DOI:10.3969/j.issn.1673-4785.201109003
网络出版地址:http://www.cnki.net/kcms/detail/23.1538.TP.20120510.1619.001.html

最坏情况下 Min-2SAT 问题的上界

谷文祥[1,2],姜蕴晖[1],周俊萍[1],殷明浩[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:最坏情况下 MaxSAT 问题上界的研究已成为一个热门的研究领域.与 MaxSAT 问题相对的是 MinSAT 问题,在求解某些组合优化问题时,将其转化为 MinSAT 问题比转化为 MaxSAT 问题有着更快的速度,因此对 MinSAT 问题进行研究.针对 Min-2SAT 问题提出算法 MinSATAlg,该算法首先利用化简算法 Simplify 对公式进行化简,然后通过分支树的方法对不同情况的子句进行分支.从子句数目的角度分析算法的时间复杂度并证明 Min-2SAT 问题可在 $O(1.134\,3^m)$ 时间内求解,对于每个变量至多出现在3个2-子句中的情况,得到最坏情况下的上界为 $O(1.122\,5^n)$,其中 n 为变量的数目.

关键词:MaxSAT;MinSAT;Min-2SAT;MaxSAT 问题的上界;Min-2SAT 问题的上界;子句数目;分支树
中图分类号:TP301 **文献标志码**:A **文章编号**:1673-4785(2012)03-0241-05

New worst-case upper bounds for Min-2SAT problems

GU Wenxiang[1,2], JIANG Yunhui[1], ZHOU Junping[1], YIN Minghao[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 rigorous theoretical analyses of algorithms for solving the upper bounds of maximum satisfiability (MaxSAT) problems have been proposed in research literature. The opposite problem of MaxSAT is the minimum satisfiability (MinSAT) problem. Solving some combinatorial optimization problems by reducing them to MinSAT form may be substantially faster than reducing them to MaxSAT form, so the MinSAT problem was researched in this paper. An algorithm (MinSATAlg) was presented for the minimum two-satisfiability (Min-2SAT) problem. In this paper, first, the simplification algorithm Simplify was used for simplification of formulas. Secondly, the branching tree method was used for branching on different kinds of clauses. It was proven that this algorithm can solve the Min-2SAT problem in $O(1.134\,3^m)$, regarding the number of clauses as parameters. The upper bound in the worst case was derived as $O(1.122\,5^n)$, where n is the number of variables in an input formula for a particular case of Min-2SAT in which each variable appears in three 2-clauses at most.

Keywords: maximum satisfiability; minimum satisfiability; minimum two-satisfiability; upper bounds for maximum satisfiability; upper bounds for minimum two-satisfiability; number of clauses; branching tree

人工智能研究领域存在着很多计算困难的问题,如 SAT(satisfiability)问题、QBF(quantified boolean formula)问题、智能规划问题、模型诊断问题等.若 P≠NP 成立,人们将无法为这些问题找到多项式时间的求解算法.这时从理论上分析这些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.以 SAT 问题为例,作为第一个被证明是 NP 完全的问题,改进其在最坏情况下的时间上界受到了研究人员的广泛关注.MaxSAT(maximum satisfiability)问题是 SAT 问题的扩展,它

收稿日期:2011-09-06. **网络出版日期**:2012-05-10.
基金项目:国家自然科学基金资助项目(61070084);国家自然科学青年基金资助项目(60803102);中央高校基本科研业务费专项资金资助项目(11QNJJ006).
通信作者:姜蕴晖. E-mail: jiangyh215@nenu.edu.cn.

指的是给定一个命题逻辑公式,找到一组使真值赋值同时满足最多的子句. 与 SAT 问题一样,MaxSAT 问题在计算机科学领域有着十分重要的地位,因为它是求解人工智能和组合优化问题的基础[1-2]. 当公式中子句的长度至多为 2 时称之为 Max-2SAT (maximum two-satisfiability)问题,它是一个 NP 完全的问题[3]. 近年来,众多学者对 Max-2SAT 问题进行了研究[4-7],已经将其最坏情况下的上界缩小到 $O(2^{m/6.312})$[8]. 与 MaxSAT 问题相对的是 MinSAT (minimum satisfiability)问题,它指的是给定一个命题逻辑公式,找到一组使真值赋值同时满足最少的子句. 虽然人们对 MaxSAT 问题已经做了非常多的研究,但对 MinSAT 问题的研究却并不深入. 目前,对 MinSAT 问题的求解主要采用近似方法[9-10]. 此外,LI 等在文献[11]中将 MinSAT 问题转换为 Max-SAT 问题,给出了一个 MinSAT 求解器. 事实上,在求解某些组合优化问题时,将其转化为 MinSAT 问题比转化为 MaxSAT 问题有着更快的速度,所以研究 MinSAT 问题也有着十分重要的理论意义和实际应用价值. 正如人们对 MaxSAT 问题的研究主要集中在 Max-2SAT 问题上,笔者同样着眼于 MinSAT 问题中子句长度不超过2 的情况,即对 Min-2SAT 问题进行研究,提出了一种求解 Min-2SAT 问题的算法,并证明了算法在 $O(1.134\ 3^m)$时间内可解.

1 基础知识

1.1 基本概念

为了讨论方便,首先介绍本文需要的相关概念.

定义 1 真值赋值. 给定一个布尔变量集合$V=\{x_1,x_2,\cdots,x_n\}$,定义在 V 上的真值赋值是一个函数 $\mu:V\to\{\text{true},\text{false}\}$. 每个真值赋值可以用一个 n 元布尔向量表示. 对 V 中任意一个布尔变量 x_i,若它在真值赋值 μ 下取真,则 $\mu(x_i)=1$,否则 $\mu(x_i)=0$.

定义 2 文字. 对任意一个布尔变量 x,称符号 x 和 $\neg x$ 是其文字,其中 x 是正文字,$\neg x$ 是负文字.

定义 3 纯文字. 给定一个公式 F 和 F 中任意一个布尔变量 x,若 x 只以正文字或负文字的形式出现在公式 F 中,则称 x 为纯文字.

定义 4 子句. 子句是若干文字的析取,用集合 C 表示,$C=l_1\vee l_2\vee\cdots\vee l_k$,其中 $l_1,l_2,\cdots,l_k$ 是文字. 子句 C 中文字的个数称为子句的长度,记作$|C|$.

k-子句是指子句长度为 k 的子句. 永真子句 T 指的是对子句的变量任意赋值时子句都是可满足的.

定义 5 CNF 范式. CNF 范式是若干子句的合取,用 F 表示,$F=C_1\wedge C_2\wedge\cdots\wedge C_i$,其中 $C_1,C_2,\cdots,C_i$ 是子句. CNF 范式 F 也称为公式,当且仅当每一个子句都可满足时,公式 F 在赋值 μ 下可满足.

公式 F 同时可满足的最少的子句数记为 PesVal(F). 称文字 l 出现在子句中如果子句包含 l,称变量 x 出现在子句中如果子句包含 x 或 $\neg x$. 本文中,用#(F, l)表示文字 l 出现在公式 F 中的次数,用 $\#_k(F,l)$表示文字 l 在公式 F 中的 k-子句中出现的次数.

用 $F[l]$表示将 F 中所有包含 l 的子句替换为 T,消去所有 $\neg l$ 和所有 F 中的空子句. 用 $F[l_1=l_2]$ 表示将所有的 l_1 用 l_2 替代,并将所有的 $\neg l_1$ 用 $\neg l_2$ 替代.

定义 6 Min-2SAT 问题. 给定一个命题逻辑公式 F,找到一组使真值赋值满足最少的子句,如果子句的长度至多为 2,则称其为 Min-2SAT 问题.

定义 7 变量的度. 变量 x 的度是指包含 x 的 2-子句的个数,也就是说,如果变量 x 的度为 k,则 $\#_2(F,x)+\#_2(F,\neg x)=k$.

定义 8 变量的邻居. 变量 x 的邻居是指所有和 x 一起出现在同一个 2-子句的变量.

用 $V(F)$表示 F 中所有变量的集合,对于变量集合 $V_0\subset V(F)$,用 $N(F,V_0)$来表示变量集合 V_0 的所有邻居.

用 $G(F)$表示一个无向图,$V(F)$是所有顶点的集合,如果 F 中 2 个变量出现在同一个 2-子句中,则在这 2 个顶点之间添加 1 条边.

1.2 复杂性分析方法

本节介绍分支算法的复杂性分析方法,首先给出分支树的概念[12].

分支树是由 $i(i>0)$个结点组成的有限集合 Q,其中一个特定的结点为根结点,标记为公式 F,除根结点以外的其他结点被划分为 $j(j\geqslant 0)$个互不相交的有限集合 $Q_1,Q_2,\cdots,Q_j$,分别标记为公式 $F_1,F_2,\cdots,F_j$. 其中每一个集合又都是分支树,称为根结点的子分支树,分别表示对公式 F 中的某些变量进行赋值后得到的公式,即公式 F 的子公式. 叶子结点标记的公式为空公式或者其中存在一个空子句.

分支树中的每一个结点都有 1 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 $r_i=f(F_0)-f(F_i)$,$f(F_0)=m(F_0)$是公式 F_0 中子句的数目,$f(F_0)=n(F_0)$是公式 F_0 中变量的数目.

每个结点的分支向量的值被称为结点的分支数,可用式(1)计算.

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

定理 1[12]　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 $(\tau_{\max})^{f(F)}$,其中 $f(F)$ 是公式 F 中子句的数目或变量的数目.

由分支树的定义可以看出,分支树的构造过程相当于基于 DPLL(Davis-Putnam-Logemann-Loveland)算法的执行过程. 算法逐一对公式 F 中的变量进行赋值,直到确定任意一个赋值满足或不满足公式 F 为止. 假设基于 DPLL 的算法在分支树 T 中任意一个结点执行的操作都是多项式的,那么从子句数目的角度考虑,算法的执行时间 t 为

$$t \leqslant \#\{T_{\text{node}}\} \times \text{ploy}(F_i) \leqslant (2 \times (\#\{T_{\text{leaves}}\}) - 1) \times \text{ploy}(F_i) \leqslant (\tau_{\max})^m \times \text{ploy}(F_i).$$

若从变量数目的角度考虑,算法的执行时间 t 为

$$t \leqslant \#\{T_{\text{node}}\} \times \text{ploy}(F_i) \leqslant (2 \times (\#\{T_{\text{leaves}}\}) - 1) \times \text{ploy}(F_i) \leqslant (\tau_{\max})^n \times \text{ploy}(F_i).$$

用 F 表示 2-CNF 公式,用 $N_i(F)$ 表示在公式 F 的 2-子句中出现 i 次的变量的个数. 很容易得到

$$K_2(F) = \sum_{i=1}^{N(F)} \frac{i \times N_i(F)}{2}.$$

式中:$K_2(F)$ 表示 F 中 2-子句的数目. 根据文献[13]可以对其稍作修改,得到新的复杂性测度为

$$r(F) = N_3(F) + 1.9N_4(F) + \sum_{i=5}^{N(F)} \frac{i \times N_i(F)}{2}.$$

2　化简规则

在给出求解 Min-2SAT 问题的算法之前,首先给出一个化简算法如下.

化简算法 Simplify(F).

输入:一个 2-CNF 范式 F.

输出:一个化简后的 2-CNF 范式 F.

1)如果 $F = F_0 \cup \{C, D\}$,并且对于一个文字 l 有 $C \backslash \{l\} = D \backslash \{\neg l\}$,则返回 Simplipy($F_0 \cup \{C \backslash \{l\}, T\}$).

2)如果 F 中存在一个文字 l 满足 $\#(F, \neg l) = 0$,则返回 Simplipy($F[\neg l]$).

3)如果 F 中存在一个文字 l 满足 $\#_1(F, l) \geqslant \#(F, \neg l)$,则返回 Simplipy($F[\neg l]$).

4)如果 F 中存在 2 个变量 x_1 和 x_2,并且 x_1 至多出现在一个不含 x_2 的 2-字句中,令 α 和 β 分别为 $F[x_2]$ 和 $F[\neg x_2]$ 中通过化简规则对 x_1 所赋的布尔值(true 或者 false),则根据 α 和 β 的值将公式 F 化简的方法为:

a)如果 α = false, β = false,则返回 Simplipy($F[\neg x_1]$).

b)如果 α = false, β = true,则返回 Simplipy($F[x_1 = \neg x_2]$).

c)如果 α = true, β = false,则返回 Simplipy($F[x_1 = x_2]$).

d)如果 α = true, β = true,则返回 Simplipy($F[x_1]$).

对于给定的范式,重复使用化简算法 Simplify(F)对其进行化简,直到范式不能再应用算法中任何一条化简规则为止. 这样在求解 Min-2SAT 问题时可以减少算法需要考虑的情况,以减弱算法的复杂性.

根据化简规则,可以得到下面的引理.

引理 1　令 F 是一个 2-CNF 范式, F' = Simplify(F),则 F'中不包含度至多为 2 的变量.

证明　若变量的度为 1,则可通过化简算法 Simplify(F)中化简规则 2)对其化简;若变量的度为 2,则可通过化简规则 4)进行化简. 因此公式化简后只包含度至少为 3 的变量.

引理 2　令 F 是化简后的公式,a、x 是邻居,a 的度为 3,则在 $F[x]$和 $F[\neg x]$中,至少有一个化简规则会对 a 赋值.

证明　考虑变量 a 所有可能出现的情况.

1)当$(a \vee x) \wedge (a \vee x_1) \wedge (\neg a \vee x_2)$时,若 x = 0,则由化简规则 3)可知 a = 1.

2)当$(a \vee x) \wedge (\neg a \vee x_1) \wedge (\neg a \vee x_2)$时,若 x = 1,则由化简规则 2)可知 a = 0.

3)当$(a \vee x) \wedge (a \vee x_1) \wedge (a \vee x_2) \wedge \neg a$时,若 x = 0,则由化简规则 1)可知消除了 a 和$\neg a$,再由化简规则 2)可知 a = 1.

4)当$(a \vee x) \wedge (a \vee x_1) \wedge (a \vee x_2) \wedge \neg a \wedge \neg a$时,若 x = 1,则由化简规则 3)可知 a = 0.

5)当$(a \vee x) \wedge (a \vee x_1) \wedge (\neg a \vee x_2) \wedge \neg a$时,若 x = 1,则由化简规则 3)可知 a = 0.

6)当$(a \vee x) \wedge (\neg a \vee x_1) \wedge (\neg a \vee x_2) \wedge a$时,若$x$ = 0,则由化简规则 3)可知 a = 1.

由于公式 F 已化简,所以不存在其他情况,引理得证.

3　算法 MinSATAlg

本文给出一个求解 Min-2SAT 问题的算法 MinSATAlg,它是一个典型的分支算法. 具体算法描述如下.

输入:一个 2-CNF 范式 F.

输出:PesVal(F).

1) F = Simplify(F).

2)如果 F 只包含 T 字句,则返回 $L(F)$.

3)如果 $F=F_1\cup F_2$,并且 F_1 和 F_2 没有相同的变量,则返回 MinSATAlg(F_1) + MinSATAlg(F_2).

4)选择变量 x 使 $\tau(r(F)-r(\text{Simplify}(F[x])),r(F)-r(\text{Simplify}(F[\neg x])))$最小,并返回min(MinSATAlg($F[x]$),MinSATAlg ($F[\neg x]$)).

在算法 MinSATAlg 中,首先,应用化简规则对公式进行化简;第2步是指如果公式中只包含永真子句 T,则返回公式中子句的数目;第3步考虑公式中包含组件的情况,即将公式分支成2个更简单的公式并对其递归调用;最后根据每次递归调用返回的结果得到最后的结果. 定理1中给出了算法的时间复杂度.

定理 2 给定一个 2-CNF 范式,算法 MinSATAlg 在$O(1.134\ 3^m)$时间内返回公式中同时可以满足的最少的子句数.

证明

1)当公式 F 中包含一个度大于6的变量时. 由化简规则可知,F 中的所有变量至少出现在2个没有 x 出现的2-子句中. 度至少为3的变量使 r 至少减少1/2,这是因为 N_i 的2个系数的差至少为1/2. 这样,当对 x 赋值后,r 至少减少$w(w/2(x$ 被消去) $+w/2(x$ 的邻居的度减少)). 因此,对 x 进行分支的复杂度至少为 $O(\tau(6,6)^m)=O(1.122\ 5^m)$.

2)当公式 F 中变量的度都不超过5时. 令 x 是一个度为5的变量,用 k_{ij} 表示在公式 F 中出现 j 次且和 x 一起出现 i 次的变量个数,其中 $1\leqslant i\leqslant j\leqslant 5$. 由于 F 已化简,所以当 $j\leqslant 2$ 或 $j-i\leqslant 1$ 时 $k_{ij}=0$. 由于 x 出现在5个2-子句中,所以有

$$k_{13}+k_{14}+k_{15}+2k_{24}+2k_{25}+3k_{35}=5.$$

对于共出现 j 次并且与 x 一起出现 i 次的变量,在对 x 赋值后其出现在2-子句中的次数为 $j-i$. 令 F' 为对 F 中的变量 x 赋值后得到的公式,则有

$$r(F)-r(F')=(k_{13}+1.9(k_{14}+k_{24})+2.5(k_{15}+k_{25}+k_{35}+1))-(k_{14}+k_{25}+1.9k_{15})\geqslant 0.6(k_{13}+k_{14}+k_{15}+2k_{24}+2k_{25}+3k_{35})+2.5=0.6\times 5+2.5=5.5.$$

由此可见,对 x 进行分支的复杂度至少为 $O(\tau(5.5,5.5)^m)=O(1.134\ 3^m)$.

3)当公式 F 中变量的度都不超过4时. 与第2种情况类似可得到

$$k_{13}+k_{14}+2k_{24}=4,$$

$$r(F)-r(F')=(k_{13}+1.9(k_{14}+k_{24}+1))-(k_{14})\geqslant 0.9(k_{13}+k_{14}+2k_{24})+1.9=0.9\times 4+1.9=5.5.$$

因此,对 x 进行分支的复杂度至少为 $O(\tau(5.5,5.5)^m)=O(1.134\ 3^m)$.

4)当公式 F 中所有变量的度均为3时. 在这种情况下很容易看出 $r(F)=N(F)$,所以只需要证明以变量数目为参数时的复杂度为 $O(\tau(6,6)^n)=O(1.122\ 5^n)$. 由于变量的度均为3,所以变量的个数必为偶数,$F$ 的每一次分支都为$(2k,2l)$,其中 k 和 l 都是整数.

a)如果公式 F 中包含3个变量且其中任意2个变量都出现在同一个2-子句中,则在图 $G(F)$ 中形成1个三角形. 令 a、b、c 为形成三角形的3个变量,可以看出 $N(\{a,b,c\})$ 包含至少2个变量,至少1个邻居只和$\{a,b,c\}$中的1个一起出现,将其标记为 d 并和 c 一起出现,如图1(a)所示. 对 d 进行分支的复杂度为 $O(\tau(6,6)^n)=O(1.122\ 5^n)$. 这是因为对 d 赋值后,c 只和 a、b 一起出现,或出现在单元子句中,化简规则或者对 c 直接赋值,或者用包含 a、b 的子句替代含 c 的子句. 对于前一种情况,a、b 也会通过化简规则被消去,对于后一种情况通过化简规则4)会消去 a、b 中的一个. 所以对 d 赋值后还会至少再消去4个变量,因此对 d 进行分支的复杂度为$O(\tau(6,6)^n)=O(1.122\ 5^n)$.

b)考虑图 $G(F)$ 中不包含三角形的情况. 假设 F 包含一个变量 x,其邻居为 a、b、c,这样在 $F[x]$ 和 $F[\neg x]$ 中化简规则会对 a、b、c 中至少一个赋值. 很容易看出对 x 进行分支的复杂度为 $O(\tau(6,6)^n)=O(1.122\ 5^n)$.

c)下面需要考虑的是对 F 中的一个变量进行分支时,化简规则会在一个分支中对其全部邻居赋值的情况. 考虑一个变量 x,$N(\{x\})=\{a,b,c\}$. 如果 $|N(\{a,b,c\})|\geqslant 5$,则对 x 分支的复杂度为 $O(\tau(4,10)^n)\subset O(1.112\ 0^n)$. 如果$|N(\{a,b,c\})|<5$,这就意味着或者存在一个变量 $y\neq x$ 并且$N(y)=\{a,b,c\}$(如图1(b)所示),或者存在2个变量$y,z\neq x$并且 $|N(y)\cup\{a,b,c\}|=2$,$|N(z)\cup\{a,b,c\}|=2$(如图1(c)所示). 前一种情况对 b 分支的复杂度为 $O(\tau(6,6)^n)=O(1.122\ 5^n)$,后一种情况对 x 赋值的复杂度为 $O(\tau(4,10)^n)\subset O(1.112\ 0^n)$.

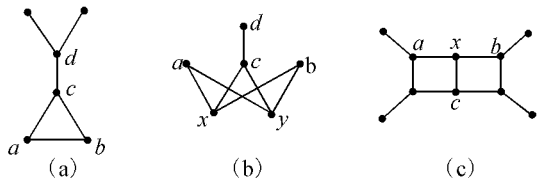


图1 情况4)的不同例子

Fig. 1 Different cases in 4)

由此可见,以变量数目为参数时的复杂度为 $O(\tau(6,6)^n)=O(1.122\ 5^n)$. 从而得到一个化简后的公式 F 只包含度为3的变量时,则对其中一个变量进行分支的时间复杂度为 $O(\tau(6,6)^m)=$

$O(1.122\,5^m)$.

综上所述,给定一个 2-CNF 范式,算法 MinSATAlg 在 $O(1.134\,3^m)$ 时间内返回公式中同时可以满足的最少的子句数,即定理 2 得证.

4　结束语

MaxSAT 问题是求解人工智能和组合优化问题的基础,而在求解某些组合优化问题时,将其转化为 MinSAT 问题比 MaxSAT 问题有着更快的速度,因此求解 MinSAT 问题并分析其复杂度有着很高的实际价值.本文研究了 MinSAT 问题中每个子句长度不大于 2 的情况,即 Min-2SAT 问题,提出了一个求解 Min-2SAT 问题的算法并证明了算法的时间复杂度为 $O(1.134\,3^m)$,其中 m 为公式中子句的数目.笔者运用了一种新的复杂性测度来测量算法的运行时间.另外还证明了对于每个变量至多出现在 3 个 2-子句中的情况,其最坏情况下的时间复杂度为 $O(1.122\,5^n)$,其中 n 为变量的数目.

在今后的工作中,可以进一步修正复杂性测度或通过加入冲突子句等方法,来改善算法并减少算法的时间复杂度.

参考文献:

[1] HANSEN P, JAUMARD B. Algorithms for the maximum satisfiability problem[J]. Computing, 1990, 44(4): 279-303.

[2] WALLACE R J. Enhancing maximum satisfiability algorithms with pure literal strategies[C]//Proceedings of the 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence. London, UK: Springer-Verlag, 1996: 388-401.

[3] NIEDERMEIER R, ROSSMANITH P. New upper bounds for MaxSat[C]//Proceedings of the 26th International Colloquium on Automata, Languages and Programming. London, UK: Springer-Verlag, 1999: 575-584.

[4] GRAMM J, HIRSCH E A, NIEDERMEIER R, et al. Worst-case upper bounds for MAX-2-SAT with an application to MAX-CUT[J]. Discrete Applied Mathematics, 2003, 130(2): 139-155.

[5] KNEIS J, ROSSMANITH P. A new satisfiability algorithm with applications to Max-Cut: Technical Report AIB2005-08[R]. Aachen, Germany: Department of Computer Science, RWTH Aachen University, 2005.

[6] KULIKOV A S, KUTZKOV K. New bounds for MAX-SAT by clause learning[C]//2nd International Symposium on Computer Science in Russia. Ekaterinburg, Russia, 2007: 194-204.

[7] BINKELE-RAIBLE D, FERNAU H. A new upper bound for Max-2-SAT: a graph-theoretic approach[J]. Journal of Discrete Algorithms, 2010, 8(4): 388-401.

[8] GASPERS S, SORKIN G B. A universally fastest algorithm for Max 2-Sat, Max 2-CSP, and everything in between[J]. Journal of Computer and System Sciences, 2012, 78(1): 305-335.

[9] AVIDOR A, ZWICK U. Approximating MIN 2-SAT and MIN 3-SAT[J]. Theory of Computing Systems, 2005, 38(3): 329-345.

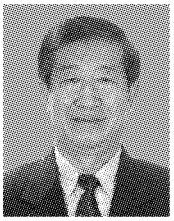
[10] MARATHE M V, RAVI S S. On approximation algorithms for the minimum satisfiability problem[J]. Information Processing Letters, 1996, 58(1): 23-29.

[11] LI Chumin, MANYA F, QUAN Zhe, et al. Exact MinSAT solving[C]//Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing. Berlin/Heidelberg, Germany: Springer-Verlag, 2010: 363-368.

[12] 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 the Twenty-Fourth AAAI Conference on Artificial Intelligence. Atlanta, USA, 2010.

[13] KOJEVNIKOV A, KULIKOV A S. A new approach to proving upper bounds for MAX-2-SAT[C]//Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithms. New York, USA: ACM, 2006: 11-17.

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.主持或参与国家自然科学基金项目 5 项、教育部重点项目 2 项、省科委项目 1 项.发表学术论文 130 余篇,出版专著《智能规划与规划识别》,2011 年获得吉林省专著类一等奖.

姜蕴晖,女,1987 年生,硕士研究生,主要研究方向为智能规划与自动推理.

周俊萍,女,1981 年生,讲师,主要研究方向为智能规划与自动推理,发表学术论文 5 篇.