CN 43-1258/TP
ISSN 1007-130X

计算机工程与科学
Computer Engineering & Science

第 41 卷第 8 期 2019 年 8 月
Vol. 41, No. 8, Aug. 2019

文章编号:1007-130X(2019)08-1374-07

基于频次的 SAT 问题学习子句混合评估算法*

吴贯锋[1,2],徐　扬[2,3],陈青山[1,2],何星星[2,3],常文静[1,2]

(1. 西南交通大学信息科学与技术学院,四川 成都 610031;
2. 西南交通大学系统可信性自动验证国家地方联合工程实验室,四川 成都 610031;
3. 西南交通大学数学学院,四川 成都 610031)

摘　要:为了有效管理学习子句,避免学习子句规模呈几何级增长,减少冗余学习子句对系统内存占用,从而提高布尔可满足性问题 SAT 求解器的求解效率,需要对学习子句进行评估,然后删减学习子句。传统的评估方式是基于学习子句的长度,保留较短的子句。当前主流的做法一个是变量衰减和 VSIDS 的子句评估方式,另外一个是基于文字块距离 LBD 的评估方式,也有将二者结合使用作为子句评估的依据。通过对学习子句参与冲突分析次数与问题求解的关系进行分析,将学习子句使用频率与 LBD 评估算法混合使用,既反映了学习子句在冲突分析中的作用,也充分利用了文字与决策层之间的信息。以 Syrup 求解器(GLUCOSE 4.1 并行版本)为基准,在评估算法与并行子句共享策略方面做改进测试,通过实验对比发现,混合评估算法比 LBD 评估算法有优势,求解问题个数明显增多。

关键词:SAT 问题;并行求解器;LBD;GLUCOSE

中图分类号:TP311;TP391　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2019.08.006

A hybrid learnt clause evaluation algorithm for SAT problem based on frequency

WU Guan-feng[1,2], XU Yang[2,3], CHEN Qing-shan[1,2], HE Xing-xing[2,3], CHANG Wen-jing[1,2]

(1. School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031;
2. National-Local Joint Engineering Laboratory of System Credibility Automatic Verification, Southwest Jiaotong University, Chengdu 610031;
3. School of Mathematics, Southwest Jiaotong University, Chengdu 610031, China)

Abstract: In order to effectively manage learnt clauses, avoid a geometrical growth of their scale, reduce the memory cost of redundant learnt clauses and improve the efficiency of the Boolean satisfiability problem (SAT) solver, we need to evaluate learnt clauses and delete some redundant ones. Traditional evaluation methods are based on the length of learnt clauses, and short-length ones are kept. Two current mainstream clause evaluation methods are the variable state independent decaying sum (VSIDS), and a method based on the evaluation of the literals blocks distance (LBD), and the combination of the above two is also used as the basis for clause evaluation. We analyze the relationship between the number of learnt clauses used in conflict analysis and problem solving, and combine the frequency of learnt clauses with the LBD evaluation algorithm, which not only reflects the role of learnt clauses in conflict analysis, but also makes full use of the information between text and decision-making layer. Taking the Syrup solver (GLUCOSE 4.1 parallel version) as baseline, experiments are carried out to evaluate the algorithm and the parallel clause sharing strategy. The experimental comparison shows that the the pro-

* **收稿日期**:2018-08-13;**修回日期**:2018-11-08
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(2682017ZT12, 2682018ZT10, 2682018CX59, 2682018ZT25)
通信地址:610031 四川省成都市西南交通大学信息科学与技术学院
Address: School of Information Science and Technology, Southwest Jiaotong University, Chengdu 610031, Sichuan, P. R. China

posed hybrid evaluation algorithm outperforms the LBD evaluation algorithm, and the number of solving problems is significantly increased.

Key words: SAT problem; parallel solver; LBD; GLUCOSE

1 引言

SAT(SATisfiability)问题是逻辑学和计算机科学的基本问题,也是第一个被证明了的NP完全问题[1]。自然科学中的很多问题,如交通运输,集成电路自动布线/软件的自动生成和程序正确性验证等许多问题都可转化为SAT问题进行求解[2-4]。研究高效的SAT问题求解算法,具有很重要的理论和现实意义。

SAT求解算法又可分为完备算法和不完备算法2种。完备算法在理论上一定能够判定出公式的可满足性,而不完备算法则无法保证能够找到问题的解。本文所涉及的求解器及算法均指完备算法。

从1960年DPLL(Davis-Putnam-Logemann-Loveland)算法的提出[5,6]到冲突子句学习CDCL(Conflict-Driven Clause Learning)算法提出以及不断改进[7-10],SAT求解器经过几十年的发展,组成模块趋于稳定。主流的SAT求解器(MiniSAT、GLUCOSE、MapleCOMSPS、Kiel等)都是基于DPLL和CDCL算法框架,主要包含预处理、决策变量选择、布尔约束传播BCP(Boolean Constraint Propagation)、CDCL冲突子句学习、分支回溯、重启、子句管理模块[11~16]。

基于CDCL的SAT求解器,在求解过程中会产生大量的学习子句,以SAT竞赛2017年Main Track组中的g2-ak128modbtbg2msisc.cnf为例,文字数296 051,子句数968 713,在GLUCOSE求解过程中产生了16 007 084条学习子句,在求解过程中被删除的学习子句有15 204 538条,删除比例达95%。过多的学习子句不但占用内存资源,而且还增加运算过程中子句遍历的时间复杂度,不利于问题求解。

GRASP[7]和Chaff[8]以学习子句长度和未赋值的变量个数作为子句评估条件,虽然初期解决了学习子句过度增长问题,但评估方式过于简单,并未反映学习子句在搜索中的作用。

Minisat最先采用VSIDS(Variable State Independent Decaying Sum)评估方式,为子句设置活跃度[13],并根据活跃度为子句排序,周期性地删除一半活跃度低的学习子句,评估策略被后续多个求解器采用或改进[17]。基于VSIDS的评估算法类似于生物的用进废退规则,在求解的初期,无法确切判断子句的重要程度,未被使用的子句因活跃度衰减而被删除,一些对后期搜索有用的子句也可能会被误删除。

为了全面评估学习子句,避免出现VSIDS中部分学习子句持续被保留现象。文献[11]在求解器GLUCOSE中提出了一套新的评估算法,以LBD(Literals Blocks Distance)的值作为保留学习子句的依据[11],刻画了学习子句的复杂程度。

VSIDS和LBD属于激进的学习子句评估方式,对学习子句的特征描述比较单一,在周期性删除学习子句时,有可能会误删除对搜索起桥梁作用的一些子句。Lingeling就采用了VSIDS与LBD混合的评估方式,在判断LBD学习子句分布不均衡时,切换到VSIDS的评估方式[18]。

虽然SAT求解发展迅速,可处理的问题规模已经达到千万级,但是依然无法满足实际应用需求。2017年SAT竞赛共350个基准例(SAT竞赛中很大一部分问题来自于实际的工业应用问题),冠军求解器在规定时间内只求解出208个,仍然有142个问题未能求解[19]。

并行求解器从一定程度上提高了SAT问题求解效率,但是受限于并行的硬件环境等因素,本质上还是要研究有效的SAT问题求解算法或改进现有SAT求解算法。

本文将简化的基于学习子句趋势强度的评估方法与LBD评估算法结合,提出混合子句评估算法,更精确地描述学习子句的特征。对Syrup评估算法、并行节点间子句共享策略进行了改进。

2 相关工作

2.1 基于VSIDS的评估算法

VSIDS评估算法是SAT问题求解中的经典算法[13],其过程描述如下:

算法为每个原始子句设置活跃度属性,初始值设置为1。变量 cla_inc 为活跃度增量,初始值为1/0.999。

在搜索过程中发生冲突时,若子句被用于冲突

分析,则其活跃度值 $activity = activity * cla\_inc$。当活跃度值超过 10^{20} 时,以 10^{-20} 缩放所有子句的活跃度值,并缩放 cla_inc 为 $cla\_inc * 10^{-20}$。

每次冲突分析结束时,调整活跃度增量 cla_inc 为 $cla\_inc * (1/0.999)$。

2.2 基于 LBD 的子句评估算法

LBD 最早是由 Audemard 等人[11]在 GLUCOSE(2011 年 SAT 竞赛 Main Track 组冠军)的早期版本提出来的,用以评估学习子句的质量。*LBD* 的值表示一个子句中的文字所在不同决策层的个数。在同一个决策层的文字被认为是一个块,一个学习子句把不同块上的文字联系在了一起。

假设有如下 2 条学习子句:

$$C_1 = x_1(6) \vee x_3(6) \vee x_4(3) \vee x_5(6)$$

$$C_2 = x_7(1) \vee x_8(5) \vee x_9(7)$$

$x_1(6)$表示变量 x_1 在第 6 层决策时被赋值。则子句 C_1 中变量的决策层集合为{3,6},其 *LBD* 值为 2。同理子句 C_2 的变量来自 3 个决策层,其 *LBD* 值为 3。

Gilles 把 *LBD* 值为 2 的子句叫做'glue'子句,如上述学习子句 C_2。因为这样的子句中一个文字来源于冲突发生的决策层,其他文字来自另外一个决策层。'glue'子句被认为是最好的,最应该保留的子句。GLUCOSE 周期性地删除 *LBD* 值大于 2 的子句,用以减少学习子句的规模。

2.3 混合评估算法

混合评估算法是将已有知名评估算法混合使用。

Lingeling 采用内外 Inner-Outer 方案管理学习子句,又叫做缩减方案 reduce schedule。Inner 方案中采用 LBD 的评估方式,当'glue'子句分布不均衡时,动态切换到 VSIDS 的子句评估方式。Outer 方案采用 Luby[20]序列控制学习子句总数,这样的方案被证明在精心构造的例子求解中十分有效[18]。

MapleComSPS 求解器突出的贡献是在决策变量选择上,提出了学习比率分支策略 LRB(Learning Rate Branching heuristic)算法和基于历史冲突的分支策略 CHB(Conflict History Based branching heuristic)算法[21,22],通过这 2 个算法选择的决策变量能够产生高质量的学习子句,同时,在对学习子句的管理方面,MapleCOMSPS 也采用了 LBD 与 VSIDS 混合的子句删除策略,在运算的前 2 500 s 用 LBD 的方式评估子句,随后采用 VSIDS 活跃度的方式评估学习子句[14]。

文献[23]提出以学习子句演绎长度作为评估指标与 LBD 评估算法混合使用,在 GLUCOSE 3.0版本上做改进,取得了不错的效果。

3 基于学习子句使用频次与 LBD 混合评估算法

3.1 基于学习子句使用频次的评估算法

我们通过对 2015 年和 2016 年 SAT 竞赛中的实例进行测试分析发现,基于 VSIDS 评估方式平均删除了 84%的学习子句,LBD 评估方式删除了 80%的学习子句。最先生成的学习子句在后续搜索过程中所占比重越来越少。VSIDS 和 LBD 的子句评估方法都采用比较严格的删除策略,学习子句的利用率不高。

VSIDS 的活跃度设置是针对所有子句的,虽然包含有学习子句,但是区别性不强。

基于学习子句使用频次的评估策略将学习子句用于冲突分析的次数量化,相关定义如下所示:

定义 1 频次上限 *LN*(Limit Num)表示基于频次的评估算法在删除学习子句时所采用的分界值,低于该值的子句将不被保留。

定义 2 使用频次 *UC*(Used Count)表示学习子句 *C* 被应用于冲突分析的计数值,取值为[0, *LN*]。

为了研究学习子句使用频次与子句 *LBD* 值之间的关系,以 GLUCOSE 为基础,加入学习子句在冲突分析中的使用次数计数功能。数据结构方面在 Clause 结构体中加入 *UC* 成员,每次学习子句参与冲突分析时,*UC* 计数器值加 1。统计输出所有学习子句的 *LBD* 值与使用频次分布情况。同样以 g2-ak128modbtbg2msisc. cnf 为例,在 GLUCOSE 共执行 94 次周期性的删除操作,实验结果中 *LBD* 值大于 50 的子句占总学习子句数不足 3%,且被使用次数趋近于 0。如图 1 和图 2 所示为执行周期性删除策略前后学习子句 *LBD* 与使用次数以及对应子句数之间的分布曲线变化(截取 *LBD* 小于或等于 50 的数据段)。为了分析不同量级的曲线之间的关联关系,采用双 Y 轴表示,左侧 Clause Count 表示对应 *LBD* 值的学习子句总数;右侧 Used Times 表示对应 *LBD* 值的所有学习子句被使用次数总和。

从图 1 和图 2 可以看出,无论是否周期性删除学习子句,学习子句使用次数与 *LBD* 值直接相

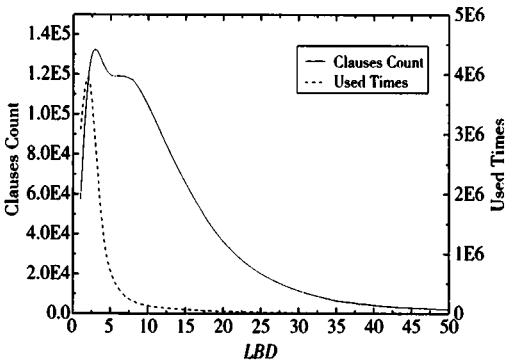

Figure 1 Distribution curve not using periodic deletion

图1 不执行删除策略时的分布曲线

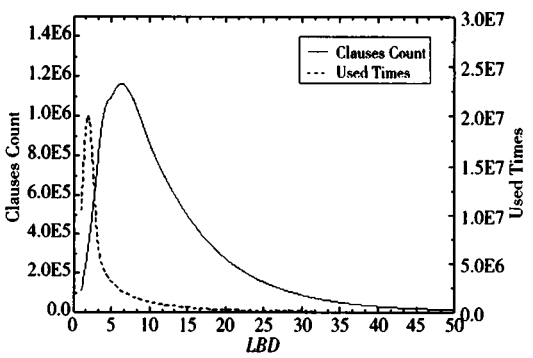

Figure 2 Distribution curve using periodic deletion

图2 执行周期删除策略时的分布曲线

关。从学习子句*LBD*值分布上看,绝大部分的学习子句集中在*LBD*值在[1 25]上。*LBD*值为2的子句使用频率最高,且*LBD*值越大,使用频率越低。

3.2 基于频次与LBD混合评估算法

基于频次与LBD混合的评估算法,在LBD评估算法基础上加入学习子句被应用于冲突分析频次计数功能,简称H-LBD算法。

LBD算法描述了学习子句的构成特征,*LBD*值越大,表示子句构成越复杂,跨越的决策层次越多。*LBD*值越小,表示学习子句内聚性越好。基于LBD的评估方法已被众多求解器证明了其有效性,因此混合算法应保留LBD算法部分在搜索引导方面的优势。

子句评估算法的主要作用在于评价学习子句,为删除学习子句压缩内存占用提供依据。

按照定义1和定义2的描述,H-LBD算法为每个学习子句设置学习子句使用频次计数器,并在算法运行过程中采用一个全局变量作为频次上限值。

对应于搜索模块,在每次发生冲突分析时,更新学习子句被用于冲突分析的频次计数值。具体描述见算法1中的步骤2。

算法1 更新学习子句被应用于冲突分析的频次

步骤1 执行布尔约束传播算法,若发生冲突,分析冲突产生原因,获取所有参与冲突分析的子句,执行步骤2。若不存在冲突且未找到可满足赋值时,执行步骤4。

步骤2 对于每个参与冲突分析的子句C,若C是学习子句,且使用频率未达到预设值,则子句使用频率累加1后执行步骤3;若C不是学习子句或使用频率已达到预设值,不执行任何操作,转步骤3。

步骤3 分析冲突产生原因,生成新的学习子句,将学习子句加入到子句集中。

步骤4 通过启发式策略选择新的赋值文字,若能选择出新的未赋值文字,则继续转步骤1执行;若所有文字均已赋值,所有子句被满足,则返回可满足,算法结束;若搜索过程回退到顶层决策层未找到可满足例,则返回不可满足,算法结束。

对应于周期性删除学习子句模块,算法2根据学习子句的使用频率以及是否被应用于单文字传播标识决定是否删除该学习子句(学习子句被应用于单文字传播过程则后续学习子句由此推导产生,因此被应用于单文字传播过程的学习子句不能在当前周期内删除,设置标记在下一周期中删除)。

算法2 使用混合评估算法删除学习子句

步骤1 设置删除个数上限*limit*为学习子句总数的一半;

步骤2 遍历所有学习子句,判断是否达到删除上限,若未达到终止条件,则执行步骤3,否则执行步骤4;

步骤3 对于任意学习子句C,若C被使用次数低于限制阈值且长度大于2,且未用于单文字子句传播过程,则删除学习子句C,否则标记学习子句,在下一周期中删除;

步骤4 压缩学习子句存储空间,执行内存回收,算法结束。

4 并行求解器中改进的子句共享策略

4.1 并行节点间子句分享策略

4.1.1 共享的数据结构

Syrup本质上是基于多线程的并行求解器,所有子线程共享学习子句,通过锁的方式实现线程间的数据同步操作。依靠MultiSolvers类管理ParallelSolver类,维护并执行多个线程并行求解。同

时依赖于 SolverCompanion 及其子类 ShareComparion 进行学习子句共享的管理,UML 类图如图 3 所示。

4.1.2 学习子句的导出与导入

Syrup 在处理并行节点间的子句分享时比较慎重,其认为 *LBD* 值小于 8,子句长度小于 40 的子句是"好的"子句,应该被分享,其次,为了避免节点之间在求解的初期相互干扰,在每个节点的冲突次数未达到 5 000 次的情况下,禁止分享学习子句。

其执行懒散的子句分享策略,在子句被分享前,先观察其被应用于冲突分析的次数,只有在冲突分析中出现过的子句才可以被共享。与串行版本中策略一致,限制分享子句 *LBD* 值小于或等于 8、子句长度小于 40 的子句,单元子句则直接被分享。

Syrup 在搜索过程中如果冲突次数已经达到 5 000次,则可执行外部学习子句的导入操作。首先导入一元子句,然后导入其它满足评估条件的学习子句。

4.2 混合评估方式下的子句分享策略

在并行节点的 search 函数执行过程中,当赋值发生冲突时,analyze 函数分析冲突产生原因并获得学习子句。如果学习子句是单元子句则直接分享到共享内存队列,根据子句共享策略判断是否共享当前学习子句。算法 3 为混合评估方式下的子句分享策略,以子句被使用次数限制作为条件代替原有的部分 LBD 限制条件。

算法 3 混合评估方式下的子句共享策略

输入:冲突学习子句 C。

输出:无。

步骤 1 判断当前子句 *LBD* 值是否小于或等于 2,若是,直接进入步骤 3;否则执行步骤 2。

步骤 2 判断子句被使用次数 UC 是否大于预设限制值 LN 且子句长度是否小于 40,若是,则执行步骤 3;否则转步骤 5,算法结束。

步骤 3 从当前线程导出学习子句 C 到全局共享内存队列中,等待被其他线程获取。

步骤 4 将该学习子句 C 标识为"已导出",防止被重复导出。

步骤 5 算法结束。

实验中学习子句使用次数限制条件 LN 需根据实验获取的经验值设置。

当前线程将优质学习子句导出到共享内存队列中,当其他线程中的 CDCL 算法执行重启操作时从共享队列中获取学习子句。

5 实验与分析

GLUCOSE 是国际知名的 CDCL 求解器,其作者最主要的贡献是提出 LBD 评估算法并成功应用在求解器上,在 SAT 竞赛中多次获奖。2016 年 SAT 竞赛在 Main Track 组外还专门设置了最佳 GLUCOSE 改进组奖项。本文以 2017 年并行组冠军 Syrup(GLUCOSE 4.1 的并行版本)为基础版本,采用混合子句评估方法分别对 *reduceDB*()函数和并行节点间子句共享策略进行改进。

测试机软硬件环境与参与实验对比分析的求解器版本说明分别如表 1 和表 2 所示。

Table 1 Operation environment

表 1 运行环境

操作系统	编译平台	CPU	内存
Windown7 x64	Cygwin64 GCC6.4.0	Intel ® Core™ i7-4790 CPU @3.60 GHz 3.6 GHz	16 GB

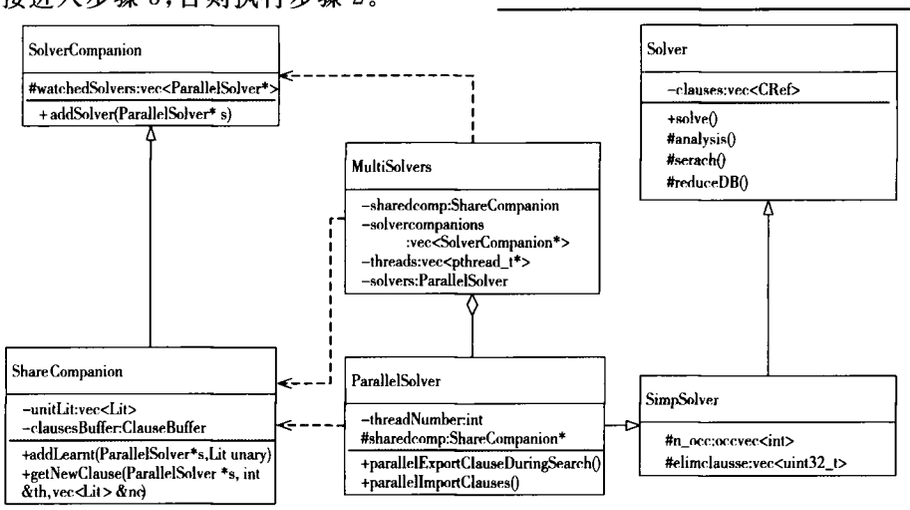

Figure 3 Class diagram of Syrup

图 3 Syrup 相关类图

Table 2 Description of SAT solvers

表 2 实验中的求解器说明

求解器	说明
Syrup	GLUCOSE 并行原版
Syrup-H	加入混合子句评估策略,未改变并行子句分享策略
Syrup-HE	加入混合子句评估策略与并行子句共享策略

实验以 2017 年 SAT 竞赛 Main Track 组 350 个基准例作为测试例,限时 3 600 s,所有测试机保证软硬件环境一致。根据前期实验分析结果显示,*LN* 值取 150 时效果最好,即在规定时间内求解问题总数最多。并行版本中采用相同数值,每个测试例运行 5 次,取平均值作为统计结果。

实验中,Syrup 配置与其在 SAT 竞赛中的配置相同,默认运行 24 个线程并行求解。为了分析混合子句评估与子句分享策略对求解结果的影响,设置了不同的组合方式构成不同的并行求解器。参与对比分析的并行求解器均以 Syrup 为前缀命名,详细说明如表 2 所示。

并行求解实验结果如表 3 所示。以求解问题个数为评价指标。从表 3 中的运行结果可以看出,加入混合子句评估策略的 2 个求解器求解问题的个数均比原版 Syrup 要多,这也证明了混合学习子句评估方式的有效性;另一方面,添加混合子句分享策略的版本 Syrup-HE 与单纯混合子句评估算法的版本 Syrup-H 相比,求解问题总数虽然少了 1 个,但是在求解不可满足问题上共有 100 个,比其他 2 个版本都多。表 3 中平均用时为所有问题的平均求解时间。

Table 3 Result comparison of parallel SAT solvers

表 3 并行求解结果对比

求解器	SAT	UNSAT	合计	平均用时/s
Syrup	97	95	192	634.031
Syrup-H	99	99	**198**	707.568
Syrup-HE	97	100	**197**	713.459

图 4 表示并行求解个数与时间分布关系,3 个版本在前 100 个问题的时间分布上高度重合,因此横坐标从 100 开始。

从图 4 的曲线形态上分析得出,改进版本 Syrup-HE 与 Syrup-H(图中三角标注实线和虚线)大部分区间低于代表原始版本 Syrup 的曲线,特别是在最后几个较难问题的求解上,代表 Syrup-HE 的带三角标注实线明显低于其余线条,表示所用时间比其他几个求解器要少。虽然求解问题总个数增加了 5 个,但是集中在较难的问题上,每个问题用

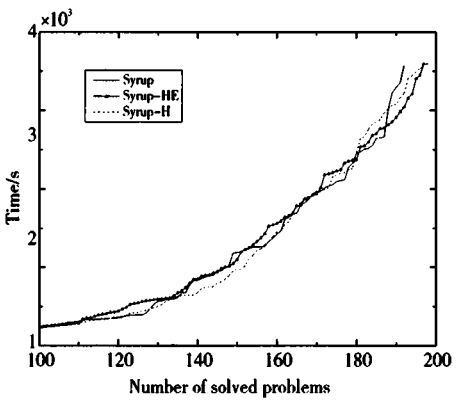

Figure 4 Relationship used time and the number of parallel solved problems

图 4 并行求解问题个数与时间分布关系

时接近 3 600 s,因此拉高了所有问题的平均求解时间,这也是表 3 中改进版本平均用时比原版高的原因。

取 Syrup 与 Syrup-HE 做进一步分析,GLUCOSE、Syrup 与 Syrup-HE 求解问题交叉部分个数为 166 个。剔除 2 条时间差异比较大(加速比大于 100)的噪声数据后,Syrup 的平均加速比为 1.778 63,Syrup-HE 的平均加速比为 1.894 48。在相同线程数目下,求解相同的问题,Syrup-HE 所获得的加速比要比原版程序 Syrup 的高。因此,采用混合评估方式的改进版本不仅增加了求解问题的个数,同时也提高了求解速度。

考虑到机器性能的影响,改变并行线程数,则 Syrup 与 Syrup-HE 的执行结果如表 4 所示。2 个求解器均在线程个数为 8 的时候达到求解个数的峰值,然后求解个数开始随着线程个数的增加回落,基于混合评估算法的求解器回落趋势较基于 LBD 评估方式的 Syrup 有所减慢,求解个数比 Syrup 多,如表 4 中的粗体显示部分所示。与此同时,虽然在线程个数为 8 时,Syrup 与 Syrup-HE 求解问题的个数完全相同,但是求解的问题并不完全相同。Syrup-HE 求解的问题中有 7 个 Syrup 并未求解出来。

Table 4 Result under different number of threads

表 4 不同线程下求解结果对比

Threads Numbers	Syrup			Syrup-HE		
	SAT	UNSAT	SUM	SAT	UNSAT	SUM
4	95	110	205	88	106	194
8	97	111	208	97	111	**208**
12	97	106	203	95	106	201
16	95	103	198	93	107	**200**
20	97	99	196	94	102	**196**
24	97	95	192	97	100	**197**

6 结束语

本文采用混合学习子句评估算法对 GLUCOSE 的并行版本 Syrup 进行改进。实验对比发现,基于频次与 LBD 混合学习子句评估方式,在一定程度上能够达到并超过基于 LBD 评估算法的 SAT 求解器的求解效率。不足之处在于混合评估算法中的参数设置是基于统计分析得到的,其内在关系还需要进一步研究。

后续可以考虑针对具体的问题类型,分析学习子句使用频率与问题本身的关系,针对题类型设置不同的参数,以争取在规定时间内求解出更多的问题。

参考文献:

[1] Cook S A. The complexity of theorem proving procedures [C]// Proc of the 3rd Annual ACM Symposium on Theory of Computing,1971: 151-158.

[2] Clarke E,Biere A,Raimi R,et al. Bounded model checking using satisfiability solving [J]. Formal Methods in System Design,2001,19(1):7-34.

[3] Vizel Y, Weissenbacher G, Malik S. Boolean satisfiability solvers and their applications in model checking [J]. Proceedings of the IEEE,2015,103(11): 2021-2035.

[4] Kuper L,Katz G,Gottschlich J,et al. Toward scalable verification for safety-critical deep networks [J]. arXiv preprint arXiv:1801.05950,2018.

[5] Davis M,Putnam H. A computing procedure for quantification theory [J]. Journal of the ACM (JACM),1960,7(3): 201-215.

[6] Franco J,Paull M. Probabilistic analysis of the Davis Putnam procedure for solving the satisfiability problem [J]. Discrete Applied Mathematics,1983,5(1): 77-87.

[7] Marques-Silva J P, Sakallah K A. GRASP: A search algorithm for propositional satisfiability [J]. IEEE Transactions on Computers,1999,48(5):506-521.

[8] Moskewicz M W,Madigan C F,Zhao Y,et al. Chaff: Engineering an efficient SAT solver[C]// Proc of the 38th Annual Design Automation Conference,2001: 530-535.

[9] Eén N,Sörensson N. An extensible SAT-solver[C]// Proc of International Conference on Theory and Applications of Satisfiability Testing,2003: 502-518.

[10] Marquessiva J, Lynce I, malik S, et al. Conflict-driven clause learning SAT solvers [J]. Frontiers in Artificial Intelligence and Applications,2009, 185(1)131-153.

[11] Audemard G,Simon L. GLUCOSE: A solver that predicts learnt clauses quality [C]// Proc of the SAT Competition, 2009: 7-8.

[12] Ehlers T. SAT and CP-parallelisation and applications [D]. [S. l.]:Universitätsbibliothek Kiel,2017.

[13] Sorensson N,Een N. Minisat v1. 13-a sat solver with conflict-clause minimization [C]// Proc of SAT Competition, 2005:1-2.

[14] Liang Jia-hui,Oh C,Ganesh V,et al. Maple-COMSPS,MapleCOMSPS LRB,MapleCOMSPS CHB [C]// Proc of the SAT Competition,2016:52-53.

[15] Guo Ying,Zhang Chang-sheng,Zhang Bin. Research advance of SAT solving algorithm [J]. Computer Science,2016,43(3):8-17. (in Chinese)

[16] Cheng Rui,Zhou Cai-lan,Xu Ning,et al. Comprehensive analysis of restart strategies of CDCL SAT solver [J]. Journal of Computer-Aided Design & Computer Graphics,2018, 30(6): 1136-1144. (in Chinese)

[17] Biere A. PrecoSAT@ SC'09[Z]// 2009 Competitive events booklets. SAT, 2009: 41-43.

[18] Biere A. Lingeling and friends entering the SAT Challenge 2012[C]// Proc of SAT Challenge,2012: 33-34.

[19] SAT competition 2017[DB/OL]. [2016-12-20]. https://baldur.iti.kit.edu/sat-competition-2017/index.php.

[20] Luby M,Sinclair A,Zuckerman D. Optimal speedup of Las Vegas algorithms[C]// Proc of the 2nd Israel Symposium on Theory and Computing Systems,1993: 128-133.

[21] Liang Jia-hui, Ganesh V, Poupart P, et al. Learning rate based branching heuristic for SAT solvers[C]// Proc of International Conference on Theory and Applications of Satisfiability Testing,2016: 123-140.

[22] Liang J H,Ganesh V,Poupart P,et al. Exponential recency weighted average branching heuristic for SAT solvers[C]// Proc of the 30th AAAI Conference on Artificial Intelligence, 2016: 3434-3440.

[23] Chang Wen-jing,Xu Yang,Wu Guan-feng. Learned clauses reduction strategy based on length of deduction [J]. Computer Engineering and Applications,2018,54 (16):30-36. (in Chinese)

附中文参考文献:

[15] 郭莹,张长胜,张斌.求解 SAT 问题的算法的研究进展[J].计算机科学,2016,43(3):8-17.

[16] 程睿,周彩兰,徐宁,等.CDCL SAT 求解器的重启策略分析[J].计算机辅助设计与图形学学报,2018,30(6): 1136-1144.

[23] 常文静,徐扬,吴贯锋.基于演绎长度的学习子句删除策略[J].计算机工程与应用,2018,54(16):30-36.

作者简介:

吴贯锋(1986 -),男,河南新密人,博士生,CCF 会员(95465G),研究方向为智能信息处理和并行计算。**E-mail**:wuguanfengfeng@126.com

WU Guan-feng, born in 1986, PhD candidate,CCF member(95465G),his research interests include intelligent information processing, and parallel computing.