合取范式 3 可满足问题的局部搜索近似算法

朱大铭 马绍汉 张平平
(山东大学计算机科学技术学院 济南 250101)

摘 要 合取范式最大可满足问题是理论计算机科学的核心问题.局部搜索被许多求解实践证明是解答合取范式最大可满足问题十分有效的方法,但未见关于局部搜索算法解答该问题性能分析的结果.文中讨论最大 3 可满足问题(Max-(3)-Sat)的局部搜索算法并分析算法性能.证明 Max-(3)-Sat 问题的一位跳变局部搜索算法的近似性能比为 4/3;证明一位跳变局部搜索后跟有条件全体跳变算法,解答 Max-(3)-Sat 问题的近似性能比为 5/4.设计一位跳变加全体跳变的新局部搜索算法,证明新算法解答 Max-(3)-Sat 问题的近似性能比为 8/7.将 8/7-近似局部搜索算法推广为解答 Max-(k)-Sat 问题的局部搜索算法,证明算法的近似性能比为$(2k+2)/(2k+1)$,$k\geq 4$.设计解答 Max-(k)-Sat 问题的两位跳变局部搜索算法,证明两位跳变局部搜索算法的近似性能比为 $1+1/(2k+1+k(k-1)/(n-k))$,$k\geq 4$.局部搜索算法经多次运行可进一步提高求解性能.文中结果显示,局部搜索算法在合取范式最大可满足问题求解实践中表现出高性能,有其必然性.

关键词 算法;复杂性;近似性能比;可满足性;局部搜索
中图法分类号 TP301 **DOI 号**: 10.3724/SP.J.1016.2010.01127

The Local Search Algorithms for Maximum 3-Satisfiablility Problem

ZHU Da-Ming MA Shao-Han ZHANG Ping-Ping
(*School of Computer Science and Technology, Shandong University, Jinan* 250101)

Abstract Maximum satisfiability problem is the central problem in theoretical computer science. Maximum 3-satisfiability problem(Max-(3)-Sat) is one of the most important sub problem of the maximum satisfiablity problem. Local search has been testified to be very effective for solving this problem by many times of real computation. However, there is no related results for analyzing the performance of the local search method to solve the problem. This paper approaches the local search method to solve the maximum satisfiablity problems and analyzes the performance of the method. The central issue of the paper is to discuss the local search algorithm to solve the Max-(3)-Sat. The authors show that the one-bit-flip local search algorithm can achieve the performance ratio 4/3 for solving the Max-(3)-Sat; and the algorithm of one-bit-flip local search following the conditionally all-bits-flip can achieve the performance ratio 5/4 for solving Max-(3)-Sat. And the authors design a new local search algorithm to solve the Max-(3)-Sat problem using the one-bit-flip local search following the all-bits-flip, and show the new algorithm have the performance ratio 8/7. The 8/7-approximation algorithm can be extended to solve the Max-(k)-Sat problem with performance ratio $(2k+2)/(2k+1)$ for $k\geq 4$. Finally the authors give the two-bits-flip local search algorithm to solve the Max-(k)-Sat problem, and prove the algorithm can have the performance ratio $1+\frac{1}{2k+1+k(k-1)/(n-k)}$ for $k\geq 4$. Repeatedly running the local search

收稿日期:2008-10-23;最终修改稿收到日期:2009-06-16.本课题得到国家自然科学基金(60573024)和教育部博士点基金(200901311100009)资助.**朱大铭**,男,1963 年生,博士,教授,博士生导师,主要从事算法设计与分析、计算分子生物学研究.E-mail:dmzhu@sdu.edu.cn.**马绍汉**,男,1938 年生,教授,博士生导师,主要从事算法设计与分析、人工智能研究.**张平平**,男,1981 年生,硕士,工程师,主要从事算法设计与分析、软件工程研究.

algorithm can further improve the performance of the solution. The results of the paper demonstrate the inevitability for local search to have high performances in solving the maximum satisifiablity problems.

Keywords algorithm; complexity; performance ratio; satisfiability; local search

1 引 言

合取范式最大可满足问题是理论计算机科学的核心问题.由布尔变量集合与项(clause)集合给定的布尔表达式即是合取范式.合取范式最大可满足问题要求计算布尔变量赋值,使可满足的项数目最大.该问题简称为 Max-Sat.因 Max-Sat 是 NP-Hard[1],且是 Max-SNP-Hard[2],因此其近似算法和实用算法设计研究成为该问题研究的重心.Johnson最先设计了解答 Max-Sat 问题的近似算法,并证明算法的近似性能比为 2[3].Chen、Friesen、Zheng 证明 Johnson 算法的近似性能比实际为 3/2[4].Yannakakis 于 1994 年给出解答 Max-Sat 问题的随机算法,平均近似性能比为 4/3.Goemans 与 Williamson 利用线性规划松弛法,给出一个更为简单的随机近似算法[5],其近似性能比仍为 4/3.

限制项含有字母数的 Max-Sat 子问题,在 Max-Sat 问题研究中占有重要位置,也是计算机科学工作者最早尝试讨论算法与复杂性所研究的问题之一.本文将每个项含有不多于 k 个字母的 Max-Sat 子问题称为 Max-k-Sat 问题;将每个项含有不少于 k 个字母的 Max-Sat 子问题称为 Max-(k)-Sat 问题.早在 1974 年,Johnson 就设计了 Max-(k)-Sat 问题的组合近似算法,近似性能比为 $2^k/(2^k-1)$[3].实际上,为每个布尔变量以 $\frac{1}{2}$ 概率随机赋值 T 或 F 的方法,解答 Max-(k)-Sat 的平均近似性能比即为 $2^k/(2^k-1)$[6].Goemans 与 Williamson 于 1995 年给出了解答 Max-2-Sat 问题的 1.139-近似随机算法[7],他们所采用的半定规划松弛法突破了原有近似算法的设计,进一步提高了算法性能.Feige 与 Goemans 利用同样方法进一步将算法的近似性能比改进为 1.074[8].利用半定规划松弛法也可适当改进解答 Max-Sat 问题的算法性能.Goemans 与 Williamson 将解答 Max-Sat 问题算法的近似性能比改进为 1.32[7],Asano 与 Williamson 又将算法近似性能比改进为 1.254[9].

Trevisan、Sorkin、Sudan 与 Williamson 将 Max-3-Sat 归约为 Max-2-Sat,得到 Max-3-Sat 问题的 1.248-近似算法[10],Trevisan 又将解答 Max-3-Sat 问题的近似性能比改进为 1.211,并将解答 Max-Sat 问题的近似性能比改进为 1.25[11].

另一方面,Hastad 证明:若 P≠NP,则每个项均含有恰好 3 个字母的 Max-Sat 子问题不能多项式时间近似到小于 8/7[12].因此若 P≠NP,则 Max-3-Sat 问题和 Max-(3)-Sat 问题均不能多项式时间近似到小于 8/7.而为布尔变量随机赋值的简单算法解答每个项均含有 3 个字母的 Max-Sat 子问题,恰好使平均近似性能比达到 8/7.Karloff 与 Zwick 给出解答 Max-3-Sat 问题的半定规划松弛法,解答可满足 Max-3-Sat 实例的近似性能比能够达到 8/7[13],后来他们证明该方法解答任意 Max-3-Sat 实例的近似性能比也为 8/7[14].Halperin 与 Zwick 又给出了解答 Max-4-Sat 问题的半定规划松弛算法,近似性能比为 1.147[14].

Johnson 算法是我们所知解答 Max-(3)-Sat 问题性能最好的确定近似算法;Karloff 与 Zwick 给出的半定规划松弛法是我们所知解答 Max-3-Sat 问题性能最好的随机近似算法.实际上前述近似算法中,仅有 Johnson 算法为确定算法.其他算法均为随机近似算法,因此算法近似性能比均为平均近似性能比.因为随机算法去随机过程的时间复杂性太大,这些算法的实用性仍显不足.

以寻找问题精确解为目标的实用算法设计研究在 Max-Sat 求解实践中同样显示出活力,这样的算法可分为回溯搜索法和局部最优搜索法两大类.回溯算法要得到精确解,总不能摆脱指数时间或弱指数时间复杂性[15-16].

多数程序设计者更倾向于采用局部搜索方法设计解答 Max-Sat、Max-k-Sat 及 Max-(k)-Sat 问题的求解算法,这是因为局部搜索算法步骤简单,且实际求解经验表明,局部搜索算法的求解性能往往十分理想.最早的 Max-Sat 局部搜索算法研究可追溯到

Gu[17]与 Selman 等[18]的求解实验以及 Papadimitriou[19]与 Koutsoupias[20]的理论分析. Max-Sat 问题局部搜索算法的基本步骤为：(1) 随机为布尔变量赋初值；(2) 重复修正部分布尔变量赋值，以改善当前解质量，直到不能修正为止. 这种局部搜索算法需要重新选择初始赋值，多次运行，才能保证以较大概率获得 Max-Sat 精确解[21]. 另一类局部搜索算法，其修正布尔变量赋值的步骤也是随机的，可保证算法只需选择一次初始赋值，就可概率为 1 地获得 Max-Sat 精确解[22]. 合取范式可满足问题的算法及其复杂性的其他研究结果可见文献[23-25].

虽然实践表明局部搜索算法的求解性能十分理想，但未见有人给出过局部搜索算法解答 Max-Sat 问题及其子问题的性能分析结果. 本文研究解答 Max-(3)-Sat 问题的局部搜索算法，并分析算法的求解性能. 证明一位跳变局部搜索算法解答 Max-(3)-Sat问题的近似性能比为 4/3；证明一位跳变加全体跳变的局部搜索算法解答 Max-(3)-Sat 问题的近似性能比为 5/4；在前述性能分析的基础上，设计一个解答 Max-(3)-Sat 问题的新局部搜索算法，证明新局部搜索算法的近似性能比为 8/7. 我们可针对 3 个算法分别给出 Max-(3)-Sat 实例，使算法解答它们的近似性能比恰好达到前面所证明的近似性能比. 根据 Hastad 的反面结果，新算法所能达到的近似性能比‘8/7’，是多项式时间能够逼近 Max-(3)-Sat 问题的最好近似性能比. 另外可将解答 Max-(3)-Sat 问题的 8/7-近似局部搜索算法推广为解答 Max-(k)-Sat 问题的局部搜索算法，近似性能比为 $(2k+2)/(2k+1)$，$k\geq 4$. 最后设计解答 Max-(k)-Sat 问题的两位跳变局部搜索算法，证明算法近似性能比为 $1+\dfrac{1}{2k+1+k(k-1)/(n-k)}$，$k\geq 4$.

本文局部搜索算法可以运行多次，从中寻找最好的解. 因为每次运行的近似性能比最大为 8/7，所以多次运行后，算法得到的解往往十分接近最优或已经是最优解. Johnson 算法虽然解答 Max-(3)-Sat 问题的近似性能比也为 8/7，但算法每次运行只能得到同样的解. 随机算法虽然通过去随机过程可以保证算法能够达到其平均近似性能比 8/7，但去随机的时间复杂性太高，影响算法的实用性.

第 2 节给出解答 Max-(3)-Sat 问题的一位跳变局部搜索算法，证明其近似性能比为 4/3；第 3 节给出 Max-(3)-Sat 问题的一位跳变后跟全体跳变局部搜索算法，证明其近似性能比为 5/4；第 4 节仍然利用一位跳变和全体跳变，设计解答 Max-(3)-Sat 问题的新局部搜索算法，证明其近似性能比为 8/7；第 5 节利用解答 Max-(3)-Sat 问题的局部搜索方法，设计解答 Max-($k\geq 4$)-Sat 问题的局部搜索算法；第 6 节分析两位跳变局部搜索算法的性能.

2　一位跳变的 4/3-近似局部搜索算法

Max-(3)-SAT 问题形式化描述为

实例. 布尔变量集合 $U=\{u_1,\cdots,u_n\}$，项集合 $C=\{C_1,\cdots,C_m\}$，每个项均含有不少于 3 个布尔变量字母. $C_t=\{x[t,1],\cdots,x[t,k_t]\}$，其中 $x[t,j]\in\{u_1,\cdots,u_n,\bar{u}_1,\cdots,\bar{u}_n\}$，$1\leq t\leq m$，$1\leq j\leq k_t$，$k_t\geq 3$.

询问. 求 U 中布尔变量的赋值 $a:U\to\{T,F\}$，最大化 $|\{C_t:a(x[t,1])\vee\cdots\vee a(x[t,k_t])=T\}|$.

在 Max-(3)-SAT 实例中，u_i 和 $\bar{u}_i$ 均被称为布尔变量字母，$u_i(\bar{u}_i)$ 称为 $\bar{u}_i(u_i)$ 的反变量或反变量字母，$1\leq i\leq n$. 布尔变量赋值也称为布尔变量字母赋值. $a(x[t,j])$ 表示由 a 为 U 中布尔变量赋值后，布尔变量字母 $x[t,j]$ 的取值.

为表述简单，我们总是假设每个项含有 3 个布尔变量字母，并只针对这种问题实例给出算法和性能分析. 适当修改算法即可用于解答那些每个项含有不少于 3 个字母的 Max-(3)-Sat 实例，近似性能比分析的结论仍然成立.

设 U 的赋值 a 使 $a(x[t,1])\vee a(x[t,2])\vee a(x[t,3])=T$，则称 C_t 被 a 满足. 下面也直接称 C_t 被满足为 C_t 满足，C_t 不被满足为 C_t 不满足. 若一个项同时含有 u_i 和 $\bar{u}_i$，则无论 u_i 怎样赋值，这个项均被满足，因此下面总是假设 Max-(3)-SAT 实例中，任意项均不同时含有 u_i 和 $\bar{u}_i$.

定义 1.　给定布尔变量赋值 $a(U)$，若项 C_t 含有布尔变量字母 u_i 且 $a(u_i)=T$，或 C_t 含有布尔变量字母 $\bar{u}_i$ 且 $a(u_i)=F$，称 C_t 被 $a(u_i)$ 满足；若项 C_t 含有布尔变量字母 u_i 且 $a(u_i)=F$，或 C_t 含有布尔变量字母 $\bar{u}_i$ 且 $a(u_i)=T$，称 C_t 不被 $a(u_i)$ 满足. 将 C_t 被 $a(u_i)$ 满足简记为 $a(C_t,u_i)=T$，C_t 不被 $a(u_i)$ 满足简记为 $a(C_t,u_i)=F$. 若 C_t 不含有布尔变量字母 u_i，则记 $a(C_t,u_i)=N$.

C_t 被 $a(u_i)$ 满足，则 C_t 肯定被 $a(U)$ 满足. 考虑

如下的局部搜索方法:先任意选择 U 中布尔变量的赋值,然后选择一个布尔变量将其赋值取反,以增加 C 中被满足的项数目.重复选择布尔变量并将赋值取反的步骤,一直进行到不存在布尔变量,将其赋值取反可增加 C 中被满足的项数目为止.将这种方法称为解答 Max-(3)-Sat 问题的一位跳变算法.

设 $C[j,T]$ 表示被 $a(u_j)$ 满足,但不被其它布尔变量赋值满足的项集合,$C[j,F]$ 表示不被 $a(u_j)$ 满足,且不满足的项集合,即 $C[j,T]=\{C_t \mid a(C_t,u_j)=T,\ a(C_t,u_x)\in\{F,N\},\ x\neq j\}$,$C[j,F]=\{C_t \mid a(C_t,u_j)=F,\ a(C_t,u_x)\in\{F,N\},\ x\neq j\}$,$1\leq j\leq n$.为布尔变量赋值的一位跳变局部搜索算法如下.

算法 1. *one-flip*(U,C).

1. $a(u_k)\leftarrow Rand\{T,F\}$, $1\leq k\leq n$;
2. While (存在 u_j,使 $|C[j,T]|<|C[j,F]|$) do
3. $\quad a(u_j)\leftarrow \overline{a(u_j)}$;
4. End While
5. Return $a(U)$.

算法中,$Rand\{T,F\}$ 表示随机选择'T'或'F'的函数.算法的 While 循环中,执行一次语句 3 的变量赋值取反操作,则 $C[j,F]$ 中的项由不满足变为满足,$C[j,T]$ 中的项由满足变为不满足,满足的项数目至少增加 1.因 C 中只有 m 个项能被满足,所以算法 While 循环一定能够结束.对于每个布尔变量,计算 $C[\cdot,T]$ 和 $C[\cdot,F]$ 的时间复杂性为 $O(m)$,所以算法在 While 循环中找到一个 u_j 满足 $|C[j,T]|<|C[j,F]|$ 的时间复杂性为 $O(mn)$.每循环一次,被满足的项数目至少增加 1,所以算法时间复杂性为 $O(m^2n)$.下面分析算法的性能.

引理 1. 算法结束时,得到布尔变量赋值 $a(U)$,对于每个 u_j,$1\leq j\leq n$,总有 $|C[j,T]|\geq|C[j,F]|$.

证明. 若存在 u_j,满足 $|C[j,T]|<|C[j,F]|$,由 While 循环的条件可知算法循环不会结束.

证毕.

引理 2. 设算法结束时,被 $a(U)$ 满足的项集合为 $S[a]$,则 $|S[a]|\geq\sum_{j=1}^{n}|C[j,T]|$.

证明. 设 $C_t=\{x[t,1],x[t,2],x[t,3]\}\in S[a]$,$x[t,j]\in\{u_1,\cdots,u_n,\bar{u}_1,\cdots,\bar{u}_n\}$,$j\in\{1,2,3\}$.

若 C_t 中至少有两个布尔变量字母取值为 T,则 $x[t,1]$,$x[t,2]$,$x[t,3]$ 中每个字母赋值取反,C_t 仍然满足;其它字母赋值取反,不影响 C_t 的可满足性.所以由 $C[j,T]$ 的定义可知,$C_t\notin C[j,T]$,$1\leq j\leq n$.

若 C_t 中只有一个布尔变量字母取值为 T,不妨设 $a(x[t,1])=T$,则仅当 $x[t,1]$ 赋值取反时,C_t 由满足变为不满足.设 $x[t,1]\in\{u_j,\bar{u}_j\}$,则 $C_t\in C[j,T]$,但对于 $x\neq j$,$C_t\notin C[x,T]$,即 C_t 最多出现在一个 $C[j,T]$ 中,所以 $|S[a]|\geq\sum_{j=1}^{n}|C[j,T]|$.

证毕.

引理 3. 设算法结束时,不被 $a(U)$ 满足的项集合为 $NS[a]$,则 $\sum_{j=1}^{n}|C[j,F]|\geq 3|NS[a]|$.

证明. 我们证明 $NS[a]$ 中的每个项在 $C[1,F],\cdots,C[n,F]$ 中至少出现 3 次,即任意 $C_t\in NS[a]$,存在至少 3 个正整数 j_1,j_2,j_3,使 $C_t\in C[j_1,F]$,$C_t\in C[j_2,F]$,$C_t\in C[j_3,F]$.

设 $C_t=\{x[t,1],x[t,2],x[t,3]\}\in NS[a]$,$a(x[t,1])=F$,$a(x[t,2])=F$,$a(x[t,3])=F$.$x[t,1]$ 或 $x[t,2]$ 或 $x[t,3]$ 赋值取反,C_t 会由不满足变为满足,$a(C_t,\overline{x[t,1]})=a(C_t,\overline{x[t,2]})=a(C_t,\overline{x[t,3]})=T$.设 $x[t,1]\in\{u_{j_1},\bar{u}_{j_1}\}$,$x[t,2]\in\{u_{j_2},\bar{u}_{j_2}\}$,$x[t,3]\in\{u_{j_3},\bar{u}_{j_3}\}$,则 $C_t\in C[j_1,F]$,$C_t\in C[j_2,F]$,$C_t\in C[j_3,F]$.所以 $\sum_{j=1}^{n}|C[j,F]|\geq 3|NS[a]|$.

证毕.

定理 1. 设算法结束时,被 $a(U)$ 满足的项集合为 $S[a]$,则 $|S[a]|\geq\frac{3}{4}|C|$.

证明. 由引理 2、引理 1 和引理 3 得

$$|S[a]|\geq\sum_{j=1}^{n}|C[j,T]|\geq\sum_{j=1}^{n}|C[j,F]|\geq 3|NS[a]| \tag{1}$$

所以,$|C|=|S[a]|+|NS[a]|\leq\frac{4}{3}|S[a]|$,即 $|S[a]|\geq\frac{3}{4}|C|$.

证毕.

由定理 1 可知,算法 *one-flip*$(\cdot)$ 解答 Max-(3)-Sat 问题的近似性能比不大于 4/3.

例. $U=\{u_1,u_2,u_3\}$,$C=\{C_1=\{u_1,u_2,u_3\},C_2=\{\bar{u}_1,u_2,u_3\},C_3=\{u_1,\bar{u}_2,u_3\},C_4=\{u_1,u_2,\bar{u}_3\}\}$.当 $a(u_1)=a(u_2)=a(u_3)=F$ 时,算法 *one-flip*$(\cdot)$ 得到一个局部最优解,C_1 不能被 $a(U)$ 满足,C_2,C_3,C_4 均被满足,算法解答这个实例的近似性能比恰为 4/3.

若 Max-(3)-Sat 实例中,某些项含有多于 3 个布尔变量字母,算法 *one-flip*$(\cdot)$ 并不需要任何改

变，即可解答这样的实例，近似性能比仍为 4/3.

3　一位跳变加全体跳变 5/4-近似局部搜索算法

一个项 C_t 被满足，其中的 3 个布尔变量字母可能有 1 个字母赋值为 T，或有两个字母赋值为 T，或有 3 个字母赋值为 T. 还可观察到，前面定义的 $C[j,T](1\leq j\leq n)$ 中仅含有一个字母赋值为 T 而被 $a(U)$ 满足的项. 为更准确地度量被满足的项数目，我们根据一个项中含有的被赋值为‘T’的字母数目，进一步将项分类.

定义 2.　给定 Max-(3)-Sat 实例 U，C，设 $a:U\rightarrow\{T,F\}$ 为布尔变量赋值，$C_t\in C$ 为任意含有 $k_t(\geq 3)$ 个字母的项. 对于 j，$0\leq j\leq k_t$，若 C_t 中有 j 个布尔变量字母赋值为 T，另外字母赋值为 F，则称 C_t 关于 $a(U)$ 是 j-满足的.

一个项是 0-满足即这个项不被满足；一个项是 $j(\geq 1)$-满足的，则这个项肯定被满足. 若一个项含有的所有布尔变量字母均被赋值为 T，则称这个项为全满足的；若一个项被满足，但不是全满足的，则称这个项为半满足的.

利用全满足和不满足的项可以因布尔变量赋值全体跳变而相互转化，可给出解答 Max-(3)-Sat 问题性能更好的算法. 下面仍然假设每个项总含有 3 个布尔变量字母.

设 $a(U)$ 为 U 的赋值，仍然设 $S[a]$ 和 $NS[a]$ 分别表示被 $a(U)$ 满足和不被 $a(U)$ 满足的项集合，其中设 $S[a,1]$、$S[a,2]$、$S[a,3]$ 分别表示关于 $a(U)$ 是 1-满足、2-满足和 3-满足的项集合. 显然 $S[a]=S[a,1]\cup S[a,2]\cup S[a,3]$.

引理 4.　任给布尔变量赋值 $a(U)$，$\bar{a}(U)$ 是 $a(U)$ 中每个布尔变量赋值均取反得到的布尔变量赋值，则 $S[a,1]=S[\bar{a},2]$，$S[a,2]=S[\bar{a},1]$，$S[a,3]=NS[\bar{a}]$，$NS[a]=S[\bar{a},3]$.

证明.　考察 $C_t=\{x[t,1],x[t,2],x[t,3]\}$，若 $C_t\in S[a,1]$，不妨设 $a(C_t,x[t,1])=T$，则 $a(C_t,x[t,2])=F$，$a(C_t,x[t,3])=F$. 将 U 中布尔变量赋值全部取反，得到 $\bar{a}(C_t,x[t,1])=F$，$\bar{a}(C_t,x[t,2])=T$，$\bar{a}(C_t,x[t,3])=T$. 所以 C_t 关于 $\bar{a}(U)$ 是 2-满足的. 每个关于 $\bar{a}(U)$ 为 2-满足的项均可由关于 $a(U)$ 为 1-满足的项其布尔变量赋值取反得到. 所以 $S[a,1]=S[\bar{a},2]$. 同理可证：$S[a,2]=S[\bar{a},1]$，$S[a,3]=NS[\bar{a}]$，$NS[a]=S[\bar{a},3]$.

证毕.

推论 1.　任给布尔变量赋值 $a(U)$，则必有 $|S[a,3]|\geq|NS[a]|$ 或 $|S[\bar{a},3]|\geq|NS[\bar{a}]|$.

证明.　若 $|S[a,3]|\geq|NS[a]|$，则结论成立. 否则由引理 4 的结论立即得到 $|S[\bar{a},3]|=|NS[a]|>|S[a,3]|=|NS[\bar{a}]|$.　证毕.

由引理 4 和推论 1，可以在一位跳变局部搜索结束后，利用所有布尔变量赋值全部取反，进一步改善算法性能.

算法 2.　*all-flip*(U,C).

1. 调用 *one-flip*(U,C) 得到布尔变量赋值 $a(U)$；
2. If ($|S[a,3]|\geq|NS[a]|$) then $b(U)\leftarrow a(U)$；
3. Else $b(U)\leftarrow\bar{a}(U)$；
4. End if
5. Return $b(U)$

因为算法 *one-flip*$(\cdot)$ 一定结束，所以该算法也必定结束. 调用 *one-flip*$(\cdot)$ 得到布尔变量赋值 $a(U)$，计算 $S[a,1]$、$S[a,2]$、$S[a,3]$、$NS[a]$ 的时间复杂性为 $O(m)$，所以执行算法步 2～5 的时间复杂性为 $O(m)$. 所以算法时间复杂性与 *one-flip*$(\cdot)$ 的时间复杂性相同，为 $O(m^2n)$. m，n 分别为 Max-(3)-SAT 实例中项数目和布尔变量数目. 下面分析算法 *all-flip*$(\cdot)$ 的性能.

引理 5.　设算法 *all-flip*$(\cdot)$ 得到的布尔变量赋值为 $b(U)$，则 $|S[b,1]|+|S[b,2]|\geq 3|NS[b]|$.

证明.　设调用算法 *one-flip*$(\cdot)$ 得到的布尔变量赋值为 $a(U)$，由 $C[j,T]$ 的定义可知，对于任意 j，$1\leq j\leq n$，有 $S[a,3]\cap C[j,T]=\varnothing$ 且 $S[a,2]\cap C[j,T]=\varnothing$，$S[a,1]$ 中的每个项包含在一个且仅一个 $C[j,T]$ 中，所以引理 2 的结论可更确切地表示为 $|S[a,1]|=\sum_{j=1}^{n}|C[j,T]|$，即 $|S[a,1]|+|S[a,2]|\geq\sum_{j=1}^{n}|C[j,T]|$. 因此由定理 1 的证明可得到 $|S[a,1]|+|S[a,2]|\geq 3|NS[a]|$. 由引理 4 和推论 1 知，$|S[b,1]|+|S[b,2]|=|S[a,1]|+|S[a,2]|$，$|NS[a]|\geq|NS[b]|$. 所以 $|S[b,1]|+|S[b,2]|\geq 3|NS[b]|$.　证毕.

定理 2.　设算法 *all-flip*$(\cdot)$ 得到的布尔变量赋值为 $b(U)$，$S[b]$ 和 $NS[b]$ 为被 $b(U)$ 满足和不被 $b(U)$ 满足的项集合，则 $|S[b]|\geq\frac{4}{5}|C|$.

证明. 由算法 2～4 步的条件语句和推论 1 可知，$|S[b,3]| \geq |NS[b]|$. 由引理 5，$|S[b,1]|+|S[b,2]| \geq 3|NS[b]|$. 所以 $|S[b]|=|S[b,1]|+|S[b,2]|+|S[b,3]| \geq 4|NS[b]|$. 所以 $|S[b]| \geq \frac{4}{5}|C|$. 证毕.

由定理 2 可知，算法 $all\text{-}flip(\cdot)$ 的近似性能比为 5/4.

例. $U=\{u_1,u_2,u_3\}$，$C=\{C_1=\{u_1,u_2,u_3\}$，$C_2=\{\bar{u}_1,u_2,u_3\}$，$C_3=\{u_1,\bar{u}_2,u_3\}$，$C_4=\{u_1,u_2,\bar{u}_3\}$，$C_5=\{\bar{u}_1,\bar{u}_2,\bar{u}_3\}\}$. 当 $a(u_1)=a(u_2)=a(u_3)=F$ 时，算法 $all\text{-}flip(\cdot)$ 得到一个局部最优解，C_1 不被 $a(U)$ 满足，C_2、C_3、C_4 和 C_5 均被 $a(U)$ 满足，算法解答这个实例的近似性能比恰为 5/4.

若 Max-(3)-Sat 实例中存在含有多于 3 个字母的项，则在算法 $all\text{-}flip(\cdot)$ 中，应比较全满足与不满足的项数目，决定是否将所有布尔变量赋值取反. 所有布尔变量赋值有条件取反的步骤，使得全满足项的数目不少于不满足项的数目，且不改变半满足项的数目，半满足的项数目仍然不少于不满足项数目的 3 倍. 所以算法解答这种实例的近似性能比仍不大于 5/4.

4 一位跳变加全体跳变 8/7-近似局部搜索算法

前面算法考虑的布尔变量赋值所满足的项，仍然未能包括 2-满足的项. 下面假设每个项均含有 3 个布尔变量字母，我们利用一位跳变，最大化 1-满足和 2-满足项集合的局部搜索方法，改善算法性能.

给定布尔变量赋值 $a(U)$. 选择任意一个布尔变量赋值取反，任意项 C_t 的可满足性变化有如下性质：

(1) 若 $C_t \in NS[a]$，则 C_t 仍为不满足或变为 1-满足，不可能变为 2-满足或 3-满足；

(2) 若 $C_t \in S[a,1]$，则 C_t 仍为 1-满足或变为 2-满足或不满足，不可能变为 3-满足；

(3) 若 $C_t \in S[a,2]$，则 C_t 仍为 2-满足或变为 1-满足或 3-满足，不可能变为不满足；

(4) 若 $C_t \in S[a,3]$，则 C_t 仍为 3-满足或变为 2-满足，不可能变为 1-满足或不满足.

由此，设 $C[j,0]$ 表示将 $a(u_j)$ 取反，由不满足变为 1-满足的项集合；$C[j,1]$ 表示将 $a(u_j)$ 取反，由 1-满足变为不满足的项集合；$C[j,2]$ 表示将 $a(u_j)$ 取反，由 2-满足变为 3-满足的项集合；$C[j,3]$ 表示将 $a(u_j)$ 取反，由 3-满足变为 2-满足的项集合. 即

$$C[j,0]=\{C_t \mid C_t \in NS[a],\ a(C_t,u_j)=F\} \quad (2)$$

$$C[j,1]=\{C_t \mid C_t \in S[a,1],\ a(C_t,u_j)=T\} \quad (3)$$

$$C[j,2]=\{C_t \mid C_t \in S[a,2],\ a(C_t,u_j)=F\} \quad (4)$$

$$C[j,3]=\{C_t \mid C_t \in S[a,3],\ a(C_t,u_j)=T\} \quad (5)$$

其中，$1 \leq j \leq n$. 一个布尔变量赋值取反跳变，最大化 1-满足和 2-满足项集合的算法如下.

算法 3. $oneflip\text{-}max3sat(U,C)$.

1. $a(u_k) \leftarrow Rand\{T,F\}$，$1 \leq k \leq n$；
2. While(存在 u_j，使 $|C[j,1]|+|C[j,2]|<|C[j,0]|+|C[j,3]|$) do
3. $\quad a(u_j) \leftarrow \overline{a(u_j)}$；
4. End While
5. Return $a(U)$

算法的 While 循环中，执行一次语句 3 的变量赋值取反操作，则 $C[j,0]$ 中的项由不满足变为 1-满足；$C[j,3]$ 中的项由 3-满足变为 2-满足；$C[j,1]$ 中的项由 1-满足变为不满足；$C[j,2]$ 中的项由 2-满足变为 3-满足，1-满足与 2-满足的项数目之和至少增加 1. 因 C 中只有 m 个项，所以算法 While 循环一定能够结束. 对于每个布尔变量，计算 $|C[j,\cdot]$ 的时间复杂性为 $O(m)$，所以算法在 While 循环中找到一个 u_j 满足 $|C[j,1]|+|C[j,2]|<|C[j,0]|+|C[j,3]|$ 的时间复杂性为 $O(mn)$. 每循环一次，1-满足和 2-满足的项数目至少增加 1，所以算法时间复杂性为 $O(m^2n)$. 下面再分析算法的性能.

引理 6. 设算法 $oneflip\text{-}max3sat(U,C)$ 得到布尔变量赋值 $a(U)$，则对于任意 u_j，有 $|C[j,1]|+|C[j,2]| \geq |C[j,0]|+|C[j,3]|$.

证明. 若存在 u_j，满足 $|C[j,1]|+|C[j,2]|<|C[j,0]|+|C[j,3]|$，由算法的循环条件可知 While 循环不会结束. 证毕.

引理 7. 设算法结束时得到布尔变量赋值 $a(U)$，则 $|S[a,1]|+|S[a,2]|=\sum_{j=1}^{n}|C[j,1]|+\sum_{j=1}^{n}|C[j,2]|$.

证明. 设 $C_t=\{x[t,1],x[t,2],x[t,3]\}$，$x[t,j] \in \{u_1,\cdots,u_n,\bar{u}_1,\cdots,\bar{u}_n\}$，$j \in \{1,2,3\}$. 若 $C_t \in S[a,1]$，不失一般性，设 $a(x[t,1])=T$，$a(x[t,2])=a(x[t,3])=F$. 当且仅当 $x[t,1]$ 赋值取反时，C_t 变为不满足，其它布尔变量赋值取反，C_t 仍为 1-满足或

变为 2-满足，设 $x[t,1]\in\{u_j,\bar{u}_j\}$，则 $C_t\in C[j,1]$，但对于 $x\neq j$，$C_t\notin C[x,1]$，即 C_t 出现在一个且仅一个 $C[j,1]$ 中，所以 $|S[a,1]|=\sum_{j=1}^{n}|C[j,1]|$. 同理可证 $|S[a,2]|=\sum_{j=1}^{n}|C[j,2]|$. 即有 $|S[a,1]|+|S[a,2]|=\sum_{j=1}^{n}|C[j,1]|+\sum_{j=1}^{n}|C[j,2]|$.　　证毕.

引理 8.　设算法结束时得到布尔变量赋值 $a(U)$，则 $3|S[a,3]|+3|NS[a]|=\sum_{j=1}^{n}|C[j,3]|+\sum_{j=1}^{n}|C[j,0]|$.

证明.　我们证明任意 $C_t\in NS[a]$，存在 3 个且仅 3 个正整数 j_1，j_2，j_3，使 $C_t\in C[j_1,0]$，$C_t\in C[j_2,0]$，$C_t\in C[j_3,0]$.

设 $C_t=\{x[t,1],x[t,2],x[t,3]\}\in NS[a]$，$a(x[t,1])=a(x[t,2])=a(x[t,3])=F$. 只有 $x[t,1]$ 或 $x[t,2]$ 或 $x[t,3]$ 赋值取反，C_t 会由不满足变为 1-满足，即 $a(C_t,\overline{x[t,1]})=a(C_t,\overline{x[t,2]})=a(C_t,\overline{x[t,3]})=T$. 设 $x[t,1]\in\{u_{j_1},\bar{u}_{j_1}\}$，$x[t,2]\in\{u_{j_2},\bar{u}_{j_2}\}$，$x[t,3]\in\{u_{j_3},\bar{u}_{j_3}\}$，则 $C_t\in C[j_1,0]$，$C_t\in C[j_2,0]$，$C_t\in C[j_3,0]$，而对于任意 $x\notin\{j_1,j_2,j_3\}$，有 $a(C_t,u_x)=a(C_t,\bar{u}_x)=N$，所以 $C_t\notin C[x,0]$. 所以 $\sum_{j=1}^{n}|C[j,0]|=3|NS[a]|$. 同理可证 $\sum_{j=1}^{n}|C[j,3]|=3|S[a,3]|$. 所以，$\sum_{j=1}^{n}|C[j,0]|+\sum_{j=1}^{n}|C[j,3]|=3|NS[a]|+3|S[a,3]|$.　　证毕.

定理 3.　设算法结束时得到布尔变量赋值 $a(U)$，则 $|S[a,1]|+|S[a,2]|\geq 6\min\{|NS[a]|,|S[a,3]|\}$.

证明.　由引理 7、引理 6 和引理 8 得

$$\begin{aligned}|S[a,1]|+|S[a,2]|&=\sum_{j=1}^{n}|C[j,1]|+\sum_{j=1}^{n}|C[j,2]|\\&\geq\sum_{j=1}^{n}|C[j,0]|+\sum_{j=1}^{n}|C[j,3]|\\&=3|NS[a]|+3|S[a,3]|\\&\geq 6\min\{|NS[a]|,|S[a,3]|\}\end{aligned}\qquad(6)$$

定理得证.　　证毕.

由定理 3，若 $|S[a,3]|\geq|NS[a]|$，则算法 *one-flip-max3sat*(·) 得到的解 $a(U)$，使 1-满足与 2-满足的项数目之和至少为不满足项数目的 6 倍. 但若 $|S[a,3]|<|NS[a]|$，如 $|S[a,3]|=0$，则算法得到的解只能保证 1-满足和 2-满足的项数目不少于不满足项数目的 3 倍. 在算法中增加所有布尔变量赋值有条件取反的步骤，可使算法的近似性能比达到 8/7.

算法 4.　*allflip-max3sat*(U,C).

1. 调用 *oneflip-max3sat*(U,C) 得到布尔变量赋值 $a(U)$；
2. If ($|S[a,3]|\geq|NS[a]|$) then $b(U)\leftarrow a(U)$；
3. Else $b(U)\leftarrow\bar{a}(U)$；
4. End if
5. Return $b(U)$

该算法的时间复杂性仍为 $O(m^2n)$，我们不再具体分析算法的时间复杂性.

引理 9.　设算法 *allflip-max3sat*(·) 得到的布尔变量赋值为 $b(U)$，则 $|S[b,3]|\geq|NS[b]|$.

证明.　观察算法 *allflip-max3sat*(·) 的步 2～4，由引理 4 和推论 1 即得 $|S[b,3]|\geq|NS[b]|$.

定理 4.　设算法 *allflip-max3sat*(·) 得到的布尔变量赋值为 $b(U)$，则 $|S[b]|\geq\frac{7}{8}|C|$.

证明.　设调用算法 *oneflip-max3sat*(·) 得到的布尔变量赋值 $a(U)$. 由引理 4 和推论 1 得，$|S[b,1]|+|S[b,2]|=|S[a,1]|+|S[a,2]|$，由引理 9，$|NS[b]|=\min\{|S[b,3]|,|NS[b]|\}$. 所以由定理 3 我们得到 $|S[b]|=|S[b,1]|+|S[b,2]|+|S[b,3]|\geq 6\min\{|S[b,3]|,|NS[b]|\}+|S[b,3]|\geq 7|NS[b]|$. 所以 $|S[b]|\geq\frac{7}{8}|C|$.　证毕.

定理 4 说明算法 *allflip-max3sat*(·) 的近似性能比为 8/7.

例.　设 $U=\{u_1,u_2,u_3,u_4,u_5,u_6\}$，$C=\{C_1,C_2,C_3,C_4,C_5,C_6,C_7,C_8,C_9,C_{10},C_{11},C_{12},C_{13},C_{14},C_{15},C_{16}\}$. 每个项的布尔变量字母集表示如下：

$NS[a]$	$C_1=\{u_1,u_2,u_3\}$，$C_2=\{u_4,u_5,u_6\}$.
$S[a,3]$	$C_3=\{\bar{u}_1,\bar{u}_2,\bar{u}_3\}$，$C_4=\{\bar{u}_4,\bar{u}_5,\bar{u}_6\}$.
$S[a,1]$	$C_5=\{\bar{u}_1,u_4,u_5\}$，$C_6=\{\bar{u}_2,u_5,u_6\}$，$C_7=\{\bar{u}_3,u_6,u_4\}$，$C_8=\{\bar{u}_4,u_1,u_2\}$，$C_9=\{\bar{u}_5,u_2,u_3\}$，$C_{10}=\{\bar{u}_6,u_3,u_1\}$.
$S[a,2]$	$C_{11}=\{u_1,\bar{u}_4,\bar{u}_5\}$，$C_{12}=\{u_2,\bar{u}_5,\bar{u}_6\}$，$C_{13}=\{u_3,\bar{u}_6,\bar{u}_4\}$，$C_{14}=\{u_4,\bar{u}_1,\bar{u}_2\}$，$C_{15}=\{u_5,\bar{u}_2,\bar{u}_3\}$，$C_{16}=\{u_6,\bar{u}_3,\bar{u}_1\}$.

当 $a(u_1)=a(u_2)=a(u_3)=a(u_4)=a(u_5)=a(u_6)=F$ 时，算法得到局部最优解，$NS[a]=\{C_1,C_2\}$，$S[a]=\{C_3,\cdots,C_{16}\}$，$S[a]=\frac{7}{8}|C|$. 但算法的最优解可使所有项都被满足，算法解答这个实

例的近似性能比恰为 8/7.

若 Max-(3)-Sat 实例中存在含有多于 3 个字母的项,则在算法 $oneflip\text{-}max3sat(\cdot)$ 中,应利用一位跳变最大化半满足的项,从而可证明局部最优解得到的布尔变量赋值所产生的半满足项数目,不少于全满足和不满足项数目总和的 3 倍.因此在算法 $allflip\text{-}max3sat(\cdot)$ 中应比较全满足与不满足的项数目,决定是否将所有布尔变量赋值取反.所有布尔变量赋值有条件取反的步骤,使得全满足项的数目不少于不满足项的数目,且不改变半满足的项数目.因此半满足项与全满足项数目之和不少于不满足项数目的 7 倍,算法近似性能比仍不大于 8/7.

5 Max-(k)-Sat 问题$(2k+2)/(2k+1)$-近似算法

每个项均含有不少于 k 个布尔变量字母的 Max-Sat 子问题即为 Max-(k)-Sat 问题.不难推知,算法 $one\text{-}flip(\cdot)$ 解答 Max-Sat 问题的近似性能比为 2,解答 Max-(2)-Sat 问题的近似性能比为 3/2.算法 $all\text{-}flip(\cdot)$ 解答 Max-(2)-Sat 问题的近似能比为 4/3.本节考虑 $k\geq 4$ 的情况,利用类似于解答 Max-(3)-Sat 问题的局部搜索方法,给出解答 Max-(k)-Sat 问题的一位跳变后跟全体跳变的局部搜索算法,使其近似性能比达到 $\frac{2k+2}{2k+1}$,下面总假设 $k\geq 4$,且假设 Max-(k)-Sat 问题实例中,每个项均含有 k 个布尔变量字母.不再讨论实例中存在含有多于 k 个字母项的情况.

设 $C_t=\{x[t,1],x[t,2],\cdots,x[t,k]\}$,给定布尔变量赋值 $a(U)$.延用前面项关于(被)布尔变量赋值 j-满足的概念,$0\leq j\leq k$,其中 0-满足为不满足;k-满足为全满足;$[1\sim k-1]$-满足均为半满足.另设 $S[a,j]$ 为关于布尔变量赋值 $a(U)$ 是 j-满足的项集合.$S[a]$ 和 $NS[a]$ 为被 $a(U)$ 满足和不被 $a(U)$ 满足的项集合.

给定 Max-(k)-Sat 问题实例的布尔变量赋值 $a(U)$,一个布尔变量赋值取反,任意项 C_t 的可满足性变化有如下性质:

(1)若 $C_t\in NS[a]$,则 C_t 仍为不满足或变为 1-满足,不可能变为 $j(\geq 2)$-满足.

(2)若 $C_t\in S[a,j]$,$1\leq j\leq k-1$,则 C_t 仍为 j-满足或变为 $(j+1)$-满足或 $(j-1)$-满足,不可能变为 $(j+x)$-满足或 $(j-y)$-满足,$2\leq x\leq k-j$,$2\leq y\leq j$.

(3)若 $C_t\in S[a,k]$,则 C_t 仍为 k-满足或变为 $(k-1)$-满足,不可能变为 $(k-x)$-满足,$2\leq x\leq k$.

一个布尔变量赋值取反,对于满足 $2\leq j\leq k-2$ 的 j,j-满足的项不可能变为不满足或 k-满足;不满足或 k-满足的项也不可能变为 j-满足.由此我们仍然假设 $C[j,0]$ 表示将 $a(u_j)$ 取反,由不满足变为 1-满足的项集合;$C[j,1]$ 表示将 $a(u_j)$ 取反,由 1-满足变为不满足的项集合;$C[j,k-1]$ 表示将 $a(u_j)$ 取反,由 $(k-1)$-满足变为 k-满足的项集合;$C[j,k]$ 表示将 $a(u_j)$ 取反,由 k-满足变为 $(k-1)$-满足的项集合.即

$$C[j,0]=\{C_t\mid C_t\in NS[a],a(C_t,u_j)=F\} \quad (7)$$

$$C[j,1]=\{C_t\mid C_t\in S[a,1],a(C_t,u_j)=T\} \quad (8)$$

$$C[j,k-1]=\{C_t\mid C_t\in S[a,k-1],\ a(C_t,u_j)=F\} \quad (9)$$

$$C[j,k]=\{C_t\mid C_t\in S[a,k],\ a(C_t,u_j)=T\} \quad (10)$$

其中,$1\leq j\leq n$.下面给出一个布尔变量赋值取反跳变,最大化 1-满足至 $(k-1)$-满足的项集合,然后所有布尔变量赋值有条件取反的算法.

算法 5. $flip\text{-}maxksat(U,C)$.

1. $a(u_k)\leftarrow Rand\{T,F\}$,$1\leq k\leq n$;
2. While (存在 u_j,使 $|C[j,1]|+|C[j,k-1]|<|C[j,0]|+|C[j,k]|$) do
3. $\quad a(u_j)\leftarrow\overline{a(u_j)}$;
4. End While
5. If ($|S[a,k]|\geq|NS[a]|$) then $b(U)\leftarrow a(U)$;
6. $\quad$ Else $b(U)\leftarrow\bar{a}(U)$;
7. End if
8. Return $b(U)$.

在算法 While 循环条件中,仅比较了 1-满足转化为不满足和 $(k-1)$-满足转化为 k-满足项,与不满足转化为 1-满足和 k-满足转化为 $(k-1)$-满足项的数目,这是因为 2-满足至 $(k-2)$-满足的项不可能因一个布尔变量赋值取反转化为不满足或 k-满足,反之亦然.算法的 While 循环中,执行一次语句 3 的变量赋值取反操作,则 $C[j,0]$ 中的项由不满足变为 1-满足;$C[j,k]$ 中的项由 k-满足变为 $(k-1)$-满足;$C[j,1]$ 中的项由 1 满足变为不满足;$C[j,k-1]$ 中的项由 $(k-1)$-满足变为 k-满足,$[1\sim k-1]$-满足的项数目至少增加 1.因 C 中只有 m 个项,所以算法 While 循环一定能够结束.注意,执行一次语句 3 的变量赋值取反操作,1-满足和 $(k-1)$-满足的项数目未必一定增加,这是因为 1-满足的项可能变为 2-满足;$(k-1)$-满足的项可能变为 $(k-2)$-满足.对于每个布尔变量,计算 $C[j,x]$ 的时间复杂性为

$O(km)$，$1\leq j\leq n$，$x\in\{0,1,k-1,k\}$. 算法在 While 循环中找到一个 u_j 满足 $|C[j,1]|+|C[j,k-1]|<|C[j,0]|+|C[j,k]|$ 的时间复杂性为 $O(kmn)$. 每循环一次，[1～k−1]-满足的项数目至少增加 1，所以算法时间复杂性为 $O(km^2n)$. 下面分析算法的近似性能比.

引理 10.　设算法 *flip-maxksat*(U,C) 步 1～4 的 While 循环结束时，得到布尔变量赋值 $a(U)$，则对于任意 u_j，$|C[j,1]|+|C[j,k-1]|\geq|C[j,0]|+|C[j,k]|$).

证明.　若存在 u_j，满足 $|C[j,1]|+|C[j,k-1]|<|C[j,0]|+|C[j,k]|$，由算法 While 循环条件可知，While 循环不会结束.　　证毕.

引理 11.　给定布尔变量赋值 $a(U)$，所有布尔变量赋值均取反得到 $\bar{a}(U)$，则 $S[\bar{a},1]\cup\cdots\cup S[\bar{a},k-1]=S[a,1]\cup\cdots\cup S[a,k-1]$，$NS[\bar{a}]\cup S[\bar{a},k]=NS[a]\cup S[a,k]$.

证明.　所有布尔变量赋值取反，j-满足的项变为$(k-j)$-满足，因 $1\leq j\leq k-1$，所以 $S[\bar{a},1]\cup\cdots\cup S[\bar{a},k-1]=S[a,1]\cup\cdots\cup S[a,k-1]$. 同理 $NS[\bar{a}]\cup S[\bar{a},k]=S[a,]\cup S[a,k]$.　　证毕.

推论 2.　设算法 *flip-maxksat*$(\cdot)$ 得到布尔变量赋值 $b(U)$，则 $|S[b,k]|\geq|NS[b]|$.

证明.　由算法步 5～7 和引理 11 可知，$|NS[b]|=\min\{|NS[a]|,|S[a,k]|\}$，$|S[b,k]|=\max\{|NS[a]|,|S[a,k]|\}$. 所以 $|S[b,k]|\geq|NS[b]|$.　　证毕.

引理 12.　设算法结束时得到布尔变量赋值 $b(U)$，则 $\sum_{j=1}^{k-1}|S[b,j]|\geq\sum_{j=1}^{n}|C[j,1]|+\sum_{j=1}^{n}|C[j,k-1]|$.

证明.　先证明算法步 1～4 的 While 循环结束时，得到的布尔变量赋值 $a(U)$ 满足 $\sum_{j=1}^{k-1}|S[a,j]|\geq\sum_{j=1}^{n}|C[j,1]|+\sum_{j=1}^{n}|C[j,k-1]|$.

设 $C_t=\{x[t,1],\cdots,x[t,k]\}$，$x[t,j]\in\{u_1,\cdots,u_n,\bar{u}_1,\cdots,\bar{u}_n\}$，$1\leq j\leq k$. 若 $C_t\in S[a,1]$，不失一般性，设 $a(x[t,1])=T$，$a(x[t,2])=\cdots=a(x[t,k])=F$. 当且仅当 $x[t,1]$ 赋值取反时，C_t 变为不满足，其它布尔变量赋值取反，C_t 仍为 1-满足或变为 2-满足. 设 $x[t,1]\in\{u_j,\bar{u}_j\}$，则 $C_t\in C[j,1]$，但对于 $x\neq j$，$C_t\notin C[x,1]$，即 C_t 出现在一个且仅一个 $C[j,1]$ 中，另外 $C[j,1]$ 中仅含有 $S[a,1]$ 中的项，所以 $|S[a,1]|=\sum_{j=1}^{n}|C[j,1]|$. 同理可证 $|S[a,k-1]|=\sum_{j=1}^{n}|C[j,k-1]|$. 所以 $\sum_{j=1}^{k-1}|S[a,j]|\geq\sum_{j=1}^{n}|C[j,1]|+\sum_{j=1}^{n}|C[j,k-1]|$.

由引理 11，$\sum_{j=1}^{k-1}|S[b,j]|=\sum_{j=1}^{k-1}|S[a,j]|$，所以引理得证.　　证毕.

引理 13.　设算法结束时得到布尔变量赋值 $b(U)$，则 $k|S[b,k]|+k|NS[b]|=\sum_{j=1}^{n}|C[j,0]|+\sum_{j=1}^{n}|C[j,k]|$.

证明.　先证明算法步 1～4 的 While 循环结束时，得到的布尔变量赋值 $a(U)$ 满足，$k|S[a,k]|+k|NS[a]|=\sum_{j=1}^{n}|C[j,k]|+\sum_{j=1}^{n}|C[j,0]|$.

设 $C_t=\{x[t,1],\cdots,x[t,k]\}\in NS[a]$，$a(x[t,1])=\cdots=a(x[t,k])=F$. C_t 中每个布尔变量字母赋值取反，C_t 会由不满足变为 1-满足，即 $a(C_t,\overline{x[t,1]})=\cdots=a(C_t,\overline{x[t,k]})=T$. 设 $x[t,y]\in\{u_{j[y]},\bar{u}_{j[y]}\}$，则 $C_t\in C[j[y],0]$，$1\leq y\leq k$. 而对于任意 $x\notin\{j[y]:1\leq y\leq k\}$，有 $a(C_t,u_x)=a(C_t,\bar{u}_x)=N$，所以 $C_t\notin C[x,0]$. 所以 $\sum_{j=1}^{n}|C[j,0]|=k|NS[a]|$. 同理可证 $\sum_{j=1}^{n}|C[j,k]|=k|S[a,k]|$. 所以，$\sum_{j=1}^{n}|C[j,0]|+\sum_{j=1}^{n}|C[j,k]|=k|NS[a]|+k|S[a,k]|$.

由引理 11 得，$|NS[a]|+|S[a,k]|=|NS[b]|+|S[b,k]|$，所以引理得证.　　证毕.

定理 5.　设算法结束时得到布尔变量赋值 $b(U)$，则 $\sum_{j=1}^{k}|S[b,j]|\geq\frac{2k+1}{2k+2}|C|$.

证明.　由引理 12、引理 10 和引理 13 得

$$\begin{aligned}\sum_{j=1}^{k-1}|S[b,j]|&\geq\sum_{j=1}^{n}|C[j,1]|+\sum_{j=1}^{n}|C[j,k-1|\\&\geq\sum_{j=1}^{n}|C[j,0]|+\sum_{j=1}^{n}|C[j,k]|\\&=k|NS[b]|+k|S[b,k]|\\&\geq 2k\min\{|NS[b]|,|S[b,k]|\}\end{aligned}\tag{11}$$

由推论 2，$|NS[b]|\leq|S[b,k]|$，所以

$$\begin{aligned}\sum_{j=1}^{k-1}|S[b,j]|&\geq 2k\min\{|NS[b]|,|S[b,k]|\}\\&=2k|NS[b]|\end{aligned}\tag{12}$$

且有$\sum_{j=1}^{k}|S[b,j]|=\sum_{j=1}^{k-1}|S[b,j]|+|S[b,k]|\geq(2k+1)|NS[b]|$，所以$\sum_{j=1}^{k}|S[b,j]|\geq\frac{2k+1}{2k+2}|C|$.

证毕.

由定理 5 可得，算法 *flip-maxksat*(•)的近似性能比为$\frac{2k+2}{2k+1}$.

6 两位跳变局部搜索算法

直观想法告诉我们，每次将多个布尔变量赋值取反的局部搜索方法，能够尝试更多可能的布尔变量赋值，因而可能获得更好的解. 现在讨论每次选择两个布尔变量赋值取反，解答 Max-(*k*)-Sat 问题的局部搜索方法. 将这种方法称为两位跳变的局部搜索算法. 下面叙述中总假设 $k\geq3$，每个项均含有 *k* 个布尔变量字母.

给定 Max-(*k*)-Sat 问题实例：*U*、*C* 和布尔变量赋值 $a(U)$，*C* 中的项分为 1-满足，…，*k*-满足和不满足，共 $k+1$ 类，$S[a,j]$ 为 *j*-满足的项集合，$1\leq j\leq k$，$NS[a]$ 为不满足的项集合. 另外假设 $C[i,j,x]$ 表示布尔变量 u_i 和 u_j 赋值取反，由 *x*-满足变为不满足或 *k*-满足的项集合；$C[i,j,0]$ 表示布尔变量 u_i 和 u_j 赋值取反，由不满足变为 *x*-满足的项集合；$C[i,j,k]$ 表示布尔变量 u_i 和 u_j 赋值取反，由 *k*-满足变为 *x*-满足的项集合，其中 $1\leq x\leq k-1$.

为方便，假设 $S[a,1\sim k-1]=\bigcup_{j=1}^{k-1}S[a,j]$；$C[i,j,1\sim k-1]=\bigcup_{x=1}^{k-1}C[i,j,x]$. 因对于 $i,j,i\neq j$，$S[a,i]\cap S[a,j]=\varnothing$，所以 $|S[a,1\sim k-1]|=\sum_{j=1}^{k-1}|S[a,j]|$，$|C[i,j,1\sim k-1]|=\sum_{x=1}^{k-1}|C[i,j,x]|$. 对于 $k=3$ 和 $k\geq4$ 两种情况，$S[a,1\sim k-1]$ 和 $C[i,j,1\sim k-1]$ 中的项无法用统一形式表示，因而算法性能也有所不同. 我们先给出两位跳变解答 Max-(*k*)-Sat 问题算法的统一描述形式，再分 $k=3$ 和 $k\geq4$ 两种情况分析算法的性能.

算法 6. *twoflip-maxksat*(*U*,*C*).

1. $a(u_k)\leftarrow Rand\{T,F\}$，$1\leq k\leq n$；
2. While (存在 u_j，使 $|C[i,j,1\sim k-1]|<|C[i,j,0]|+|C[i,j,k]|$) do
3. $a(u_i)\leftarrow\overline{a(u_i)}$；$a(u_j)\leftarrow\overline{a(u_j)}$；
4. End While
5. If ($|S[a,k]|\geq|NS[a]|$) then $b(U)\leftarrow a(U)$；
6. Else $b(U)\leftarrow\bar{a}(U)$；
7. End if
8. Return $b(U)$.

算法的 While 循环执行一次，需要针对两个布尔变量 u_i,u_j 分别计算项集合 $S[a,1]$，…，$S[a,k]$，$NS[a]$ 和 $C[i,j,0]$，…，$C[i,j,k]$，可在 $O(km)$ 时间内完成. 每执行一次变量赋值取反操作，$S[a,1\sim k-1]$ 中的项数目至少增加 1. 因 *C* 中只有 *m* 个项，所以算法 While 循环一定能够结束. 算法在 While 循环中找到两个布尔变量 u_i,u_j 满足 $|C[i,j,1\sim k-1]|<|C[i,j,0]|+|C[i,j,k]|$ 的时间复杂性为 $O(kmn^2)$. 每循环一次，$[1\sim k-1]$-满足的项数目至少增加 1，所以算法时间复杂性为 $O(km^2n^2)$.

下面所采用的符号与前面相同，总假设 $S[a,\bullet]$ 和 $NS[a]$ 表示算法 While 循环结束时，被赋值 $a(U)$ 满足的项集合和不被 $a(U)$ 满足的项集合；$S[b,\bullet]$ 和 $NS[b]$ 表示算法结束时，被赋值 $b(U)$ 满足的项集合和不被 $b(U)$ 满足的项集合. 另外我们下面只给出结论的简要证明.

引理 14. 布尔变量赋值 $a(U)$ 满足：$|C[i,j,1\sim k-1]|\geq|C[i,j,0]|+|C[i,j,k]|$，$i\neq j,1\leq i,j\leq n$.

证明. 若存在 u_i,u_j，满足 $|C[i,j,1\sim k-1]|<|C[i,j,0]|+|C[i,j,k]|$，由算法 While 循环条件可知，While 循环不会结束. 证毕.

(1) 解答 Max-(3)-Sat 问题的性能分析

引理 15. 布尔变量赋值 $b(U)$ 满足

$$(n-2)|S[b,1]|+(n-2)|S[b,2]|=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,1]|+|C[i,j,2]|).$$

证明. 每个 $S[a,1]$ 中的项在 $C[i,j,1\sim2]$ 中出现 $1+(n-3)=n-2$ 次；每个 $S[a,2]$ 中的项在 $C[i,j,1\sim2]$ 中出现 $1+(n-3)=n-2$ 次. 所以

$$(n-2)|S[a,1]|+(n-2)|S[a,2]|=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,1]|+|C[i,j,2]|).$$

由引理 11 可知，$|S[a,1]|+|S[a,2]|=|S[b,1]|+|S[b,2]|$. 所以引理结论为真. 证毕.

引理 16. 布尔变量赋值 $b(U)$ 满足

$$3(n-2)|NS[b]|+3(n-2)|S[b,3]|=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,0]|+|C[i,j,3]|).$$

证明. 每个 $NS[a]$ 中的项在 $C[i,j,0]$ 中出现

$3+3(n-3)=3(n-2)$次；每个 $S[a,3]$中的项在 $C[i,j,3]$中出现 $3+3(n-3)=3(n-2)$次. 所以

$$3(n-2)|NS[a]|+3(n-2)|S[a,3]|=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,0]|+|C[i,j,3]|).$$

由引理 11 可知，$|NS[a]|+|S[a,3]|=|NS[b]|+|S[b,3]|$，所以结论为真. 证毕.

定理 6. 算法结束时得到布尔变量赋值$b(U)$，则$|S[b]|\geq\frac{7}{8}|C|$.

证明. 由引理 15、引理 14 和引理 16 得

$$\begin{aligned}&(n-2)|S[b,1]|+(n-2)|S[b,2]|\\&=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,1]|+|C[i,j,2]|)\\&\geq\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,0]|+|C[i,j,3]|)\\&=3(n-2)|NS[b]|+3(n-2)|S[b,3]|\end{aligned}\tag{14}$$

又由算法 5～7 步的条件语句可知，$|NS[b]|\leq|S[b,3]|$，所以

$$\begin{aligned}|S[b,1]|+|S[b,2]|&\geq6\min\{|NS[b]|,S[b,3]|\}\\&=6|NS[b]|\end{aligned}\tag{15}$$

$$\begin{aligned}|S[b]|&=|S[b,1]|+|S[b,2]|+|S[b,3]|\\&\geq7|NS[b]|\end{aligned}\tag{16}$$

由$|C|=|S[b]|+|NS[b]|$，即得$|S[b]|\geq\frac{7}{8}|C|$. 证毕.

由定理 6 我们知道，选择两个布尔变量赋值取反跳变的局部搜索算法，在解答 Max-(3)-Sat 问题时，近似性能比仍为 8/7.

(2) 解答 Max-(k)-Sat($k\geq4$)问题的性能分析

引理 17. 布尔变量赋值 $a(U)$满足

$$(n-k)|S[a,1]|+|S[a,2]|+|S[a,k-2]|+(n-k)|S[a,k-1]|=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|C[i,j,1\sim k-1]|.$$

证明. 若 $k=4$，则

① 每个 $S[a,1]$中的项在 $C[i,j,1]$中出现 $n-4$ 次；

② 每个 $S[a,2]$中的项在 $C[i,j,2]$中出现 2 次；

③ 每个 $S[a,3]$中的项在 $C[i,j,3]$中出现 $n-4$ 次. 所以结论为真.

若 $k\geq5$，则

① 每个 $S[a,1]$中的项在$C[i,j,1]$中出现 $n-k$ 次；

② 每个 $S[a,2]$中的项在$C[i,j,2]$中出现 1 次；

③ 每个 $S[a,k-2]$中的项在 $C[i,j,k-2]$中出现 1 次；

④ 每个 $S[a,k-1]$中的项在 $C[i,j,k-1]$中出现 $n-k$ 次. 所以结论为真. 证毕.

引理 18. 布尔变量赋值 $a(U)$满足

$$(k(n-k)+C_k^2)|NS[a]|+(k(n-k)+C_k^2)|S[a,k]|=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,0]|+|C[i,j,k]|).$$

证明. 每个 $S[a,0]$中的项在 $C[i,j,0]$中出现 $k(n-k)+C_k^2$次；每个 $S[a,k]$中的项在 $C[i,j,k]$中出现 $k(n-k)+C_k^2$)次. 所以结论为真. 证毕.

当 $n\leq5$ 时，只须枚举 n 个布尔变量的 2^5种取值，可得到 Max-(4)-Sat 问题的精确解. 另外若 $n=k$，则只要 $m<2^k$，即可给出布尔变量赋值，使所有项都满足，而若 $m=2^k$，则任意布尔变量赋值均为 Max-(k)-Sat 问题的精确解.

因此，下面总假设 Max-(k)-Sat 问题实例满足：① $n\geq6$；② 若 $k\geq5$，则 $n\geq k+1$. 我们在下面的定理证明中，要求 Max-(k)-Sat 实例总满足这个关于布尔变量数目的假设.

定理 7. 布尔变量赋值 $b(U)$满足$|S[b]|\geq\left(1-\frac{1}{2k+2+k(k-1)/(n-k)}\right)|C|$.

证明. 因 $n-4\geq2$，所以算法解答 Max-(4)-Sat 实例时，可由引理 17 得到

$$\begin{aligned}&(n-4)(|S[a,1]|+|S[a,2]|+|S[a,3]|)\\&\geq(n-4)|S[a,1]|+2|S[a,2]|+(n-4)|S[a,3]|\\&=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|C[i,j,1\sim3]|\end{aligned}\tag{17}$$

又因若 $k\geq5$，则 $n\geq k+1$，所以算法解答 Max-(k)-Sat 问题实例时，由引理 17 得到

$$\begin{aligned}&(n-k)(|S[a,1]|+|S[a,2]|+|S[a,k-2]|+|S[a,k-1]|)\geq(n-k)|S[a,1]|+|S[a,2]|+|S[a,k-2]|+(n-k)|S[a,k-1]|\\&=\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|C[i,j,1\sim k-1]|\end{aligned}\tag{18}$$

因此对于 $k\geq4$，总有

$$(n-k)(|S[a,1]|+|S[a,2]|+|S[a,k-2]|+|S[a,k-1]|)\geq\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|C[i,j,1\sim k-1]|\tag{19}$$

由不等式(19)、引理 14 和引理 18 得

$$\begin{aligned}&(n-k)(|S[a,1\sim k-1]|)\geq(n-k)(|S[a,1]|+|S[a,2]|+|S[a,k-2]|+|S[a,k-1]|)\\&\geq\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}|C[i,j,1\sim k-1]|\end{aligned}$$

$$\geq\sum_{i=1}^{n-1}\sum_{j=i+1}^{n}(|C[i,j,0]|+|C[i,j,k]|)$$
$$=(k(n-k)+C_k^2)|NS[a]|+$$
$$(k(n-k)+C_k^2)|S[a,k]| \quad (20)$$

再由引理 11 可得到

$$(n-k)(|S[b,1\sim k-1]|)\geq$$
$$(k(n-k)+C_k^2)|NS[b]|+(k(n-k)+C_k^2)|S[b,k]| \quad (21)$$

即 $|S[b,1\sim k-1]|\geq\left(k+\frac{k(k-1)}{2(n-k)}\right)(|NS[b]|+|S[b,k]|)$. 因 $|NS[b]|\leq|S[b,k]|$,所以 $|S[b,1\sim k]|\geq\left(\frac{k(k-1)}{n-k}+2k+1\right)|NS[b]|$,所以 $|S[b]|\geq\left(1-\frac{1}{2k+2+k(k-1)/(n-k)}\right)|C|$. 证毕.

由定理 7 可知,算法 *twoflip-maxksat*(•)的近似性能比为 $1+\frac{1}{2k+1+k(k-1)/(n-k)}$.

参 考 文 献

[1] Cook S A. The complexity of theorem-proving procedures//Proceedings of the 3rd Annual ACM Symposium on Theorey of Computing, Shaker Heights. Ohio,USA, 1971: 151-158

[2] Papadimitirou C H, Yanakakis M. Optimization, approximation, and complexity classes. Journal of Computer and System Sciences, 1991, 43(3): 425-440

[3] Johnson D S. Approximation algorithms for combinatorial problems. Journal of Computer and System Sciences, 1974, 9(3): 256-278

[4] Chen J, Friesen D, Zheng H. Tight bound on Johnson's algorithm for maximum satisfiability. Journal of Computer and System Sciences, 1999, 58(3): 622-640

[5] Goemans M X, Williamson D P. New 3/4-approximation algorithms for the maximum satisfiability problem. SIAM Journal on Discrete Mathematics, 1994, 7(4): 656-666

[6] Motwani R, Raghavan P. Randomized Algorithms, Cambridge, UK: Cambridge University Press, 1995

[7] Goemans M X, Williamson D P. Improved approximation algorithms for maximum cut and satisfiability problems using semi-definite programming. Journal of the ACM, 1995, 42(6): 1115-1145

[8] Feige U, Goemans M X. Approximating the value of two power proof systems, with applications to Max-2Sat and Max-Dicut//Proceedings of the 3rd Israel Symposium on Theorey and Computing Systems. Tel Aviv, Israel, 1995: 182-189

[9] Asano T, Williamson D P. Improved approximation algorithms for Max-Sat. Journal of Algorithms, 2002, 42(1): 173-202

[10] Trevisan L, Sorkin G B, Sudan M, Williamson D P. Gadets, approximation, and linear programming//Proceedings of the 37th Annual IEEE Symposium on Foundations of Computer Science. Burlington, Vermont, 1996: 617-626

[11] Trevisan L. Approximating satisfiable satisfiability problems. Algorithmica, 2000, 28(1): 145-172

[12] Hastad J. Some optimal inapproximability results//Proceedings of the 28th Annual ACM Symposium on Theorey of Computing. El Paso, Texas, 1997: 1-10

[13] Karloff H, Zwick U. A 7/8-approximation algorithm for Max 3Sat? //Proceedings of the 38th IEEE Symposium on Foundations of Computer Science. Miami Beach, Florida, 1997: 406-415

[14] Halperin E, Zwick U. Approximation algorithms for MAX 4-SAT and rounding procedures for semidefinite programs. Journal of Algorithms, 2001, 40(2): 184-211

[15] Paturi R, Pudlak P, Saks M E, Zane F. An improved exponential-time algorithm for k-SAT. Journal of the ACM, 2005, 52(3): 337-364

[16] Alexander S K, Konstantin K. New bounds for MAX-SAT by clause learning//Proceedings of the 2nd International Computer Science Symposium. Ekaterinburg, Russia, 2007: 194-204

[17] Gu J. Efficient local search for very large-scale satisfiability problems. ACM SIGART Bulletin, 1992, 3(1): 8-12

[18] Selman B, Levesque H, Mitchell D. A new method for solving hard satisfiability problems//Proceedings of the 10th National Conference on Artificial Intelligence. Pasadena, Calefornia, USA, 1992: 440-446

[19] Papadimitriou C H. On selecting a satisfying truth assignment//Proceedings of the 32nd Annual IEEE Symposium on Foundations of Computer Science. San Juan, Puerto Rico, USA, 1991: 163-169

[20] Koutsoupias E, Papadimitriou C H. On the greedy algorithm for satisfiability. Information Processing Letters, 1992, 43(1): 53-55

[21] Hoos H H. On the run-time behavior of stochastic local search algorithms for SAT//Proceedings of the 16th National Conference on Artificial Intelligence. Orlando, Florida, 1999: 661-666

[22] Hirsch E A, Kojevnikov A. UnitWalk: A new SAT solver that uses local search guided by unit clause elimination. Annals of Mathematics and Artificial Intelligence, 2005, 43(1): 91-111

[23] Williams R R. Algorithms and resource requirements for fundamental problems[Ph. D. dissertation]. Carnegie Mellon University, Pittsburgh, USA, 2007

[24] Kautz H, Selman B. The state of sat. Discrete Applied Mathematics, 2007, 155(12): 1514-1524

[25] Chris C, Russell I, Valentine K, Ramamohan P. The complexity of unique k-SAT: An isolation lemma for k-CNFs. Journal of Computer and System Sciences, 2008, 74(3): 386-393

ZHU Da-Ming, born in 1963, Ph.D., professor, Ph. D. supervisor. His research interests include algorithm design and analysis, computational molecular biology.

MA Shao-Han, born in 1938, professor, Ph. D. supervisor. His research interests include algorithm design and analysis, artificial intelligence.

ZHANG Ping-Ping, born in 1981, M. S., engineer. His research interests include algorithm design and analysis, software engineering.

Background

Maximum satisfiablity problem, Max-Sat briefly, is the central problem in theoretical computer science. Since Max-Sat is NP-Hard and Max-SNP-Hard, the research activities on this problem mainly focus on the design of approximation algorithms. Max-k-Sat and Max-(k)-Sat are the most important sub problems of Max-Sat. The Johnson's algorithm can achieve the performance ratio $2^k/(2^k-1)$ to solve Max-(k)-Sat. Thus it can achieve the performance ratio 8/7 to solve Max-(3)-Sat. Karloff and Zwick design a random algorithm to solve Max-3-Sat. The expected performance ratio of their algorithm is also 8/7. Moreover, Hastad shows that either of Max-3-Sat and Max-(3)-Sat cannot be approximated within ratio 8/7, if P$\neq$NP. On the other hand, many local search algorithms are designed to solve the Max-Sat and its sub-problems. Those local search algorithms are testified to be useful in the practice of exact solution computation for Max-Sat. It seems difficult to analyze the performance of the local search algorithms. We cannot see any related results to analyze the performance of the local search algorithms to solve Max-Sat.

This paper approaches the local search method to solve the Max-Sat and analyzes the performance of the method. The central issue of the paper is to discuss the local search algorithm to solve Max-(3)-Sat. The authors show that the one-bit-flip local search algorithm can achieve the performance ratio 4/3 for solving Max-(3)-Sat; and the algorithm of one-bit-flip local search following the conditionally all-bits-flip can achieve the performance ratio 5/4 for solving Max-(3)-Sat. The authors then design a new local search algorithm to solve Max-(3)-Sat using the one-bit-flip local search following the all-bits-flip, and show the new algorithm have the performance ratio 8/7. The 8/7-approximation algorithm can be extended to solve the Max-(k)-Sat problem with performance ratio $(2k+2)/(2k+1)$ for $k\geq 4$.

Our algorithm has the same performance ratio 8/7 for solving Max-(3)-Sat as Johnson's algorithm. Johnson's algorithm can only output the same solution for each time of running. Our algorithm can improve the performance of the solution by repeatedly running. In fact, the algorithm can produce the exact solution with probability 1, if it runs for infinite times. Karloff and Zwick's algorithm for Max-3-Sat can also achieve expected performance ratio 8/7. The time complexity of de-randomization of the algorithm is too high. Thus their algorithm seems not very useful in practice. The results of the paper demonstrate the inevitability for local search to have high performances in solving the maximum satisifiablity problems.

This paper is supported by National Natural Science Foundation of China (60573024) and Ph. D. Station Foundation of Chinese Education Department (200901311100009). The research project aims at discussing some combinatorial problems about comparing genomes, so that new algorithms can be designed. Discussing the algorithm and complexity of Max-Sat and its sub problems is one part of contents for the project. In the past, we design new algorithms for the translocation sorting and transposition sorting problems, and prove the complexity of unsigned translocation sorting and fingerprint clustering problems.