DOI:10.3969/j.issn.1673-4785.201206010
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.TP.20121116.1701.009.html

基于分支回溯的NAE-3SAT问题求解算法

谷文祥[1,2],傅琳璐[1],周俊萍[1],姜蕴晖[1]
(1.东北师范大学 计算机科学与信息技术学院,吉林 长春 130117;2.长春建筑学院 基础教学部,吉林 长春 130607)

摘 要:NAESAT问题是可满足性问题的一个重要扩展,在集合分裂、最大割集等NP完全问题中有着重要的应用.针对NAESAT问题的泛化NAE-3SAT问题,提出了一个基于分支回溯的精确算法NAE.算法给出了多种化简规则,这些化简规则很好地提高了算法的时间效率.最后证明了算法在最坏情况下的时间复杂度上界为$O(1.618^n)$,其中n为公式中的变量数目.

关键词:NAESAT;NAE-3SAT;时间复杂性;NAE-3SAT问题上界;变量数目;分支回溯
中图分类号:TP301 **文献标志码**:A **文章编号**:1673-4785(2012)06-0506-06

Solution algorithm for the NAE-3SAT problem based on DPLL

GU Wenxiang[1,2], FU Linlu[1], ZHOU Junping[1], JIANG Yunhui[1]
(1. School of Computer Science and Information Technology, Northeast Normal University, Changchun 130117, China; 2. Department of Basic Subjects Teaching, Changchun Architecture & Civil Engineering College, Changchun 130607, China)

Abstract: The not-all-equal satisfiability (NAESAT) problem is an important extension of the satisfiability (SAT) problem. It has an important application in many NP-complete problems, such as the SET SPLITTING problem and the MAXCUT problem. An exact algorithm NAE based on Davis-Putnam-Logemann-Loveland (DPLL) for the not-all-equal 3-satisfiability (NAE-3SAT) problem is proposed. In the algorithm some new reduction rules are used to simplify the formula. The reduction rules increase the efficiency of the algorithm. The worst-case upper bound for the algorithm is proved to be $O(1.618^n)$, where n is the number of variables in the formula.

Keywords: not-all-equal satisfiability; not-all-equal 3-satisfiability; time complexity; upper bounds for Not-All-Equal 3-satisfiability; number of variables; DPLL

可满足性问题(satisfiability,SAT)作为计算机理论与应用的核心问题,在智能规划、机器视觉、逻辑推理机以及电子设计自动化等领域中有着重要的理论研究与实际应用价值[1-4].SAT问题的计算复杂性是NP完全的[5],这意味着如果P≠NP成立,则无法在多项式时间内解决SAT问题.这时从理论上分析该问题在最坏情况下的时间复杂性上界尤为重要,因为此时对该问题上界的一个微小的改进,例如从$O(c^k)$改进为$O((c-\varepsilon)^k)$就会使得问题求解的算法在效率上获得指数级别的提高.多年来,研究人员不断改进SAT问题及其约束子问题的求解算法在最坏情况下的时间复杂度上界[6-11].以3SAT问题为例,最初Monien和Speckenmeyer在文献[6]中给出了求解3SAT问题的上界为$O(2^{m/3})$,其中m是命题公式中的子句数目,目前获得的最好上界为$O(1.234^m)$[8].SAT问题理论上的不断发展使得其系统的求解能力有了很大的提高.目前Zchaff、Survey Propagation等SAT求解系统已可以求解包含100 000个以上变量的SAT问题实例[12-14].

NAESAT(not-all-equal satisfiability)问题作为SAT问题的一个重要扩展,判断是否存在一组真值赋值使得给定的命题公式为可满足,并且使公式的每个子句中所有文字的取值不全为真.NAESAT和

收稿日期:2012-06-08. 网络出版日期:2012-11-16.
基金项目:国家自然科学基金资助项目(61070084,60803102);中央高校基本科研业务费专项资金资助项目(11QNJJ006);浙江师范大学计算机软件与理论省级重中之重学科开放基金资助项目(ZSDZZZZXK37).
通信作者:傅琳璐. E-mail:full820@nenu.edu.cn.

NAE-k-SAT(即公式中所有子句的长度都小于等于 k,$k\geqslant 3$)都是 NP 完全问题[15-17],其在集合分裂、最大割集等问题中都有着重要的应用[16]. 近来,从理论上研究 NAESAT 问题及其约束子问题在最坏情况下的时间复杂度上界受到了广泛的关注. Monien 等提出 NAESAT 问题可在 $O(1.587^m)$ 时间内求解,其中 m 为公式中的子句数目[18]. Riege 等将 NAE-SAT 问题算法在最坏情况下的时间复杂度上界改进为 $O(1.522\ 8^m)$ [19]. 2010 年,Shi 等则运用生物学方法提出了一种基于 DNA 的 NAE-3SAT 求解算法[20].

事实上,算法的时间复杂度是根据问题实例的大小计算所得. 而对于涉及命题公式的问题,问题实例的大小不仅依赖于公式中子句的数目,还依赖于变量的数目. 以变量数目为参数研究 NAESAT 问题及其约束子问题在最坏情况下的时间复杂度上界,不仅可以从另一个角度衡量算法的好坏,而且在某种程度上更能准确地反映出算法的性能. 本文正是以此为着眼点,从变量数目的角度探讨 NAE-3SAT 问题在最坏情况下的时间复杂度上界,提出了一个基于分支回溯(DPLL)的精确算法 NAE. 该算法给出了多种化简规则,证明该算法在最坏情况下的时间复杂度上界为 $O(1.618^n)$,其中 n 为公式中的变量数目.

1　基本概念

文中符号 $x,y,z\cdots$表示布尔变量;符号 $l,l_1,l_2\cdots$表示文字. 符号 $C,C_1,C_2\cdots$表示子句. 符号 $F,F_1,F_2\cdots$表示 CNF 公式.

定义 1　布尔变量. 布尔变量(简称变量)取值为{true, false}. 布尔变量 x 的度 $\varphi(x)$ 为变量 x 在公式中出现的次数. 把在公式中只出现一次的变量称为 singleton.

定义 2　真值赋值. 设变量集合 $X=\{x_1, x_2, \cdots, x_n\}$,定义在变量集合 X 上的真值赋值是一个函数 $\mu: X\rightarrow\{\text{true, false}\}$. 在变量集合 X 上存在 2^n 组不同的真值赋值. 文中把真值赋值简称为赋值.

定义 3　文字. 设 x 是一个变量,则称 x 和 $\neg x$ 为文字,其中称 x 是正文字,$\neg x$ 是负文字. 当且仅当变量 x 赋值为 true,正文字 x 在赋值 μ 下取真值;当且仅当变量 x 赋值为 false,文字 $\neg x$ 在赋值 μ 下取真值;当且仅当 F 中恰好包含 i 个文字 l,文字 l 称为 i-文字;当且仅当公式 F 中恰好包含 i 个文字 l 和 j 个文字 $\neg l$,文字 l 称为 (i,j)-文字. 公式中已经被赋值为 true(false)的文字称为 true(false)文字.

定义 4　子句. 子句由一组文字的析取而成,如 $C=l_1\vee l_2\vee\cdots\vee l_k$. 当且仅当 C 中至少有一个文字取值为 true,子句 C 在赋值 μ 下为可满足. 子句的长度是指子句中文字的个数,用符号 $|C|$ 表示. 把长度为 k 的子句称为 k-子句,长度为 0 的子句称为空子句,用 empty 表示. 空子句为不可满足.

定义 5　合取范式. 合取范式(conjunctive normal form, CNF)由一组子句的合取而成,如 $F=C_1\wedge C_2\wedge\cdots\wedge C_i$. CNF 范式也称为公式 F. 当且仅当 F 中每一个子句都为可满足,公式 F 在赋值 μ 下为可满足,空公式为可满足.

定义 6　NAESAT 问题. NAESAT 问题对于一个给定的命题公式 F,判断是否存在一组赋值使得公式 F 为可满足并且使公式 F 的每个子句中所有文字的取值不全为真,即要求公式 F 的每个子句中至少有一个文字为真,至少有一个文字为假. 如果存在这样的一组赋值,则称公式 F 为 NAE-可满足的,否则称公式 F 为 NAE-不可满足. 使得公式 F 为 NAE-可满足的赋值称为该公式的 NAE-可满足赋值.

如果给定的命题公式 F 中所有子句长度都小于等于 3,则判断是否存在一组赋值使得公式 F 为可满足并且使公式 F 的每个子句中所有文字的取值不全为真的问题称为 NAE-3SAT 问题.

另外需要说明的是,在文中符号 $F[x/\text{false}]$ 表示将公式 F 中的 x 赋值为 false,相应地 $\neg x$ 赋值为 true 后得到的子公式;符号 $F[x/\neg y]$ 表示替换 $x=\neg y$,即将公式 F 中的 x 用 $\neg y$ 替换,$\neg x$ 用 y 替换后得到的子公式;符号 $F[z/y, t/\neg y]$ 则表示替换 $z=y$ 和 $t=\neg y$,即同时将公式 F 中的 z 用 y 替换,t 用 $\neg y$ 替换,并相应地将 $\neg z$ 用 $\neg y$ 替换,$\neg t$ 用 y 替换后得到的子公式.

2　算法 NAE

2.1　化简规则

在求解 NAE-3SAT 问题时,使用化简规则对公式进行约简. 应用化简规则的目的是减少公式中的变量数目,缩小搜索空间,提高算法的效率.

规则 1:若公式 F 中存在 1-子句,则进行如下化简:

$$x\wedge F_1\rightarrow \text{empty}\wedge F_1.$$

规则 2:若公式 F 中存在 2 个相同的子句,则进行如下化简:

$$C_1\wedge C_1\wedge F_1\rightarrow C_1\wedge F_1.$$

规则 3:若公式 F 中存在包含重复变量的子句,则进行如下化简:

1) $(x \vee x) \wedge F_1 \rightarrow \text{empty} \wedge F_1$;

2) $(x \vee \neg x) \wedge F_1 \rightarrow F_1$;

3) $(x \vee x \vee x) \wedge F_1 \rightarrow \text{empty} \wedge F_1$;

4) $(x \vee x \vee \neg x) \wedge F_1 \rightarrow F_1$;

5) $(x \vee \neg x \vee y) \wedge F_1 \rightarrow F_1$;

6) $(x \vee x \vee y) \wedge F_1 \rightarrow F_1[y/\neg x]$;

7) $(\text{true} \vee x \vee x) \wedge F_1 \rightarrow F_1[x/\text{false}]$;

8) $(\text{false} \vee x \vee x) \wedge F_1 \rightarrow F_1[x/\text{true}]$;

9) $(\text{true} \vee x \vee \neg x) \wedge F_1 \rightarrow F_1$;

10) $(\text{false} \vee x \vee \neg x) \wedge F_1 \rightarrow F_1$.

规则4：若公式 F 中存在2-子句，则进行如下化简：

1) $(\text{false} \vee \text{false}) \wedge F_1 \rightarrow \text{empty} \wedge F_1$;

2) $(\text{true} \vee \text{true}) \wedge F_1 \rightarrow \text{empty} \wedge F_1$;

3) $(\text{false} \vee \text{true}) \wedge F_1 \rightarrow F_1$;

4) $(\text{false} \vee x) \wedge F_1 \rightarrow F_1[x/\text{true}]$;

5) $(\text{true} \vee x) \wedge F_1 \rightarrow F_1[x/\text{false}]$;

6) $(x \vee y) \wedge F_1 \rightarrow F_1[y/\neg x]$.

规则5：若公式 F 中存在包含2个或以上 true 或 false 文字的3-子句，则进行如下化简：

1) $(\text{false} \vee \text{false} \vee \text{false}) \wedge F_1 \rightarrow \text{empty} \wedge F_1$;

2) $(\text{true} \vee \text{true} \vee \text{true}) \wedge F_1 \rightarrow \text{empty} \wedge F_1$;

3) $(\text{false} \vee \text{true} \vee \text{false}) \wedge F_1 \rightarrow F_1$;

4) $(\text{false} \vee \text{true} \vee \text{true}) \wedge F_1 \rightarrow F_1$;

5) $(\text{true} \vee \text{true} \vee x) \wedge F_1 \rightarrow F_1[x/\text{false}]$;

6) $(\text{false} \vee \text{false} \vee x) \wedge F_1 \rightarrow F_1[x/\text{true}]$;

7) $(\text{false} \vee \text{true} \vee x) \wedge F_1 \rightarrow F_1$.

规则6：若公式 F 中存在包含1个 true 或 false 文字和 singleton 的3-子句，则进行如下化简：

1) $(\text{true} \vee x \vee y) \wedge F_1 \rightarrow F_1$;

2) $(\text{false} \vee x \vee y) \wedge F_1 \rightarrow F_1$.

其中 x 或 y 为 singleton 或者 x、y 是 singleton.

规则7：若公式 F 中存在包含2个或以上的 singleton 的3-子句，则进行如下化简：

$$(x \vee y \vee z) \wedge F_1 \rightarrow F_1.$$

其中 x、y、z 或者 x、y 是 singleton.

规则8：若公式 F 中存在2个子句，它们包含的变量都相同，则进行如下化简：

1) $(x \vee y \vee z) \wedge (x \vee y \vee \neg z) \wedge F_1 \rightarrow F_1[y/\neg x]$;

2) $(x \vee y \vee z) \wedge (x \vee \neg y \vee \neg z) \wedge F_1 \rightarrow F_1[y/\neg z]$;

3) $(x \vee y \vee z) \wedge (\neg x \vee \neg y \vee \neg z) \wedge F_1 \rightarrow (x \vee y \vee z) \wedge F_1$.

化简规则的执行时间均为多项式时间. 一旦某个规则的条件满足，该规则就可被执行，且所有规则按照顺序依次执行，即后面的规则只有在前面的规则都不适用时才会被执行. 待处理的公式 F 重复地被以上8条规则化简，直到所有规则的条件都不满足为止.

定理1 以上8条化简规则都是正确的，不会影响算法对公式 F 的 NAE-可满足性的判断.

证明 由 NAE-3SAT 问题的定义可得规则1～5是正确的. 在化简过程中，把 NAE-可满足的子句直接从公式中消去，把 NAE-不可满足的子句赋为空子句，即 empty.

规则6化简包含一个 true 或 false 文字和 singleton 的子句，因为对 singleton 的赋值不影响公式中其他变量的取值，可直接把 singleton 赋值为 false（规则6中1））或 true（规则6中2））使子句为 NAE-可满足. 因此可直接把该子句从公式中消去. 规则7与规则6类似.

在规则8的1）中，假设公式 F 包含2个子句为 $(x \vee y \vee z)$ 和 $(x \vee y \vee \neg z)$，如果 $y = x = \text{true}$ 或 $y = x = \text{false}$，则2个子句中肯定有一个是 NAE-不可满足，所以必须使 $y = \neg x$；规则8中2）与规则8中1）类似；规则8中3），2个子句的文字皆为互补，由 NAE-3SAT 问题的定义可得若存在一组赋值使得其中一个子句为 NAE-可满足，则该赋值肯定也能使另一个子句为 NAE-可满足，所以可消去其中一个子句. 因此规则8是正确的.

综上所述，8条化简规则都是正确的，化简过程不影响公式的可满足性，即若化简后的公式 $F' \in$ NAE-3SAT，则可判定原公式 $F \in$ NAE-3SAT.

化简后公式中将不存在1-子句、2-子句、包含重复变量的子句、包含2个或以上 true 或 false 文字的3-子句、包含1个 true 或 false 文字和 singleton 的3-子句、包含2个或以上的 singleton 的3-子句，且不会存在2个包含3个相同变量的3-子句.

2.2 NAE 算法框架

算法 NAE 的基本思想是给定任意的3-CNF 公式 F，首先利用2.1节中的化简规则对公式进行简化；然后在化简后的公式中选取某些变量进行分支，分别判断各个分支的 NAE-可满足性，进而判断出原公式的 NAE-可满足性；递归地执行这个过程直到公式为空或者判定公式为 NAE-不可满足后停止.

算法的输入是一个待处理的3-CNF 公式 F，输出为 true 或者 false，true 表示公式 F 为 NAE-可满足，false 表示公式 F 为 NAE-不可满足. 下面给出了算法的基本框架.

算法　NAE(F).

输入:A formula F in 3-CNF.

输出:If F is NAE-satisfiable, return true;

Else, return false.

1) Apply the rule 1 ~8 to F as long as one of them is applicable.

2) If F is empty, return true.

3) If F contains an empty clause, return false.

4) If F contains a clause with one true or false literal, C_1 = true $\vee x \vee y$ or C_1 = false $\vee x \vee y$, then return (NAE ($F[x/y]$) $\vee$ NAE($F[x/\neg y]$)).

5) If F contains two clauses $C_1 = (x \vee y \vee z)$, $C_2 = (x \vee y \vee t)$, then return (NAE($F[x/y]$) $\vee$ NAE ($F[x/\neg y]$)).

6) If F contains two clauses $C_1 = (x \vee y \vee z)$ and $C_2 = (x \vee \neg y \vee t)$, then return (NAE($F[x/y]$) $\vee$ NAE ($F[x/\neg y]$)).

7) If F contains two clauses $C_1 = (x \vee y \vee z)$ and $C_2 = (\neg x \vee \neg y \vee t)$, return(NAE($F[x/y]$) $\vee$ NAE ($F[x/\neg y]$)).

8) Else choose a clause $C_1 = (x \vee y \vee z)$, return (NAE($F[x/y]$) $\vee$ NAE($F[x/\neg y]$)).

9) End.

定理 2　算法 NAE 能正确地判断一个 3-CNF 公式是否为 NAE-可满足.

证明　下面的证明过程将对算法的每一行进行分析.

算法首先对公式 F 进行化简(第 1 步),当所有化简规则都不可用时,进行判断:因为在化简过程中将 NAE-可满足的子句直接从公式中消去,所以若化简后的公式 F 为空,则说明公式为 NAE-可满足,返回 true(第 2 步);因为在化简过程中将 NAE-不可满足的子句赋为空子句,所以若化简后公式 F 包含空子句,则说明公式为 NAE-不可满足,返回 false(第 3 步).

若化简后的公式 F 既不为空也不包含空子句,则算法进入分支过程(第 4 ~9 步). 分支过程根据化简后的公式分几种情况进行:第 4 行:考虑公式中存在 true 或 false 文字的情况. 由定理 1 可知包含该文字的子句一定为 C_1 = (true $\vee x \vee y$) 或 C_1 = (false $\vee x \vee y$),其中 x、y 为非 singleton. 不失一般性,假设子句 C_1 = (true $\vee x \vee y$) 并对 C_1 进行分支,分别判断 2 个子公式 $F[x/y]$ 和 $F[x/\neg y]$ 的 NAE-可满足性,只要其中一个为 NAE-可满足,则可说明原公式为 NAE-可满足;第 5 ~7 步:考虑公式中存在 2 个子句包含 2 个相同变量的 3 种情况. 对每一种情况,都进行分支,分别判断 2 个子公式 $F[x/y]$ 和 $F[x/\neg y]$ 的 NAE-可满足性;第 8 步:若以上情况都不存在,则公式中的任意 2 个子句间至多包含一个相同变量,算法任意选择公式中的一个子句进行分支,分别判断 2 个子公式 $F[x/y]$ 和 $F[x/\neg y]$ 的 NAE-可满足性.

综上所述,算法 NAE 是正确并且完备的,能正确判断一个给定的 3-CNF 公式是否为 NAE-可满足.

3　算法 NAE 时间复杂性分析

在本节中,用分支树的方法来分析算法 NAE 的时间复杂性,并证明了其在最坏情况下的时间复杂度上界为 $O(1.618^n)$,其中 n 为公式中的变量数目. 首先给出分支树的概念.

分支树是一个由 $i(i>0)$ 个结点组成的集合 T,是一棵层级结构树[21-22]. 分支树的每个结点标记一个 CNF 公式,其中一个特定的结点为根结点,标记为公式 F,除根结点外的其他结点被划分为 $j(j\geqslant 0)$ 个互不相交的有限集合 $T_1, T_2, \cdots, T_j$,每一个集合又都是分支树,称为根的子分支树,分别标记为公式 $F_1, F_2, \cdots, F_j$. 公式 $F_1, F_2, \cdots, F_j$ 是对公式 F 中的某些变量进行赋值后得到的公式,为公式 F 的子公式. 分支树的叶子结点标记的是空公式或者包含空子句的公式.

分支树中的每一个结点都有一个分支向量. 设分支树中某一个结点是 F_0,它的子结点分别是 $F_1, F_2, \cdots, F_k$,则结点 F_0 的分支向量为 $\tau(r_1, r_2, \cdots, r_k)$,其中 $r_i = f(F_0) - f(F_i)$,$f(F_i)$ $(0\leqslant i\leqslant k)$ 是公式 F_i 中变量的数目. 每个结点所对应的分支向量的值称为该结点的分支数,可用式(1)计算:

$$\sum_{i=1}^{k} x^{-r_i} = 1, \quad x > 0. \tag{1}$$

所有结点分支数中最大的被定义为分支树的分支数,用符号 $\tau_{\max}$ 表示.

定理 3[7]　设分支树 T 的根结点标记为公式 F,则分支树 T 中叶子的个数不超过 $(\tau_{\max})^n$,n 是公式 F 中变量的数目.

因为分支树的构造过程相当于基于分支回溯算法的执行过程,它们逐一地对公式 F 中的变量进行赋值,直到确定公式的可满足性为止. 所以可用分支树的方法对基于分支回溯的算法进行时间复杂性的分析. 假设算法在分支树 T 中任意一个结点执行的操作都是多项式时间,那么算法的执行时间 t 为

$$t \leqslant \#\{T_{\text{node}}\} \times \text{poly}(F_i) \leqslant$$
$$(2 \times (\#\{T_{\text{leaves}}\}) - 1) \times \text{poly}(F_i) \leqslant$$

$$(\tau_{\max})^n \times \mathrm{poly}(F_i). \quad (2)$$

式中:符号#{T_{node}}表示分支树 T 中结点的个数,符号#{T_{leaves}}表示分支树 T 中叶子结点的数目,符号 ploy(F_i)表示每个结点操作所用的时间是多项式时间.在本文中,忽略多项式时间.

定理4 算法 NAE 在最坏情况下的时间复杂度上界为 $O(1.618^n)$,其中 n 为公式中变量的数目.

证明 下面将对算法 NAE 的每一行进行时间复杂度分析.

第1~3步:多项式时间内时间复杂变为 $O(1)$.

第4步:不失一般性,假设子句 $C_1 = \mathrm{true} \vee x \vee y$.在分支 $F[x/y]$ 中 $C_1 = (\mathrm{true} \vee y \vee y)$,由化简规则3中7)可得,$y = \mathrm{false}$,因此消去 x、y 2个变量.在分支 $F[x/\neg y]$ 中 $C_1 = (\mathrm{true} \vee \neg y \vee y)$,由化简规则3中9)可得子句 C_1 可直接消去,因此消去 x 一个变量.执行时间为 $O(\tau(2,1)^n) = O(1.618^n)$.

第5步:在分支 $F[x/y]$ 中子句 $C_1 = (y \vee y \vee z)$,$C_2 = (y \vee y \vee t)$,由化简规则3中6)可得,$z = t = \neg y$,消去 x、z、t 3个变量.分支 $F[x/\neg y]$ 中 $C_1 = (\neg y \vee y \vee z)$,$C_2 = (\neg y \vee y \vee t)$,由化简规则3中5)可得子句 C_1、C_2 可直接消去,消去 x 一个变量.执行时间为 $O(\tau(3,1)^n) = O(1.466^n)$.

第6步:在分支 $F[x/y]$ 中由化简规则3中5)和6)可得,$z = \neg y$,消去 x、z 2个变量.同理在分支 $F[x/\neg y]$ 中可消去 x、t 2个变量.执行时间为 $O(\tau(2,2)^n) = O(1.414^n)$.

第7步:在分支 $F[x/y]$ 中由化简规则3中6)可得,$z = \neg y$,$t = y$,消去 x、z、t 3个变量.在分支 $F[x/\neg y]$ 中由化简规则3中5)可得子句 C_1、C_2 可直接消去,消去 x 一个变量.执行时间为 $O(\tau(3,1)^n) = O(1.466^n)$.

第8步:在分支 $F[x/y]$ 中由化简规则3中6)可得,$z = \neg y$,消去 x、z 2个变量.在分支 $F[x/\neg y]$ 中由化简规则3中5)可得子句 C_1 可直接消去,消去 x 一个变量.执行时间为 $O(\tau(2,1)^n) = O(1.618^n)$.

由以上分析可得,算法 NAE 最坏情况下的时间复杂度上界为 $O(1.618^n)$.

4 结束语

本文从变量规模的角度探讨了 NAE-3SAT 问题在最坏情况下的时间上界.针对 NAE-3SAT 问题新提出的8条化简规则有效地提高了算法的时间效率.对于算法时间复杂性的分析,本文采用的是标准测度.近年来,众多学者也在采用非标准测度来分析算法的时间复杂性,如 measure and conquer 方法.后续的工作将考虑利用非标准测度来分析算法的时间复杂性.

参考文献:

[1] COLMERAUER A. Opening the Prolog III universe[J]. Byte Magazine, 1987, 12(9): 177-182.

[2] KLEER J. Exploiting locality in a TMS[C]//Proceedings of the 8th National Conference on Artificial Intelligence. Boston, USA, 1990: 264-275.

[3] GU J. Local search for satisfiability (SAT) problem[J]. IEEE Transactions on Systems, Man and Cybernetics, 1993, 23(4): 1108-1129.

[4] 李未, 黄文奇. 一种求解合取范式可满足性问题的数学物理方法[J]. 中国科学: A 辑, 1994, 24(11): 1208-1217.
LI Wei, HUANG Wenqi. A mathematic physical approach to the satisfiability problems[J]. Science in China: Series A, 1994, 24(11): 1208-1217.

[5] COOK S A. The complexity of the theorem-proving procedures[C]//Proceedings of the Third Annual ACM Symposium on Theory of Computing. New York, USA, 1971: 151-158.

[6] MONIEN B, SPECKENMEYER E. Upper bounds for covering problems[R]. Universitat-Gesamtho Chschule -Paderborn: Reihe Theoretische Informatik, 1980.

[7] HIRSCH E A. New worst-case upper bounds for SAT[J]. Journal of Automated Reasoning, 2000, 24(4): 397-420.

[8] YAMAMOTO M. An improved $O(1.234^m)$-time deterministic algorithm for SAT[J]. Lecture Notes in Computer Science, 2005, 3827: 644-653.

[9] MONIEN B, SPECKENMEYER E. Solving satisfiability in less than $2n$ steps[J]. Discrete Applied Mathematics, 1985, 10(3): 287-295.

[10] BRUEGGEMANN T, KERN W. An improved local search algorithm for 3-SAT[J]. Theoretical Computer Science, 2004, 329(1/2/3): 303-313.

[11] DANTSIN E, GOERDT A, HIRSCH E A, et al. Adeterministic $(2-2/(k+1))n$ algorithm for k-SAT based on local search[J]. Theoretical Computer Science, 2002, 289(1): 69-83.

[12] ZHANG Lintao, MADIGN C F, MOSKEWICZ M H, et al. Efficient conflict driven learning in a Boolean satisfiability solver[C]//Proc of the ACM/IEEE ICCAD 2001. New York, USA, 2001: 279-285.

[13] ZECCHINA R, MEZARD M, PARISI G. Analytic and algorithmic solution of random satisfiability problems[J]. Science, 2002, 297(5582): 812-815.

[14] ZECCHINA R, MONASSON R, KIRKPATRICK S, et al.

Determining computational complexity from characteristic phase transitions[J]. Nature, 1999, 400(6740): 133-137.

[15] THOMAS J S. The complexity of satisfiability problems [C]//Conference Record of the Tenth Annual ACM Symposium on Theory of Computing. New York, USA, 1978: 216-226.

[16] PORSCHEN S, RANDERATH B, SPECKENMEYER E. Linear time algorithms for some not-all-equal satisfiability problems[C]//Proceedings of the 6th International Conference on Theory and Application of Satisfiability Testing(SAT2003). Santa Margherita Ligure, Italy, 2003: 172-187.

[17] WALSH T. The interface between P and NP: COL, XOR, NAE, 1-in-k, and Horn SAT[C]//Eighteenth National Conference on Artificial Intelligencet. Edmonton, Canada, 2002: 685-700.

[18] MONIEN B, SPECKENMEYER E, VORNBERGER O. Upper bounds for covering problems[J]. Methods of Operations Research, 1981, 43: 419-431.

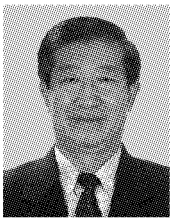
[19] RIEGE T, ROTHE J, SPAKOWSKI H. An improved exact algorithm for the domatic number problem[J]. Information Processing Letters, 2006, 101(3): 101-106.

[20] SHI Nungyue, CHU Chihping. A DNA-based algorithm for the solution of not-all-equal 3-SAT problem[C]//ASE International Conference on Information Engineering. Taiyuan, China, 2009: 94-97.

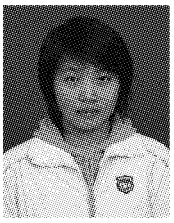
[21] ZHOU Junping, YIN Minghao, ZHOU Chunguang. New worst-case upper bound for #2-SAT and #3-SAT with the number of clauses as the parameter[C]//Proceedings of 24rd AAAI Conference on Artificial Intelligence. Atlanta, USA, 2010: 217-222.

[22] 周俊萍, 殷明浩, 周春光, 等. 最坏情况下#3-SAT 问题最小上界[J]. 计算机研究与发展, 2011, 48(11): 2055-2063.
ZHOU Junping, YIN Minghao, ZHOU Chunguang, et al. The worst case minimized upper bound in #3-SAT[J]. Journal of Computer Research and Development, 2011, 48(11): 2055-2063.

作者简介:



谷文祥,男,1947 年生,教授,博士生导师,主要研究方向为智能规划与规划识别、形式语言与自动机、模糊数学及其应用. 主持国家自然科学基金项目 3 项,发表学术论文 100 余篇.



傅琳璐,女,1988 年生,硕士研究生,主要研究方向为自动推理和智能规划.

周俊萍,女,1981 年生,讲师,博士,主要研究方向为自动推理和智能规划,主持东北师范大学青年基金项目 1 项,发表学术论文 10 篇.