计算机科学
COMPUTER SCIENCE

优化概率选择求解SAT问题

贾书恒, 付慧敏

引用本文

贾书恒, 付慧敏. 优化概率选择求解SAT问题[J]. 计算机科学, 2026, 53(3): 366-374.

JIA Shuheng, FU Huimin. Optimizing Probabilistic Choice for Solving SAT Problems[J]. Computer Science, 2026, 53(3): 366-374.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.241200211

优化概率选择求解 SAT 问题

贾书恒[1]　付慧敏[1,2]
1 西南财经大学计算机与人工智能学院　成都 611130
2 西南财经大学智能金融教育部工程研究中心　成都 611130
(Suqin76067@qq.com)

摘　要　在 SAT 问题的随机局部搜索算法中,主流变量决策策略基于概率选择变量,如 probSAT 求解器通过计算变量的 break 值确定选择概率。然而,该方法易陷入局部最优,尤其在应用类问题中表现不佳。为此,提出了一种结合配置检测策略的变量决策方法,动态调整变量选择概率函数。当环境不变时,优先选择 break 值较低的变量,增强全局优化能力。针对长子句的高扫描开销问题,引入重要邻居数组策略,将高活跃度变量纳入数组,降低计算复杂度。同时,设计了重启机制,利用 probSAT 在初期快速降低不可满足子句数量的优势,避免后期全局重复翻转现象,提升求解效率。改进后的 probSAT_PCCR 求解器在长期未解决的数学应用问题测试中表现显著提升,比原始 probSAT 多解决了 142 个案例,性能提升 546.1%。在美国联邦通信委员会(FCC)的实际应用问题测试中,多解决了 1596 个案例,性能提升 33.5%。结果表明,通过多种策略改进的 probSAT 求解器在解决 SAT 问题的应用类问题上性能大幅提升,具有重要应用价值。

关键词:配置检测;重要邻居数组;可满足性问题;SAT 求解器;变量决策策略
中图分类号　TP181

Optimizing Probabilistic Choice for Solving SAT Problems

JIA Shuheng[1] and FU Huimin[1,2]
1 School of Computing and Artificial Intelligence,Southwestern University of Finance and Economics,Chengdu 611130,China
2 Engineering Research Center of Intelligent Finance,Ministry of Education,Southwestern University of Finance and Economics,Chengdu 611130,China

Abstract　In stochastic local search algorithms for SAT problems,mainstream variable decision strategies rely on probabilistic variable selection,such as the probSAT solver,which determines selection probabilities by calculating the break value of variables. However,this approach is prone to falling into local optima,particularly underperforming in application-oriented problems. To address this,a variable decision method incorporating a configuration detection strategy is proposed,dynamically adjusting the variable selection probability function. When the environment remains unchanged,variables with lower break values are prioritized,enhancing global optimization capabilities. To tackle the high scanning overhead of long clauses,an important neighbor array strategy is introduced,incorporating highly active variables into the array to reduce computational complexity. Additionally,a restart mechanism is designed to leverage the advantage of probSAT in rapidly reducing the number of unsatisfied clauses during the initial phase,avoiding the global repeated flipping phenomenon in later stages,thereby improving solving efficiency. The improved probSAT_PCCR solver demonstrates significant performance enhancements in long-unsolved mathematical application problem tests,solving 142 more cases than the original probSAT,with a performance improvement of 546.1%. In practical application problem tests conducted by the FCC,it solves 1596 more cases,with a performance improvement of 33.5%. In summary,the enhanced probSAT solver,through multiple strategic improvements,achieves substantial performance gains in solving application-oriented SAT problems,demonstrating significant application value.

Keywords　Configuration detection,Important neighbor array,Satisfiability problem,SAT solver,Variable decision strategy

到稿日期:2024-12-30　返修日期:2025-04-01
基金项目:国家自然科学基金(62206227,61976130)
This work was supported by the National Natural Science Foundation of China(62206227,61976130).
通信作者:付慧敏(fuhuimin@swufe.edu.cn)

1 引言

命题可满足性（Propositional Satisfiability，SAT）问题是NP完全问题中备受关注的一类，其因在计算机科学与人工智能领域兼具理论意义与实际价值而引起广泛关注[1]。SAT问题的核心是：在一个定义于布尔变量集上的合取范式（Conjunctive Normal Form，CNF）公式 FFF 中，是否存在一种变量的真值分配，使得公式 FFF 的所有子句均被满足。这一问题不仅是数理逻辑领域的经典研究课题[2]，还在组合优化、统计物理、电路验证、机器学习、约束满足、实时调度以及计算理论等多个实际应用场景中发挥着基础性作用[3-4]。

按照是否能证明实例无解[5]，解决 SAT 问题的算法可以分为完备算法[5]和不完备算法[6]。不完备求解算法不能证明实例无解，但因其能够在较短时间内找到一个解，故成为解决实际问题的重要工具[7]。其中，随机局部搜索（Stochastic Local Search，SLS）作为不完备算法的主流方法[8]，以其高效的变量翻转机制，能够快速在搜索空间中游走以定位问题的解。为提高求解效率，大多数 SLS 求解器通过优化变量选择的启发式方法来提升算法性能。而当前的启发式方法主要分为两类，即双模式 SLS 算法[8]和聚焦随机游走（Focused Random Walk，FRW）算法[9]。

FRW 算法的核心思想是通过选择未满足子句中的变量进行翻转。早在 1994 年，Selman 等[8]提出了经典的 walkSAT 算法，通过引入噪声参数调节选择最小 break 值变量的概率，以平衡贪婪性和随机性。在 walkSAT 的基础上，研究者提出了多种改进方案。例如，Luo 等[10]引入了新的变量选择机制，提出了 FrwCB 算法，Cai 等[11]则加入了线性 make 策略，推出了 walkSATlm 算法，均在不同场景下取得了更好的性能表现。

2010 年提出的 ProbSAT 算法[12]引入概率分布函数选择翻转变量，以其简洁高效的设计，成为许多后续算法的实现基础。例如，2016 年 Biere[13]基于 ProbSAT 进行改进提出了 YalSAT 求解器，2022 年 Fu 等[14]利用 ProbSAT 框架推出的 SelectNTS 求解器，在随机实例的求解上均表现出色。然而，ProbSAT 算法仅依赖于 break 值的概率分布，这种局限性导致算法在某些情况下容易产生无效的重复翻转，并可能陷入循环[14]，从而降低求解效率。这些不足为进一步研究改进 SLS 算法提供了重要方向。

双模式 SLS 算法是一种通过状态切换搜索模式来提升求解性能的 SAT 求解策略。该算法能够在不同状态下适应复杂的搜索环境，从而提高算法的效率与鲁棒性。然而，循环问题是双模式算法的一大挑战，为解决这一问题，2011 年 Cai 等[15]提出了格局检测（Configuration Checking，CC）策略，为双模式算法注入了新的活力。格局检测通过分析子句和变量的结构关系，在特定时间内限制某些变量的翻转行为，从而有效跳出循环。该策略以其实现简单、易于理解和运行高效的特点，成为改进双模式算法的一项关键技术。

格局检测策略最早被用于优化经典的 WalkSAT 算法。2012 年，Cai 等[16]基于 WalkSAT 框架，引入了贪心策略与格局检测策略，并对传统的变量评分机制（score 属性）进行替换，提出了 CCASat 算法，显著提升了算法在多个问题上的求解效率。2014 年，Luo 等[17]进一步将基于变量的格局检测策略与基于子句的格局检测策略相结合，开发出 DCCASat 求解器，其在随机 k-SAT 问题上取得了良好的效果。这种结合变量和子句特性的改进方式，不仅有效解决了算法循环问题，还为随机实例的求解开辟了新的思路。

针对 FRW 求解器存在的局限性，依托双模式算法中的关键策略，是提升求解器效率的有效路径。受到格局检测策略的启发，本文提出了一种基于环境信息的动态概率分布选择方法，以帮助 probSAT 算法更高效地跳出循环，改进其求解性能。此外，针对长子句可能导致邻居数组失效的问题，本文创新性地引入了重要邻居数组的概念，优化了邻居选择机制。通过这一改进，probSAT 算法在实际编码实例中的表现得到了显著提升。

2 SAT 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 或者 0。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]。$C=\bigvee_{j=1}^{m} l_j$是由 m 个文字析取而成的，若子句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。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[20]。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）$break(x,\alpha)$[21]。在当前赋值 α 下，翻转变量 x 将会使得可满足子句的数量变成不可满足子句的数量。

5）邻居[15]。如果这两个变量出现在同一个子句中，那么这两个变量是邻居。变量 x 的邻居变量的集合表示为 $N(x)=\{y\mid y\in V(F)$且 y 出现的子句中，至少有一个含有 $x\}$。

2.2 probSAT 算法

基于概率选择变量的 probSAT 算法[12]的具体实现如算法 1 所示。该算法的核心思想是通过概率分布函数选择变量进行翻转，以优化搜索效率。算法首先随机选取一个赋值为假的子句，然后基于当前赋值状态，计算子句中每个变量的 break 值，并将其代入对应的概率分布函数，得出各变量的概率值。

为了适应不同长度的子句，probSAT 算法针对子句长度采取了差异化策略：对于长度较短的子句，采用多项式分布函数（见式(1)）进行概率计算；对于长度大于 6 的子句，采用指数式分布函数（见式(2)）计算变量概率，以减轻指数衰减的影响。接下来，依据式(3)计算出概率值，即从随机选取的赋值为假的子句 C 中选择一个变量 x 进行翻转，如下所示：

$$f(\nu,\alpha)=(0.9+break(\nu,\alpha))^{-cb_1} \tag{1}$$

$$f(\nu,\alpha)=(cb_2)^{-break(\nu,\alpha)} \tag{2}$$

$$\frac{f(x,a)}{\sum_{z\in C} f(z,a)} \tag{3}$$

算法 1 probSAT 算法

输入:CNF 公式 F,最大尝试次数,最大步骤次数
输出:公式 F 可满足的赋值指派 α,或者 Unkown
1. for try:=1 to maxtries do
2. α:=a randomly generated truth assignment;
3. for step:= 1 to MaxSteps do
4. if α satisfies F then return α;
5. C:=a random unsatisfied clause;
6. for ν in C do;
7. compute f(ν,α);
8. end for
9. $\nu:=x\in C selected with probability \frac{f(x,\alpha)}{\sum_{z\in C} f(z,\alpha)}$;
10. α:=α with ν flipped;
12. end for
13. return Unknow;
14. end

2.3 格局检测策略

格局检测策略由 Cai 等[16]提出,旨在解决 SAT 问题求解过程中常见的局部循环现象[14]。所谓循环现象,是指在变量翻转过程中,当某个变量被选择翻转后,其邻居变量未被及时选择翻转,导致算法在后续迭代中再次选择该变量进行翻转,从而使得算法返回到原先的状态。这种现象不仅影响了算法的全局搜索能力,还显著降低了求解效率。

在具体实现过程中,格局检测策略通过维护一个配置数组 $conf[\ ]$来跟踪变量的环境变化状态。初始时,所有变量的配置值均被设置为 1,即 $conf[x]=1$,表示变量尚未被翻转或其环境信息未发生变化。当某个变量 x 被选择并执行翻转操作时,算法会将 $conf[x]$的值更新为 0,表示当前变量已被翻转;同时,对于变量 x 的邻居集合 $N(x)$中的所有变量 y,其配置值 $conf[y]$被设置为 1,表明这些变量的环境信息可能已发生变化。

在每次选择变量进行翻转之前,算法会检查目标变量 x 的配置值。

1)若 $conf[x]=0$,说明变量 x 的环境信息未发生变化,即自变量 x 上次被翻转以来,其邻居集合 $N(x)$中的任何变量均未被选择翻转。在这种情况下,翻转变量 x 可能导致算法陷入局部循环。

2)若 $conf[x]=1$,表明变量 x 的环境信息发生了变化,意味着其邻居集合 $N(x)$中至少有一个变量已被翻转,从而打破了可能的循环依赖,允许算法对变量 x 进行翻转。

3 基于环境信息的概率选择策略和重要邻居数组策略

本章介绍了一系列旨在提升 probSAT 算法在实际应用实例中的效率的优化策略。首先,提出了 PC(Probability and Configuration)策略,该策略通过动态结合变量的环境信息与概率选择机制来优化算法。随后,针对 PC 策略的弊端,进一步提出了 PCC(Probabilistic Configuration Checking)策略,该策略优先选择 break 值为 0 的高价值变量。最后,为了降低长子句实例中的计算开销,引入了重要邻居数组策略,通过聚焦关键变量的更新,减少了不必要的配置数组更新操作。此外,在第 4 章的实验部分,为了重新激发算法的探索能力,设计了重启机制。

3.1 基于环境信息的概率选择策略

由于 probSAT 算法仅通过概率选择变量进行翻转,这种机制可能导致算法在连续两次操作中选择相同的变量进行翻转,从而造成资源的浪费。据大量实验数据统计,在解决布尔-勾股三元组问题(Boolean Pythagorean Triples, PTN)[22]的测试中,无效的重复翻转占总翻转次数的比例高达 4% ~ 20%(见图 1)。

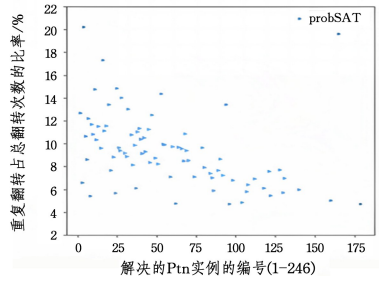

图 1 probSAT 算法重复翻转比率

Fig. 1 probSAT's repeated flip ratio

然而,简单地禁止重复翻转的策略并非最佳解决方案,往往会引发算法性能的下降。为克服这一问题,本文借鉴格局检测策略在突破局部最优解方面的有效性,并将其与 probSAT 算法的概率选择机制相结合,提出了一种全新的 PC(Probability and Configuration)策略。PC 策略通过动态结合变量的环境信息与概率选择,既避免了重复翻转造成的资源浪费,又保持了算法的灵活性和全局搜索能力。

PC 策略的核心思想是利用环境信息判断变量的可选性:当变量的环境信息保持不变时,其被选择的概率直接设定为 0,从而避免无效的重复翻转;而当环境信息发生变化时,则依据预设的概率分布函数计算该变量的选择概率。这一机制通过引入动态调节,不仅能有效降低冗余操作,还能增强算法的全局搜索效率和适应性。其伪代码如算法 2 所示。

算法 2 PC 策略

输入:变量 x,赋值指派 α,配置数组 conf[x]
输出:变量 x 被选择的概率值
1. if conf[x]== 0
2. Probability[x]=0;
3. else if conf[x] == 1
4. Probability[x]=compute f(x,α);
5. return Probability[x]

算法 2 详细阐述了 PC 策略的实现逻辑:首先,维护一个记录变量环境状态的配置数组,用于判断环境信息是否发生变化;其次,在变量选择过程中,通过检测其配置状态来动态调整选择概率。当环境信息发生变化时,利用概率分布函数计算选择概率;否则,直接排除该变量的选择可能性。

然而,Cai 等指出[15],局部搜索算法中的格局检测策略由于过于严格,可能会暂时限制某些变量的翻转。这种限制在

某些情况下会导致算法错失具有高价值(即 break 值为 0)的变量，影响整体求解效率。为解决这一问题，PCC(Probabilistic Configuration Checking)策略应运而生。

PCC 策略在设计上对传统模式检测进行了改进，其核心思想是：在环境信息未发生变化的情况下，优先考虑 break 值为 0 的变量。通过这一调整，PCC 策略不仅有效避免了无效循环翻转，同时提高了高价值变量被选中的概率。相比传统的模式检测，PCC 策略在保持循环规避能力的基础上，进一步提升了算法在局部搜索过程中的灵活性和效率。

PCC 策略的设计与变量选取过程中概率分布函数的计算密切相关。表 1 列出了在不同 break 值下，分布函数 f 计算所得的概率值分布情况。由于当 break 值大于或等于 1 时，f 的计算结果始终小于 1，因此对其值进行平方处理后，能够显著提高 break 值较低的高价值变量被选中的概率。这种概率调整机制在避免重复翻转的同时，强化了算法对潜在高价值解的探索能力。

表 1　原概率值与平方后的概率值对比

Table1　Comparison of original probability values and squared probability values

	break			
	0	1	2	3
prob	1.24240	0.26654	0.11155	0.06059
prob^2	1.54355	0.07105	0.01244	0.00367

算法 3 详细描述了 PCC 策略的实现细节。在该策略中，当环境信息保持不变时，变量的选中概率被设定为 $\frac{f(x,\alpha)}{\sum_{z\in C} f(z,\alpha)}$；而当环境信息发生变化时，变量的选中概率被调整为 $\frac{f(x,\alpha)^2}{\sum_{z\in C} f(z,\alpha)^2}$。这种动态调整机制通过强化对高价值变量的优先选择能力，提升算法的求解效率。

算法 3　PCC 策略

输入：变量 x，赋值指派 α，配置数组 conf[x]

输出：变量 x 的被选择的概率值

```
1. if conf[x] == 0
2.    Probability[x]=compute f(x,α)^2;
3. else if conf[x] == 1
4.    Probability[x]=compute f(x,α);
5. return Probability[x]
```

通过应用 PCC 策略，在 PTN 实例的测试中，无效翻转次数显著减少至 1%～5%，如图 2 所示。

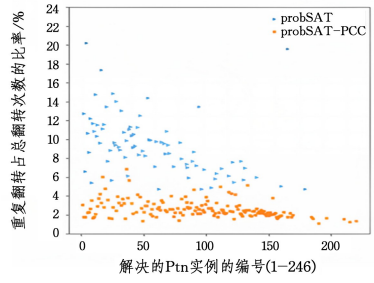

图 2　probSAT 算法和 probSAT_PCC 算法重复翻转的概率对比(电子版为彩图)

Fig. 2　Repeated flip probability: probSAT vs. probSAT_PCC

3.2　重要邻居数组策略

当选择某个变量进行翻转后，算法需要将变量 x 的配置值 $conf[x]$更新为 0，同时同步更新其邻居数组中所有变量 y 的配置值 $conf[y]$。然而，在包含大量长子句的情况下，这种全面更新操作的时间成本可能过高，从而降低了算法的整体求解效率。

直观上看，循环现象的发生主要与翻转频率较高的变量相关，这一点从 3.1 节和图 1 也可以看出。因此，仅对翻转频率较高的变量以及部分翻转频率较低的变量进行更新，即可有效防止循环现象的出现。这是因为 $conf[\ ]$数组的动态变化更多地依赖于频繁翻转的变量，而非所有变量的状态更新。

基于这一观察，减少对翻转频率较低变量的更新操作，不仅能够显著降低计算开销，还能在不影响算法性能的情况下维持其原有的效果。此优化策略通过聚焦关键变量更新，平衡算法效率与资源消耗，为处理长子句占比较高的复杂实例提供了更高效的求解方案。

重要邻居数组策略通过初始时随机将部分邻居变量加入重要邻居数组，并为每个变量维护一个指针来实现优化。当某变量发生翻转时，记录其被选择的次数；当次数超过预设的翻转阈值时，将该变量加入其所有邻居变量的“重要邻居”数组(记为 $ImNeighbor$)中，同时指针向后移动。指针的数组坐标在 $ImNeighbor$ 数组长度范围内循环移动，从而动态调整数组内容。

该策略引入了两个核心参数：重要邻居数组的长度和加入重要邻居数组所需的翻转次数阈值。变量 x 的重要邻居数组 $ImNeighbor[x]$ 长度越长，更新 $ImNeighbor[x]$的开销越大，但由于数组中包含了更多的变量，当其中任意一个变量被选中时，都需要更新 $conf[x]$，从而提高了变量 x 被选中的概率；相反，较短的数组长度减少了更新开销，同时降低了变量的选中概率。

直观来看，在长子句实例中，选择较短的 $ImNeighbor$ 数组长度有助于提高变量被访问的频率，从而加速算法的迭代；而在短子句实例中，使用较长的 $ImNeighbor$ 数组长度可以增强算法的贪婪性，提升局部搜索效率。

若不采用重要邻居数组策略，仅通过基于环境信息的概率选择机制优化 probSAT 算法，在选择变量 x 进行翻转后，需执行以下步骤(不妨设一共有 N 个子句)。

1)计算子句中所有变量的概率：假设目标子句长度为 L，这一部分的时间复杂度为 $O(L)$，相较于动则几万的子句数量，属于常量级别[23]。

2)更新变量 x 所涉及子句的 break 值：假设变量 x 出现在 aN 个子句中，这部分的时间复杂度为 $O(aN)$。原版 probSAT 算法在这一步执行完毕，引入 PCC 策略后还需执行下一步。

3)更新配置数组的值：此步骤涉及变量 x 的所有邻居，假设邻居数量为 bN，更新配置数组的开销为 $O(bN)$。

因此，忽略常量级的第一步后，probSAT 原版算法的时间复杂度为 $O(aN)$，添加 PCC 策略后，总时间复杂度为 $O((a+b)N)$[23]。

在未采用重要邻居数组策略的情况下，随着实例中长子

句数量的增加以及变量邻居数量的增多，更新配置数组的开销显著上升，甚至可能超越算法前两步的计算成本。对于随机局部搜索算法，在解空间中快速移动是其核心要求[15]，而配置数组更新的高昂开销会对整体效率造成严重影响。

为应对这一问题，引入重要邻居数组策略显得尤为必要。通过缩小配置数组更新的范围，仅针对重要邻居变量进行更新，能够将这一步的时间复杂度降低到常量级别。这种优化设计不仅减轻了因变量邻居数量过多或长子句密集而带来的性能瓶颈，还能有效提升算法的整体求解效率。

值得注意的是，重要邻居数组策略是对 PCC 策略的进一步优化，专注于修改 PCC 策略中更新邻居数组的操作方式。为验证重要邻居数组策略对算法效率的影响，本文将在 4.3 节进行对比实验。

3.3 probSAT_PCCR 算法

为充分融合两种策略进行优势，本文在 probSAT 算法中整合 PCC 策略与重要邻居数组策略，提出了一种全新的算法，即 probSAT_PCCR 算法。该算法在 probSAT 的基础上，针对变量选择的逻辑进行了重要改进：将基于 break 值计算变量概率值的原始逻辑替换为结合环境信息与 break 值的概率计算方法，从而增强了算法对变量选择的动态适应能力；此外，probSAT_PCCR 算法对格局检测策略中的邻居数组长度进行了合理限制，以适应不同实例的特性，有效平衡了计算开销与求解效率。

算法 4 详细阐述了 probSAT_PCCR 算法的具体实现。通过引入这两种策略的优化，probSAT_PCCR 不仅继承了 PCC 策略在避免循环翻转方面的优势，还利用重要邻居数组策略降低了邻居更新的开销，进一步提升了算法在处理长子句及复杂实例时的整体性能。

算法 4 probSAT_PCCR 算法

输入：CNF 公式 F，最大尝试次数，最大步骤次数

输出：公式 F 可满足的赋值指派 α，或者 Unkown

```
for try:=1 to maxtries do
  α:=a randomly generated truth assignment;
    for step:= 1 to MaxSteps do
      if α satisfies F then return α;
        C:=arandom unsatisfied clause;
        for v in C do;
          compute f(ν,α)by PCC;
        end for
        v:=x∈C selected with probability f(x,α)/∑_{z∈C} f(z,α);
        updata conf(v) by limit array length
        α:=α with ν flipped;
  end for
return Unknow;
end
```

4 实验结果与分析

4.1 实验综述和实例介绍

本实验从 3 个方面对 probSAT_PCCR 算法的有效性进行了全面验证。

1）关键参数分析：研究重要邻居数组策略中的两个核心参数——重要邻居数量（*numIN*）和最大翻转次数（*mostFN*）在不同实例上的表现。通过实验分析这两个参数的设置对求解器求解效率的影响，为优化参数配置提供数据支持。

2）策略组合效果：探讨单独引入 PCC 策略与 PCC 策略结合重要邻居数组策略时，对求解器效率提升的差异。通过比较两种策略的独立与联合效果，明确重要邻居数组策略在提升算法性能中的实际贡献。

3）对比实验：与其他主流求解器进行对比实验，进一步验证 probSAT_PCCR 算法在求解效率上的改进程度。实验覆盖多种类型的实例，以全面评估新求解器的性能优势。

通过这 3 个方面的实验分析，本文全面展示 probSAT_PCCR 算法的改进效果，并为后续算法优化与应用提供了重要参考依据。

本文实验的实验环境为 Intel core i5-10400F CPU，3.8GHz，16 GB 内存，运行系统为 Windows 11 + Cygwin 2.905。

为了全面验证新算法在短子句和长子句实例上的有效性，实验选用了短子句实例 PTN 和长子句实例 FCC 作为测试对象。实验将求解时间统一设置为 100 秒，对于未解决的实例，其运行时间按求解时间的 10 倍计算。实验进行 1 次，以测试算法在有限时间内的实际表现。以下是对两类测试实例的具体介绍。

FCC（Federal Communications Commission）实例[24]：包含 10000 个实例，变量从 41 到 60 分为 20 组，涵盖了可满足和不可满足的实例。美国联邦通信委员会利用 SAT 求解器分析优化频谱打包收益时产生这样的实例。

PTN 和 PTN-more 实例[24]：共有 23 个公开的可满足实例，称为 PTN 实例，其中 2 个（plain7824.cnf 和 bce7824.cnf）由 Heule 等提供，其余 21 个来自 SAT 竞赛在 2016 年创建的基准测试。此外，通过 PTN 生成器生成了 246 个额外的可满足实例（PTN-More），以进一步研究 probSAT_PCCR 在 PTN 实例上的性能。SAT 技术被广泛应用于解决一个长期悬而未决的数学问题，即布尔毕达哥拉斯三元组（PTN）。

为了提高实验的效率，4.2 节的参数实验和 4.3 节的策略比较实验并不采用全部的 PTN 实例和 FCC 实例。PTN 实例使用 246 个 PTN-more 实例，FCC 实例采用 680 个 46－47 范围的实例。

4.2 参数实验

重要邻居数组的长度直接影响变量禁止翻转的强度以及算法每一步的执行开销。数组长度越长，变量禁止翻转的强度越大，但同时每一步的执行开销也随之增加。相反，较短的数组长度能够降低执行开销，但可能削弱对变量翻转的约束效果。

此外，重要邻居数组的最大翻转次数阈值决定了添加邻居的频率。阈值越低，重要邻居的添加频率越高，从而加强了变量间的交互和算法的贪婪性，但同时也增加了算法的开销；相反，较高的阈值降低了重要邻居的添加频率，从而减少了算法的计算负担。这种调整机制通过在翻转频率与执行效率之

间寻求平衡，为算法在不同实例上的适配性提供了灵活性。

本节将测试不同参数对 probSAT_PCCR 算法效率的影响。实验中，在较大范围内测试翻转阈值参数，分别将翻转阈值设置为 100，1000 和 10000，针对不同实例进行详细测试。

对于 PTN 实例，其特点是子句长度较短、数量较少，且邻居数组长度较短。在这种情况下，可以选择较长的邻居数组长度以进一步优化算法性能。因此，将邻居数组长度设置为 10，20，30，40，50，60，70，80，90，100，200，同时考虑在大范围和小范围内，这些参数对算法效率的影响。

相比之下，FCC 实例的特点是子句长度较长、数量较多，邻居数组长度较长且相似度较高。在这种情况下，过长的邻居数组可能会削弱对翻转的约束力，同时增加算法的开销，从而降低了求解效率。因此，对于 FCC 实例，选择较短的邻居数组长度进行测试，设置为 1，2，3，4，5，6，7，8，9，10，20，以评估不同邻居数组长度对算法性能的影响。通过这些实验详细分析翻转阈值与邻居数组长度在不同类型实例上的作用和表现，从而为 probSAT_PCCR 算法参数的优化提供更科学的指导。实验结果将展示各参数组合对求解效率的具体影响。

表 2 列出了对 PTN 实例进行参数调整测试后的结果。为了更好地观察参数的影响，图 3 展示了重要邻居数组长度参数在不同翻转次数阈值下解决实例的个数的折线图。表 3 和图 4 给出了 FCC 实例测试下的结果。

表 2　probSAT_PCCR 算法在 PTN 实例上的参数测试

Table 2　Parameter testing for probSAT_PCCR algorithm on PTN

numIN	100		1000		10000	
	Solved	PARTime	Solved	PARTime	Solved	PARTime
10	113	550.1413	112	551.1394	95	530.1419
20	148	405.5341	153	384.5086	148	403.4142
30	155	376.1045	154	380.5728	165	**305.0717**
40	**157**	**370.1104**	**162**	**323.6367**	160	353.9105
50	154	381.7177	152	388.6793	**168**	320.6099
60	157	365.2432	148	405.7894	143	425.2654
70	145	417.2395	149	401.2654	141	432.6251
80	146	412.7816	148	395.1475	147	412.2655
90	156	372.3642	142	429.3594	144	422.6539
100	149	398.0452	149	398.9106	157	364.8642
200	141	435.7954	141	436.2065	141	436.2556

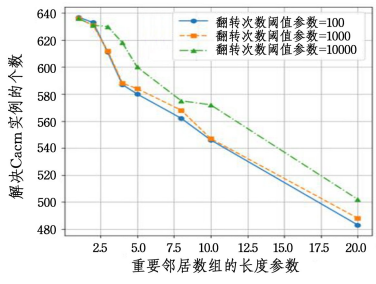

图 3　算法在不同参数下解决 PTN 实例的数量

Fig. 3　Algorithm-solved PTN instances under different parameters

从图 3 和表 2 可以看出，当翻转次数阈值分别设置为 100，1000 和 10000 时，重要邻居数组长度的最优值均集中在 30 到 50 之间。这表明，尽管翻转次数阈值的变化幅度达到了 10 倍，但其对算法结果的影响相对有限；相比之下，重要邻居数组长度参数的变化对结果的影响更加显著，仅通过调整数组长度增加 10 的幅度，就能够对求解效率产生明显的影响。

表 3　probSAT_PCCR 算法在 FCC 实例上的参数测试

Table 3　Parameter testing for probSAT_PCCR algorithm on FCC

numIN	mostFN					
	100		1000		10000	
	Solved	PARTime	Solved	PARTime	Solved	PARTime
1	**637**	**64.27266**	**636**	**65.66305**	**636**	**65.74411**
2	633	71.31881	631	71.02534	631	73.00519
3	611	103.86900	612	101.79400	630	81.91022
4	587	138.03210	588	136.08080	618	92.22362
5	580	148.14690	584	142.04550	600	118.67780
6	573	156.68540	577	148.56920	592	130.69810
7	567	163.54780	570	159.35640	584	135.86520
8	562	175.28470	568	165.99500	575	172.79490
9	553	183.65230	560	173.56230	574	161.65420
10	546	191.25840	547	196.72550	572	159.85830
20	483	208.58470	488	283.50790	502	263.56940

从图 4 和表 3 的数据可以看出，相较于翻转次数阈值，重要邻居数组长度参数对算法性能的影响更为显著。同时，随着重要邻居数组长度的增加，能够成功解决的 FCC 实例数量逐渐减少。尤其是在仅使用一个重要邻居数组的情况下，解决实例的数量达到了峰值。这一结果表明，对于 FCC 实例，仅需一个经常翻转的变量即可有效限制变量的翻转频率，从而实现对格局检测策略中翻转约束与邻居数组维护开销之间的良好平衡。通过这一优化，算法能够充分发挥其性能优势，显著提高求解效率。

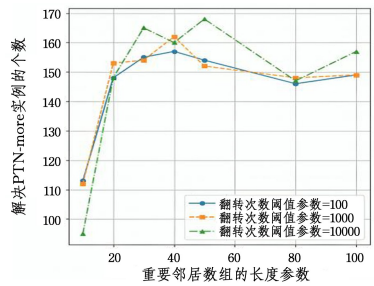

图 4　不同参数下解决 FCC 实例的数量

Fig. 4　FCC instances under different parameters

综上所述，通过重要邻居数组长度、最大翻转次数参数与已解决实例数量的散点图分析可以发现，重要邻居数组长度参数对求解结果的影响更为显著，尤其在 FCC 实例中这一趋势表现得尤为明显。实验结果表明，与最大翻转次数相比，邻居数组长度对算法性能的优化效果更加突出，是决定求解效率的关键因素。

基于实验数据的深入分析，最终确定了两类实例的最佳参数设置：对于 PTN 实例，将重要邻居数组长度参数设置为 50，最大翻转次数参数设置为 10000；对于 FCC 实例，则将重要邻居数组长度参数优化为 1，最大翻转次数参数设置为 100。

4.3　策略比较和重启机制

大量实验表明，在 probSAT 算法已解决的实例中，有 80% 的实例能够在 10 秒内成功求解。这表明，probSAT 算

法在运行初期具有快速降低不可满足子句的数量并找到解的能力。然而,在运行后期,算法往往会出现全局范围内的重复翻转现象[14],导致求解效率显著下降。

为解决这一问题,本文设计了一种重启机制,旨在充分利用 probSAT 算法前期快速降低不可满足子句的特点。通过在算法陷入效率瓶颈时及时触发重启,该机制能够有效避免全局重复翻转的积累,并重新激发算法的探索能力,进一步提升求解效率。

PTN 实例中,子句数量最多的 PTN-7819 实例,会在 50 万次翻转内,首次使得不满足子句数量降低到 20 个,不可满足子句数量占总子句数量的 0.105%。FCC 实例中,子句数量最多的 60-13823 会在 100 万次翻转内,首次将不可满足子句数量的占比降低到 0.112%。为了更好地平衡贪婪性和随机性,probSAT_PCCR 算法的重启机制会在翻转了 2 000 万次后生成初始解并重新开始解决实例。本文设计了添加重启策略的 probSAT_PCCR 算法和没有添加重启策略的 probSAT_PCCR* 算法的对比实验,来验证重启策略的有效性,结果如表 4 所列。

表 4 各种算法策略对比

Table 4 Comparison of various algorithm strategies

solvers	PTN-more		FCC	
	Solved	PARTime	Solved	PARTime
probSAT	26	895.52	499	2082.22
probSAT_PCC	76	693.47	405	405.65
probSAT_PCCR*	95	617.07	584	200.52
probSAT_PCCR	**168**	**320.60**	**637**	**64.27**

同时,为了验证两种新策略的有效性,实验测试了以下 3 种算法的性能表现:单独添加 PCC 策略的 probSAT_PCC 算法、同时添加 PCC 策略和重要邻居数组策略的 probSAT_PCCR 算法,以及原始 probSAT 算法。实验中,所有算法的参数均统一设置为 4.2 节中测定的最佳参数,以确保比较结果的公平性和可靠性。

从表 4 的实验结果可以看出,单独添加 PCC 策略的 probSAT_PCC 算法在 PTN-more 实例上比原始 probSAT 算法多解决了 50 个实例,性能提升达 192%。这充分说明了 PCC 策略在短子句实例中快速降低不可满足子句数量的优势。然而,在 FCC 实例上,probSAT_PCC 算法却少解决了 94 个实例,表明过长的邻居数组在复杂长子句实例中降低了算法的求解效率。

相比之下,同时添加 PCC 策略和重要邻居数组策略的 probSAT_PCCR* 算法表现更加优异。在 PTN-more 实例上,probSAT_PCCR* 比原始 probSAT 多解决了 69 个实例,性能提升达 265%;在 FCC 实例上,probSAT_PCCR 多解决了 85 个实例,性能提升 17.03%。这一结果表明,probSAT_PCCR 算法成功结合了 PCC 策略的快速求解能力与重要邻居数组策略的效率优化能力,在两类实例中均实现了显著的性能提升。

综上所述,probSAT_PCCR 算法综合了两种策略的优势,不仅克服了单一策略的局限性,还显著增强了算法在不同类型实例上的适应性与求解效率,展现了最优的性能表现。实验结果进一步验证了 probSAT_PCCR 算法在 SAT 问题求解中的实用性与可靠性。

添加了重启机制的 probSAT_PCCR 算法在解决实例数量上实现了不同程度的提升。与未添加重启策略的 probSAT_PCCR* 算法相比,其在 PTN-more 实例测试中多解决了 73 个实例,性能提升达 76.84%;在 FCC 实例测试中多解决了 53 个实例,性能提升了 9.07%。这一结果充分说明了重启策略的有效性。

重启机制通过在算法运行的后期打破全局范围的重复翻转现象,有效避免了算法陷入局部最优解,从而重新激发搜索效率。尤其在 PTN-more 实例中,重启策略利用 probSAT 算法初期快速降低不可满足子句数量的特性,进一步增强了算法的求解能力。而在 FCC 实例中,尽管实例更为复杂,重启策略依然实现了性能的显著提升。

综上所述,重启机制不仅有效弥补了 probSAT_PCCR 算法在后期求解效率下降的不足,还进一步强化了其在不同类型实例中的性能优势。

4.4 对比实验

为了验证 probSAT_PCCR 算法在性能提升方面的实际效果,将其与几种表现优秀的经典局部搜索求解器进行对比实验。

WalkSATlm 算法[11]:这是对经典 WalkSAT 算法的改进版本,采用了线性 make 策略(lm)以增强算法的贪婪性和局部搜索能力。作为局部搜索算法中的代表性方法,WalkSATlm 在许多实例中展现了稳定而高效的求解性能。

Sparrow 算法[22]:该算法是 2011 年随机竞赛组的冠军,以其独特的变量选择机制和高效的随机策略而著称。在复杂 SAT 实例求解中,Sparrow 算法凭借卓越的表现成为随机局部搜索领域的标杆之一。

Dimetheus 求解器[25]:2018 年表现最佳的 SLS 求解器之一,并两次获得随机竞赛组的冠军。其结合了精细的搜索策略与高效的实例处理能力,在随机实例中具有极高的求解效率,是当前局部搜索求解技术的重要参考基准。

FrwCBlm 求解器[26]:基于 WalkSATlm 算法的改进版本,通过增加新的启发式机制,在解决长子句 SAT 实例的实验中表现优异。

DCCASat 求解器[17]:是对 CCASat 算法的优化版本,创新性地结合了基于变量和基于子句的 CC 策略,并深入开发了格局检测策略。其在求解高难度随机 k-SAT 问题中表现卓越,至今仍是众多求解器的设计范本。

CSCCSat 求解器[27]:在 2016 年 SAT 竞赛随机组中荣获亚军。该求解器巧妙地将 FrwCB 与 DCCASat 相结合,根据问题的子句变量比动态选择执行不同的算法,从而具备了更强的适应性和灵活性。在求解器性能对比实验中,CSCCSat 是一个重要的参考标准。

通过与这些经典求解器的对比实验,旨在全面评估 probSAT_PCCR 算法在局部搜索领域的性能提升幅度。

本节实验所有参数依照 4.2 节测出的最佳参数设置,表 5 列出了不同求解器在 PTN 实例和 PTN-more 实例中的对比实验结果。表 6 列出了 FCC 实例测试求解器性能对比结果。

实验结果表明，probSAT_PCCR 算法在解决 PTN 实例和 PTN-more 实例的问题上表现优异，显著优于其他所有求解器。与次优的 DCCASat 算法相比，probSAT_PCCR 在 PTN-more 实例上多解决了 45 个实例，效率提升达 26.7%，同时惩罚时间减少了 108 秒。在 PTN 实例上，probSAT_PCCR 也多解决了 10 个实例，效率提升 100%，惩罚时间同样降低了 426 秒，降幅达到 75.2%。

这一实验结果充分展示了 probSAT_PCCR 算法在短子句实例求解中的显著优势。其在实例数量和时间开销上的双重优化，验证了 PCC 策略与重要邻居数组策略结合的有效性，不仅提高了求解效率，还在降低惩罚时间方面表现出色。总体而言，probSAT_PCCR 算法在这类实例的求解中展现了卓越的性能，为短子句 SAT 问题提供了一种高效的解决方案。

在 FCC 实例的实验测试中，改进算法在每一个编号的实例范围内均展现出较为优异的性能。与原始 probSAT 算法相比，改进算法多解决了 1 596 个实例，性能提升 33.5%；同时，惩罚时间减少了 160 秒，降幅达到 30.17%。这一结果表明，改进算法显著优化了原始算法在长子句实例中的求解效率。

相较于其他求解器，改进算法也表现出了优异的性能。与 sparrow 算法、WalkSATlm2013 算法、dimetheus 求解器相比，改进算法的求解实例数量分别提升了 3.6%，16.5%，87.7%，与 sparrow 算法性能接近；而相较于处理长子句实例更为优秀的 FrwCBlm 求解器、CSCCSat 求解器，以及 DCCASat 求解器，分别降低了 18.8%，18.6%，18.0%。

总体而言，通过融合 PCC 策略与重要邻居数组策略，本文改进算法能够针对不同实例动态调整参数，在处理 PTN 实例时表现出色，能够达到卓越的性能水平。而在面对长子句 FCC 实例时，新算法有效优化了变量选择与翻转操作，相较于原始算法显著提升了求解效率和资源利用率，进一步验证了本文算法在长子句 SAT 问题上的良好适应性。

表 5 PTN 实例，PTN，PTN-more 实例测试求解器性能对比

Table 5 Performance comparison of solvers for PTN instances and PTN，PTN-more instances

solvers	PTN-more		FCC	
	Solved	PARTime	Solved	PARTime
probSAT_PCCR	**168**	**320.60**	**20**	**140.04**
probSAT	26	895.52	1	956.83
dimetheus	11	1 016.26	3	871.65
sparrow	61	753.64	10	573.22
WalkSATlm2013	46	814.60	4	829.79
CSCCSat	21	915.29	8	654.08
DCCASat	123	501.40	10	566.03
FrwCBlm	21	920.43	8	660.74

表 6 FCC 实例测试求解器性能对比

Table 6 Performance comparison of solvers for FCC instances

name	#num	probSAT_PCCR		probSAT		FrwCBlm		sparrow		WalkSATlm2013		CSCCSat		DCCASat		dimetheus	
		solved	Par10	solved	Par10	solved	Par10	solved	Par10	solved	Par10	solved	Par10	solved	Par10	solved	Par10
41	469	323	311.88	180	617.85	**373**	**205.07**	314	332.13	231	509.71	373	205.05	372	207.32	23	951.24
42	491	336	316.68	254	483.22	**366**	**244.79**	314	360.84	306	377.23	366	254.83	366	254.94	25	949.71
43	485	297	388.11	236	514.28	**340**	**289.15**	288	406.35	251	483.46	340	299.15	341	297.23	58	881.33
44	520	316	393.10	278	468.54	**422**	**189.42**	302	419.38	296	431.78	422	189.46	413	206.58	32	938.73
45	524	298	432.08	226	570.24	**367**	**299.92**	290	447.43	247	529.42	367	299.92	368	298.09	36	931.68
46	528	372	295.73	311	411.39	**424**	**197.64**	365	309.34	318	397.81	424	197.61	423	199.46	62	883.54
47	516	332	356.73	246	525.08	**349**	**323.77**	331	358.94	291	436.16	349	323.78	350	321.91	45	913.43
48	494	305	382.74	253	488.42	**359**	**273.63**	304	384.80	299	395.33	359	273.59	359	273.77	36	927.20
49	475	284	402.29	234	508.10	**327**	**311.94**	282	406.40	227	523.91	327	311.93	324	318.10	40	916.18
50	512	376	265.84	231	550.019	**412**	**196.04**	370	277.64	299	417.45	412	196.09	412	196.24	48	906.82
51	558	412	262.17	284	491.46	**475**	**149.22**	408	270.22	306	452.88	461	174.09	451	192.05	60	893.32
52	491	339	311.24	252	487.03	**393**	**200.28**	292	407.51	252	486.78	393	200.29	395	196.33	37	925.01
53	484	289	402.98	250	483.70	**339**	**301.14**	289	403.28	264	455.29	339	301.06	339	301.22	35	928.10
54	540	226	582.48	194	641.28	**406**	**249.45**	216	600.63	206	618.67	408	245.94	405	250.90	42	922.94
55	557	408	268.44	300	462.15	**460**	**174.42**	400	282.31	343	385.80	460	174.41	453	187.05	53	905.76
56	493	259	475.22	174	647.74	**368**	**255.28**	247	500.15	200	595.28	368	255.21	372	247.49	8	8510.74
57	448	267	405.58	217	519.71	**359**	**199.34**	248	446.69	240	465.71	359	199.33	358	201.46	25	7171.46
58	496	336	324.12	219	560.33	**362**	**270.72**	308	379.73	266	464.52	362	270.72	357	280.55	32	7735.00
59	423	238	437.75	157	629.28	**272**	**358.04**	235	444.92	193	545.30	272	358.10	262	381.56	30	6551.73
60	495	343	309.21	264	467.24	**382**	**228.56**	322	351.05	272	450.83	382	228.52	382	228.42	49	7437.36
total	10000	6356	365.09	4760	525.05	**7555**	**245.14**	6125	388.14	5307	470.158	7543	246.33	7502	250.43	776	1162.22

结束语 综上所述，本文在 probSAT 算法的基础上提出了 probSAT_PCCR 算法，该算法结合了基于环境信息的概率选择策略（PCC 策略）和重要邻居数组策略。在 PTN 实例和 FCC 实例的实验中，probSAT_PCCR 算法展现出卓越的性能，其显著优势如下所示。

1）降低重复翻转概率：有效解决了 probSAT 算法中全局范围重复翻转的问题，从而提高了求解效率。

2）减少高价值变量被忽略的风险：通过 PCC 策略的引入，避免了格局检测策略可能导致的高价值变量选择偏差。

3）适应不同实例特性：融入重要邻居数组策略后，算法能够灵活调整参数以适应不同子句长度的实例，从而提升了对复杂实例的求解能力。

实验结果表明，在加入重启策略后，probSAT_PCCR 算法的性能得到了进一步提升，展现出更加稳定和高效的求解能力。

然而，本文算法也存在不足，在实现重要邻居数组策略时，代码实现尚未达到最高效的实现方式，以挖掘出策略的全部潜力，后续可以继续优化代码。

值得一提的是，格局检测策略已被广泛应用于多种算法中，而重要邻居数组策略则是对格局检测策略的一种改进和补充。基于 probSAT 算法提出的新策略，其通用性和灵活性为更多算法的优化提供了可能性。未来的研究将着眼于探索

这两种新策略在更广泛算法中的应用潜力，进一步推动随机局部搜索方法的进步和创新。

参 考 文 献

[1] COOK S A. The Complexity of Theorem-proving Procedures [C]// Proceedings of the ACM Symposium on Theory of Computing. 1971: 151-158.

[2] DARWICHE A. New advances inCompiling CNF into Decomposable Negation Normal Form[C]// Proceedings of the Eureopean Conference on Artificial Intelligence. 2004: 328-332.

[3] CHANG W J, XU Y. A Dynamic Decision-Making Strategy Based on Identifying Redundant Paths [J]. Journal of Computer Science and Technology, 2019, 42(10): 2309-2322.

[4] XU L, YU J P. Improved Bounded Model Checking on Verification of Valid ACTL Properties [J]. Computer Science, 2013, 40(6A): 99-102.

[5] KOCHEMAZOV S, ZAIKIN O, KONDRATIEV V, et al. MapleLCMDistChronoBT-DL, Duplicate Learnts Heuristic-Aided Solvers at the SAT Race 2019[EB/OL]. https://helda.helsinki.fi/bitstream/handle/10138/306988/sr2019_proceedings.pdf?sequence=1&isAllowed=y.

[6] BRAUNSTEIN A, MÉZARD M, ZECCHINA R. et al. Survey propagation: an Algorithm for Satisfiability, Random Struct [C]// Algorithms. 2005: 201-226.

[7] CAI S W, SU K. Comprehensive score: Towards Efficient Local search for SAT with Long Clauses[C]// Proceedings of IJCAI 2013. 2013: 489-495.

[8] SELMA B, KAUTZ H A, COHEN B. Noise Strategies for Improving Local Search [C]// Proceedings of the 12th National Conference on Artificial Intelligence. AAAI, 1994: 337-343.

[9] BALINT A, FROHLICH A. Improving Stochastic Local Search for Sat with A New Probability Distribution[C]// Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing. Springer, 2010: 10-15.

[10] LUO C, CAI S W, SU K. FocusedRandom Walk with Configuration Checking and Break Minimum for Satisfiability[C]// Proceedings of the International Conference on Principles and Practice of Constraint Programming. Springer, 2013: 481-496.

[11] CAI S W, LUO C. Improving WalkSAT forRandom k-satisfiability Problem with K > 3[C]// Proceedings of the 27th AAAI Conference on Artificial Intelligence. AAAI, 2013: 145-151.

[12] HUTTER F, TOMPKINS D A D, HOOS H H. Scaling and Probabilistic Smoothing: Efficient Dynamic Local Search for SAT[C]// Proceedings of Principles and Practice of Constraint Programming—CP 2002. 2002: 233-248.

[13] BIERE A. Splatz, Lingeling, Plingeling, Treeneling, YalSAT Entering the SAT Competition 2016 [C]// Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing (SAT 2016): Solver and Benchmark Descriptions. 2016: 44-45.

[14] FU H M, LIU J, WU G, et al. Improving Probability Selection Based Weights for Satisfiability Problems [J], Knowledge-Based Systems, 2022, 245: 108572.

[15] CAI S W, SU K. Configuration Checking with Aspiration in Local Search for SAT[C]// Proceedings of the 26th AAAI Conference on Artificial Intelligence. AAAI, 2012: 434-440.

[16] CAI S W, LUO C, SU K. Local Search for Boolean Satisfiability with Configuration Checking and Subscore[J]. Artificial Intelligence, 2013, 204: 75-98.

[17] LUO C, CAI S W, WU W, et al. Double Configuration Checking in Stochastic Local Search for Satisfiability[C]// Proceedings of the 28th National Conference on Artificial Intelligence. AAAI, 2014: 2703-2709.

[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] BALYO T. Modelling and Solving Problems Using SAT Techniques[D]. Charles University in Prague, 2014.

[21] BALINT A. Engineering Stochastic Local Search for the Satisfiability Problem[D]. University Ulm, 2014.

[22] BALINT A, FROHLICH A. Improving Stochastic Local Search for SAT with A New Probability Distribution[C]// Proceedings of the 13th International Conference on the Theory and Applications of Satisfiability Testing(SAT 2010). 2010: 10-15.

[23] KNUTH D E. The Art of Computer Programming, Volume 1: Fundamental Algorithms[M]//Boston: Addison-Wesley, 1997: 100-150.

[24] LUO C, HOOS H, CAI S. PbO-CCSAT: Boosting Local Search for Satisfiability Using Programming by Optimisation[C]// Proceedings of 16th International Conference. 2020: 373-389.

[25] BALINT A, MANTHEY N. DIMETHEUS. Dimetheus: solver description[C]// Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing. 2016: 37-38.

[26] LUO C, CAI S W, SU K, et al. Clause states based configuration checking in local search for satisfiability[J]. IEEE Transactions on Cybernetics, 2015, 45(5): 1028-1041.

[27] LUO C, CAI S W, WU W, et al. CSCCSat: solver description [C]// Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing(SAT 2016). 2016: 10-11.

JIA Shuheng, born in 2000, postgraduate. His main research interest is local search algorithms for solving SAT problems.

FU Huimin, born in 1991, Ph.D, master supervisor. Her main research interests include combinatorial optimization, constraint solving and automated reasoning.

（责任编辑：何杨）