计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.201000142

基于学习子句删除策略的SAT求解器分支策略

王钇杰　徐　扬　吴贯锋
西南交通大学数学学院　成都 610031
系统可信性自动验证国家地方联合工程实验室　成都 610031
(806657831@qq.com)

摘　要　对于SAT求解器,目前流行的分支变量决策策略大多是基于冲突的变量活跃度评估算法,选择具有最大活性的未赋值变量作为决策变量,优先解决最近的冲突。但是,它们都忽略了包含决策变量的子句数目对布尔约束传播(BCP)的影响。针对此问题,提出了一种基于学习子句删除策略的分支变量决策策略(VDALCD),在删除学习子句的同时减小被删除子句中变量的活跃度。基于VDALCD策略分别对Glucose4.1,MapleLCMDistChronoBT-DL-v2.1进行改进,形成了求解器Glucose4.1_VDALCD和Maple-DL_VDALCD。以2018年、2019年SAT国际竞赛题为基准测试例,将改进版本与原版本求解器进行比较。实验结果表明,在2018年的例子测试中,Gluose4.1_VDALCD比Gluose4.1多求出26个例子,增加了15.5%。在2019年的例子测试中,Maple-DL_VDALCD比MapleLCMDistChronoBT-DL-v2.1多求出17个例子,增加了7.6%。

关键词:活跃度;学习子句;可满足性问题;学习子句删除策略;完备算法
中图法分类号　TP181

Branching Heuristic Strategy Based on Learnt Clauses Deletion Strategy for SAT Solver

WANG Yi-jie,XU Yang and WU Guan-feng
School of Mathematics,Southwest Jiaotong University,Chengdu 610031,China
National-Local Joint Engineering Laboratory of System Credibility Automatic Verification,Chengdu 610031,China

Abstract　For the SAT solver,most popular branch variable decision-making strategies are based on the variable activity evaluation of conflict. The unassigned variable with the maximum activity is selected as the decision variable,and the most recent conflict is solved first. However,they all ignore the impact of the number of clauses containing decision variables on the Boolean constraint propagation (BCP). To solve this problem,this paper proposes a branch variable decision strategy (VDALCD) based on the learning clause deletion strategy,which reduces the activity of variables in the deleted clause when the clause is deleting. Based on the VDALCD strategy,Glucose4.1 and MapleLCMDistChronoBT-DL-v2.1 are improved to solvers Glucose4.1_VDALCD and Maple-DL_VDALCD. This paper uses 2018 and 2019 SAT international competition questions as benchmark test cases to compare the improved version with the original version of the solver. The experimental results show that Gluose4.1_VDALCD finds out 26 more examples than Gluose4.1,an increase of 15.5% in the 2018 example test. In the 2019 example test,Maple-DL_VDALCD finds out 17 more examples than MapleLCMDistChronoBT-DL-v2.1,an increase of 7.6%.

Keywords　Activity,Learnt clause,Satisfiability problem,Learnt clause deletion strategy,Complete algorithms

1　引言

布尔可满足性问题(Boolean Satisifiability Problem,SAT)是计算机学科理论和人工智能中的著名问题。1971年,Cook证明了SAT问题是NP完全问题[1]。由于所有的NP完全问题在多项式时间内是可以互相转换的[2],因此,如果能找到一个高效的求解SAT问题的方法,那么具有相同时间复杂度的NP完全问题都可以被解决,从而奠定了求解SAT问题的理论基础。自然科学和社会科学的问题中也存在着大量的NP完全问题,如何设计和实现求解SAT问题的有效算法,对于求解其他类型的NP完全问题也有着极为重要的理论和现实意义。求解SAT问题的算法按完备性主要分为两种:完备算法和不完备算法。不完备算法主要是指随机局部搜索算法(Stochastic Local Search,SLS)[3],此算法对

到稿日期:2020-10-26　返修日期:2021-01-23　本文已加入开放科学计划(OSID),请扫描上方二维码获取补充信息。
基金项目:国家自然科学基金(61673320);中央高校基本科研业务费专项资金(2682020CX59)
This work was supported by the National Natural Science Foundation of China(61673320) and Fundamental Research Funds for the Central Universities(2682020CX59).
通信作者:吴贯锋(wgf1024@swjtu.edu.cn)

于随机类问题可以快速得出结论,并且时间复杂度是多项式级的,但是此算法不保证一定能够找到对应 SAT 问题的解。完备算法不仅能正确判定问题的可满足性,同时也可以在问题无解时给出完备的证明。但是完备算法效率低下,理论上需要穷尽所有二叉搜索树的空间去寻找问题的解。完备算法又包括两大类:一类是基于二叉树的优先搜索算法,利用深度优先算法对二叉树进行遍历搜索;另一类是基于归结原理的演绎算法[4],即递归地选择两个子句归结,若原始字句集是不可满足的,则可以通过递归推导出空子句。

由于在实际应用中要求对于可满足问题要给出可满足解,对于不可满足问题能够给出不可满足的证明。因此,本文的主要工作是基于完备算法进行研究。2001 年,Davis 在 DPLL(Davis Putnam Logemann Loveland)[5-6] 的基础上提出了 CDCL(Conflict Driven Clause Learning)[7-8] 算法,极大地提升了 SAT 求解器的性能。目前较为流行的求解器,如 Maple_LCM_Dist[9],Glucose[10] 和 Lingeling[11] 等都是在 CDCL 的基础框架上分别针对变量决策策略、冲突分析、学习子句生成策略、学习子句删除策略和重启策略等方面进行改进的。

早期的分支变量决策有 JW[12](Jeroslow-Wang)算法、Bohm[13] 算法、最短子句出现频率最大(Maximum Occurrence in Minimization,MOM)算法[14] 等,它们都依赖一些子句所固有的特征,如子句的数目大小、子句长度、未赋值变量的个数等来进行分支变量选择。直到 2001 年,Weidenbach 等提出了具有里程碑意义的变元状态独立下降和策略(Variable State Independent Decaying Sum,VSIDS)[15],其引入了动态衰减的思想,当学习子句加入到子句库时,该子句所包含的所有文字对应的计数器都加 1。这使得求解器能根据活跃度的大小动态地选择分支变量,尽可能地优先满足冲突子句。ACIDS[16] 对活跃度增量做了改进,考虑了其与冲突次数的关系,使得与最近冲突相关的变量具有更高的活跃度。Rvan 提出了变量前移(Variable Move-to-Front,VMTF)策略[17],发生冲突时将学习子句中的文字整体移动到活跃度队列之前。但是,以上策略都只考虑了与冲突有关的变量,优先选择参与冲突次数最多的变量,而忽略了删除学习子句对分支变量决策的影响。当删除学习子句时,如果此时所选择的决策变量所在的子句数目恰好大量减少,那么从优先满足较多子句的角度来考虑,这个变量并不是最优的。因此,本文提出了基于学习子句删除策略的分支变量决策策略,在选择分支变量时不仅要考虑文字参与冲突的次数,还要考虑它所在的子句的数目,使得求解器优先选择冲突次数较多且所在的子句数目也比较多的变量作为决策变量。

2 SAT 问题的求解算法

2.1 基础知识

(1)文字[18]。布尔变元集合 $X=\{x_1,x_2,\cdots,x_{n-1},x_n\}$,$x_i\in\{\text{true},\text{false}\}$中每个布尔变量 x_i 可以被赋值为 1(true)或者 0(false)。布尔变量 x_i 及其否定 $\neg x_i$ 称为文字。x_i 称为正文字,$\neg x_i$ 称为负文字。

(2)子句[18-19]。子句 $C=\bigvee_{j=1}^{m} l_j$ 是由 m 个文字析取组成的,若子句可满足当且仅当子句中至少有一个文字为 1,则当子句中所有的文字都为 0 时子句不可满足。不含任何文字的子句称为空子句,它永远不会被满足,记为□。若一个子句中只有一个文字没有被赋值,其余文字被赋值为 0(或子句只含一个文字),为了保证子句的可满足性,必须将这个文字赋值为 1,这样的子句称为单元子句。

(3)合取范式[19](Conjunction Normal Formula,CNF)是由若干子句的合取组成,如 $\bigwedge_{i=1}^{n} C_i$,记为 F。当合取范式中的每个子句都满足时,合取范式 F 才被称为可满足。

(4)归结式[20]。给定子句 $C_1=(x\vee C_1')$和 $C_2=(\neg x\vee C_2')$,其中 x 为文字。C_1'为子句 C_1 中除文字 x 外的剩余部分,C_2'为子句 C_2 中除文字 $\neg x$ 外的剩余部分,则称子句 C_1 与 C_2 基于文字 x 的归结式为 $R(C_1,C_2,x)=C_1'\vee C_2'$。

2.2 CDCL 算法

基于二叉非时序回溯搜索的 CDCL 算法[17] 的实现如算法 1 所示。通过变量决策分支函数 *Pick_Brancling_Variable*() 选择分支决策变量 *var*,对其赋值并加入变量赋值集中。如果所有变量都已赋值,且可以推出所有的字句都是满足的,则返回 SAT,否则决策层数加 1,执行 *Unit_Propagation*()单元传播函数。若单元传播的过程中发生冲突 *conflict*,则执行冲突分析函数 *Analyze_Conflict*()生成学习子句。*Computed_Level*()返回非时序回退层数 *dl*,若 *dl*==0,则说明 *F* 不可满足并返回 UNSAT。*Restar*()是重启函数,求解器在搜索过程中容易陷入局部最优的局面。为了避免这种情况发生,设置重启条件。重启时,直接回退到第一决策层,并撤销之前所有变量的赋值。但是变量的活跃值和学习子句并没有被删除。*ReduceDB*()是学习子句删除函数,2009 年,Simon 等[21] 证明了 90%的学习子句对于求解过程是无用的,因此设置删除条件 *Time_To_Reduce*(),删除那些"无用"的学习子句,以减少搜索空间,加快布尔约束传播。

算法 1 基于二叉非时序回溯搜索的 CDCL 算法

输入:CNF 公式 F

输出:F 的可满足性(SAT 或 UNSAT)

```
Δ=Ø;//Δ 表示学习子句集
ϕ=Ø;//ϕ 表示变量赋值集
1. dl=0;//dl 表示决策层数
2. while not all variable assigned do var=Pick_Beanching_Variable(F);
3.   if (ϕ|=F) then return SAT
4.   else dl++;ϕ=ϕ∪{var};
5.   While (Unit_Propagation(F,ϕ)==confilict) do Learnt_Clause=
     Analyse_Conflict(F,ϕ);dl=Compute_Leve(Learnt_Clause,ϕ);
6.     if (dl==0) then return UNSAT;
7.     Δ=Δ∪{Learnt_Clause};
8.     if (Restart ()) then dl=0;
9.     else then Back_Jump(dl);
10.    if (Time_To_Reduce ()) then ReducedDB (Δ);
11.  end
12. end
```

3 已有的分支决策策略

3.1 MOM 分支策略

最短子句出现频率最大(Maximun Occurrence in Minimiaztion,MOM)算法[14],优先满足最短长度且出现频率最高的子句。

设 $f_n(l)$是长度为 n 且包含文字 l 的子句数,$f^*(l)$是包含文字 l 且长度最短的子句数。MOM 算法选择满足如下条件的文字作为分支决策文字。

(1)$f^*(l)$最大;

(2)$f^*(\neg l)$最大;

(3)$f^*(l)$与 $f^*(\neg l)$之差尽可能小。

因为只要子句中某一文字得到满足,那么该子句就是可满足的,所以一个子句所含文字越多,那么它越容易满足。该算法在文字较少的子句集中调出出现频率较高的文字使其优先满足,目的是让不易满足的子句尽可能优先得到满足,从而加快搜索过程。

3.2 VSIDS 分支策略

变量状态独立下降和(Variable State Independent Decaying Sum,VSIDS)策略[15]是近年来主流的启发式分支策略,该算法在分支变量决策策略中具有里程碑意义,其具体步骤描述如下。

(1)给每个文字都设置一个计数器,计算文字的活性(activity)初始化为 0。

(2)每当将一个学习子句加入公式集中时,学习子句中有关文字的活性加 1。

(3)每次选择分支变量时,求解器会选择具有最大活性值的未赋值文字作为分支决策变量。若最大活性文字有多个,则随机选取一个。

(4)所有文字的计数器会周期性地乘以参数 $q(0<q<1)$,以避免求解器陷入局部最优。

该算法通过监测每个文字参与的冲突次数,当产生冲突时增加子句所包含文字的活性。该算法优先考虑了与冲突有关的文字,实验表明其求解性能较优,且时间消耗较少。

4 VDALCD 变量决策算法

变量决策策略是 CDCL SAT 求解器重要的一环,本节提出了一种新的变量决策算法,称为基于学习子句删除策略的变量决策算法(Variable Decision Algorithm Based on Learnt Clause Deletion,VDALCD)。

4.1 子句删除与决策变量之间的关系

为了方便对比,使用 Minisat 求解器对一个简单例子进行求解。随机选取实例"20191023131917244_600_3300.cnf",定义一个数组 nums[i]记录每个文字所在的子句数目,每当执行 *ReduceDB*()函数时输出包含决策变量的子句数目。

如图 1 所示,学习子句删除前后包含决策变量的子句数目波动较大,在搜索前期决策变量所在的子句数目甚至会减少一半。

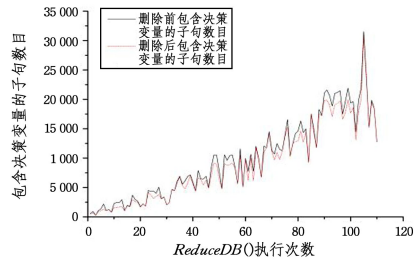

图 1 *ReduceDB*()执行前后包含决策变量的子句数目

Fig. 1 Number of clauses containing decision variables before and after *ReduceDB*() is executed

在求解器求解过程中,随着冲突次数的增加,学习子句的个数也不断增大,每当执行学习子句删除策略时,那些"无用"的学习子句将被删除,从而影响了包含决策变量的子句数目。设 v_{max} 是具有最大活性的文字,v_{second} 是具有第二大活性的文字。随机选取 2018 年 SAT 官方竞赛的 5 个实例,测试求解器采用 Glucose4.1 版本,此求解器采用的是 EVSIDS(基于 VSIDS 改进的)版本的决策策略。分析求解过程中每次学习子句删除前包含 v_{max} 和 v_{second} 的子句数目,如果包含 v_{second} 的子句数比包含 v_{max} 的子句数多,则可选择 v_{second} 为决策变量。根据 EVSIDS 策略可知,变量 v 的活性增加量会随着冲突次数的增加而变大。当冲突次数为 10 000 时,每次冲突时参与冲突的变量的活性会增加大约 1×10^{222},而实验表明 v_{max} 的活性与 v_{second} 的活性的差值一般低于 1×10^{100}。所以在后续求解过程中,v_{max} 与 v_{second} 参与冲突次数的差异并不明显,基于冲突次数的变量决策策略并不能高效地区分哪个变量对于后续布尔约束传播(Boolean Constraint Propagate,BCP)的作用更大。

表 1 列出了求解过程中学习子句删除策略的执行次数和包含 v_{max} 与包含 v_{second} 的子句数目对比,其中 $m(v)$表示包含变量 v 的子句数目,f 表示 $m(v_{second})$和 $m(v_{max})$之差与 $m(v_{max})$之比。由表 1 可得出,有大约一半的情况是 $m(v_{second})$大于 $m(v_{max})$,甚至有时 $m(v_{second})$比 $m(v_{max})$多了 30%。因此,在选择分支变量时应当将包含变量的子句数目考虑进去,综合考虑变量参与的冲突次数和所在的子句数目。

表 1 包含 v_{max} 与包含 v_{second} 的子句数目对比

Table 1 Contrast the number of clauses containing v_{max} with that containing v_{second}

Instance type	*ReduceDB*()执行次数	$f=\frac{m(v_{second})-m(v_{max})}{m(v_{max})}$	
		$f>0$	$f>30\%$
cnf_320_1120.cnf	8075	3977	315
gto_p60c239.cnf	2424	1409	780
CNP-5-1400.cnf	44	21	13
dist9.c.cnf	32	21	12
dist4.c.cnf	31	12	3

4.2 VDALCD 策略变量活性的计算方式

图 1 和表 1 表明,在学习子句删除过程中会删除大量的学习子句,其中一部分被删除的学习子句包含决策变量 x,若此时依然选取具有最大活性的文字 x 作为决策变量,那么此次决策所影响的子句较少,不利于后续的 BCP 传播。本文综合考虑了两种分支决策算法,即 MOM 和 VISDS。本文新提

出的变量决策算法重新构造了变量活性的增加方式,在删除学习子句时减少删除子句所包含文字的活性。

当删除学习子句 c_i 时,有:

$$s(v)=s(v)-p$$

$$p=\left(\frac{1}{\lambda}\right)^m$$

其中,v 为 c_i 中的文字,$s(v)$为文字 v 的活性,p 为"惩罚值",m 为学习子句删除策略执行的次数,λ 为活性减少因子。

当生成学习子句 c_i 时,有:

$$s(v)=s(v)+\left(\frac{1}{f}\right)^n$$

其中,n 为冲突次数,f 为增量因子。

因为文字活性的增加量会随着冲突次数的增加而增大,为了平衡文字活性减少量和增加量之间的关系,使 p 与学习子句删除次数有关。由 p 的定义可知,当执行学习子句删除策略时,删除的学习子句所包含的变量活性将减少,并且随着学习子句删除次数的增加,文字的活性将减少得越快。在求解过程中子句删除策略执行的次数远少于冲突次数,也就是 $m \ll n$,因此不用考虑变量活性为负值的情况。此策略折中优先选择既参与了冲突又被较多子句所包含的文字作为分支变量。

4.3 VDALCD 算法

算法 2 将活性计算方式嵌入学习子句删除策略中,在删除学习子句时降低所包含变量的活性。生成学习子句时变量活性的计算方式和原求解器变量活性的计算方式一样,变量活性随着冲突次数的增加而增大。当删除学习子句时,子句中所包含变量的活性就会得到一个"惩罚值","惩罚值"随着学习子句删除执行次数的增加而增大。

算法 2 基于学习子句删除的变量决策算法

输入:CNF 公式 F

输出:记录变量 v 的活性。

```
Δ=Ø;//Δ 表示学习子句集
Δ=Ø;//Δ 表示学习子句集
ϕ=Ø;//ϕ 表示变量赋值集
1. activity[v]=0 //记录每个变量的活性
2. Learn_Clause=Analyse_Conflict(F,ϕ)
3. for (v involved in Learn_Clause) do activity[v]=activity[v]+
   (1/f)^n //n 是冲突次数
4. end / * 当学习子句删除时降低涉及变量的活性 * /
5. when ReducedDB(Δ) do
6.    if (delete(c)) then //c 是被删除的学习子句
7.       for (v∈c) do //v 是学习子句 c 中的变量 activity[v]=activity
         [v]-p//p 为"惩罚值"
8.       end
9. end
```

传统变量决策策略在求解过程后期不能高效地区分哪个变量对于后续 BCP 传播作用更大,而本文的 VDALCD 变量决策策略考虑了学习子句删除对 BCP 传播的影响,解决了此问题。下一节将通过实验来进一步证明此算法的可行性。

5 实验结果

本文将提出的 VDALCD 分支变量策略分别嵌入到 Glucose4.1 和 MapleLCMistChronoBT-DL-v2.1(下文简称 Maple-DL)求解器中,表示为 Glucos4.1_VDALCD 和 MapleLCMDistChronoBT-DL_VDALCD(下文简称 Maple-DL_VDALCD)。Glucos4.1 求解器是国际上知名的 CDCL 求解器,近几年国际 SAT 竞赛专设一个 Glucose 组,对那些基于 Glucose 改进的求解器进行评比。MapleLCMistChronoBT-DL-v2.1 是基于 Maple 求解器改进的,是 2019 年 SAT 竞赛 main 组亚军。

5.1 实验环境

本次实验对 2018 年的例子是在 64 位 Windows7+Cygwin2.905,Intel core i3-3240CPU,8 GB 内存,3.4 GHz 处理器的环境下测试的,对 2019 年的例子是在 64 位 Windows10+Cygwin2.905,Intel core i7-7700CPU,32 GB 内存,3.6 GHz 处理器的环境下测试的。此外,Glucos4.1 和 Gluose4.1_VDALCD 的运行环境相同,Maple-DL 和 Maple-DL_VDALCD 的运行环境相同。

5.2 "活性减少因子"λ 的评估

本文第 3 节提出了新的变量活性增加方式,引入了"活性减少因子"λ。λ 的取值决定了算法的求解速度和效率,该如何选取 λ 的值呢?

本文引入 λ 的目的是在选择分支决策变量时既考虑变量参与冲突的次数,又考虑包含此变量的子句数目。综合分析选择最优的未赋值的变量作为决策变量,因此 λ 的取值应该在 f 的左右波动。当认为包含此变量的子句数目的权重大于变量参与冲突的次数时,λ 的取值应大于 f。相反,λ 的取值应小于 f。

5.3 Glucose4.1 和 Glucose4.1_VDALCD(λ)的对比分析

根据参数 λ 的取值将改进过后的 Glucos4.1 求解器记为 Glucos4.1_VDALCD(0.90),Glucos4.1_VDALCD(0.93),…。本文所有的对比实验都是在相同环境下进行的,Glucos4.1 与 Glucos4.1_VDALCD(λ)对 2018 年 SAT 竞赛实例进行测试。

表 2 列出了 Glucose4.1 与 Glucose4.1_VDALCD(λ)的结果对比,从表中数据可得出改进过后的版本优于原版本,最差的也比原版本多求出一个例子。其中,当参数 λ 取 0.99 时,Glucose4.1_VDALCD(0.99)比 Glucose4.1 多求出了 26 个例子,较原版求解器的求解实例个数增加了 15.5%,并且求解时间也减少了 28%。

表 2 Glucose4.1 与 Glucose4.1_VDALCD(λ)的对比

Table 2 Comparison of Glucose4.1 and Glucose4.1_VDALCD(λ)

Solvers	Solved(SAT-UNSAT)	Average time
Glucose4.1	168(86-82)	878.73
Glucose4.1_VDALCD(0.90)	169(93-76)	807.07
Glucose4.1_VDALCD(0.93)	171(97-74)	864.87
Glucose4.1_VDALCD(0.94)	177(102-75)	829.77
Glucose4.1_VDALCD(0.95)	190(106-84)	590.35
Glucose4.1_VDALCD(0.96)	178(99-79)	814.41
Glucose4.1_VDALCD(0.97)	189(106-83)	657.81
Glucose4.1_VDALCD(0.98)	184(105-79)	768.03
Glucose4.1_VDALCD(0.99)	194(105-89)	632.60

图 2 给出了 Glucos4.1 和 Glucos4.1_VDALCD(λ)在总体为 400 个 SAT 竞赛实例上的运行时间比较,x 轴表示求解个数,y 轴表示求解时间。曲线越接近 x 轴,表示求解器所用

的时间越少。从图 2 可以得出，参数 λ 取值 0.99 和 0.95 时，曲线走势较为接近，求解性能显著提升。因此，初步得出 VDALCD 分支策略是有效的，参数 λ 的取值应该大于活性增量因子 f 的取值。

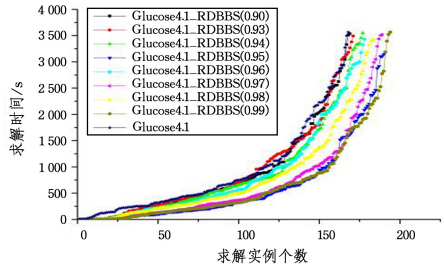

图 2 不同求解器的性能对比

Fig. 2 Comparison of solving performance of different solvers

5.4 Maple-DL 和 Maple-DL_VDALCD 的对比分析

经过初步实验确定了 VDALCD 分支策略是有效的，因此本节将其嵌入 2019 年亚军求解器 Maple-DL 中，进一步证明它的能力。5.3 节表明，参数 λ 的取值应该大于活性增量因子 f 的取值。因此，分别对 λ 为 0.90，0.95，0.96，0.97，0.98 和 0.99 的 6 个求解器版本与原版求解器进行比较， 实验结果如表 3 和图 3 所示。可以看出，嵌入 VDALCD 分支策略的求解器表现稳定，总体上 Maple-DL_VDALCD(λ)是优于 Maple-DL 的。当 λ 取值为 0.98 时比原版求解器多求解出了 17 个实例，求解个数增加了 7.6%，求解时间减少了 3.7%。

表 3 Maple-DL 与 Maple-DL_VDALCD(λ)的对比

Table 3 Comparison of Maple-DL and Maple-DL_VDALCD(λ)

Solvers	Solved(SAT-UNSAT)	Average time
Maple-DL	223(130-93)	819
Maple-DL_VDALCD(0.90)	232(135-97)	791.67
Maple-DL_VDALCD(0.95)	234(136-98)	901.32
Maple-DL_VDALCD(0.96)	236(138-98)	927.64
Maple-DL_VDALCD(0.97)	238(140-98)	789.16
Maple-DL_VDALCD(0.98)	240(142-98)	789.34
Maple-DL_VDALCD(0.99)	225(131-94)	814.85

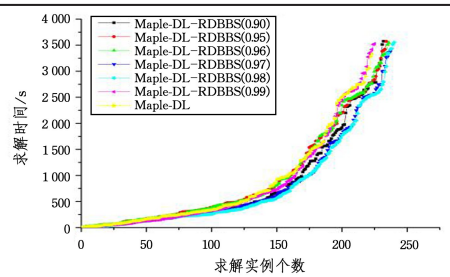

图 3 不同求解器的性能对比

Fig. 3 Comparison of solving performance of different solvers

结束语 传统的分支变量策略不考虑包含决策变量的子句个数与 BCP 传播之间的联系，实验表明，在求解过程中 90%的时间都浪费在了 BCP 传播的过程中[21]。本文综合考虑变量参与冲突的次数和包含该变量的子句数目，提出了基于学习子句删除策略的分支变量决策(VDALCD)。VDALCD 可以选择所在子句数目较多且参与冲突次数较多的未赋值文字作为分支决策变量，加快了 BCP 的传播，减少了重启次数，提高了求解效率。

本文提出的 VDALCD 策略中参数的设置是经过不断实验得出的，因此，在接下来工作中可以针对参数的动态设置进行进一步的研究。

参 考 文 献

[1] COOK S A. The complexity of Theorem-Proving Procedures [C]// Proceedings of the 3rd Annual ACM Symposium on Theory of Computing. New York: ACM, 1971: 151-158.

[2] KARP R M. Reducibility Among Combinatorial Problems [M]// Complexity of Computer Conputation. Berlin: Spring US, 1972: 85-103.

[3] HOOS H H, STYUTZLE T. Stochastic Local Search: Foundations & Applications [M]// Access Online Via Elsecier. 2004.

[4] ROBINSON J A. A Machine-Oriented Logic Based on the Resolution Principle[J]. Journal of the ACM (JACM), 1965, 12(1): 23-41.

[5] DAVIS M, PUTNAM H. A Computing Procedure for Quantification Theory[J]. Journal of the ACM(JACM), 1960, 7(3): 201-215.

[6] DAVIS M, LOGEMANN G, LOVELAND D. A machine Program for Theorem Proving[J]. Communication of the ACM, 1962, 5(7): 394-397.

[7] MARQUES-SILVA J, LYNCE I, MALIK S. Conflict-Driven Clause Learning SAT Solves [M]. Amsterdam: IOS Press, 2009: 127-149.

[8] MARQUES-SILVA J, SAKALLAH K. GRASP: A New Search Algorithm for Satisfiability [M]// The Best of ICCAD. New York: Springer, 2003: 73-89.

[9] LUO M, LI C M, XIAO F, et al. An Effective Learnt Clause Minimization Approach for CDCL SAT Solvers[C]// Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence. 2017: 703-711.

[10] SIMON L, AUDEMARD G. Predicting Learnt Clauses in Modern SAT Solver[C]// Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI'09). Menlo Park: AAAI Press, 2009: 399-404.

[11] BIEKE A. Lingeling and Friends Entering the SAT Challenge 2012[C]// Proceedings of SAT Challenge. 2012: 33-34.

[12] GLOUER F, KELLY J P, LAGUNA M. Genetic Algorithms and Tabu Search: Hybrids for Optimization[J]. Computer & Operations Research, 1995, 22(1): 111-134.

[13] BURO M, BÜNING H K. Report on a SAT Competition [M]// Fachbereich Math-Informatik. Univ. Gesamthochschule. West Germany, 1992.

[14] FREEMAN J W. Improvements to Propositional Satisfiability Search Algorithms [D]. Philadelphia: University of Pennsylvania, 1995.

[15] WEIDENBACH C, DIMOVA D, FIETZKE A, et al. SPASS Version 3.5[C]// CADE. 2009: 140-145.

[16] BIERE A, FROHLICH A. Evaluating CDCL Variable Scoring Schemes[C]// Theory and Applications of Satisfiability Testing-

SAT 2015. Switzerland:Springer USA,2015:405-422.

[17] RVAN L. Efficient Algorithms for Clause-Learning SAT Solvers [D]. Vancouver:Simon Fraser University,2004.

[18] CHANG W J. Study of Satisfiability Solving Systems and Application Based on Contradiction Separation Based Multiple Dynamic Automated Deduction[D]. Chengdu:Southwest Jiaotong University,2019.

[19] WANG G J. Introduction to Mathematical Logic and Resolution Principle[M]. Beijing:Science Press,2003.

[20] CHEN Q S,XU Y,HE X X. Heuristic Complete Algorithm for SAT Problem by Using Logical Deduction[J]. Journal of Southwest Jiaotong University,2017,52(6):1224-1232.

[21] AUDEMAND G,SIMON L. Predicting Learnt Clauses Quality in Modern SAT Solvers[C]// Proceeding of the 21st International Joint Conference on Artificial. Intelligence. San Francisco, CA:Margan Kaufmann,2009:399-404.

WANG Yi-jie, born in 1997, postgraduate. His main research interests include processing and automated reasoning.

WU Guan-feng, born in 1986, Ph.D, is a member of China Computer Federation. His main research interests include intelligent information processing and parallel computing.