软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2015,26(9):2155−2166 [doi: 10.13328/j.cnki.jos.004799]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于聚类和划分的 SAT 分治判定*

范全润[1], 段振华[1,2]

[1](西安电子科技大学 计算理论与技术研究所,陕西 西安 710071)

[2](ISN 国家重点实验室(西安电子科技大学),陕西 西安 710071)

通讯作者: 段振华, E-mail: zhduan@mail.xidian.edu.cn

摘 要: 提出了一种将布尔公式划分为子句组来进行布尔可满足性判定的方法.CNF(conjunctive normal form)公式是可满足的当且仅当划分产生的每个子句组都是可满足的,因此,通过判定子句组的可满足性来判定原公式的可满足性,相当于用分治法将复杂问题分解为多个子问题来求解.这种分治判定方法一方面降低了原公式的可满足性判定复杂度;另一方面,由于子句组的判定可以并行,因而判定速度能够得到进一步的提高.对于不能直接产生布尔子句组划分的情形,提出了一种利用聚类技术将 CNF 公式聚类成多个簇,然后消去簇间的公共变量来产生子句组划分的方法.

关键词: 合取范式;布尔可满足性;划分;聚类

中图法分类号: TP301

中文引用格式: 范全润,段振华.基于聚类和划分的 SAT 分治判定.软件学报,2015,26(9):2155−2166. http://www.jos.org.cn/1000-9825/4799.htm

英文引用格式: Fan QR, Duan ZH. Clustering and partition based divide and conquer for SAT solving. Ruan Jian Xue Bao/Journal of Software, 2015,26(9):2155−2166 (in Chinese). http://www.jos.org.cn/1000-9825/4799.htm

Clustering and Partition Based Divide and Conquer for SAT Solving

FAN Quan-Run[1], DUAN Zhen-Hua[1,2]

[1](Institute of Computing Theory and Technology, Xidian University, Xi'an 710071, China)

[2](State Key Laboratory of Integrated Services Networks (Xidian University), Xi'an 710071, China)

Abstract: A partition based Boolean satisfiability solving method is proposed. By partitioning a CNF(conjunctive normal form) formula into several clause groups, Boolean satisfiability problem can be divided into small sub-problems, hence reducing the complexity of the original problem. Meanwhile, the satisfiability of different clause group can be solved in parallel, thus further speeding up the decision procedure. For the formula that clause group partition cannot be generated directly, a clustering algorithm is given to cluster clauses into clusters so that clause group partition can be generated by eliminating common variables among clusters.

Key words: conjunctive normal form; Boolean satisfiability; partition; clustering

布尔可满足性问题(propositional satisfiability,简称 SAT)是首个被证明的 NP-Complete(NP-C or NPC)问题[1].给定一个布尔命题公式,SAT 判定就是要找到一组变量的赋值使得该命题公式为真(即,该公式可满足),或者证明该命题公式不可能为真(即,该公式不可满足).SAT 判定技术可以应用到人工智能规划(AI planning)、模型分析(model analysis)、定理证明、软件验证、电子设计自动化及验证等领域.近年来,SAT 判定技术取得了很大的进展,研究人员开发了很多高效的 SAT 判定工具,进一步拓展了 SAT 判定的应用领域.在实际应用中,虽然基于 SAT 判定的方法并不一定是最好的,但与通过设计专门的工具来解决问题相比,将问题转换为 SAT 判定问题,利用现有的 SAT 判定工具来求解往往更为简单.

* 基金项目: 国家自然科学基金(61133001, 61322202, 61420106004)
收稿时间: 2014-08-03; 修改时间: 2014-10-21; 定稿时间: 2014-11-28

本文针对 CNF(conjunctive normal form)公式的可满足性判定问题,目前,绝大部分 SAT 判定工具都要求输入是 CNF 形式,如果是其他形式的布尔公式,通过添加辅助变量,可以很容易地将其变换为 CNF 形式[2].

大部分 SAT 判定器采用的算法都是基于系统搜索的.在最坏情况下,系统搜索需要穷举所有变量的取值组合来判定公式的可满足性,判定算法的时间复杂度为 $O(2^n)$,其中,n 为公式中变量的个数.

虽然有大量的研究对判定技术做出了改进,但并不能改变 SAT 判定问题的指数复杂度属性.根据强指数时间复杂度猜想[3],一般 SAT 问题的判定复杂度为 $O(2^{\varepsilon n})$,当问题规模趋于∞时,ε的值均趋近于 1[4,5].无论在理论上还是在实践中,都有很多 SAT 判定问题目前无法解决.

与已有的 SAT 判定技术相比,本文并不试图改进判定算法本身或者算法采用的数据结构来提高 CNF 公式的可满足性判效率,而是利用分治的思想,将复杂的 CNF 表示的 SAT 的判定问题划分为多个子问题来求解.方法的基本思路是:基于对 CNF 公式特点的分析,即:如果一个布尔公式 F 中的子句可以划分为 m 个子句组 C_1,C_2,...,C_m,每个子句组中包含的变量集合分别为 S_1,S_2,...,S_m,这些变量集两两不相交,即:对任意的 $1\leqslant i\leqslant m$,$1\leqslant j\leqslant m$,$i\neq j$,有 $S_i\cap S_j=\varnothing$,则布尔公式 F 是可满足的,当且仅当它的划分子句组 C_1,C_2,...,C_m 都是可满足的;布尔公式 F 是不可满足的,当且仅当它某个的划分子句组 $C_i(1\leqslant i\leqslant m)$是不可满足的.

考虑采用穷举的方法来实现 SAT 判定的情形.如果将公式作为一个整体来判定,算法的时间复杂度为 $O(2^n)$,其中,n 为公式中变量的总个数.利用划分的方法,上述的子句组的判定复杂度为 $O(2^{|s_1|}+2^{|s_2|}+...+2^{|s_m|})$.考虑到 $n=|S_1|+|S_2|+...+|S_m|$,直接进行判定时的复杂度为 $O(2^n)=O(2^{|s_1|}\times 2^{|s_2|}\times...\times 2^{|s_m|})$.可见,利用划分的判定方法会极大地降低判定的复杂度.

当然,很少有 CNF 公式能够直接产生子句组划分.为解决这一问题,本文提出了一种基于聚类的方法,先将子句聚类为多个簇,然后在公式中消去簇之间的公共变量,从而产生子句组划分.

子句组的可满足性判定,可以运用现有的 SAT 判定方法或者工具来求解.如果对每个子句组并行进行判定,对整个公式的判定时间将接近于单个子句组判定时间的最大值.

本文第 1 节介绍相关的研究工作.第 2 节给出相关的术语定义.第 3 节描述基于划分的 SAT 判定方法.第 4 节给出利用聚类来产生子句组划分的方法.第 5 节给出算法的实验结果.第 6 节对全文进行总结,并给出进一步的研究方向.

1 相关研究

基于 SAT 解决问题的方法可以分为 3 个步骤,即:编码、预处理和 SAT 判定.编码是指通过某种形式的模型描述将问题用命题逻辑来表示,目前,大部分 SAT 判定工具都要求输入是 CNF 形式;预处理则是对原始公式进行某种形式的变换或化简,以减少经过预处理之后公式的可满足性判定时间.对公式的化简主要目的是减少公式中变量或者子句的个数,相关技术包括变量消除[6]和阻塞子句消除[7]等;SAT 判定问题已经有了深入的研究,目前取得较好效果的 SAT 判定方法主要有两类,即:CDCL(conflict driven clause learning)方法和 SLS(stochastic local search)方法[8].

大部分采用 CDCL 方法的 SAT 判定工具都是基于 DPLL[9]的搜索方法.对于 CNF 形式的布尔公式,DPLL 选定一个未赋值的变量,将其赋值为 1(表示“真”)或者 0(表示“假”),然后对公式运用隐含规则、单子句规则(unit clause rule)和布尔约束传播(Boolean constraint propagation,简称 BCP)来进行推理.如果在某个步骤中发现存在某个子句的所有文字为假,则说明在当前赋值下公式不可能为真.这时需要通过回溯,尝试改变一些变量的赋值.和此前的方法相比,DPLL 方法避免了原有的 SAT 判定算法的指数次方的空间复杂度问题,可以用于解决规模较小的实际问题.

GRASP[10]是第 1 个具有实用价值的 SAT 判定器,它主要在 3 个方面做出了改进,即:启发式决策、冲突驱动的学习和非时间序回溯.启发式决策是指在给变量赋值时,它总是选择能使最多的未确定值的子句为真的变量;冲突驱动的学习是指通过将当前产生冲突的信息编码成子句并添加到公式中,从而帮助减少未来的搜索空间;非时间序回溯是指当发现冲突需要回溯时,不是简单地回溯到最近的决策层,而是通过冲突分析来决定回溯的

层次.GRASP 可以处理包含 1 000 个左右变量的 SAT 判定问题.

zChaff[11]判定器实现了 Chaff 算法,Chaff 算法的改进主要是两个方面,即:基于与变量状态无关的衰减和(variable state independent decaying sum,简称 VSIDS)的启发式分支决策与基于两文字监视的布尔约束传播.在判定过程中,Chaff 选择 VSIDS 值最大的文字对应的自由变量来进行分支决策,并对 VSIDS 值进行动态调整.由于 VSIDS 值与变量赋值无关,因此它的维护成本较小.两文字监视方法避免了每次对变量进行赋值时都要判断是否需要进行布尔约束传播的处理,并且在进行布尔约束传播时,只需要对监视文字被设为假的那些子句进行处理.zChaff 可以处理包含 10 000 个左右变量的 SAT 判定问题.

BerkMin[12]主要在选择赋值变量和子句删除策略方面做了改进;MiniSat[13]则采用类似于 Chaff 算法,但在多个方面进行了调整.

Glucose[14]在 MiniSat 的基础上进行了改进.如果在冲突过程中学习得到的子句太多,将会影响到判定处理的效率.Glucose 将这些学习得到的子句中的“坏子句”移除,以提高判定的效率.

此外,由于变量的决策顺序对于 SAT 的判定效率会产生很大的影响,因此,一些研究关注如何在预处理的过程中寻找较优的变量决策顺序,如文献[15]中提出一种方法,通过变量的活跃性、子句的连接性、变量在子句中的分布以及变量之间的关联来确定变量的决策顺序,以提高 SAT 判定的效率.

文献[16]提出用超图(hypergraph)划分的方法来加速布尔可满足性判定.将 CNF 公式中的子句看作顶点,将两个子句中的共同变量看作连接两个子句的边,可以将公式转换为一个超图.使用图的平衡二分法,可以将子句划分成两组,这两组子句间的共同变量称为割变量.如果把含有割变量的子句分出来,可以进一步得到第 3 个子句组.由于该文献采用的是平衡割的方法,因此得到的割变量个数往往很多,因此无法用消去变量的方法来将问题分解为独立的子问题求解,而只能是将割变量作为优先赋值的变量,通过调用其他判定工具来求解.另外,由于公式中子句的个数往往很多,因此图的分割也需要很长的时间.

SLS 算法的基本处理过程是:首先给每个布尔变量设定一个初始值(一般是随机的),在接下来的每次处理中会选择一个变量,改变它的值.如此反复,直至发现使得布尔公式可满足的一种取值,或者是达到某种终止条件.近来,SLS 算法也取得了很大的进展[17−19],尤其是对随机的 SAT 问题判定方面.

通过对判定算法和数据结构方面的改进,目前 SAT 判定工具可以解决大部分含有上万个变量的 CNF 公式的可满足性判定问题,但是由于问题本身的复杂性和实际应用(例如电子设计自动化)中问题规模的不断增大,目前的工具还远不能满足实际应用中的一些需要.

针对同一个 SAT 判定问题,由于不同的判定工具的效率不同,一种最直观、最简单的方法就是将同一 SAT 判定问题使用不同的判定工具,然后用并行或者分布式的方式来求解;然后,采用最先运行结束的判定工具的结果.如果使用随机判定工具,可以给判定工具设定不同的参数,如随机种子值.

也有一些研究尝试将问题划分为多个复杂度更小的子问题,然后用并行或分布式的方法来求解.例如,文献[20]中提出了一种基于划分的方法,该方法针对 QBF(quantified Boolean formulas)公式的可满足性,根据公式中存在量词和全称量词的特点动态地将公式划分为多个子公式来求解.但该方法不能用于一般 SAT 问题的判定.

针对网格和云计算,文献[21]提出一种方法,递归地将 SAT 判定问题划分为子问题树,然后将树上的节点代表的子问题交给网络或者云,用并行的方式来进行判定.但该划分方法产生的子问题的判定复杂度与原问题的判定复杂度仍较为接近,就降低问题的复杂度而言,效果并不明显.

本文提出的方法是基于分治的思想,通过把 CNF 公式中的子句聚类到不同的簇,再消去簇之间的公共变量,产生子句组划分,从而将 CNF 公式的 SAT 判定问题转化为多个独立的子问题来求解.子问题的求解还可以并行,以进一步加快判定速度.

2 术语定义

大部分 SAT 判定的相关工具开发和研究都限定了输入是 CNF 形式的公式,如果是其他形式的布尔公式,通过适当的转换方法,可以将公式转换为 CNF 形式.下面给出与 CNF 公式相关的一些定义.

定义 1(文字 *l*(literal)). 布尔变量的正或者负形式.

例如:对于布尔变量 x,它对应的文字为 x 和$\neg x$.

定义 2(子句 *c*(clause)). 一个或者多个文字的析取.

例如:$(\neg x_1+x_2+\neg x_3)$就是一个子句.

定义 3(CNF 公式). 由一个或多个子句的合取构成的布尔公式.

例如:$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6)$就是一个 CNF 形式的布尔公式.

定义 4(子句组). CNF 布尔公式中的一个或者多个子句合取构成子句组.

例如:$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)$和$(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6)$都是布尔公式$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6)$的子句组.

定义 5(子句组变量集). 一个子句中包含的文字对应的布尔变量的集合称为该子句的变量集.子句组中包含的所有子句的变量集的并称为该子句组的变量集.

例如:$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)$和$(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6)$对应的变量集分别为$\{x_1,x_2,x_3\}$和$\{x_4,x_5,x_6\}$.

定义 6(子句组划分). 如果一个布尔公式可以分成多个子句组,且任意两个子句组的变量集的交集均为空,则称这些子句组构成该布尔公式的一个划分.

例如:子句组$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)$和子句组$(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6)$构成布尔公式$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6)$的一个划分,但是子句组$(\neg x_1+x_2+\neg x_3)(x_4+\neg x_5+\neg x_6)$和子句组$(x_1+\neg x_2+\neg x_3)(\neg x_4+x_5+\neg x_6)$则不能构成布尔公式$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6)$的划分.

3 基于划分的 SAT 判定

3.1 基本思想

SAT 判定问题是 NPC 问题,意味着这类问题的时间复杂度是指数次方的.如果问题的规模比较大,则 SAT 判定就很难.与现有 SAT 判定的研究不同的是,本文试图通过将复杂的 CNF 公式的 SAT 判定问题分解为多个子句组的 SAT 判定问题,以降低判定的复杂度.为了便于理解,先看前面提到的 CNF 公式:

$$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6) \tag{1}$$

如果使用传统的 SAT 判定方法,如果采用穷举的方法,需要考虑 6 个布尔变量的取值组合,也就是要考虑 2^6 种情形.但是如果我们把公式分为两个子公式,即:

$$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3) \tag{2}$$

$$(x_4+\neg x_5+\neg x_6)(\neg x_4+x_5+\neg x_6) \tag{3}$$

直观来看:公式(1)是可满足的,当且仅当公式(2)和公式(3)都是可满足的.

对于公式(2),因为只有 3 个变量,所以判定公式(2)的可满足性在穷举的情况下只需要考虑 2^3 种情形.同样,对于公式(3),也只需要考虑 2^3 种情形.因此,当把公式(1)的判定问题转化为两个子公式的判定问题来求解后,只需要考虑 $2^3+2^3=2^4$ 种情形.

就一般情况而言,假定一个布尔公式,它可以划分为 m 个子句组,即 $C_1,C_2,...,C_m$,则此公式的判定问题可以转换为对子句组 $C_1,C_2,...,C_m$ 分别进行判定来求解.

定理 1. 如果一个布尔公式 C 可以划分为 m 个子句组 $C_1,C_2,...,C_m$,每个子句组中包含的变量集合分别为 $S_1,S_2,...,S_m$,对任意的 $1\leqslant i\leqslant m,1\leqslant j\leqslant m,i\neq j$,有 $S_1\cap S_2=\varnothing$,则布尔公式 C 是可满足的,当且仅它的划分子句组 $C_1,C_2,...,C_m$ 都是可满足的.

证明:

- 先证明充分性:

如果 C 是可满足的,因为 $C=C_1\wedge C_2\wedge...\wedge C_m$,故必然存在一组布尔变量的赋值,使得 $C_1,C_2,...,C_m$ 都同时满足,因此,$C_1,C_2,...,C_m$ 都是可满足的.

如果 C 是不可满足,则必然存在某个子句组是不可满足的,故有子句组不都是可满足的.

- 再证明必要性:

假定子句组 $C_1,C_2,...,C_m$ 都是可满足的,则对于变量集合 $S_1,S_2,...,S_m$,必然存在相应集合中的一组变量取值,使得 $C_1,C_2,...,C_m$ 分别是可满足的.因为对任意的 $1\leqslant i\leqslant m,1\leqslant j\leqslant m,i\neq j$,有 $S_1\cap S_2=\varnothing$,即:这些变量组中,变量的取值是不存在冲突的,故将这些变量组的取值合起来,就可以得到针对所有变量的一组取值,这组取值可以使 $C=C_1\wedge C_2\wedge...\wedge C_m$ 取值为真,从而有 C 是可满足的.

假定子句组 $C_1,C_2,...,C_m$ 不都是可满足的,即:至少存在某个子句组 $C_i(1\leqslant i\leqslant m)$的取值不可能为真,当然也不可能找到针对所有变量的一组取值,能使 $C=C_1\wedge C_2\wedge...\wedge C_m$ 取值为真,从而有 C 是不可满足的.

3.2 基于划分的SAT判定方法

如果一个 CNF 公式能够划分为多个子句组,则该 CNF 公式的可满足性判定可以转化为子句组的可满足性判定.因为子句组的规模小于原公式的规模,所以判定的复杂度会变低.

基于子句组划分的布尔可满足性判定方法的基本思路是:先将公式划分这子句组,然后对每个子句组分别进行可满足性判定:如果每个子句组都是可满足的,则原布尔公式是可满足的;如果存在某个子句组是不可满足的,则原布尔公式是不可满足的.

4 基于聚类的划分

4.1 基本思想

在大部分情况下,可能无法直接产生布尔公式的子句组划分,本节提出了一种对布尔公式进行处理来产生布尔公式子句组划分的方法.为了便于理解,先看一个 CNF 形式的布尔公式:

$$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)(x_2+x_3+x_4)(x_3+x_5+x_6)(x_5+\neg x_6+\neg x_7)(\neg x_5+x_6+\neg x_7) \tag{4}$$

这个公式没有办法直接划分为更小的子句组,但是如果我们考虑对 x_3 进行赋值,采用一种类似于二叉决策的方法对公式进行化简,就可以产生子句组的划分:

- x_3 取值为 0 时,公式化简为$(x_2+x_4)(x_5+x_6)(x_5+\neg x_6+\neg x_7)(\neg x_5+\neg x_6+\neg x_7)$,该公式可以划分为两个子句组,即:$(x_2+x_4)$和$(x_5+x_6)(x_5+\neg x_6+\neg x_7)(\neg x_5+\neg x_6+\neg x_7)$;
- x_3 取值为 1 时,公式化简为$(\neg x_1+x_2)(x_1+\neg x_2)(x_5+\neg x_6+\neg x_7)(\neg x_5+\neg x_6+\neg x_7)$,可以进一步划分为两个子句组,即:$(\neg x_1+x_2)(x_1+\neg x_2)$和$(x_5+\neg x_6+\neg x_7)(\neg x_5+\neg x_6+\neg x_7)$.

直观地看,原公式不能直接产生子句组划分的原因,是因为有 x_3 的存在,把整个公式中的子句联系在一起了.而通过赋值把 x_3 从公式中消除后,这一联系不复存在,因而就可以产生子句组划分.

受此启发,我们认为:可以通过寻找能产生子句组划分的变量,然后通过这些变量来产生子句组划分.首先定义一个割变量(集)的概念,见定义 7.

定义 7(割变量(集)). 在 CNF 表示的布尔公式中,如果存在一个或者多个变量,当从公式中消除这些变量后,原本不能直接产生子句组划分的公式变成了可以产生子句组划分的公式,则这一个或者多个变量就称为割变量(集).

例如:对公式(4)而言,x_3 就是一个割变量,如果消去 x_3,该公式就可以产生子句组划分.

4.2 基于聚类的划分

聚类(clustering)就是将数据分组成为多个簇(cluster)或类,使得在同一个簇内的对象之间具有较高的相似度,而属于不同簇之间的对象差别较大.聚类是数据挖掘、模式识别等研究方向的重要研究内容之一,在识别数据的内在结构方面具有极其重要的作用.

没有任何一种聚类算法或者技术可以普遍适用于各种类型数据集的聚类,这其中的原因之一就是因为很难给出簇的概念的精确定义[22].根据数据在聚类中的集聚规则以及应用这些规则的方法,可以将聚类算法大致划分为层次化的聚类算法(基于联接的聚类算法)、基于划分的聚类算法、基于密度和网格的聚类算法和其他聚类算法.

层次聚类又可以分为凝聚的层次聚类和分裂的层次聚类:

- 凝聚的层次聚类采用自底向上的策略,首先将每个对象作为一个簇,然后合并这些原子簇为越来越大的簇,直到所有的对象都在一个簇中,或者某个终结条件被满足.绝大多数层次聚类方法属于这一类,它们只是在簇间相似度的定义上有所不同;
- 与凝聚的层次聚类相反,分裂的层次聚类采用自顶向下的策略,它首先将所有对象置于同一个簇中,然后逐渐细分为越来越小的簇,直到每个对象自成一簇,或者达到了某个终止条件.

从上面的分析可以看出:要从一个 CNF 公式产生好的子句组划分,应该尽量减少不同的子句组之间的共同变量个数,也就是:应该将包含相同变量多的子句归到同一个子句组,将含共同变量少的子句归到不同的子句组.这样,才能降低消去这些变量所需的代价.

为此,本文提出了一种凝聚的层次聚类算法来实现这一目标.基本思想是:通过聚类将包含共同变量多的子句聚类到同一个簇中;然后,通过消去不同簇之间的共同变量来产生子句组划分.如图 1 所示:算法先从文件中读出子句,把每个子句作为一个簇,然后用聚类方法将它们聚类到不同的簇(簇用圆圈表示)中,这些簇之间可能存在公共变量(用圆圈间的连线表示),将这些公共变量消去后,得到的簇就构成了原 CNF 公式的一种划分.

以公式(4)为例,由于子句$(\neg x_1+x_2+\neg x_3)$,$(x_1+\neg x_2+\neg x_3)$和$(x_2+x_3+x_4)$都包含变量 x_2 和 x_3,因此将它们聚类到同一个簇.同样道理,将子句$(x_3+x_5+x_6)$,$(x_5+\neg x_6+\neg x_7)$和$(\neg x_5+x_6+\neg x_7)$聚类到同一个簇.消去两个簇之间的公共变量 x_3,就可以得到公式(4)的一种子句组划分.如图 2 所示.

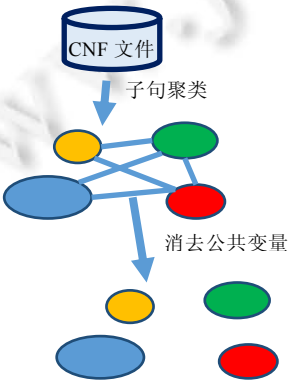

Fig.1 CNF clustering and partition

图 1 CNF 公式的聚类和划分

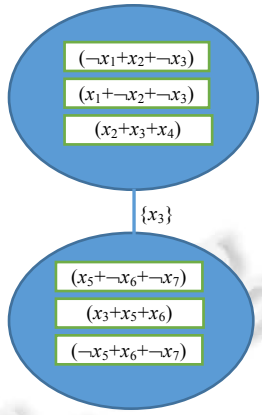

Fig.2 Clustering result of a CNF

图 2 一个 CNF 公式的一种聚类

算法首先将每个子句作为一个簇,然后用层次聚类的办法来产生最终的聚类结果.为了记录簇中包含的子句和变量,簇定义为一个二元组,其中,*clauseSet* 表示该簇包含的子句的集合,*variableSet* 表示该簇包含的变量的集合.为简单起见,在下文中,簇 C_i 中包含的变量集将直接用 C_i 表示.

针对 CNF 表示的布尔公式的特点,两个簇 C_1 和 C_2 之间的相似度定义为:

定义 8(簇之间的相似度):

$$sim(C_1,C_2)=\max\left(\frac{|C_1\cap C_2|}{|C_1|},\frac{|C_1\cap C_2|}{|C_2|}\right),$$

其中,$|C_1|$,$|C_2|$和$|C_1\cap C_2|$分别表示簇 C_1,C_2 中变量的个数以及 C_1,C_2 中包含的变量集的交集中变量的个数.

子句组的聚类算法所算法 1 所示.算法中的 3 个输入 *DIMACS_file*,min_*cluster* 和 *threshold* 分别表示存储 CNF 公式的文件名(DIMACS 格式[23])、期望的簇个数的最小值以及相似度阈值.为避免将所有子句聚类到一个簇,算法中用 min_*cluster* 来控制聚类结果中簇的个数:如果聚类后簇数结果小于 min_*cluster*,则返回本次聚类之前的结果,即 $C_{\min}$;否则,重复聚类过程.这是为了避免出现当 *threshold* 设置过小时,会将所有子句聚类成一个簇

的情况.变量 *threshold* 决定两个簇之间的相似度大到什么程度时对两个簇进行合并:如果 *threshold* 设置得较大,则最终产生的簇的个数就会比较多,相应的割变量总个数也会增多;反之,如果 *threshold* 设置得较小,则簇的个数和割变量数也会相应地减少.

算法 1. *CNF_Clustering*(*DIMACS_file*,min_*cluster*,*threshold*).

//通过聚类将子句划分到不同簇中

1. read clauses from *DIMACS_file*
2. create an empty cluster *C*
3. generate a cluster for every clause and added it into *C*
4. *merged*=**true**
5. *loopUperBound*=*sizeof*(*C*)
6. **while** *merged* **do**
7. *merged*=**false**
8. **if** *sizeof*(*C*)>min_*cluster* **then**
9. $C_{\min}=C$
10. **else**
11. **break**
12. **end if**
13. *i*=0
14. **while** (*i*<*loopUperBound*) **do**
15. *j*=0
16. max_*similarity*=0
17. max_*index*=*i*
18. **while** (*i*<*loopUperBound*) **do**
19. **if** *i*!=*j* **then**
20. $sim(C_i,C_j)=\max\left(\frac{|C_i\cap C_j|}{|C_i|},\frac{|C_i\cap C_j|}{|C_j|}\right)$
21. **if** *sim*(C_i,C_j)>max_*similarity* **then**
22. max_*similarity*=*sim*(C_i,C_j)
23. max_*index*=*j*
24. **end if**
25. **end if**
26. *j*=*j*+1
27. **end while**
28. **if** max_*similarity*≥*threshold* **then**
29. *merge*(C_i,C_{max_index})
30. *merged*=**true**
31. *loopUperBound*=*loopUperBound*−1
32. **end if**
33. *i*=*i*+1
34. **end while**
35. **end while**
36. **if** *sizeof*(*C*)>min_*cluster* **then**

37. **return** C
38. **else**
39. **return** $C_{\min}$
40. **end if**

4.3 算法的时间复杂度分析

聚类算法的时间复杂度为 $O(c^3)$,其中,c 为公式中子句的个数.

对 CNF 公式进行聚类后,假定聚类结果包含划 m 个簇,每个簇中的子句包含的变量数分别为 $n_1,n_2,...,n_m$ 个,割变量的个数为 k.因为需要分 2^k 种情形将这 k 个变量消去,然后产生 m 个子句组分别进行判定,所以划分的时间复杂度为 $O(2^k)$.

本文提出的算法主要针对 k 较小的情况,即,k 小于某个特定的常数.在此情况下,聚类和划分的算法是多项式时间复杂度的.

如果子句组的判定采用穷举的方法,判定的总时间复杂度为 $O(2^k\cdot(2^{n_1}+2^{n_2}+...+2^{n_m}))$, 其中,$n_1+n_2+...+n_m=n$,$n$ 为原公式中变量的个数.

4.4 一个简单的实例

给出聚类和划分方法应用的一个简单例子.假定要判定图 3 中的电路是否是可满足,用 SAT 方法,可以先将电路转换为布尔公式,如图 3 所示.

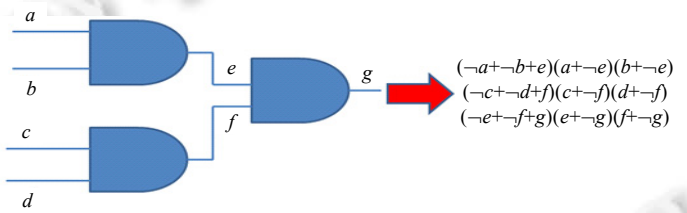

Fig.3 A circuit and its corresponding CNF formula

图 3 一个电路及其对应的布尔公式

在 SAT 的研究领域,往往将布尔公式用 CNF 形式存储在文件中.将图 3 中布尔公式中的变量(a,b,c,d,e,f,g)分别用数字 1~7 代替,并用负号代替取反符号,在每个子句后面加一个 0 作为子句的结束标志,文字之间用空格分隔,每个子句一行,并在文件的第 1 行加上子句数、变量数等信息,就得到了该公式对应的 CNF 文件,如图 4 所示.

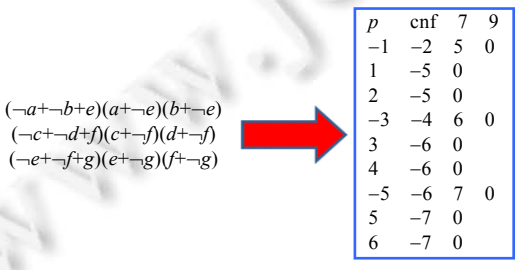

Fig.4 CNF formula and its file format

图 4 CNF 公式对应的文件存储形式

如果将相似度定义为两个子句中的共同变量数与两个子句中包含的变量个数的比值中的较大者,对 CNF 文件中的子句进行聚类,可以将子句聚类成 3 个簇.消去簇间的共同变量,就可以将该公式划分为 3 个可以独立进行判定的子公式.这种划分相当于将电路划分成了 3 个子电路,如图 5 所示.这一例子表明:通过聚类,可以从

CNF 公式中发现一些结构信息,利用这些结构信息,可以将复杂问题划分为相对简单的子问题.

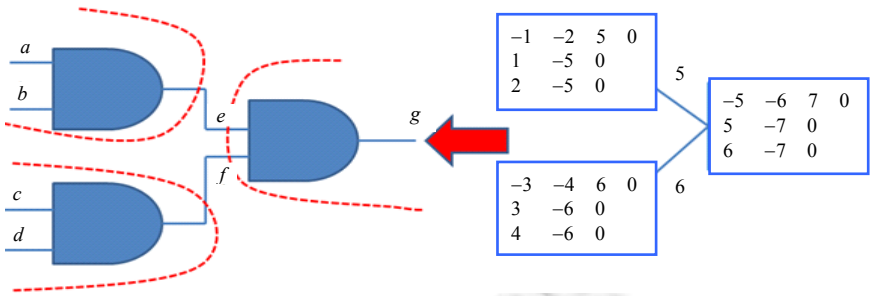

Fig.5 Relationship of CNF partition and circuit

图 5 布尔子句划分及其与电路对应关系

在电子设计自动化领域,往往通过先设计一些小的功能模块,再用这些小的功能模块来构建复杂的电路,各个电路模块之间的联系较少.在其他领域,如软件开发等,也大量存在这种情况.这意味着在很多情况下,可以将复杂的问题分解为多个子问题,且子问题之间相对独立.

4.5 对划分方法的改进

在实现实验算法时,我们并未采用消去割变量的方法来产生子句组划分,因为这种做法在割变量数目较多时效率会很低.

为便于理解,以第 4.1 节中提到的公式(4)为例,通过聚类,将公式分成两个簇,即:$(\neg x_1+x_2+\neg x_3)(x_1+\neg x_2+\neg x_3)(x_2+x_3+x_4)$和$(x_3+x_5+x_6)(x_5+\neg x_6+\neg x_7)(\neg x_5+x_6+\neg x_7)$,分别用 C_1,C_2 表示它们.对 C_1 和 C_2 分别进行可满足性判定:

- 如果 C_1 或 C_2 是不可满足的,则公式(4)不可满足;
- 如果 C_1 和 C_2 都是可满足的,且分别能找到使 C_1 和 C_2 都可满足的一组变量取值 $V_1(x_1,x_2,x_3,x_4)$和 $V_2(x_3,x_5,x_6,x_7)$,其中,x_3 的取值不冲突(即,x_3 的取值都为 0,或者都为 1),则公式(4)是可满足的;否则,公式(4)是不可满足的.因为存在 $V_1(0,0,0,1)$和 $V_2(0,1,1,0)$能使 C_1 和 C_2 皆可满足,且 x_3 的取值不冲突,故公式(4)是可满足的.

一般地,如果一个布尔公式 C 聚类为 m 个子句组 $C_1,C_2,...,C_m$,为判定其可满足性,可以对每个子句组寻找使其满足的一组变量取值:

- 如果 $C_1,C_2,...,C_m$ 中的某个子句组是不可满足的,则公式不可满足;
- 如果每个子句组都是可满足的,则判断使它们取值为真的变量值中割变量取值是否一致:
  - 若一致,则公式 C 是可满足的;
  - 如果当前能使所有子句组都满足的变量取值中存在割变量取值冲突的情况,则寻找下一组能使 $C_1,C_2,...,C_m$ 满足的变量取值;如果无法找到能使 $C_1,C_2,...,C_m$ 为真,且割变量取值不冲突的变量取值组合,则公式 C 也是不可满足的.

这一改进避免了穷举割变量的组合来消去割变量以产生子句组划分,且对现有的 SAT 判定工具稍加修改即可实现.

5 实 验

针对 SATLIB[24]和 SAT Competition 2013 中 Application Benchmarks[25]的部分测试用例,用算法 1 进行聚类,得出的实验结果见表 1.在表 1 和表 2 中,省去了.cnf 的扩展名,并且将过长的文件名使用其缩写表示.例如:bivium-39-200-0s0-0x5fa955de2b4f64d00226837d226c955de4566ce95f660180d7-30.cnf 和 bivium-39-200-0s0-0x28df9231b320bd56dfb68bfc7c3f0ca20dbae6b0eba535ad91-98.cnf 分别缩写为 bivium-39-200-0s0-0x5f 和 bivium-39-200-0s0-0x28.

Table 1 Results of CNF file clustering

表 1 CNF 文件聚类结果

文件名	变量数	子句数	相似度值	聚类后簇数	割变量总数
001	4 128	126 548	0.2	65	0
002	4 128	126 548	0.2	65	0
2bitadd_10	590	1 422	0.15	2	26
3blocks	283	9 690	0.2	2	16
4blocks	758	47 820	0.1	2	25
4blocksb	410	24 758	0.15	2	25
Beempgsol2b1	21 464	61 561	0.05	2	42
Beempgsol5b1	21 465	61 564	0.1	2	27
Bivium-39-200-0s0-0x5f	978	5 514	0.2	240	0
Bivium-39-200-0s0-0x28	973	5 570	0.2	240	0
Bivium-40-200-0s0-0x66	971	5 469	0.2	241	0
Bivium-40-200-0s0-0x92	970	5 432	0.2	241	0
Bmc-ibm-2	2 810	11 683	0.05	2	7
C6288_miter	5 684	16 957	0.1	2	37
C7552_miter	3 267	9 181	0.05	2	32
Ctl_3791_556_unsat	16 541	111 990	0.1	39	0
Hitag2-10-60-0-0x8e	2 271	30 273	0.1	73	0
Hitag2-10-60-0-0x22	2 260	29 903	0.1	71	0
Hitag2-10-60-0-0xbc	2 321	30 992	0.2	73	0
Logistics.b	843	7 301	0.15	5	0

Table 2 Comparison of SAT solving

表 2 SAT 判定时间比较

文件名	Lingeling 判定用时	聚类时间	聚类处理后 lingeling 判定时间
001	16 281	13	70
002	87 276	22	3 386
2bitadd_10	27	2	2
Beempgsol2b1	26 716	121	12
Beempgsol5b1	27 036	136	21
Bivium-39-200-0s0-0x5f	4 433	2	3 457
Bivium-39-200-0s0-0x28	2 291	2	2 276
Bivium-40-200-0s0-0x66	5 683	2	2 573
Bivium-40-200-0s0-0x92	1 591	2	1 466
C6288_miter	>200000	10	10
Ctl_3791_556_unsat	>200000	54	32 400
Hitag2-10-60-0-0x8e	3 639	40	3 195
Hitag2-10-60-0-0x22	6 537	40	5 295
Hitag2-10-60-0-0xbc	6 322	42	5 682

实现中发现:如果相似度阈值大,则聚类后生成的簇数量可能就越多.为使实验结果更易理解,实验中将相似度阈值设得较小,使得最终产生的簇数量比较少.

表 1 所示的实验结果表明:大多数测试用例都只需要消去较少的割变量甚至不用消去任何割变量,就可以将问题分解为几个子问题来求解.割变量总数为 0,意味着聚类后可直接将原问题转化为规模更小的几个子问题来求进行判定.

表 2 将直接运用 SAT 工具判定所用的时间和先用本文的提出的方法进行聚类后再运用 SAT 工具进行相应的处理判定所需的时间做了比较.实验上所用的 SAT 判定工具是 lingeling[26],它在近年来的 SAT Competition 比赛中夺得了数个奖项.表 2 中第 1 列表示待判定问题的文件名;第 2 列表示直接运用 lingeling 时所需的判定时间;第 3 列表示聚类所花的时间;第 4 列表示聚类处理后再用 lingeling 所花的判定时间,单位均为 s.

从表 2 中的实验结果可以看出:与直接 SAT 判定工具相比,在问题复杂度较低的情况下,由于本文提出的方法在聚类上花费的时间相对较多,因此相对而言需要花费更多的时间.但对于较为复杂的问题,即使算上聚类所用的时间,本文提出的方法还是可以大幅提高 SAT 判定的效率.运用本文提出的方法,绝大部分实验用例的判定都可以在较短时间内完成.

表 1 中列出的有些文件由于判定所需时间太短而难以比较,因此在表 2 中并未列出.有些问题虽然割变量

总数为 0,但聚类处理后判定速度提高并不明显,如 Bivium-39-200-0s0-0x28.分析发现:这是由于聚类后很多簇中只包含一个变量的单子句,对于这些子句,SAT 判定工具利用单子句规则就能直接处理,因此这一结果也在情理之中.

6 结 语

在计算机科学领域,针对复杂问题,往往考虑采用分治法,将问题划分为多个规模更小的子问题来求解.本文正是基于分治的思想,通过将复杂的 CNF 表示的 SAT 的判定问题划分为多个子问题以降低判定的复杂度.提出了 CNF 形式的布尔公式的子句组划分的概念,并证明了布尔公式的可满足性问题可以转化为子句组的可满足性问题来求解.

给出了一种从 CNF 公式产生子句组划分以及利用子句组划分来进行 SAT 判定的方法.对于不能直接产生子句组划分的公式,提出了聚类的方法,将子句聚类成多个簇,把包含共同变量多的子句聚类到同一个簇,包含共同变量少的子句聚类到不同的簇,将不同簇之间的共同变量作为割变量,通过对每个簇单独判定,必要时再判断割变量的取值是否冲突,来判定公式的可满足性.

本文提出的方法主要有 3 个方面的优点:首先,利用子句组划分,可将问题分解为规模较小的子问题,从而降低 SAT 判定的复杂度;其次,由于每个子句组划分的判定可以并行,因而判定速度能得到进一步的提高;最后,由于此方法不涉及具体的 SAT 判定,因此,SAT 判定技术的研究进展可以直接应用到对子句组划分的判定中.

因为本文提出的算法需要在聚类和判定割变量的取值是否冲突上消耗时间,因此如果对于简单的 SAT 判定问题,本文提出的算法可能比直接应用 SAT 判定算法要花费更多的时间代价;但是对于复杂的 SAT 问题,如果聚类后产生子句组划分所涉及的割变量个数较少,因此有可能大幅降低总的判定时间.

由于当前的聚类和划分工具只是一个初步的实现,相应的算法和工具都存在很大的改进空间.例如,实验所用的聚类和划分工具都是用 Python 语言来实现,而不是采用效率较高的 C 等语言,因此聚类处理用时较长也在预料之中.下一步将对聚类算法和工具进行改进,以大幅提高其运行速度,提高聚类质量.

此外,本文采用聚类的方法来产生子句组划分,实验中,相似度阈值选择是通过反复尝试来找到合适值的,目前我们还没有找到如何更好地确定相似度阈值的办法.此外,不同类型数据的簇的特征定义可能不同.因此,针对不同问题领域的 CNF 公式的聚类,可能需要不同的聚类算法,才能更好地利用问题本身的特征以获得最好的聚类效果.例如涉及逻辑电路的 SAT 判定,如果电路可以分成两个或者多个相对独立的子电路,就可以将这些子电路对应的 CNF 子句划分为相应的子句组.如何利用特定应用领域的知识来更好地产生子句组的划分,还需要进一步研究.

References:

[1] Cook S. The complexity of theorem proving procedures. In: Proc. of the ACM SIGACT Symp. on the Theory of Computing. 1971. [doi: 10.1145/800157.805047]

[2] Tseitin. G. On the complexity of derivation in propositional calculus. In: Proc. of the Structures in Constructive Mathematics and Mathematical Logic, Part II: Seminars in Mathematics (translated from Russian). Steklov Mathematical Institute, 1968. 115−125. [doi: 10.1007/978-3-642-81955-1_28]

[3] Impagliazzo R, Paturi R, Zane F. Which problems have strongly exponential complexity? Journal of Computer and System Sciences, 2001,63(4):512−530. [doi: 10.1006/jcss.2001.1774]

[4] Calabro C, Impagliazzo R, Paturi R. A duality between clause width and clause density for sat. In: Proc. of the 21st Annual IEEE Conf. on Computational Complexity. Washington: IEEE Computer Society, 2006. 252−260. [doi: 10.1109/CCC.2006.6]

[5] Impagliazzo R, Paturi R. On the complexity of *k*-SAT. Journal of Computer and System Sciences, 2001,62(2):367−375. [doi: 10.1006/jcss.2000.1727]

[6] Eén N, Biere A. Effective preprocessing in SAT through variable and clause elimination. In: Proc. of the 8th Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT 2005). LNCS 3569, 2005. 61−75. [doi: 10.1007/11499107_5]

[7] Järvisalo M, Biere A, Heule M. Simulating circuit-level simplifications on CNF. Journal of Automated Reasoning, 2012,49(4): 583−619. [doi: 10.1007/s10817-011-9239-9]

[8] Biere A, Heule M, Maaren H, Walsh T, eds. Handbook of Satisfiability. IOS Press, 2009. [doi: 10.3233/978-1-58603-929-5-3]

[9] Davis M, Logemann G, Loveland D. A machine program for theorem proving. Communications of the ACM, 1962,5(7):394−397. [doi: 10.1145/368273.368557]

[10] Marques-Silva J, Sakallah K. GRASP: A search algorithm for propositional satisfiability. IEEE Trans. on Computers, 1999,48(5): 506−521. [doi: 10.1109/12.769433]

[11] Moskewicz M, Madigan C, Zhao Y, Zhang L, Malik S. Chaff: Engineering an efficient SAT solver. In: Proc. of the 38th Design Automation Conf. (DAC 2001). 2001. 530−535. [doi: 10.1145/378239.379017]

[12] Goldberg E, Novikov Y. Berkmin: A fast and robust SAT solver. In: Proc. of the 5th Design Automation and Test in Europe (DATE 2002). 2002. 142−149. [doi: 10.1016/j.dam.2006.10.007]

[13] Sörensson N, Eén N. MiniSat V1.13—A SAT solver with conflict-clause minimization. In: Proc. of the Posters of the Int'l Symp. on the Theory and Applications of Satisfiability Testing (SAT 2005). 2005.

[14] Audemard G, Simon L. Predicting learnt clauses quality in modern SAT Solver. In: Proc. of the 21st Int'l Joint Conf. on Artificial Intelligence (IJCAI 2009). 2009.

[15] Durairaj V, Kalla P. Guiding CNF-SAT search by analyzing constraint-variable dependencies and clause lengths. In: Proc. of the High-Level Design Validation and Test Workshop (HLDVT). 2006. 155−161. [doi: 10.1109/HLDVT.2006.319983]

[16] Durairaj V, Kalla P. Exploiting hypergraph partitioning for efficient Boolean satisfiability. In: Proc. of the 9th IEEE Int High Level Design Validation and Test Workshp. 2004. [doi: 10.1109/HLDVT.2004.1431257]

[17] Cai SW, Su KL, Sattar A. Local search with edge weighting and configuration checking heuristics for minimum vertex cover. Artificial Intelligence, 2011,175(9-10):1672−1696. [doi: 10.1016/j.artint.2011.03.003]

[18] Cai SW, Su KL. Configuration checking with aspiration in local search for SAT. In: Proc. of the AAAI 2012. 2012.

[19] Cai SW, Su KL. Local search for Boolean satisfiability with configuration checking and subscore. Artificial Intelligence, 2013,204: 75−98. [doi: 10.1016/j.artint.2013.09.001]

[20] Samulowitz H, Bacchus F. Dynamically partitioning for solving QBF. In: Proc. of the 10th Int'l Conf. on Theory and Applications of Satisfiability Testing (SAT 2007). 2007. 215−229. [doi: 10.1007/978-3-540-72788-0_22]

[21] Hyvärinen A, Junttila T, Niemelä I. Partitioning SAT instances for distributed solving. In: Proc. of the Int'l Conf. on Logic for Programming, Artificial Intelligence, and Reasoning. LNCS 6397, 2010. 372−386. [doi: doi=10.1.1.174.6279]

[22] Estivill-Castro V. Why so many clustering algorithms? ACM SIGKDD Explorations Newsletter, 2002,4(1):65−75. [doi: 10.1145/568574.568575]

[23] CNF file format. 2013. http://people.sc.fsu.edu/~jburkardt/pdf/dimacs_cnf.pdf

[24] Hoos H, Stützle T. SATLIB: An online resource for research on SAT. In: Gent IP, Maaren HV, Walsh T, eds. Proc. of the SAT 2000. IOS Press, 2000. 283−292.

[25] sc13-Benchmarks-Application. 2013. http://www.satcompetition.org/2013/files/sc13-benchmarks-application.tgz

[26] Biere A. Lingeling, plingeling and treengeling entering the SAT competition 2013. In: Balint A, Belov A, Heule M, Järvisalo M, eds. Proc. of the SAT Competition 2013. vol.B-2013-1 of Department of Computer Science Series of Publications B, University of Helsinki, 2013. 51−52.

范全润(1973－),男,云南宣威人,博士生,副教授,主要研究领域为布尔可满足性,形式化验证,电子设计自动化.

段振华(1948－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为可信软件.