Analyses on the 2 and 3-Flip Neighborhoods for the MAX SAT

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}@kuamp.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 $r \geq 2$, and propose, for $r = 2, 3$, new implementations that reduce the number of candidates in the neighborhood without sacrificing the solution quality. For 2-flip (resp., 3-flip) neighborhood, we show that its expected size is $O(n + m)$ (resp., $O(m + t^2 n)$), which is usually much smaller than the original size $O(n^2)$ (resp., $O(n^3)$), where n is the number of variables, m is the number of clauses and t is the maximum number of appearances of one variable. Computational results tell that these estimates by the expectation well represent the real performance.

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

1 Introduction

Given n 0-1 variables $x_j, j \in N$, m clauses $C_i, i \in M$, and weights $w_i (\geq 0), i \in M$, 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 to maximize is given by

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

On the other hand, 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 algorithms based on the local search (abbreviated as LS) for SAT and MAX SAT have been proposed [6, 7, 13]. 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 a better 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 $NB(v)$.

Deviating from the basic LS algorithms, there are many successful LS type algorithms, such as [1, 3, 5, 6, 7, 10, 13, 14, 15, 16, 17], which are generally called metaheuristic algorithms. One of the important features of these algorithms is that they do not stop at locally optimal solutions but allow the subsequent moves to nonimproving solutions. In making nonimproving moves, not only FA and BA moves but also other move strategies are possible, as extensively studied (e.g., [5]).

However, in this paper we restrict our attention to the basic LS, since our objective is to clarify theoretical aspects of the neighborhood structure.

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, 7, 10, 11, 12, 13, 14, 15, 16, 17].

In this paper, we consider r-flip neighborhood for general r, and propose efficient implementations for these neighborhoods, which make use of the memory structure as described in Section 3. Usually the quality of locally optimal solutions improves if larger neighborhood is used (e.g., see Figure 1); however, the computational time to search

NB_r increases exponentially with r, since $|NB_r| = O(n^r)$ holds in general. To overcome this, we propose, for $r = 2$ and 3, a method to reduce the number of candidates in the neighborhood without sacrificing the solution quality.

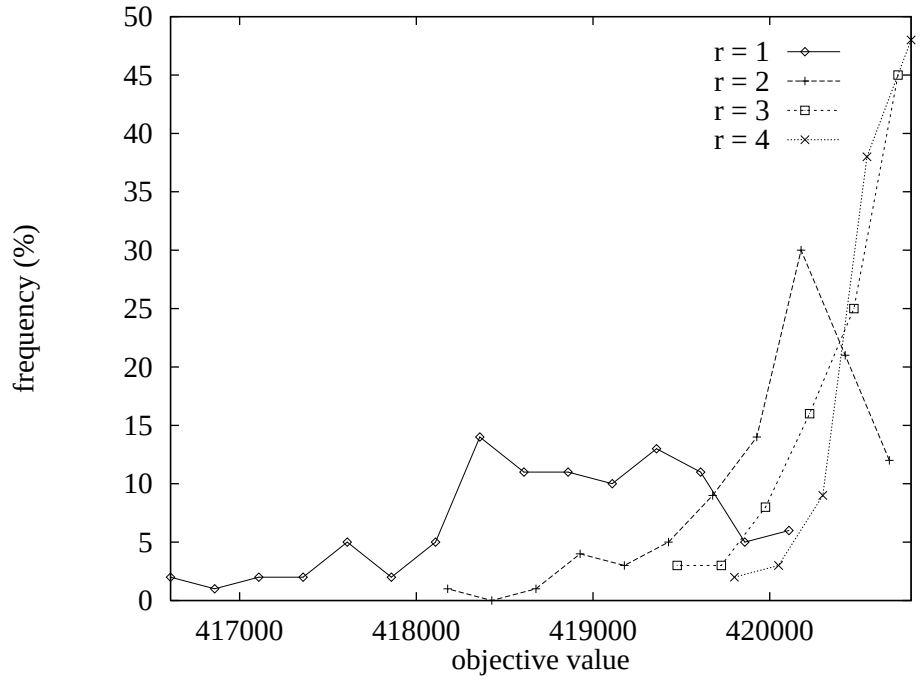

Figure 1: Distribution of the objective values of 100 locally optimal solutions with four neighborhoods NB_r ($r = 1, \ldots, 4$) for a randomly generated weighted MAX SAT instance of $n = 100$ (taken from the ftp site mentioned in [12]).

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 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. But this computational effort can be reduced if we use a clever memory structure. Here we assume that each memory access takes $O(1)$ time, and necessary memory cells can be stored in the linear order, whose precise definition and validity will be discussed in Section 3. With this memory structure, in the worst case, $O(2^r)$ time is needed for each evaluation and $O(rtl^r)$ time is required to update the memory structure after a move. Hence, one-round worst-case time becomes $O(2^r|NB_r| + rtl^r) = O(2^r n^r + rtl^r)$. The size of memory space needed for

this algorithm is $O(n + ml^r)$.

Then we introduce the restricted 2 and 3-flip neighborhoods NB'_2 and NB'_3, which do not sacrifice solution quality. It will be shown in Section 3 that the size of NB'_2 in the worst case is $O(n + ml)$, and that of NB'_3 is $O(ml^3 + t^2l^2n)$. Therefore, the worst case one-round time with the restricted neighborhoods becomes $O(n + ml + tl^2)$ if $r = 2$ and $O(ml^3 + t^2l^2n)$ if $r = 3$. These neighborhood sizes may seem rather complicated; however, they are usually much smaller than the original sizes as t and l are usually very small.

Furthermore, the situation becomes much better if we consider the average-case complexity. Let r be a fixed constant. Then it will be shown that the expected time needed for each update of the memory structure is $O(t)$, and the expected memory requirement is $O(n + m)$. It will also be shown in Section 3 that the expected size of NB'_2 is $O(n + m)$, and that of NB'_3 is $O(m + t^2n)$. Therefore, the expected one-round time with the restricted neighborhood becomes $O(n + m)$ if $r = 2$. In the case of $r = 3$, although additional computation is needed for the restriction of the neighborhood, the expected one-round time is kept as $O(m + t^2n)$. 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 1-flip neighborhood, similar result is already reported in [3].

Computational experiments for problem instances with up to $n = 1000$ are conducted to see the effectiveness of these neighborhood restrictions. It is observed that the above probabilistic estimates represent the real performance of LS well. Computational experiments to evaluate the effectiveness of these neighborhoods are also conducted. Here we note that the idea of enlarging the neighborhood does not conflict with the strategy of allowing moves to nonimproving solutions. Therefore, the restricted 2 and 3-flip neighborhoods can be incorporated into various metaheuristic algorithms such as tabu search. According to our computational experiment, it is observed that the proposed 2 and 3-flip neighborhood search algorithms with the restricted neighborhoods are more effective than 1-flip neighborhood under various metaheuristic frameworks if about the same computational time is allowed. The details of these experiments, as well as comparison with other algorithms, can be found in [20] (some results are also available in [19]). In this paper, we focus on the average-case estimation of the sizes of NB'_2 and NB'_3, and the time to manipulate them, as well as empirical justification of these theoretical results.

2 Preliminaries

For convenience, let

$$\begin{aligned} CI_j &= \{i \in M |\ x_j \in C_i \text{ or } \overline{x}_j \in C_i\}, \quad (3) \\ t &= \max_{j \in N} |CI_j|, \\ VI_i &= \{j \in N |\ x_j \in C_i \text{ or } \overline{x}_j \in C_i\}, \quad (4) \\ l &= \max_{i \in M} |VI_i| = \max_{i \in M} |C_i|, \end{aligned}$$

where t denotes the maximum number of appearances of a variable and l denotes the maximum number of literals in a clause. These always satisfy $t \leq m$ and $l \leq n$, and in most cases $t \ll m$ and $l \ll n$. For a vector $v \in \{0,1\}^n$ and a subset $S \subseteq N$, 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*,

$$\begin{aligned} \Delta f(v, S) &= f(v \updownarrow S) - f(v) \quad (5) \\ &= \sum_{i \in M} \Delta f_i(v, S), \end{aligned}$$

where

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

3 LS Implementation with Memory

In this section, we propose an efficient method to search solutions in the neighborhood by making use of the memory structure. The memory stores the information that enables us to evaluate each flip gain in $O(1)$ time. This information is then used to reduce the number of candidates in the neighborhood, as will be explained in 3.4 and 3.5.

3.1 Calculation of the Flip Gain

In this subsection, we explain the above memory structure in detail. We first prepare flip gains of all candidates in the 1-flip neighborhood. For the candidates in 2-flip neighborhood, instead of storing the flip gains, we evaluate the flip gain $\Delta f(v, \{j_1, j_2\})$ as the sum of $\Delta f(v, \{j_1\})$, $\Delta f(v, \{j_2\})$ and the adjustment $g(v, \{j_1, j_2\})$, whose definition will be given by (7) later. For example, consider a clause C_i where $j_1, j_2 \in VI_i$ and $P_i(v) = \emptyset$. In this case, $\Delta f_i(v, \{j_1\}) = \Delta f_i(v, \{j_2\}) = w_i$ holds, and we need the adjustment term $-w_i$ so that we obtain $\Delta f_i(v, \{j_1, j_2\}) = \Delta f_i(v, \{j_1\}) + \Delta f_i(v, \{j_2\}) -$

$w_i = w_i$. The value $g(v, \{j_1, j_2\})$ represents the sum of such adjustment for all clauses C_i. By storing such adjustments, it will be shown below that the flip gain of each solution 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 $j_1, j_2 \in VI_i$ holds 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 in the average case. This idea can be generalized to the case of $r \geq 3$ by using the principle of inclusion and exclusion.

In general, let

$$g_i(v, S) = \begin{cases} (-1)^{|S|-|P_i(v)|+1}\, w_i, & \text{if } P_i(v) \subseteq S \subseteq VI_i \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

and

$$\begin{aligned} g(v, S) &= \sum_{i \in M} g_i(v, S) \\ &= \sum_{i|\ i \in \cap_{j \in S} CI_j, P_i(v) \subseteq S} (-1)^{|S|-|P_i(v)|+1}\, w_i. \end{aligned} \tag{7}$$

Here, it is noted that $g(v, \{j\}) = \Delta f(v, \{j\})$ holds for all $j \in N$.

Lemma 3.1 *The flip gain for a $v \in \{0,1\}^n$ and $S \neq \emptyset$ can be obtained by*

$$\Delta f(v, S) = \sum_{S' \subseteq S, S' \neq \emptyset} g(v, S'). \tag{8}$$

Proof: Let $S_i = S \cap VI_i$. If $S_i = \emptyset$, both $\Delta f_i(v, S)$ and $\sum_{S' \subseteq S, S' \neq \emptyset} g_i(v, S')$ are 0; otherwise we consider the following three cases.

Case 1 ($|P_i(v)| = 0$):

$$\begin{aligned} \sum_{S' \subseteq S, S' \neq \emptyset} g_i(v, S') &= \sum_{S' \subseteq S_i, S' \neq \emptyset} (-1)^{|S'|+1} w_i \\ &= w_i \sum_{k=1}^{|S_i|} (-1)^{k+1} \binom{|S_i|}{k} \\ &= w_i \left(-(1-1)^{|S_i|} + \binom{|S_i|}{0} \right) \\ &= w_i \\ &= \Delta f_i(v, S). \end{aligned} \tag{9}$$

Case 2 ($1 \leq |P_i(v)| \leq |S_i|-1$): In this case, $\Delta f_i(v,S) = 0$. If $P_i(v) \not\subseteq S_i$, $\sum_{S' \subseteq S, S' \neq \emptyset} g_i(v,S')$ is clearly 0 by definition (6); otherwise

$$\begin{aligned}
\sum_{S' \subseteq S, S' \neq \emptyset} g_i(v,S') &= \sum_{P_i(v) \subseteq S' \subseteq S_i} (-1)^{|S'|-|P_i(v)|+1} w_i \\
&= w_i \sum_{k=0}^{|S_i|-|P_i(v)|} (-1)^{k+1} \binom{|S_i|-|P_i(v)|}{k} \\
&= 0.
\end{aligned}$$

Case 3 ($|P_i(v)| = |S_i|$): If $P_i(v) = S_i$, $\Delta f_i(v,S) = -w_i$ and $\sum_{S' \subseteq S, S' \neq \emptyset} g_i(v,S') = -w_i$; otherwise $\Delta f_i(v,S) = 0$ and $\sum_{S' \subseteq S, S' \neq \emptyset} g_i(v,S') = 0$ by definitions.

Thus the flip gain can be obtained by

$$\begin{aligned}
\Delta f(v,S) &= \sum_{i \in M} \Delta f_i(v,S) \\
&= \sum_{i \in M} \sum_{S' \subseteq S, S' \neq \emptyset} g_i(v,S') \\
&= \sum_{S' \subseteq S, S' \neq \emptyset} \sum_{i \in M} g_i(v,S') \\
&= \sum_{S' \subseteq S, S' \neq \emptyset} g(v,S').
\end{aligned}$$

□

The important point is that we keep only nonzero $g(v,S)$ in the memory. Here we assume the following property, whose validity will be discussed in Subsection 3.6.

Assumption 3.1 *Each access to $g(v,S)$ is possible in $O(1)$ time and all nonzero $g(v,S)$ necessary for neighborhood $NB_r(v)$ can be stored in $O(|\{S \subseteq N|\ g(v,S) \neq 0$ and $|S| \leq r\}|)$ space.*

As will be shown in Subsection 3.3, the expected memory size can be estimated as $O(n+m)$ for any constant r. Denote by $\hat{g}(S)$ the value of $g(v,S)$ stored in the memory for the current solution v. The updates of $\hat{g}(S)$ for all S corresponding to the moves from v to v' with $|D(v,v')| \leq r$ are done as follows.

Algorithm UPDATE
for each $i \in \cup_{j \in D(v,v')} CI_j$ **do**
  for each S such that $P_i(v) \subseteq S \subseteq VI_i$ and $1 \leq |S| \leq r$ **do**
    Set $\hat{g}(S) := \hat{g}(S) - g_i(v,S)$
  end

for each S such that $P_i(v') \subseteq S \subseteq VI_i$ and $1 \leq |S| \leq r$ **do**
Set $\hat{g}(S) := \hat{g}(S) + g_i(v', S)$
end
end.

The computation time needed for this is $O(rtl^r)$ in the worst case, since

$$\left| \bigcup_{j \in D(v,v')} CI_j \right| \leq rt$$

and

$$\begin{aligned} &|\{S \subseteq N |\ P_i(v) \subseteq S \subseteq VI_i \text{ and } 1 \leq |S| \leq r\}| \\ &\qquad \leq \ |\{S \subseteq N |\ S \subseteq VI_i \text{ and } 1 \leq |S| \leq r\}| \\ &\qquad \leq \ \sum_{r'=1}^{r} \binom{l}{r'} = O(l^r) \end{aligned}$$

hold for any $v \in \{0,1\}^n$ and $i \in M$. By (8), $\Delta f(v, S)$ can be calculated in $O(2^r)$ time for each S with $|S| \leq r$. Therefore, one-round time is $O(|NB_r|2^r + rtl^r) = O(n^r 2^r + rtl^r)$ in the worst case.

The most expensive computation to update $\hat{g}(S)$ occurs if either $|P_i(v)| = 0$ or $|P_i(v')| = 0$ holds, where the possible number of S to be considered in the inner loop of the above algorithm UPDATE is $\binom{|C_i|}{r} = O(l^r)$. It is pointed out by Uno [18] that this time complexity might be reduced by postponing such expensive computation until it becomes necessary to compute $g(v, S)$, though the worst case one-round time will not change. The details of this idea is rather complicated and we have not tried it yet; however, it is one of the future possibilities of our research.

Note: An alternative idea is to store the values of $\Delta f(v, S)$ for all S with $|S| \leq r$. With this scheme, we can also evaluate $f(v, S)$ of each solution in $NB_r(v)$ in $O(1)$ time. In the case of $r = 1$, this is a good idea, since the necessary memory size is $O(n)$ and the update of such data for one move can be done in $O(tl)$ time (the algorithm to realize this time complexity will be discussed later). This memory structure for $r = 1$ is quite straightforward, and may be used in most of the algorithms based on LS cited in Section 1, although we could not find any descriptions about such implementation details in most papers except [3]. However, if $r \geq 2$, this idea does not work by the following two reasons. The first is that it requires large memory space. Roughly speaking, even if we restrict the memory to those with $\Delta f(v, S) \neq 0$, it is expected that $|\{j \in N |\ \Delta f(v, \{j\}) \neq 0\}| =$

$\Omega(n)$, and hence, $|\{S \subseteq N |\ \Delta f(v,S) \neq 0, |S| \leq r\}| = \Omega(n^r)$. The second is that the computational time needed for the update is too much. Suppose the value of $\Delta f(v,\{j\})$ has changed in the move from v to v', i.e., $\Delta f(v',\{j\}) \neq \Delta f(v,\{j\})$. Then, again roughly speaking, almost all $S \subseteq N$ with $|S| \leq r$ and $j \in S$ satisfy $\Delta f(v',S) \neq \Delta f(v,S)$. Therefore, $\Omega(n^{r-1})$ time is needed to update the memory even for a small change in the flip gain.

3.2 Expected Time Needed to Update Memory

In this subsection, we consider the expected time needed to update $\hat{g}(S)$ in one move from v to v'. Let $\tau_r(v)$ be the number of memory accesses (i.e., the number of updates of $\hat{g}(S)$) by the algorithm UPDATE in Subsection 3.1 for the r-flip neighborhood. Then the following lemma gives its worst-case and expected bounds.

Lemma 3.2 *The above $\tau_r(v)$ has worst-case bound of $O(rtl^r)$ for any v. If we assume that the current solution v is randomly chosen from $\{0,1\}^n$ with probability $1/2^n$, then $E(\tau_r(v)) = O(t)$ holds for any constant r.*

Proof: The worst case time complexity was already shown in the previous subsection, after the description of UPDATE. To consider the expected bound, let $D(v,v') = \{j_1,\ldots,j_r\}$. For simplicity, we consider the number of memory accesses by the following algorithm, which is larger than that required by algorithm UPDATE: Apply UPDATE for the move from v to $v\updownarrow\{j_1\}$, from $v\updownarrow\{j_1\}$ to $v\updownarrow\{j_1,j_2\}$, ..., and from $v\updownarrow\{j_1,\ldots,j_{r-1}\}$ to $v\updownarrow\{j_1,\ldots,j_r\}$, sequentially. Let

$$\begin{aligned} T_{r'hi} &= \left|\{S \subseteq N |\ P_i(v\updownarrow\{j_1,\ldots,j_{h-1}\}) \subseteq S \subseteq VI_i \text{ and } |S| = r'\}\right| \\ &\quad + \left|\{S \subseteq N |\ P_i(v\updownarrow\{j_1,\ldots,j_h\}) \subseteq S \subseteq VI_i \text{ and } |S| = r'\}\right|, \end{aligned} \tag{10}$$

which is the random variable representing the number of memory accesses in the loop of $i \in CI_{j_h}$ of UPDATE for the move from $v\updownarrow\{j_1,\ldots,j_{h-1}\}$ to $v\updownarrow\{j_1,\ldots,j_h\}$ (here we assume that $v\updownarrow\emptyset = v$). Then, by the linearity of expectation,

$$E\left(\tau_r(v)\right) \leq E\left(\sum_{r'=1}^{r}\sum_{h=1}^{r}\sum_{i\in CI_{j_h}} T_{r'hi}\right) = \sum_{r'=1}^{r}\sum_{h=1}^{r}\sum_{i\in CI_{j_h}} E\left(T_{r'hi}\right) \tag{11}$$

holds. To evaluate $T_{r'hi}$, note that, even though the indices of variables $j_1,\ldots,j_r$ to be flipped are fixed, the 0-1 assignments of the variables $j_1,\ldots,j_r$ are random by the assumption. Therefore,

$$Pr\left(|P_i(v\updownarrow\{j_1,\ldots,j_{h-1}\})\backslash\{j_h\}| = k\right) = \frac{\binom{|C_i|-1}{k}}{2^{|C_i|-1}}$$

holds for all $k = 0, 1, \ldots, r$, and the number of memory accesses to $\hat{g}(S)$ with $|S| = r'$ when $|P_i(v \updownarrow \{j_1, \ldots, j_{h-1}\}) \backslash \{j_h\}| = k$ is at most

$$\binom{|C_i| - k}{r' - k} + \binom{|C_i| - k - 1}{r' - k - 1},$$

by (10). Therefore, we have

$$\begin{aligned} E(T_{r'hi}) &\leq \sum_{k=0}^{r'} \frac{\binom{|C_i|-1}{k}}{2^{|C_i|-1}} \cdot \left\{ \binom{|C_i| - k}{r' - k} + \binom{|C_i| - k - 1}{r' - k - 1} \right\} \\ &\leq \frac{1}{2^{|C_i|-2}} \sum_{k=0}^{r'} \binom{|C_i|}{k} \binom{|C_i| - k}{r' - k} \\ &= \frac{1}{2^{|C_i|-2}} \sum_{k=0}^{r'} \binom{|C_i|}{r'} \binom{r'}{k} \\ &= \frac{\binom{|C_i|}{r'}}{2^{|C_i|-r'-2}}. \end{aligned} \tag{12}$$

Therefore, by (11) and $\sum_{h=1}^{r} |CI_{j_h}| \leq rt$,

$$E(\tau_r(v)) \leq rt \sum_{r'=1}^{r} \max_{1 \leq l' \leq l} \frac{\binom{l'}{r'}}{2^{l'-r'-2}}$$

holds. For a constant r', $\binom{l'}{r'}/2^{l'-r'-2}$ is a function of l' with $\lim_{l' \to \infty} \binom{l'}{r'}/2^{l'-r'-2} = 0$. Hence, $\max_{l' \geq 1} \binom{l'}{r'}/2^{l'-r'-2}$ is a finite constant dependent only on r'. Thus we have

$$E(\tau_r(v)) \leq O(t)$$

for any constant r. □

For example, we have

$$\begin{aligned} E(\tau_1(v)) &\leq 4t \\ E(\tau_2(v)) &\leq 20t \\ E(\tau_3(v)) &\leq 60t, \end{aligned}$$

since $\max_{l' \geq 1} \binom{l'}{1}/2^{l'-3} = 4$, $\max_{l' \geq 1} \binom{l'}{2}/2^{l'-4} = 6$, and $\max_{l' \geq 1} \binom{l'}{3}/2^{l'-5} = 10$.

For the 1-flip neighborhood, if we store the set $\{j \in N \mid g(v, \{j\}) > 0\}$ (all of which give better solutions) in a linked list, we can always find a better solution v' (i.e., $f(v') > f(v)$) in $O(1)$ time since such a solution can be given by the first element in the list (recall that $g(v, \{j\}) = \Delta f(v, \{j\})$ holds for all $j \in N$); we can also conclude that

no better solution exists in $NB_1(v)$ by checking if the list is empty or not. The update of this linked list is possible in $O(1)$ time for each change of $g(v, \{j\})$ value. Thus, if we use the move strategy FA, one-round time for 1-flip neighborhood is $O(tl)$ in the worst case and its expectation is $O(t)$ by Lemma 3.2.

3.3 Analysis of the Memory Size

In this subsection, we estimate the memory size needed to store the values of $g(v, S)$ for S with $|S| \leq r$. As mentioned in Subsection 3.1, we store only those $g(v, S)$ with nonzero values. Let

$$\mu_r(v) = |\{S \subseteq N |\ g(v, S) \neq 0 \text{ and } |S| \leq r\}|\,,$$

i.e., the memory size needed in our algorithm.

Lemma 3.3 *$\mu_r(v) = O(n + \min\{l^r m, n^r\})$ holds for any v. If we assume that the current solution v is randomly chosen from $\{0,1\}^n$ with probability $1/2^n$, then $E(\mu_r(v)) = O(n + m)$ holds for any constant r.*

Proof: First, we consider the worst case size of μ_r. By definition (6), $g(v, S) \neq 0$ holds only if $\exists i \in M$ such that $P_i(v) \subseteq S \subseteq VI_i$. The number of subsets S with $P_i(v) \subseteq S \subseteq VI_i$ is maximized when $|P_i(v)| = 0$ and $|VI_i| = l$. In this case, the number of S is $\sum_{k=1}^{r} \binom{l}{k}$. By summing up this for all $i \in M$, and using $|\{S \subseteq N |\ |S| = 1\}| = n$, we have

$$\mu_r(v) \leq n + m \sum_{k=2}^{r} \binom{l}{k}.$$

By combining this with the trivial bound $|\{S \subseteq N |\ |S| = k\}| = \binom{n}{k}$, we have

$$\begin{aligned} \mu_r(v) &= n + \sum_{k=2}^{r} \min\{\binom{l}{k} m, \binom{n}{k}\} \\ &= O(n + \min\{l^r m, n^r\}). \end{aligned} \tag{13}$$

Next, we consider the expected size of $\mu_r(v)$ when the solution v is randomly chosen from $\{0,1\}^n$ with probability $1/2^n$. By the linearity of expectation, we have

$$\begin{aligned} E\left(\mu_r(v)\right) &\leq E\left(n + \sum_{r'=2}^{r} \sum_{i \in M} \left|\{S \subseteq N |\ g_i(v, S) \neq 0 \text{ and } |S| = r'\}\right|\right) \\ &= n + \sum_{r'=2}^{r} \sum_{i \in M} E\left(\left|\{S \subseteq N |\ g_i(v, S) \neq 0 \text{ and } |S| = r'\}\right|\right). \end{aligned} \tag{14}$$

Let us restrict our attention to clause i and S with $|S| = r'$. For $k = 0, 1, \ldots, r'$,

$$Pr(|P_i(v)| = k) = \frac{\binom{|C_i|}{k}}{2^{|C_i|}},$$

and if $|P_i(v)| = k$,

$$\left|\{S \subseteq N |\ S \supset P_i(v) \text{ and } |S| = r'\}\right| = \binom{|C_i| - k}{r' - k}.$$

Hence, we have

$$\begin{aligned} E\left(\left|\{S \subseteq N |\ g_i(v,S) \neq 0 \text{ and } |S| = r'\}\right|\right) &= \frac{1}{2^{|C_i|}} \sum_{k=0}^{r'} \binom{|C_i|}{k}\binom{|C_i| - k}{r' - k} \\ &= \frac{\binom{|C_i|}{r'}}{2^{|C_i| - r'}} \quad \text{(see (12) for the derivation)}, \end{aligned}$$

and, by (14),

$$E\left(\mu_r(v)\right) \leq n + m \sum_{r'=2}^{r} \max_{1 \leq l' \leq l} \frac{\binom{l'}{r'}}{2^{l' - r'}}.$$

By an argument similar to the proof of Lemma 3.2, we then have

$$E\left(\mu_r(v)\right) = O(n + m)$$

for any constant r. □

For example, we have

$$\begin{aligned} E\left(\mu_2(v)\right) &\leq n + \frac{3}{2}m \\ E\left(\mu_3(v)\right) &\leq n + 4m, \end{aligned} \tag{15}$$

since $\max_{l' \geq 1} \binom{l'}{2}/2^{l'-2} = 3/2$ and $\max_{l' \geq 1} \binom{l'}{3}/2^{l'-3} = 5/2$.

3.4 Restriction of the 2-Flip Neighborhood

In this subsection, we derive a condition that reduces the number of candidates in 2-flip neighborhood without sacrificing the solution quality. The following Lemma is immediate from (8).

Lemma 3.4 *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 the above lemma, we have only to check $NB_2'(v)$ to find a better solution in $NB_2(v)$ or to conclude local optimality. To do this efficiently, we store all the elements in $\{S \subseteq N \mid g(v,S) > 0 \text{ and } |S| \leq 2\}$ in a linked list; thereby all candidates in $NB_2'(v)$ can be scanned in $O(|NB_2'|)$ time. The update of this list is possible in $O(1)$ time for each change of the $g(v,S)$ value, by Assumption 3.1, if we keep the link information from the memory space containing $g(v,S)$. Therefore, applying this to the argument in Section 1, we see that the worst case one-round time of LS using the restricted 2-flip neighborhood is $O(|NB_2'| + tl^2)$ and its expected one-round time is $O(|NB_2'| + t)$.

Now we consider the size $|NB_2'|$ of this restricted 2-flip neighborhood. First, we consider the worst case. Obviously $|NB_2'| = O(n + ml^2)$ holds, since $|NB_2'| \leq \mu_2 = O(n + ml^2)$, which is then improved as follows.

Lemma 3.5 *$|NB_2'(v)| = O(n + \min\{ml, n^2\})$ holds for any v.*

Proof: First, $|NB_2'(v)| = O(n^2)$ clearly holds. By definition (6), for $|S| = 2$, $g(v,S) > 0$ is possible only if $|P_i(v)| = 1$. Therefore, we have

$$\begin{aligned} |NB_2'(v)| &\leq n + \sum_{i \in M} |\{S \subseteq N \mid g_i(v,S) > 0 \text{ and } |S| = 2\}| \\ &= n + \sum_{i \in M,\ |P_i(v)|=1} |VI_i \backslash P_i(v)| \ \text{(by the above argument)} \\ &\leq n + m(l-1) \\ &= O(n + ml). \ \square \end{aligned} \tag{16}$$

Next we consider the average case when the solution v is randomly chosen from $\{0,1\}^n$ with probability $1/2^n$. By (15), we can conclude $E(|NB_2'(v)|) \leq E(\mu_2(v)) \leq n + \frac{3}{2}m$. However, if we take into account the condition "for $|S| = 2$, $g_i(v,S) > 0$ only if $|P_i(v)| = 1$," and do calculation similar to the proof of Lemma 3.3, we can improve the constant factor as follows.

Lemma 3.6 *If we assume that the current solution v is randomly chosen from $\{0,1\}^n$ with probability $1/2^n$, then $E(|NB_2'(v)|) \leq n + \frac{3}{4}m$.*

Proof: Omitted. $\square$

In summary, the following theorem holds for the restricted 2-flip neighborhood.

Theorem 3.1 *Under Assumption 3.1, the LS with the restricted 2-flip neighborhood NB_2' requires $O(n + ml + tl^2)$ one-round time and $O(n + ml^2)$ memory space in the worst case. If we assume that the current solution v is randomly chosen from $\{0,1\}^n$ with probability $1/2^n$, then the expected one-round time and the expected memory size both become $O(n + m)$.*

3.5 Restriction of the 3-Flip Neighborhood

In this subsection, we derive a condition that reduces the number of candidates in 3-flip neighborhood without sacrificing the solution quality.

Lemma 3.7 *If $\Delta f(v, \{j_a\}) \leq 0$ for all $a \in \{1,2,3\}$ and $\Delta f(v, \{j_a, j_b\}) \leq 0$ for all $a, b \in \{1,2,3\}$ with $a \neq b$ hold, then $\Delta f(v, \{j_1, j_2, j_3\}) > 0$ is possible only if at least one of the following two conditions holds: (1) $g(v, \{j_1, j_2, j_3\}) > 0$, (2) $g(v, \{j_a, j_b\}) > 0$ and $g(v, \{j_b, j_c\}) > 0$ for some $a, b, c \in \{1,2,3\}$ (a, b and c are all distinct).*

Proof: By the stated assumption, we have the following inequalities:

$$\begin{aligned} g(v, \{j_a\}) &\leq 0, \qquad \forall a \in \{1,2,3\} \\ g(v, \{j_a\}) + g(v, \{j_b\}) + g(v, \{j_a, j_b\}) &\leq 0, \qquad \forall a \neq b \in \{1,2,3\} \\ \sum_{S \subseteq \{j_1, j_2, j_3\}, S \neq \emptyset} g(v, S) &> 0. \end{aligned}$$

Suppose $g(v, \{j_1, j_2, j_3\}) \leq 0$, $g(v, \{j_1, j_2\}) \leq 0$ and $g(v, \{j_2, j_3\}) \leq 0$ hold. Then by

$$g(v, \{j_3\}) + g(v, \{j_1\}) + g(v, \{j_3, j_1\}) \leq 0$$

and

$$g(v, \{j_2\}) \leq 0,$$

we have

$$\sum_{S \subseteq \{j_1, j_2, j_3\}, S \neq \emptyset} g(v, S) \leq 0,$$

which is contradiction. Other cases are similar. □

Let

$$\begin{aligned} P2(v) \quad = \quad \{\{j_1, j_2, j_3\} \subseteq N \mid\ & g(v, \{j_a, j_b\}) > 0 \text{ and } g(v, \{j_b, j_c\}) > 0 \\ & \text{for some } \{a, b, c\} = \{1,2,3\}\} \end{aligned}$$

and

$$NB'_3(v) \quad = \quad \{v \updownarrow S |\ g(v,S) > 0 \text{ and } |S| \leq 3\} \cup \{v \updownarrow S |\ S \in P2(v)\}.$$

Then, by Lemma 3.7, we have only to check $NB'_3(v)$ to find a better solution in $NB_3(v)$ or to conclude local optimality.

To do this efficiently, we store all the sets

$$S \in \{S' \subseteq N |\ g(v,S') > 0 \text{ and } |S'| \leq 3\} \cup P2(v)$$

in a linked list, which we call NB-list. We can search all the candidates in $NB'_3(v)$ in $O(|NB'_3|)$ time by scanning this NB-list. To update NB-list efficiently, we store the index set $\{h \in N |\ g(v,\{j,h\}) > 0\}$ in another linked list for each $j \in N$, which we call P2-list. The update of P2-list can be done in $O(1)$ time for each change of $g(v,S)$ with $|S| = 2$. We now consider the update of NB-list. In algorithm UPDATE in Subsection 3.1, whenever $\hat{g}(\{j_1,j_2\})$ becomes positive from negative or zero (resp., becomes negative or zero from positive), all the candidates in

$$\{\{j_1,j_2,h\} |\ \hat{g}(\{j_1,h\}) > 0 \text{ or } \hat{g}(\{j_2,h\}) > 0\}$$

are added to (resp., deleted from) NB-list. Let

$$q_j(v) = |\{h \in N |\ g(v,\{j,h\}) > 0\}|.$$

Then the update of NB-list is done in $O(q_{j_1}(v) + q_{j_2}(v))$ time for each change of $\hat{g}(S)$ with $|S| = 2$.

First, we evaluate the worst and average values of $q_j(v)$ in the following lemma.

Lemma 3.8 *In the worst case, $q_j(v) \leq tl$ holds.*

Proof: By definition (6), $g_i(v,\{j,h\}) > 0$ holds only if $P_i(v) = \{j\}$ or $P_i(v) = \{h\}$. Therefore, $|\{h \in N |\ g_i(v,\{j,h\}) > 0\}| \leq l$ holds for any i. Thus we have

$$q_j(v) \leq \sum_{i \in M} |\{h \in N |\ g_i(v,\{j,h\}) > 0\}| \leq |CI_j| l \leq tl$$

in the worst case. □

Thus the computational time to update NB-list in one round of UPDATE, i.e., $\tau_3(v) + \tau_2(v) \cdot O(q_{j_1}(v) + q_{j_2}(v))$, is $O(t^2 l^3)$ in the worst case and $O(t^2 l)$ on the average by Lemma 3.2.

Next, we consider the size of the restricted 3-flip neighborhood $|NB'_3|$.

Lemma 3.9 *In the worst case,* $|NB'_3(v)| = O(ml^3 + t^2l^2n)$ *holds for any* $v \in \{0,1\}^n$. *If we assume that the solution* v *is randomly chosen from* $\{0,1\}^n$ *with probability* $1/2^n$, *then* $E(|NB'_3(v)|) = O(m + t^2n)$ *holds.*

Proof: The average and worst case sizes of the set

$$\{S \subseteq N |\ g(v,S) > 0 \text{ and } |S| \leq 3\}$$

do not exceed the number of memory cells needed to store the value of $g(v,S)$ ($|S| \leq 3$) with nonzero value, which were already analyzed in Subsection 3.3. Here we consider the size of $P2(v)$, defined by (17). Using $q_j(v) \leq tl$ of Lemma 3.8, we see that

$$|P2(v)| \leq \sum_{j \in N} \binom{q_j(v)}{2} \leq \sum_{j \in N} \binom{tl}{2} \leq t^2l^2n$$

holds in the worst case.

To consider the average case, let

$$X_{ji} = |\{h \in N |\ g_i(v, \{j,h\}) > 0\}|\,,\ j \in N,\ i \in M,$$

which is the random variable representing the contribution of clause C_i to $q_j(v)$, and let

$$X_j = \sum_{i \in M} X_{ji} = \sum_{i \in CI_j} X_{ji}.$$

Then $q_j(v) \leq X_j$ holds. By the linearity of expectation,

$$\begin{aligned} E(|P2(v)|) &\leq \sum_{j \in N} E\left(\binom{q_j(v)}{2}\right) \\ &\leq \frac{1}{2} \sum_{j \in N} E\left(X_j^2\right) \\ &= \frac{1}{2} \sum_{j \in N} \left\{ \sum_{i \in CI_j} E\left(X_{ji}^2\right) + \sum_{i_1 \neq i_2 \in CI_j} E\left(X_{ji_1} X_{ji_2}\right) \right\}. \end{aligned} \tag{17}$$

By definition (6), $g_i(v,\{j,h\}) > 0$ holds only if $P_i(v) = \{j\}$ or $P_i(v) = \{h\}$. As j is fixed, $Pr(P_i(v) = \{j\}) = \frac{1}{2^{|C_i|}}$ and $Pr(P_i(v) = \{h\}, h \neq j) = \frac{|C_i|-1}{2^{|C_i|}}$; hence we have

$$\begin{aligned} E\left(X_{ji}^2\right) &= \frac{1}{2^{|C_i|}} \cdot (|C_i| - 1)^2 + \frac{|C_i| - 1}{2^{|C_i|}} \cdot 1^2 \\ &= \frac{|C_i|(|C_i| - 1)}{2^{|C_i|}} \\ &\leq \frac{3}{4}. \end{aligned}$$

To evaluate $E(X_{ji_1}X_{ji_2})$, we must consider the following two cases. Without loss of generality, we assume that $|C_{i_1}| \leq |C_{i_2}|$.

Case 1 (x_j (resp., $\overline{x}_j$) $\in C_{i_1}$ and x_j (resp., $\overline{x}_j$) $\in C_{i_2}$): By definition (6), $X_{ji_1}X_{ji_2} > 0$ holds only if $P_{i_1}(v) = P_{i_2}(v) = \{j\}$, or $P_{i_1}(v) = \{h_1\}$ and $P_{i_2}(v) = \{h_2\}$ for some $h_1, h_2 \neq j$. In the former case,

$$\begin{aligned} Pr(P_{i_1}(v) = \{j\} \text{ and } P_{i_2}(v) = \{j\}) &\leq Pr(P_{i_1}(v) = \{j\}) \\ &= \frac{1}{2^{|C_{i_1}|}} \\ X_{ji_1} &= |C_{i_1}| - 1 \\ X_{ji_2} &= |C_{i_2}| - 1 \end{aligned}$$

(note that this bound is tight if $C_{i_2} \subseteq C_{i_1}$), and in the latter case,

$$\begin{aligned} &Pr(P_{i_1}(v) = \{h_1\} \text{ and } P_{i_2}(v) = \{h_2\} \text{ for some } h_1, h_2 \neq j) \\ &\qquad \leq Pr(P_{i_1}(v) = \{h_1\} \text{ for some } h_1 \neq j) \\ &\qquad = \frac{|C_{i_1}| - 1}{2^{|C_{i_1}|}} \\ X_{ji_1} &= X_{ji_2} = 1. \end{aligned}$$

Therefore, we have

$$\begin{aligned} E(X_{ji_1}X_{ji_2}) &\leq \frac{1}{2^{|C_{i_1}|}} \cdot (|C_{i_1}| - 1) \cdot (|C_{i_2}| - 1) + \frac{|C_{i_1}| - 1}{2^{|C_{i_1}|}} \cdot 1 \cdot 1 \\ &\leq \frac{|C_{i_1}|(|C_{i_1}| - 1)}{2^{|C_{i_1}|}} \\ &\leq \frac{3}{4}. \end{aligned}$$

Case 2 (x_j (resp., $\overline{x}_j$) $\in C_{i_1}$ and $\overline{x}_j$ (resp., x_j) $\in C_{i_2}$): By definition (6), $X_{ji_1}X_{ji_2} > 0$ holds only if $P_{i_a}(v) = \{j\}$ and $P_{i_b}(v) = \{h\}$ for some $h \neq j$ ($a \neq b \in \{1,2\}$). Then, in this case,

$$\begin{aligned} Pr(P_{i_a}(v) = \{j\} \text{ and } P_{i_b}(v) = \{h\} \text{ for some } h \neq j) &\leq Pr(P_{i_a}(v) = \{j\}) \\ &= \frac{1}{2^{|C_{i_a}|}} \\ X_{ji_a} &= |C_{i_a}| - 1 \\ X_{ji_b} &= 1 \end{aligned}$$

hold for $a \neq b \in \{1,2\}$, and we have

$$E(X_{ji_1}X_{ji_2}) \leq \sum_{a \in \{1,2\}} \frac{|C_{i_a}| - 1}{2^{|C_{i_a}|}} \leq \frac{1}{2}.$$

Therefore, $E(X_{ji_1}X_{ji_2}) \leq \frac{3}{4}$ holds in all cases, and hence, we have

$$\begin{aligned} E(X_j^2) &\leq \frac{3}{4}|CI_j| + \frac{3}{4}|CI_j|(|CI_j| - 1) \quad \text{(by (17))} \\ &= \frac{3}{4}|CI_j|^2 \end{aligned}$$

and

$$\begin{aligned} E(|P2(v)|) &\leq \frac{1}{2}\sum_{j \in N} E(X_j^2) \\ &\leq \frac{3}{8}\sum_{j \in N} |CI_j|^2 \\ &\leq \frac{3}{8}t^2 n. \ \square \end{aligned}$$

In conclusion, we have the following theorem.

Theorem 3.2 *Under Assumption 3.1, the LS with the restricted 3-flip neighborhood NB_3' requires $O(ml^3 + t^2l^2n)$ one-round time and memory space in the worst case. If we assume that the current solution v is randomly chosen from $\{0,1\}^n$ with probability $1/2^n$, then the expected one-round time and the expected memory space both become $O(m + t^2n)$.*

3.6 Discussions about Assumption 3.1

To the authors' knowledge, there is no data structure to realize Assumption 3.1 in the worst case. However, this assumption is realistic, in practice, if we use the ordinary hash technique [9], which is actually used to implement the proposed algorithms. In our experiment for random instances with up to $n \leq 1000$, the maximum overhead steps to access the memory space is at most 1 or 2 if we use appropriately large bucket size for the hash.

Let WT, AT, WM and AM be the worst case one-round time, average case one-round time, worst case memory space and average case memory space, respectively, stated in one of the theorems shown in Section 3 (e.g., for $r = 2$, $WT = O(n + ml + tl^2)$ $WM = O(n + ml^2)$ and $AT = AM = O(n + m)$ by Theorem 3.1). Then, even without Assumption 3.1, we can realize the same worst and average case time complexities as WT and AT, if we store $\hat{g}(S)$ in r dimensional array, although $O(n^r)$ memory space is required. We can also realize the same worst and average case memory requirements as WM and AM if we store $\hat{g}(S)$ in hash. In this case, we cannot bound the worst

case access time to each memory cell by $O(1)$, and hence, we cannot bound the worst case time complexity by the WT. Finally, we also note that by using the perfect hash technique proposed in [2], the worst and average case time complexities of WT and AT, and worst case memory requirement WM can be realized, after spending WM expected time for preprocessing (this preprocessing is needed only once in the whole run).

4 Experimental Results

In order to confirm the average case results in Lemma 3.2, and Theorems 3.1 and 3.2, we conducted numerical experiments of LS for some MAX SAT instances. The algorithms were coded in C language and run on a workstation Sun Ultra 2 Model 2300 (300 MHz, 1 GB memory). The clauses were generated according to [8] and integer weights w_i were randomly chosen from [1, 1000], as was done in [12]. The following data are the average of independent 10 runs of LS from randomly generated initial solutions.

Tables 1, 2 and 3 show the average number of memory accesses for the updates of $g(v, S)$ in one move of r-flip neighborhoods, $r = 1, 2, 3$, respectively. The results in Table 3 include also the memory accesses to update NB-list. The ratios between the observed values and the theoretical expectation t for $r = 1$ and 2, and tl^2 for $r = 3$, are also shown. It is observed that this ratio is almost constant, as asserted in Lemmas 3.2 and 3.8.

Tables 4 and 5 show the average sizes of the restricted 2 and 3-flip neighborhoods, $NB'_2(v)$ and $NB'_3(v)$. The ratios between the observed values and the theoretical expectation (i.e., $E(|NB'_2(v)|) = O(n + m)$ and $E(|NB'_3(v)|) = O(m + t^2 n)$), and the ratios between the observed values and the original 2 and 3-flip neighborhood sizes (i.e., $|NB_2(v)| \simeq \binom{n}{2}$ and $|NB_3(v)| \simeq \binom{n}{3}$) are also shown. It is observed that the ratio between the observed values and the theoretical expectation is almost constant (i.e., justifying the results in Theorems 3.1 and 3.2). It is also observed that the restriction is quite effective in reducing the neighborhood sizes, especially for larger n.

5 Conclusion

In this paper, we proposed efficient implementations of 2 and 3-flip neighborhoods for the MAX SAT problem. It is shown that the expected one-round time and the memory space for 2-flip (resp., 3-flip) neighborhood is $O(n + m)$ (resp., $O(m + t^2 n)$). It is also shown that the expected one-round time for 1-flip neighborhood with first admissible

Table 1: Average number of memory accesses for the update with the 1-flip neighborhood.

n	m	l	t	$\tau_1(v)$	$\frac{\tau_1(v)}{t}$
100	850	14	63	30.08	0.48
100	1150	12	76	40.43	0.53
100	2300	15	138	79.40	0.58
1000	7700	14	60	28.25	0.47
1000	11050	15	84	37.53	0.45
1000	22100	15	145	73.78	0.51

Table 2: Average number of memory accesses for the update with the restricted 2-flip neighborhood.

n	m	l	t	$\tau_2(v)$	$\frac{\tau_2(v)}{t}$
100	850	14	63	181.45	2.88
100	1150	12	76	245.43	3.23
100	2300	15	138	532.47	3.86
1000	7700	14	60	168.98	2.82
1000	11050	15	84	237.27	2.82
1000	22100	15	145	504.26	3.48

Table 3: Average number of memory accesses for the update with the restricted 3-flip neighborhood. Note that the memory accesses to update NB-list is also included, i.e., $\#(\text{access}) = \tau_3(v) + \tau_2(v) \cdot O(q_{j_1}(v) + q_{j_2}(v))$ as argued in Subsection 3.5.

n	m	l	t	$\#(\text{access})$	$\frac{\#(\text{access})}{t^2 l}$
100	850	14	63	2341.61	0.042
100	1150	12	76	3744.95	0.054
100	2300	15	138	11348.60	0.040
1000	7700	14	60	2317.89	0.046
1000	11050	15	84	4181.37	0.040
1000	22100	15	145	14726.80	0.047

Table 4: Average sizes of the restricted 2-flip neighborhoods for random instances.

n	m	l	t	avr. $\lvert NB'_2\rvert$	$\frac{\lvert NB'_2\rvert}{n+m}$	$\frac{\lvert NB'_2\rvert}{\binom{n}{2}}$
100	850	14	63	470.86	0.496	0.095
100	1150	12	76	617.78	0.494	0.125
100	2300	15	138	1088.08	0.453	0.220
1000	7700	14	60	4637.18	0.533	0.009
1000	11050	15	84	6493.01	0.539	0.013
1000	22100	15	145	12453.54	0.539	0.025

Table 5: Average sizes of the restricted 3-flip neighborhoods for random instances.

n	m	l	t	avr. $\lvert NB'_3\rvert$	$\frac{\lvert NB'_3\rvert}{m+t^2n}$	$\frac{\lvert NB'_3\rvert}{\binom{n}{3}}$
100	850	14	63	5488.72	0.0138	0.0339
100	1150	12	76	8714.01	0.0151	0.0539
100	2300	15	138	22784.83	0.0119	0.1409
1000	7700	14	60	56335.04	0.0156	0.0003
1000	11050	15	84	102766.59	0.0145	0.0006
1000	22100	15	145	343105.15	0.0163	0.0021

(FA) move strategy is $O(t)$. The computational results with up to $n = 1000$ random instances indicate that these theoretical analyses represent the real performance of LS well.

Acknowledgement

The authors are grateful to Kazuo Iwama, Kyoto University, who encouraged us to analyze the one-round time. The authors are also grateful to Wolfgang Slany at Technische Universität Wien, Takeaki Uno at Tokyo Institute of Technology and Koji Nonobe at Kyoto University for their valuable discussions. 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," *Proc. AAAI* (1997) 263–268.

[2] M.L. Fredman, J. Komlós and E. Szemerédi, "Storing a Sparse Table with $O(1)$ Worst Case Access Time," *J. ACM*, 31 (1984) 538–544.

[3] A.S. Fukunaga, "Variable-Selection Heuristics in Local Search for SAT," *Proc. AAAI* (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] I.P. Gent and T. Walsh, "Towards an Understanding of Hill-Climbing Procedures for SAT," *Proc. AAAI* (1993) 28–33.

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

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

[8] 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.

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

[10] B. Mazure, L. Saïs and É. Grégoire, "Tabu Search for SAT," *Proc. AAAI* (1997) 281–285.

[11] P.M. Pardalos, L.S. Pitsoulis and M.G.C. Resende, "A Parallel GRASP for MAX-SAT Problems," *LNCS*, 1180 (1996) 575–585.

[12] M.G.C. Resende, L.S. Pitsoulis and P.M. Pardalos, "Approximate Solution of Weighted MAX-SAT Problems Using GRASP," *Dimacs Series on Discrete Mathematics and Theoretical Computer Science* (1997) to appear.

[13] B. Selman, H. Levesque and D. Mitchell, "A New Method for Solving Hard Satisfiability Problems," *Proc. AAAI* (1992) 440–446.

[14] B. Selman and H.A. Kautz, "An Empirical Study of Greedy Local Search for Satisfiability Testing," *Proc. AAAI* (1993) 46–51.

[15] B. Selman and H.A. Kautz, "Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems," *Proc. IJCAI* (1993) 290–295.

[16] B. Selman, H.A. Kautz and B. Cohen, "Noise Strategies for Improving Local Search," *Proc. AAAI* (1994).

[17] B. Selman, H.A. Kautz and B. Cohen, "Local Search Strategies for Satisfiability Testing," *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, 26 (1996) 521–531.

[18] T. Uno, private communication.

[19] M. Yagiura and T. Ibaraki, "Efficient 2 and 3-Flip Neighborhood Search Algorithms for the MAX SAT," *LNCS* (*Proc. COCOON*), 1449 (1998) 105–116.

[20] M. Yagiura and T. Ibaraki, "Efficient 2 and 3-Flip Neighborhood Search Algorithms for the MAX SAT: Experimental Evaluation," submitted for publication.