最大不全 k 满足问题的局部搜索近似算法

咸爱勇 朱大铭

(山东大学计算机科学技术学院 济南 250101)

摘 要 合取范式可满足与最大可满足问题是理论计算机科学的核心问题.最大不全满足问题是最大可满足问题的一般化.限制每个子句均含有 $k(\geq 2)$ 个字母的最大不全满足问题又称为最大不全 k 满足问题.最大不全满足问题的算法进展,以解答该类问题的半定规划松弛法最具代表性.关于最大不全 2 满足、3 满足和 4 满足问题,目前性能最好的近似算法分别由 Goemans 与 Williamson、Zwick、Karloff 与 Zwick 给出,近似性能比分别为 1.139(1/0.878)、1.10047(1/0.9087)和 8/7.当 $k\geq 5$ 时,最大不全 k 满足问题的近似算法则未曾见到.文中给出了一个解答最大不全 k 满足问题的局部搜索算法,近似性能比可达到 $2^{k-1}/(2^{k-1}-1)$,$k\geq 2$;进一步将该方法推广到解答由不少于 k 个字母的子句构成的最大不全 k 满足问题,近似性能比亦可达到 $2^{k-1}/(2^{k-1}-1)$.利用解答最大不全 k 满足问题的近似算法,给出了解答最大 k 可满足问题的新近似算法,近似性能比可达到 $2^k/(2^k-1)$.文中最后证明了若 P≠NP,则 $k\geq 4$ 的最大不全 k 满足问题不能近似到小于 $2^{k-1}/(2^{k-1}-1)$,从而说明文中解答最大不全 k 满足问题的算法近似性能比是最优的.

关键词 局部搜索;算法;近似性能比;合取范式;可满足性

中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2015.01561

The Local Search Approximation Algorithms for Maximum Not-All-Equal k-Satisfiability Problems

XIAN Ai-Yong ZHU Da-Ming

(*School of Computer Science and Technology, Shandong University, Jinan* 250101)

Abstract The satisfiability problem as well as the maximum satisfiability problem of conjunctive norm forms are the central problems in theoretical computer science. The maximum not-all-equal satisfiability problem is a generalization of the maximum satisfiability problem. The maximum not-all-equal satisfiability problem is named maximum not-all-equal k satisfiability problem, if each clause of its instance contains $k(\geq 2)$ literals. Semi-definite programming relaxation is the frequently-used also the most typical method for solving the maximum not-all-equal satisfiability problems. To our knowledge at present, the best algorithms for the maximum not-all-equal 2,3 and 4 satisfiability problems come from the semi-definite programming relaxation methods given by Goemans and Williamson, Zwick, Karloff and Zwick, with performances 1.139(1/0.878), 1.10047(1/0.9087) and 8/7 respectively. When $k\geq 5$, we have not seen any approximation algorithm for the maximum not-all-equal k satisfiability problems. In this paper, we propose a local search algorithm to solve the maximum not-all-equal k satisfiability problem. This algorithm can achieve the performance ratio $2^{k-1}/(2^{k-1}-1)$ for $k\geq 2$. We extend the method to propose a local search algorithm to solve the more generalized version of the maximum not-all-equal k satisfiability problem, with each clause containing at least k literals. This algorithm can still achieve the

收稿日期:2013-12-05;最终修改稿收到日期:2015-01-07.本课题得到国家自然科学基金(61070019)、教育部博士点基金(20090131110009)和山东省自然科学基金(ZR2012FZ002)资助.**咸爱勇**,男,1986 年生,硕士研究生,主要研究方向为算法设计与分析、软件工程.E-mail:xianaiyong@msn.com.**朱大铭**,男,博士,教授,研究领域为算法设计与分析、生物信息学.

performance ratio $2^{k-1}/(2^{k-1}-1)$. Using the method for the generalized version of the maximum not-all-equal k satisfiability problem, we propose a new $2^k/(2^k-1)$-approximation algorithm for generalized version of the maximum k satisfiability problem. Finally, we prove that if P≠NP, then the maximum not-all-equal k satisfiability problem cannot be approximated within $2^{k-1}/(2^{k-1}-1)$ for $k \geq 4$. This implies the performance ratio of our algorithm for the maximum not-all-equal k satisfiability problem is optimal.

Keywords local search; algorithm; performance ratio; conjunctive normal form; satisfiability

1 引 言

合取范式可满足问题(简称 SAT)是理论计算机科学的核心问题.合取范式最大可满足问题(简称 Max-SAT)是 SAT 问题的优化形式,要求根据给定布尔变量字母组成的子句集合,计算布尔变量赋值,使满足的子句(Clause)数目达到最大.早在 1971 年,Max-SAT 就被证明为 NP-Hard[1-2].最大不全满足问题是最大可满足问题的一般化[3],要求根据给定布尔变量字母组成的子句集合,计算布尔变量赋值,使满足且不全满足的子句数目达到最大.最大不全满足问题简称为 Max NAE-SAT,当然也是 NP-Hard 问题[4].Max NAE-SAT 也可看作最大割问题及最大集合分割问题的一般化[3,5].

若每个子句含有固定数目的布尔变量字母,则 Max-SAT 和 Max NAE-SAT 演变为它们的一类重要子问题,这类子问题的算法与复杂性研究同样吸引了许多学者的研究兴趣.本文将每个子句含有 k 个布尔变量字母的 Max-SAT 子问题称为 Max-k-SAT;将每个子句含有不多于 k 个布尔变量字母的 Max-SAT 子问题称为 Max-[k]-SAT;将每个子句含有不少于 k 个布尔变量字母的 Max-SAT 子问题称为 Max-(k)-SAT.类似地,将每个子句含有 k 个布尔变量字母的 Max NAE-SAT 子问题称为 Max NAE-k-SAT;将每个子句含有不多于 k 个布尔变量字母的 Max NAE-SAT 子问题称为 Max NAE-[k]-SAT;将每个子句含有不少于 k 个布尔变量字母的 Max NAE-SAT 子问题称为 Max NAE-(k)-SAT.

Johnson[6]于 1974 年最先设计出近似性能比为 2 的 Max-SAT 问题近似算法,并设计出近似性能比为 $2^k/(2^k-1)$的 Max-(k)-SAT 问题近似算法.为每个布尔变量以 1/2 概率随机赋值 T/F 的简单方法,也可使解答 Max-(k)-SAT 的平均近似性能比达到 $2^k/(2^k-1)$[7].Yannakakis[8]于 1994 年给出解答 Max-SAT 问题的随机算法,其平均近似性能比为 4/3. Goemans 与 Williamson[9]利用线性规划松弛法,给出一个更简单的随机近似算法,近似性能比仍为 4/3, 1995 年,他们采用半定规划松弛法将解答 Max-[2]-SAT 问题的近似性能比改进到 1.139[10],该方法修改后用于解答 Max-SAT 问题,近似性能比为 1.32[9].Asano 与 Williamson 重新分析了 Goemans 和 Williamson 的 4/3 近似算法,利用 Feige 和 Goemans[11]给出的解答 Max-[2]-SAT 与 Karloff 和 Zwick[12]给出的解答 Max-[3]-SAT 的半定规划松弛法,将解答 Max-SAT 问题的近似性能比改进为 1.275[13].

最大不全满足问题的半定规划松弛法,代表了该问题算法研究的主要进展.解答 Max NAE-2-SAT 的半定规划松弛法可由 Goemans 和 Williamson[10]解答 Max-[2]-SAT 的半定规划松弛法直接得到,近似性能比为 1.139;解答 Max NAE-4-SAT 的半定规划松弛法则可由 Karloff 与 Zwick[12]解答 Max-[3]-SAT 的半定规划松弛法直接得到,近似性能比为 8/7. Andersson 和 Engebretsen 于 1998 年给出解答 Max NAE-SAT 的半定规划松弛法,近似性能比为 1.3812(1/0.7240)[14]. Han、Ye 与 Zhang[5]利用被称作外部旋转(Outward Rotation)的取整技术将 Andersson 和 Engebretsen 的算法的近似性能比加强为 1.3335(1/0.7499). Zwick[15]于 1999 年进一步利用半定规划松弛法和外部旋转取整技术给出Max NAE-SAT 问题猜测近似性能比为1.2536(1/0.7977)的近似算法,并给出 Max NAE-3-SAT 近似性能比为 1.10047(1/0.9087)的近似算法. 2006 年,Abidor 和 Berkovitch 等人将解答 Max NAE-SAT 的猜测近似性能比改进为 1.2079(1/0.8279),并将解答 Max-SAT 的(无猜测)近似性能比改进为 1.25502[3](1/0.7968).所谓猜测近似性能比,是指在文献[15]中给出的一个猜测概率表达式成立时,算法所能够达到的近似性能比.当 $k \geq 5$ 时,目前未见任何解答

Max NAE-k-SAT 的半定规划松弛法.

虽然半定规划松弛法解答最大可满足与最大不全满足问题获得了求解性能的全面提高，求解 Max-[3]-SAT 的近似性能比甚至已经达到极限值. 然而半定规划松弛法所耗费的运行时间尚不令人满意. 因为半定规划松弛法所给出的近似性能比还须经历去随机过程才能准确地达到，目前去随机的时间复杂性仍是一个指数很大的多项式函数，一般不低于 $O(n^{30})$[16].

局部搜索是解答 NP-Hard 优化问题的典型方法，解答 SAT 问题的局部搜索算法也不少见. 我们所知的 SAT 问题局部搜索算法有 GSAT[17-18]、WSAT[19]、NSAT[20]、TSAT[21-22]、SDF[23] 等. 目前，局部搜索仍然是人们在实际 SAT 求解中最常使用的方法[24]. 目前尚无研究结果用于分析这些在实践中表现优越的局部搜索算法的性能. 2010 年，朱大铭等人[25]给出 Max-3-SAT 问题的局部搜索近似算法，近似性能比可达到 8/7，算法可推广到解答 Max-(k)-SAT，近似性能比为 $\frac{2k+2}{2k+1}$. 该算法用于解答 Max NAE-(k)-SAT[26]，近似性能比可达到 $\frac{k+1}{k}$. 另外，Hastad[27] 曾证明：若 P$\neq$NP，则 $k\geq 3$ 的 Max-k-SAT 不能多项式时间近似到小于 $2^k/(2^k-1)$，$k\geq 3$. 文献[25]的算法用于解答 $k\geq 4$ 的 Max-(k)-SAT，达到的近似性能比与 Hastad 给出的解答该问题拒绝达到的近似性能比上界仍有较大差距.

1994 年，Khanna 等人[28] 提出所谓盲目的(anonymous)局部搜索算法，用于解答 Max-k-SAT 问题，其近似性能比为 $2^k/(2^k-1)$. 该算法解答 Max-k-SAT，近似性能比已经达到 Hastad 关于解答 Max-k-SAT 拒绝达到的近似性能比上界. 然而该方法并不能简单地推广到解答 Max-(k)-SAT. 2000 年，Dantsin 等人[29]给出一个解答 k-SAT 的局部搜索精确算法，时间复杂性为 $(2-2/(k+1))^n$.

本文讨论 Max NAE-SAT 问题的局部搜索求解方法. 首先给出解答 Max NAE-k-SAT 问题的一个局部搜索算法，算法近似性能比可达到 $\frac{2^{k-1}}{2^{k-1}-1}$；再将解答 Max NAE-$k$-SAT 的局部搜索方法，推广到解答 Max NAE-(k)-SAT 问题，近似性能比也可达到 $\frac{2^{k-1}}{2^{k-1}-1}$；然后利用解答 Max NAE-($k$)-SAT 的算法，给出解答 Max-($k$)-SAT 的一个新算法，近似性能比达到 $\frac{2^k}{2^k-1}$. Khanna 等人的局部搜索算法解答 Max-k-SAT 也可以达到同样近似性能比，但不能解答 Max-(k)-SAT 问题实例. 本文算法则可直接用于解答 Max-(k)-SAT 问题实例. 本文最后证明当 $k\geq 4$ 时，Max NAE-k-SAT 不能多项式时间近似到小于 $\frac{2^{k-1}}{2^{k-1}-1}$.

2　最大不全 k 满足问题的局部搜索

一个布尔变量 u 对应两个字母 u 和 $\bar{u}$，分别称为 u 的正变量字母和反变量字母. 给予布尔变量 u 赋值后，其字母也相应得到确定的布尔值，本文我们并不区分布尔变量赋值与布尔变量字母赋值的不同.

一个子句是若干布尔变量字母的集合. 设 $C_t=\{x[1],x[2],\cdots,x[k]\}$ 为一个子句，其中 $x[1],\cdots,x[k]$ 为布尔变量字母，若赋值函数 $a(\cdot)$ 使得 $a(x[1])\vee\cdots\vee a(x[k])=T$，则称 C_t 是满足的；若 $a(x[1])\vee\cdots\vee a(x[k])=T$ 且 $a(x[1])\wedge\cdots\wedge a(x[k])=T$，则称 C_t 是全满足的；若 $a(x[1])\vee\cdots\vee a(x[k])=T$，且 $a(x[1])\wedge\cdots\wedge a(x[k])=F$，则称 C_t 是不全满足的；若 $a(x[1])\vee\cdots\vee a(x[k])=F$，则 C_t 为不满足的. 最大不全 k 满足问题由布尔变量集与 k 个字母的子句集给定，欲寻求每个布尔变量的真值指派，使给定子句集中不全满足的子句数目达到最大. 布尔变量的真值指派也称为布尔变量的赋值(函数). 最大不全 k 满足问题可形式化为：

实例：布尔变量集合 $U=\{u_1,\cdots,u_n\}$，子句集合 $C=\{C_1,\cdots,C_m\}$，每个子句均含有 k 个布尔变量字母. 子句 C_t 可表示为 $C_t=\{x[t,1],x[t,2],\cdots,x[t,k]\}$，其中，$1\leq t\leq m$，$x[t,l]\in\{u_1,\cdots,u_n,\bar{u}_1,\cdots,\bar{u}_n\}$，$1\leq l\leq k$.

目标：求 U 中布尔变量的赋值函数 $a:U\rightarrow\{T,F\}$，最大化 C 中不全满足子句的数目.

该问题即为 Max NAE-k-SAT. 若求解目标修改为寻求 U 的赋值函数，使 C 中满足的子句数目达到最大，则问题变为 Max-k-SAT.

Max NAE-k-SAT 是 Max-k-SAT 的一般化，或可将 Max-k-SAT 看作 Max NAE-k-SAT 的子问题，这是因为可将任一 Max-k-SAT 实例变换为一个 Max NAE-k-SAT 实例，并使两个实例有对应等价的赋值函数. 变换方法如下：

设$\langle U,C\rangle$是任一 Max-k-SAT 实例，增加一个布尔变量 o，得到 $U'=U\cup\{o\}$；在每个 C 的子句中添加布尔变量字母 o，得到 C'. 即设 $C_t=\{x[t,1],x[t,2],\cdots,x[t,k]\}\in C$，则$C'_t=\{x[t,1],x[t,2],\cdots,x[t,k],o\}$，$C'=\{C'_t|C_t\in C\}$. 将$\langle U',C'\rangle$视为一个 Max NAE-$k$-SAT 实例，设 $a(U')$为 U'的赋值函数，若 $a(x[t,1]),a(x[t,2]),\cdots,a(x[t,k]),a(o)$使得 C'_t 不全满足，当且仅当 $a(x[t,1])\oplus a(o)$，$a(x[t,2])\oplus a(o),\cdots,a(x[t,k])\oplus a(o)$使 C_t满足，其中“$\oplus$”表示异或运算. 因此 $a(U')$使得 C' 中不全满足的子句数目为 M，当且仅当 $a(u_1)\oplus a(o)$，$a(u_2)\oplus a(o),\cdots,a(u_n)\oplus a(o)$使 C 中满足的子句数目也为 M.

因此任一 Max-k-SAT 实例等价于一个每个子句均含有一个公共字母(例如 o)的 Max NAE-$(k+1)$-SAT 实例.

本节，每个子句均含有 k 个布尔变量字母. 因一个子句若同时含有一个布尔变量的正变量字母和反变量字母，则该子句总是不全满足的，也是满足的，所以本文总假设每个子句不同时含有一个布尔变量的两个字母.

2.1 算法及其性能

定义 1. 给定 U 的赋值函数 $a(U)$，若子句 C_t 含有布尔变量字母 u_j，且 $a(u_j)=T$，或 C_t含有布尔变量字母 $\bar{u}_j$且 $a(u_j)=F$，称 C_t被 $a(u_j)$满足或被 u_j满足；若子句 C_t含有布尔变量字母 u_j且 $a(u_j)=F$，或 C_t含有布尔变量字母 $\bar{u}_j$且 $a(u_j)=T$，称 C_t不被 $a(u_j)$满足或不被 u_j满足. 将 C_t被 $a(u_j)$满足简记为 $a(C_t,u_j)=T$，C_t不被$a(u_j)$满足简记为$a(C_t,u_j)=F$.

定义 2. 给定 U 的赋值函数 $a(U)$，$C_t\in C$. 若 C_t中有 i 个布尔变量字母赋值为 T，$0\le i\le k$，另外 $k-i$ 个字母赋值为 F，则称C_t关于$a(U)$是 i-满足的.

一个子句是 0-满足的，则为不满足的. 一个子句是 k-满足的，则为全满足的. 由 $a(U)$为 U 中布尔变量赋值后，设 S_i表示 C 中所有 i-满足的子句集合；再设 $C[i,j]$表示 i-满足，且被 u_j满足的子句集合；$N[i,j]$表示 i-满足，且不被 u_j满足的子句集合，即 $C[i,j]=\{C_t|C_t\in S_i,a(C_t,u_j)=T\}$，$N[i,j]=\{C_t|C_t\in S_i,a(C_t,u_j)=F\}$.

因每个子句不同时含有 u_j和 $\bar{u}_j$，所以若将 u_j的赋值取反，则$C[i,j]$中的子句均由 i-满足变为$(i-1)$-满足，$0<i\le k$；而 $N[i,j]$中的子句均由 i-满足变为 $(i+1)$-满足，$0\le i<k$.

任给 U 的赋值函数 $a(U)$，则 C 中不全满足的子句数目为$|S_1|+\cdots+|S_{k-1}|$. 解答 Max NAE-k-SAT 的局部搜索方法需要首先确定一个目标函数，然后利用如下方法，获得布尔变量的赋值：随机为布尔变量赋初值，然后选择一个布尔变量，将其赋值取反，使目标函数值增大，重复该操作直到不能选择布尔变量，其赋值取反可使目标函数值增大为止. 将该方法称为一位跳变局部搜索. 若直接采用$|S_1|+\cdots+|S_{k-1}|$做为目标函数，我们并不知道算法能否达到所期望的性能. Khanna 等人[28] 关于 Max-k-SAT 的局部搜索算法以$|S_1|,\cdots,|S_k|$的加权和作为目标函数. 我们同样建立一个由$|S_0|,|S_1|,\cdots,|S_k|$的加权和构成的目标函数，一般地表示为

$$F=\alpha_0|S_0|+\alpha_1|S_1|+\cdots+\alpha_{k-1}|S_{k-1}|+\alpha_k|S_k| \quad (1)$$

我们期望利用一位跳变的局部搜索使目标函数 F 最大化，从而使 F 达到局部最大时，$|S_1|+\cdots+|S_{k-1}|$得到不小于$(2^{k-1}-1)(|S_0|+|S_k|)$的取值. 在给出系数 $\alpha_1,\cdots,\alpha_{k-1}$的取值之前，先讨论选择赋值取反的布尔变量应满足的条件.

任意选择u_j，将其赋值取反，由i-满足变为$(i-1)$-满足的子句使 F 获得增量：$(\alpha_{i-1}-\alpha_i)|C[i,j]|$，$1\le i\le k$；由 i-满足变为$(i+1)$-满足的子句使 F 获得增量：$(\alpha_{i+1}-\alpha_i)|N[i,j]|$，$0\le i\le k-1$. 所以 F 因 u_j赋值取反而产生的增量为

$$\Delta F=\sum_{i=1}^{k}(\alpha_{i-1}-\alpha_i)|C[i,j]|+\sum_{i=0}^{k-1}(\alpha_{i+1}-\alpha_i)|N[i,j]| \quad (2)$$

如果存在一个布尔变量，其赋值取反使得 $\Delta F>0$，则将该变量的赋值取反就可增大 F 的取值. 下面给出求解 Max NAE-k-SAT 的局部搜索算法，并将算法命名为算法 1：Max NAE-k-SAT(U,C). 在算法描述中，$\overline{a(u_j)}$表示将布尔变量 u_j由赋值函数 $a(\cdot)$给予的布尔值取反得到的布尔值.

算法 1. Max NAE-k-SAT(U,C).

1. 随机为 U 中布尔变量赋值，得到 $a(U)$；
2. While(存在 u_j，使$\sum_{i=1}^{k}(\alpha_{i-1}-\alpha_i)|C[i,j]|+\sum_{i=0}^{k-1}(\alpha_{i+1}-\alpha_i)|N[i,j]|>0$) do
3. $a(u_j)\leftarrow\overline{a(u_j)}$；
4. End while
5. Return $a(U)$

现在需要给出目标函数 F 中$|S_i|$前的系数 α_i的取值. 在函数 F 中，S_0，S_k中的子句是不满足和全满足的，所以取 $\alpha_0=\alpha_k=0$. 其他系数 $\alpha_i(1\le i\le k)$由下

述递推关系给出，其中 $k\geq 2$.

$$\alpha_i=\begin{cases}0, & i=0\\ \alpha_{i-1}+\dfrac{2^{k-1}-1-\sum\limits_{j=1}^{i-1}\binom{k}{j}}{(k-i+1)\binom{k}{i-1}}, & 1\leq i\leq k\end{cases}\tag{3}$$

先证明由式(3)给出的系数 α_i 满足对称性质. 从而说明由式(3)计算得到的 $\alpha_k=0$.

性质 1.　若 $0\leq i\leq k$，则 $\alpha_i=\alpha_{k-i}$.

证明.　由式(3)，可以知道

$$\alpha_i-\alpha_{i-1}=\frac{2^{k-1}-1-\sum\limits_{j=1}^{i-1}\binom{k}{j}}{(k-i+1)\binom{k}{i-1}}\tag{4}$$

另外，

$$\begin{aligned}\alpha_{k-i}-\alpha_{k-(i-1)}&=-\frac{2^{k-1}-1-\sum\limits_{j=1}^{k-i+1-1}\binom{k}{j}}{(k-(k-i+1)+1)\binom{k}{k-i+1-1}}\\&=-\frac{2^{k-1}-1-\sum\limits_{j=1}^{k-i}\binom{k}{j}}{i\binom{k}{k-i}}\\&=-\frac{2^{k-1}-1-\sum\limits_{j=1}^{k-i}\binom{k}{j}}{(k-i+1)\binom{k}{i-1}}\\&=\frac{2^{k-1}-1-\sum\limits_{j=1}^{i-1}\binom{k}{j}}{(k-i+1)\binom{k}{i-1}}\end{aligned}\tag{5}$$

所以，

$$\alpha_i-\alpha_{i-1}=\alpha_{k-i}-\alpha_{k-(i-1)}\tag{6}$$

当 k 为偶数时，设 $\gamma=\left\lceil\frac{k}{2}\right\rceil=\frac{k}{2}$，由式(6)可以得到 $\alpha_\gamma-\alpha_{\gamma-1}=\alpha_\gamma-\alpha_{\gamma+1}$，所以 $\alpha_{\gamma-1}=\alpha_{\gamma+1}$. 依此类推，由 $\alpha_{\gamma-j}=\alpha_{\gamma+j}$ 和式(6)可以得到 $\alpha_{\gamma-j-1}=\alpha_{\gamma+j+1}$，$1\leq j\leq\frac{k}{2}-1$.

当 k 为奇数时，设 $\gamma=\left\lceil\frac{k}{2}\right\rceil=\frac{k+1}{2}$. 那么

$$\begin{aligned}\alpha_\gamma-\alpha_{\gamma-1}&=\frac{2^{k-1}-1-\sum\limits_{j=1}^{\gamma-1}\binom{k}{j}}{(k-\gamma+1)\binom{k}{\gamma-1}}\\&=\frac{2^{k-1}-1-\left(\binom{1}{k}+\cdots+\binom{\gamma-1}{k}\right)}{(k-\gamma+1)\binom{k}{\gamma-1}}\\&=\frac{2^{k-1}-1-\left(2^k-\binom{0}{k}-2^{k-1}\right)}{(k-\gamma+1)\binom{k}{\gamma-1}}=0\end{aligned}\tag{7}$$

所以 $\alpha_\gamma=\alpha_{\gamma-1}$. 由此利用式(6)可依次推得 $\alpha_{\gamma-j-1}=\alpha_{\gamma+j}$，$0\leq j\leq\frac{k-1}{2}$.

综上，对于满足 $0\leq i\leq k$ 的 i，总有 $\alpha_i=\alpha_{k-i}$.

引理 1.　任给 U 的赋值函数 $a(U)$，则有

$$\sum_{j=1}^{n}|C[i,j]|=i|S_i|,\sum_{j=1}^{n}|N[i,j]|=(k-i)|S_i|.$$

证明.　设 $C_t=\{x[t,1],x[t,2],\cdots,x[t,k]\}\in S_i$. 不失一般性假设 $a(x[t,1])=a(x[t,2])=a(x[t,i])=T$.

将 $\{x[t,1],\cdots,x[t,i]\}$ 中某一个字母的赋值取反，C_t 会由 i-满足变为 $(i-1)$-满足，设 $x[t,y]\in\{u_{j_y},\bar{u}_{j_y}\}$，$1\leq y\leq i$，则 $C_t\in C[i,j_y]$. 但对于 $x\notin\{j_y|1\leq y\leq i\}$，$C_t\notin C[i,x]$，即 C_t 出现在 i 个且恰好 i 个集合 $C[i,j]$ 中，所以 $\sum\limits_{j=1}^{n}|C[i,j]|=i|S_i|$. 而当 $\{x[t,i+1],\cdots,x[t,k]\}$ 中某个布尔变量字母赋值取反时，C_t 会由 i-满足变为 $(i+1)$-满足，设 $x[t,y]\in\{u_{j_y},\bar{u}_{j_y}\}$，$i+1\leq y\leq k$，则 $C_t\in N[i,j_y]$. 但对于任意 $x\notin\{j_y|i+1\leq y\leq k\}$，$C_t\notin N[i,x]$，所以 C_t 恰好出现在 $k-i$ 个 $N[i,j]$ 中，即 $\sum\limits_{j=1}^{n}|N[i,j]|=(k-i)|S_i|$.

证毕.

引理 2.　设算法 1 结束时得到 U 的赋值函数 $a(U)$，S_i 为由 $a(U)$ 为 U 中布尔变量赋值所产生的 i-满足的子句集合，则 $|S_1|+\cdots+|S_{k-1}|\geq(2^{k-1}-1)(|S_0|+|S_k|)$.

证明.　当算法结束时，任意布尔变量 u_j 赋值取反，$1\leq j\leq n$，目标函数 F 数值均不再增加. 那么对于每个 $j=1,2,\cdots,n$，总有

$$\sum_{i=1}^{k}(\alpha_{i-1}-\alpha_i)|C[i,j]|+\sum_{i=0}^{k-1}(\alpha_{i+1}-\alpha_i)|N[i,j]|\leq 0\tag{8}$$

将式(8)的 n 个不等式左右分别相加，得到

$$\begin{aligned}&\sum_{j=1}^{n}\sum_{i=1}^{k}(\alpha_{i-1}-\alpha_i)|C[i,j]|+\\&\sum_{j=1}^{n}\sum_{i=0}^{k-1}(\alpha_{i+1}-\alpha_i)|N[i,j]|=\\&\sum_{i=1}^{k}\sum_{j=1}^{n}(\alpha_{i-1}-\alpha_i)|C[i,j]|+\\&\sum_{i=0}^{k-1}\sum_{j=1}^{n}(\alpha_{i+1}-\alpha_i)|N[i,j]|\leq 0\end{aligned}\tag{9}$$

由引理1,将$\sum_{j=1}^{n}|C[i,j]|=i|S_i|$和$\sum_{j=1}^{n}|N[i,j]|=(k-i)S_i$代入式(9),得到

$$k\alpha_1|S_0|+\sum_{i=1}^{k-1}[(\alpha_{i-1}-\alpha_i)i+(\alpha_{i+1}-\alpha_i)(k-i)]|S_i|+k\alpha_{k-1}|S_k|\leq 0 \tag{10}$$

由$\alpha_i-\alpha_{i-1}=\dfrac{2^{k-1}-1-\sum_{j=1}^{i-1}\binom{k}{j}}{(k-i+1)\binom{k}{i-1}}$,$\alpha_0=0$得$\alpha_1=\dfrac{2^{k-1}-1}{k}$. 由性质1得到$\alpha_{k-1}=\dfrac{2^{k-1}-1}{k}$,$\alpha_k=0$. 所以,

$$\begin{aligned}k\alpha_1|S_0|+k\alpha_{k-1}|S_k|&=k\alpha_1(|S_0|+|S_k|)\\&=(2^{k-1}-1)(|S_0|+|S_k|)\end{aligned} \tag{11}$$

于是不等式(10)变为

$$-\sum_{i=1}^{k-1}[(\alpha_{i-1}-\alpha_i)i+(\alpha_{i+1}-\alpha_i)(k-i)]|S_i|\geq(2^{k-1}-1)(|S_0|+|S_k|) \tag{12}$$

还需要说明不等式(12)左边恰好为$|S_1|+|S_2|+\cdots+|S_{k-1}|$,为此,只需证明$(\alpha_{i-1}-\alpha_i)i+(\alpha_{i+1}-\alpha_i)(k-i)=-1$, $1\leq i\leq k-1$. 事实上,

$$\begin{aligned}&(\alpha_{i-1}-\alpha_i)i+(\alpha_{i+1}-\alpha_i)(k-i)\\&=-\frac{2^{k-1}-1-\sum_{j=1}^{i-1}\binom{k}{j}}{(k-i+1)\binom{k}{i-1}}i+\frac{2^{k-1}-1-\sum_{j=1}^{i}\binom{k}{j}}{(k-i)\binom{k}{i}}(k-i)\\&=-\frac{2^{k-1}-1-\sum_{j=1}^{i-1}\binom{k}{j}}{\binom{k}{i}}+\frac{2^{k-1}-1-\sum_{j=1}^{i}\binom{k}{j}}{\binom{k}{i}}\\&=-\frac{\binom{k}{i}}{\binom{k}{i}}=-1\end{aligned} \tag{13}$$

综上所述,命题得证. 证毕.

因$C=S_0\cup S_1\cup\cdots\cup S_{k-1}\cup S_k$,由引理2可得$|S_1|+\cdots+|S_{k-1}|\geq\dfrac{2^{k-1}-1}{2^{k-1}}|C|$. 所以算法1解答$k\geq 2$的Max NAE-$k$-SAT任意实例,总有近似性能比为$\dfrac{2^{k-1}}{2^{k-1}-1}$.

当$k=2,3$时,算法1的近似性能比分别为2,$\dfrac{4}{3}$,并不好于Goemans、Williamson[10]及Zwick[15]给出的半定规划松弛法. 当$k=4$时,算法1的近似性能比与Karloff、Zwick[12]的半定规划松弛法所达到的近似性能比相同. 当$k\geq 5$时,目前未见有确定近似性能比的Max NAE-k-SAT求解算法. Zwick曾在文献[15]中指出,半定规划松弛法用于解答$k\geq 5$的Max NAE-k-SAT问题,显得十分困难. 本文算法弥补了这一不足.

2.2 算法的时间复杂性

设$|U|=n$,$|C|=m$. 给定一个U的赋值函数$a(U)$,判定一个子句属于集合$S_1,\cdots,S_k$中的哪一个,需要$O(k)$时间. 判定一个子句属于哪些$C[i,j]$,$1\leq i\leq k$,$1\leq j\leq n$及哪些$N[i,j]$,$0\leq i\leq k-1$,$1\leq j\leq n$,也需要$O(k)$时间. 因此子句集合$C[i,j]$,$N[i,j]$和S_i可在$O(km)$时间内得到. 找到一个布尔变量u_j,$1\leq j\leq n$,满足$\Delta F>0$,需要$O(kn)$时间. 因此算法1的一个While循环所花费的时间为$O(k(m+n))$.

算法1中While循环的执行次数必然受到目标函数F中的$|S_i|$前面系数α_i的影响. 当$k=2,3,\cdots,9$时,分别按照式(3)计算α_i,将α_i的数值列在表1中. 由式(3)不难知道,α_i是k的指数函数,但仅与k有关,与Max NAE-k-SAT实例中的布尔变量与子句数目无关.

表1 目标函数中参量α_i的取值

k	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8	α_9
2	$\frac{1}{2}$	0	∅	∅	∅	∅	∅	∅	∅
3	1	1	0	∅	∅	∅	∅	∅	∅
4	$\frac{7}{4}$	2	$\frac{7}{4}$	0	∅	∅	∅	∅	∅
5	3	$\frac{7}{2}$	$\frac{7}{2}$	3	0	∅	∅	∅	∅
6	$\frac{31}{6}$	6	$\frac{37}{6}$	6	$\frac{31}{6}$	0	∅	∅	∅
7	9	$\frac{31}{3}$	$\frac{34}{3}$	$\frac{34}{3}$	$\frac{31}{3}$	9	0	∅	∅
8	$\frac{127}{8}$	18	$\frac{445}{24}$	$\frac{56}{3}$	$\frac{445}{24}$	18	$\frac{127}{8}$	0	∅
9	$\frac{85}{3}$	$\frac{127}{4}$	$\frac{391}{12}$	$\frac{197}{6}$	$\frac{197}{6}$	$\frac{391}{12}$	$\frac{127}{4}$	$\frac{85}{3}$	0

因目标函数F中每个系数α_i均以分数形式给出,所以α_i的任何取值误差均可使算法1的近似性能比不能达到$\dfrac{2^{k-1}}{2^{k-1}-1}$. 若要保证算法1的近似性能比总能达到$\dfrac{2^{k-1}}{2^{k-1}-1}$,可将每个参量$\alpha_i$均乘以一个正整数$\beta$,使$\beta\cdot\alpha_i$被放大为正整数,$1\leq i\leq k-1$. 当$k$较小时,这种系数的整数倍放大对于算法的时间复杂性影响不大. 如当$k=8$时,每个α_i乘以24,则均变为正整数,其中$24\alpha_4=448$为最大. 此时有$0\leq F\leq 448m$,因此算法1的While循环最多执行$448m$次,由此可知,算法1的时间复杂性为$O(448\times 8m(m+n))$.

下面默认 $\beta(k)$ 为将 $\beta\alpha_1,\cdots,\beta\alpha_{k-1}$ 均放大为整数，且取值最小的正整数，显然 $\beta(k)\alpha_{\lfloor k/2\rfloor}$ 是 $\beta(k)\alpha_1,\cdots,\beta(k)\alpha_{k-1}$ 中的最大者，且是 k 的指数函数. 我们计算了 $k\leq 26$ 的所有目标函数的系数，将 k 取值 10～26 时，$\beta(k)$ 的数值列在表 2 中. 因 $\alpha_{\lfloor k/2\rfloor}\leq 2^{k-1}$，所以，$\beta(k)\alpha_{\lfloor k/2\rfloor}\leq 2^{k-1}\beta(k)$，数值最大.

表 2　将 α_i 放大为整数需要放大的倍数 $\beta(k)$

k	$\beta(k)$	k	$\beta(k)$	k	$\beta(k)$
10	30	16	1680	22	6930
11	15	17	840	23	3465
12	60	18	1260	24	27720
13	30	19	630	25	13860
14	210	20	1260	26	90090
15	105	21	630		

下面给出 $\beta(k)$ 上界的一个估计值.

性质 2.　设 $\beta(k)=\min\{\beta\mid\beta\alpha_i$ 为正整数，$1\leq i\leq k-1\}$，其中 α_i 由式(3)给定，$0\leq i\leq k$，则当 $k\geq 6$ 时，$\beta(k)\leq\lfloor k/2\rfloor!$.

证明. 若 k 为偶数，则 $\binom{k}{0}+\binom{k}{1}+\cdots+\left(\begin{matrix}k\\ \frac{k}{2}-1\end{matrix}\right)+\frac{1}{2}\binom{k}{k/2}=2^{k-1}$. 将式(3)中的 α_{i-1} 移到等号左边，将 $\binom{k}{l}=\frac{k(k-1)\cdots(k-l+1)}{l!}$ 代入该式右边的分数表达式中，可将 $(\alpha_i-\alpha_{i-1})$ 的表达式整理成一个分数和与另一个整数乘积的形式，即 $(\alpha_i-\alpha_{i-1})=N\cdot M$，其中

$$N=\left[\frac{1}{i!}+\cdots+\frac{(k-i)\cdots\left(k-\frac{k}{2}+2\right)}{\left(\frac{k}{2}-1\right)!}+\frac{(k-i)\cdots\left(k-\frac{k}{2}+1\right)}{2\left(\frac{k}{2}\right)!}\right],\ 1\leq i\leq k/2,$$

M 按照如下方式取值：若 $i=1$，则 $M=1$；若 $i\geq 2$，则 $M=(i-1)!$. 考察 N 的表达式最后一个分数项，因 $k\geq 6$，所以若 $i=1$ 或 $i=2$，则该分数项的分子部分 $(k-i)\cdots(k-k/2+1)$ 总能被 2 整除；若 $i\geq 3$，则 $M=(i-1)!$ 可以被 2 整除. 由此可知，$(k/2)!(\alpha_i-\alpha_{i-1})$ 必为整数. 由 $\alpha_0=0$ 及性质 1 立即得到，$(k/2)!\alpha_i$ 为整数，$1\leq i\leq k$. 因 $\beta(k)$ 是满足 $\beta(k)\alpha_i$ 为整数的最小正整数，$1\leq i\leq k-1$，所以 $\beta(k)\leq(k/2)!$.

若 k 为奇数，则 $\binom{k}{0}+\binom{k}{1}+\cdots+\left(\begin{matrix}k\\ \frac{k-1}{2}-1\end{matrix}\right)+\left(\begin{matrix}k\\ \frac{k-1}{2}\end{matrix}\right)=2^{k-1}$. 类似于 k 为偶数时的讨论，有 $(\alpha_i-\alpha_{i-1})=N\cdot M$，其中，

$$N=\left[\frac{1}{i!}+\cdots+\frac{(k-i)\cdots\left(k-\frac{k-1}{2}+2\right)}{\left(\frac{k-1}{2}-1\right)!}+\frac{(k-i)\cdots\left(k-\frac{k-1}{2}+1\right)}{\left(\frac{k-1}{2}\right)!}\right],\ 1\leq i\leq\frac{k-1}{2},$$

M 取值方式如下：若 $i=1$，则 $M=1$；若 $i\geq 2$，则 $M=(i-1)!$. 由此可知，$((k-1)/2)!(\alpha_i-\alpha_{i-1})$ 必为整数. 由 $\alpha_0=0$ 及性质 1 立即得到，$\left(\frac{k-1}{2}\right)!\alpha_i$ 为整数，$1\leq i\leq k$. 所以 $\beta(k)\leq\left(\frac{k-1}{2}\right)!$.

当 $k=8$ 时，$\beta(k)=24=4!$，由性质 2 给出的上界等于表 1 中实际计算出的上界. 另外表 2 中的 $\beta(k)$ 取值远小于性质 2 给出的上界.

定理 1.　算法 1 解答任意 Max NAE-k-SAT 实例的近似性能比均不大于 $\frac{2^{k-1}}{2^{k-1}-1}$. 时间复杂性为 $O(\beta(k)2^{k-1}k(m+n)m)$. 其中 n,m 分别为布尔变量数目和子句数目，当 $k\geq 6$ 时，$\beta(k)\leq\lfloor k/2\rfloor!$.

在实际 Max NAE-k-SAT 求解中，一般并不需要将 F 中的系数 α_i 转化为整数，可直接采用小数来表示 α_i 并计算 F 的数值. 我们采用 64 位浮点数来表示每个系数 α_i，以 Java 程序实现算法 1，随机产生 Max NAE-k-SAT 实例，在各种笔记本及台式机电脑上进行了实际求解测试. 在针对 $k\leq 40$ 的随机 Max NAE-k-SAT 实例的求解测试中，我们从未遇到算法 1 的求解性能大于 $\frac{2^{k-1}}{2^{k-1}-1}$ 的情况. 分别取 $k=4,5,16,19,28,29$，利用算法 1 的 Java 程序，将在以酷睿 U9400 为中央处理器的笔记本电脑上实际求解的运行时间列在表 3 中. 表 3 给出的测试结果中，最大和最小运行时间分别为随机产生的 20 个实例的求解时间的最大者和最小者.

表 3　算法 1 的实际运行时间（$|U|$、$|C|$ 表示布尔变量数目和子句数目；Max、Min 分别表示算法求解 20 个实例的最大和最小运行时间，Aver 为平均运行时间）

k	$\|U\|$	$\|C\|$	Aver/ms	Min/ms	Max/ms
4	2000	35000	86277	82633	100637
5	2000	35000	92315	83742	100939
16	500	20000	31644	28112	39847
19	200	60000	23913	17769	33228
28	200	60000	29687	19017	37281
29	300	60000	62535	48548	72947

实际测试表明,直接采用浮点数表示目标函数 F 中的系数 α_i,并不影响算法 1 的求解性能.然而算法 1 仅适合于 k 为小常数时的 Max NAE-k-SAT 求解,如 $k\leq 40$.

3 一般最大不全 k 满足问题的局部搜索

若 Max NAE-SAT 实例不仅含有 k 个字母的子句,也含有多于 k 个字母的子句,则问题演变为更一般的最大不全 k 满足问题,即 Max NAE-(k)-SAT.第 2 节给出的算法并不能直接用于解答 Max NAE-(k)-SAT 实例.下面考虑推广算法 1,使之能够解答 Max NAE-(k)-SAT.

Max NAE-(k)-SAT 问题的求解目标与 Max NAE-k-SAT 的求解目标相同,实例为:布尔变量集合 $U=\{u_1,\cdots,u_n\}$,子句集合 $C=\{C_1,\cdots,C_m\}$,子句 $C_t=\{x[t,1],\cdots,x[t,k_t]\}$,其中 $x[t,j]\in\{u_1,\cdots,u_n\}\cup\{\bar{u}_1,\cdots,\bar{u}_n\}$,$1\leq t\leq m$,$1\leq j\leq k_t$,$k_t\geq k$.

由于 Max NAE-(k)-SAT 实例中,每个子句的布尔变量字母数并不相等,所以需要进一步区分 C 中 i-满足但所含字母总数不同的子句集合.

给定布尔变量赋值 $a(U)$,设 $S[b,i]$表示含有 b 个布尔变量字母,且为 i-满足的子句集合;$C[b,i,j]$ 表示含有 b 个布尔变量字母,为 i-满足,且被 u_j 满足的子句集合;$N[b,i,j]$表示含有 b 个布尔变量字母,为 i-满足且不被 u_j 满足的子句集合.

设 Max NAE-(k)-SAT 的子句集合 C 中,每个子句最多含有 $k_{\max}$ 个字母,则 C 中子句可含有 $k,\cdots,k_{\max}$ 个字母.欲增大 C 中不全满足的子句数目,考虑利用局部搜索盲目地最大化如下目标函数:

$$G=\sum_{b=k}^{k_{\max}}(\alpha_{b,1}|S[b,1]|+\cdots+\alpha_{b,b-1}|S[b,b-1]|) \tag{14}$$

目标函数 G 是 F 的推广.分别考虑 C 中含有 $k,\cdots,k_{\max}$ 个字母的子句集合,依照式(1)选择解答 Max NAE-b-SAT 实例的目标函数,$k\leq b\leq k_{\max}$,将这些目标函数相加即得到式(14).类似于式(3)中 α_i 的取值,目标函数 G 中各个系数取值如下:

$$\alpha_{b,i}=\begin{cases}0, & i=0\\ \alpha_{b,i-1}+\dfrac{2^{b-1}-1-\sum_{j=1}^{i-1}\binom{b}{j}}{(b-i+1)\binom{b}{i-1}}, & 1\leq i\leq b\end{cases} \tag{15}$$

由性质 1 可知式(15)给出的 $\alpha_{b,i}$ 满足 $\alpha_{b,i}=\alpha_{b,b-i}$,$0\leq i\leq b$.给定 U 的赋值函数 $a(U)$,将布尔变量 u_j 的赋值取反,$S[b,i]$中的子句只可能变为 $S[b,i+1]$或者 $S[b,i-1]$中的子句,不会变为 $S[b',\cdot]$中的子句,$b\neq b'$.所以,由 i-满足变为($i-1$)-满足的子句导致 G 产生增量:$\sum_{b=k}^{k_{\max}}(\alpha_{b,i-1}-\alpha_{b,i})|C[b,i,j]|$,$1\leq i\leq k$;由 i-满足变为($i+1$)-满足的子句导致 G 产生增量:$\sum_{b=k}^{k_{\max}}(\alpha_{b,i+1}-\alpha_{b,i})|N[b,i,j]|$,$0\leq i\leq k-1$.于是 G 因 u_j 赋值取反所产生的增量为

$$\begin{aligned}\Delta G=&\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b}(\alpha_{b,i-1}-\alpha_{b,i})|C[b,i,j]|+\\&\sum_{b=k}^{k_{\max}}\sum_{i=0}^{b-1}(\alpha_{b,i+1}-\alpha_{b,i})|N[b,i,j]|\\=&\sum_{b=k}^{k_{\max}}\Big\{\sum_{i=1}^{b}(\alpha_{b,i-1}-\alpha_{b,i})|C[b,i,j]|+\\&\sum_{i=0}^{b-1}(\alpha_{b,i+1}-\alpha_{b,i})|N[b,i,j]|\Big\}\end{aligned} \tag{16}$$

因此解答 Max NAE-(k)-SAT 的局部搜索步骤与算法 1 完全相同,只需将 While 循环中 u_j 赋值取反的条件修改为 $\Delta G>0$ 即可,其中 ΔG 的计算公式由式(16)给出.当算法运行结束时,任意布尔变量 u_j 赋值取反,总有 $\Delta G\leq 0$.即

$$\begin{aligned}&\sum_{b=k}^{k_{\max}}\Big(\sum_{i=1}^{b}(\alpha_{b,i-1}-\alpha_{b,i})|C[b,i,j]|+\\&\sum_{i=0}^{b-1}(\alpha_{b,i+1}-\alpha_{b,i})|N[b,i,j]|\Big)\leq 0,\ 1\leq j\leq n\end{aligned} \tag{17}$$

将算法 1 中 u_j 赋值取反条件修改为 $\Delta G>0$,并将赋值取反条件修改后的算法重新命名为算法 2:Max NAE-(k)-SAT(U,C).

引理 3. 任给布尔变量赋值函数 $a(U)$,则有 $\sum_{j=1}^{n}|C[b,i,j]|=i|S[b,i]|$,$\sum_{j=1}^{n}|N[b,i,j]|=(b-i)|S[b,i]|$.

证明. 设 C 中含有 b 个布尔变量字母的子句集合为 $S[b]$,则有 $S[b]=S[b,0]\cup S[b,1]\cup\cdots\cup S[b,b]$,任意布尔变量赋值取反,$S[b]$保持不变.由引理 1 可知该引理成立. 证毕.

引理 4. 设算法结束时得到布尔变量赋值 $a(U)$,则 $\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b-1}|S[b,i]|\geq\sum_{b=k}^{k_{\max}}(2^{b-1}-1)(|S[b,0]|+|S[b,b]|)$.

证明. 当算法结束时,将任意布尔变量 u_j 赋值取反,目标函数 G 取值均不增加.那么就有式(17)

成立. 将式(17)的 n 个不等式左右分别相加，得到

$$\sum_{j=1}^{n}\Big\{\sum_{b=k}^{k_{\max}}\Big(\sum_{i=1}^{b}(\alpha_{b,i-1}-\alpha_{b,i})|C[b,i,j]|+\sum_{i=0}^{b-1}(\alpha_{b,i+1}-\alpha_{b,i})|N[b,i,j]|\Big)\Big\}=\sum_{b=k}^{k_{\max}}\Big(\sum_{i=1}^{b}(\alpha_{b,i-1}-\alpha_{b,i})\sum_{j=1}^{n}|C[b,i,j]|+\sum_{i=0}^{b-1}(\alpha_{b,i+1}-\alpha_{b,i})\sum_{j=1}^{n}|N[b,i,j]|\Big)\leq 0 \tag{18}$$

由引理 3，将 $\sum_{j=1}^{n}|C[b,i,j]|=i|S[b,i]|$，$\sum_{j=1}^{n}|N[b,i,j]|=(b-i)|S[b,i]|$ 代入式(18)得到

$$\sum_{b=k}^{k_{\max}}\Big\{b\alpha_{b,1}|S[b,0]|+b\alpha_{b,b-1}|S[b,b]|+\sum_{i=1}^{b-1}[(\alpha_{b,i-1}-\alpha_{b,i})i+(\alpha_{b,i+1}-\alpha_{b,i})(b-i)]|S[b,i]|\Big\}\leq 0 \tag{19}$$

由式(15)可知，$\alpha_{b,1}=\dfrac{2^{b-1}-1}{b}$，再由性质 1 得 $\alpha_{b,b-1}=\dfrac{2^{b-1}-1}{b}$. 由式(15)还可得到

$$(\alpha_{b,i-1}-\alpha_{b,i})=-\frac{2^{b-1}-1-\sum_{j=1}^{i-1}\binom{b}{j}}{(b-i+1)\binom{b}{i-1}},\ 1\leq i\leq b-1 \tag{20}$$

$$(\alpha_{b,i+1}-\alpha_{b,i})=\frac{2^{b-1}-1-\sum_{j=1}^{i}\binom{b}{j}}{(b-i)\binom{b}{i}},\ 1\leq i\leq b-1 \tag{21}$$

由式(20)、(21)可推得 $(\alpha_{b,i-1}-\alpha_{b,i})i+(\alpha_{b,i+1}-\alpha_{b,i})(b-i)=-1$. 将 $\alpha_{b,1}=\alpha_{b,b-1}=\dfrac{2^{b-1}-1}{b}$ 和 $(\alpha_{b,i-1}-\alpha_{b,i})i+(\alpha_{b,i+1}-\alpha_{b,i})(b-i)=-1$ 代入式(19)，整理后即得到

$$\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b-1}|S[b,i]|\geq\sum_{b=k}^{k_{\max}}(2^{b-1}-1)(|S[b,0]|+|S[b,b]|) \tag{22}$$

命题得证. 证毕.

已知 Max NAE-(k)-SAT 实例中的子句集合为 C，则 $|C|=\sum_{b=k}^{k_{\max}}\sum_{i=0}^{b}|S[b,i]|$. 所以由引理 4 可得到 $\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b-1}|S[b,i]|\geq\dfrac{2^{k-1}}{2^{k-1}-1}|C|$，即算法 2 的近似性能比为 $\dfrac{2^{k-1}}{2^{k-1}-1}$. 算法 2 的时间复杂性可由算法 1 的时间复杂性推得.

定理 2. 算法 2 解答 Max NAE-(k)-SAT 实例的近似性能比不大于 $\dfrac{2^{k-1}}{2^{k-1}-1}$. 时间复杂性为 $O\Big(\sum_{b=k}^{k_{\max}}\beta(b)2^{b-1}b(m+n)m\Big)$. 其中 n,m 分别表示布尔变量数目和子句数目，$\beta(b)\leq\lfloor b/2\rfloor!$.

例如，若一个Max NAE-(4)-SAT 实例含有4个字母和 5 个字母的子句，则利用算法 2 解答该实例的目标函数应为 $\frac{7}{4}|S[4,1]|+2|S[4,2]|+\frac{7}{4}|S[4,3]|+3|S[5,1]|+\frac{7}{2}|S[5,2]|+\frac{7}{2}|S[5,3]|+3|S[5,4]|$，若将每个系数均放大为整数，则目标函数为 $7|S[4,1]|+8|S[4,2]|+7|S[4,3]|+12|S[5,1]|+14|S[5,2]|+14|S[5,3]|+12|S[5,4]|$. 布尔变量 u_j 赋值取反的条件应为 $-7|C[4,1,j]|-2|C[4,2,j]|+2|C[4,3,j]|+7|C[4,4,j]|+7|N[4,0,j]|+2|N[4,1,j]|-7|N[4,2,j]|-2|N[4,3,j]|-12|C[5,1,j]|-2|C[5,2,j]|+2|C[5,4,j]|+12|C[5,5,j]|+12|N[5,0,j]|+2|N[5,1,j]|-2|N[5,3,j]|-12|N[5,4,j]|>0$.

分别取 3 组 Max NAE-(k)-SAT 实例，第 1 组含有 4 字母和 5 字母子句；第 2 组含有 5 字母、6 字母和 8 字母子句；第 3 组含有 7、10、11 字母子句，以 Java 编程实现该算法，采用 64 位浮点数表示目标函数中的系数 $\alpha_{b,i}$，将在 U9400 笔记本电脑上的实际求解运行时间列在表 4 中.

表 4　Max NAE-(k)-SAT 实际求解运行时间(Max、Min 分别表示算法求解 20 个随机生成实例的最大和最小运行时间)

k	$\lvert U\rvert$	$\lvert C\rvert$	Max/ms	Min/ms
4,5	1000	20000	134987	102711
5,6,8	1000	15000	139386	91588
7,10,11	500	12000	45291	60061

4　解答 Max-(k)-SAT 的新近似算法

Khanna 等人[28]给出了解答 Max-k-SAT 的局部搜索算法，近似性能比为 $\dfrac{2^k}{2^k-1}$. 但该算法并不能用于解答一般最大 k 可满足问题. 本节我们利用解答 Max NAE-(k)-SAT 的局部搜索算法解答一般最大 k 可满足问题. 一般最大 k 可满足问题简记为 Max-(k)-SAT.

Max-(k)-SAT 的实例与 Max NAE-(k)-SAT

的实例相同,求解目标则为:寻求 U 中布尔变量的赋值 $a:U\rightarrow\{T,F\}$,最大化 $|\{C_t:a(x[t,1])\vee\cdots\vee a(x[t,k_t])=T\}|$.

任一 Max-(k)-SAT 实例亦可看作 Max NAE-(k)-SAT 实例.所以算法 2 可用于解答 Max-(k)-SAT 实例.然而这样做还不能保证算法的近似性能比达到 $\frac{2^k}{2^k-1}$.下面我们说明,只需将 Max-(k)-SAT 实例作为 Max NAE-(k)-SAT 实例以算法 2 求解,再将得到的解加以改造,即为原 Max-(k)-SAT 实例的近似性能比为 $\frac{2^k}{2^k-1}$ 的解.

任给 Max-(k)-SAT 实例,以式(14)做目标函数,利用算法 2,可给出一个 U 的赋值函数 $a(U)$.由引理 4,赋值函数 $a(U)$ 使得

$$\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b-1}|S[b,i]|\geq(2^{k-1}-1)\sum_{b=k}^{k_{\max}}(|S[b,0]|+|S[b,b]|)$$

$$\geq(2^k-2)\min\left\{\sum_{b=k}^{k_{\max}}|S[b,0]|,\sum_{b=k}^{k_{\max}}|S[b,b]|\right\}\quad(23)$$

若 $\sum_{b=k}^{k_{\max}}|S[b,b]|\geq\sum_{b=k}^{k_{\max}}|S[b,0]|$,则

$$\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b}|S[b,i]|\geq(2^k-1)\sum_{b=k}^{k_{\max}}|S[b,0]|\quad(24)$$

否则,式(24)未必成立.然而可以利用将所有布尔变量赋值取反,使 $\sum_{b=k}^{k_{\max}}|S[b,b]|\geq\sum_{b=k}^{k_{\max}}|S[b,0]|$ 得以成立.由此给出解答 Max-(k)-SAT 问题的算法如下.

算法 3. Max-(k)-SAT(U,C).

//U 为布尔变量集合,C 为子句集合

1. 调用 MaxNAE-(k)-SAT(U,C),得到 $a(U)$;
2. If $\left(\sum_{b=k}^{k_{\max}}S[b,b]\geq\sum_{b=k}^{k_{\max}}S[b,0]\right)$ then $a'(U)\leftarrow a(U)$;
3. Else $a'(U)\leftarrow\bar{a}(U)$;
4. End if
5. Return $a'(U)$

在算法 3 中,$\bar{a}(U)$ 表示将 U 中每个布尔变量由 $a(\cdot)$ 赋予的布尔值取反,得到的 U 的赋值函数.

引理 5. 设算法 3 返回布尔变量赋值 $a'(U)$,$S[b,i]$ 表示以赋值函数 $a'(U)$ 为 U 中布尔变量赋值后,C 中含有 b 个字母且为 i-满足的子句集合.则 $\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b}|S[b,i]|\geq\frac{2^k-1}{2^k}|C|$.

证明. 为更确切地表达赋值函数对 C 中 i-满足子句集合的影响,再设 $S[f(\cdot),b,i]$ 表示以赋值函数 $f(U)$ 为 U 中布尔变量赋值后,C 中含有 b 个字母且为 i-满足的子句集合,因此有 $S[b,i]=S[a'(\cdot),b,i]$.若 $a'(U)=a(U)$,由引理 4 知式(23)成立.若 $a'(U)=\bar{a}(U)$,则 $S[b,i]=S[\bar{a}(\cdot),b,i]=S[a(\cdot),b,b-i]$,$0\leq i\leq b$,所以,

$$\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b-1}|S[b,i]|=\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b-1}|S[a(\cdot),b,i]|\quad(25)$$

同理,

$$\sum_{b=k}^{k_{\max}}(|S[b,0]|+|S[b,b]|)=$$

$$\sum_{b=k}^{k_{\max}}(|S[a(\cdot),b,0]|+|S[a(\cdot),b,b]|\quad(26)$$

所以式(23)仍然成立.另外由算法 3 可知,无论 $a'(U)=a(U)$ 还是 $a'(U)=\bar{a}(U)$,总有

$$\sum_{b=k}^{k_{\max}}S[b,b]\geq\sum_{b=k}^{k_{\max}}S[b,0]\quad(27)$$

所以,由式(23)和(27)可得到

$$\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b}|S[b,i]|>(2^k-1)\sum_{b=k}^{k_{\max}}|S[b,0]|\quad(28)$$

因 $\sum_{b=k}^{k_{\max}}\sum_{i=0}^{b}|S[b,i]|=|C|$,所以,

$$\sum_{b=k}^{k_{\max}}\sum_{i=1}^{b}|S[b,i]|>\frac{2^k-1}{2^k}|C|\quad(29)$$

引理得证. 证毕.

算法 3 与算法 2 的时间复杂性相同.因此有如下定理.

定理 3. 算法 3 解答任意 Max-(k)-SAT 实例的近似性能比不大于 $\frac{2^k}{2^k-1}$,时间复杂性为 $O\left(\sum_{b=k}^{k_{\max}}\beta(b)2^{b-1}b(m+n)m\right)$.其中 n,m 分别表示布尔变量数目和子句数目,$\beta(b)\leq\lfloor b/2\rfloor!$.

5 最大不全 k 满足问题的近似计算复杂性

Hastad[27] 曾证明,对于任意 $k\geq3$,不存在解答 Max-k-SAT 的多项式时间算法,近似性能比小于 $\frac{2^k}{2^k-1}$,除非 P=NP.我们利用该结果证明,对于任意 $k\geq4$,不存在解答 Max NAE-k-SAT 的多项式时间算法,近似性能小于 $\frac{2^{k-1}}{2^{k-1}-1}$.即有如下定理.

定理 4. 对于任意 $k\geq4$,不存在解答 Max NAE-k-SAT 的多项式时间算法,近似性能达到 $\frac{2^{k-1}}{2^{k-1}-1}-\varepsilon$,除非 P=NP.

证明. 将Max-l-SAT 问题归约到 Max NAE-k-SAT，$k=l+1$. 设 $l\geq 3$，$\langle U,C\rangle$ 是Max-l-SAT的任一实例，构造 Max NAE-$(l+1)$-SAT 实例 $\langle U',C'\rangle$ 如下.

U'包含 U 的所有布尔变量，仅比 U 多一个布尔变量 o，即设 $U=\{u_1,\cdots,u_n\}$，则 $U'=\{u_1,\cdots,u_n,o\}$. 将 C 中每个子句均增加布尔变量字母 o，形成 C'的子句. 即设 $C=\{C_1,\cdots,C_m\}$，其中 $C_t=(x[t,1],\cdots,x[t,l])$，则对应每个$C_t$，构造一个子句$D_t=(x[t,1],\cdots,x[t,l],o)$，$1\leq t\leq m$，而 $C'=\{D_1,\cdots,D_m\}$.

设有 U 的真值指派 $a(u_1),\cdots,a(u_n)$，使 C 中 M 个子句满足. 为 o 赋值 $a(o)=F$，则 $a(u_1),\cdots,a(u_n)$ 与 $a(o)$必然使C'中恰好 M 个子句不全满足. 这是因为若 $a(u_1),\cdots,a(u_n)$使 C_t满足，则其中必有一个布尔变量字母取值为 T，又 $a(o)=F$，所以 D_t是不全满足的；若 $a(u_1),\cdots,a(u_n)$使 C_t不满足，又因 $a(o)=F$，所以 D_t也是不满足的.

另一方面设有 U'的真值指派 $a(u_1),\cdots,a(u_n)$，$a(o)$使C'中 M 个子句不全满足，则 $a(u_1)\oplus a(o),\cdots,a(u_n)\oplus a(o)$必然使 C 中恰好 M 个子句满足. 理由如下：

若 $a(o)=F$，则 C'中不存在全满足的子句，且 $a(u_i)\oplus a(o)=a(u_i)$，$1\leq i\leq n$. 若 $a(u_1),\cdots,a(u_n)$，$a(o)$使 D_t不全满足，则 D_t必有一个布尔变量字母 $x[t,\cdot]$取值为 T，所以 $a(x[t,\cdot])$使 C_t满足. 若 $a(u_1),\cdots,a(u_n),a(o)$使 D_t不满足，则 C_t中每个字母 $x[t,\cdot]$均取值 F，所以 C_t也是不满足的.

若 $a(o)=T$，则 C'中不存在不满足的子句，且 $a(u_i)\oplus a(o)=\overline{a(u_i)}$，$1\leq i\leq n$. 若 $a(u_1),\cdots,a(u_n)$，$a(o)$使D_t不全满足，则D_t必有一个布尔变量字母$x[t,\cdot]$取值为 F，所以$\overline{a(x[t,\cdot])}$使 C_t满足. 若$a(u_1),\cdots,a(u_n),a(o)$使 D_t全满足，则$\overline{a(x[t,1])},\cdots,\overline{a(x[t,l])}$使 C_t不满足.

因此，若存在 U 的真值指派，使 C 中 M 个子句满足，当且仅当存在 U'的真值指派，使 C'中 M 个子句不全满足. 设 $OPT(U,C)$与 $OPT(U',C')$分别为 C 和 C' 中最多可满足的和最多能够不全满足的子句数目，则 $OPT(U,C)=OPT(U',C')$.

若存在解答 Max NAE-k-SAT 的多项式时间算法 A，对于任意 Max NAE-k-SAT 实例$\langle U',C'\rangle$，总能得到 U'的真值指派 $a(U')$，使 C'中不全满足的子句数目达到 $A(U',C')\geq\left(\frac{2^{k-1}-1}{2^{k-1}}-\varepsilon\right)OPT(U',C')$，则由 $a(u_1)\oplus a(o),\cdots,a(u_n)\oplus a(o)$为 U 中布尔变量赋值，可使 C 中满足的子句数目同样为 $A(U',C')\geq\left(\frac{2^{k-1}-1}{2^{k-1}}-\varepsilon\right)OPT(U,C)$，与 Hastad 的结论矛盾. 证毕.

定理 4 说明本文解答 Max NAE-k-SAT 的算法所达到的近似性能比，当 $k\geq 4$ 时是最优的.

6 结束语

本文给出 Max NAE-k-SAT 问题的局部搜索算法，近似性能比为 $\frac{2^{k-1}}{2^{k-1}-1}$. 进一步给出 Max NAE-$(k)$-SAT 问题的局部搜索算法，近似性能比亦为$\frac{2^{k-1}}{2^{k-1}-1}$. 本文算法可用于解答 Max-$(k)$-SAT 问题，近似性能比可达到$\frac{2^k}{2^k-1}$. 本文最后证明对于任意 $k\geq 4$，不存在解答 Max NAE-k-SAT 的多项式时间算法，近似性能小于$\frac{2^{k-1}}{2^{k-1}-1}$. 因而说明本文算法 1 解答 Max NAE-$k$-SAT 所能达到的近似性能比是最优的，$k\geq 4$.

本文算法的时间复杂性仍是 k 的指数函数，可认为是关于 k 的参数化算法. 能否设计出解答最大不全 k 满足问题以及最大 k 可满足问题的局部搜索算法，使其时间复杂性既是布尔变量个数及子句个数的多项式函数，又是关于 k 的多项式函数，仍是十分有趣的问题. 本文将算法目标函数中系数放大为整数时，给出了放大倍数 $\beta(k)$ 的上界，然而这个 $\beta(k)$的上界的估计值距离实际计算得到的数值仍有较大差距. 给出一个目标函数系数放大倍数更紧密的上界，有助于更确切地分析算法的时间复杂性，仍然值得进一步研究.

参 考 文 献

[1] Cook S A. The complexity of theorem-proving procedures// Proceedings of the 3rd Annual ACM Symposium on Theory 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] Abidor A, Berkovitch I, Zwick U. Improved approximation algorithms for Max NAE-SAT and Max SAT//Proceedings of the WAOA 2005. Palma de Mallorca, Spain, 2005. LNCS-3879. 2006: 27-40

[4] Papadimitirou C H. Computational Complexity. California, USA: Addison Wesley, Redwood City, 1993

[5] Han Q, Ye Y, Zhang J. Improved approximation for Max Set Splitting and Max NAE SAT. Discrete Applied Mathematics, 2004, 142(1-3): 133-149

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

[7] Motwani R, Raghavan P. Randomized Algorithms. The Edinburgh Building, Cambridge CB2 2RU, UK: Cambridge University Press, 1995

[8] Yannakakis M. On the approximation of Maximum Satisfiability. Journal of Algorithms, 1994, 17(3): 475-502

[9] 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

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

[11] Feige U, Goemans M X. Approximating the value of two prover proof systems, with applications to MAX 2SAT and MAX DICUT//Proceedings of the 3rd Israel Symposium on Theory of Computing and Systems. Tel Aviv, Israel, 1995: 182-189

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

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

[14] Andersson G, Engebretsen L. Better approximation algorithms for set splitting and not-all-equal SAT. Information Processing Letters, 1998, 65(6): 305-311

[15] Zwick U. Outward rotations: A tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems//Proceedings of the 31st Annual ACM Symposium on Theory of Computing. Atlanta, Georgia, 1999: 679-687

[16] Mahajan S, Ramesh H. Derandomizing semidefinite programming based approximation algorithms//Proceedings of the 36th Annual IEEE Symposium on Foundations of Computer Science. Milwaukee, Wisconsin, 1995: 162-169

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

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

[19] Selman B, Kautz H A, Cohen B. Noise strategies for improving local search//Proceedings of the AAAI'94. Seattle, Washington, USA, 1994: 337-343

[20] McAllester D, Selman B, Kautz H. Evidence for invariants in local search//Proceedings of the AAAI'1997. Providence, USA, 1997: 321-326

[21] Mazure B, Sais L, Gregoire E. Tabu search for SAT//Proceedings of the 14th National Conference on Artificial Intelligence (AAAI'97). Providence, USA, 1997: 281-285

[22] Huang W, Zhang D, Wang H. An algorithm based on tabu search for satisfiability problem. Journal of Computer Science and Technology, 2002, 17(3): 340-346

[23] Schurmans D, Southey F. Local search characteristics of incomplete SAT procedures. Artificial Intelligence, 2001, 132(2): 121-150

[24] Cai S, Su K. Configuration checking with aspiration in local search for SAT//Proceedings of the AAAI'2012. Toronto, Ontario, Canada, 2012: 434-440

[25] Zhu Da-Ming, Ma Shao-Han, Zhang Ping-Ping. The local search algorithms for maximum 3-satisfiablility problem. Chinese Journal of Computers, 2010, 33(7): 1127-1139(in Chinese)
(朱大铭，马绍汉，张平平. 合取范式 3 可满足问题的局部搜索近似算法. 计算机学报，2010, 33(7): 1127-1139)

[26] Zhu Da-Ming, Ma Shao-Han, Zhang Ping-Ping. Tight bounds on local search to approximate the maximum satisfiability problems//Proceedings of the COCOON'2011. LNCS 6842. Dallas, Texas, USA, 2011: 49-61

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

[28] Khanna S, Motwani R, Madhu S, Umesh V. On syntactic versus computational views of approximability. SIAM Journal on Computing, 1998, 28(1): 164-191

[29] Dantsin E, Goerdt A, Hirsch E A, et al. A deterministic $(2-2/(k+2))^n$ algorithm for k-SAT based on local search. Theoretical Computer Science, 2002, 289(1): 69-83

XIAN Ai-Yong, born in 1986, M. S. candidate. His research interests include algorithm design and analysis, software engineering.

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

Background

This paper involves the algorithms and complexity of satisfiability problems for conjunctive norm forms. We focus on local search algorithms to solve the maximum not-all-equal satisfiability problems, and evaluate their performances.

The satisfiability problem is the central problem in theoretical computer science. The maximum satisfiability problem (Max-SAT) is the optimization version of the satisfiability problem. The maximum not-all-equal satisfiability problem (Max NAE-SAT) is a generalization of Max-SAT. Local search has been testified to be very effective for solving the satisfiability problems. However, there are rare related results for the local search method to solve the not-all-equal satisfiability problems. Max NAE-SAT is named Max NAE-k-SAT if each clause in its instance contains $k(\geq 2)$ literals. To our knowledge, the latest algorithmic approaches for the problems of not-all-equal satisfiability are the semi-definite programming relaxation algorithm of Avidor et al to solve Max NAE-SAT, which can achieve the conjectured performance ratio 1.2079 ($=1/0.8279$). The best algorithms for maximum not-all-equal 2, 3 and 4 satisfiability problems come from the semi-definite programming relaxation methods given by Goemans and Williamson, Zwick, Karloff and Zwick, with performances 1.139 (1/0.878), 1.100 47 (1/0.9087) and 8/7 respectively. But when $k \geq 5$, the semi-definite programming relaxation method becomes very difficult to solve Max NAE-k-SAT. Zwick also expressed the same viewpoint in his paper for using semi-definite programming relaxation to solve Max NAE-3-SAT.

In this paper, we first propose a local search algorithm to solve Max NAE-k-SAT. This algorithm can achieve the performance ratio $2^{k-1}/(2^{k-1}-1)$ for $k \geq 2$. Relative to this result, we have not seen any approximation algorithms for Max NAE-k-SAT with $k \geq 5$. Secondly, we follow the idea for solving Max NAE-k-SAT to propose a local search algorithm to solve the more generalized version of Max NAE-k-SAT which has clauses each containing k or more literals. This algorithm can still achieve the performance ratio $2^{k-1}/(2^{k-1}-1)$. The extension of the algorithms from solving Max NAE-k-SAT to solving its generalized version is non-trivial, because the algorithm for Max NAE-k-SAT cannot be directly used for its generalized version. Using the method for the generalized version of Max NAE-k-SAT, we propose a new $2^k/(2^k-1)$-approximation algorithm for the generalized version of the maximum k satisfiability problem (Max-k-SAT) which has clauses each containing k or more literals. Finally, we prove that if P$\neq$NP, then the maximum not-all-equal k satisfiability problem cannot be approximated within $2^{k-1}/(2^{k-1}-1)$ for $k \geq 4$. This implies the performance ratio of our algorithm for the maximum not-all-equal k satisfiability problem is optimal, when $k \geq 4$.

This paper is supported by the National Natural Science Foundation of China (61070019). This project aims at a class of problems on genome comparations and other areas to design approximation algorithms. Local search is the heavy point in our research for designing algorithms. Designing algorithms for the problems of our project needs to try every effort to minimize the performance ratio, in order to make sure the algorithm can give efficient solutions. We have designed the local search algorithm for maximum 3 satisfiability problem with performance ratio 8/7 in 2009. We have also designed approximation algorithms for the genome rearrangement sorting problems. For example, we give a 1.75-approximation algorithm for unsigned translocation sorting in 2005, and a 2.25-approximation algorithm for unsigned cut-and-paste sorting in 2012.