基于趋势强度的 SAT 问题学习子句评估算法

陈青山[1,2] 徐 扬[2] 吴贯锋[1,2]
(西南交通大学信息科学与技术学院 成都 611756)[1]
(系统可信性自动验证国家地方联合工程实验室 成都 610031)[2]

摘 要 针对命题逻辑公式求解过程中难以有效评估学习子句是否有利于后续搜索的问题,提出了一种基于学习子句趋势强度的评估算法。该算法首先通过分析学习子句在生存期内参与冲突分析的时间分布特征,将随机、离散的时间分布转换为连续的累积趋势强度;然后在删除周期达到时,通过设定趋势强度阈值删除在后续搜索过程中"不大可能"被使用的子句,保留"可能"被使用的子句;最后采用 2015 年、2016 年 SAT 问题国际竞赛实例,将该算法与经典的活跃度评估算法和文字块距离(LBD)评估算法进行对比。实验结果表明,趋势强度评估算法在效率上明显优于活跃度评估算法,且求解的实例更多,同时与 LBD 算法基本持平。

关键词 命题逻辑,趋势强度,学习子句,子句评估,周期性删除
中图法分类号 TP301.6 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.12.021

Learnt Clause Evaluation Algorithm of SAT Problem Based on Trend Strength

CHEN Qing-shan[1,2] XU Yang[2] WU Guan-feng[1,2]
(School of Information Science and Technology, Southwest Jiaotong University, Chengdu 611756, China)[1]
(National-Local Joint Engineering Laboratory of System Credibility Automatic Verification, Chengdu 610031, China)[2]

Abstract For the problem that it is difficult to effectively evaluate whether the learnt clause is beneficial to the following search in the solving process of propositional logic equation, a clause evaluation algorithm was proposed based on the trend strength. First of all, the distribution characteristic of time involved in conflict analysis for the learnt clauses during the lifecycle is analyzed, and the random, discrete time distribution is transformed into the continuous cumulative trend strength. Then, when the deletion period arrives, the clauses with little possibility of being used in the subsequent search process will be deleted by setting the threshold of trend strength, while the clauses of high probability will be kept. Lastly, by using SAT international testing examples in 2015 and 2016, two state-of-the-art algorithms (i.e., activity evaluation algorithm and literal block distance (LBD) algorithm) are adopted for comparison purpose. The experimental results show that the proposed evaluation algorithm with trend strength can significantly outperform the activity evaluation algorithm in efficiency and can obtain more solving instances, and has the comparable performance with the LBD algorithm.

Keywords Propositional logic, Trend strength, Learnt clause, Clause evaluation, Periodical deletion

1 引言

布尔可满足性(Satisfiability Problem, SAT)问题是首个被证明的 NP(Non-deterministic Polynomial)完全问题[1]。工业领域中的很多问题(如集成电路验证、模型检验、软件验证等)都可以被转化为 SAT 问题来求解。传统的 SAT 求解器大多基于 DPLL(Davis Putnam Logemann Loveland)算法框架。近年来,基于冲突驱动的子句学习算法[2](Conflict-Driven Clause Learning, CDCL)得到了快速发展,显著提升了 SAT 求解算法的能力,使得将相应求解器广泛应用于工业领域的实际问题求解成为可能。

当 SAT 求解过程发生冲突时,通过冲突分析可以得到等价于当前冲突空间的子句,称其为学习子句。学习子句是逻辑推理得到的约束子句,将其放入原始公式中不会改变问题的逻辑属性。然而,受计算机内存条件、CPU 运行速度的限制,演绎结果无限制地增长将导致内存溢出、搜索效率降低。因此,有必要对学习子句进行甄别以决定是保留还是删除。"好"的学习子句无论是对于可满足公式的求解,还是不可满足公式的判定都具有至关重要的加速作用。但无论从理论还是实验的角度,都很难评估学习子句的"好坏",即难以有效评估学习子句与后续搜索的相关性。本文通过分析删除周期点上的学习子句的来源分布,提出一种不同于已有评估算法的

到稿日期:2017-11-07 返修日期:2018-01-04 本文受国家自然科学基金项目(61673320,11526171,61305074),中央高校基本科研业务费项目(2682017ZT12)资助。

陈青山(1982—),男,博士生,主要研究方向为智能信息处理,E-mail:qschen@home.swjtu.edu.cn(通信作者);徐 扬(1956—),男,博士,教授,博士生导师,主要研究方向为自动推理;吴贯锋(1986—),男,博士生,主要研究方向为智能信息处理。

动态趋势评估算法。与经典的活跃度(VSIDS)和文字块距离(Literals Blocks Distance,LBD)评估方法相比,本文所提算法显著优于活跃度方法,与 LBD 方法基本持平。

2 相关研究

学习子句与原始公式中的子句的性质,都可以用于后续搜索。由于 SAT 问题的复杂性,学习子句对于后续搜索是否具有决定性作用,在生成学习子句时是未知的。现有的评估方法主要分为静态评估和动态评估两类。

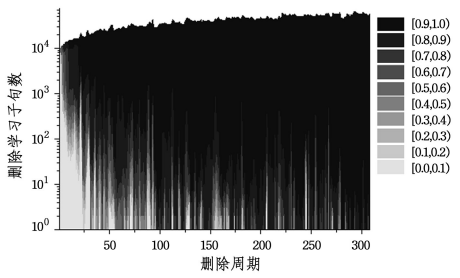
静态评估通常与搜索冲突无关。文献[3]提出了 ith-order 方法,该方法删除了文字个数超过 i 的子句。在 GRASP[4] 方法中,设定了一个临界值 k,子句中文字数小于或等于 k 的学习子句或大于 k 的单元子句都会被保留,其余学习子句则尽可能地被删除。iRelevance-bounded 算法[5] 只保留未赋值文字数不超过 i 且蕴涵有其他文字的子句。Chaff 求解器[6] 提出了一种懒散的子句删除策略,当学习子句中未被赋值的文字数首次超过 N 个(一般为 100～200)时,该子句将被打上标记,并在清理内存时被统一删除。Berkmin 求解器[7] 认为越晚被推导出的学习子句越具价值,因为从原问题中推导出这些子句耗费了更长的时间。Berkmin 求解器使用队列(先进先出结构)来维护学习子句,每次删除总学习子句队列中前 1/16 的子句,剩下 15/16 子句中超过 42 个文字的子句也会被删除;同时,最后一次冲突得到的子句和活跃的子句均不受这一限制。

动态评估体现了学习子句与搜索过程的相关性,是目前主流的评估方法。MiniSat 求解器[8] 为每个学习子句设置活跃度,当学习子句产生或被用于冲突分析时,增加其活跃度,周期性删除文字数大于 2 且排在队列前 1/2 或者活跃度小于临界值的子句。Audemard 等[9] 发现子句中某些文字是在同一决策层被赋值的,通常在其他搜索中也有很大可能性在同一决策层被赋值。基于这一思想,他们提出了文字块距离评估方法,该方法统计子句中文字所在的决策层个数 LBD,并周期性删除超过预设阈值的子句。文献[10]提出了一种基于相位保存的学习子句相关度评估算法(Progress Saving based quality Measure,PSM),通过动态比较子句中已赋值文字比例与移动汉明距离的大小,来决定子句是否应该被冻结、激活或删除。文献[11]提出了回溯层活跃度评估算法(BackTrack Level,BTL)。BTL 记录了学习子句中回溯层的大小,通过实验发现 BTL 小于 3 的子句包含了更多搜索树顶部中的文字,且被使用的频率远大于其他子句。因此,BTL 越小的子句应具有更大相关度并被保存下来。文献[12]提出了有界大小随机策略(Size-Bounded Randomized,SBR),保留短子句(文字数小于 k)并随机删除文字数大于 k 的子句,实验表明,随机保留部分较长的子句对于部分 SAT 问题的推导证明是有效的。文献[13]提出了一种基于群落结构检测学习子句关联度的方法,通过构造 2 群落得到的学习子句能够较好地提升求解性能。文献[14]综合了多种子句评估方式,提出了折中度(Degree of Compromise)概念,通过比较折中度,删除受支配的学习子句。

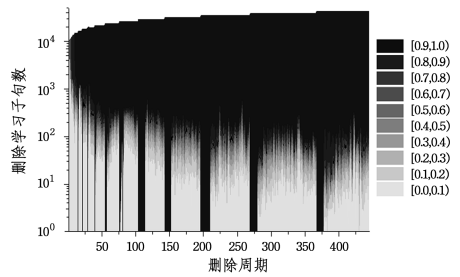
3 删除周期点上的学习子句来源分布

学习子句的"好坏"依赖于不同的评估方法,动态活跃度评估和 LBD 评估是目前最为流行的两种评估方式。前者为每个子句设置一个活跃度计算器,子句活跃度为 $s'=s+(1/f)^n$,其中,f 为衰减因子,$0<f<1$;n 为总冲突次数。子句活跃度与总冲突次数有关,与子句中文字所在决策层无关。而 LBD 评估方式正好与活跃度相反,其与决策层按文字划分的个数有关,与总冲突次数的关系不大。例如给定子句 C_1 和 C_2,用 $\mathscr{A}(C)$ 和 $\mathscr{L}(C)$ 分别表示其活跃度和 LBD,有 $\mathscr{A}(C_1)=1000$,$\mathscr{A}(C_2)=2000$,$\mathscr{L}(C_1)=3$,$\mathscr{L}(C_2)=5$。对于 Minisat 而言,C_2 的活跃度更大,显然比 C_1 好;而对于 Glucose 而言,C_1 仅被划分为 3 块,少于 $\mathscr{L}(C_2)$,因此 C_1 比 C_2 好。

不同的评价方式也使得不同的学习子句被删除,且使得搜索过程出现明显差异。但无论是活跃度评估还是 LBD 评估,都采用了比较激进的删除策略,使得学习子句的利用效率不高。图 1 所示为活跃度评估和 LBD 评估方式删除的学习子句的来源分布,图 1(a)和图 1(b)分别为活跃度评估和 LBD 评估测试 50bits_13. dimacs. cnf(SAT Race 2015)实例的情况,其中横坐标为删除周期 k,纵坐标为相应删除的学习子句数,不同颜色表示第 k 次删除的学习子句的来源 d,d 越大(颜色越深)表示越晚产生的学习子句的占比越大。可以明显看出,活跃度评估方式和 LBD 评估方式都删除了大量新学习到的子句,且保留的前期学习到的子句越来越少。相较活跃度评估方式,LBD 删除学习子句的方式相对谨慎,保留的前期学习子句相对较多。



(a)活跃度评估方式



(b)LBD 评估方式

图 1 活跃度评估和 LBD 评估方式删除学习子句的情况

Fig. 1 Distribution of learnt clauses being deleted in VSIDS and LBD

对 SAT Race 2015 年和 SAT Competition 2016 年的实例进行大量测试,在 1h 内能够求解出的实例中,活跃度方式平均删除了 84.4%的使用过的学习子句,LBD 方式平均删除了 79.8%的使用过的学习子句,两种方式都大量地删除了使用过的学习子句。因此,有必要对学习子句参与冲突分析的行为进行进一步的分析和刻画,以确保使用过的子句中更有用的句子能够被保留。

4　学习子句动态趋势评估方法

4.1　学习子句被用于冲突分析的趋势强度量化

用 $t_i(i\geqslant 0)$ 表示学习子句 C 在生存期内对应的搜索总冲突次数，其中 t_0 为生成 C 时对应的搜索总冲突次数，t_1，t_2，…，t_k 为 C 被用于推导新子句时对应的搜索总冲突次数，t_i 在本质上为时间序列。用 $\overrightarrow{t_{i-1}t_i}(i\geqslant 1)$ 表示子句在 t_{i-1} 时刻出现（或被使用），并在 t_i 时刻再次被使用的区间；$\Delta t_i=t_i-t_{i-1}(i\geqslant 1)$ 为相邻两次被用于冲突分析对应总冲突次数的间隔。子句参与冲突分析的序列如图 2 所示。学习子句参与冲突分析的次数越多，对应的 k 值越大。但对于 k 值均较大且接近的两个子句，其对后续搜索的影响不能简单用 k 来衡量。整体上看，学习子句在 t_i 时刻参与冲突是一个随机的过程。图 3 所示为实例 50bits_13.dimacs.cnf 中某个学习子句参与冲突分析的时间分布，从第 390785 次冲突产生该子句，直至被删除，该学习子句总共被使用了 22204 次，在其整个生存期出现的规律中很难找到分布特征。且一般情况下，求解器冲突次数在百万级甚至千万级，所有子句参与冲突分析的时刻 t_i 的个数可达数百亿级，若将其完全驻留在内存并分析其分布特征将严重降低求解效率。因此，从概率的角度预测子句在删除周期点 t_{reduce} 后是否出现或何时出现是不可行的。

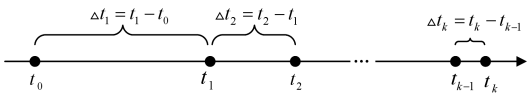


图 2　学习子句参与冲突分析的时间序列示意图

Fig. 2　Schematic point-in-time diagram of learned clause used in conflict analysis

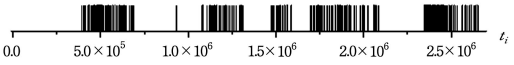


图 3　50bits_13.dimacs.cnf 求解过程中某学习子句参与冲突分析的时间序列

Fig. 3　Point-in-time of one learnt clause used in conflict analysis of 50bits_13.dimacs.cnf

由前述分析可知，如能找到轻量级的分布特征刻画方法，即类似于 $\omega'=\omega+\Delta\omega$ 的方式来刻画分布特征，将有利于计算机实现。假定存在 3 个学习子句 C_1，C_2 和 C_3，均被用于冲突分析 3 次，并在 t_{reduce} 处决定是否被删除，则 $t_i(1\leqslant i\leqslant k)$ 的分布如图 4 所示。在 C_1 中，其冲突分析的时间序列间隔为 $\Delta t_1>\Delta t_2>\Delta t_3$；在 C_2 中，有 $\Delta t_1<\Delta t_2<\Delta t_3$；在 C_3 中，有 $\Delta t_1<\Delta t_3<\Delta t_2$。

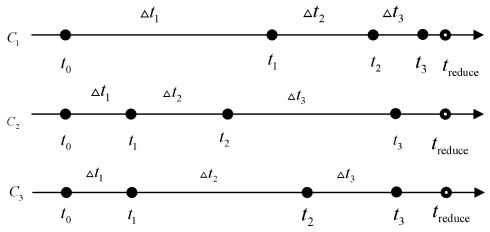


图 4　学习子句参与冲突分析的不同分布

Fig. 4　Different distributions of learnt clauses used in conflict analysis

若某个子句在删除节点前仍然被频繁使用，则在后续搜索中很有可能依然会用到，若此时将其删除，将不利于后续搜索。由此可以看出，C_1 所在冲突分析的时间序列间隔 Δt 逐渐缩小，相比 C_2 和 C_3，C_1 更“好”。为便于直观理解，用时间序列间隔 Δt 的倒数形式 $\Delta f=\dfrac{1}{\Delta t_i}$ 表示子句在 Δt_i 时间段出现的频率，对于子句从 $\overrightarrow{t_{i-2}t_{i-1}}$ 到 $\overrightarrow{t_{i-1}t_i}$ 的局部变化趋势强度 T_i，可进行如下量化定义。

定义 1　对于学习子句 C，存在连续相邻两次被用于冲突分析的时间间隔 $\dfrac{1}{\Delta t_{i-1}}$ 和 $\dfrac{1}{\Delta t_i}$，如果 $\Delta f_i=\dfrac{1}{\Delta t_i}-\dfrac{1}{\Delta t_{i-1}}>0(i\geqslant 2)$，则称 C 的趋势为强趋势，记为 $\mathcal{T}_i=1$。

定义 2　对于学习子句 C，存在连续相邻两次被用于冲突分析的时间间隔 $\dfrac{1}{\Delta t_{i-1}}$ 和 $\dfrac{1}{\Delta t_i}$，如果 $\Delta f_i=\dfrac{1}{\Delta t_i}-\dfrac{1}{\Delta t_{i-1}}<0(i\geqslant 2)$，则称 C 的趋势为弱趋势，记为 $\mathcal{T}_i=-1$。

定义 3　对于学习子句 C，存在连续相邻两次被用于冲突分析的时间间隔 $\dfrac{1}{\Delta t_{i-1}}$ 和 $\dfrac{1}{\Delta t_i}$，如果 $\Delta f_i=\dfrac{1}{\Delta t_i}-\dfrac{1}{\Delta t_{i-1}}=0(i\geqslant 2)$，则称 C 的趋势为无趋势，记为 $\mathcal{T}_i=0$。

局部趋势变化可直观地用图形说明，图 5 为图 4 中学习子句 C_1，C_2，C_3 参与冲突分析的连续趋势变化。由局部变化趋势的定义可知，未被使用的子句（$k=0$）和仅被使用一次的子句（$k=1$）不存在变化趋势，因此对此部分子句可采取设置阈值的方法直接删去。对于 $k\geqslant 2$ 的子句 C，其被用于冲突分析的总体趋势 $\mathcal{G}(C)$ 可由局部趋势累积得到，故有：

$$\mathcal{G}(C)=\begin{cases}0, & i\leqslant 1\\ \sum\limits_{i=2}^{k}\mathcal{T}_i, & 2\leqslant i\leqslant k\end{cases}\tag{1}$$

总体趋势强度 $\mathcal{G}(C)$ 也可表示为 $\mathcal{G}(C)=\mathcal{G}_{i-1}(C)+\mathcal{T}_i(2\leqslant i\leqslant k)$，其重要意义是 $\mathcal{G}(C)$ 可通过增量方式计算得到，而不需要保留大量的历史数据，因此有助于轻量级的计算机程序实现。

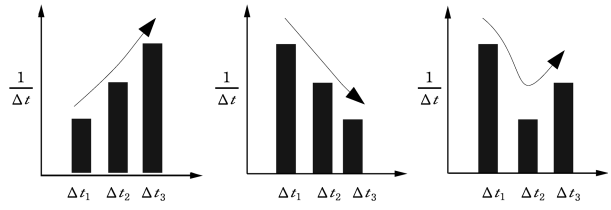


图 5　学习子句参与冲突分析的局部趋势变化

Fig. 5　Local varying tendencies of learnt clauses used in conflict analysis

回顾图 4 中的子句 C_1，C_2 和 C_3：

对于 C_1，由于 $\Delta t_1>\Delta t_2>\Delta t_3$，有 $\Delta f_2=\dfrac{1}{\Delta t_2}-\dfrac{1}{\Delta t_1}>0$，$\Delta f_3=\dfrac{1}{\Delta t_3}-\dfrac{1}{\Delta t_2}>0$，对应的趋势强度 $\mathcal{T}_2=1$，$\mathcal{T}_3=1$，由式(1)可知，C_1 被用于冲突分析的趋势强度为 $\mathcal{G}(C_1)=\sum\limits_{i=2}^{3}\mathcal{T}_i=2$。

类似地，对于 C_2，由于 $\Delta t_1<\Delta t_2<\Delta t_3$，有 $\Delta f_2=\frac{1}{\Delta t_2}-\frac{1}{\Delta t_1}<0$，$\Delta f_3=\frac{1}{\Delta t_3}-\frac{1}{\Delta t_2}<0$，对应的趋势强度 $\mathcal{T}_2=-1$，$\mathcal{T}_3=-1$，C_2 被用于冲突分析的趋势强度为 $\mathcal{G}(C_2)=\sum_{i=2}^{3}\mathcal{T}_i=-2$。

对于 C_3，由于 $\Delta t_1<\Delta t_3<\Delta t_2$，有 $\Delta f_2=\frac{1}{\Delta t_2}-\frac{1}{\Delta t_1}<0$，$\Delta f_3=\frac{1}{\Delta t_3}-\frac{1}{\Delta t_2}>0$，对应的趋势强度 $\mathcal{T}_2=-1$，$\mathcal{T}_3=1$，C_3 被用于冲突分析的趋势强度为 $\mathcal{G}(C_3)=\sum_{i=2}^{3}\mathcal{T}_i=0$。

量化结果表明，$\mathcal{G}(C_1)>\mathcal{G}(C_3)>\mathcal{G}(C_2)$，说明 C_1 的趋势强度最大，这与直观分析 C_1 被用于冲突分析的频率在增加是一致的。

图 6 为图 5 示例中学习子句参与冲突分析的时间序列的总体趋势强度 $\mathcal{G}(C)$的变化。通过刻画趋势强度，可将离散的、随机的子句用于冲突分析的特征转换为连续的趋势变化。如果某个子句 $\mathcal{G}(C)$持续增大，则可保留，否则可在删除周期将其从子句库中删除。

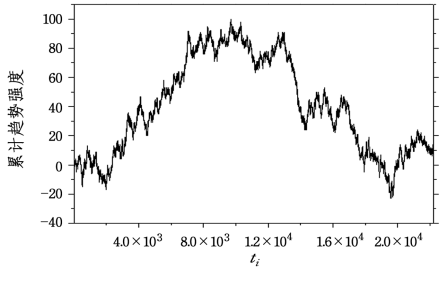


图 6 图 5 中学习子句参与冲突分析的时间序列累计趋势强度分布

Fig. 6 Distribution of cumulative trends of point-in-time of learnt clause in Fig. 5

4.2 实现算法

累积趋势强度的计算不需要记录某个子句的所有历史冲突时间点数据，仅需要在子句类中增加 *sum_TS*，*T_lastconf*，*delta_T* 成员变量，分别记录子句的累积趋势强度 $\mathcal{G}(C)$、最近一次被用于冲突分析的系统总冲突次数和最近一个冲突次数的间隔大小。当某学习子句被推导或用于冲突分析时，更新相应的成员值，其伪代码如算法 1 所示。

算法 1 更新学习子句累积趋势强度 updateTS

```
输入:学习子句 C,系统总冲突次数 conflicts
输出:更新趋势强度后的学习子句
1. if C.learnt then /* 仅对学习子句进行操作 */
2.    Δt ← conflicts−C.T_lastconf
3.    if C.delta_T > 0 then /* 存在前驱 delta,即至少被使用过一次,
      更新子句趋势强度 */
4.    if Δt >C.delta_T then ts← −1
5.       else if Δt <C.delta_T then ts← 1
6.       else ts← 0
7.    end
8.       C.sum_TS ←C.sum_TS+ts/ * 更新累积趋势强度 */
9.    end
10.   C.delta_T←Δt / * 更新为当前冲突区间 */
11.   C.T_lastconf←conflicts/ * 更新为系统总冲突次数 */
12. end
```

在删除周期 t_{reduce} 到达时，调用删除子句的过程。*C.size*() 返回学习子句 *C* 中的文字个数；*locked*(*C*)返回学习子句 *C* 是否已用于推导其他必须赋值的文字。算法 2 遍历学习子句库 *Learnt_Clauses*，将同时满足以下 3 个条件的子句删除：

1)趋势强度小于指定阈值 *TS_Lim*；

2)子句中的文字数大于 2；

3)未被用于推导其他必须赋值的文字。

对于条件 2)，由于单元子句和二元子句有利于化简问题和推导冲突子句，因此保留文字数不超过 2 的子句。对于条件 3)，主要是为了保证求解器的完备性，即确保生成完整的不可满足公式的证明过程，对相关子句予以保留。

算法 2 周期性删除学习子句 reduceDB

```
输入:学习子句库 Learnt_Clauses,趋势强度阈值 TS_Lim
输出:更新后的学习子句库
1. for C in Learnt_Clauses do
2. if C.sum_TS<TS_Lim&&C.size() > 2 && ! locked(C) then
3.       removeClause(C)
4. end
5. return Learnt_Clauses
```

5 实验与结果分析

5.1 实验方案

Minisat 是国际上知名的 CDCL SAT 求解器，集成了子句活跃度评估算法，历年 SAT 国际竞赛中很多冠军求解器都是 Minisat 的改进版，但由于 Minisat 中未集成更多的优化技术，在对比测试中容易看出改进算法的性能差异。本文以 Minisat 为基础工具，分别实现了 LBD 评估方式和趋势强度评估方式。本文称 Minisat 的原版为 Minisat+ACT，LBD 版本为 Minisat+LBD，趋势强度版本为 Minisat+TS(k)。其中 k 为趋势强度阈值，小于 k 的子句将被强制删除。

实验测试实例来自于 2015 年 SAT-Race 的 main track 组和 2016 年 SAT Competition 的 main track 组，涉及密码验证、多机器人路径规划等领域，其中包含 300 个 2015 年的竞赛实例、500 个 2016 年的竞赛实例(200 个 Crafted、300 个 Application)。每个测试例无预处理且限时 1000 s。硬件环境为 Intel i7-6700 3.4Ghz CPU，16 GB 物理内存，Red Hat Enterprise 7.3 操作系统。删除周期阈值 t_{reduce} 采用 Luby 序列[15]，有 $t_{\text{reduce}}=\{100,100,200,100,100,200,400,100,100,200,100,100,200,400,800,\cdots\}$。

5.2 实验结果

实验结果如表 1 所列，SAT 和 UNSAT 列分别表示求解实例的属性为可满足和不可满足。总体而言，趋势强度评估方式的性能介于活跃度方式和 LBD 评估方式之间，显著优于活跃度评估方式，稍弱于 LBD 方式。当趋势强度阈值设置为 $3\leqslant k\leqslant 6$ 时，Minisat+TS 的求解数量呈增长趋势，说明保留趋势强度较大的学习子句有利于后续搜索。当 $k=6$ 时，Mini-

sat+TS 求解 SAT Race 2015 的实例总数比 Minisat+ACT 多 13 个，增加了 8.3%；求解 SAT Competition 2016 的实例总数比 Minisat+ACT 多 19 个，增加了 19.6%，性能提升明显。在求解 SAT Race 2015 和 SAT Competition 2016 总实例数时，表现最好的趋势强度版本解出 286 个，比 Minisat+ACT 多 32 个，增加了 12.6%；比 Minisat+LBD 少了 7 个，减少了 2.4%。

表 1　不同趋势强度阈值与活跃度、LBD 评估方式的对比测试结果

Table 1　Testing results of different threshold sets to VSIDS and LBD

求解器	测试集(SAT 2015)			测试集(SAT 2016)			合计
	SAT	UNSAT	小计	SAT	UNSAT	小计	
Minisat+ACT	105	52	157	50	47	97	254
Minisat+LBD	113	58	171	54	68	122	293
Minisat+TS(3)	108	57	165	55	59	114	279
Minisat+TS(4)	112	57	169	56	59	115	284
Minisat+TS(5)	112	57	169	56	59	115	284
Minisat+TS(6)	113	57	170	56	60	116	286
Minisat+TS(7)	113	57	170	56	60	116	286
Minisat+TS(30)	113	57	170	56	61	111	287
Minisat+TS(60)	113	57	170	56	60	116	286

当 $k \geqslant 6$ 时，对应趋势强度版求解器的求解个数趋于稳定，说明：1)绝大多数学习子句趋势强度集中于 $k<6$ 的区间，不同版本均删除了相同的子句，使得结果差异不大；2)学习子句参与冲突分析的时间序列分布并非完全随机，存在阶段性随机、阶段性非随机的可能。后续可针对上述两个方面进行进一步的研究。

结束语　针对现有学习子句评估算法评估效果不理想的问题，本文分析了已有评估方式的局限性，研究了学习子句被使用的时间序列分布，提出了一种学习子句累积趋势强度量化算法，将随机、离散的时间分布转换为连续的累积趋势强度，通过设定趋势强度阈值来进一步确定子句是否应该被删除。对比实验表明，趋势强度评估算法反映了子句活跃度特征，且性能明显优于活跃度评估算法，与 LBD 算法基本持平，且算法占用内存较小，为学习子句的评估提供了一种新的思路。

参 考 文 献

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

[2] BIERE A, HEULE M, VAN MAAREN H, et al. Conflict-driven clause learning SAT solvers [M]// Handbook of Satisfiability, Frontiers in Artificial Intelligence and Applications. Amsterdam: IOS Press, 2009: 131-153.

[3] VAN DER TAK P, RAMOS A, HEULE M. Reusing the assignment trail in cdcl solvers [J]. Journal on Satisfiability, Boolean Modeling and Computation, 2011, 7(4): 133-138.

[4] SILVA J P M, SAKALLAH K A. GRASP—a new search algorithm for satisfiability [C]// Proceedings of the 1996 IEEE/ACM International Conference on Computeraided Design. Los Alamitos: IEEE Computer Society Press, 1996: 220-227.

[5] BAYARDO J R R J, SCHRAG R. Using CSP look-back techniques to solve real-world SAT instances [C]// Proceedings of the 14th National Conference on Artificial Intelligence. Menlo Park: AAAI Press, 1997: 203-208.

[6] MOSKEWICZ M W, MADIGAN C F, ZHAO Y. Chaff: Engineering an efficient SAT solver[C]// Proceedings of the 38th Annual Design Automation Conference. New York: ACM, 2001: 530-535.

[7] GOLDBERG E, NOVIKOV Y. BerkMin: A Fast and Robust Sat-Solver [C]// Proceedings of the Conference on Design, Automation and Test in Europe (DATE 2002). Los Alamitos: IEEE Computer Society Press, 2002: 142-149.

[8] SÖRENSSONN, EÉN N. MiniSat v1.13-A SAT solver with conflict-clause minimization [C]// Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing. Berlin: Springer, 2005: 502-518.

[9] SIMON L, AUDEMARD G. Predicting learnt clauses quality 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.

[10] AUDEMARD G, LAGNIEZ J-M, MAZURE B, et al. On freezing and reactivating learnt clauses [C]// Proceedings of the International Conference on Theory and Applications of Satisfiability Testing. Berlin: Springer, 2011: 188-200.

[11] GUO L, JABBOUR S, LONLAC J, et al. Diversification by clauses deletion strategies in portfolio parallel SAT solving [C]// Proceedings of 2014 IEEE 26th International Conference on Tools with Artificial Intelligence (ICTAI 2014). Los Alamitos: IEEE Computer Society Press, 2014: 701-708.

[12] JABBOUR S, LONLAC J, SAIS L, et al. Revisiting the learned clauses database reduction strategies [OL]. [2016-10-20]. https://arxiv.org/pdf/1402.1956.

[13] ANSÓTEGUI C, GIRÁLDEZ-CRU J, LEVY J, et al. Using community structure to detect relevant learnt clauses [C]// Proceedings of the International Conference on Theory and Applications of Satisfiability Testing. Switzerland: Springer, 2015: 238-254.

[14] LONLAC J, NGUIFO E M. Towards Learned Clauses Database Reduction Strategies Based on Dominance Relationship[OL]. [2016-10-20]. https://arxiv.org/pdf/1705.10898.

[15] LUBY M, SINCLAIR A, ZUCKERMAN D. Optimal speedup of Las Vegas algorithms [J]. Information Processing Letters, 1993, 47(4): 173-180.