($\kappa = 8$)	1	0.5113	1736761.0	1.0		300.0
ILS ($\kappa = 16$)	2	0.5005	117468.7	1.0		300.0
ILS ($\kappa = 16$)	3	0.5213	3285.6	1.0		301.0
TS ($\tau = 80$)	1	0.4027	2022823.5	1.0		300.0
TS ($\tau = 80$)	2	0.4443	33335.2	1.0		300.0
TS ($\tau = 80$)	3	0.6271	992.2	1.0		300.7
GSAT		0.5484	5000000.0	1.0		290.9
WSAT ($p = 0$)		0.4525	25000000.0	1.0		306.5
TS-CSP		0.4606	36374.7	1.0		300.0

Table 4 shows the results for 17 RNDW instances of $n = 100$ and $m = 850$. The results reported in [17] are also included as GRASP for comparison purpose, where the results with 100,000 and 10,000,000 restarts are shown. As the exact values of computational time for GRASP are not reported in [17], we show rough estimates from Figure 9 of [17]. We can observe that 2-flip

Table 3: Results for 10 RNDW instances of $n = 1000$ and $m = 11050$.

algorithm	r	error (%)	#moves	#restarts	#moves/trial	CPU secs.
MLS	1	0.6461	2349333.3	5500.0	427.2	300.0
MLS	2	0.3525	320681.5	586.8	546.5	300.3
MLS	3	0.3056	14760.5	22.8	647.4	305.2
ILS ($\kappa = 16$)	1	0.2825	1956733.9	1.0		300.0
ILS ($\kappa = 64$)	2	0.2244	208575.2	1.0		300.1
ILS ($\kappa = 16$)	3	0.2437	4724.1	1.0		302.4
TS ($\tau = 120$)	1	0.1923	568919.4	1.0		300.0
TS ($\tau = 120$)	2	0.2205	26602.0	1.0		300.1
TS ($\tau = 140$)	3	0.2823	1229.1	1.0		301.4
GSAT		0.7348	15000.0	1.0		385.8
WSAT ($p = 0.2$)		0.2733	100000.0	1.0		390.6
TS-CSP		0.2812		1.0		540.0

neighborhood gives the best results with respect to the error for all of MLS, ILS and TS, which is similar to Tables 2 and 3. The solution quality of GRASP with 10,000,000 restarts is the best among the tested algorithms; however, it consumes much larger computational time. With this exception, the performance of ILS with $r = 2$ and TS with $r = 1$ and 2 is better than other tested algorithms.

Table 5 shows the results for 10 GCP instances of $n = 800$ and $m = 8200$. We can observe that 2-flip neighborhood gives the best results with respect to the error for all of MLS, ILS and TS, among which ILS with 2-flip neighborhood gives the best performance. This result is better than GSAT and WSAT, but worse than TS-CSP. A conceivable reason of this is that the solutions space of CSP formulation directly corresponds to colorings of vertices, but the solution space of MAX SAT formulation does not.

Table 6 shows the results for 10 SCP instances of $n = 1000$ and $m = 1200$. We can observe that 3-flip neighborhood gives the best results with respect to the error for all of MLS, ILS and TS, among which ILS with 3-flip neighborhood gives the best performance.

For STS instances A_{135} and A_{243}, the best costs reported in [12, 13] are 103 and 198, respectively. For A_{243}, the solution with the cost 198 was found very often (i.e., under various parameter settings) within a few seconds. For A_{135}, the solution with the cost 103 was found, but not very often; however, the cost 104 was found very frequently within a few seconds. As there were not much differences among the tested algorithms, we just show in Table 7 the best costs found by our TS to encourage future challenge of solving larger STS instances.

Table 8 shows the results for the TTP instance of $n = 900$ and $m = 236539$. The result reported

Table 4: Results for 17 RNDW instances of $n = 100$ and $m = 850$.

algorithm	r	error (%)	#moves	#restarts	#moves/trial	CPU secs.
MLS	1	0.0598	314982.5	8075.1	39.0	30.0
MLS	2	0.0081	43643.0	887.0	49.2	21.1
MLS	3	0.0147	3676.4	64.7	56.9	21.9
ILS ($\kappa = 32$)	1	0.0170	133477.7	1.0		19.6
ILS ($\kappa = 16$)	2	0.0052	15702.4	1.0		10.9
ILS ($\kappa = 8$)	3	0.0064	1795.4	1.0		15.8
TS ($\tau = 20$)	1	0.0026	95857.0	1.0		11.0
TS ($\tau = 20$)	2	0.0025	15043.9	1.0		12.1
TS ($\tau = 20$)	3	0.0078	1000.5	1.0		16.3
GRASP		0.0134		100,000		($\approx 1 \times 10^2$)†
		0.0011		10,000,000		($\approx 1 \times 10^4$)†
GSAT		0.4890	2000.0	1.0		33.1
WSAT ($p = 0.5$)		0.0099	12264.2	1.0		32.6
TS-CSP		0.0055		5.0		100.0

†CPU seconds on a Silicon Graphics Challenge computer (196MHz MIPS R10000 processor).

Table 5: Results for 10 GCP instances of $n = 800$ and $m = 8200$.

algorithm	r	error (%)	#moves	#restarts	#moves/trial	CPU secs.
MLS	1	2.6682	1561888.9	6945.0	224.9	300.0
MLS	2	0.7979	570302.9	984.1	579.5	300.2
MLS	3	0.8399	40096.2	72.3	554.6	301.4
ILS ($\kappa = 4$)	1	0.9183	1082669.5	1.0		300.0
ILS ($\kappa = 64$)	2	0.7095	310070.2	1.0		300.1
ILS ($\kappa = 16$)	3	0.7382	9251.9	1.0		300.7
TS ($\tau = 100$)	1	0.7936	465733.1	1.0		300.0
TS ($\tau = 50$)	2	0.7143	36756.2	1.0		300.0
TS ($\tau = 50$)	3	0.7658	1975.6	1.0		300.6
GSAT		1.8191	30000.0	1.0		401.5
WSAT ($p = 0.1$)		0.7439	70000.0	1.0		333.2
TS-CSP		0.6526	915431.7	3.0		300.0

Table 6: Results for 10 SCP instances of $n = 1000$ and $m = 1200$.

algorithm	r	error (%)	#moves	#restarts	#moves/trial	CPU secs.
MLS	1	6.8618	14830875.3	34525.8	429.6	300.0
MLS	2	1.2164	2698703.0	5460.4	494.2	300.0
MLS	3	0.7315	51890.7	95.2	545.1	301.0
ILS ($\kappa = 8$)	1	0.7606	4369174.8	1.0		300.0
ILS ($\kappa = 16$)	2	0.7261	186745.3	1.0		300.0
ILS ($\kappa = 16$)	3	0.7223	4437.3	1.0		300.8
TS ($\tau = 160$)	1	1.0586	384690.2	1.0		300.0
TS ($\tau = 100$)	2	0.8396	82043.7	1.0		300.0
TS ($\tau = 50$)	3	0.7418	2076.2	1.0		300.7
GSAT		1.5009	600000.0	1.0		461.4
WSAT ($p = 0$)		0.7797	6000000.0	1.0		368.7
TS-CSP		0.7327	210691.8	1.0		300.0
optimal		0.7214				

Table 7: Results for the STS instances of various sizes.

instance	best cost
A_{135}	103
A_{243}	198
A_{405}	337
A_{729}	617

Table 8: Results for the TTP instance of $n = 900$ and $m = 236539$.

algorithm	r	#unsat	#moves	#restarts	#moves/trial	CPU secs.
MLS	1	124	416164	869	478.9	1500
MLS	2	105	207052	379	546.3	1500
MLS	3	99	33357	62	538.0	1500
ILS ($\kappa = 4$)	1	86	490890	5		1500
ILS ($\kappa = 1$)	2	85	126068	5		1500
ILS ($\kappa = 1$)	3	91	5711	5		1500
TS ($\tau = 20$)	1	97	788431	5		1500
TS ($\tau = 40$)	2	88	147049	5		1500
TS ($\tau = 10$)	3	94	4599	5		1500
CIKM		93	5000000			—
GSAT		125	3000000	5		2497
WSAT ($p = 0$)		94	3000000	5		1517
TS-CSP[1]		85	101555	10		1800

[1]Size of CSP formulation: 900 0-1 variables, 774 constraints.

in [1] is also indicated as CIKM for comparison purpose. We can observe that 3-flip neighborhood is best for MLS, and 2-flip is best for ILS and TS. The overall best performance is given by ILS with 2-flip neighborhood, which is competitive with TS-CSP. Here we note that the size of CSP formulation of this instance has 900 0-1 variables and 774 constraints, which is much smaller than the MAX SAT formulation.

In summary, we have the following observation.

1. The best neighborhood size r changes with problem types and with algorithms.

2. For randomly generated instances, 1-flip neighborhood under the framework of TS and WSAT gives the best performance; however, their performances are quite sensitive to the parameter values.

3. For problems with special structures, such as the graph coloring problem, the set covering problem and the time tabling problem, the restricted 2 and 3-flip neighborhoods exhibit good prospects under various metaheuristic frameworks.

5 Conclusion

In this paper, we examined the effectiveness of the efficient implementations of 2 and 3-flip neighborhoods for the MAX SAT problem proposed in [23]. These neighborhoods were tested using simple metaheuristic frameworks: (i) random multi-start local search (MLS), (ii) iterated local search (ILS), and (iii) tabu search (TS). These algorithms were compared on five types of problem instances: (1) random instances with unit weight (RNDU), (2) random instances with general weight (RNDW), (3) graph coloring instances (GCP), (4) set covering instances (SCP), and (5) a time tabling instance (TTP). The results indicate that the proposed 2 and 3-flip neighborhoods are effective for structured problems such as GCP, SCP and TTP, while 1-flip neighborhood gives better performance for random instances such as RNDU and RNDW. The best choice for the neighborhood size changes with problem types and with algorithms like this; however, from the view point of robustness, ILS with 2-flip neighborhood seems quite attractive.

Acknowledgement

The authors are grateful to Kazuo Iwama, Kyoto University, who gave us their TTP instance and also had valuable discussion. The authors are indebted to Koji Nonobe, Kyoto University, for providing us the computational results by TS-CSP. The authors are also grateful to Wolfgang Slany at Technische Universität Wien and Takeaki Uno at Tokyo Institute of Technology for their valuable discussion. This research was partially supported by Scientific Grant-in-Aid, by the Ministry of Education, Science and Culture of Japan.

References

[1] B. Cha, K. Iwama, Y. Kambayashi and S. Miyazaki, "Local Search Algorithms for Partial MAXSAT," *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference* (1997) 263–268.

[2] J.W. Freeman, "Improvements to Propositional Satisfiability Search Algorithms," Dissertation, University of Pennsylvania, 1995.

[3] A.S. Fukunaga, "Variable-Selection Heuristics in Local Search for SAT," *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference* (1997) 275–280.

[4] M.R. Garey and D.S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, 1979.

[5] J. Gu, "Efficient Local Search for Very Large-Scale Satisfiability Problems," *SIGART Bulletin*, 3 (1992) 8–12.

[6] P. Hansen and B. Jaumard, "Algorithms for the Maximum Satisfiability Problem," *Computing*, 44 (1990) 279–303.

[7] J.N. Hooker and C. Fedjki, "Branch-and-Cut Solution of Inference Problems in Propositional Logic," *Annals of Mathematics and Artificial Intelligence*, 1 (1990) 123–139.

[8] N. Karmarkar, M.G.C. Resende and K.G. Ramakrishnan, "An Interior Point Algorithm to Solve Computationally Difficult Set Covering Problems," *Mathematical Programming*, 52 (1991) 597–618.

[9] D.E. Knuth, *The Art of Computer Programming, Vol. 3, Sorting and Searching*, Addison-Wesley, 1973.

[10] C. Mannino and A. Sassano, "Solving Hard Set Covering Problems," *Operations Research Letters*, 18 (1995) 1–5.

[11] B. Mazure, L. Saïs and É. Grégoire, "Tabu Search for SAT," *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Innovative Applications of Artificial Intelligence Conference* (1997) 281–285.

[12] K. Nonobe and T. Ibaraki, "A Tabu Search Approach to the CSP (Constraint Satisfaction Problem) as a General Problem Solver," *European Journal of Operational Research, Special Issue on Tabu Search*, 106 (1998) 599–623.

[13] M.A. Odijk and H. van Maaren, "Improved Solutions to the Steiner Triple Covering Problem," *Information Processing Letters*, 65 (1998) 67–69.

[14] P.M. Pardalos, L.S. Pitsoulis and M.G.C. Resende, "A Parallel GRASP for MAX-SAT Problems," *Lecture Notes in Computer Science*, 1180 (1996) 575–585.

[15] M.G.C. Resende and T.A. Feo, "A GRASP for Satisfiability," in: D.S. Johnson and M.A. Trick, eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, Vol. 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science* (American Mathematical Society, 1996) 499–520.

[16] M.G.C. Resende, L.S. Pitsoulis and P.M. Pardalos, "Approximate Solution of Weighted MAX-SAT Problems Using GRASP," in: D.-Z. Du, J. Gu and P.M. Pardalos, eds., *Satisfiability Problem: Theory and Applications*, Vol. 35 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science* (American Mathematical Society, 1997) 393–405.

[17] M.G.C. Resende, L.S. Pitsoulis and P.M. Pardalos, "FORTRAN Subroutines for Computing Approximate Solutions of Weighted MAX-SAT Problems Using GRASP," *Discrete Applied Mathematics*, to appear.

[18] B. Selman, H. Levesque and D. Mitchell, "A New Method for Solving Hard Satisfiability Problems," *Proceedings of the 10th National Conference on Artificial Intelligence* (1992) 440–446.

[19] B. Selman and H.A. Kautz, "An Empirical Study of Greedy Local Search for Satisfiability Testing," *Proceedings of the 11th National Conference on Artificial Intelligence* (1993) 46–51.

[20] B. Selman and H.A. Kautz, "Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems," *Proceedings of the 13th International Joint Conference on Artificial Intelligence* (1993) 290–295.

[21] B. Selman, H.A. Kautz and B. Cohen, "Noise Strategies for Improving Local Search," *Proceedings of the 12th National Conference on Artificial Intelligence* (1994) 337–343.

[22] B. Selman, H.A. Kautz and B. Cohen, "Local Search Strategies for Satisfiability Testing," in: D.S. Johnson and M.A. Trick, eds., *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, Vol. 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science* (American Mathematical Society, 1996) 521–531.

[23] M. Yagiura and T. Ibaraki, "Analyses on the 2 and 3-Flip Neighborhoods for the MAX SAT," *Journal of Combinatorial Optimization*, 3 (1999) 95–114.