



中国科学院大学  
University of Chinese Academy of Sciences

# 博士学位论文

## SAT 和 SMT 问题的混合求解算法及应用

作者姓名: 张昕荻

指导教师: 蔡少伟 研究员 中国科学院软件研究所

学位类别: 工学博士

学科专业: 计算机科学与技术

培养单位: 中国科学院软件研究所

2024 年 6 月



# **Hybrid Algorithms for SAT and SMT and Their Applications**

---

**A dissertation submitted to  
University of Chinese Academy of Sciences  
in partial fulfillment of the requirement  
for the degree of  
Doctor of Engineering  
in Computer Science and Technology**

**By**

**Zhang Xindi**

**Supervisor: Professor Cai Shaowei**

**Institute of Software, Chinese Academy of Sciences**

**June, 2024**



## **中国科学院大学**

### **学位论文原创性声明**

本人郑重声明：所呈交的学位论文是本人在导师的指导下独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的研究成果。对论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明或致谢。

作者签名：

日 期：

## **中国科学院大学**

### **学位论文授权使用声明**

本人完全了解并同意遵守中国科学院有关保存和使用学位论文的规定，即中国科学院有权保留送交学位论文的副本，允许该论文被查阅，可以按照学术研究公开原则和保护知识产权的原则公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存、汇编本学位论文。

涉密及延迟公开的学位论文在解密或延期后适用本声明。

作者签名：

导师签名：

日 期：

日 期：



## 摘要

命题逻辑可满足性问题 (SAT) 是判定给定的命题逻辑公式可满足性的判定问题。它是数理逻辑的基础问题，也是芯片设计和密码分析等领域的重要推理工具。可满足性模理论 (SMT) 在 SAT 上拓展了特定的一阶逻辑背景理论，表述能力更好，在软件验证与软件测试等领域有重要应用。SAT/SMT 算法主流方法是一种基于冲突驱动子句学习 (CDCL) 框架的算法，结合了回溯搜索和逻辑推理技术；另外，局部搜索方法是求解 SAT 和 SMT 的启发式搜索算法。本文聚焦于 SAT/SMT 的混合求解算法，研究了 SAT 问题 CDCL 算法的混合重启算法、SAT 和整数算术理论 SMT 的结合系统搜索和局部搜索的混合求解算法及应用，具体贡献如下：

(一) 基于 CDCL 的主流 SAT 求解器采用了高频的热重启策略，在重启时会保留变元序、变元相位和学习子句三类主要的搜索信息。我们发现随着变元序打分和相位的扰动，主流 CDCL 求解器仍呈现出运行时间的重尾分布现象，这启发我们利用该特点来提高算法的求解能力。本文针对重启信息是否应当保留这一问题展开了研究，提出了冷重启（即每次重启时丢掉部分或全部搜索信息）的概念，设计了低频冷重启与高频热重启的混合算法。我们深入分析了三类搜索信息的作用，并成功改进了求解性能。依赖分析结果，我们进一步提出了其并行版本。集成了相关技术的并行求解器在 SC22 测试样例集上相较于其他顶尖求解器，性能领先了 24%。该求解器还以大幅优势获得了国内首个 SAT 比赛并行主赛道的冠军。

(二) “Bart Selman” 命题逻辑十大挑战的第七项为：“设计一个系统搜索与局部搜索的混合算法，其效果能超过混合前两种算法的最优性能”。在本文之前，虽然已有大量混合求解相关的工作，但由于求解引擎之间没有求解过程中的深度信息交流，因此均未能满足该挑战的要求。本文提出了一种以 CDCL 为主体，在有希望的分支调用局部搜索对搜索空间进行采样的混合求解算法。同时，我们也深入研究了局部搜索和 CDCL 之间的深度信息交互技术。局部搜索从 CDCL 获取一致性较强的初始解，并反馈冲突频率信息和高质量赋值，用以辅助 CDCL 算法。实验结果表明，该方法可以大幅提高性能，并成功求解出大量在混合前无

法求解的样例，从而成功应对了挑战。例如，集成了我们技术的三个主流求解器在 2020 年 SAT 比赛标准测试集合上，多求解出了 62、67 和 10 个实例。集成了相关技术的求解器在 2020 年 SAT 比赛主赛道上获得了一项冠军，并且自此以后，主流求解器均采用了相关混合技术。

(三) 整数算术理论 SMT (简称 SMT(IA)) 的主流算法之一是在 CDCL 上拓展了整数算数理论求解技术的 CDCL(T) 算法。本文将上述高效的 SAT 混合算法迁移到了整数算术理论 SMT 问题，设计了 CDCL(T) 与两种局部搜索的双层混合算法。外层为 CDCL(T) 与“独立局部搜索”的交替调用。内层则以 CDCL(T) 为主，并在有希望的命题骨架分支下调用轻量级的“CDCL(T) 指导的局部搜索”进行搜索空间采样。CDCL(T) 也会从局部搜索中提取命题骨架变元的冲突频率和相位信息来改进自身的启发式。实验表明该方法在整数算术理论大幅领先于其它主流 SMT 求解器，尤其是对于非线性整数算术 (NIA) 问题。与最佳求解器相比，我们的方法能在标准测试样例集合中多求解出 10.3% 的实例。应用了该技术的求解器参与了 2023 年 SMT 比赛，并荣获了模型验证总比分最大领先奖、最大贡献奖以及多个整数算术理论相关细分赛道的冠军。

(四) 本文还深入研究了 SAT 问题在 EDA 和密码学中的应用。组合等价性验证 (CEC) 是 EDA 芯片设计优化和验证的核心问题，但是在面临数据通路类型的实际工业样例时，仅依靠 SAT 的求解能力不足以完成包含大量乘加混合单元的电路验证任务，本文在主流的基于 SAT-sweeping 的 CEC 算法中融合了一套基于精确概率仿真 (EPS) 的验证方法，从而大幅提高了验证效率。实验结果表明本文的方法在 12% 工业实例上能超出主流求解器 ABC 千倍性能。此外，本文选择“环上前馈序列密码”作为研究案例，探讨了混合 SAT 求解器在密码分析中应用的方法，提出了一种基于“轮增量编码”的策略。相比需要数千核心超级计算机数周完成分析的传统方法，本文的方法可以在单核心数小时内、并行百秒级别内完成初态恢复，有巨大的速度优势。该方法也荣获了“强网杯”密码数学专项赛全国冠军。

**关键词：**可满足性问题, 可满足性模理论, 混合求解算法, 组合等价性验证

## Abstract

The satisfiability problem (SAT) is a decision procedure for determining the satisfiability of a given propositional logic formula. It is a fundamental problem in mathematical logic and serves as a core reasoning tool in areas such as chip design and cryptanalysis. The Satisfiability Modulo Theory (SMT) extends specific first-order logic background theories on SAT, with better expressive power and important applications in software validation and testing. The mainstream method of SAT/SMT algorithm is a systematic search algorithm based on the Conflict-Driven Clause Learning (CDCL) framework, combined with backtracking search and logic reasoning; On the other hand, stochastic local search is a heuristic search algorithm for solving SAT and SMT. This thesis focuses on the hybrid algorithms of SAT/SMT, studying the hybrid restart methods of SAT CDCL algorithm, hybrid SAT/SMT algorithms for tight cooperating the systematic search algorithms and local search, and the applications of hybrid methods. Specifically, the contributions are as follows:

(1) The mainstream CDCL-based SAT solvers adopt a high-frequency warm restart strategy, which preserves three main search information during restart: variable order, variable phase, and learning clause. We found that with the perturbations of variable branching score and phase, current CDCL solvers still exhibit a heavy-tailed distribution according to the running time, which can be used to improve the performance of SAT solvers. This thesis studies the question of ‘Should the learned information be kept after restart’, proposes the concept of cold restart (i.e., forgetting part or all search information after restart), designs a hybrid algorithm for combining the low-frequency cold restart and high-frequency warm restart, deeply analyzes the role of three types of search information, and improves the performance of solvers. This thesis proposes a parallel version based on the previous analysis results, and a solver that integrates relevant technologies outperforms the other top solvers with at least 24% PAR2 score on standard SC22 benchmarks, which won the championship of the parallel main track in the SAT competition with a large advantage.

(2) The 7th challenge of Bart Selman et al.'s propositional reasoning and search is “Demonstrate the successful combination of stochastic search and systematic search techniques, by the creation of a new algorithm that outperforms the best previous examples of both approaches” . There were numerous works about hybrid algorithms before this thesis, but there is no in-depth information exchange between solve engines during searching , and none of them met the requirements of this challenge. This thesis proposes a hybrid paradigm that takes CDCL as the main body, which calls local search in promising branches to sample the search space. Meanwhile, we further design a tight information interaction technology between them. Local search obtains high-consistency initial solutions from CDCL and provides feedback information to improve CDCL with variables conflict frequency and high-consistency variable phases. The experiment shows that this method can significantly improve performance and solve a large number of instances that cannot be solved before mixing, first successfully answering the challenge. For example, the integration of our techniques allows the three CDCL solvers to solve 62, 67, and 10 more instances in the benchmark of SAT Competition 2020. A solver that integrates relevant technologies won a championship in the Main Track of SAT competition 2020, and since then, mainstream solvers have adopted related hybrid technologies.

(3) One of the mainstream algorithms of SMT with integer arithmetic, SMT(IA), is the CDCL(T) algorithm, which extends IA theoretical reasoning techniques on CDCL. This thesis migrates the efficient SAT hybrid algorithm mentioned above to SMT(IA) and designs a two-layer hybrid algorithm. The outer layer is an alternating call of CDCL(T) and the ‘Independent local search’. The inner layer takes CDCL(T) as the main body, and a lightweight ‘CDCL(T)-guided local search’ is used for sampling search space in promising proposition skeleton branches. CDCL(T) also uses conflict frequency and phase information of proposition skeleton variables from local search to accelerate the solving process. The experiment shows that this method significantly outperforms other mainstream SMT solvers on the IA benchmark, especially for non-Linear integer arithmetic (NIA), and our solver can solve 10.3% more instances than the best competitor on the standard benchmark. The solver that applied this technology

participated in the 2023 SMT competition, winning the biggest Lead Award, Largest Contribution Award, and multiple championships in NIA-related sub-tracks.

(4) This thesis investigates the application of the SAT problem in EDA and cryptanalysis. Combinational Equivalence Checking (CEC) is a key problem in EDA chip design optimization and verification. However, when faced with practical industrial datapath circuits, relying solely on the solving ability of SAT is not enough to complete circuit verification problems, which contain a large number of multipliers and adders. This thesis integrates the exact probability simulation verification method into the mainstream SAT-sweeping CEC algorithm, greatly improving the verification efficiency. The experimental results show that the method of this thesis outperforms the mainstream solver ABC by 1000 times on 12% industrial instances. Meanwhile, this thesis chooses a stream cipher as a study case to analyze the methodology of hybrid SAT solvers in cryptanalysis, and proposes a strategy named “round incremental encoding”. Compared to traditional methods that require thousands of cores to finish the attack in weeks, we can finish the initial state recovery within hours on a single core and a few minutes on the cluster, which has a significant advantage in terms of runtime. This method has won the national championship in the ‘QiangWang Cup’ cryptomathematics special competition.

**Keywords:** SAT, SMT, Hybrid Methods, CEC



## 目 录

<b>第 1 章 绪论</b>	1
1.1 选题背景及研究意义	1
1.2 研究内容与主要贡献	4
1.3 行文结构	7
<b>第 2 章 SAT、SMT 问题背景知识与相关工作</b>	9
2.1 SAT 与 SMT 问题基本概念	9
2.2 SAT 问题求解算法	14
2.2.1 SAT 问题完备算法	14
2.2.2 SAT 问题的非完备算法	23
2.2.3 SAT 问题并行求解技术	27
2.3 SMT 问题求解算法	28
2.3.1 惰性算法	28
2.3.2 积极算法	31
2.3.3 SMT 问题的其它求解算法	32
2.4 SAT、SMT 问题相关工作历史	34
<b>第 3 章 SAT 问题混合重启策略研究</b>	39
3.1 引言	39
3.2 热重启下的求解时间差异性现象	41
3.2.1 初始变元的变化研究	41
3.2.2 初始相位的变化研究	42
3.3 冷启动策略的实现及其评估	44
3.3.1 冷重启策略的实现	45
3.3.2 冷重启策略的评估	46
3.3.3 冷重启策略的进一步分析	50
3.4 本章小结	56
<b>第 4 章 SAT 问题混合求解算法</b>	59
4.1 引言	59
4.2 (回退) 松弛 CDCL 混合求解框架	61
4.3 基于局部搜索的 CDCL 改进策略	63
4.3.1 基于局部搜索解的相位重置技术	64
4.3.2 基于局部搜索冲突频率信息的变元分支策略	66

4.4 实验评估 .....	68
4.4.1 混合技术有效性分析 .....	69
4.4.2 两种引擎间的协作性分析 .....	72
4.5 本章小节 .....	74
<b>第 5 章 整数算术理论 SMT 问题的混合求解算法 .....</b>	<b>75</b>
5.1 引言 .....	75
5.2 结合 CDCL(T) 与局部搜索的混合求解框架 .....	77
5.2.1 SMT(IA) 问题的双层混合框架 .....	77
5.2.2 CDCL(T) 指导的局部搜索 .....	80
5.3 基于局部搜索的 CDCL(T) 改进策略 .....	82
5.3.1 基于局部搜索解的 CDCL(T) 相位重置技术 .....	83
5.3.2 基于局部搜索冲突频率的 CDCL(T) 变元分支策略 .....	84
5.4 实验评估 .....	85
5.4.1 与前沿求解器的对比 .....	87
5.4.2 组件策略有效性分析 .....	92
5.4.3 两种引擎间的协作性分析 .....	95
5.5 本章小结 .....	97
<b>第 6 章 混合求解技术在 EDA 和密码分析中的应用 .....</b>	<b>99</b>
6.1 基于 SAT 求解和精确仿真的组合等价性混合验证方法 .....	99
6.1.1 背景知识 .....	101
6.1.2 混合求解算法设计 .....	104
6.1.3 相同结构检测 .....	109
6.1.4 实验评估 .....	111
6.2 混合 SAT 求解器在流密码分析中的应用 .....	118
6.2.1 环上前馈序列密码问题 .....	120
6.2.2 基本求解思路 .....	121
6.2.3 轮增量分析与 SAT 求解 .....	124
6.2.4 实验评估 .....	128
6.3 本章小结 .....	132
<b>第 7 章 总结与展望 .....</b>	<b>133</b>
7.1 工作总结 .....	133
7.2 未来工作 .....	134
<b>参考文献 .....</b>	<b>137</b>
<b>致谢 .....</b>	<b>161</b>
<b>作者简历及攻读学位期间发表的学术论文与研究成果 .....</b>	<b>163</b>

## 图形列表

1.1 主要研究内容与贡献 .....	7
1.2 文章章节内容组织结构 .....	8
2.1 蕴含图与 First UIP 示例 .....	16
2.2 不同的回退方法示例。 .....	17
2.3 CDCL 框架核心组件间的交互逻辑概览 .....	18
2.4 CDCL(T) 框架核心组件间的交互逻辑概览 .....	28
3.1 对数尺度下不同初始变元序的求解时间变化比率 .....	43
3.2 串并行冷重启技术与对比求解器的 CDF 图 .....	56
4.1 CDCL 松弛示意图 .....	62
5.1 CDCL(T) 与局部搜索的混合框架 .....	77
5.2 HybridSMT (Hybrid+Z3) 与主流 SMT(IA) 求解器、基求解器间的比较	91
5.3 HybridSMT 与 Hybrid+Z3 与其他主流求解器在 SAT、UNSAT、ALL 样例上的 CDF 图 .....	93
5.4 五个不同变种的关键差异图 .....	95
6.1 Miter 电路的结构示意图 .....	101
6.2 基于 Sweeping 的 CEC 典型算法流程图 .....	103
6.3 Sweeping 示意图 .....	104
6.4 CEC 算法的主框架算法 .....	107
6.5 将 PI 概率信号取值分割成 $K$ 个小的数 .....	107
6.6 相同结构检测 (IDS) 技术的示意图 .....	111
6.7 不同异或链密度打分的待测等价点对上 EPS 引擎与 SAT 求解器验证时间的对比 .....	114
6.8 一个精心设计的流密码算法 .....	121
6.9 $f_0, f_1, f_2, f_3$ 的相对独立性 .....	121
6.10 轮增量编码示意图 .....	127
6.11 轮增量编码 SAT 求解技术 .....	128



## 表格列表

2.1 常见的 SMT 问题背景理论 .....	11
3.1 随机采样实验结果汇总 .....	42
3.2 遗忘顺序 (FO)、遗忘相位 (FP) 和遗忘子句 (FC) 冷重启的实验结果 .....	48
3.3 遗忘多类信息的冷重启策略最优配置 .....	50
3.4 不同 LBD 阈值标准下的 FC 实验结果 .....	52
3.5 不同 LBD 层级的学习子句的使用率标准化统计数据 .....	52
3.6 并行 FO 在两个主流并行 SAT 求解器上的实验结果 .....	53
3.7 并行 FO 的加速比分析 .....	54
4.1 相位重置技术不同相位选择概率 .....	66
4.2 SAT 比赛 2017 年–2020 年数据集上的实验结果 .....	70
4.3 在 FCC 实例集上与主流方法的对比 .....	72
4.4 有关局部搜索在 CDCL 中作用的分析 .....	73
5.1 每一种相位的概率分布 .....	84
5.2 与近期 SMT 比赛中 5 个主流求解器在 LIA 样例的实验结果 .....	88
5.3 与近期 SMT 比赛中 4 个主流求解器和 barcelogic 最佳版本的对比 .....	89
5.4 Hybrid+Z3: 利用 HybridSMT 改进 Z3 求解器 .....	92
5.5 混合方法的有效性分析 .....	93
5.6 组件的高效性分析 .....	94
5.7 分析 HybridSMT 中内部局部搜索求解器的作用 .....	96
6.1 SAT 和 EPS 工具验证 ‘ec_h1’ 样例的求解时间对比 .....	106
6.2 hybridCEC 与其它方法的对比 .....	115
6.3 hybridCEC 的过程中求解器数据 .....	116
6.4 hybridCEC 关键技术的有效性分析 .....	117
6.5 不同条件的 CNF 编码的第 2 分位求解情况 .....	129
6.6 不同条件的 CNF 编码的第 3 分位求解情况 .....	130



## 符号列表

### 缩写

SAT	命题逻辑可满足性（问题）
SMT	可满足性模理论
IA	整数算术理论
NIA	非线性整数算术理论
CEC	组合等价性验证
CNF	合取范式
CDCL	冲突驱动的子句学习
SLS	随机局部搜索
EPS	精确概率仿真
FO	遗忘变元选择顺序
FP	遗忘变元相位选择
FC	遗忘学习子句
EDA	电子设计自动化
SC	国际 SAT 比赛
SMT-COMP	国际 SMT 比赛
VBS	虚拟最佳求解器



## 第1章 绪论

*The story of satisfiability is the tale of a triumph of software engineering,  
blended with rich doses of beautiful mathematics.*

—— Donald E. Knuth, *The art of computer programming*

可满足性问题的故事是软件工程胜利的传说，  
融合了丰富、美妙的数学。

—— 高德纳, 《计算机程序设计艺术》

### 1.1 选题背景及研究意义

广义的可满足性问题指的是判定给定的逻辑可满足性的问题，狭义的可满足性问题特指命题逻辑可满足性问题（SAT），它是第一个被证明的 NP-完全问题，是计算机科学、人工智能等领域的核心研究问题。可满足性模理论（SMT）属于 SAT 问题的拓展版，允许判定的逻辑公式为特定背景理论下的一阶逻辑公式，对于实际问题的建模和表述能力更强。可满足性问题的研究历史悠久，有重要的理论应用价值和经济实用价值，长期吸引着国际上学术界和工业界的兴趣。

可满足性问题的研究有悠久的历史。人们对于可满足性问题有广泛的兴趣主要在于它高效的建模和求解能力，可以说可满足性问题站在了逻辑学、图论、计算机科学、运筹学等多门学科交织的十字路口。很多源自于这些学科的问题一般都会存在一种归结到可满足性问题的方法。由于可满足问题与逻辑学有紧密的联系，可满足性问题的历史可以视作是对逻辑学历史根源的展开。尽管目前该问题已经逐渐发展成为解决各种实际问题的建模工具，但是它最初是为了通过计算机设计来模拟人类的思维和科学推理而开发的一门学科<sup>[1]</sup>。逻辑学最早出现于雅典（384-322 B.C.）亚里士多德的《工具论》，其中提出了“三段论法”、命题可满足的概念；后来，斯多葛学派（300s-200s B.C.）提出了目前常见的逻辑连接词。两大古典哲学学派建立了以演绎逻辑为中心的逻辑体系，指引了后续 2000 多年逻辑学的发展。中世纪时期逻辑学拓展了量词、模态算子、认知算子、限制性和非限制性关系从句等更丰富的逻辑语言，建立了经院逻辑体系，通过教育和实践普及了逻辑学知识。文艺复兴时期，培根提出了归纳法，笛卡尔和莱布

尼茨首次提出用代数系统来进行科学推理，形成了数理逻辑的基础。近代逻辑学的发展进入蓬勃发展时期，出现了首个可以解决形式逻辑的机器，且在 Boole 的布尔代数、Jevons 的逻辑钢琴和 Veen 的文氏图等工作的帮助下，终结了传统的亚里士多德演绎推理 2000 年的统治地位，使得逻辑学转向更加严谨和基于数学的方法。19 世纪，柯西等人建立更加精确的数学基础概念的思想间接的促进了康托尔朴素集合论和罗素修正的集合论的提出，弗雷德提出了数学可以划归为逻辑的思想和谓词逻辑的概念。与此同时，希尔伯特也提出了“将数学形式化，以便通过有限步骤自动证明任何数学命题真假”的构想，并与贝奈斯将逻辑系统严格的分为一阶逻辑与高阶逻辑。1931 年，哥德尔通过不完备性定理否定了希尔伯特的构想，并提出了递归函数的概念。命题逻辑和一阶逻辑在语法上是完备的。后来，丘奇和图灵分别独立的提出了  $\lambda$  演算<sup>[2]</sup> 与图灵机<sup>[3]</sup> 两种计算模型，并被克莱尼证明与哥德尔的递归函数之间是等价的<sup>[4]</sup>，它们在本质上都是描述可计算函数的方法，探索了用计算机回答一阶逻辑可满足性的问题的可能性。由于当时人们普遍认为，在给定适当公理的前提下，所有经典数学的推理均可以表示为一阶逻辑<sup>[5]</sup>，早期的自动推理多旨在证明一阶逻辑范畴的定理，然而没有充分的意识到问题的复杂性和求解器存储的需求，研究出现了多次失败的尝试。直到 1960 年基于归结原理的 DP 算法以合取范式成功替代了析取范式<sup>[6]</sup>、1962 年 DPLL 算法<sup>[7]</sup> 的提出，该问题才得到了改善，使得逻辑学求解器可以在实际应用中发挥作用。自此，以 SAT 问题为代表的逻辑学求解器进入了算法研究的蓬勃发展时期。

可满足性问题有重要的理论应用价值。可满足性问题的理论价值在于它揭示了逻辑、计算、推理和算法等多个领域的深刻联系，并为这些领域的研究提供了强大的工具和框架，具体体现在多个方面。SAT 问题作为逻辑学中的一个基本问题，为数学和逻辑学，甚至信息学之间的联系提供了桥梁<sup>[8]</sup>。同时，SAT 问题也是研究逻辑系统完备性和可判定性的关键，也是计算复杂性理论的基石<sup>[9]</sup>。一方面，SAT、SMT 问题作为形式化方法中的核心方法，交互式定理证明工具的研制上发挥了重要价值，例如高阶逻辑的交互式理论证明器 HOL<sup>[10]</sup>、Isabelle<sup>[11]</sup>、ACL2<sup>[12]</sup>、Coq<sup>[13]</sup>。另一方面，借助逻辑求解器，很多数学问题得到了推动或解决，例如，Heule 等人借助 SAT 工具推动了布尔毕达哥拉斯三元组问题<sup>[14]</sup>、Schur 数 5 问题<sup>[15]</sup>、Keller 猜想<sup>[16]</sup> 等问题的发展，Bright 等人利用 SAT 求解器证明了

几何学中 Lam 问题是不存在的<sup>[17]</sup>，Huang 和 Liu 等人对正交 Golf 设计、幂等拟群大集和带洞拉丁方的存在性等问题的结果做出了改进<sup>[18,19]</sup>。

可满足性问题有重要的经济实用价值。SAT 求解器已经被应用于多个领域，如为美国联邦通讯委员会盈利 70 亿美元的带宽拍卖项目<sup>[20]</sup>、Intel Core7 处理器设计、Windows 7 操作系统驱动的验证<sup>1</sup>、密码分析<sup>[21]</sup>、形式化验证和模型检查<sup>[22]</sup>等。尤其对于电子设计自动化（EDA）行业，SAT 求解器是其中不可或缺的核心工具。比如，EDA 的许多环节，包括逻辑综合<sup>[23,24]</sup>、自动测试向量生成<sup>[25]</sup>、等价性验证<sup>[26,27]</sup>、功耗检查<sup>[28]</sup>等关键任务，都需要调用 SAT 求解器。另一方面，SMT 借助其更强的表达能力，在神经网络验证<sup>[29]</sup>、模型检查<sup>[30]</sup>、博奕论<sup>[31]</sup>和程序综合<sup>[32]</sup>等方面都有重要的应用价值。SMT 在软件验证和软件分析中的价值也备受关注，微软、英特尔、华为、Cadence、NEC 等公司都在开展与 SMT 相关的研究项目。SMT 求解器已经成为软硬件系统测试与验证领域的核心工具，被应用于状态可达性验证<sup>[33]</sup>、不变式生成<sup>[34]</sup>和终止性/非终止性证明<sup>[35,36]</sup>等问题上。著名的 SMT 求解器，包括微软的 Z3<sup>[37]</sup>，爱荷华大学和斯坦福大学的 CVC5<sup>[38]</sup>，以及斯坦福国际研究院的 Yices2<sup>[39]</sup>，都在工业领域发挥了重要作用，被集成于多个研究领域的自动工具之中，如模型检测工具 BLAST<sup>[40]</sup>，程序分析工具 KLEE<sup>[41]</sup>等。

可满足性问题有重要的影响力。SAT 问题是计算机科学领域的一个基础且重要的问题，Stephen Cook 开启了 NP 完全理论的研究，凭借对 SAT 问题 NP-完全性的证明工作，被授予了计算机科学最高领域的最高奖项——“图灵奖”。2000 年，美国克莱数学研究所提出了千禧年的七大数学难题之首——“P vs. NP” 问题，SAT 问题作为该问题的研究焦点一直备受关注。可满足性问题也获得另外两位图灵奖得主的高度评价。高德纳评价 SAT 问题“显然为一个必杀技应用”（evidently a killer app），他也将 SAT 算法与编码技术纳入到“计算机程序设计艺术”系列丛书，并单列为第 4 卷第 6 篇，并为此编程超过 300 个程序；Edmund Clarke 评价 SAT 问题为“高效的 SAT 求解是 21 世纪计算机科学的关键技术”（efficient SAT solving is a key technology for 21st century computer science）。图灵本人在其被誉为“计算机理论的奠基之作”的《可计算术及其在可判定性问题上的应用》（《On Computable Numbers, with An Application to the Entscheidungsproblem》）文章中的

<sup>1</sup><https://www.cs.utexas.edu/~marijn/sat15/>

核心工作被视为是用机器回答一阶逻辑可满足性的一次探索<sup>[18]</sup>。为了提高 SAT、SMT 问题的影响力，国际 SAT 协会举办了 SAT 会议（CCF-B，首届举办于 1996 年，至 2023 年 12 月已经举办 26 届），同时也举办了国际 SAT 比赛<sup>2</sup>（自 2002 年以来每年举办一次）和国际 SMT 比赛<sup>3</sup>（自 2005 年以来每年举办一次）等赛事，吸引了来自麻省理工、斯坦福、卡耐基梅陇、普林斯顿等国际顶尖高校和谷歌、微软、华为等国内外公司的学者们竞相角逐。

## 1.2 研究内容与主要贡献

现代 SAT 求解器成型的标志是 1996 年 Silva 等人提出的 CDCL 方法。之后，国际上不同团队对 CDCL 算法做了改进<sup>[42–46]</sup>，使得其性能得到了大幅改进<sup>[47]</sup>，在学术界和工业界发挥了巨大的应用价值。另一方面，局部搜索算法自 90 年代提出以来，在经历了多轮迭代之后也有了长足的发展，虽然与 CDCL 在综合求解能力上有巨大差距，但是在解决一些数学问题和实际应用问题上有一定的互补能力<sup>[48,49]</sup>。近五年，有关 SAT 问题的研究的焦点主要集中于 CDCL 的改进，并产生了两个比较有代表性的工作。2020 年，德国形式化专家 Biere 等人研制了一套工程高度优化的 SAT 求解器 kissat<sup>[50]</sup>。该求解器在过程中处理、组件协调、算法调度上做了大量的优化，成为了目前的标杆。近两年 SAT 比赛中大部分求解器都是基于 kissat 系列求解器开发的。另一方面，CDCL 与局部搜索算法间深度混合求解技术成为了近期研究的热点，凭借该技术，CDCL 的求解性能得到了大幅的提升。目前，例如 kissat、CaDiCaL<sup>[51]</sup>、CryptoMiniSat<sup>[52]</sup>、Relaxed 等前沿求解器，均采用了混合求解技术，该技术已经成为了 SAT 求解器中的一项标准技术。

SMT 问题作为 SAT 问题的拓展问题，相关的研究工作起步于 2000 年左右，涉及了诸如位向量、算术理论、字符串理论等背景知识，在软件测试、形式化验证等领域发挥了巨大的价值。SMT 求解器的代表算法主要有积极算法与惰性算法，均可以视作是基于 SAT 算法的拓展算法。非线性整数算术领域，适配性最好的技术为以 CDCL(T) 为主的惰性算法，此外，积极算法<sup>[53]</sup>、MCSat<sup>[54]</sup>、线性化<sup>[55]</sup>等求解技术也有不错的性能。

<sup>2</sup><http://www.satcompetition.org/>

<sup>3</sup><https://smt-comp.github.io/>

相比之下，SMT 局部搜索算法的研究比较薄弱。但是最近几年，局部搜索算法也开始出现于 SMT 领域。2015 年，局部搜索算法首次于出现在位向量领域，效果达到了与主流位向量 SMT 求解器互补的效果。2021 年，整数差分领域出现了第一个算术领域的局部搜索算法，综合性能超出了已有完备算法，且该算法与 CDCL(T) 算法的简单组合取得了当年 SMT 相关赛道的冠军<sup>[56]</sup>。后来，局部搜索算法也进入了一段小的爆发期，算术理论的不同子领域也相继出现了局部搜索算法<sup>[57-59]</sup>，这些算法的性能在子领域内超过了已有经典算法的性能且表现出一定的互补性。

本文的主要研究内容为 SAT、SMT 问题的混合求解算法，以提升其求解实际问题中的能力，本文主要围绕以下几个方面开展研究：

(1) 早期的组合优化算法中存在严重的运行时间的重尾分布现象，即启发式受到少许扰动，程序的求解时间会发生巨大的变化，学者们提出采用彻底重启的策略，来防止其在不利搜索区域一直搜索，以提升求解效率。主流 SAT 问题求解框架多为基于 CDCL 的回溯型搜索算法，并采用了保留历史搜索信息的热重启策略。通过预实验，本文发现目前主流的 CDCL 求解器仍存在明显的求解时间的重尾分布现象，这引发我们思考：“重启时的历史信息是否应该被保留？”即信息的保留是否会使得求解器在一个不利的搜索空间内闲逛，并需要引入一种更为彻底的重启来改进效率。为此，本文提出了冷重启的概念，会在重启时遗忘变元顺序、相位和学习子句三类常见的学习信息，并通过大量的实验分析研究了其中的规律。本文发现不同的冷热重启混合策略会影响求解器在不同种类实例上的性能，且通过采用冷重启策略，本文可以较大幅度的改进可满足实例的求解能力。此外，本文选取了其中综合性能最佳的冷重启策略，遗忘变元序，对细分样例上的性能和其并行化做了进一步的探索。我们也进一步针对学习子句和共享子句的管理展开了讨论，研究了学习子句在串行求解中发挥的作用、冷重启时学习子句的取舍规律以及学习子句在并行子句共享时的作用。

(2) 1997 年，Bart Selman 等人提出了命题逻辑十大挑战，其中的第七项为关于“混合算法设计”的挑战—“成功设计一个局部搜索与系统搜索的混合求解算法，新设计的算法可以超过混合前的两个方法的最佳性能”。自该挑战提出以来，产生了很多有关混合算法的研究工作，但是本文之前的工作大都是两类技术的顺序调用和简单信息传递，没有在求解过程中深度的交互启发式信息，均没有

完成该挑战。本文提出了一种以 CDCL 为主体，并在希望分支上调用 SLS 算法进行搜索空间采样的方法。CDCL 与局部搜索算法之间会进行深度信息交流。每当遇到希望分支时，局部搜索算法会进入一种非回溯的松弛 CDCL 模式，并为局部搜索算法构建一个高一致性的完全赋值。另一方面，CDCL 算法会从局部搜索算法中获取信息来改进自身的启发式。局部搜索算法会搜集过程中遇到的变元冲突频率和高一致性赋值，分别用于改进 CDCL 算法的变元分支启发式（选择顺序）与相位决策启发式（赋值倾向）。本文在三个主流的 SAT 求解器上实现了本文的混合求解技术，对提出的策略执行了有效性分析，实验表明该混合求解技术可以大幅提高了三个基求解器的性能，解决了大量基求解器无法求解的实例。我们首次解决了 Bart Selman 命题逻辑十大挑战的第七项挑战，相关成果多次获国际 SAT 比赛冠亚军、并获得 SAT 会议“最佳论文奖”。

(3) SMT(IA) 问题在 SAT 问题的基础上拓展了非线性整数算术背景理论，在软硬件测试领域有重要的应用价值。根据 SMT 比赛的结果，本文之前综合性能最好的方法为 CDCL(T) 算法，且并不存在一套高效的深度混合求解算法，据此，我们拟将 SAT 成功的混合求解技术迁移到了 SMT(IA) 问题的求解中。本文提出了一套双层混合策略。外层为 CDCL(T) 与“独立局部搜索”算法的交替调用。内层则是一种以 CDCL(T) 为主，并在遇到希望分支时，调用“CDCL(T) 指导的局部搜索”算法进行空间采样的方法。类似地，CDCL(T) 会给局部搜索算法传递高质量的子公式，而局部搜索算法也会利用过程中收集到的高一致性赋值和冲突频率来改进 CDCL(T) 的启发式。为了实现上述的深度信息交互策略，我们设计了一系列的交互逻辑与启发式技术，以适配 CDCL(T) 中的抽象命题逻辑骨架。大量的实验分析证明了该方法的有效性，我们的求解器可以在 SMT-LIB 的 NIA 样例集合上比最好的求解器多求解 10.3% 的例子。实现了该技术的求解器获得了 SMT 竞赛模型验证总比分“最大领先奖”与“最大贡献奖”两项金牌。此外，实验表明本文的方法对于求解软件工程终止性验证等问题有明显的推动作用。

(4) 本文进一步研究了 SAT 在电子设计自动化 (EDA) 和密码学中的应用。EDA 方面，本文选取了组合等价性验证 (CEC) 问题进行研究，本文针对数据通路电路的 CEC 问题，提出将一种精确概率仿真 (EPS) 的验证方法嵌入到以 SAT-sweeping 为主体的求解框架中，形成了一种混合 CEC 求解策略，以适应数据通路电路中存在大量乘法器和加法器单元的问题。实验表明，该混合方法成功

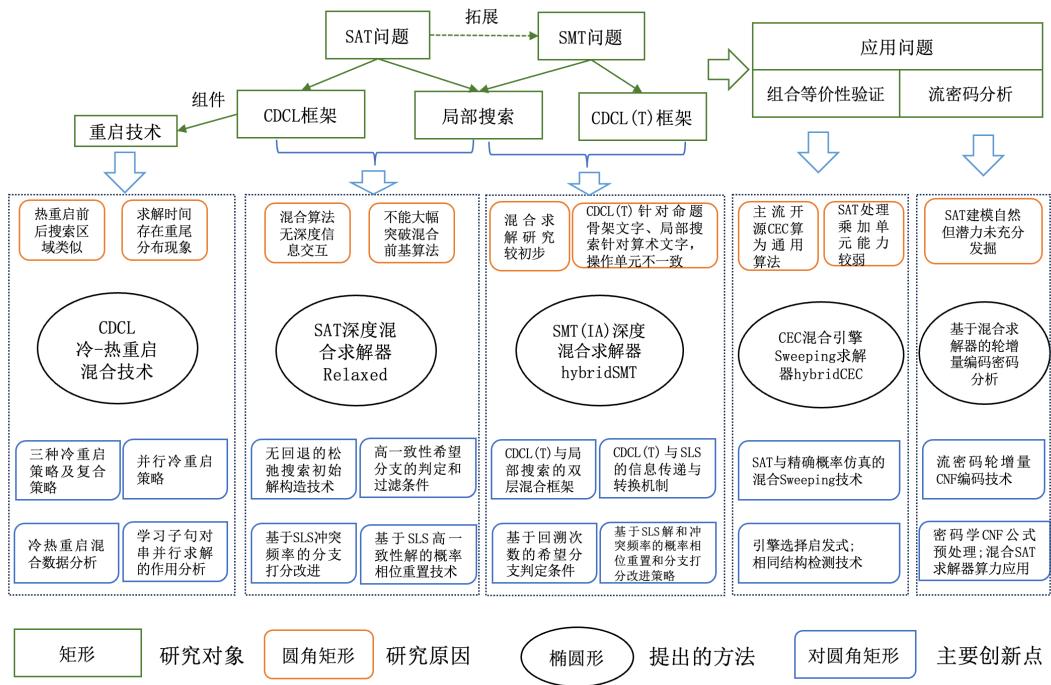


图 1.1 主要研究内容与贡献

Figure 1.1 Main research content and contributions

结合了 SAT 与 EPS 各自的优势，相较于主流方法 ABC，可以在 12% 的样例上提高三个数量级的求解效率。密码学方面，本文选择“环上前馈序列密码”作为研究案例，设计了一套“轮增量编码”的流密码分析算法，本文的方法可以在单核数小时内完成初态恢复的任务，而传统的方法需要在超级计算机上采用数周的时间完成密码分析。凭借该成果，该工作取得了“强网杯”密码数学专项赛的全国冠军。

图1.1对本文的研究问题、研究出发点（现状与存在的问题）、核心解决方法和对应的主要创新点进行了总结。

### 1.3 行文结构

如图1.2所示，本文共包含七个章节，除本章外，剩余章节的内容为：

第 2 章简要介绍了本文涉及的预备知识，主要包括 SAT 问题与 SMT 问题求解的基础概念与求解算法。

第 3 章介绍了 SAT 问题的冷-热混合重启求解策略，以及相关的实验规律。

第 4 章介绍了 SAT 问题的 CDCL 与局部搜索算法的深度混合求解策略。

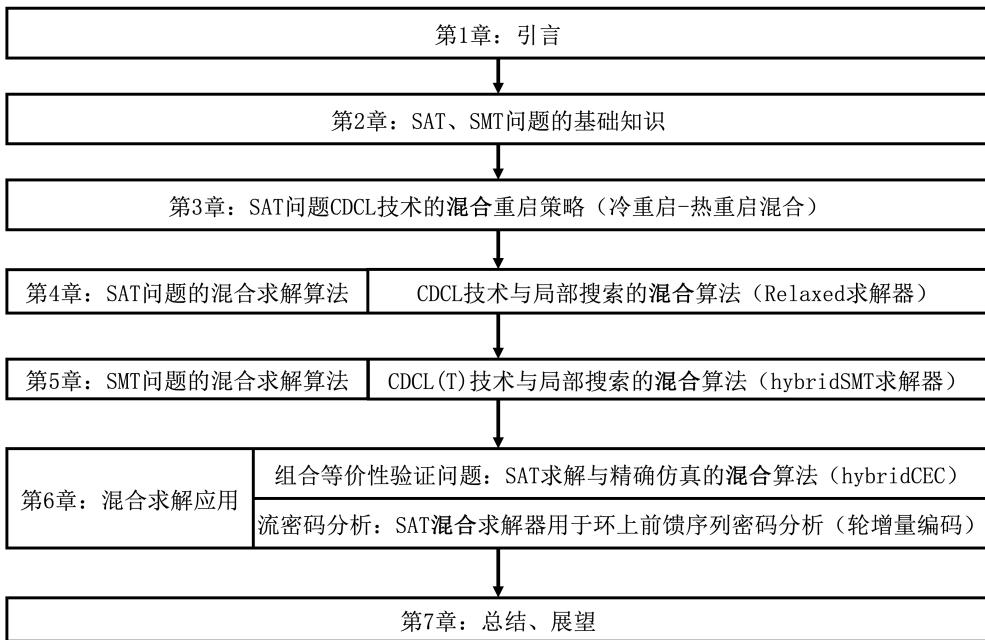


图 1.2 文章章节内容组织结构

Figure 1.2 Organizational structure of thesis chapters

第 5 章介绍了 SMT 问题的 CDCL(T) 与局部搜索算法的深度混合求解策略。

第 6 章介绍了 SAT 问题在组合等价性验证的混合求解技术和 SAT 混合求解器在密码分析中的案例分析。

第 7 章对本文的研究工作进行了总结，并展望了未来的研究方向。

## 第2章 SAT、SMT问题背景知识与相关工作

本章主要介绍可满足性问题与可满足性模理论问题的相关概念，并概括性的介绍了当前的主流求解算法和技术，包括可满足性问题的完备算法和非完备算法，非线性整数算术背景理论下的可满足性模理论的主流求解算法。

### 2.1 SAT与SMT问题基本概念

可满足性问题（Propositional Satisfiability Problem，简称SAT）聚焦于最简单的一种形式逻辑系统—命题逻辑。SAT问题为命题逻辑公式上的判定问题。下面将通过给出命题逻辑中有关于命题变元与逻辑联结词的定义。

**定义2.1(命题变元).** 命题为可以判断真( $\top$ )、假( $\perp$ )的陈述句，为命题逻辑的基本研究对象。命题可形式化地表示为**命题变元** (variable)，一般记作 $p, q, v$ 等符号。命题变元可唯一赋值为 $\{\top, \perp\}$ 中的某真值 (truth value)，且真、假真值也可以对应的表示为布尔值1、0。

**定义2.2(命题逻辑联结词).** 逻辑联结词可以将一组命题变元按照一定规则组合成命题公式。一元联结词只有‘非’( $\neg$ )一种；常见的二元联结词有合取( $\wedge$ )、析取( $\vee$ )、蕴含( $\rightarrow$ )、等价( $\leftrightarrow$ )等。

根据命题逻辑的基本符号，我们可以进一步给出有关逻辑公式中文字、子句和合取范式的定义。

**定义2.3(命题文字).** 命题文字 (literal) 由命题变元与一元联结词‘非’组成，一般记为符号 $l$ 。对于任意命题变元 $p$ ，本文称其本身( $p$ )为**正文字**，称其非( $\neg p$ ，或简记为 $\bar{p}$ )为**负文字**。本文用 $var(l)$ 表示文字 $l$ 对应的变元，正(负)文字 $l$ 满足当且仅当 $var(l)$ 赋值为 $\top$ ( $\perp$ )。

**定义2.4(子句).** 子句 (clause) 为一组文字的析取 (形如 $l_1 \vee l_2 \vee \dots \vee l_k$ ，或简记为文字集合 $\{l_1, \dots, l_k\}$ )。子句满足当且仅当该子句中至少一个文字被满足。**空子句**为不包含任何文字的子句，不可被满足，一般用于表示冲突，记作符号 $\square$ 。**单元子句**为只包含一个文字的子句，若使其满足则必令其中文字满足。

**定义 2.5(合取范式).** 合取范式 (Conjunctive Normal Form, 简称 CNF) 为子句的合取 (形如  $c_1 \wedge c_2 \dots \wedge c_m$ )，一般记作  $\phi, \psi, \varphi, F$  等符号。CNF 满足当且仅当所有其子句都满足。

通过 Tseitin 编码，我们可以在引入线性数目的新变元的代价下，将任意给定的命题逻辑公式均在多项式时间内转化为可满足性等价的 CNF 的形式<sup>[60]</sup>。CNF 公式为大部分命题逻辑求解器的标准输入形式。

为了更好的描述公式的可满足性，本文给出了赋值的定义。

**定义 2.6(命题公式的赋值).** 令  $V = \{p_1, p_2, \dots, p_n\}$  为由  $n$  个命题变元构成的集合，若  $F$  为构建于  $V$  上的命题公式，映射  $\alpha : V \rightarrow \{\top, \perp\}$  被称为一组**赋值** (truth assignment) 或真值指派，并用  $\alpha(p)$  表示变元  $p$  的赋值。如果  $\alpha$  将所有的  $n$  个命题变元赋值，则  $\alpha$  为**完全赋值**；否则  $\alpha$  为**部分赋值**。未赋值的变元被称为**自由变元**，对应的文字为**自由文字**。若子句  $c$  中无已满足文字且存在自由文字，则  $c$  为**自由子句**。

**定义 2.7(逻辑公式的解与可满足性).** 给定一个逻辑公式  $\varphi$  (也可以为后续 SMT 公式)，若存在一组赋值  $\alpha$  可使  $\varphi$  满足，则  $\varphi$  是**可满足的** (satisfiable)，且  $\alpha$  为  $\varphi$  的**解** (model, 或译为模型)。若不存在一组赋值使得  $\varphi$  满足，则  $\varphi$  是**不可满足** (unsatisfiable) 的。若逻辑公式  $\varphi$  在任意真值指派下均满足，则  $\varphi$  为**重言式** (tautology)，或**可靠的** (valid)。

**例 2.1.** 给定 CNF 公式  $\varphi = c_1 \wedge c_2 \wedge c_3, c_1 = p_1 \vee p_2, c_2 = \neg p_1 \vee \neg p_3, c_3 = \neg p_2 \vee \neg p_3$ 。  
 $\alpha_0 = \{p_1 \rightarrow \top, p_2 \rightarrow \perp\}$  是一个部分赋值，此时  $c_2$  为自由子句。在完全赋值  $\alpha_1 = \{p_1 \rightarrow \top, p_2 \rightarrow \perp, p_3 \rightarrow \top\}$  下， $c_2$  是不满足的，因此  $\varphi$  无法被  $\alpha_1$  满足。在完全赋值  $\alpha_2 = \{p_1 \rightarrow \top, p_2 \rightarrow \perp, p_3 \rightarrow \perp\}$  下，所有子句已满足，故  $\varphi$  满足，即  $\alpha_2$  是  $\varphi$  的一个解。

**可满足性模理论** (Satisfiability Modulo Theory, 简称 SMT) 问题在命题逻辑可满足性问题的基础上拓展了特定的背景理论 (background theories)，属于一阶逻辑 (First-Order Theory) 的范畴。相比 SAT 公式，SMT 公式有更强的表达能力。SMT 问题为 SMT 公式上的判定问题，常见的背景理论如表 2.1 所示<sup>[61,62]</sup>。

本文聚焦于非线性整数算术 (Non-linear Integer Arithmetic, 简称 NIA) 理论

表 2.1 常见的 SMT 问题背景理论

Table 2.1 Typical SMT background theories.

背景理论	示例公式
无理论 (命题逻辑)	$(p_1 \vee p_2) \wedge (\neg p_1 \vee \neg q)$
量词布尔公式 (QBF)	$\forall p. \exists q. \forall x. p \rightarrow (q \wedge x)$
未解释函数 (UF)	$g(f(x)) = 1 \vee (h(y) \wedge f(y))$
等式 (Equality)	$x = y \vee \neg(y = z) \rightarrow \neg(x = z)$
整数、实数差分 (IDL、RDL)	$(x - y > 3) \wedge (x - z \leq 4)$
整数、实数线性算术 (LIA、LRA)	$(2x - 3y > 3) \wedge (6x + 5z \leq 0)$
整数、实数非线性算术 (NIA、LRA)	$(2xyz - y^2 > 3) \vee (x^2y^3 - yz \leq 4)$
位向量 (Bit Vector)	$a > (a \& (b >> c))$
数组 (Array)	$(i = j \vee a[i] = 9) \rightarrow a[j] = 9$
指针逻辑 (Pointer logic)	$(p = q \vee *q = 3) \rightarrow *p = 3$
组合理论 (Combined theories)	$a[i] \geq 9 \wedge (f(x) \vee x^2y = 0) \rightarrow (i << x) \leq 4$

下的 SMT 问题，简称 SMT(NIA) 问题；且如无特殊说明，本文探讨的 SMT 问题均无量词。

SMT(NIA) 公式中的文字为 NIA 公式，为此，本文描述了与 NIA 公式中与多项式、NIA 操作符、NIA 公式相关的定义。

**定义 2.8(单项式).** 在给定的算术变元集合  $X = \{x_1, x_2, \dots, x_n\}$  下，**单项式** (monomial) 为形如  $x_1^{d_1} x_2^{d_2} \dots x_i^{d_i}$  的表达式，其中  $x_1, \dots, x_i$  互不相同， $d_1, \dots, d_n$  为正整数，表示对应算术变元的度 (degree)。单项式的度为其中所有变元度之和。我们称单项式为线性 (linear) 的当且仅当其度为 1。

**定义 2.9(多项式).** **多项式** (polynomial) 为形如  $a_1m_1 + a_2m_2 + \dots + a_jm_j$  形式的表达式，其中  $m_1, \dots, m_j$  为不同的单项式， $a_1, \dots, a_j$  为单项式对应的常数系数，多项式的度为其中最大的单项式度数。多项式是线性的，当且仅当其度为 1。

**定义 2.10(非线性整数算术约束).** 算术 (A) 约束是形如  $\sigma : \sum_i a_i m_i \sim k$  的表达式，其中左侧为一个多项式，右侧  $k$  为常数，操作符  $\sim \in \{>, <, =, \neq, \leq, \geq\}$ 。若多项式  $P$  中涉及的的算术变元值域均为整数，则  $\sigma$  为整数 (I) 算术约束；若存在值域为实数的算术变元，则  $\sigma$  为实数 (R) 算术约束。若  $P$  为线性多项式，则  $\sigma$  为线性 (L) 算术约束；若  $P$  非线性，则  $\sigma$  为非线性 (N) 算术约束。**线性整数算术** (LIA) 约束只允许存在线性算术约束且要求为整数算术约束。**非线性整数算术** (NIA) 约束允许存在非线性算术约束且要求为整数算术约束。

借助命题联结词  $\vee$ ,  $\wedge$  和  $\neg$ , NIA 算术约束均可以转化为  $\sigma' : \sum_i a_i m_i \leq k$  的形式, 我们称该不等式为 NIA 原子公式 (atom), 具体的转化方法为:

- $\sum_i a_i m_i < k$  可表示为  $\sum_i a_i m_i \leq k - 1$ ;
- $\sum_i a_i m_i > k$  可表示为  $\neg(\sum_i a_i m_i \leq k)$ ;
- $\sum_i a_i m_i \geq k$  可表示为  $\neg(\sum_i a_i m_i \leq k - 1)$ ;
- $\sum_i a_i m_i = k$  可表示为  $(\sum_i a_i m_i \leq k) \wedge \neg(\sum_i a_i m_i \leq k - 1)$ ;
- $\sum_i a_i m_i \neq k$  可表示为  $(\sum_i a_i m_i \leq k - 1) \vee \neg(\sum_i a_i m_i \leq k)$ 。

**定义 2.11 (SMT(NIA) 公式).** 给定一组命题变元  $V = \{p_1, p_2, \dots, p_n\}$  和一组算术变元  $X = \{x_1, x_2, \dots, x_m\}$ , 算术文字为 NIA 原子公式  $\sum_i a_i m_i \leq k$  或其非  $\neg(\sum_i a_i m_i \leq k)$  的形式。类似于 SAT 问题, SMT(NIA) 公式中的文字允许出现算术文字, 且所有的 SMT(NIA) 公式也有对应的 CNF 表示。

类似地, 我们也可以给出 SMT 问题上有关于赋值和解的定义。

**定义 2.12. (SMT(NIA) 公式的赋值与解)** 给定一组命题变元  $V = \{p_1, p_2, \dots, p_n\}$  和一组整数域 ( $\mathbb{Z}$ ) 值域上的算术变元  $X = \{x_1, x_2, \dots, x_m\}$ , 若  $F$  为建立在  $V \cup X$  上的 SMT(NIA) 公式。映射  $\alpha : V \rightarrow \{\top, \perp\}, X \rightarrow \mathbb{Z}$  为  $F$  的赋值。其它有关赋值和解的定义均与**定义 2.6**保持一致。

需要注意的是: LIA 可以看做是 NIA 的特例, 为了加以区分, 本文中 NIA 特指不包含 LIA 约束的整数算术 (IA)。前文的介绍中为了概念介绍的方便, IA 多以 NIA 出现。后续的内容中, 如若出现了 IA 的段落, 均对 IA 和 NIA 的概念进行严格的区分。

本文聚焦于 SAT 公式和 SMT(IA) 公式下的判定问题, 给定公式下的**判定问题** (decision problem) 指的是判定一个公式可靠性 (或可满足性) 的问题。本文中**求解器** (solver) 指的是用于求解判定问题的程序。此外, 我们称用于研制算法和对比而选取的基准求解器为**基求解器** (base solver)。

目前, 学者们在 SAT、SMT 问题求解复杂度分析上取得了众多的理论成果<sup>[63–67]</sup>, 但是目前主流的 SAT、SMT 求解器包含了来自算法设计、组件协作、求解实例优化、参数优化等多个维度的技术, 难以抽取出简洁的算法模型。而且, 实际应用中, 求解器的求解效果一般远好于复杂度分析的理论下界, 因此目前对于主流的 SAT、SMT 求解器效果的评估多建立在对实验结果的直接分析上。

**定义 2.13 (完备性与可判定性).** 当求解器返回可靠的（或者可满足的）时，输入的公式为可靠的（或者可满足的），我们称求解器是**安全的**（sound）。当求解器能终止，且当输入公式为可靠的（或者可满足的）时求解器的输出为可靠的（或者可满足的），我们称求解器是**完备的**（complete）。我们称某背景理论  $T$  下的求解器为一个**决策过程**（decision procedure），当求解器对于  $T$  下的任意公式都是安全、完备的。若  $T$  为**可判定的**（decidable）当且仅当存在  $T$  下的决策过程。

求解 SAT 问题和 SMT 公式等逻辑公式的判定问题的方法归根结底为形式推理方法<sup>[62]</sup>，主要依赖“推理”与“搜索”两类技术，且两类技术交互揉合在一起，缺一不可。一方面，推理技术是一种基于证明论（proof theoretic）的方法，主要依赖演绎推理（deductive reasoning）的机制，用逻辑系统中的公理（axiom）和推理规则（inference rule），由一系列前因推出一系列结论。我们则可以从结论中拿到一些引理用于进一步推理或得到矛盾来反驳之前的假设。另一方面，搜索技术则更依赖于如何高效的遍历整个搜索空间，或者依赖启发式（直觉主义）在有限的资源内在部分空间内搜索目标，典型的代表分别为模型论（model theoretic）中的真值表枚举技术和随机局部搜索技术（stochastic local search）。

SMT 求解器可以看作为 SAT 求解器的拓展，SMT 求解器中为了更好的利用 SAT 求解器，引入了**布尔骨架**（Boolean Skeleton）的概念。该操作可以将公式中的命题逻辑推理独立出来，并采用 SAT 的 CDCL 技术进行推理。

**定义 2.14 (SMT 公式的布尔骨架).** 如果只考虑原子公式的满足性，我们可以将 SMT 公式中算术文字用对应的抽象布尔文字表示，如果将 SMT 公式中所有的算术文字均用抽象布尔文字表示，得到的新公式被称作**布尔骨架**。

**例 2.2.** 给定一组算术变元  $X = \{x_1, x_2, x_3, x_4\}$  和命题变元  $P = \{p_1, p_2\}$ ,  $F_{SMT(NIA)}$  是一个 SMT(NIA) 公式。通过为三个原子公式 ( $3x_1x_2 \leq 2$ ,  $-x_2 - 3x_4 \leq 0$  和  $2x_3^3x_4 + 4x_2 + x_1 \leq 8$ ) 对应地引入三个新的抽象布尔变元 ( $p_{\sigma_1}$ ,  $p_{\sigma_2}$  和  $p_{\sigma_3}$ )，我们可以得到  $F_{SMT(NIA)}$  的布尔骨架  $S_F$ 。

$$\begin{aligned} F_{SMT(NIA)} &= (p_1 \vee \neg p_2) \\ &\wedge (\neg(3x_1x_2 \leq 2) \vee (-x_2 - 3x_4 \leq 0)) \\ &\wedge (p_2 \vee (2x_3^3x_4 + 4x_2 + x_1 \leq 8)) \end{aligned} \tag{2.1}$$

$$S_F = (p_1 \vee \neg p_2) \wedge (\neg p_{\sigma_1} \vee p_{\sigma_2}) \wedge (p_2 \vee p_{\sigma_3}) \quad (2.2)$$

## 2.2 SAT 问题求解算法

按照算法是否具有完备性，SAT 求解算法可以分为完备算法与非完备算法，本节中对两种技术进行了描述：2.2.1 节介绍了以冲突驱动的子句学习技术为主体的完备算法；2.2.2 节介绍了以随机局部搜索为主体的非完备算法。当今串行求解能力提升速度已难以追赶工业和学术界对 SAT 求解器性能的需求增长，并行求解可以借助多核心资源优势解决更复杂、更困难的应用问题。作为第 3 章内容的基础知识补充，2.2.3 节补充了 SAT 问题并行求解相关的技术背景。

### 2.2.1 SAT 问题完备算法

最主流的 SAT 完备算法为冲突驱动的子句学习技术（Conflict-Driven Clause Learning，简称 CDCL），CDCL 中集成了推理和搜索两类技术，并可以抽象为一个包含了多种推理技术的深度优先搜索算法。

#### 2.2.1.1 布尔约束传播与归结原理

以 CDCL 框架为主体的 SAT 求解器基本都包含了两种重要的推理技术。第一个是布尔约束传播（Boolean Constraint Propagation，简称 BCP），又称单元传播（Unit Propagation）；第二个是归结原理（Resolution）。在 CDCL 求解器中，BCP 是推导变元赋值的最重要手段，占据了求解的绝大部分时间，但是可以大幅减少不必要的搜索代价；归结原理最主要的用途为参与学习子句的生成和公式化简中的变元消除。

**1. 布尔约束传播** 单元子句可以唯一确定其中变元的赋值，BCP 依赖这个规则不断的寻找求解过程中的单元子句，并赋值新的变元，或者推出一个矛盾。具体地，BCP 会维护一个文字的序列  $Q$ ，存放当前已经识别出的单元子句中的文字。若  $Q$  不为空，BCP 会迭代地从中取出一个文字  $l$ ，并会访问公式中所有包含  $var(l)$  变元的子句  $c$ 。如果  $c$  中所有文字均不满足，则 BCP 推出一个矛盾，该子句被称为冲突子句；如果  $c$  中除了一个自由文字之外其他的文字均不满足，则找到了一个新的单元子句，那么 BCP 会将该文字放入序列  $Q$  中。实际实现中，

为了高效地查询变元所在的子句和减少重复访问，求解器一般会采用两文字监督（2-watch-literals）技术，它采用了一种 lazy 的数据结构<sup>[42]</sup>。

**2. 归结原理** 归结原理可以利用两条包含相反文字的子句产生一条不包含该两种文字的新子句。若给定的 CNF  $F$  中存在两条子句， $c_1$  和  $c_2$ ，其中包含关于变元  $v$  的两个相反文字，记  $c_1 = \{v, p_1, p_2, \dots, p_i\}$ ,  $c_2 = \{\neg v, q_1, q_2, \dots, q_j\}$ 。根据归结原理， $c_1$  和  $c_2$  可以推出一条新的子句  $c_r = \{p_1, p_2, \dots, p_i, q_1, q_2, \dots, q_j\}$ ，记为  $c_r = c_1 \otimes_v c_2$ 。 $c_r$  可满足当且仅当  $c_1$  和  $c_2$  可满足，故  $c_r$  该子句可以直接加入到  $F$  中且不影响  $F$  的可满足性。

该规则可以拓展到集合上，若  $F$  中有一组包含文字  $v$  的子句集合  $S_v = \{c_1, c_2, \dots, c_i\}$ ，且有一组包含文字  $\neg v$  的子句集合  $S_{\neg v} = \{c'_1, c'_2, \dots, c'_j\}$ 。根据归结原理，我们可以生成  $i \times j$  条新的子句，构成子句集合  $S_r = S_v \otimes_v S_{\neg v} = \{c_1 \otimes_v c'_1, c_1 \otimes_v c'_2, \dots, c_i \otimes_v c'_j\}$ ，若在  $F$  上添加子句集合  $S_r$  不会影响  $F$  的可满足性。此外，若  $F$  中除了  $S_v$  和  $S_{\neg v}$  之外不包含任何其它包含变元  $v$  的子句，则可以用  $S_r$  替换掉  $F$  中的  $S_v$  和  $S_{\neg v}$  从而减少  $F$  中的一个变元，该过程能保证变化前后公式是可满足性等价的。

### 2.2.1.2 蕴含图、子句学习与回退

在搜索的过程中，启发式选择并赋值的变元以及通过推理赋值的变元之间的推理关系会通过蕴含图（implication graph）维护。CDCL 中一个启发式赋值的变元被称作决策变元，决策变元与该变元赋值之后新推出的变元处于同一决策层（decision level），决策层的标号为启发式选择的变元数目，即从原始 CNF 公式中能直接推出的变元均属于 0 决策层。蕴含图中的点表示现存变元的赋值情况，并用“文字 @ 决策层”的形式表示；图中的边表示变元之间的推理关系，并在边上标注了推理关系产生时使用的子句标号。通过蕴含图，我们可以清晰的得到所有参与到当前冲突的变元与子句。图2.1给出了一个蕴含图的示例，表示在第 4 层决策层的推理过程中遇到了冲突，且冲突发生的原因是因为子句  $c_6 = \{v_5, \neg v_6\}$  在当前的赋值下为空子句。

当传播遇到冲突时，则需要从中学习到一些新的知识，以保证后续的搜索过程不会再次步入同一搜索场景，即对搜索空间进行剪枝。学习到的知识一般采用学习子句（learnt clause）的形式保存，一般来说，子句长度越短约束越强，剪枝

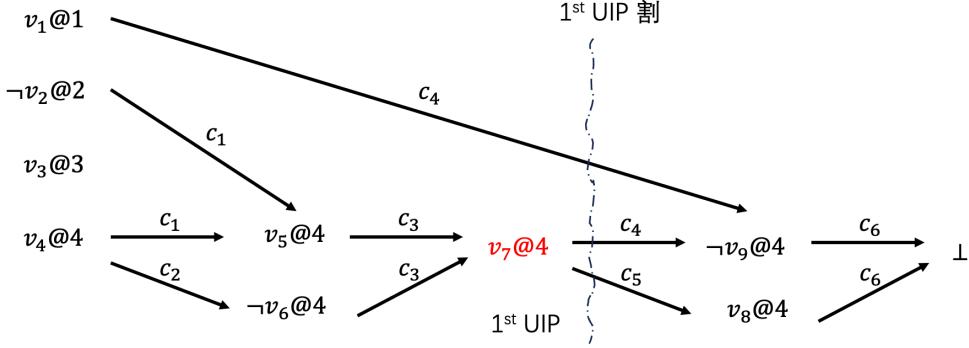


图 2.1 蕴含图与 First UIP 示例

Figure 2.1 An Example of implication graph and First UIP

力度越大。为了更好的评估学习子句的质量，Audemard 等人于 2009 年提出了利用学习子句中涉及到的决策层数目来衡量其质量，并命名为文字块距离（Literal Block Distance，简称 LBD），该评估标准也被后续的 SAT 求解器普遍采用。<sup>[68]</sup>

为了从冲突中得到一个高质量的学习子句，目前主流的求解器会维护一个冲突子句 *learnt*，并初始化为遇到的空子句。随后算法会沿变元的赋值的逆序，依照出现在 *learnt* 中的变元，将 *learnt* 与蕴含图上对应的子句进行逐次归结，直到遇到一个蕴含图上 First UIP。此时，算法会得到一个较为理想的学习子句。蕴含图上的唯一蕴含点（Unique Implication Point，简称 UIP）指的是冲突层中从决策变元到冲突的所有图上路径均经过的点，且称与冲突距离最近的一个点为 First-UIP。图2.1中  $v_4 @ 4$  与  $v_7 @ 4$  对应的两个点均为 UIP， $v_7 @ 4$  为 First-UIP。UIP 在蕴含图上有很好的性质。在当前的决策层下，若保证 UIP 的赋值不变，则必会产生一个相同的冲突。

此外，我们可以从 UIP 处构造一条割，使得靠近决策变元的一侧只有一个冲突层的 UIP。此时该割的左侧构成的文字的非组成的子句即为归结之后产生的学习子句。如图2.1和图2.2a所示，该 First-UIP 割产生的学习子句为 *leanrt* =  $\{\neg v_1, \neg v_7\}$ ，长度为 2，LBD 为 2。不同的 UIP 产生的学习子句不一致，且质量也不一致。如图2.2b所示，该图的 Last-UIP 产生的学习子句为 *leanrt* =  $\{\neg v_1, v_2, \neg v_4\}$ ，长度为 3，LBD 为 3。最初的 CDCL 求解器会在多个 UIP 处均产生一条学习子句，但是现代的 CDCL 求解器一般只会在 First-UIP 处产生一条学习子句并退出归结的过程。

当 CDCL 推理过程中遇到冲突时，需要抹去一部分变元的赋值，来帮助推

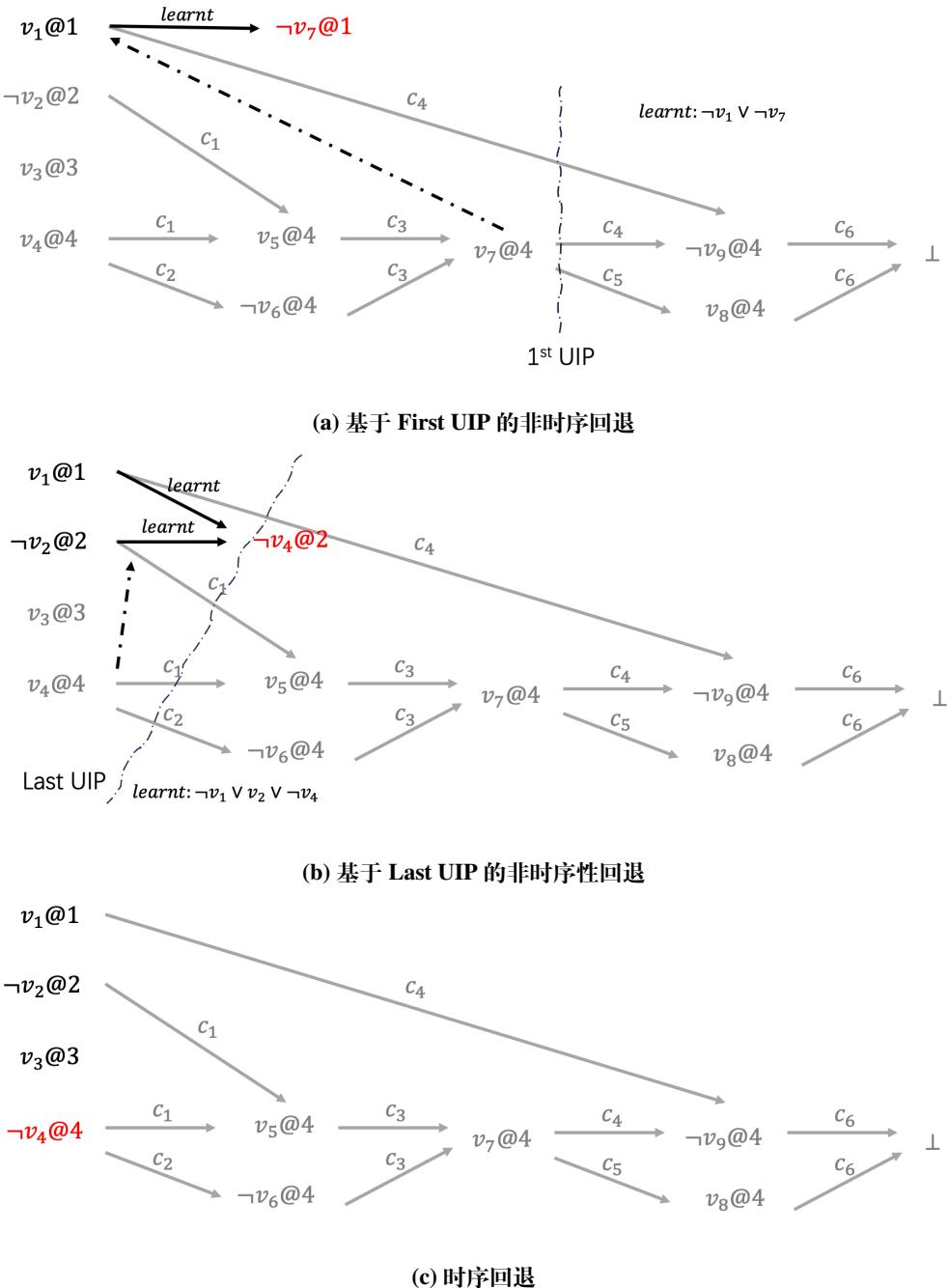


图 2.2 不同的回退方法示例。

Figure 2.2 Examples for different backtrack methods.

理继续执行。一般来说冲突的回退分为时序回退（chronological backtracking）和非时序回退（non-chronological backtracking）。非时序回退指的是跨多层回退，一般是回退到冲突分析的到的学习子句中的第二高层。因为 CDCL 的学习子句的产生是依赖 UIP 的，其中只有一个最高层的变元，故回退之后，得到的学习子句必为一个单元子句，可以将 UIP 变元取反赋值。图2.2a和图2.2b分别给出了依赖 First-UIP 和 Last-UIP 的非时序回退方案。时序回退指的是回溯到最近的一次决策的位置将其取反，或者回到上层决策层，不会跨多层回退。图2.2c展示了一个时序回退的样例。

### 2.2.1.3 CDCL 框架

图2.3展示了 CDCL SAT 求解器中核心组成成分之间的交互逻辑。其中最核心的部件有四个：单元传播组件负责执行布尔逻辑的推理；冲突分析组件负责从冲突中积累知识用于剪枝；分支组件负责控制搜索的顺序；回溯组件负责控制回跳位置。

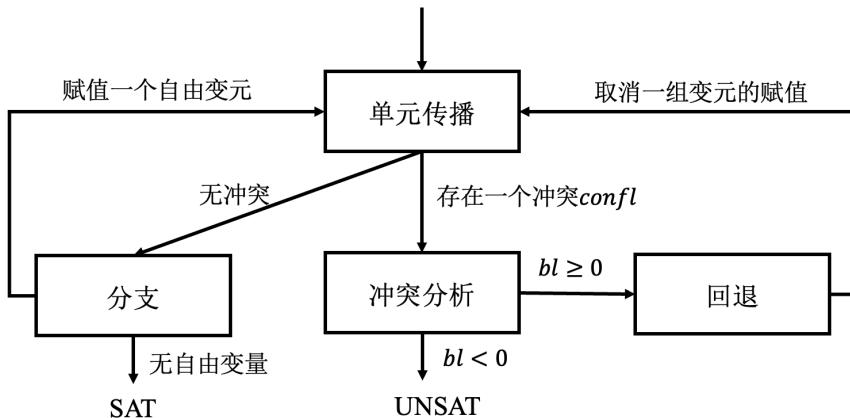


图 2.3 CDCL 框架核心组件间的交互逻辑概览

Figure 2.3 An overview of the interaction logic of core components in the CDCL framework

算法1描述了 CDCL 的大致工作逻辑：它会在顶层根据当前的变元赋值情况执行 BCP 并观察当前的赋值是否会推出矛盾（第 2 行）。如果 BCP 推出矛盾，算法则会根据产生矛盾的上下文信息推出一条新的学习子句，并用于后续的搜索空间剪枝，以避免步入同样的矛盾路径（第 4 行）。冲突分析的过程中也会得到一个回退层，并随后撤销一系列变元的赋值（第 6 行）。如果算法发现需要回退到 0 层以下，则冲突是发生在第零决策层，即原始公式能在没有任何赋值的前提下

---

**Algorithm 1:** CDCL 算法框架伪代码

---

**输入:**一个CNF公式  
**输出:**求解结果‘SAT’或者‘UNSAT’

```

1 while true do
2   conflict  $\leftarrow$  propagate();
3   if c  $\neq \emptyset$  then
4     bl  $\leftarrow$  analyseConflict(c);
5     if bl < 0 then return ‘UNSAT’;
6     backtrack(bl);
7   else
8     if !decide() then return ‘SAT’;

```

---

下推出矛盾，则返回‘UNSAT’（第5行）。如果BCP无法推出矛盾，且所有变元均已经得到赋值，decide函数返回分支失败，因此算法找到一组解，返回‘SAT’；否则，CDCL会根据分支启发式选择一个自由变元赋值，并继续执行BCP的操作（第8行）。分支启发式主要设计变元排序启发式（根据上下文，有时分支启发式默认为变元排序启发式）和变元赋值倾向（相位）启发式。

#### 2.2.1.4 变元分支启发式

分支启发式的质量直接决定了CDCL求解器的性能。因为对于可满足问题，若存在一个理想的启发式可以精准的选择变元的赋值，则可使算法在不超过变元数次的分支决策下结束搜索；对于不可满足问题，分支启发式影响了子句归结的顺序，进而影响了矛盾的推出效率。

早期的DPLL框架的SAT求解器多基于子句的原始信息，例如MOMS（Maximum number of Occurrences in clauses of Minimum Size）会优先选择在最短子句中出现频率最高的文字<sup>[69]</sup>。目前主流的SAT求解器变元分支启发式多为动态策略，且均遵守同一个原则：利用最近累计的知识，尽可能选择一个能高概率推出冲突的变元<sup>[70]</sup>。一种基于该原则的变元前移（Variable Move To Front，简称VMTF）贪心策略<sup>[71]</sup>在不可满足样例上展示出了不错的效果。目前应用最广的一个分支策略为变元状态独立衰退和策略（Variable State Independent Decaying Sum，简称VSIDS）<sup>[42]</sup>及其各种变种策略<sup>[44,70,72,73]</sup>。另一类有效的策略为基于强化学习的变元分支策略，代表性的技术为学习率分支策略（Learning Rate Branching，简称

LRB) [74]。

目前主流的分支启发式可以进一步拆分为变元分支启发式和相位选择启发式，分别负责分支决策时变元的选取顺序和变元选取后的赋值选择。早期的分支启发式同时负责两者<sup>[42]</sup>，但是随着任务的细分，如果没有其它的说明，分支启发式特指细化后的变元分支启发式。常见的相位选择启发式有相位保留<sup>[75]</sup>和基于靶相位的相位重置技术<sup>[50]</sup>。

**变元状态独立衰退和策略（VSIDS）** VSIDS 是最早提出的一类动态变元打分策略，也是目前使用频率最高的分支策略。该启发式最早使用于 Chaff 求解器<sup>[42]</sup>，它为每一个文字维护了一个打分，并根据冲突对该打分进行修正。BerkMin 的启发式在该算法的基础上进行了改进，然后变元的选择与变元赋值拆分为两部分<sup>[76]</sup>。

主流的几类求解器，如 MiniSat<sup>[44]</sup>、glucose<sup>[68]</sup>、lingeling<sup>[77]</sup>、CryptoMiniSat<sup>[52]</sup>、CaDiCaL<sup>[78]</sup>、kissat<sup>[50]</sup> 中使用的 VSIDS 算法均沿用了 BerkMin 的算法形式，并可以看作其改进版。kissat 求解器中也实现了 VSIDS 的另一种变种版本 ACIDS，利用变量的平均冲突标号进行打分<sup>[70]</sup>。多伦多大学的 Kuldeep 团队 DurianSat 求解器中使用的 LSIDS 方法则恢复了 Chaff 中选择文字的思路，并发现在密码学实例上有更好的表现<sup>[72]</sup>。

典型的 VSIDS 有如下的几个特点：

- 算法为每个变元都维护了一个活跃度打分 activity，初始时为 0。当求解器遇到冲突，且变元  $v$  参与到了该冲突中时，该变元的活跃度分数  $\text{activity}[v]$  会增加一个 var\_inc 打分，该操作被称作提高（bump）。
- 近期的变元冲突信息更重要。var\_inc 打分初始时为 1，每次算法遇到冲突时，var\_inc 则会更新为  $\text{var\_inc}/\text{var\_decay}$ ，其中 var\_decay 是一个处于  $(0, 1)$  区间内的浮点数，即该值会不断增大。例如，var\_decay 在 MiniSat 和 kissat 中取值为 0.95；在 glucose 中取值为 0.8，并逐渐增加至 0.95。
- 如果 var\_inc 值过大，有越界的风脸，算法则会把所有变元的 activity 分数和 var\_inc 同时除以一个较大的数值，该操作不会影响整体的打分逻辑和变元排序。
- 当 CDCL 选择分支变元时，会优先选择活跃度打分最高的自由变元。

**学习率分支启发式 (LRB)** LRB<sup>[74]</sup> 是冲突历史分支启发式 (CHB)<sup>[79]</sup> 的改进版本。该类方法将 SAT 问题的变元分支建模到了强化学习领域的多臂老虎机 (Multi-Armed Bandit, 简称 MAB) 问题上。

变元  $v$  的冲突率 (learning rate)  $LR(v)$  指的是变元  $v$  赋值到取消赋值期间参与到冲突中的比率。如果我们用  $I$  表示变元  $v$  从赋值到取消赋值的时间片段，其长度为  $L(I)$  次冲突，并用  $P(v, I)$  表示变元在时间片  $I$  内遇到的冲突次数，则  $LR(v)$  可以定义为  $\frac{P(v, I)}{L(I)}$ 。

LRB 的核心思想为 CDCL 每一个变元的赋值即为 MAB 问题中的一次操作 (action) 的执行，当变元赋值之后就有可能参与到冲突分析中，当其被取消赋值时则可以来计算其学习率，并当作执行该操作的奖励 (reward)。为了变元分支建模到 MAB 问题上，CHB/LRB 采用了指数衰退加权平均 (Exponential Recency Weighted Average, 简称 ERWA) 策略。

ERWA 为每一个变元  $v$  维护了一个自己的时间序列 (time series)  $ts_v$ ，包含了变元的赋值历史和对应的奖励取值，并通过计算该序列的指数衰退平均值 (Exponential Moving average, 简称 EMA) 来计算  $ts_v$  的打分  $ema_\alpha(ts_v)$ ，该计算方式可以充分考虑变元赋值打分的时效性。在 CDCL 选择分支变元时，该算法会优先选择变元  $v^* = argmax_{v \in U}(ema_\alpha(ts_v))$ ，其中  $U$  表示自由变元的集合。

LRB 为每一个变元维护了一个打分函数  $Q_v$ ，初始值为 0，每次  $ts_v$  更新时，会执行  $Q_v \leftarrow (1-\alpha) \cdot Q_v + \alpha \cdot r$ ，其中  $r$  为本次的  $LR(v)$ 。LRB 采用了典型的 ERWA 技术，即初始时  $\alpha = 0.4$ ，每一步操作中会下降 0.06，直到  $\alpha = 0.06$  为止。CHB 与 LRB 不同的点在于 CHB 的奖励值  $r = \frac{multiplier}{L(I)}$ ，其中  $multiplier$  为常数 1 或者 0.9。LRB 的作者也提出了一种 LRB 的改进版本<sup>[74]</sup>，奖励值  $r = RL(v) + RSR(v)$  中添加了推理侧比率 (reason side rate)，并定义  $RSR(v) = \frac{A(v, I)}{L(I)}$ ，其中  $A(V, I)$  表示变元  $v$  在区间内出现在冲突的原因子句中的次数。

考虑到临近性，即不鼓励选择最近不活跃的变元，LRB 提出在冲突分析之后，将未赋值的变元的打分通过  $Q_v \leftarrow 0.95 \cdot Q_v$  的方式降低。

**变元前移启发式 (VMTF)** 求解器的变元选择算法可以被看作是一种变元打分的在线更新算法，这个观点也建议求解器使用一种有效摊销复杂性的在线更新算法，例如最早的 move-to-front 策略<sup>[80]</sup>。Lawrence 的硕士论文中也提到了

类似的思想，提出了 Seige SAT 求解器和变元前移的变元分支方法<sup>[71]</sup>。具体地，VMTF 维护了一个变元的队列，当遇到冲突时会将学习子句中的变元移动到队首，并在选择变元的时候优先选择队首的变元。考虑到每次移动操作平均会移动上百个变元，因此 Seige 求解器会将学习子句中的最多 8 个变元直接移动到队列的前面<sup>[71]</sup>。kissat 沿用了 VMTF 技术，优化了数据结构的设计，使得在线更新的摊销复杂度很低，因此采用了一种无限制的变元前移操作，即学习子句中所有变元均会保持在队列中的原有顺序，并直接移动到队首<sup>[50]</sup>。VMTF 有不错的求解效果，且这种简洁的算法在分析 CDCL 求解器复杂度上有一定优势<sup>[70]</sup>。

### 2.2.1.5 CDCL 核心启发式技术

CDCL 凭借其高性能和高拓展性被主流的求解器沿用，其中 BCP、子句学习技术和非时序性回退作为最主要的特征一直被沿用至今，其核心框架自提出以来并没有大幅的变动。在三十年的发展历程中，学者们提出了众多高效的组件、启发式策略和多种变种改进版本，并研究了策略间的组合，使得求解器的性能得到了大幅的提升<sup>[47]</sup>。最具有代表性的技术有分支启发式、重启、子句管理、公式化简等技术。其中分支启发式已经在上面的一节展开讨论，重启等技术将在第三章进一步讨论。

**2. 重启** 1997 年，Gomes 等人发现 SAT 问题等组合优化问题存在求解时间的重尾分布现象 (heavy-tailed distribution)，求解器随机种子、参数上的轻微扰动会使得求解器的求解时间发生剧烈的变化，因此作者提出定期的重启来减少该现象，减少求解时间，并提高求解时间的稳定性<sup>[81]</sup>。按照重启是否依赖求解过程中的信息，可以将重启分为静态策略和动态策略<sup>[82]</sup>。静态策略的代表为 Luby 等人于 1993 年提出的 Luby 重启策略<sup>[83]</sup>，并已经证明了在 Las Vegas 类算法上的最优性。动态策略的代表为 Glucose-Style 重启策略<sup>[84]</sup>，它将局部学习子句平均质量和全局学习子句的平均质量的比较作为重启的依据。学者们发现可满足问题和不可满足问题需要采用不同种类的重启策略<sup>[43,46]</sup>。

**3. 子句管理** 学习子句的添加会帮助 CDCL 进行剪枝，但是过多的学习子句会拖累 BCP 的性能，因此需要定期的删除掉一部分质量较低的学习子句来防止空间爆炸。学者们针对该问题提出了不同的方法。BerkMin 求解器认为新产

生的子句更难得到故价值更高，它会优先删除长度大于 8 的早期学到的学习子句<sup>[76]</sup>；MiniSat 求解器利用动态的子句活跃程度来衡量子句的质量并定期的删除不活跃的子句<sup>[44]</sup>；Glucose 求解器利用 LBD 来衡量子句的质量，并定期删掉 LBD 差的子句<sup>[68]</sup>，CoMiniSatPS 求解器在此基础之上提出了著名的三层子句管理策略<sup>[43]</sup>；Jabour 等人提出了定期从长度大于某阈值的学习子句中随机删一部分子句的策略，提出了多种动态的调整策略，且提出了采用回退深度和子句中决策层的差异性作为衡量子句质量的标准<sup>[85]</sup>。且 CDCL 中的重启与子句管理需要精细的设计，否则有可能会影响算法的完备性。

**4. 公式化简** 建模时的自动编码技术有时会引入多余的变元，预处理器可以通过化简公式的形式帮助求解器减轻一部分求解负担，最早实现于 SatELite 中的限界变元消解（bounded variable elimination）技术使得求解器的性能得到了大幅的提高，并仍然为当代最有效的预处理技术之一<sup>[86,87]</sup>。此外，当代求解器常见的化简技术有例如自包含归结（self-subsuming elimination）<sup>[86]</sup>、受限子句消除（blocked clause elimination）<sup>[88]</sup>、等价文字替换（equivalent-literal substitution）<sup>[89]</sup>、超二元归结（hyper binary resolution）<sup>[90]</sup>、影响区域化简（cone-of-influence reduction）<sup>[91]</sup>、子句蒸馏（distillation/vivification）<sup>[92–94]</sup>等技术。预处理技术（Pre-processing）属于编码阶段优化，考虑到 CDCL 会不断的学习新知识，这些知识有可能会帮助公式进一步化简，因此学者们提出了过程中处理技术（In-processing），定期的在求解过程中对公式进行化简<sup>[95–97]</sup>。对于单条子句来说，可以尝试在子句产生时删除一部分其中的文字<sup>[94,98,99]</sup> 或查看该子句是否已经被包含<sup>[100–102]</sup>，尝试将该子句用于化简其它子句<sup>[86,95,103]</sup>。

一般来说，化简的目的是使的公式变的更加简洁、子句的长度和 LBD 更短，但实际上冗余子句的添加也有可能会提高求解的效率<sup>[104]</sup>。

### 2.2.2 SAT 问题的非完备算法

SAT 问题的非完备算法主要基于随机局部搜索（Stochastic Local Search，简称 SLS）技术，它将 SAT 问题的求解抽象为不可满足子句数目的优化问题，当不存在未满足子句时问题得到解决。但是由于 SLS 一般不存储搜索历史信息，无法完整遍历搜索空间，故一般无法证明问题的不可满足性。另外，SLS 一般存在比较严重的循环现象，即会重复的搜索同一个子空间，且无法保证终止性。破

---

**Algorithm 2:** SLS 算法框架伪代码

---

**输入:** 一个 CNF 公式

**输出:** 求解结果 ‘SAT’，或者没有找到解。

```

1  $\alpha \leftarrow initialize();$ 
2 while 资源未耗尽 do
3   if 所有子句都被满足 then
4     return ‘SAT’;
5    $op \leftarrow choose\_operator();$ 
6    $perform(\alpha, op);$ 
7    $update();$ 
8 return ‘unknown’;

```

---

环技术是局部搜索的核心研究问题之一。局部搜索与 CDCL 有各自擅长的领域，有求解能力的互补性。当问题规模较大且无明显结构特征时，以 CDCL 为代表的系统搜索算法搜索代价较高，且可收集的局部信息有限，此时轻量的 SLS 通常为最有效的方法之一<sup>[105]</sup>。

为了使用 SLS 求解 SAT 问题，本文首先需要定义搜索空间和邻域关系的概念。

**定义 2.15 (SAT 问题的搜索空间与邻域关系).** 包含  $n$  个变元的 SAT 公式的**搜索空间**为所有可能的完全赋值组成的集合  $\{\perp, \top\}^n$ ，并称集合中的任意一个元素为一个**候选解** (candidate solution)。对于 SAT 问题中的一个候选解  $\alpha$ ，本文定义  $\alpha$  的**邻域** (neighborhood) 为其 Hamming 邻域，即与  $\alpha$  只有一个变元赋值有区别的候选解的集合，集合中的任一元素为  $\alpha$  的邻居 (neighbor)。

根据以上的定义，局部搜索算法可以从搜索空间中的任意候选解出发，不断的评估其邻域的邻居，然后行走至一个看起来可行的邻居位置，直到找到一个解为止。根据该方法论，本文可以将算子定义为变元赋值的翻转操作  $flip(\alpha, v)$ 。

### 2.2.2.1 随机局部搜索算法框架

一个通用的 SLS 的算法框架如算法2所示，该框架也同样也适用于其他问题。初始时，SLS 会采用贪心或者随机策略构造一个完全赋值  $\alpha$  作为初始解，并且会于这个阶段初始化评分函数和各种数据结构（第 1 行）。随后，算法会在给定的时间、内存等资源限制下（第 3 行）不断的根据评分函数和破环策略选择一

个算子（第5行），执行该算子（第6行），并更新相关的打分和数据结构（第7行），直到 $\alpha$ 为一个解为止（第3–4行）。

为了设计SAT问题的SLS算法，需要针对其四个主要的元素：初始解构造，评分函数，算子选择，破坏技术进行设计。

**1. 初始解构造** 大部分主流的SAT求解器例如CCAnr<sup>[48]</sup>, Sattime<sup>[106]</sup>, Prob-SAT<sup>[107]</sup>等采用了随机初始化的方法。YalSAT会在每次重启时利用历史上搜索到的局部最优解来初始化<sup>[49]</sup>。一些MaxSAT求解器<sup>[108,109]</sup>和SAT求解器<sup>[110]</sup>会利用BCP来构造初始解。此外，也有一些混合算法采用CDCL求解器来构造初始解<sup>[111,112]</sup>，或者采用机器学习产生一个高质量的初始解<sup>[113]</sup>。

**2. 评分函数** 评分函数分包含对候选解、变元和子句的打分，分别用于启发式的衡量候选解与实际解的差异性、翻转变元带来的收益与子句的重要程度。

对于给定的CNF  $F$ ，若子句评分权重一致时，定义  $cost(F, \alpha)$  为赋值  $\alpha$  下未满足的子句数；若采用动态子句权重打分时，定义  $cost(F, \alpha)$  为  $\alpha$  下不可满足的子句的动态权重之和<sup>[114]</sup>。如果  $\alpha$  的所有邻居的评分都差于  $\alpha$ ，则SLS的搜索遇到了一个**局部最优** (local optima)。

翻转变元的打分函数一般有两种，一种为变元最近一次变动的时间戳，倾向于选择时间戳更早的变元，主要考虑到了搜索空间的探索性<sup>[115]</sup>；另一类主要借助了候选解质量的变化作为评估，常见的打分有： $score(F, v) = cost(F, \alpha) - cost(F, \alpha')$ ，其中  $\alpha'$  表示直接执行了翻转操作  $flip(\alpha, v)$  之后的候选解<sup>[116]</sup>； $make(F, v)$  表示翻转变元赋值之后由不满足变为满足的子句权重之和； $break(F, v)$  表示翻转变元赋值之后由满足变为不满足的子句权重之和<sup>[117]</sup>。 $score(F, v) = make(F, v) - break(F, v)$ 。此外，CCAsubscore求解器中提出了一种基于二阶邻居关系的打分策略<sup>[118]</sup>。一般我们称打分正收益 ( $score > 0$ ) 的操作为下降 (decreasing) 操作。

动态子句打分属于动态局部搜索领域的关键技术，代表性的打分方式有：

- SDF(smoothed descent and flood)<sup>[119]</sup>。SLS局部最优时增加不满足子句的权重，并归一化子句权重，使得任意两次翻转之间的最大成本差保持不变；随后，所有的满足子句的权重会通过函数  $w(c) = \rho w(c) + (1 - \rho)\overline{w_{sat}}$  进行平滑，其

中  $\bar{w}_{sat}$  是所有满足子句的权重的均值，参数  $0 < \rho < 1$ 。

- ESG(exponentiated subgradient)<sup>[120]</sup>。局部最优的时候，以固定概率从所有出现在未满足子句中的变元中随机选择变元翻转；否则，才会更新子句权重。更新打分时，满足子句和未满足子句会乘不同的系数，然后所有的权重会通过  $w(c) = \rho w(c) + (1 - \rho)\bar{w}$  进行平滑，其中  $\bar{w}$  是所有子句权重的均值，参数  $0 < \rho < 1$ 。
- SAPS(scaling and probabilistic smoothing)<sup>[121]</sup>。局部最优时，以固定概率从所有出现在未满足子句中的变元中随机选择变元翻转；否则才会更新子句权重。更新打分时，未满足子句会乘一个系数，然后会以一定的概率通过  $w(c) = \rho w(c) + (1 - \rho)\bar{w}$  函数平滑权重，其中  $\bar{w}$  是所有子句权重的均值，参数  $0 < \rho < 1$ 。
- PAWS(pure additive weighting scheme called PAWS)<sup>[122]</sup>。所有未满足的子句权重会在局部最优时自加 1，然后当子句权重增加一定次数之后会将所有子句的权重同时减一。该技术及改进版被应用于 CCASat<sup>[123]</sup>、EagleUP<sup>[124]</sup>、Sparrow2011<sup>[106]</sup> 等主流的求解器。

**3. 破坏技术** SLS 会经常的访问同一个候选解，严重影响求解性能。破坏技术即打破搜索循环的技术，SAT 问题 SLS 技术中常用的方法可以分为两类。一类是基于禁忌策略 (tabu) 的方法，它会禁止翻转禁忌步长内曾经发生过翻转的变元。另一类则是基于格局检测 (configuration checking)<sup>[125]</sup> 的方法，当一个变元的邻居没有发生过变化或者子句状态<sup>[126]</sup> 没有发生变化时，则不允许翻转该变元。

**4. 算子选择** SAT 问题的算子选择可以看作是如何选择翻转操作  $flip(\alpha, v)$  中的翻转变元  $v$ 。主流的变元选择策略可以分为两类，一类是以 WalkSAT<sup>[117]</sup>、Novelty 系列算法<sup>[127–129]</sup>、ProbSAT<sup>[107]</sup> 为代表的纯随机游走算法。该类算法会按照不同的概率分布或者策略从不满足的子句中随机地选择一个变元来翻转<sup>[117]</sup>；另一类则是以 GWSAT<sup>[130]</sup>、Sparrow2011<sup>[131]</sup>、Sattime<sup>[106]</sup>、CCAnr<sup>[48]</sup> 等求解器为代表的双模式选择启发式。该类方法会综合的考虑集中性 (intensification) 与疏散性 (diversification)，在集中性模式下依照变元的打分函数贪心地选择，在疏散性模式下采用随机游走的模式跳出局部最优，并更新子句权重。

### 2.2.3 SAT问题并行求解技术

SAT问题并行技术可以分为三类：基于组件并行化、基于分解的方法和基于策略组合的方法。并主要以后两者为主。

组件并行化指的将算法中时间开销较大的组件通过并行的方法加速，例如将开销最大的组件布尔约束传播组件并行化<sup>[132,133]</sup>、子句化简并行化<sup>[134]</sup>、预处理并行化<sup>[135]</sup>等。另有一些在GPU上实现的化简技术<sup>[136]</sup>。近期，该类技术中比较成功的技术为Frost，实现了过程中处理技术的并行化<sup>[137]</sup>。该系列技术受到推理间约束制约会因为同步原因造成线程等待，并行度相对较差。

基于分解的方法可以分为求解公式分解<sup>[138]</sup>和求解空间分解<sup>[139,140]</sup>两种：前者主要利用一些如Craig差值等近似方法的帮助下最小化共享变量的数量<sup>[138]</sup>；后者更常见，主要为以分治法(divide-and-conquer)为代表的方法，将搜索空间分解为多个小空间然后单独求解。好的分解策略对线程间负载均衡至关重要，据此也涌现了诸如基于前瞻<sup>[141]</sup>、传播率<sup>[142]</sup>、机器学习<sup>[143]</sup>等技术的预划分和过程中动态划分<sup>[144]</sup>技术。分治法早期的代表有基于网格(grid)划分的方法<sup>[145,146]</sup>，而近期的代表方法为立方攻克法(cube-and-conquer)<sup>[141]</sup>。立方攻克法采用前瞻法<sup>[147]</sup>将搜索空间分割为上万字文字，并用增量式<sup>[148]</sup>求解器求解，在重划分技术<sup>[149]</sup>的帮助下负载不均衡现象得以缓解。在分布式、云计算等大规模分布式求解技术<sup>[141,149]</sup>的帮助下，该技术在毕氏三元组<sup>[150]</sup>和舒尔数<sup>[15]</sup>等难解数学问题上取得了突破成果，但该类技术在工业样例上表现较差<sup>1</sup>。

基于策略组合(portfolio)的方法简单的在各线程上运行不同的SAT求解器或者同一求解器的不同参数配置，后者被特称为多样性(diversity)方法<sup>[151–155]</sup>。学者提出了各样的多样性策略，如借助变量关系<sup>[156]</sup>、社区邻居结构<sup>[157]</sup>和最主流的互补组件和参数的组合配置<sup>[151,154,155]</sup>等技术的多样性策略。借助于子句共享(clause sharing)技术<sup>[158]</sup>，线程间重复搜索相同子空间的情况得到了缓解。尽管实现简单，但是策略组合策略是相比下更流行，且在SAT比赛中综合表现更好的方法<sup>[151–153,155]</sup>，促进了诸如ManySAT<sup>[153]</sup>，HordeSAT<sup>[152]</sup>，PaInLeSS<sup>[151]</sup>，PaKis<sup>[155]</sup>等高效并行求解器的诞生。伴随着UNSAT并行证明<sup>[159]</sup>、确定性并行策略组合技术<sup>[160,161]</sup>、配置选择<sup>[162]</sup>等技术的深入研究，该类技术愈发成熟，但是该类技术面临求解能力受最优配置限制和互补参数数目受限难以大规模并行

<sup>1</sup>从SAT比赛历年的求解结果可知<http://www.satcompetition.org/>。

的难题。

### 2.3 SMT 问题求解算法

本文介绍了 SMT 问题的惰性算法、积极算法，并介绍了 SMT(IA) 问题，尤其是 SMT(NIA) 问题求解中常见的其他技术，例如局部搜索算法、MCSat/NLSat 和线性化等技术。

#### 2.3.1 惰性算法

目前用于求解 SMT 问题最流行的算法是惰性算法，以 DPLL(T) 框架闻名<sup>[163–165]</sup>。DPLL(T) 也被称为 CDCL(T)，是 SMT 问题领域中研究的最热点问题，主流的求解器 MathSAT5<sup>[166]</sup>，CVC5<sup>[38]</sup>，Yices2<sup>[39]</sup>，Z3<sup>[37]</sup> 等主流的 SMT 求解器中均实现了该技术。

CDCL(T) 框架可以看作是在 CDCL 框架的基础上拓展了特定背景理论的推理技术，如图2.4中的交互逻辑所示，除了 CDCL 中单元传播、冲突分析、回退、分支等 CDCL 的原有组件之外，CDCL(T) 中添加了理论分析相关的组件。早期的 CDCL(T) 会在所有布尔骨架已经完全赋值且一致的情况下，才会进入理论分析的阶段<sup>[167]</sup>；目前主流的 CDCL(T) 会尽早的进行理论推理，即在布尔骨架的部分赋值一致的情况下，提前进行轻量级的判断，然后当全部布尔变元均赋值时执行彻底的理论分析<sup>[168]</sup>。

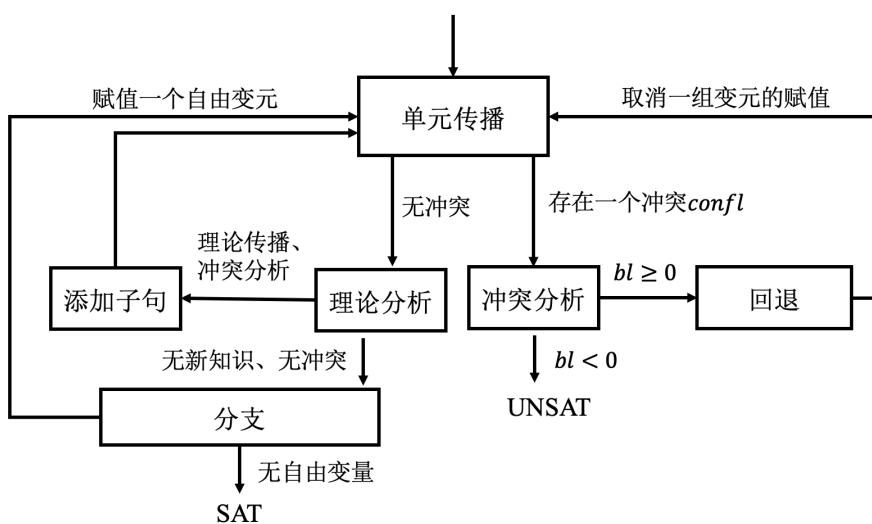


图 2.4 CDCL(T) 框架核心组件间的交互逻辑概览

Figure 2.4 An overview of the interaction logic of core components in the CDCL(T) framework

理论分析中主要在执行**理论传播** (theory propagation) [167–172]，主要调用一个理论求解器 (theory solver) 来判断当前 (部分) 赋值下的公式是否蕴含有价值的新理论约束<sup>[172–174]</sup>，或者观察是否存在矛盾，并可以从中学习到一些新的知识<sup>[163,164]</sup>。例2.3构造了一个例子，用于演示理论求解器在理论传播遇到冲突、和可以推理出新公式的两种情形。

**例 2.3.**  $F_{org} = (\neg p \vee \neg(x \leq 2)) \wedge (p \vee (x^2 - 4x + 3 \leq 0)) \wedge (p \vee (x^2 - 2x \leq 0))$  是一个建立在命题变元  $p$  和算术变元  $x$  上的 SMT(NIA) 公式，通过为算术文字 ( $x \leq 2$ ), ( $x^2 - 4x + 3 \leq 0$ ) 和 ( $x^2 - 2x \leq 0$ ) 引入三个抽象布尔变元  $\sigma_1, \sigma_2$  和  $\sigma_3$ ，可以得到其布尔骨架为  $S = (\neg p \vee \neg\sigma_1) \wedge (p \vee \sigma_2) \wedge (p \vee \sigma_3)$ 。

假如第一次的部分赋值为  $\alpha_1 = \{\sigma_1 \rightarrow \perp, \sigma_3 \rightarrow \top\}$ ，此时对应的理论公式为  $(x > 2) \wedge (x^2 - 1x \leq 0)$ ，理论传播推出矛盾，因此可以得到一条学习子句  $\sigma_1 \vee \neg\sigma_3$ ，并得到  $S' = (\neg p \vee \neg\sigma_1) \wedge (p \vee \sigma_2) \wedge (p \vee \sigma_3) \wedge (\sigma_1 \vee \neg\sigma_3)$ 。

若第二次部分赋值为  $\alpha_2 = \{p \rightarrow \perp\}$ ，可以单元传播得到布尔完全赋值  $\alpha_3 = \{p \rightarrow \perp, \sigma_2 \rightarrow \top, \sigma_3 \rightarrow \top, \sigma_1 \rightarrow \top\}$ 。此时对应的理论公式为  $(x^2 - 4x + 3 \leq 0) \wedge (x^2 - 1x \leq 0) \wedge (x \leq 2)$ 。假设，理论求解器没有在规定的时间内判断出来该公式的一致性，但是通过理论推理得到了公式  $x \geq 1$ 。通过标准化和布尔抽象，我们可以得到一个新的抽象布尔变元  $\sigma_4$  为  $x < 0$ ，从而得到新的布尔骨架  $S'' = (\neg p \vee \neg\sigma_1) \wedge (p \vee \sigma_2) \wedge (p \vee \sigma_3) \wedge (\sigma_1 \vee \neg\sigma_3) \wedge (\neg\sigma_4)$ 。

再次使用单元传播，没有推出矛盾，且得到赋值  $\alpha_4 = \{p \rightarrow \perp, \sigma_2 \rightarrow \top, \sigma_3 \rightarrow \top, \sigma_1 \rightarrow \top, \sigma_4 = \perp\}$ 。然后，通过理论求解器的分析证明了无冲突，该公式是一致的，故返回 ‘SAT’。此外，我们可以通过理论求解器找到一个解  $\alpha = \{p \rightarrow \perp, x \rightarrow 1\}$ 。

通过上述例子，读者已可大体了解了 CDCL(T) 框架的工作原理，算法3给出了一个典型的 CDCL(T) 框架的伪代码<sup>[62]</sup>，与算法1相比，不同的点主要在于第 8–11 行的部分。当算法单元传播无法推出矛盾时，CDCL(T) 不会直接进入第 10 行的分支部分，而是会采用理论求解器来检查当前赋值下的理论公式的可满足性，并返回理论推理的知识  $f$  和理论传播的结果  $r$  (第 8 行)。理论求解器如果在给定资源限制下证明了当前赋值下的理论公式是不一致的，则会根据矛盾推出的原因，并产生新的子句加入到公式里；但是，无论理论传播是否能推出矛盾，理论分析部分都有可能得到一些新的知识并作为约束或者子句加入到公式

**Algorithm 3:** CDCL(T) 算法框架伪代码

**输入:**一个 SMT 问题的 CNF 公式  
**输出:**求解结果 ‘SAT’ 或者 ‘UNSAT’

---

```

1 while true do
2   conflict  $\leftarrow$  propagate();
3   if c  $\neq \emptyset$  then
4     bl  $\leftarrow$  analyse_conflict(c);
5     if bl < 0 then return ‘UNSAT’;
6     backtrack(bl);
7   else
8     <f, r>  $\leftarrow$  deduction();
9     if r = ‘SAT’ then
10       if !decide() then return ‘SAT’;
11     if f  $\neq \emptyset$  then add_clause(trans(f));

```

---

中(第 11 行)。另一方面,若理论求解器可以证明当前赋值下的理论公式是可满足的(第 9 行),则会进入布尔骨架公式下的分支部分,若当前的赋值是一个完全赋值,因为此时布尔骨架和对应的逻辑公式均是一致的,所以算法返回‘SAT’(第 10 行)。

### 2.3.1.1 算术理论求解方法与 SMT 理论求解器

SMT 问题的理论求解器核心的任务是判断给定的一个无命题逻辑约束的算术公式的一致性,且 SMT 算术理论求解器可以看作是一个针对 CDCL(T) 框架特殊优化的、无目标的线性规划 (Linear Programming) 或整数规划 (Integer Programming) 求解器。

为了求解不同的算术理论 SMT 问题,SMT 求解器中引入了很多方法。为了求解线性实数理论 (LRA),除了区间约束传播 (interval constraint propagation,简称 ICP) [175] 外,SMT 求解器也会引入一些线性规划领域的方法,例如单纯形法 (simplex) [176]。为了求解非线性实数理论 (NRA),SMT 求解器通常会采用代数的决策过程,例如虚拟替代法 (virtual substitution, 简称 VS) [177] 或者柱形代数分解 (cylindrical algebraic decomposition, 简称 CAD) [178]。为了求解整数线性理论 (LIA),例如 Z3<sup>[37]</sup>、CVC4<sup>[179]</sup>、MathSAT5<sup>[166]</sup>、veriT<sup>[180]</sup> 和 Sateen<sup>[181]</sup> 等众多主流的求解器主要沿用了 Bruno Dutertre 等人的求解思路<sup>[182]</sup>,设计并优

化了一种基于分支限界法 (branch and bound, 简称 BaB) 的求解框架<sup>[166,183]</sup>。除 BaB 之外, 常见的方法还有 Fourier-Motzkin 变元消解法<sup>[184]</sup>、Omega Test<sup>[185]</sup> 等技术。

SMT(NIA) 问题可以看作是在 SAT 问题上拓展了有关 NIA 的算术文字。通常, SMT(NIA) 问题是不可判定的。实际上, Matiyasevich 定理<sup>[186]</sup> 通过证明 NIA 的可满足性在理论上是不可判定的, 从而表明 Hilbert 的第十个问题是否定的。因此, 除非是有界或特殊构造的问题, 所有面向 SMT(NIA) 问题的通用算法都是不完备的。由于其求解难度高, 支持该理论可满足性判定的求解器相对较少, 其理论求解器要么是针对特定有限域下的完备算法, 要么为各种非完备算法, 处于百花齐放的阶段<sup>[187]</sup>。有一些求解器采用了包含基于线性化的方法<sup>[188]</sup>; 如 iSAT3<sup>[189]</sup>、raSAT<sup>[190]</sup> 等求解器则采用了在整数域上适配的 ICP 方法; 此外, BaB、VS 和 CAD 也被拓展到了 NIA 理论求解器和 SMT(NIA) 的求解算法中<sup>[187]</sup>。

然而, 从实际解决的角度来看, 许多来自应用领域的 SMT(NIA) 问题可以通过精心设计的求解器来解决。自提出以来, 学者们已经做出了相当大的努力, 并开发了许多有效的方法来解决 SMT(NIA) 问题, 这些方法可以为许多实际的 SMT(NIA) 问题产生安全的结果。

在程序验证、终止性分析等领域, SMT(NIA) 问题有重要的应用价值。具体地, 在状态可达性验证<sup>[33]</sup>、不变式生成<sup>[34]</sup> 和终止性/非终止性证明<sup>[35,36]</sup> 等问题上, 一个流行的方法是为不变式或者排序函数假设一个线性的模板, 并用 Farkas 引理来消除全称量词。这些方法被称为不变式综合或者排序函数综合<sup>[191]</sup>, 其中 Farkas 引理会引入累如  $x \times y$  的非线性项<sup>[34]</sup>。

### 2.3.2 积极算法

惰性算法会在 CDCL(T) 与理论求解器之间不断切换, 且惰性体现在在于只有抽象布尔骨架是一致时才会去启用相关的理论求解器; 而积极的方法则会同时编码命题逻辑推理与理论求解两个部分, 一次性交由 SAT 问题求解, 常见的方法为位展开 (bit-blasting) 算法, 也被称作平展 (flattening) 技术<sup>[62]</sup>。

位展开技术是目前可求解 SMT(NIA) 问题的求解器中实现最多的一类方法。一般该类技术会将 SMT(NIA) 公式转化为位向量 SMT (SMT(BV)) 公式, 然后进一步规约到与后者可满足性等价的 SAT 公式, 借助高效的 SAT 求解器得到 SMT(BV) 公式的可满足性。若原公式中的算术变元取值范围有限制, 则可以通

过该方法判定原公式的可满足性；若原公式中的算术变元是无穷的，则会假定算术变元为位宽固定的位向量，并在该假设下判定公式的可满足性，若求解后为满足则原公式满足；如果假定空间内没有解，则会依赖启发式对假定的空间进行拓展或修正，然后再次求解；如果启发式选择的得当，该类方法也有可能证明一些不可满足的逻辑公式<sup>[62,192]</sup>。

最早期的 SMT(BV) 问题求解方法出现于 SVC<sup>[193]</sup> 和 ICS<sup>[194]</sup> 求解器。早期，SVC 采用二叉决策图来求解命题逻辑公式，后来被 CVC<sup>[195]</sup> 求解器取代，然后被 CVC-Lite<sup>[196]</sup> 和 STP<sup>[197]</sup> 等求解器取代；ICS 则被 Yicse 求解器取代。现今所有主流的求解器均采用 SAT 求解器作为命题逻辑公式的推理引擎。Boolector<sup>[198]</sup> 和其改进版本 Bitwuzla<sup>[199]</sup> 是专注于求解 SMT(BV) 公式的求解器，在字层级化简、位层级化简上做了很多工作。有一些 SMT(BV) 问题的工作会将公式分为不同的层级，并增量式的逐层展开<sup>[62,200,201]</sup>。在目前主流的一些 SMT(NIA) 求解器中也实现了位展开方法，与其它方法互补，并发挥了重要的作用，如 AProVE<sup>[202]</sup>、CVC4<sup>[179]</sup>、Z3<sup>[37]</sup>，SMT-RAT<sup>[203]</sup> 等求解器。位展开时变元值域的选择对于求解有很大的影响，近期的 BLAN 求解器<sup>[57]</sup> 提出了多种启发式策略来提高 SMT(NIA) 公式位展开后的求解效率。

### 2.3.3 SMT 问题的其它求解算法

本节介绍了 SMT 问题常见的其他求解策略，并着重强调 NIA 理论有关的技术。

SMT(NIA) 问题的求解方法没有一个固定的求解范式，且单一方法已经难以胜任该类公式的求解需求。这引起了学者们高度关注，也提出了除积极和惰性算法之外的多种求解方法，其中代表性的求解方法有局部搜索、NLSat(Non-Linear Satisfiability)<sup>[204]</sup>/MCSat(Model-constructing Satisfiability)<sup>[205]</sup> 方法、线性化等技术。

值得一提的是，本文之前 SMT 问题的随机局部搜索算法研究相对较少。

#### 2.3.3.1 NLSat/MCSat

CDCL(T) 框架中 CDCL 的算法会直接执行在命题骨架上，NLSat/MCSat 则是一种完全 CDCL 结构的算法。该类算法的伪代码几乎完全等同于算法1(19页)，区别在于第 2 行的传播与第 4 行，NLSat 和 MCSat 拓展了算术层面的推理方法。

相比 CDCL(T), NLSat/MCSat 算法最核心的数据结构为赋值序列, 它可以看作当前 CDCL 算法中赋值序列的拓展版本, 维护了 SMT 公式中命题变元和算术变元的部分指派。当求解结果是满足时, 算法可以从这个赋值序列中直接得到变元的所有赋值。赋值序列里面的变元可以分为通过分支启发式选择的决策变元和通过其它变元赋值推导出赋值的变元, 通过该序列和推理关系可以得到一个蕴含图。如果当前的赋值推出了矛盾 (有可能是命题逻辑层面的冲突, 也有可能为算术层面的冲突), 则会执行冲突分析, 然后通过布尔归结原理、调节法基本规则 (ground paramodulation rules) 得到一条学习子句<sup>[205]</sup>。

NLSat/MCSat 作为一类新型方法, 已经被许多主流的 SMT 求解器实现, 甚至可以当作 CDCL(T) 框架的替代。目前实现了该技术的代表求解器有 Z3<sup>[37]</sup>, Yices2<sup>[206]</sup> 等。

### 2.3.3.2 线性化与分支限界

最常见的线性化方法被称为增量线性化 (incremental linearization)<sup>[207]</sup>, 主流的求解器例如 CVC5<sup>[208]</sup> 和 MathSAT<sup>[166,209]</sup> 等在求解 SMT(NIA) 公式时主要采用该方法。CVC 系列求解器在线性化方面作出了很多研究, 并与位展开方法做了深度的结合<sup>[207,210]</sup>。最近, 线性化的 SMT(NIA) 求解工作通常将求解 SMT(NRA) 问题的一些技术与 BaB 做了一定的结合<sup>[187,205]</sup>。

线性化方法区别于 CDCL(T) 和 MCSat, 线性化是一种将 SMT(NIA) 公式转化为 SMT(LIA) 公式进行求解的方法, 在实践中也有不错的求解效果。线性化和位展开算法的主要目标一致, 为在假定的空间内找解的方法。其核心思想是将 NIA 变元松弛为 NRA 变元并采用 NRA 的求解方法求解, 因为松弛后的问题取值范围为 NIA 问题的上界, 因此若松弛后的问题没有解, 原始问题也为不可满足的。该思想属于分支限界的范畴。若求解松弛问题得到一组解, 且存在非整数变元赋值  $x \rightarrow a$ , 则向 NRA 公式添加一条新的约束  $x \leq \lfloor a \rfloor \vee x \geq \lceil a \rceil$ , 然后再次求解, 直到找到一组整数解为止<sup>[62,205]</sup>。此外, 增量线性化相关的技术会将 NIA 单项式抽象为一个实数算术变元处理, 并于线性化之后进行剪枝<sup>[211]</sup>; 此外, 有的方法会类似于位展开方法, 为无界变元设置一个抽象的界然后迭代扩大<sup>[212]</sup>; 为了更好的选择分支变元或者扩大变元的值域, 学者们提出了多种启发式策略<sup>[212]</sup>。

### 2.3.3.3 SMT 问题的局部搜索算法

SMT 局部搜索相关的研究较少。最初，Alberto 尝试将 WalkSAT 局部搜索求解器替换掉 CDCL(T) 中的 CDCL 引擎，将局部搜索算法直接用于求解布尔骨架的求解，这是局部搜索在 SMT 问题领域的第一个尝试<sup>[213]</sup>。

位向量领域的局部搜索算法首次出现于 Andreas 等人的工作，他们将 SAT 局部搜索算法应用到了 SMT(BV) 问题的求解上，并通过考虑了位向量结构设计了一系列的邻域操作和打分函数<sup>[214]</sup>。后来，Niemetz 等人借鉴了自动测试向量生成领域中“backtrack”的思想，提出了一种非打分的局部搜索算法，其核心思想为按照语法树的顺序，按照从输出到输入的顺序不断的执行值传播，直到找到解或者搜索空间全部搜索完毕为止。该方法极大的改进了求解性能，并被集成于 Boolector 求解器中。<sup>[215,216]</sup>

针对算术理论的局部搜索算法直到近期才首次出现。中科院软件所蔡少伟团队于 2021 年提出了首个整数差分逻辑 (IDL) 的局部搜索求解器，针对算术文字提出了基于“critical move”的邻域操作策略，并将其拓展到了 NIA、NRA、LIA 等领域，形成了 LocalSMT 求解器，极大的改进了算术理论 SMT 问题的求解能力。<sup>[56,59,217]</sup> 此外，针对 NIA 问题，中科院软件所张健团队提出了一种基于位展开的局部搜索算法。<sup>[57]</sup>

另外，北京大学夏壁灿团队提出了针对 SMT(NRA) 问题的算法，设计了一种基于实根隔离的新操作，称为胞腔跳跃，它沿着给定的方向更新当前赋值，使赋值可以从一个胞腔“跳跃”到另一个胞腔。一个胞腔跳跃可以调整几个变量的值。另外，他们还设计了两级操作选择，以平衡成功率和效率。该算法在含有高次多项式的公式中效果尤为显著。<sup>[58]</sup>

## 2.4 SAT、SMT 问题相关工作历史

按照是否具备完备性，SAT 问题的求解算法可以分为完备算法与非完备算法。完备算法主要是基于系统搜索的方法，利用推理剪枝加速求解；非完备算法主要是随机局部搜索算法，用于短时间内在某一给定空间附近利用启发式快速找解。最早有关 SAT 问题的有效算法为 1960 年基于归结的 DP 算法<sup>[6]</sup>，1962 年 DPLL 算法的提出奠定了 SAT 求解器回溯法的雏形<sup>[7]</sup>。到了 90 年代中期，Silva 等人在 DPLL 的算法的基础上突破性的引入了非时序回退<sup>[218]</sup> 与冲突分析<sup>[219]</sup>

等技术<sup>[220]</sup>，形成了最早期的 CDCL 的求解器 Grasp<sup>[221]</sup>。与此同时，最早的 SAT 局部搜索算法于 1992 年分别出现于 Gu<sup>[222]</sup> 和 Selman<sup>[116]</sup> 的工作中。

**完备求解技术历史** CDCL 算法的提出是当代 SAT 问题的完备算法的基石，之后国际团队的完备算法基本均沿用了该框架。1998 年 Gomes 等人提出用重启的策略来改进可满足性样例的求解能力<sup>[81,223]</sup>。2001 年，Chaff 求解器中 VSIDS 变元分支启发式的提出<sup>[42]</sup> 和惰性单元传播数据结构<sup>[100]</sup> 的提出，大幅的改进了求解效率。2003 年，Een 开发了一个结构简洁的工程优化版本求解器 Minisat<sup>[44]</sup>，迭代了数个版本，成为了当代受众最多的 SAT 求解器。2007 年，Pipatsrisawat 提出了相位保存技术<sup>[75]</sup>，修复了非时序回溯赋值擦除引发的负面效果；同年，Huang 提出了将 Luby 重启引入到 CDCL 中<sup>[46]</sup>。2009 年，Audemard 等人设计的 Glucose 求解器中提出了衡量学习子句质量启发式评分 LBD，并据此采用统计学的方法对 CDCL 的子句管理、重启等多个组件进行了优化<sup>[68]</sup>。2015 年，Oh 等人提出了基于 LBD 的三层学习子句管理策略<sup>[43]</sup>。2016 年以来，Liang 等人提出了 MapleSat 系列求解器，采用多臂老虎机和设计了一种基于学习率的启发式策略<sup>[74]</sup>；后来，华中科技大学吕志鹏教授和法国 Chumin Li 教授等人采用单元传播定期对子句进行化简<sup>[94,224,225]</sup>，Nadel 提出了一种时序与非时序回退相混合的方法<sup>[226]</sup>，均改进了 Maple 求解器。Soos 针对密码学问题专门设计了 CryptoMiniSat 求解器<sup>[52]</sup>。德国形式化专家 Armin Biere 教授长期研究 SAT 求解器，研制了一系列的高性能 SAT 求解器，其研制的 SAT 求解器 CaDiCaL 和 kissat 代表了近期最前沿的完备求解器<sup>[50]</sup>，最近几年也涌现了诸多在其上改进的算法<sup>[227–229]</sup>。

**非完备求解技术历史** 非完备算法方面，自 1992 年 GSAT<sup>[116]</sup> 提出以来，学者们进行了大量的改进，比较有代表性的方法为基于冲突随机游走的 WalkSAT 算法<sup>[117]</sup>。后来，一个较关键的技术为 2005 年法国 Chumin Li 教授等人提出的希望下降变元的概念，并被集成于 G<sup>2</sup>WSAT 算法中，并于后续几年被广泛采用<sup>[129]</sup>。随着工业样例规模的激增，局部搜索与 CDCL 的差距愈发明显，而局部搜索算法大部分局限于求解随机问题的缺陷逐渐被暴露出来。2010 年以后，兴起了一批新型的局部搜索算法，其求解能力大幅超过之前的算法，且在人工结构样例上，尤其是一些图论和数学的问题上，取得了长足的进步，并与 CDCL 算法表现出互补的性能。其中，比较有代表性的方法包括：法国 Chumin Li 教授设计的基

于子句满足时间的 Sattime<sup>[106]</sup>、德国乌尔姆大学 Balint 设计基于概率分布技术的 ProbSAT<sup>[107]</sup>、形式化专家 Biere 的 YalSAT<sup>[49]</sup> 以及中科院软件所蔡少伟研究员提出的基于格局检测策略的 CCAAnr<sup>[48]</sup>。此外，局部搜索算法在对 SAT 求解的上界和最差时间复杂度的分析上也发挥了重要的理论价值。

**其它 SAT 问题求解技术** 为了充分利用两类算法的优势，学者们尝试将两种方法进行结合，但是之前的方法均互相把彼此当作为黑盒来调用，并在不同的时机启用<sup>[111,112,230–232]</sup>。混合求解是本文的重点讨论内容，相关内容的讨论将于第 4、5 章详细展开。

为了更好的融入实际应用，增量式 SAT 求解技术被提出，并被应用于连续求解一系列相似的 SAT 问题<sup>[148]</sup>。近期，随着机器学习方法的流行，也出现了尝试通过机器学习技术设计 SAT 求解器<sup>[233]</sup> 或提高 SAT 求解器能力<sup>[113,234]</sup> 的工作。为了解决单核心求解能力受限的难题，并行求解技术也应运而生<sup>[235]</sup>。

**SMT 问题求解技术历史** SMT 问题的算法研究起步于 2000 年左右，相对 SAT 问题来说，起步较晚，是一个新兴的领域。最初人们尝试用 SAT 问题来建模词级别（word-level）的软件测试模型时，会出现建模复杂、规模严重膨胀、模型难以解读等问题，SMT 问题的建模则完美的解决了该问题，既保留了建模文件的可读性，又可以增加求解的灵活性<sup>[62]</sup>。

SMT 问题由于其强大的建模表达能力，在软硬件验证领域发挥了重要的作用，同时也涌现了大量关于 SMT 求解的工作<sup>[163]</sup>。不同的背景理论下的 SMT 求解算法各有不同，最常见的一类方法为基于 SAT 求解技术的方法，并按照调用 SAT 求解器的方式可以分为积极（eager）算法和惰性（lazy）算法<sup>[62]</sup>。积极算法会将 SMT 问题直接转化为 SAT 公式求解，而 lazy 方法则会从 SMT 问题上抽象出一个命题逻辑骨架，采用 CDCL 算法求解该骨架，并在合适的时机调用相关的理论求解器求解。lazy 方法也被称为 CDCL(T) 框架。

相对来说，SMT 问题领域的局部搜索算法比较稀少，进而也不存在 CDCL(T) 与局部搜索的深度混合算法。2009 年，Alberto 等人将局部搜索用于求解 SMT 问题的布尔骨架，但是不涉及背景理论的求解<sup>[213]</sup>。2015 年，Armin Biere 团队提出了第一个求解 SMT(BV) 问题的局部搜索算法 BV-SLS<sup>[214]</sup>，并紧接着加入了位层级（bit-level）路径传播技术<sup>[215]</sup>，并于次年加入了词级别传播技术<sup>[216]</sup>。相关

算法可以达到与 eager 方法互补的效果，证明了局部搜索技术在 SMT 领域的巨大潜力。2021 年以后，中国科学院蔡少伟团队<sup>[56,217]</sup>、中国科学院张健、马菲菲团队<sup>[57]</sup>、北大夏壁灿团队<sup>[58]</sup>相继提出了算术理论的局部搜索算法。

相比于其它的算术理论，SMT(NIA) 问题的判定更具有挑战性。1970 年，Matiyasevich 等人解决了 Hilbert 的第十项挑战，其中证明了 NIA 问题是不可判定问题<sup>[186]</sup>。因此 SMT(NIA) 问题不存在一个行之有效的完备算法，但是从公式求解角度来看，通过精心的设计，很多实际应用背景下的 SMT(NIA) 问题可以得到高效地求解。在 SMT(NIA) 求解器发展的历程中，也出现了很多除积极算法和惰性算法以外的 SMT(NIA) 求解算法，如 MCSat/NLSat 框架、增量线性化。

SMT 求解器领域目前仍由国外的少数团队主导。最具有代表性的求解器为微软组织研制的 Z3<sup>[37]</sup>，它是目前支持的背景理论种类最多的求解器之一，且已经被成功的应用于公司内部的软件验证问题中。另外两个比较知名的求解器为斯坦福大学与爱荷华大学主导研制的 CVC5<sup>[38]</sup>，以及斯坦福国际研究院研制的 Yices2<sup>[39]</sup> 求解器。在最近几年的 SMT 比赛中，上述三个 SMT 求解器影响力最大，得奖数目多，逐渐形成了三足鼎立的格局。另外一个比较出名的 SMT 求解器为 MathSAT5<sup>[166]</sup>，它是由布鲁诺·基斯勒基金会与特伦托大学联合研制，虽然存在一定的互补性，但是综合能力相对前三者来说仍然较弱。



## 第3章 SAT问题混合重启策略研究

第2章中介绍了SAT问题有关的背景知识，CDCL求解器重启时普遍采用了保留所有学习信息的“热重启”策略，这种策略会使得重启前后的搜索方向极为相似，有可能会使得求解器陷入一个不利的空间，3.1节主要介绍了本文的研究背景和动机。3.2节通过实验发现，主流的CDCL求解器仍然存在求解时间的重尾分布现象，这为我们开展“冷重启”的研究动机提供了实验数据支撑。

3.3节针对上述问题，提出了在重启时遗忘单一类型信息的冷重启策略和遗忘多类信息的复合冷重启策略，开展了一系列的实验论证，并得到了很多有趣的结论。结果表明通过采用冷重启技术，可以较大幅度的提高可满足实例的求解能力，为此我们深入展开了讨论。基于冷重启仍然存在互补实例的情况，本文实现并测试了并行的冷重启策略，发现其同样在可满足实例求解上有巨大的帮助。同时，我们也对冷重启中学习子句的选择、学习子句在求解中的作用、并行子句共享的阈值做了进一步的探讨。实验结果也表明，为了开发一个面向可满足实例的SAT求解器，很有必要重新思考其中固有的启发式。

本文相关技术被应用于并行求解器的研制中，以领先第二名1/4 PAR2打分的成绩获SAT比赛2022年并行主赛道冠军，改进版本蝉联了2023年并行主赛道的冠军。

### 3.1 引言

SAT问题是第一个被证明的NP完全问题，也是计算机科学的核心问题之一<sup>[236]</sup>。当代SAT求解器中，CDCL框架是最高效也是最流行的SAT求解技术，这归功于其中的多种求解技术的混合<sup>[42,221]</sup>。

重启是对SAT求解性能贡献较大的一类技术。Gomes等人发现当求解可满足实例的时候，在完备SAT求解算法的启发式中引入一定的扰动，会使其求解时间产生巨大的波动，并服从重尾分布<sup>[81]</sup>。后来，该团队建议在SAT问题的系统搜索算法中引入重启的技术，可以防止算法长时间卡死在一个不好的搜索方向上，从而提高求解性能<sup>[223,237,238]</sup>。在后续的研究中，学者们发现，重启策略同样对不可满足实例的求解有帮助，并通过实验表明有可能存在两个方面的原因。

一方面，重启能帮助 CDCL 压缩变元赋值的队列的长度，在变元赋值数量较少时产生解<sup>[153]</sup>；另一方面，实验表明重启能帮助当代 CDCL 求解器产生高质量的学习子句<sup>[239]</sup>。

重启策略决定了什么时候将一个 SAT 求解器当前的搜索暂停，并从头开始搜索。自 CDCL 提出以来，学者们提出了各式各样的重启方法，研究主要集中在 CDCL 算法重启时机上<sup>[238,240]</sup>。SAT 求解器上的时间主要通过冲突次数衡量<sup>[241]</sup>，根据两次重启之间的时间间隔是否是求解过程中决定的可以将重启分为静态和动态两类。

静态重启的间隔一般是由一组固定的间隔序列的倍数构成的。其中最有影响力的技术叫做 Luby 重启，其重启间隔基于 Luby 序列设计<sup>[83]</sup>并于 2007 年首次被用于 SAT 问题求解中<sup>[46]</sup>。Luby 序列已经被证明在一类名为 Las Vegas 随机算法上是最优的策略<sup>[83]</sup>。Luby 序列为一组型为按照“1, 1, 2, 1, 1, 2, 4, 1, 1, 2, 1, 1, 2, 4, 8, …”的比例分布间隔的序列，并可以将序列的第  $i$  个元素  $luby(i)$  递归的定义为：

$$luby(i) = \begin{cases} 2^{k-1}, & \text{if } t = 2^k - 1 \\ luby(i - 2^{k-1} + 1), & \text{if } 2^{k-1} \leq i < 2^k - 1 \end{cases}, \text{ 其中 } k \text{ 为自然数} \quad (3.1)$$

动态重启的间隔主要依赖当前求解过程的运行情况决定，例如 GLucose-style 重启策略采用了近期学习子句的质量和平均总体学习子句质量作为重启依据，当近期的平均质量明显差于总体均值时，则执行重启<sup>[84]</sup>。一般学习子句质量采用 LBD 作为衡量标准<sup>[68]</sup>。后来，Biere 等人提出了一个该技术的变种策略，采用 EMA (exponential moving averages) 来计算近期学习子句序列的质量和总体学习子句序列的质量<sup>[82]</sup>。此外，当启发式评估后认为求解器的搜索状态有可能快要找到可满足解时，学者们提出将当前的重启先暂时禁止掉的技术，以防重启过度，并有效的提升了可满足实例上的求解性能<sup>[84,242,243]</sup>。

根据 Oh 等人的观点，可满足实例需要长间隔的重启，不可满足实例倾向于快速重启<sup>[43,153]</sup>。静态策略间隔一般较长；动态策略一般间隔较短，因此主流的重启策略多采用动静态交替使用的策略。值得一提的是，现代求解通常采用的重启模式几乎均为高频次、低间隔的重启策略，以 MiniSat<sup>[44]</sup> 和 kissat<sup>[50]</sup> 为例，其中实现的技术使得 CDCL 每秒钟可以执行甚至上千次重启，即便采用了重启不频繁的静态重启策略。

得益于信息保留（尤其是学习子句），即便如此频繁的重启，在精心设计之后也不会影响 CDCL 求解器的完备性。当前的重启技术可以视为是一种“热重启”技术，因为它们均在重启之后完全保留了之前的搜索信息。且目前主要有三类搜索信息：变元分支决策启发式的打分和顺序；变元相位，即分支启发式选中之后赋值的倾向性<sup>[75,244]</sup>；和用于搜索空间剪枝的学习子句。这些信息在重启之后被保存下来，以免之前的计算的努力被浪费<sup>[238]</sup>。

然而，“这些求解信息是否应该被保留下？”即，主流热重启是否会过度地干预搜索过程？到本文工作之前，该问题仍然没有被研究过。本章主要的目标则是通过一系列分析的手段来经验性的回答这个问题。从实验中，我们也得到了一些有趣的发现，且通过实现我们的彻底的重启技术，前沿 SAT 求解器的性能也得到了提高。

### 3.2 热重启下的求解时间差异性现象

基于回溯式的 SAT 求解器存在一个有趣的现象：当使用不同随机种子或者启发式策略稍有变动时，它们在某个特定实例上的性能会发生巨大变化，且服从重尾分布。基于该现象，重启策略被提出。最初，当 Gomes 等人提出利用 DPLL 求解器（CDCL 的前身）的求解时间变化来设计重启时，他们向分支顺序中引入了一些随机策略<sup>[223]</sup>。后来，为了重复利用之前的搜索信息，防止浪费，更多的重启策略保留了搜索信息，本文称之为热重启。例如，MiniSat 等求解器在重启后保持了分支顺序的得分等信息<sup>[44]</sup>。目前，最先进的 CDCL 求解器仍普遍采用热重启。通过这种方式，可以提高 CDCL 求解器的性能，但代价是求解器对启发式方法（包括变元分支序和相位选择等）较为敏感。因为热重启保留了之前的变量分支得分和相位，现代 CDCL 求解器更倾向于访问之前的搜索空间。

在本节中，本文研究了代表性的先进 CDCL 求解器在不同随机初始变元序和相位启发式下的求解时间变化。

#### 3.2.1 初始变元的变化研究

为了研究初始变元顺序对于 CDCL 求解器性能的影响，本文选择 kissat-MAB<sup>[227]</sup> 作为案例研究的基础求解器，该求解器获得了 SC21 主赛道的冠军。为了防止参数过拟合，本文在 SC20 测试集上进行实验。本文选择 kissat-MAB 可以在 5000 秒内求解的例子作为本文的测试集，并将求解时间记做  $T_{baseline}$ 。

表 3.1 随机采样实验结果汇总

**Table 3.1 Results for the random sampling experiment.**

#Selected	Fail	2×	4×	10×	32×	0.5×	0.25×
Random Initial Orders							
SAT(151)	3.58	25.72	12.37	4.72	1.14	25.21	15.03
UNSAT(121)	2.1	1.7	0.06	0.0	0.0	7.78	5.22
ALL(272)	2.92	15.03	6.89	2.62	0.63	17.46	10.67
Random Initial Phases							
SAT(151)	3.71	25.29	13.23	3.87	1.03	21.95	10.91
UNSAT(121)	1.02	2.95	0.79	0.03	0.0	5.31	2.2
ALL(272)	2.51	15.35	7.7	2.17	0.57	14.54	7.03

对于选中的例子，本文会调用 kissat-MAB 求解 100 次，并随机的设置了不同的初始变元选择顺序。对于每一次采样，若其能在 5000s 内完成求解，本文会将采样求解的求解时间记做  $T_{sample}$ ，并通过计算  $\frac{T_{baseline}}{T_{sample}}$  进行归一化，表示了与默认参数下求解时间之间的变化。

本文将筛选出来的例子分为 SAT 和 UNSAT，并将求解时间的变化在图3.1中呈现。图中的纵坐标为数尺度表示，上、下子图分别用圆圈、星号表示了 SAT 和 UNSAT 的例子的求解时间变化。例子按照 kissat-MAB 的求解时间排序。此外，图中只汇报了相对 kissat-MAB 的默认配置可以在 10 分钟内求解的实例，其数据相对稳定。根据实验结果，本文可以得到下述的实验结果：

- 对于 SAT 实例，求解器的求解时间会根据初始顺序的不同而有显著变化。在 33.3% 的 SAT 实例上，出现了比初始顺序可以提高至少 32× 加速的情形。此外，有 33 个可满足的实例原始求解器无法求解，但可以通过随机初始采样序的求解器求解。
- 对于 UNSAT 实例，求解时间的变化要小得多。对于大多数 UNSAT 实例，在 100 次具有不同随机初始变元序的采样测试中，最优、最差的求解时间与默认求解时间之间的差一般在 4 倍以内。

### 3.2.2 初始相位的变化研究

除了变元分支顺序，初始相位是另一个指导 CDCL 搜索顺序的关键因素。因此，本文研究了不同随机变元初始相位对于求解时间的影响。实验设置与上小节类似，因为绘图结果上两者极其类似，本章不再赘述。

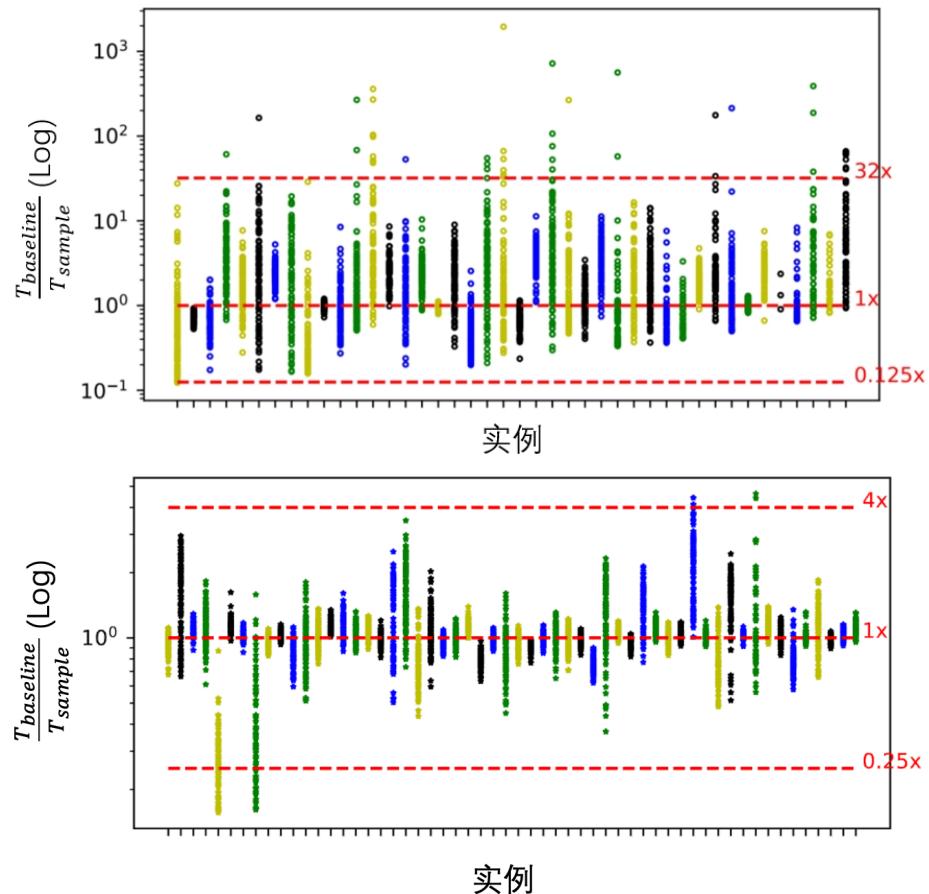


图 3.1 对数尺度下不同初始变元序的求解时间变化比率

**Figure 3.1 Ratios of the runtime variation with different initial variable orders in log scale**

在此，本文需要指出最近有一些求解器采用了相位重置技术来阶段性的重置变元的相位。它们更倾向赋值为可以引导搜索进入更深搜索空间的相位，虽然这些方法中引入了一定概率将相位重新初始化，但是这类方法并不足以完全消除重尾分布的现象。

表3.1给出了随机初始变元序和随机初始相位对于求解时间影响的实验结果，本文可以从表中观察到更详细的统计数据。对于每一个选中实例，本文统计了其100个采样结果中超时（‘Fail’列），达到 $2\times$ 、 $4\times$ 、 $10\times$ 、 $32\times$ 倍加速，和 $0.5\times$ 、 $0.25\times$ 退化的采样数。表中分为SAT、UNSAT、ALL三类，分别汇报了相关数据的平均结果。

不同初始相位求解时间的差异性促使我们思考：是否存在一个好的启发式能利用这个现象来进一步提高求解器的性能。本文尝试了通过单元传播构造一个一致性较高的初始解，这种策略类似于Knuth的热身（warm-up）策略<sup>[236]</sup>和

Relaxed 求解器（第 4 章中的技术）中初始化局部搜索候选解时的方法。但是预实验证明，该类方法并不能在 100 次采样时比纯随机方法产生更好的峰值性能。

**热重启实验观察** 整体来说，本文观察到 SAT 实例的求解时间变化要明显大于 UNSAT 实例。对于每一个例子，本文计算了 100 次采样上的求解时间的变异系数 (CV)，CV 数值越高离散程度越高。对于随机初始变元分支序的实验，SAT 实例的平均 CV 值是 71.5%，UNSAT 实例的平均 CV 值是 27.2%；对于随机初始相位的实验，SAT 实例的平均 CV 值是 71.8%，UNSAT 实例的平均 CV 值是 36.8%。通过本节的实验，本文给出了一个重要的观察：

**观察 1：**当代 CDCL 求解器的性能（常用求解时长来量化体现）对于初始变元分支序和初始相位十分敏感，特别是对于可满足实例。

### 3.3 冷启动策略的实现及其评估

对于很多组合优化问题，随机噪声的引入会导致求解时间的巨大差异，重启是解决该问题的一种高效策略<sup>[223,245]</sup>。然而，当代 SAT 求解器普遍采用频繁重启的策略，且两次重启之间不会有太多的冲突。加上变元顺序的调整主要依赖于冲突，当代 SAT 求解器更倾向于按照与重启前相似的变元搜索顺序进行搜索<sup>[70,246]</sup>。同时，因为相位保存技术的引入，变元通常会赋值为重启前的真值。因此，当代的 CDCL 重启方法会引导重启后的搜索向重启前的方向前进。

上一节中的观察 1 表明，在不同的初始变元序下，当代求解器呈现出求解时间的剧烈变化。这很有可能是因为重启时的信息保存导致的。长时间按照同一种习惯进行搜索有可能会使得求解器长时间卡在一个无解的搜索空间中，本文拟采用彻底重启的方法来缓解该现象。本节研究了本章的核心研究问题：

重启之后是否应该保存之前的搜索信息？

本文称重启后保存所有搜索信息的策略为“热重启”。几乎所有主流的 CDCL 求解器均采用该策略，且目前主要有三种被保留的信息：分支顺序、变元相位和学习子句。为了方便，本文称重启后遗忘掉之前所有搜索信息的方法为“完全冷重启”，并称遗忘部分信息的方法为“部分冷重启”。两种均属于“冷重启”的范畴，与目前主流的 CDCL 求解器中的重启技术有所区别。

### 3.3.1 冷重启策略的实现

本节介绍了三种遗忘单一类别学习信息的冷重启策略的实现，以及遗忘了多类学习信息的复合冷重启策略的实现。

根据遗忘学习信息的类型，遗忘单一信息的冷重启技术可以分为以下的三种：

- 遗忘顺序 (FO)：重启时忘记之前的变元分支顺序信息。
- 遗忘相位 (FP)：重启时忘记之前的变元相位信息。
- 遗忘子句 (FC)：重启时忘记之前的学习子句。

具体实现时，本文没有直接用冷重启替代掉热重启，而是将冷重启与热重启混合在了一起，实现了一种混合重启策略<sup>1</sup>。事实上，冷热重启之间的协作也可以看作是搜索算法中常遇到的一类问题，即如何平衡好探索新知识（冷重启）和利用之前学习信息（热重启）之间的关系。

**冷重启频率** 本文使用了一种朴素的静态重启策略，并采用了线性增长的重启间隔来保证完备性。本文用  $r$  来表示自从上次重启以来遇到的冲突次数， $n$  为执行过的冷重启次数。当  $r \geq p \times n$  时，下一次的热重启就会被替换为冷重启，其中  $p$  是一个参数。本文采用当代求解器中常用于衡量时间间隔的方法（冲突次数）作为执行冷重启的依据，这种好处在于可以使得冷重启的执行次数正相关于热重启的执行次数<sup>2</sup>。

**参数调节** 冷重启的实验中，只涉及一个重启间隔的超参数  $p$ 。对于每一个实例集和求解器， $p$  会在  $10^5$  次冲突到  $10^6$  次冲突之间选择，调参数的粒度为  $10^5$  次冲突，总共产生了 10 种参数配置结果。本文中报告了其中效果最好的参数对应的结果，并将对应参数取值列于相关表格中的“ $p$  列”，所有的结果均以科学计数法表示。此外，本文并没有针对 SAT 和 UNSAT 实例分别调参汇报结果。

<sup>1</sup>本文也实现了一个直接替换的版本。该版本的性能会严重退化，因为过多高频次的冷重启会扰乱整体的搜索逻辑，甚至会影响完备性。

<sup>2</sup>冷重启的执行次数一般比热重启的执行次数小 3、4 个数量级，一般会在求解过程中最多执行数十次，中位数为 3。

**FO 冷重启实现** 每当 FO 冷重启执行时，对于每一个变元，本文会将其对应的打分随机置为  $[0,1]$  之间的一个随机浮点数。这是一个比较小的数字，其赋值影响通常会在几次冲突之后就会被消除。此外，变元对应的排序数据结构也需要相应的做调整。对于一些没有变元打分的启发式，例如 VMTF，本文会随机的打乱优先队列中的变元顺序。

**FP 冷重启实现** 每当执行 FP 冷重启时，本文会将每一个变元的赋值倾向赋值为一个  $\{\perp, \top\}$  中的随机真值。本文也在三个基求解器上执行了相似的实验。

**FC 冷重启实现** FC 冷重启策略也比较简单：每当执行时，求解器会从求解器内部维护学习子句的数据结构中，删除掉当前求解器中不影响正确性的全部学习子句。本文在同样三个基求解器上实现了 FC 冷重启策略。

**复合冷重启策略** 通过遗忘多类学习信息，产生了以下四种组合的复合冷重启策略：

- **FO+FP**: 每次执行冷重启时，求解器会随机打乱变元序并随机重置相位。
- **FO+FC**: 每次执行冷重启时，求解器会随机打乱变元序并删除所有的学习子句。
- **FP+FC**: 每次执行冷重启时，求解器会随机重置相位并删除所有的学习子句。
- **FO+FP+FC**: 每次执行冷重启时，求解器会随机打乱变元序、随机重置相位并删除所有的学习子句。

### 3.3.2 冷重启策略的评估

本节首先介绍了相关的实验设置，并对三种遗忘单一信息的冷重启技术（表3.2）和复合遗忘策略的冷重启技术进行了评估（表3.3）。

**实验设置** 本章中的所有实验都是在具有两个 AMD EPYC 7763 CPU @ 2.45Ghz 的集群上进行的，每一台机器上都有 128 个物理核心和 1T RAM，操作系统为 Ubuntu 20.04 LTS (64 位)。本文的实验数据来自于 SAT 比赛 2020 年与 2021 年的标准测试集，简称为 SC20 与 SC21。与 SAT 比赛一样，我们的求解器会对每

个实例求解一次。为了公平，我们设置所有求解器的随机种子为 0，因此每个求解器都是确定性且可复现的。

对于每个求解器，本章报告了可以求解的满足实例（SAT）与不满足实例（UNSAT）的求解数目，标记为 #SAT 和 #UNS，以及求解的实例总数 #ALL = #SAT + #UNS。此外，所有的实例截止时间为 5000s，我们也报告了 PAR2 记分，即在规定时间内没有成功求解的实例惩罚为两倍的截止时间<sup>3</sup>。因为某些实例求解状态未知，对于 #SAT 和 #UNS 的 PAR2 计算，本文忽略了比赛中所有求解器和本文求解器均不能求解器的实例。并行求解时，一个例子最多可以使用 32 个核心（SAT 比赛求解器中最常用配置），本文会通过加速比（speedup）的计算来衡量并行算法相对于串行算法的提速效果，且加速比为并行算法领域常见的一个评价指标。如果令算法单线程的求解时间为  $t_1$ ， $n$  线程的求解时间为  $t_n$ ，则  $n$  的加速比倍数 ( $\times$ ) 可以表示为  $\frac{t_1}{t_n} \times$ 。

**基求解器** 本文选择了三个串行求解器（前三个）和两个并行求解器（后两个）作为基求解器，本章中所有实验的结果和数据均可以于 GitHub 仓库下载<sup>4</sup>。

- **MapleLCMDiscChronoBT-DL** (Version 3.0) (简记为 Maple-DL)<sup>[247]</sup>: 该求解器是 Maple 系列 SAT 求解的一个代表版本。自其第一个版本 MapleCOM-SPS<sup>[74]</sup> 以来，该系列求解器在 SAT 比赛中共获得了 4 次冠军。
- **CaDiCaL\_watch\_sat** (简记为 CaDiCaLWS)<sup>[248]</sup>: 它是一个基于 CaDiCaL<sup>[78]</sup> 改进的求解器。该提高版本获得了 SAT 比赛 2021 年与 2022 年 CaDiCaL Hack 赛道的冠军。
- **kissat-MAB**<sup>[227]</sup>: 他是 SAT 比赛 2021 年主赛道的冠军，并基于国际 SAT 比赛 2020 年主赛道的冠军 kissat<sup>[50]</sup> 研制。
- **PaKis**<sup>[155]</sup>: 该求解器是一个基于 kissat<sup>[50]</sup> 研制的并行策略组合算法求解器，并在 SAT 比赛 2021 年并行主赛道求解出了最多的可满足实例。
- **P-mcomsps**<sup>[151]</sup>: PaInLeSS 系列求解器已经在 SAT 比赛并行主赛道中赢得了 2018 年、2020 年和 2021 年的冠军。P-mcomsps 是一个最新的基于 PaInLeSS 开发的求解器版本，且同样也基于策略组合算法。

根据表3.2中针对 FO 冷重启的实验，可以了解到阶段性的变元重置能提高

---

<sup>3</sup>国际 SAT 比赛同样的评价依据

<sup>4</sup><https://github.com/CDCL-cold-restart/cold-restart>

**表 3.2 遗忘顺序 (FO)、遗忘相位 (FP) 和遗忘子句 (FC) 冷重启的实验结果****Table 3.2 Results on the Forgetting Order (FO), Forgetting Phase (FP), and Forgetting Clauses (FC) cold restart policies.**

求解器	#SAT	PAR2	#UNS	PAR2	#ALL	PAR2	$p$
SAT Competition 2020 (共 400 个实例)							
Maple-DL	107	5043.39	113	3095.50	220	5078.55	-
Maple-DL +FO	113(+6)	4846.62	111(-2)	3175.93	224(+4)	5014.36	$1 \times 10^6$
Maple-DL +FP	114(+7)	4731.01	111(-2)	3179.28	225(+5)	4960.39	$8 \times 10^5$
Maple-DL +FC	114(+7)	4869.95	105(-8)	3620.68	219(-1)	5190.05	$1 \times 10^6$
CaDiCaLWS	123	4247.01	120	2853.88	243	4608.88	-
CaDiCaLWS +FO	126(+3)	3998.46	123(+3)	2704.77	249(+6)	4435.03	$1 \times 10^6$
CaDiCaLWS +FP	130(+7)	3891.82	122(+2)	2796.81	252(+9)	4418.16	$3 \times 10^5$
CaDiCaLWS +FC	125(+2)	4110.95	112(-8)	3298.22	237(-6)	4708.32	$6 \times 10^5$
kissat-MAB	151	2510.49	121	2557.91	272	3670.18	-
kissat-MAB +FO	159(+8)	2226.81	122(+1)	2515.19	281(+9)	3518.92	$8 \times 10^5$
kissat-MAB +FP	156(+5)	2363.07	122(+1)	2591.20	278(+6)	3614.02	$1 \times 10^6$
kissat-MAB +FC	158(+7)	2278.36	109(-12)	3421.98	267(-5)	3879.05	$7 \times 10^5$
SAT Competition 2021 (共 400 个实例)							
Maple-DL	115	3458.14	143	3076.04	258	4163.22	-
Maple-DL +FO	120(+5)	3270.75	146(+3)	3067.84	266(+8)	4084.84	$1 \times 10^6$
Maple-DL +FP	118(+3)	3279.54	144(+1)	3054.42	262(+4)	4082.12	$1 \times 10^6$
Maple-DL +FC	115	3520.02	132(-11)	3637.65	247(-11)	4449.51	$1 \times 10^6$
CaDiCaLWS	126	2667.20	144	2999.09	270	3811.46	-
CaDiCaLWS +FO	132(+6)	2483.37	144	2994.74	276(+6)	3735.90	$8 \times 10^5$
CaDiCaLWS +FP	132(+6)	2501.37	143(-1)	3049.62	275(+5)	3768.62	$3 \times 10^5$
CaDiCaLWS +FC	130(+4)	2557.67	138(-6)	3251.08	268(-2)	3884.82	$6 \times 10^5$
kissat-MAB	142	1654.57	147	2714.55	289	3274.09	-
kissat-MAB +FO	143(+1)	1624.87	152(+5)	2555.97	295(+6)	3188.47	$4 \times 10^5$
kissat-MAB +FP	142	1655.81	150(+3)	2634.91	292(+3)	3237.56	$1 \times 10^6$
kissat-MAB +FC	143(+1)	1677.60	137(-10)	3339.01	280(-9)	3573.68	$7 \times 10^5$

整体的求解性能，尤其是对于 SAT 实例。具体地，通过采用 FO 策略，三个基求解器 #ALL 在 SC20 和 SC21 数据集合上平均多求解出了 6.3 和 6.7 个例子；#SAT 对应的数据分别为 5.7 和 4.0。其在 SAT 实例上的性能提高主要依赖于初始变元顺序不同时导致的巨大求解时间差异。

同时，我们也观察到 FO 对于 #UNS 也有一定的改善，尤其是对于 kissat-MAB。一个合理的解释是，重启能帮助求解器找到一个能产生更高质量学习子句的新路径<sup>[153]</sup>。

**观察 2：**忘记变元分支顺序的冷重启能同时帮助改善 SAT 和 UNSAT 实例上的求解能力。

根据表3.2中针对 FP 的实验观察，FP 冷重启能帮助改善 CDCL 的整体求解性能，在三个求解器上，#ALL 的求解数量分别平均多求解了 4.5、7.0 和 4.5 个。且 FP 的性能提高更依赖于 #SAT 上的提高。

**观察 3：**忘记变元相位的冷重启能帮助改善 SAT 实例的性能，但是对于 UNSAT 实例的提高有限，甚至会变差。

根据表3.2中 FC 相关的实验，遗忘掉所有的学习子句对于 UNSAT 实例的求解器永远是不好的。但是执行 FC 冷重启，有时会帮助求解器提高 SAT 实例上的求解能力。事实上，FC 对本文采用的三个基求解器均有帮助。令人惊喜的是，FC 帮助 MapleLCMDiscChronoBT-DL 和 kissat-MAB 在 SC20 上均多求解出了 7 个 SAT 实例。

当代 CDCL 有一个子句管理的组件，会定期的删除一部分质量较差的子句来减少求解器推理的成本。本文的实验证明了定期删除一部分的学习子句能帮助改进 SAT 实例的求解性能，也激励我们更深入的研究学习子句在可满足实例求解上的作用。后续 3.3.3 小节将会对该问题进一步深入讨论。

**观察 4：**遗忘所有学习子句的冷重启总是对不可满足实例的性能有负面影响，但是它通常（不总是）对可满足实例的求解性能有帮助。且冷重启时保留更多质量较差的子句对 UNSAT 求解有益。

表3.3汇报了复合冷重启的实验结果，给出了每个求解器的最优复合配置和对应的求解性能变化情况。在 SC20 和 SC21 实例集上，对于每一个求解器，本文分别在  $\Delta_{SAT}$ 、 $\Delta_{UNS}$  和  $\Delta_{ALL}$  三列上报告了其在 SAT、UNSAT 和 ALL 上最好的求解效果的变化，和对应的信息遗忘策略（‘BConf’）。表格中汇报的策略为

表 3.3 遗忘多类信息的冷重启策略最优配置

Table 3.3 Results of the best configuration for multi-type information forgetting cold restart

求解器	BConf	$\Delta_{SAT}$	$p$	BConf	$\Delta_{UNS}$	$p$	BConf	$\Delta_{ALL}$	$p$
SAT Competition 2020 (#400)									
Maple-DL	+FO+FC	+8	$1 \times 10^6$	+FO+FP	-1	$4 \times 10^5$	+FP	+5	$8 \times 10^5$
CaDiCaLWS	+FO+FC	+12	$8 \times 10^5$	+FO	+3	$1 \times 10^6$	+FP	+9	$3 \times 10^5$
kissat-MAB	+FO+FC	+10	$4 \times 10^5$	+FO	+1	$8 \times 10^5$	+FO	+9	$8 \times 10^5$
SAT Competition 2021 (#400)									
Maple-DL	+FO	+5	$1 \times 10^6$	+FO	+3	$1 \times 10^6$	+FO	+8	$1 \times 10^6$
CaDiCaLWS	+FO+FC	+6	$1 \times 10^6$	+FO+FP	+1	$1 \times 10^6$	+FO	+6	$8 \times 10^5$
kissat-MAB	+FO+FC	+3	$4 \times 10^5$	+FO	+5	$4 \times 10^5$	+FO	+6	$4 \times 10^5$

可以求解出最多数目的复合策略，若多个，则选择 PAR2 最小的。

实验结果表明，对于 SAT 和 UNSAT 实例，各自存在对应的最优配置。

**观察 5：**FO+FC 冷重启策略下能达到最优的 SAT 实例求解能力，单独的 FO 冷重启策略能达到最优的 UNSAT 实例求解和整体 (SAT+UNSAT) 求解性能。

### 3.3.3 冷重启策略的进一步分析

根据上面一个小节的实验结果，我们可以总结出综合性能最佳的冷重启策略为 FO 冷重启技术，这促使我们对其中的细粒度实验数据产生了好奇；另外，FC 相关的研究与 CDCL 中子句管理的思想类似，也引起了我们对不彻底的 FC 策略（部分冷重启）和学习子句的作用的研究的好奇心；最后，除了重启之外，另外一种有效利用求解时间差异的方法为并行技术，我们也在本节进行深入的探讨。

#### 3.3.3.1 细分实例求解性能分析

从之前的结果中，我们了解到 FO 冷重启有最优的综合性能，这使我们对其在细分类用例集合上的表现产生了兴趣。本文将 SC20 和 SC21 上的实例按照来源做了分类。本文对细分类实例上的实验结果进行了总结，因为空间受限，更详细的实验结果在本文的 GitHub 仓库中展示。结果中 #k 表示该细分类中实例的数量。

- 冷重启更适用于“hypertree 分解” (#14)、“滑动瓷砖拼图” (#13)、一些如“最小 super-permutation 问题” (#13) 等旅行商问题和一些染色问题。FO 可以帮

助基求解器在这些家族中至少多求解出2个例子。

- 另一方面，FO在“原像攻击密码学问题”(#11)用例上表现较差，会少求解2个例子。

• 对于一些类型的例子来说，FO的引入会使得基求解器产生一定的互补性，例如kissat-MAB和kissat-MAB+FO在“电路乘法”(#13)这组例子都可以求解8个例子，但其中4个有互补性。这也启示我们，利用不同冷重启变元顺序之间显著的互补性，有潜力据此设计一个更进一步的算法。实际上，本章后续章节利用了该特点，研究了并行冷重启方法。

### 3.3.3.2 学习子句作用的进一步分析

**遗忘学习子句的LBD阈值选择** 为了更好的理解学习子句在CDCL求解器中作用，本文对FC冷重启做了更多的补充实验。根据实验结果，我们能知道在冷重启时应该选择哪些子句去遗忘。本文为FC冷重启设置了不同的阈值，用于决定哪些学习子句应该被遗忘。在提出了著名的三层子句管理启策略的研究中，Chanseok Oh指出 $LBD > 5$ 的学习子句在求解过程中的帮助不大<sup>[43]</sup>。为了保留一部分被证明有用的学习子句，本文将遗忘的规则设置为从 $LBD > 0$ (FC的默认版本)到 $LBD > 5$ <sup>5</sup>。本文采用kissat-MAB当作基求解器，并在SC20和SC21的实例集上进行了实验，且相关参数 $p$ 均设置为与默认FC相同。

实验结果被总结于表3.4中，根据实验结果，可以得到如下的结论：

- 遗忘更多的学习子句通常意味着更好的可满足实例求解能力和更差的不可满足实例的求解能力。
- 保留 $LBD \leq 3$ 的学习子句能帮助求解器达到一个最均衡的求解性能，这也为Oh的学习子句管理方法中“将 $LBD=3$ 设置为核心子句和中间层子句的界限”<sup>[43]</sup>提供了实验数据支撑。

**学习子句利用效率的研究** 本文也计算了不同LBD层学习子句的利用率。具体地，对于LBD值属于2~7的学习子句，本文分别记录了其在单元传播和冲突分析中的利用情况。当一条学习子句参与到另一条学习子句的产生时则认为它参

<sup>5</sup>当学习子句产生时会立刻计算其LBD值，如果LBD为1则其为单元子句，会立刻被用于固定变元赋值。因此该类子句不会被存储在学习子句库中。因此，如果没有其他因素，学习子句库中不会有LBD小于2的子句。但是，由于搜索过程中学习子句有可能会因为被化简等原因，从而导致LBD被重新计算，所以在FC删除的时候有可能会遇到LBD小于2的学习子句。

**表 3.4 不同 LBD 阈值标准下的 FC 实验结果****Table 3.4 FC versions that forget learnt clauses according to different LBD thresholds.**

遗忘阈值	#SAT	PAR2	#UNS	PAR2	#ALL	PAR2
SAT Competition 2020 (#400)						
LBD > 0	158	2278.36	109	3421.98	267	3879.05
LBD > 1	155	2385.65	115	3082.32	270	2689.80
LBD > 2	153	2428.49	120	2622.74	273	2513.30
LBD > 3	154	2405.37	120	2578.99	274	2481.17
LBD > 4	153	2447.50	122	2494.85	275	2668.17
LBD > 5	153	2495.89	121	2542.93	274	2516.38
SAT Competition 2021 (#400)						
LBD > 0	143	1677.60	137	3339.01	280	3573.68
LBD > 1	144	1626.29	141	3072.62	285	2403.80
LBD > 2	142	1708.98	142	2924.58	284	2362.45
LBD > 3	142	1742.83	151	2636.28	293	2223.13
LBD > 4	140	1748.33	149	2669.04	289	2243.28
LBD > 5	138	1840.09	150	2643.33	288	2271.89

**表 3.5 不同 LBD 层级的学习子句的使用率标准化统计数据****Table 3.5 Normalized statistics of the usage of learning clauses with different LBD values.**

类型	LBD=2	LBD=3	LBD=4	LBD=5	LBD=6	LBD=7
冲突	1.0	0.2641	0.1887	0.1381	0.1004	0.0670
传播	1.0	0.1899	0.1289	0.0833	0.0447	0.0215

与到冲突分析一次；当一条学习子句被直接用于推出某变元赋值时，则认为它参与到了单元传播一次。为了方便更好的数据统计，学习子句的 LBD 层级为其产生时的 LBD。

详细数据被呈现在表 3.5 中，且数据通过同时除以“LBD=2”列对应的数据进行了标准化。从数据中，本文发现在“LBD=2”和“LBD=3”两列之间变化差异巨大，其他列之间的数据的差异变化相对较小。

### 3.3.3.3 并行 FO 策略

本文注意到，在某些实例上，冷重启的引入会使其与基求解器之间产生一定的互补性能。这是因为冷重启有可能会中断一些求解时间占据较久的分支，虽然这个分支被评估较难，但是实际上有可能该分支已经为一个距离真正解比较近的分支。这促使本文使用并行技术来更好的利用互补性能的优势。本节选取综合

表3.6 并行FO在两个主流并行SAT求解器上的实验结果

Table 3.6 Experiment results of parallel forgetting order on two SOTA parallel solvers.

求解器	#SAT	PAR2	#UNS	PAR2	#Solved	PAR2
SAT Competition 2020 (#400)						
P-mcomsps (p)	158	2169.01	140	843.34	298	2872.74
P-mcomsps +FO (p)	160(+2)	2112.08	141(+1)	813.09	301(+3)	2834.36
PaKis (p)	176	1005.26	130	1801.21	306	2671.46
PaKis +FO (p)	181(+5)	800.08	130	1776.44	311(+5)	2564.32
SAT Competition 2021 (#400)						
P-mcomsps (p)	143	1332.89	179	725.23	322	2220.39
P-mcomsps +FO (p)	149(+6)	1002.65	178(-1)	778.81	327(+5)	2113.21
PaKis (p)	155	472.05	164	1705.67	319	2331.96
PaKis +FO (p)	158(+3)	173.68	166(+2)	1678.34	324(+5)	2249.03

性能最好的FO冷重启作为案例，介绍了一种并行的冷重启技术，并用来改进主流求解器的性能。

**实现** 并行的FO技术与串行的版本实现方法一致，区别在于，并行的FO不同的线程上使用了不同的随机种子。

**实验评估** 本文在两个顶尖的并行SAT求解器(PaKis<sup>6</sup>和P-mcomsps)中实现了本文的并行FO冷重启技术。实验结果在表3.6中列出，其中(p)的尾缀表示求解器为一个并行版本的求解器。PaKis+FO(p)和P-mcomsps+FO(p)的冷重启的参数p分别设置为 $6\times10^5$ 和 $4\times10^5$ 。结果表明：FO能提高两者的求解性能，尤其是#SAT的求解能力。具体地，P-mcomsps+FO(p)帮助P-mcomsps在可满足实例上平均提高了13.7%的PAR2分数；PaKis+FO(p)帮助PaKis在可满足实例上平均提高了41.81%的PAR2分数。

**加速比分析** 为了避免其他技术的影响，更好的衡量并行FO的平均加速比<sup>7</sup>，本文研制了一个并行的SAT求解器，kissat-MAB+FO。该求解器是直接在一个

<sup>6</sup>本文将PaKis的串行CDCL求解器由kissat<sup>[50]</sup>更改为一个更好的版本kissat-MAB<sup>[227]</sup>，两个版本实验结果相当。

<sup>7</sup>表中汇报的加速比只计算了单线程和C线程均可以在规定时间内求解的例子，并过滤掉了太过简单的例子（即两个求解器都可以在1s内求解的例子）以避免对平均值影响过大。另外一种计算平均加速比的方法为比较PAR2值的变化，在这种统计方法下，未被求解的例子和对应的惩罚也会被考虑在内。

表 3.7 并行 FO 的加速比分析

Table 3.7 Speedup analysis for the parallel FO

C	#SAT(PAR2)	#UNS(PAR2)	#Solved(PAR2)	Average Speedup		
				SAT	UNS	ALL
kissat-MAB +FO(p)						
1	151(2510)	121(2558)	272(3670)	1.0	1.0	1.0
2	161(1986)	123(2440)	284(3376)	4.2	1.13	2.8
4	168(1477)	123(2404)	291(3120)	12.5	1.16	7.5
8	176(1190)	125(2317)	301(2951)	10.0	1.18	6.1
16	177(1024)	124(2320)	301(2872)	15.0	1.22	8.9
32	182(766)	125(2209)	307(2707)	26.8	1.32	15.5
64	183(667)	125(2231)	308(2668)	27.5	1.35	16.0
kissat-MAB +FO(p) + Sharing (LBD≤2)						
1	151(2510)	121(2558)	272(3670)	1.0	1.0	1.0
2	162(1958)	126(2159)	288(3259)	3.5	1.3	2.5
4	167(1618)	127(1984)	294(3032)	7.5	1.9	5.0
8	171(1329)	129(1723)	300(2797)	9.4	2.9	6.5
16	178(924)	133(1436)	311(2498)	17.0	4.6	11.5
32	180(770)	131(1493)	311(2445)	22.1	6.6	15.4
64	184(583)	133(1354)	317(2304)	29.0	9.4	20.5
kissat-MAB +FO(p) + Sharing (LBD≤3)						
1	151(2510)	121(2558)	272(3670)	1.0	1.0	1.0
2	163(1859)	128(1987)	291(3148)	2.6	1.6	2.2
4	167(1603)	130(1785)	297(2951)	5.8	2.2	4.2
8	173(1197)	131(1556)	304(2672)	10.2	3.4	7.2
16	178(911)	134(1316)	312(2447)	20.7	5.2	13.9
32	177(948)	135(1169)	312(2410)	20.0	8.0	14.7
64	182(662)	140(891)	322(2171)	35.5	11.0	24.8

串行求解器 kissat-MAB 的基础上直接采用了并行 FO 技术研制。本文计算了从单 CPU 核心数目到 64 个 CPU 核心数目下求解器的性能变化情况，相关的结果被汇总于表3.7。表中汇报了在 SC20 实例集合上，不同线程数目 (C) 下的可满足 (SAT)、不可满足 (UNSAT)、全部 (ALL) 实例的求解数量、PAR2 打分和平均加速比。我们可以发现，求解器求解出的实例数量跟 CPU 的核心数量的增加呈正相关。且采用该方法的求解器可以在 SAT 实例上取得很好的性能，甚至可以超过同等 CPU 核心数量下的主流求解器，但是该方法对于 UNSAT 实例的提升有限。

**并行 FO 共享子句选择** 流行的并行 SAT 求解器通常采用子句共享<sup>[153]</sup> 在线程之间交换学习到的子句。在多个线程之间“共享学习到的子句”的想法类似于当前热重启中“保留高质量的学习子句”的想法。这引起了我们对研究“FC 中学习子句阈值规则是否也适用于并行子句共享”的兴趣。

根据表 3.5 中的结果， $LBD \leq 3$  的学习子句具有最高的成本效益。本文通过在 PaInLeSS 框架的帮助下实现了子句共享技术，并简单地共享  $LBD \leq 2$  和  $LBD \leq 3$  的学习子句，扩展了一个带有子句共享功能的并行 FO 算法<sup>8</sup>。在本文的子句共享方法中，对于每个线程，在 0.75 秒的持续时间内最多可以共享 1500 个文字。

根据表 3.7 中的实验结果和加速比统计效率，子句共享可以进一步提高 UNSAT 实例的性能，但对 SAT 实例几乎没有影响，甚至在某些实例上会产生阻碍。因此，基于并行 FO 和简单的子句共享策略足以开发一个以可满足性求解为导向的高性能并行求解器。事实上，基于本文相关技术研制的 SAT 并行求解器 Parkissat-RS，拓展了预处理求解技术，在 SAT 比赛 2022 年并行主赛道中以领先亚军 1/4 性能的成绩取得了冠军，其改进版本 PRS 蝉联了 SAT 比赛 2023 年并行主赛道的冠军。

**观察 6：FO 冷重启对并行求解是有益的，这对 SAT 实例来说很重要，对 UNSAT 实例来说则不那么重要。子句共享对 UNSAT 实例有显著的正向影响，而对 SAT 实例几乎没有影响。**

**并行 FO 与前人多样性并行方法的对比讨论** 客观上，冷重启并行方法可以被看作是一种基于多样性的并行方法。与之前的多样性方法一样，本文的并行 FO 方法利用了线程之间的互补性。以前的方法通常利用 CDCL 求解器中策略的不同配置之间的互补性<sup>[151-155]</sup>。相比之下，平行 FO 方法的互补性主要来自定期切换到搜索空间的不同起点。并行 FO 是一种轻量级的、易于实现的方法。开发人员不需要详细了解 CDCL 的策略、代码之后才可以选择配置，只需应用具有子句共享的相关并行 FO 框架，即可实现主流并行 SAT 求解器的性能。并行 FO 的另一个优点是它的强适应性。它具有良好的可扩展性，因为它可以应用于任何给定数量的 CPU 核心数而无需更改代码。根据本文的实验结果，加速比具有随着内

<sup>8</sup>PaInLeSS 框架可以根据共享的文字动态调整阈值，本文对该版本的子句共享策略进行了测试，根据实验结果，我们发现其与共享  $LBD \leq 2$  子句的版本具有相似的性能，并且可以得出类似的结论。

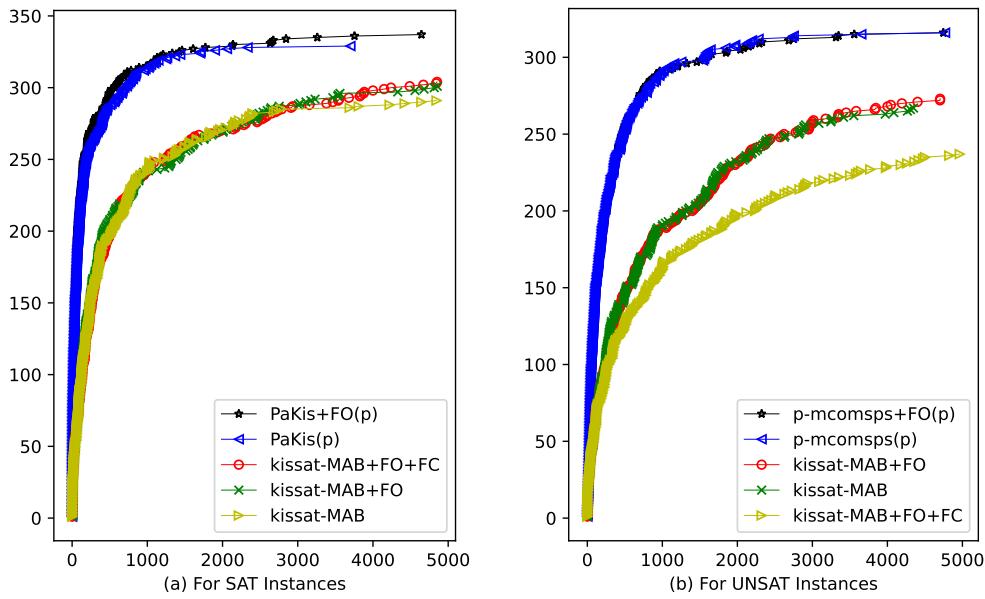


图 3.2 串并行冷重启技术与对比求解器的 CDF 图

Figure 3.2 CDF for comparing our best sequential and parallel cold restart solvers.

核数量的进一步增加而进一步增长的潜力，即 64 核心数并没有达到加速比增长的上限。

**方法有效性总结** 为了了解本文的方法如何推动最新技术发展，本文在图 3.2 中将本文最好的串行和并行求解器与表现最佳的竞争对手求解器进行了比较。本文绘制了 CDF (Cumulative Distribution Function) 图，x 轴表示求解时间，y 轴表示求解的数目。曲线越高求解能力越强。从图中可以看出，串行和并行版本的冷重启主要有助于推动 CDCL 求解器在可满足实例上的性能。对于串行求解器，kissat-MAB +FO+FC 在 SAT 实例上的表现最佳，但在 UNSAT 实例上有所下降，而 kissat-MAB +FO 在 UNSAT 实例上的表现最佳（观察结果 5）。对于并行版本，并行 FO 有助于提高可满足求解能力的性能和效率，同时对不可满足性能产生轻微影响（观察 6）。

### 3.4 本章小结

现代 CDCL 求解器采用热重启策略，在重启之间保留所有搜索信息。本文实证研究了在重启之间是否应该保留这些信息的问题，得出了几个有意义的观察结果，以及几种冷重启策略（在冷重启之间忘记一些信息）。实验表明，定期

执行冷重启对串行和并行的 CDCL 求解器都有帮助，特别是对于可满足的实例。这项工作为 CDCL 求解器提供了一个有趣的方向。在未来，本人打算设计更好的具有动态间隔的冷重启策略。在真实世界的基准测试中，即使有集成了冷重启技术，当前的 CDCL 求解器仍然存在严重的重尾分布现象，亟需未来在实验和理论上进行改进。同时，本文也可以被视为通过重新审视重启策略来开发面向可满足的 SAT 求解器的一个成功尝试。最后，随着最近大语言模型的流行，我们在未来也将开展使用大语言模型辅助求解器设计的工作。



## 第4章 SAT问题混合求解算法

第3章介绍了SAT求解器重启组件冷重启与热重启混合策略研究，本章的工作则主要集中于CDCL与局部搜索间的混合策略研究。4.1节中给出了学者们在混合策略上的尝试并介绍了本章的研究动机。4.2节介绍了本文的主要贡献—基于松弛CDCL的混合求解框架，并介绍了局部搜索算法的调用时机。4.3节进一步介绍了从局部搜索中抽取有效信息的技术，包括利用高一致性候选解改进相位选择启发式和利用变元冲突频率改进分支启发式两种技术。

本章的混合技术实现于glucose、Maple和kissat求解器上，并在包含400个实例的SAT比赛(2020年)的实例集上多求解了62、67和10个实例，且在频谱分配问题上也表现出了巨大的提升。法国SAT领域著名专家Audemard曾表明，在竞赛测试集上能帮助求解器多求解出10个实例的方法即可代表一个新的特性<sup>1</sup>，这印证了我们方法的高效性。

据我们所知，本章中提到的工作首次在通用测试集合上，成功解决了1997年Bart Selman等人提出的命题逻辑十大挑战的第七项，其内容为：“设计一个系统搜索与局部搜索的混合算法，其效果能超过混合前两种算法的最优性能”<sup>2</sup>。相关工作取得了SAT比赛2020年主赛道的一枚金牌，并获得了国际SAT会议2021年的“最佳论文奖”，这也是到目前(2024年6月)为止，国内唯一取得此荣誉的工作。

### 4.1 引言

人们一直对将系统搜索和局部搜索结合起来解决SAT问题感兴趣。事实上，1997年，Selman等人将其列为命题逻辑的一项挑战<sup>[249]</sup>。根据主体求解器的类型(DPLL/CDCL或局部搜索)，之前的尝试可分为两大类。

第一类混合求解器使用局部搜索求解器作为主体求解器。不完备混合求解器hybridGM<sup>[232]</sup>在遇到只有一个未满足子句的局部最优时才会调用CDCL搜

<sup>1</sup>To give an idea, improving a solver by solving at least ten more instances (on a fixed set of benchmarks of a competition) is generally showing a critical new feature<sup>[84]</sup>. (Audemard and Simon, 2012)

<sup>2</sup>Demonstrate the successful combination of stochastic search and systematic search techniques, by the creation of a new algorithm that outperforms the best previous examples of both approaches. (Selman et al., 1997)

索。Audemard 等人提出了一种名为 SATHYS 的混合求解器<sup>[111,250]</sup>，在每次局部搜索求解器达到局部最优时，都会启动 CDCL 求解器。CDCL 求解器的一些推理技术或信息已被用于改进局部搜索求解器。例如，归结技术被集成到局部搜索求解器中<sup>[251,252]</sup>；近期，Lorenz 和 Wörz 开发了一种混合求解器 GapSAT<sup>[253]</sup>，该求解器在运行局部搜索求解器 ProbSAT 之前使用 CDCL 求解器作为预处理器。实验表明，CDCL 求解器生成的学习子句对随机实例上的局部搜索求解器有很好的改进作用。

另一类混合求解器侧重于通过局部搜索来增强 CDCL 求解器，本文的工作属于这一系列。将局部搜索和 CDCL 混合的一种简单方法是在 CDCL 开始之前调用局部搜索，尝试求解实例或导出一些搜索信息，如 CDCL 中使用的变量分支顺序。混合求解器 Sparrow2Riss<sup>[254]</sup>、CCAnr+glucose<sup>[255]</sup> 和 SGSeq<sup>[256]</sup> 属于这个类别。

一些工作使用局部搜索来寻找 CDCL 的子公式来求解原始问题。早期的一些工作<sup>[112]</sup> 使用局部搜索求解器来找到公式中可满足的部分，这有助于将公式分为两部分，供 DPLL 求解器使用。在 HINOTOS<sup>[231]</sup> 中，使用局部搜索来以增量方式识别要传递给 CDCL 求解器的子句集。

与本文最相似的工作为在 CDCL 搜索中调用局部搜索求解器的一类工作。WalkSatz<sup>[230]</sup> 在 DPLL 求解器 Satz 的每个叶子节点上调用局部搜索求解器 WalkSAT。然而，这很耗时。这可以通过共享内存并行完成<sup>[257]</sup>。在 CaDiCaL 和 kissat<sup>[50]</sup> 中，当求解器会定期在相位重置时调用局部搜索求解器，并在局部搜索结束后立刻使用局部搜索产生的最优解更新一次相位信息。然而，CaDiCaL 和 kissat 使用局部搜索解的方式与我们基于局部搜索的相位重置方法不同。CaDiCaL 和 kissat 只记录当前的局部搜索解，且仅在局部搜索退出后使用一次，且它们不会使用之前局部搜索过程中的信息。事实上，我们还进行了补充实验，以了解局部搜索对 kissat\_sat 性能的影响，结果证明局部搜索对 kissat 性能的影响是有限的。当 kissat\_sat 在没有局部搜索的情况下工作时，在 SAT 比赛测试集上 #SAT 指标平均只下降了 5。

尽管之前很多工作已经尝试过将 CDCL 和局部搜索的优势结合起来，但它们并没有产生一个在应用实例集合上本质上优于 CDCL 求解器的混合求解器。这项工作首次在 SAT 竞赛的标准测试集合上达到了“创造一种优于这两种方法

最佳基准求解器之和的新混合算法”这一挑战的标准<sup>[249]</sup>。

本章致力于研究一种更加深度结合的混合算法，并提出了一种以 CDCL 为主体，以局部搜索为辅助工具的算法。为此，我们提出了三个关键技术点。第一个关键技术研究了局部搜索插入 CDCL 中的时机和 CDCL 应该提供给局部搜索什么信息，后两个关键技术则是研究如何从局部搜索搜索过程中抽取关键信息来改进 CDCL 的性能。三个技术点总结如下：

- 利用局部搜索探索 CDCL 希望分支（第 4.2 节）

第一个创新点是一种将局部搜索插入到 CDCL 中的创新方法。当遇到一个 CDCL 的（高一致性）希望分支时，我们会对 CDCL 的回溯过程执行松弛操作，进入一种无回退的布尔约束传播阶段，并忽略掉过程中遇到的所有冲突，直到所有的变元都得到赋值为止。通过该过程得到的完全赋值会作为初始解传递给局部搜索，并在附近搜索空间内搜索采样。如果局部搜索不能在规定时间内找到解，算法则会回退到进入松弛阶段前的状态，从断点处继续 CDCL 的搜索过程。

- 基于局部搜索解的相位选择算法（第 4.3.1 节）

相位指的是变元的赋值倾向，而相位选择技术则指的是变元分支选择时负责真值选择的启发式。主流的 CDCL 算法实现了一种相位选择启发式，名为相位保存 (phase saving)<sup>[75]</sup>，该技术会优先将变元赋值为最近一次的赋值选择。本章工作之前最新的相关工作为一种基于靶相位 (target phase) 的相位重置 (rephase) 技术<sup>[50]</sup>，它会定期的将相位按固定顺序重置为靶相位、随机相位、全 0/1、翻转相位等相位。本章则提出了一种新型的相位重置技术，主要基于局部搜索收集的高质量候选解来执行相位重置。

- 基于局部搜索冲突频率信息的变元分支策略（第 4.3.2 节）

本章使用局部搜索中搜集的变元冲突频率 (conflict frequency) 信息，即变元出现在不满足子句中的频率，来提高 CDCL 算法中的变元分支启发式。具体的，我们采用该技术来改进主流 CDCL 中 VSIDS 变元分支启发式中变元的活跃度打分和 LRB 变元分支启发式中变元的学习率。

## 4.2 (回退) 松弛 CDCL 混合求解框架

本节中，我们将介绍 CDCL 中插入局部搜索的时机和方法。该方法通过局部搜索探索希望分支，能帮助 CDCL 更快的找到解。

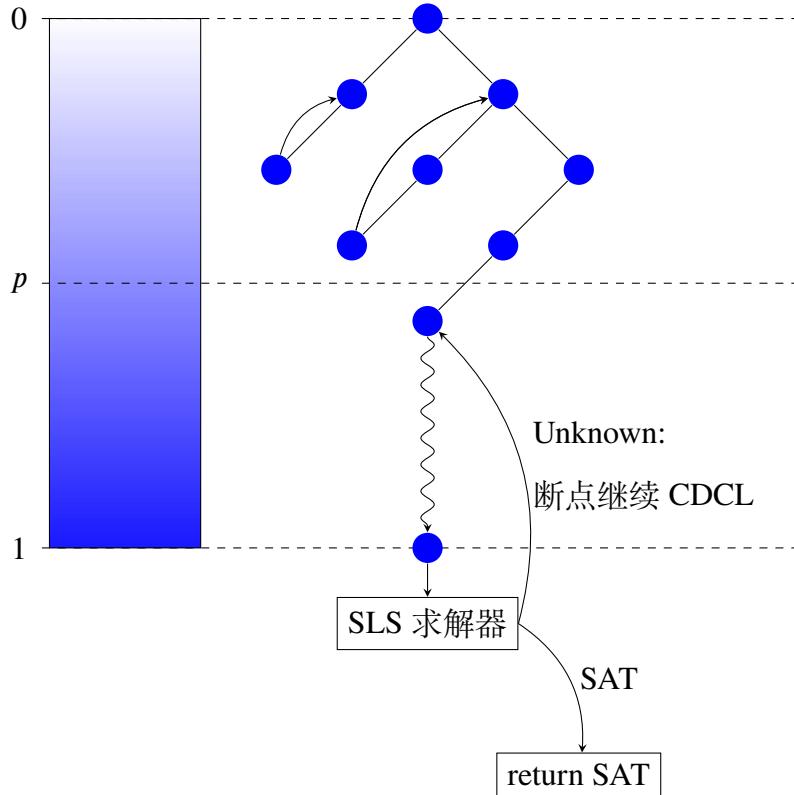


图 4.1 CDCL 松弛示意图

Figure 4.1 Overall procedure of relaxed CDCL

首先，我们先介绍本方法的动机。通过使用推理技术，CDCL 求解器能够对 CDCL 二叉搜索树的大部分分支进行剪枝，这对于解决不可满足的实例非常有用—要证明一个公式是不可满足的，CDCL 求解器需要检查整个搜索空间，因此搜索树剪枝越多，求解器的效率就越高。然而，在求解可满足的公式时，一些接近可满足赋值的有希望的分支也会因为被剪枝而没有进行深入的探索，这将使 CDCL 求解器错过一些寻找解的机会。因此，对有希望的分支的探索可能会改进 CDCL 求解器求解可满足实例的能力，而这样做的一个自然方法是：在这些分支上调用局部搜索算法。

我们提出了一种在 CDCL 求解器的搜索过程中探索希望分支的方法，该方法可以提高求解器寻解能力，同时保持求解器的完备性。这种方法的关键点在于如何确定哪些分支（即变元的部分赋值）值得探索，并称该类分支为**希望分支**。

为此，我们提出了下述判定希望分支的两个条件，当 CDCL 当前维护的部分赋值  $\alpha$  满足至少其中一个条件时，算法即认为当前的分支是一个值得探索的希望分支。

- $\frac{|\alpha|}{|V|} > p$  并且当前的部分赋值  $\alpha$  下没有任何冲突, 其中  $|V|$  为当前公式变元的数量,  $p$  为一个参数, 且根据在 SAT 比赛测试集合上进行的预实验,  $p$  设置为 0.4 时效果最好。
- $\frac{|\alpha|}{|\alpha\_max|} > q$  并且当前的部分赋值  $\alpha$  下没有任何冲突, 其中  $q$  为一个参数, 且按照类似的方法设置为 0.9。 $\alpha\_max$  代表 CDCL 历史上曾到达过的一个无冲突的部分赋值, 该部分赋值中变元数目最多,  $|\alpha\_max|$  即为该变元数。

上述第一个条件衡量了当前部分赋值的一致性, 第二个条件衡量了当前赋值和历史上最好的赋值间的一致性差异。CDCL 的搜索过程会保持一定的搜索顺序, 在一段连续的时间片段内搜索的区域差异不会太大。因此为了防止连续的两次 SLS 调用进入的搜索空间过于相似, 我们设置了一个过滤条件, 即如果距离上次局部搜索的调用没有超过  $k$  次重启, 算法则会禁止 CDCL 调用局部搜索, 类似地,  $k$  通过预实验被设置为 400。

介绍了上述希望分支的条件之后, 我们可以对 CDCL 调度局部搜索的方法进行描述。如图 4.1 所示, 在 CDCL 搜索期间, 每当到达有希望的部分分支对应的节点时, CDCL 则会进入一种无回溯的松弛模式。该模式下 CDCL 会不断的调用 BCP 算法来赋值剩余的变量, 并忽略掉赋值过程中遇到的所有冲突, 即不执行冲突分析和回退的操作。通过该方法, 我们最终可以得到一组变元的完全赋值  $\beta$ ,  $\beta$  则可以当作初始解传递给局部搜索求解器以在  $\beta$  附近搜索公式的解。如果局部搜索能找到一组解, 则返回 ‘SAT’。如果局部搜索未能在一定的时间预算内找到模型, 则该算法会返回到中断的位置 (我们称之为断点), 继续执行 CDCL 搜索。因为无回溯阶段不会改变 CDCL 搜索过程的任何数据结构, 局部搜索的插入不会影响 CDCL 部分算法的完备性, 故该混合算法仍然可以保持 CDCL 的完备性。在这项工作中, 为了保证可复现性, 我们没有采用局部搜索的运行时间当作截止条件, 而是采用局部搜索中核心数据结构的内存访问次数作为截断条件 (设置为:  $5 \times 10^7$ )。

### 4.3 基于局部搜索的 CDCL 改进策略

本节讨论了两种利用局部搜索过程中的信息来改进 CDCL 启发式的策略。

### 4.3.1 基于局部搜索解的相位重置技术

在上一小节中，我们提出了一种将局部搜索求解器插入到 CDCL 求解器中来提高求解性能的方法，而本节提出了一种基于局部搜索过程所获得的高质量候选解的相位重置启发式方法。相位是一组变元的赋值，对应了分支变元选择之后的真值赋值倾向。相位选择是 CDCL 求解器的重要组成部分。大多数现代 CDCL 求解器利用相位保存 (phase saving) 启发式算法<sup>[75]</sup>，该技术返回变元  $v$  的相位为该变元最后一次赋的真值，这种缓存方案有效减少了非时序回退造成的重复搜索和推理。本章用保存相位 (saved phase) 表示由相位保存技术记录的相位。后来，Biere 等人提出了一种相位重置 (rephase) 技术，该技术会按照一定的顺序，在靶相位、随机相位、全 0/1 相位、取反操作之间选择，这再次提高了性能<sup>[50]</sup>。其中靶相位指的是 CDCL 在搜索过程中遇到的一个无冲突部分赋值  $\alpha_{target}$ ，且自最近一次相位重置以来， $\alpha_{target}$  中的变元数量最多。

算法 4 描述了一个混合求解器的伪代码，在算法 1 的拓展了希望分支的探索和相位重置的思想。算法 4 拓展了关于公式和赋值的描述。伪代码中  $dl$  表示决策层， $bl$  表示冲突分析返回的回退层， $\alpha$  表示 CDCL 维护的变元赋值， $\beta$  表示 CDCL 松弛操作时使用的变元赋值，二元组  $(v, p)$  表示变元  $v$  与对应的真值赋值  $p$ 。每次重启 CDCL 求解器之后，该技术都会利用局部搜索产生的赋值来重置之前相位保存技术中记录的保存相位。为此，我们在每次运行局部搜索求解器时，会记录在本次搜索过程中遇到的最佳候选解  $\alpha_{SLS}$ （赋值  $\alpha_{SLS}$  下不可满足的子句的数目最少）。并且为了阐述方便，我们称某次局部搜索的赋值为该次局部搜索过程中的最佳候选解  $\alpha_{SLS}$ 。

我们的相位重置技术中考虑了以下赋值，所有这些赋值都来自过去的局部搜索赋值。

- $\alpha_{longest\_LS}$ . 这指的是一个局部搜索的赋值，产生于初始解为  $\alpha_{longest}$  (CDCL 的最深无冲突分支) 的一次局部搜索调用。因此，每当  $\alpha_{longest}$  更新时，算法均会调用局部搜索求解器来更新  $\alpha_{longest\_LS}$ 。
- $\alpha_{latest\_LS}$ . 这指的是最近产生的局部搜索的赋值。
- $\alpha_{best\_LS}$ . 在迄今为止所有局部搜索的赋值中，我们将最好（不可满足子句数目最少）的一个表示为  $\alpha_{best\_LS}$ 。

根据定义我们可以很容易看出， $\alpha_{max\_LS}$  和  $\alpha_{best\_LS}$  以最大化 CDCL

**Algorithm 4:** 基于松弛 CDCL 和相位重置的 CDCL 算法

---

输入: 一个 CNF 公式  $F$   
 输出: 求解结果 ‘SAT’ 或者 ‘UNSAT’

```

1  $dl \leftarrow 0, \alpha \leftarrow \emptyset, \alpha_{\max} \leftarrow \emptyset;$ 
2 if  $BCP(F, \alpha) \neq \emptyset$  then
3   return ‘UNSAT’
4 while  $\exists$  自由变元 do
5    $(v, p) \leftarrow branching(F, \alpha);$ 
6    $dl \leftarrow dl + 1;$ 
7    $\alpha \leftarrow \alpha \cup \{(v, p)\};$ 
8    $conf_l \leftarrow BCP(F, \alpha);$ 
9   if  $conf_l \neq \emptyset$  then
10     $bl \leftarrow conflict\_analysis(F, \alpha, conf_l);$ 
11    if  $bl < 0$  then
12      return ‘UNSAT’
13    else
14       $\alpha_{\max} \leftarrow max(\alpha_{\max}, \alpha);$ 
15       $backtrack(F, \alpha, bl), dl \leftarrow bl;$ 
/* 第 16 – 23 行与 4.2 节相关 */  

16 else if  $(|\alpha|/|V| > p \text{ OR } |\alpha|/|\alpha_{\max}| > q)$  且满足过滤条件 then
17    $\beta \leftarrow \alpha;$ 
18   while  $\beta$  不是完全赋值 do
19      $(v, p) \leftarrow branching(F, \beta);$ 
20      $\beta \leftarrow \beta \cup \{(v, p)\};$ 
21      $BCP(F, \beta);$ 
22   if  $SLS(\beta, terminate\_condition)$  then
23     return ‘SAT’
24   if 满足重启条件 then
25      $backtrack(F, \alpha, 0);$ 
26      $dl \leftarrow 0;$ 
27      $phase\_resetting();$  //与第 4.3 节相关
28 return ‘SAT’;

```

---

**表 4.1 相位重置技术不同相位选择概率****Table 4.1 Probability of different phases in our phase resetting mechanism**

相位种类	$\alpha\_longest\_LS[x]$	$\alpha\_latest\_LS[x]$	$\alpha\_best\_LS[x]$	no change
选择概率	20%	65%	5%	10%

分支的搜索深度为目标，而  $\alpha\_latest\_LS$  在某种意义上增加了多样性，因为不同的局部搜索过程从基于不同分支的初始赋值开始搜索。

考虑到现代 SAT 求解器中重启的速度，相位重置经常发生，并且以一定的概率（25%）在  $\alpha\_longest\_LS$  或  $\alpha\_best\_LS'$  上选择，使整体的搜索在优质的搜索方向上特别激进。总体而言，这种依赖局部搜索赋值的相位重置技术预期会特别适用于可满足实例的求解，我们的实验结果也证实了这一点。

**基于局部搜索赋值的相位重置** 每当 CDCL 重启时，我们都会覆盖之前相位保存技术的保存相位。每次重启时，我们会将相位根据表4.1的概率分布重置。该表中概率的选择经过了大量的参数调节测试，我们前期尝试了每种组合（增量调节 5% 的概率），最终选择的参数在综合性能和稳定性上有很好的均衡。

#### 4.3.2 基于局部搜索冲突频率信息的变元分支策略

CDCL 是一个高效的 SAT 求解框架，这主要得益于冲突信息的利用，而分支策略旨在促进冲突的产生。本节定义了一个变元的属性，局部搜索中的变元冲突频率（conflict frequency），该属性被用于改进 CDCL 的变元分支策略。

分支策略中最出名的策略为 VSIDS<sup>[42]</sup>，它在实践中效果出色且非常适用于主流的重启策略。尽管多年来提出了很多变种算法<sup>[44,76,243,258,259]</sup>，但是他们的核心思想均和 VSIDS 保持一致：更倾向于选择最近参与到冲突的变元。其中维护的变元活跃度打分 *activity* 可以被视为在冲突中出现的频繁程度，并重点关注于最近一段时间的冲突<sup>[44]</sup>。VSIDS 策略被应用于我们选择的三个基求解器中。

另一个比较流行的分支策略为 LRB<sup>[74]</sup>，自 2006 年以来，采用了该技术的在 Maple 系列求解器在 SAT 比赛主赛道中夺得了多个冠军<sup>3</sup>。LRB 中变元打分的更新主要依赖变元的学习率  $LR(v)$  来更新，衡量了变元赋值到取消赋值期间参与到冲突中频繁程度。

<sup>3</sup>LRB 的早期版本 CHB 算法<sup>[73]</sup> 被 2020 年的 kissat-bulky<sup>[260]</sup> 版本实现，但默认处于关闭状态。

**Algorithm 5:** 基于冲突频率的变元分支启发式提高算法

---

前置条件: 在局部搜中维护映射  $M : v \rightarrow confl\_count(v)$  和总步数  $steps$ 。

```

1 begin
2   if 以一半概率执行 then
3     for  $v$  in  $M$  do
4        $cf(v) \leftarrow confl\_count(v)/steps;$ 
5        $confl\_num(v) \leftarrow \lceil C \cdot cf(v) \rceil;$ 
6       if 采用 VSIDS then
7          $activity[v] \leftarrow activity[v] + var\_inc \cdot confl\_num(v);$ 
8       else if 采用 LRB then
9          $P(v, I) \leftarrow P(v, I) + confl\_num(v);$ 
10         $L(I) \leftarrow L(I) + confl\_num(v);$ 

```

---

直觉上, VSIDS 和 LRB 均更倾向于选择在最近的冲突中出现更频繁的变元。据此, 我们提出使用局部搜索过程中搜集到的有关变元冲突的信息改进这两个变元选择启发式。

**定义 4.1. (冲突频率)** 在一个局部搜索 SAT 求解器中, 某变元  $v$  的冲突频率  $cf(v) = \frac{confl\_count(v)}{steps}$ , 其中  $confl\_count(v)$  指的是变元  $v$  出现在至少一个不满足子句中的步数,  $steps$  为本次局部搜索过程执行的总步数。

有了上述的定义, 我们接下来将讨论 CDCL 如何在分支策略中使用局部搜索频率, 算法5为相关伪代码。因为  $cf(x)$  是一个处于 0 到 1 间的浮点数, 我们首先需要将其转化为一个整数, 这样才可以将其与 VSIDS 与 LRB 策略相结合。对于每个变元  $v$ , 我们会将  $cf(v)$  乘上一个常数  $C$  后再上取整 ( $C = 100$ ), 并将这个数字称为局部搜索冲突数  $confl\_num(v)$ , 然后采用下述的方法来提高两种不同的分支启发式<sup>4</sup>。此外,  $confl\_num$  的值都是根据最近一次调用的局部搜索过程来计算的, 且在每次 CDCL 重启之后, 算法会以一半的概率使用  $confl\_num$  去更新 VSIDS 的  $activity$  和 LRB 的学习率。

- VSIDS: 对于每一个变元  $v$ , 其  $activity$  分数会被提高 (执行 bump 操作)  $confl\_num(v)$  次, 即增加  $confl\_num(v) \cdot var\_inc$  次。

<sup>4</sup>kissat 求解器中采用了 VMTF 启发式, 而我们实验中绝大部分局部搜索样会更新所有变元的分数, 对应到 VMTF 中为所有变元均需要前置, 且利用冲突频率重新排序会严重影响 CDCL 的搜索顺序, 因此我们没有利用本节中的技术对 VMTF 进行改进。

- LRB: 对于每一个变元  $v$ , 在其  $I$  时间片内的参与到冲突分析中的数量会增加  $confI\_num(v)$  次, 即  $P(v, I)$  and  $L(I)$  均会分别增加  $confI\_num(v)$ 。

#### 4.4 实验评估

我们使用我们提出的三种方法来改进三个主流的 CDCL 求解器, 并进行了大量的实验来评估这些技术的有效性。本小节中, 我们介绍了基本的实验设置, 包括测试实例集合、运行环境和汇报评估的方法。

**测试实例集** 我们选取了连续 4 年的 SAT 比赛主赛道的标准测试实例集(2017 年–2020 年)。此外, 我们也选取了一个重要的应用问题的基准实例集合进行了测试, 该测试集来自于带宽拍卖场景下的频谱打包问题, 共包含 10000 个实例, 产生了约 70 亿美元的收入<sup>5</sup>[20]。

**测试环境及评估方法** 本章的所有实验都在具有 Intel Xeon Platinum 8153 @2.00GHz CPU 和 1024G RAM 的计算机集群下进行, 机器的操作系统为 CentOS 7.7.1908。对于每个实例, 每个求解器都执行一次运行, 截止时间为 5000 CPU 秒, 与 SAT 比赛的测试标准保持一致。对于每个基准的每个求解器, 我们报告了解决的 SAT/UNSAT 实例数和总解决实例数, 并记作 “#SAT”, “#UNSAT” 和 “#Solved”。此外, 我们也汇报了惩罚运行时间 “PAR2” (SAT 比赛中使用的判分标准), 其中在规定时间内未能求解的实例的运行时间被惩罚为截止时间的两倍。

**基求解器** 我们选择了三个前沿的 CDCL 求解器作为我们研究的基准求解器。三个求解器分别是 glucose (v4.2.1)<sup>6</sup>[68]、MapleLCMDistChronoBT-DL (v2.1)<sup>7</sup>[261] 和 kissat\_sat (2414b6d)<sup>8</sup>[50]。glucose 是 CDCL 求解器中的一个里程碑意义的求解器并且在 SAT 比赛中取得了多次金牌; MapleLCMDistChronoBT-DL 是 2019 年 SAT 竞赛的冠军; kissat\_sat2020 年 SAT 比赛主赛道的冠军。

我们选择 CCAnr<sup>[48]</sup> 作为局部搜索求解器, 将其集成到 CDCL 求解器 glucose 和 MapleLCMDistChronoBT-DL 中, 而 kissat\_sat 本身已经包含一个局部搜索求

<sup>5</sup>[https://www.cs.ubc.ca/labs/beta/www-projects/SATFC/cacm\\_cnfs.tar.gz](https://www.cs.ubc.ca/labs/beta/www-projects/SATFC/cacm_cnfs.tar.gz)

<sup>6</sup><http://sat-race-2019.ciirc.cvut.cz/solvers/glucose-4.2.1.zip>

<sup>7</sup><http://sat-race-2019.ciirc.cvut.cz/solvers/MapleLCMDistChronoBT-DL-v2.1.zip>

<sup>8</sup><https://github.com/arminbiere/kissat.git>

解器，复现了 YalSAT 求解器<sup>[49]</sup>中的技术。CCAnr 是一个旨在解决结构化 SAT 实例的局部搜索求解器，并在 SAT 竞赛和应用程序的各种结构化实例中展示了具有竞争力的结果。

我们在 glucose4.2.1 和 MapleLCMDistChronoBT-DL-v2.1 上实现了全部的三个技术，包括松弛 CDCL 希望分支的局部搜索交互方法（记做 rx）、基于局部搜索的相位选择技术（记做 rp）和基于局部搜索冲突频率信息的变元分支策略（记做 cf）。对于 kissat\_sat 求解器，我们只实现了 cf 技术。这是因为 kissat 数据结构的特殊设计和其中实现的多种化简技术，不利于我们高效准确地识别当前所有的学习子句（这部分子句需要传递给局部搜索），松弛 CDCL 的框架很难在该求解器上高效的实现；但是 cf 的技术在 kissat 中比较容易高效实现。本章所有源代码、实验数据和原始输出文件都可以在网上公开找到<sup>9</sup>。

#### 4.4.1 混合技术有效性分析

**在 SAT 比赛实例上的实验评估** 所有基准求解器的实验结果和采用本章中提到的技术的改进版本的实验结果均在表4.2中呈现，其中 Maple-DL-v2.1 是 Maple-LCMDistChronoBT-DL-v2.1 的缩写。从实验结果上，我们可以总结出下述的一些结论。

- rx 技术提高了 glucose 和 MapleLCMDistChronoBT-DL 在解决可满足实例方面的性能，特别是对于 2018 年的测试集（对于 #SAT，分别提高了 18 和 16）和 2020 年的测试集（对于 #SAT，分别提高了 25 和 35）。另一方面，glucose+rx 和 Maple-DL+rx 在不可满足实例上的性能略低于原始版本。在所有基准测试上，两个求解器的 #UNSAT 的平均值仅下降了 2。

- 通过添加 rp 技术，glucose+rx+rp 和 Maple-DL+rx+rp 在 #SAT 上得到了进一步的提高，且该技术对所有测试集都有巨大的提高。具体来说，在 2017、2018、2019 和 2020 年的测试集合上，glucose+rx+rp 的 #SAT 数量分别比 glucose+rx 分别多求解出 24、28、14 和 37 个；而且 Maple-DL+rx+rp 比 Maple-DL+rx 多 10、9、9 和 21 求解出。与 rx 技术类似，我们观察到在解决不可满足实例时性能略有退化，在两个求解器的 #UNSAT 求解数目平均减少 3 个。

- 从 glucose+rx+rp 与 glucose+rx+rp+cf、Maple-DL+rx+rp 与 Maple-DL+rx+rp+cf、kissat\_sat 与 kissat\_sat+cf 的比较中可以看出，cf 技术对解决所有测试

<sup>9</sup><https://github.com/shaowei-cai-group/relaxed-sat>

表 4.2 SAT 比赛 2017 年–2020 年数据集上的实验结果

Table 4.2 Experiment results on benchmarks from SAT Competitions 2017-2020

求解器	#SAT	#UNSAT	#Solved	PAR2	#SAT	#UNSAT	#Solved	PAR2
SAT Competition 2017 (351 个实例)				SAT Competition 2018 (400 个实例)				
glucose_4.2.1	83	101	184	5220.0	95	95	190	5745.9
glucose+rx	88	95	183	5237.0	113	95	208	5283.4
glucose+rx+rp	112	94	206	4618.2	141	87	228	4698.3
glucose+rx+rp+cf	110	94	204	4668.5	150	91	241	4438.2
Maple-DL-v2.1	101	113	214	4531.0	133	102	235	4533.9
Maple-DL+rx	101	112	213	4520.3	149	101	250	4148.6
Maple-DL+rx+rp	111	103	214	4447.1	158	93	251	4147.2
Maple-DL+rx+rp+cf	116	107	223	4139.4	162	97	259	3927.6
kissat_sat	115	114	229	3943.5	167	98	265	3786.4
kissat_sat+cf	113	113	226	4001.0	178	104	282	3409.4
CCAnr	13	N/A	13	9629.9	56	N/A	56	8622.0
SAT Race 2019 (400 个实例)				SAT Competition 2020 (400 个实例)				
glucose_4.2.1	118	86	204	5437.6	68	91	159	6494.6
glucose+rx	120	84	204	5443.9	93	88	181	6018.1
glucose+rx+rp	134	85	219	5096.3	130	85	215	5123.7
glucose+rx+rp+cf	140	85	225	4923.6	134	87	221	4977.9
Maple-DL-v2.1	143	97	240	4601.8	86	104	190	5835.7
Maple-DL+rx	146	93	239	4602.1	121	105	226	4977.8
Maple-DL+rx+rp	155	94	249	4416.3	142	99	241	4589.2
Maple-DL+rx+rp+cf	154	95	249	4377.4	151	106	257	4171.1
kissat_sat	159	88	247	4293.5	146	114	260	4048.8
kissat_sat+cf	162	90	252	4211.7	157	113	270	3896.8
CCAnr	13	N/A	13	9678.3	45	N/A	45	8978.7

集合上的可满足和不可满足实例都有积极作用，但在 2017 年测试集合的 glucose+rx+rp+cf 和 kissat\_sat+cf 除外（少求解了 2 个和 3 个例子）。对于 2018 年、2019 年和 2020 年的测试集合，cf 技术帮助 #Solved 数量显著增加：glucose+rx+rp+cf 分别提高了 13、6 和 6 个；Maple-DL+rx+rp+cf 分别提高了 8、0 和 7 个；kissat\_sat+cf 提高了 17、5 和 10 个。特别地，kissat\_sat 是 SAT 比赛 2020 年主赛道的冠军，代表了当时最新技术水平，这种改进对于单一技术来说是十分显著的。

- 在所有测试集合上，通过实现我们提出的三种技术，glucose 和 MapleLCM-DistChronoBT-DL 都得到了极大的改进。特别地，在 SAT 比赛 2020 年的标准测试实例集合的 400 个例子上，glucose+rx+rp+cf 比原始求解器多求解出 62 个测试实例，且 Maple-DL+rx+rp+cf 比其原始求解器多求解器出 67 个实例。
- 值的一提的是，Maple-DL+rx+rp+cf 是我们求解器 Relaxed\_LCMDCBDL\_newTech 的简化和提高版本，该求解器在 SAT 比赛 2020 年中获得了主赛道 SAT 类别的冠军和主赛道 ALL 类别的亚军。另外，Maple-DL+rx 的初期版本 ReasonLS 取得了 SAT 比赛 2020 年 NoLimits 赛道的冠军；Relaxed\_LCMCBDL\_newTech 的一个调参改进版本 lstech 取得了 SAT 比赛 2021 年主赛道的一个银牌；将 lstech 技术在 kissat 系列求解器上的一个版本 kissat-inc 取得了 SAT 比赛 2022 年 NoLimits 赛道的冠军且取得了主赛道的两枚银牌。

**在带宽拍卖问题上的实验评估** 我们也在一个有真实工程业务应用的实例集合上进行了实验测试，该组例子来自于美国联邦通信委员会（FCC）带宽拍卖问题的频谱打包项目，美国政府从这个项目中获益 70 亿美元<sup>[20]</sup>。我们在这组包含 10000 个实例（即有可满足例子也有不可满足例子）的测试集上，将我们的改进求解器与其原始 CDCL 求解器和局部搜索求解器 CCAnr 做了性能比较，相关的实验结果在表 4.3 中进行了汇报，其中 glucose+ 是 glucose+rx+rp+cf 的缩写，Maple+ 是 Maple-DL+rx+rp+cf 的缩写。实验表明，对于每一个基求解器，我们的改进版本都能在 FCC 实例集合上取得更好的实验效果。

特别地，Maple-DL+rx+rp+cf 求解器求解出了最多的实例 ( $8759 + 218 = 8977$ )，显著超过了其它求解器。

表 4.3 在 FCC 实例集上与主流方法的对比

Table 4.3 Comparing with state-of-the-art solvers on FCC

实例集	glucose	glucose+	Maple	Maple+	kissat_sat	kissat_sat+cf	CCAnr
	#SAT	#SAT	#SAT	#SAT	#SAT	#SAT	#SAT
	#UNSAT	#UNSAT	#UNSAT	#UNSAT	#UNSAT	#UNSAT	#UNSAT
	#Solved	#Solved	#Solved	#Solved	#Solved	#Solved	#Solved
	PAR2	PAR2	PAR2	PAR2	PAR2	PAR2	PAR2
FCC (10000)	7330	8075	8084	8759	8192	8214	7853
	187	197	215	218	207	211	0
	7517	8272	8299	8977	8399	8425	7853
	2555.85	1850.58	1867.13	1243.66	1760.55	1734.61	2215.35

#### 4.4.2 两种引擎间的协作性分析

我们在 glucose 和 MapleLCMDIstChronoBT-DL 两个求解器的基础上，进一步的分析了局部搜索算法在其中扮演的角色。这组实验中不包含 kissat\_sat 相关的结果，是因为我们并没有在其上实现松弛 CDCL 希望分支的局部搜索交互技术，相关的数据统计并不符合 kissat\_sat+cf 求解器。表4.4列举出了一些重要的数据，其中 #byLS 是局部搜索给出求解结果的数量，#SAT\_bonus 是基 CDCL 求解器和 CCAnr 均无法求解，但是采用了本章技术之后的混合求解器可以多求解的可满足实例的数量。#LS\_call 是 CDCL 求解过程中局部搜索平均的调用次数，LS\_time 是局部搜索算法占混合算法整体运行时间的百分比，我们分别给出了在可满足和不可满足的实例中这两个指标的计算结果。

根据之前的实验数据，我们可以了解到局部搜索求解器能够求解部分可满足实例，且不同的测试集上，这个数字的差距变化较大。一个比较自然的问题是：混合求解器的提高是否主要来自于 CDCL 求解器和局部搜索求解器可以互补求解的实例？如果是这样的话，那么简单的顺序执行两个求解器将会得到本文中混合算法的性能。为了回答这个问题，我们将混合算法可以求解的实例和混合前的 CDCL 求解器与局部搜索求解器可以求解的实例做了对比（每个实例上，CDCL 和局部搜索求解器均设置了同样的 5000s 截止时间）。我们观察到，有大量实例（用 #SAT\_bonus 表示）无法通过 CDCL 和局部搜索求解器求解，但可以通过混合求解器求解。对于这些实例，即使是选择每个实例最佳结果的虚拟最

表 4.4 有关局部搜索在 CDCL 中作用的分析

**Table 4.4 Analyses on the impact of Local Search on the CDCL solvers.**

求解器	SAT 用例分析				UNSAT 用例分析	
	#byLS	#SAT_bonus	#LS_call	LS_time(%)	#LS_call	LS_time(%)
SAT Competition 2017 (351)						
glucose+rx	20	11	24.28	21.66	16.36	5.52
glucose+rx+rp	10	33	17.77	18.46	14.33	4.86
glucose+rx+rp+cf	17	29	22.7	22.19	15.3	5.81
Maple-DL+rx	16	9	13.86	7.52	11.18	2.03
Maple-DL+rx+rp	11	15	9.63	10.43	6.54	2.36
Maple-DL+rx+rp+cf	6	16	12.59	7.49	8.59	2.12
SAT Competition 2018 (400)						
glucose+rx	50	4	11.27	20.66	29.62	4.94
glucose+rx+rp	47	31	9.46	18.4	21.66	5.64
glucose+rx+rp+cf	53	36	11.43	20.28	20.62	6.64
Maple-DL+rx	52	7	4.8	13.02	11.69	2.81
Maple-DL+rx+rp	56	13	4.84	15.21	8.7	3.04
Maple-DL+rx+rp+cf	51	18	6.52	12.53	15.62	2.94
SAT Race 2019 (400)						
glucose+rx	14	8	26.46	10.79	17.42	6.39
glucose+rx+rp	10	26	22.68	8.67	14.59	5.14
glucose+rx+rp+cf	11	26	20.39	11.82	15.51	5.95
Maple-DL+rx	14	7	12.66	2.67	12.94	1.98
Maple-DL+rx+rp	9	14	8.6	3.17	16.59	2.53
Maple-DL+rx+rp+cf	12	15	11.21	3.05	17.23	2.22
SAT Competition 2020 (400)						
glucose+rx	30	9	14.94	11.75	14.67	10.27
glucose+rx+rp	23	37	13.17	10.79	9.4	9.71
glucose+rx+rp+cf	23	37	12.78	11.67	10.52	10.28
Maple-DL+rx	19	13	14.21	6.69	10.24	5.25
Maple-DL+rx+rp	30	29	8.53	6.62	11.7	6.18
Maple-DL+rx+rp+cf	23	36	10.95	6.05	14.17	5.42

佳求解器（virtual best solver）也会求解失败。对于 glucose，四组 SAT 比赛实例集的多求解数量分别为 29、36、26 和 37 个；对于 MapleLCMDIstChronoBT-DL，四组数据集上的多求解数量分别为 16、18、15 和 36 个。这表明本章的混合技术对混合求解器的高性能做出了重要贡献。

我们还计算了每次运行中局部搜索求解器的调用次数，对于这些测试实例集，这个数字通常在 10 到 25 之间。至于局部搜索的运行时间，这可以看作是使用局部搜索的代价，因此我们计算了局部搜索所花费的时间比例：对于可满足的实例，这个数字在 6% 到 20% 之间；对于不可满足的实例，这个数字会显著下降，通常小于 7%。这与观察到的局部搜索调用次数不一定在不可满足的实例上减少的情况并不矛盾，因为局部搜索所花费的时间比例也取决于混合求解器的总时间。我们的统计数据显示，对于 glucose+rx+rp+cf 和 Maple-DL+rx+rp+cf，求解 UNSAT 实例的平均时间大约是 SAT 实例的 1.5 倍到 2 倍。简而言之，对于 UNSAT 实例来说，代价通常很小，是可以接受的，这也解释了为什么我们的技术对解决 UNSAT 实例没有明显的负面影响，尽管我们的方法是倾向于可满足实例求解的。

## 4.5 本章小节

本章的工作向 CDCL 和局部搜索的深度协作迈出了一大步。我们提出了三种使用局部搜索来改进 CDCL 求解器的方法。第一个想法是利用松弛的方法保护希望分支免受剪枝，进入一种无回退模式构造完全赋值，并使用局部搜索求解器在该解空间附近搜索。第二个想法是在相位重置阶段，启发式的选择不同的局部搜索过程中收集的候选解。最后，我们利用局部搜索过程中变元的冲突频率来增强 CDCL 求解器的分支选择策略。这些技术显著提高了前沿 CDCL 求解器在有实际应用背景的标准测试实例集合上的性能。我们提出的方法是一种通用方法，可以应用于改进其他 CDCL 求解器。

这是第一次成功的系统搜索与随机搜索的混合算法，可以在有应用背景的实例集合上，相比现有技术，有根本性的改进，从而回答了命题推理和搜索十大挑战中的第七项挑战<sup>[249]</sup>。该技术是对于混合技术的一次成功的尝试，该技术有望应用于除 SAT 问题之外的其它领域，如 SMT（将于下一章展开）、QBF 等问题。相关技术也有希望帮助提高和改进求解器在多种应用领域中的应用能力。

## 第5章 整数算术理论 SMT 问题的混合求解算法

前面的章节中介绍了 SAT 问题 CDCL 与局部搜索的深度混合算法，性能取得了巨大的提升。SMT(IA) 问题的主流求解算法为 CDCL(T) 框架，局部搜索算法研究仍比较初步，且不存在 CDCL(T) 与局部搜索算法的深度混合策略。本章主要讨论了如何将 SAT 领域的成功混合技术迁移到 SMT(IA) 问题的求解中。5.1 节阐述了相关的背景知识和动机。类似的，5.2 节介绍了一种双层的混合求解框架，并介绍了局部搜索求解器的调用时机。5.3 节给出了两种利用局部搜索信息改进 CDCL(T) 启发式的技术。5.4 节通过大量的实验和分析证明了本方法的有效性。

实验表明，我们的混合方法可以比 SMT(IA) 问题领域综合能力最好的求解器 z3 多求解 10.36% 的样例，整体提速了 52.76%。相关方法有望推动软件终止性问题的发展。基于相关策略提出的求解器参与了 2023 年的 SMT 比赛，获模型验证总比分“最大领先奖”与“最大贡献奖”金牌，并获得 IA 多个细分赛道（NIA、LIA 理论等）的冠军。

### 5.1 引言

CDCL(T)<sup>[164]</sup> 是求解包括 IA 理论在内的多种理论上的 SMT 问题的代表性框架，CDCL(T) 可以看作是在 SAT 问题的求解方法 CDCL 上拓展了理论求解和推理的算法。它将给定的 SMT 公式抽象为一个布尔骨架，并使用基于 CDCL 的 SAT 求解器来处理该骨架。同时，一个与理论相关的决策过程，即理论求解器，被用来改进布尔骨架并指导 SAT 求解器（通过学习新的原子公式或子句）的搜索。几乎所有的最先进的 SMT 求解器都实现了 CDCL(T) 方法，包括 Z3<sup>[37]</sup>，CVC5<sup>[38]</sup>，Yices2<sup>[39]</sup>，MathSAT5<sup>[166]</sup>，SMTInterpol<sup>[262]</sup> 等求解器。

除了 CDCL(T) 之外，还存在诸多解决 SMT(NIA) 问题的方法，如将公式直接转化为 SAT 公式的位展开方法<sup>[57,202,263]</sup>。目前，将变量之间的依赖关系编码为布尔约束的方法主要有：通过限制算术变量范围的方法<sup>[202,264]</sup>，或推导所有可能的传递性约束的方法<sup>[265]</sup>，或者同时使用这两种技术<sup>[266]</sup> 的方法。此外，也有工作研究了位展开相关的预处理方法<sup>[267,268]</sup>。目前，也存在基于模型构建满足性

(MCSat) 相关的研究，它们扩展了 CDCL(T) 框架，为算术变量赋予具体值，而不是为骨架赋予布尔值。另外，其它的求解方法主要包括增量线性化<sup>[55,209,211,212]</sup>，在柱形代数分解（CAD）的上下文中进行分支定界、以及虚拟替换（VS）的方法<sup>[187]</sup>。强化学习也被用来改进 CAD 中的变量顺序选择启发式<sup>[269]</sup>。最先进的 SMT(NIA) 求解器，如 Z3<sup>[37]</sup>，则是采用了以 CDCL(T) 为主的多类算法的顺序组合。

另一方面，SMT(IA) 问题领域的局部搜索算法是一个新兴的研究话题，代表求解器为 LocalSMT<sup>[217]</sup>。其中提出了一种新型的局部搜索算子，局部搜索求解器会直接在算术变元上操作，而不是在提取的布尔骨架上操作。它从一个完整的赋值开始，然后迭代地修改一些变量的值，直到找到一个模型或达到时间限制。局部搜索与 CDCL(T) 有一定的互补性，在一些由软件验证问题编码的可满足实例的基准测试中，局部搜索求解器的性能优于现有方法。其算法框架类似于第 2.2.2 节中的 SAT 局部搜索框架，区别在于，LocalSMT 维护的候选解中包含了算术变元的赋值，候选解的质量考虑的是算术文字的真假情况，且其中的一个算术文字可以对应于 CDCL(T) 算法中的一个抽象布尔文字。

上一章介绍了一套 CDCL 与局部搜索成功的深度混合算法。它以 CDCL 为主，并阶段性的调用局部搜索算法。深度混合的重要特征为 CDCL 与局部搜索之间信息的互相传递。且近几年主流的 SAT 求解器均采用了深度混合的求解策略 (kissat<sup>[50]</sup>, CaDiCaL<sup>[51]</sup>, CryptoMiniSat<sup>[270]</sup>, Relaxed)。

本章介绍的工作之前，我们并没有找到一个将 CDCL(T) 与局部搜索深度结合的 SMT(IA) 求解器，受到 SAT 的 CDCL 与局部搜索深度混合算法的启发，本工作致力于通过 CDCL(T) 与局部搜索的深度合作，设计一个高效的混合 SMT(IA) 求解器。

本章在混合算法设计时，主要关注了以下的三个问题：

**问题 1：**如何合理地调度局部搜索和 CDCL(T)？

我们设计了一个两层的混合方法。在外层，我们的方法在内层求解器和“独立的局部搜索”(I-SLS) 之间进行交替。内层以 CDCL(T) 为主体，在有希望的骨架解对应的分支上调用一个“由 CDCL(T) 引导的局部搜索”(CG-SLS) 求解器；CDCL(T) 求解器和被引导的局部搜索之间相互传递信息。

**问题 2：**如何使用 CDCL(T) 来引导局部搜索？

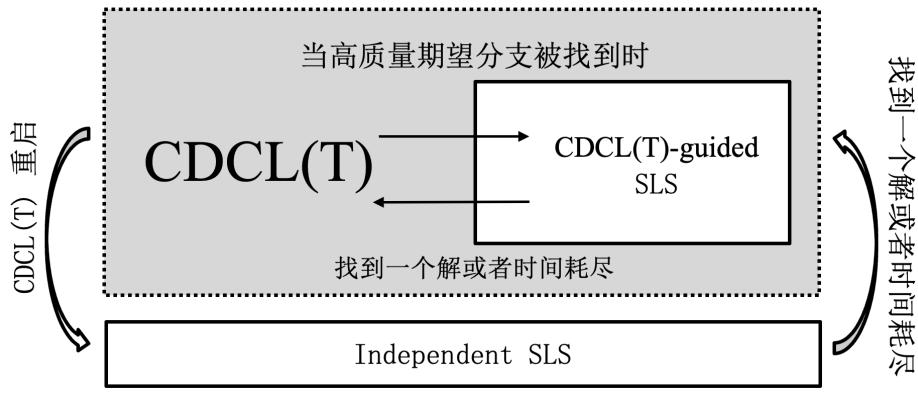


图 5.1 CDCL(T) 与局部搜索的混合框架

Figure 5.1 The CDCL(T) and SLS hybrid framework

当 CDCL(T) 达到满足输入公式的布尔骨架的赋值时，我们根据该赋值提取一个子公式，并调用局部搜索在该提取的公式上工作。如果局部搜索找到一个解，则原始公式是可满足的。如果局部搜索在时间限制内未能找到解，CDCL(T) 搜索过程将从局部搜索进入时的断点处继续。我们使用启发式的差异分数 (difference score) 来过滤掉局部搜索调用，以防止两个相邻的局部搜索在同一搜索空间中搜索。

**问题 3：**如何利用来自局部搜索的信息来增强 CDCL(T)？

**利用局部搜索赋值和冲突频率改进 CDCL(T) 分支启发式：**分支是 CDCL(T) 最关键的组成部分之一，涉及变元排序和相位选择。我们设计了一种相位重置方法，使用最近的 CG-SLS 调用中的最佳赋值，即具有最少未满足子句的赋值。此外，我们还利用原子公式的冲突频率 (conflict frequency) 来改进 CDCL(T) 的变量排序启发式。

## 5.2 结合 CDCL(T) 与局部搜索的混合求解框架

本节提出了一种 CDCL(T) 与局部搜索之间双层局部搜索框架，并介绍了调用局部搜索算法的策略。

### 5.2.1 SMT(IA) 问题的双层混合框架

我们为 SMT(IA) 问题设计了一个混合求解框架，该框架不仅利用了 CDCL(T) 求解器和局部搜索求解器的互补能力，而且通过信息交流改进了这两种求解器。这个框架中包含有一个 CDCL(T) 求解器和两个局部搜索求解器。

这两个局部搜索求解器扮演不同的角色，我们将其中一个称为独立局部搜索求解器（Independent SLS，简称 I-SLS），另一个称为  $CDCL(T)$  引导下的局部搜索求解器（ $CDCL(T)$ -guided SLS，简称 CG-SLS）。

总体上，我们的方法分为两层，如图 5.1 所示。在外层，我们的求解器可以看作是一个顺序组合的求解器，它会在内层的混合求解器和 I-SLS 求解器之间交替执行。内层的主体为  $CDCL(T)$  求解器，它与 CG-SLS 求解器进行交互，彼此通信交互关键信息。在这种架构之下，外层专门用于利用  $CDCL(T)$  和局部搜索之间的互补性能来提高求解能力；我们工作的重点在于内层，对应于深度结合  $CDCL(T)$  和局部搜索的混合求解器，也是这项工作的主要贡献。

接下来，我们将描述我们外层的实现方法和内层的细节，使用局部搜索信息改进  $CDCL(T)$  启发式的方法，将在第 5.3 节中介绍。

**I-SLS 的调用策略** 在外层，独立局部搜索仅在  $CDCL(T)$  重启后被调用。且调用 I-SLS 的条件遵循一个规则，即局部搜索被调用的频率越来越低。I-SLS 的第 1 次调用发生在  $CDCL(T)$  的第一次重启之后，第 2 次调用发生在自第 1 次调用 I-SLS 后开始计算的第  $q$  次重启后，第 3 次调用发生在自第 2 次调用 I-SLS 后开始计算的第  $q^2$  次重启后。一般来说，当自第  $(i-1)$  次调用 I-SLS 以来已经发生了  $q^{i-1}$  次重启之后，算法就会第  $i$  次启用 I-SLS。在我们的工作中， $q$  设置为 2。

在这项工作中，我们采用 LocalSMT 求解器作为独立局部搜索求解器。每次激活 I-SLS 时，它都会在没有  $CDCL(T)$  的任何指导下对整个公式进行工作，旨在搜索步骤的限制内找到一个模型，该限制由一个参数  $\theta$  控制。我们将  $\theta$  设置为  $3 \times 10^4 + (i-1) \times 10^4$ ，上限为  $3 \times 10^5$ 。

如果 I-SLS 返回 ‘SAT’，则给定的公式是可满足的；否则，当每次 I-SLS 调用的步骤限制达到时，算法将切换到内层的  $CDCL(T)$  过程。一个公式的不可满足性只能通过  $CDCL(T)$  过程来证明。

**外层设置的直觉** 在本小节末尾，我们试图对 I-SLS 的调用和终止策略提供设计的思路。（1）为了解决两个求解器（ $CDCL(T)$  和 LocalSMT）中的一个求解器比较擅长而另一个不太擅长的样例，每个求解器的时间片应该从小开始，并随着求解的进行而逐渐增加。因此，I-SLS 的步骤限制逐渐增加。（2）I-SLS 有循

**Algorithm 6:** SMT(IA) 问题混合算法伪代码

**输入:** 一个 SMT(IA) 问题的 CNF 公式

**输出:** 求解结果 ‘SAT’，或 ‘UNSAT’。

/\* 与期望分支相关的代码被标记为 **棕色**(第 5.2.2 节); 与局部搜索信息利用的相关代码被标记为 **红色**(第 5.3 节) \*/

```

1  r  $\leftarrow 0$ , p  $\leftarrow 0$ ;
2  ds  $\leftarrow 0$ , gds  $\leftarrow \epsilon$ ;
3  while True do
4      c  $\leftarrow \text{propagate}()$  /* BCP 与理论传播 */  

5      if c  $\neq \emptyset$  then
6          l  $\leftarrow \text{resolve\_conflict}(\iota)$ ;;
7          if l  $< 0$  then return ‘UNSAT’;
8          backtrack(l);
9          if  $\exists$  自由的核心子句 then
10             ds  $\leftarrow ds + 1$ 
11     else
12         /* 希望分支 (第 5.2.2 节) */  

13         if 找到一个骨架解 and ds  $> g_{ds}$  then
14             CDCL(T)_guided_SLS();
15             /* 更新相位和变元分支顺序 (第 5.3 节) */  

16             SLS_update_phase_order();
17             ds  $\leftarrow 0$ , gds  $\leftarrow g_{ds} + 1$ ;
18             if !decide() then return ‘SAT’;
19             if 满足重启条件 then
20                 restart();
21                 SLS_rephase(); /* 相位重置 (第 5.3.1 节) */  

22                 if (r++)  $\geq (1 - q^p)/(1 - q)$  then
23                     independent_SLS();
24                     SLS_update_order(); /* 冲突频率 (第 5.3.2 节) */
25                     p  $\leftarrow p + 1$ ;
26             ds  $\leftarrow 0$ , gds  $\leftarrow \epsilon$ ;

```

环现象，即它会访问之前搜索过的空间，这是因为它不记录以前的搜索历史。但是 CDCL(T) 可以利用搜索过程中学习到的信息帮助修剪搜索空间。因此，随着求解过程的进展，最合理的时间分配方式应该是：逐渐减少 I-SLS 的时间占用，并为 CDCL(T) 部分（实际上是内层的混合求解器）分配更多时间。在 I-SLS 的实现中，LocalSMT 被直接调用，不添加任何其他约束、限制和优化。

### 5.2.2 CDCL(T) 指导的局部搜索

双层框架的内层是一个 CDCL(T) 求解器与局部搜索求解器的深度混合求解器，它在两个方向上进行协同：CDCL(T) 引导局部搜索，而局部搜索将信息反馈给 CDCL(T)。下面，本节主要介绍 CDCL(T) 如何引导局部搜索。

**从骨架解中提出子公式** 根据之前的介绍，CDCL(T) 求解器由基于 CDCL 的 SAT 求解器和理论求解器组成。SAT 求解器负责处理布尔骨架，其中每个 IA 原子公式被编码为抽象布尔变元（本文中也称为“编码器”）。我们称布尔骨架的“解”为“骨架解”，他是一组针对布尔变元的（可能是部分的）赋值，包括对抽象布尔变元和原始公式中的布尔变元的赋值。骨架解需要满足初始公式对应的布尔骨架，但不需要包括在 CDCL(T) 过程中添加的抽象布尔变元和学习的子句。在 Z3 中，原始子句（可能在求解之前简化）也被称为“核心（core）子句”。

当 CDCL(T) 求解器找到一组骨架解、并满足在下一小节中介绍的过滤条件时，算法就会调用 CG-SLS 求解器。

当 CDCL(T) 求解器满足调用局部搜索求解器的条件时，会通过只保留赋值为 T 的文字的方式，从原始公式中提取子公式。显然，如果得到的子公式是可满足的，那么原始 SMT(IA) 公式也是可满足的。这个提取的子公式会被传递给局部搜索求解器。从这个意义上说，我们称局部搜索受 CDCL(T) 求解器引导，因为它在由 CDCL(T) 找到的骨架解所导出的子公式上执行搜索任务。

**例 5.1.** 给定一个 SMT(IA) 公式  $F_{SMT(IA)} = (p_1 \vee \neg p_2) \wedge (\neg(3x_1 x_2 \leq 2) \vee (\neg x_2 \wedge \neg x_4 \leq 0)) \wedge (p_2 \vee (2x_3^3 x_4 + 4x_2 + x_1 \leq 8))$ ，其布尔骨架为  $S_F = (p_1 \vee \neg p_2) \wedge (\neg p_{\sigma_1} \vee p_{\sigma_2}) \wedge (p_2 \vee p_{\sigma_3})$ 。一个部分解  $\alpha = \{p_1 \rightarrow T, p_{\sigma_1} \rightarrow F, p_2 \rightarrow F, p_{\sigma_3} \rightarrow T\}$  为该骨架的一个骨架解，因为在当前赋值下， $S_F$  中所有的子句均已经被满足。在当前的搜索分支下，通过剔除所有不满足的文字和自由文字，一个该骨架下的子公式  $S'_F = (p_1 \vee \neg p_2) \wedge (\neg p_{\sigma_1}) \wedge (p_{\sigma_3})$  可以被提取出来。通过将抽象布尔文字替换为对应

的 IA 文字，我们得到了一个 SMT(IA) 的子公式  $F_{SLS} = \{(p_1 \vee \neg p_2) \wedge \neg(3x_1 x_2 \leq 2) \wedge (2x_3^3 x_4 + 4x_2 + x_1 \leq 8)\}$ 。

值得一提的是，当提取交给 CG-SLS 的子公式时，我们只关注原始公式中的组成部分。我们不会将学习到的布尔骨架上的公式交给局部搜索求解器，因为这会给局部搜索带来巨大的搜索负担。根据前文中的介绍，LocalSMT 的运行时间表现与子句的数量密切相关。且实验表明，采用类似于典型的 SAT 求解器中的技术，从 CDCL(T) 的抽象布尔框架中拷取并传递 LBD<sup>[68]</sup> 不超过 6 的高质量学习子句给 CG-SLS，性能不会有太大提升。

**CG-SLS 的过滤条件** 当 CDCT(T) 找到一个骨架解的时候，CG-SLS 会被启用。但是，如果当每次遇到骨架解的时候都启用 CG-SLS 的话，会导致局部搜索进入的次数过多，且其中有很多相邻的局部搜索算法搜索的公式会过于相似。

**例 5.2.** 给定一个 SMT(IA) 公式  $F_{SMT(IA)} = (p_1 \vee \neg p_2) \wedge (\neg(3x_1 x_2 \leq 2) \vee (\neg x_2 - 3x_4 \leq 0)) \wedge (p_2 \vee (2x_3^3 x_4 + 4x_2 + x_1 \leq 8))$ ，并假设前两条子句为原始子句，且最后一条子句为学习子句。公式  $F_{SMT(IA)}$  的命题骨架为  $S_F = (p_1 \vee \neg p_2) \wedge (\neg p_{\sigma_1} \vee p_{\sigma_2}) \wedge (p_2 \vee p_{\sigma_3})$ 。现在有三个连续的命题骨架  $S_F$  的骨架解， $\alpha_1, \alpha_2$ , and  $\alpha_3$ ，其中： $\alpha_1 = \{p_1 \rightarrow T, p_{\sigma_1} \rightarrow F\}$ ,  $\alpha_2 = \{p_1 \rightarrow T, p_{\sigma_1} \rightarrow F, p_{\sigma_3} \rightarrow F\}$ ,  $\alpha_3 = \{p_1 \rightarrow T, p_{\sigma_1} \rightarrow F, p_{\sigma_3} \rightarrow F\}$ . 从上面的三个骨架解，我们可以提取出一个相同的子公式  $F_{sub} = \{p_1 \wedge \neg(3x_1 x_2 \leq 2)\}$  并将其交由 CG-SLS 求解。另一方面，我们可以基于骨架解  $\alpha_4 = \{p_1 \rightarrow T, p_{\sigma_1} \rightarrow F, p_2 \rightarrow F\}$  提取出一个不同的子公式  $F'_{sub} = \{(p_1 \vee \neg p_2) \wedge \neg(3x_1 x_2 \leq 2)\}$ 。

根据例 5.2，我们可以知道在搜索的过程中有可能会遇到很多不同的骨架解可以提取出相似的子公式，甚至是提取出相同的子公式。因为在 CDCL(T) 中插入局部搜索的最初目的也是为了让局部搜索能在不同的高质量（赋值的一致性较高）搜索空间内搜索，因此这些子公式的多次重复求解一般无法产生正收益。为此，我们设计了一个过滤条件来减少产生相似子公式的可能。

因为对于大部分例子来说，命题骨架上的布尔变元的数量基本上等同于算术文字的数目，直接计算两个骨架解的不同几乎等同于访问整个公式，时间开销较大。因此，我们设计了一个方法来启发式的衡量骨架解之间差异的大小。我们采用监控 CDCL(T) 中重启行为的方法来衡量距离上次遇到骨架解以来的赋值变化。

当 CDCL(T) 执行回退操作时，至少其中一个命题骨架变元的赋值会被反转，因此多次重启之后，搜索空间距离上次进入 CG-SLS 时会发生一定的变化。我们采用区别打分 (difference score，记作  $ds$ ) 来衡量距离上次进入 CG-SLS 以来执行的重启次数。值得一提的是，理论传播有可能会产生新的抽象布尔变元，这些变元不属于原始公式，不会被用于产生传递给 CG-SLS 的子公式。由于新学习的抽象布尔变元的产生，重启中反转赋值的变元有可能不会反转原始公式中的抽象布尔变元。但是如果回退时当前赋值为非骨架解，甚至由骨架解变为非骨架解时，为了产生一个新的骨架解，至少有一个原始公式中的抽象布尔变元会被重新赋值，再次赋值时该变元的值有可能与之前不同。因此， $ds$  只统计重启之后的赋值为非骨架解的重启出现的次数。

如果 CDCL(T) 遇到一个骨架解，并且  $ds$  比一个阈值  $g_{ds}$  的数目更大，我们则会启用 CG-SLS 进行搜索。如算法6所示， $g_{ds}$  被初始化为  $\epsilon$ ，且每次调用完 CG-SLS 之后，会将  $g_{ds}$  的数值增加 1。本章将阈值参数  $\epsilon$  设置为 3。

I-SLS 和 CG-SLS 都采用 LocalSMT 作为局部搜索的求解引擎。与 I-SLS 相比，CG-SLS 的调用时间通常会短很多，但调用频率会相对更高。实现中，我们也将 CG-SLS 的执行步数限制设置的更少一些。此外，如果 CG-SLS 不能找到一个解，那么因为 CDCL(T) 在连续一段时间内的搜索空间相似性较高的原因，很大概率后续的几个 CG-SLS 也会失败，但是当 CDCL(T) 重启时，搜索空间会发生较大的变化。基于以上的观察，我们对 CG-SLS 的执行步数限制  $\theta$  设置了如下的操作：每次重启时， $\theta$  被设置为  $3 \times 10^4$  (I-SLS 的最小值)，且每次调用 CG-SLS 之后  $\theta$  会自减  $1 \times 10^4$ ，且不低于  $1 \times 10^4$ 。

### 5.3 基于局部搜索的 CDCL(T) 改进策略

本节介绍了如果利用局部搜索过程中的信息来提高 CDCL(T) 的启发式。CDCL(T) 的性能会受到变元的分支顺序和相位选择启发式的显著影响。在 Z3 的 CDCL(T) 子求解器中，变元的分支顺序采用的是 VSIDS 启发式<sup>[42]</sup>，相位的选择主要选用了相位保留策略<sup>[75]</sup>。本节主要介绍如何采用局部搜索来提高上述的两类启发式。

上一章向我们展示了 CDCL 求解器的变元分支顺序和相位决策启发式能用通过局部搜索中的变元辅助和冲突频率来改进。但是上述的 SAT 问题混合求解

技术无法直接迁移到 SMT 问题的 CDCL(T) 框架中。因此本节主要介绍了如何将 SAT 求解上的成功技术迁移到 SMT 问题的求解中。

### 5.3.1 基于局部搜索解的 CDCL(T) 相位重置技术

CDCL(T) 求解器采用 CDCL 算法作为主要的推理引擎来处理抽象布尔骨架。因此, CDCL 的启发式会对 CDCL(T) 的性能产生巨大的影响。CDCL 和 CDCL(T) 中一个特别关键的组件为相位决策组件。

最近的 SAT 求解研究工作中, 高效的相位重置启发式引起了广泛的关注。Biere 等人提出了一个基于靶相位的相位重置组件<sup>[244]</sup>; 我们的工作中提出了一个基于局部搜索高质量解的相位重置组件。相关工作均获得了 SAT 比赛的冠军。

我们提出了一种基于 CG-SLS 中高质量解的相位重置方法。局部搜索求解器 LocalSMT 维护对输入公式中变元(包括布尔变元和整数变元)的一组完全赋值, 注意此处的布尔变元不是抽象布尔变元, 而是原始公式中的布尔变元。因此, 我们需要将局部搜索中的结果转换为抽象布尔骨架上变元的赋值, 以便它们可以在 CDCL(T) 引擎中使用。

局部搜索求解器会在一组完全赋值组成的空间上随机游走。借助一个 SMT 问题的“词级别”完全赋值  $\alpha$ , 我们可以很轻松的计算出其输入公式中所有原子公式的满足性。因此我们通过将抽象布尔变元的赋值取为对应的原子公式的满足性(满足赋值为 T, 未满足赋值为  $\perp$ ), 也可以轻松的得到抽象布尔骨架上对应的抽象布尔变元的一组“位级别”的(部分)赋值, 并被称为转化赋值。因为有一些抽象变元并没有生成对应的 SMT 子公式, 所以它们并不能从局部搜索的赋值中获得一个对应的赋值, 该类变元的赋值会被标记为“未知赋值”。

根据算法6中描述, 我们使用 CG-SLS 中得到的转化赋值来执行相位重置操作。为了达到这个目标, 我们会在每一次局部搜索中会记录打分最好(未满足子句的权重之和最小)的一组候选解的转化赋值, 并将最近的一次 CG-SLS 的转化赋值记做  $\alpha_{sls}$ , 来强调它是从局部搜索中的到一组转化赋值。当我们用  $\alpha_{sls}$  对抽象布尔骨架中的变元赋值时, “未知赋值”的变元会被随机的赋一个布尔真值。

类似于上一章 SAT 问题混合算法的设计, SMT 问题中的相位重置操作的目标也应该考虑集中性与多样性(或疏散性)两类相位。并尽可能集中于最优希望的分支产生的  $\alpha_{sls}$  来指导后续的搜索(考虑了集中性)。为了避免搜索空间的重

表 5.1 每一种相位的概率分布

Table 5.1 Probability distribution for each phase.

相位种类	$\alpha_{sls}$	$\alpha_{\perp}$	$\alpha_T$	$\alpha_{rnd}$	$\alpha_{rev}$	无改变
概率	50%	10%	10%	10%	10%	10%

复，我们引入了一些多样性相位（考虑了多样性），如下所示。

- 全  $\perp / T$  ( $\alpha_{\perp} / \alpha_T$ ). 将相位保存技术中的保存相位中每一个变元的取值倾向赋值为  $\perp / T$ 。
- 随机相位 ( $\alpha_{rnd}$ ). 将保存相位的值赋值为一个  $\{\perp, T\}$  中的随机值。
- 反转相位 ( $\alpha_{rev}$ ). 将当前保存相位中的赋值反转，即由  $T(\perp)$  变为  $\perp(T)$ 。
- 无变化. 跳过本次的相位重置操作。

借鉴了上一章中的相位重置技术，本节中，当每次 CDCL( $T$ ) 重启之后，我们都会执行相位重置操作，并按照表 5.1 中的概率分布选择不同类型的相位来执行重置操作。其中一半的概率用于集中性操作，另一半用作多样性操作。我们根据 5% 的概率粒度来调整了概率分布的取值，我们发现 SMT 问题概率分布的最优取值与 SAT 问题的最优概率分布的取值类似。并且，我们发现如果相关算法  $\alpha_{sls}$  的取值较小的时候，算法的整体性能会严重的下降（少求解超过 100 个例子）。对于多样性相位的概率分布来说，其赋值的设定鲁棒性会更高一些，即对于性能的影响较小（变动幅度在 30 个实例以内）。

### 5.3.2 基于局部搜索冲突频率的 CDCL( $T$ ) 变元分支策略

在 CDCL/CDCL( $T$ ) 算法中，另一个比较重要的算法是变元分支启发式，负责变元的选择顺序。尽管已经被提出了 20 多年，VSIDS<sup>[42]</sup> 仍然是最有效的一类启发式。其启发式策略的本职为在目前已有的知识下，选择一个最有可能导致冲突的变元作为分支变元，寄希望于尽快的通过冲突来对搜索空间进行剪枝。本节中，我们提出了利用局部搜索的冲突频率（conflict frequency）来改进 Z3 的 CDCL( $T$ ) 的 VSIDS 策略。

**定义 5.1.** 在一个执行了  $n$  步的 SMT 局部搜索过程中，一个原子公式  $a$ （可对应一个抽象命题骨架中的变元）的冲突频率  $cf(a)$ ，是  $a$  出现在不可满足子句中的步数占总步数的比例。

当前, CDCL(T) 的分支启发式 (如 VSIDS) 的评分函数是在选择每个变量后估计发生冲突的可能性。变量的评分越高, 它出现在冲突中的概率就越大。冲突频率是一种计算给定变量冲突发生比例的方法, 在一定程度上可以看作是 CDCL(T) 分支启发式评分函数的补充。如算法 6 所示, 在每次 I-SLS 和每  $\delta$  次 CG-SLS 之后, 对于输入公式的命题骨架中的每个变量  $a$ ,  $a$  的活跃度分数 (activity) 增加了  $\zeta \times cf(a)$ , 其中  $\zeta$  是一个参数, 其目的是在从局部搜索收集的  $cf(a)$  信息和 CDCL(T) 收集的冲突信息之间取得一个平衡。在本文中,  $\delta = 20$ ,  $\zeta = 200$ 。请注意, 我们没有将  $\delta$  设置为 1, 因为 CG-SLS 调用数量远远大于 I-SLS 调用数量, 如果我们将收集到的两个局部搜索求解器的冲突频率视为相同, 那么局部搜索对 CDCL(T) 的影响将主要由 CG-SLS 调用中提供的冲突频率主导。

## 5.4 实验评估

我们进行了大量的实验来评估 HybridSMT 的性能, 其目的是回答以下研究问题 (RQ):

- **RQ1:** 在 SMTLIB-IA 测试集合 (具体包含 NIA 与 LIA 两组) 中, HybridSMT 是否超越了其他最先进的 (SOTA) 求解器?
- **RQ2:** 所有提出的方法是否都是有效的?
- **RQ3:** 局部搜索在与 CDCL(T) 的协作过程中扮演了何种角色?

接下来, 我们将对实验环境进行介绍。

**实验环境** 本章的实验环境与第 3 章环境一致。求解器的截止时间与 SMT 比赛的规则一致, 每个求解器在 1200 秒内仅有一次机会求解每个实例。

**基准测试实例集合** 我们使用由 SMT-LIB 提供的标准 SMT (NIA) 问题、SMT (IA) 问题基准测试实例集合, 该组实例可从 <http://smtlib.cs.uiowa.edu/> 下载。这一权威基准测试集通常用于评估每年 SMT 比赛 (SMT-COMP) 中 SMT 求解器的性能<sup>[271]</sup>。该基准测试集包含丰富的工业实例, 其中大多数实例在软件工程领域具有实际应用。在此, 我们介绍了一些比例相对较高的类别。

- “ITS”、“SAT14” 和 “CInteger” 家族是由 VeryMax 求解器生成, 并在文献<sup>[272]</sup> 中提出了相关方法, 主要用于解决终止证明问题<sup>[212]</sup>。

- “AProVE” 实例家族是通过一个自动程序终止验证工具生成的<sup>[53]</sup>。
- “LassoRanker” 和 “UltimateLassoRanker” 系列样例由一种专用工具生成，该工具分析了 lasso-shaped 程序的终止和非终止性问题<sup>[35,273]</sup>。
- “UltimateAutomizer” 和 “UltimateAutomizer2023” 系列<sup>1</sup>是通过将自动软件验证工具应用于 SV-COMP (软件验证比赛) 的基准测试而生成的。<sup>[274]</sup>
- “Leipzig” 家族来源于 J. Waldmann 提供的利用矩阵解释进行程序终止分析的应用场景。
  - “calypto” 系列是由 “Calypto Design Systems 公司” 在时序等价性检查的背景下生产的一组样例。
  - “Math” 家族是由幻方、丢番图方程和出租车数生成的。
  - “Dartagnan” 家族起源于并发安全性的分析，并在弱内存模型下检查状态可达性。<sup>[275]</sup>
  - “MCM” 家族是通过使用伪布尔满足性求解多重常数乘法 (MCM) 问题的过程中得到的。

**实现** 本文实现了两个求解器：HybridSMT 和 Hybrid+Z3。其中 Hybrid+Z3 是 CDCL(T) 策略和 Z3 中许多其他 IA 算法的顺序组合。实现上，Hybrid+Z3 将 Z3 的 CDCL(T) 策略替换为了我们的混合求解技术 HybridSMT；另一个角度上讲，HybridSMT 禁用了 Hybrid+Z3 中除了我们的混合方法之外的所有 IA 算法。

我们的求解器均是用 C++ 实现的，并使用 ‘-O3’ 优化选项，并用 g++ 编译。HybridSMT 的参数是手动调整的，而其竞争对手则由各自的作者在 SMT-COMP 基准测试上进行了调整。初步实验表明，我们的方法对参数并不敏感。本章涉及的代码和实验数据可以从 GitHub 仓库中取得<sup>2</sup>。

## 指标与作图

- #Ins 表示在数据集中实例的数量。
- #S (#U) 表示已解决的可满足 (不可满足) 实例的数量，全部样例的求解情况用 #A = #S + #U 表示。

<sup>1</sup>为了节省空间，“UltimateLassoRanker”、“UltimateAutomizer” 和 “UltimateAutomizer2023” 分别记为 “ULasso”、“UAuto” 和 “UAuto23”。

<sup>2</sup><https://github.com/hybridsmt/hybridsmt>

- #B 表示与基础求解器相比, HybridSMT 能够唯一求解的实例数量。
- #ByI (或 #ByG) 是指通过 I-SLS (或 CG-SLS) 求解的实例数量。
- #Ct(I) (或 #Ct(G)) 是 I-SLS (或 CG-SLS) 的平均调用次数。
- T(I) (或 T(G)) 是 I-SLS (或 CG-SLS) 运行时的平均百分比。

CDF (cumulative distribution functions) 图显示了随着时间的推移所解决的实例数量, 曲线越高, 表示求解器性能越好。本文采用散点图来比较 HybridSMT 与其他求解器的求解时间 (单位为秒)。坐标轴以对数尺度显示。x 轴 (或 y 轴) 表示的求解器更适合于绘点位于对角线之上 (或之下) 的实例。标有橙色叉号 (或蓝色圆圈) 的点表示可满足 (或不可满足) 的实例。请注意, 在图中水平或垂直绘制的连续点是由于策略或求解引擎的切换所导致的。

#### 5.4.1 与前沿求解器的对比

根据 SMT-COMP 中的参与情况和相关排名, 我们选择了四个前沿的求解器作为我们的对比选手, 他们分别为:

- Z3 (version 4.8.17)<sup>[37]</sup>. 它是一个串行的策略组合算法, 包含了 CDCL(T)、位展开和 NLSat<sup>[204]</sup> 等算法。
- MathSAT5 (version 5.6.8)<sup>[166]</sup>. 它主要采用了增量线性化技术<sup>[209]</sup> 来求解 SMT(NIA) 问题。
- Yices2 (version 2.6.4)<sup>[39]</sup>. 它主要采用 MCSat/NLSat 算法来求解 SMT(NIA) 问题。
- CVC5 (version 1.0.2)<sup>[38]</sup>. 它同样也是一个串行的策略组合算法, 包含了位展开、增量线性化等技术。

除了上述的求解器之外, 我们也考虑了 maxsmt, omt, ninc, cores, 和 ninc-cores<sup>[212]</sup> 五个不完备求解器, 他们都是基于增量线性化的 barcelogic<sup>[276]</sup> 求解器的变种算法<sup>3</sup>。这些求解器是在解决 (非) 终止性问题时提出的, 并在过程中产生了 “ITS”, “SAT14” 和 “CInteger” 等系列实例, 数量上占据了 SMTLIB-NIA 实例集 81.7% 比例的数量。

此外, 我们也提供了两个 VBS (Virtual Best Solver) 求解器, 其求解能力是一组求解器最好求解能力的累加, 即对于每一个实例, 其求解效果视作该组求解

<sup>3</sup>实验表明其中有 13 个例子 barcelogic<sup>[212]</sup> 系列求解器汇报结果是 ‘UNSAT’ 但是其它所有算法的汇报结果均为 ‘SAT’。因此, 这几个例子在 barcelogic 系列求解器求解时都被视为是 “结果未知” 的。

**表 5.2 与近期 SMT 比赛中 5 个主流求解器在 LIA 样例的实验结果****Table 5.2 Compared to four state-of-the-art solvers on SMT-LIB LIA instances**

求解器	HybridSMT	LocalSMT	Z3	MathSAT5	Yices2	CVC5
#S(#Ins=8512)	<b>8161</b>	7390	8002	8061	7656	7008

器中效果最好的求解器的效果。 $VBS_1$  的计算考虑了本章中所有 SMT (NIA) 求解器;  $VBS_2$  的计算只考虑了 Z3 的 CDCL(T) 求解器与本章中提及的局部搜索求解器 LocalSMT。

#### 5.4.1.1 与主流求解器在 LIA 实例集上的对比结果

本节首先展示了 Hybrid+Z3 与 5 个近期在 SMT 比赛 LIA 赛道上排名靠前的求解器 (MathSAT5、CVC5、Yices2、Z3、LocalSMT) 的对比结果。

根据表 5.2 中的数据, HybridSMT 凭借混合求解器策略, 可以在该组样例上达到最好的求解效果, 且可以多求解出 45 个两个基求解器无法求解的实例。实验结果证明了混合算法的有效性。但是由于该组样例绝大部分例子均可求解 (仅有大约 4.1% 的样例无法求解) 且混合算法与主流求解器的差异不大 (可以比 MathSAT5 多求解器 1.2% 的样例), 该组实例集不利于细粒度的展示和分析, 因此本章后续的讨论, 我们均默认采用 SMT-LIB 中 NIA 实例集进行分析研究。

#### 5.4.1.2 与主流求解器在 NIA 实例集上的对比结果

我们也将 HybridSMT 与主流的几个求解器进行

- 总体而言, HybridSMT 显著优于其竞争对手。具体地, HybridSMT 相比 Z3、MathSAT5、Yices2、CVC5 和 ninc-cores, 可以多求解 1458 个 (7.43%), 3207 个 (17.95%), 4629 个 (28.15%), 6811 个 (47.76%), 以及 3332 个 (18.78%) 实例。根据每个类别的结果, 我们了解到对于终止和非终止验证问题, HybridSMT 是最佳选择。

- 在 #U 问题上的表现最佳, 平均比其五个竞争对手多解决了 1749.2 个 (42.5%) 实例。这些结果再次证实了其有效性。

- 在 #S 上, HybridSMT 排名第二, 略差于 ninc-cores。这主要是由于 HybridSMT 在 “Math” 类别上的表现较差, 而该类别实例更适合使用位展开方法求解<sup>4</sup>。当忽略“Math”类别时, 几乎所有剩余的实例都来自软件工程应用, HybridSMT

<sup>4</sup>通过将 Z3 中 CDCL(T) 的组件替换为我们的 HybridSMT, 而形成的多策略求解器 (Hybrid+Z3) 可以

表5.3 与近期SMT比赛中4个主流求解器和barcelogic最佳版本的对比

Table 5.3 Compared to four state-of-the-art solvers shown in recent SMT-COMP and the best solver in barcelogic

测试集	#Ins	VBS <sub>1</sub>			HybridSMT			Z3			MathSAT5			Yices2			CVC5			minc-cores		
		#S	#U	#A	#S	#U	#A	#S	#U	#A	#S	#U	#A	#S	#U	#A	#S	#U	#A	#S	#U	#A
AProVE	2409	1663	733	2396	1652	696	2348	1658	697	2355	1647	565	2212	1593	711	2304	1373	622	1995	<b>1663</b>	245	1908
Crypto	177	80	97	177	78	97	175	<b>80</b>	95	175	79	90	169	79	96	175	79	94	173	<b>80</b>	<b>97</b>	177
CHteger	1818	863	707	1570	<b>857</b>	<b>686</b>	<b>1543</b>	771	510	1281	717	458	1175	520	463	983	323	378	701	849	140	989
Dartagnan	374	13	341	354	13	<b>341</b>	<b>354</b>	13	<b>341</b>	<b>354</b>	<b>13</b>	326	339	9	319	328	<b>13</b>	324	337	0	148	148
ezsmt	8	8	0	8	8	0	8	8	0	8	<b>8</b>	0	8	8	0	8	<b>8</b>	0	8	0	0	0
ITS	17046	9686	4712	14398	<b>9594</b>	<b>4613</b>	<b>14207</b>	8540	3977	12517	7784	3794	11578	6816	3557	10373	5502	3172	8674	9428	2314	11742
LassoRanker	106	4	102	106	4	<b>102</b>	<b>106</b>	4	<b>102</b>	<b>106</b>	4	101	105	4	85	89	4	93	97	4	89	93
LCTES	2	0	2	2	0	2	2	0	2	2	0	1	1	0	0	0	0	1	1	0	0	0
SAT14	1926	1853	73	1926	<b>1853</b>	<b>73</b>	<b>1926</b>	1852	68	1920	1801	67	1868	1840	66	1906	1802	72	1874	<b>1853</b>	63	1916
sqrtnodinv	27	0	11	11	0	<b>10</b>	<b>10</b>	0	<b>10</b>	<b>10</b>	0	0	0	0	0	0	0	1	1	0	0	0
ULasso	32	6	26	32	6	25	31	6	<b>26</b>	<b>32</b>	6	<b>26</b>	<b>32</b>	6	<b>26</b>	<b>32</b>	6	<b>26</b>	<b>32</b>	6	<b>26</b>	<b>32</b>
UAuto	7	0	7	7	0	7	7	0	7	7	0	7	7	0	7	7	0	7	7	0	1	1
UAuto23	58	8	13	21	7	3	10	2	2	4	7	<b>10</b>	<b>17</b>	5	0	5	<b>8</b>	6	14	5	0	5
Leipzig	167	160	4	164	157	2	159	<b>159</b>	1	<b>160</b>	128	2	130	101	1	102	90	2	92	155	<b>4</b>	159
Math	1100	687	7	694	100	7	107	<b>659</b>	7	<b>666</b>	203	7	210	112	7	119	227	<b>7</b>	234	550	0	550
MCM	186	84	5	89	<b>78</b>	0	<b>78</b>	15	1	16	13	0	13	11	0	11	16	<b>4</b>	20	19	0	19
总计	25443	15115	6840	21955	14407	<b>6664</b>	<b>21071</b>	13767	5846	19613	12410	5454	17864	11104	5338	16442	9451	4809	14260	<b>14612</b>	3127	17739

在可满足性问题上的表现最佳。

#### 5.4.1.3 基于散点图的求解时间比较

求解器的时间是另一个需要考虑的重要指标。因为样例集合中的数量过多，平均值难以反映实际的分布规律，因此我们利用散点图，绘制了每一个样例的散点图，成对的对比了图 5.2 的五个子图 (a)-(e) 展示了 HybridSMT 与其主流竞争对手之间的运行时比较。散点图说明了两种求解器所解决实例的相对求解时间。

我们可以从中得出两个结论。首先，在可满足的实例上，HybridSMT 比 SMT-COMP 中的四个竞争对手要快得多；然而，在不可满足的实例上，HybridSMT 稍微慢一些，但它能解决更多的实例。总体而言，由于引入了局部搜索，其平均求解时间略低于基础 CDCL(T) 求解器。其次，在某些情况下，ninc-cores 能够在一秒内迅速求解一系列例子。这促使我们在设计组合时，在初始求解阶段可以采用与 ninc-cores 类似技术的求解器进行协作，以利用它们之间的互补优势。

#### 5.4.1.4 利用我们的混合方法改进 Z3

除了广受欢迎的 CDCL(T) 算法外，Z3<sup>[37]</sup> 还实现了其他强大的方法，如位展开<sup>[263]</sup> 和 NLSat<sup>[204]</sup>。前文已经展示了我们混合方法的强大性能，我们很感兴趣，想知道我们的方法是否能与其他方法进一步合作，并通过简单的组合获得进一步的提升。

表 5.4(92 页)<sup>5</sup> 显示了 Hybrid+Z3 的结果。Hybrid+Z3 比 HybridSMT 解决多出了 575 (或 573) 个 #S (或 #A) 实例，并且在 #U 上的表现相当，其中改进主要来自 ‘Math’ 和 ‘MCM’ 家族。Z3 在 “AProVE” 家族上表现优于我们的求解器，是因为这些实例包含许多布尔文字。这是因为 LocalSMT 在处理布尔文字方面并不强，而 Z3 中的一些方法，如 CDCL(T) 和位展开，在布尔推理方面表现突出。图 5.2h 总结了由 HybridSMT 和 Hybrid+Z3 解出的实例的求解时间。在这类实例中，两种求解器的性能相当，证实了 HybridSMT 与其他方法配合良好。

**使用 CDF 进行运行时分析** 为了更好地理解已求解的实例数量随着时间增长的变化趋势，我们绘制了比较 HybridSMT 和 Hybrid+Z3 与五个 SOTA SMT(IA) 求解器的累积分布函数 (CDFs)。从图 5.3 中的三个 CDF，我们了解到我们的求解器借助 Z3 的位展开技术来帮助提高在“Math”类别样例上的求解能力。

<sup>5</sup>为了节约空间，样例数目少于 200 的家族没有被单独汇报，并被汇总于 Other 类别中。

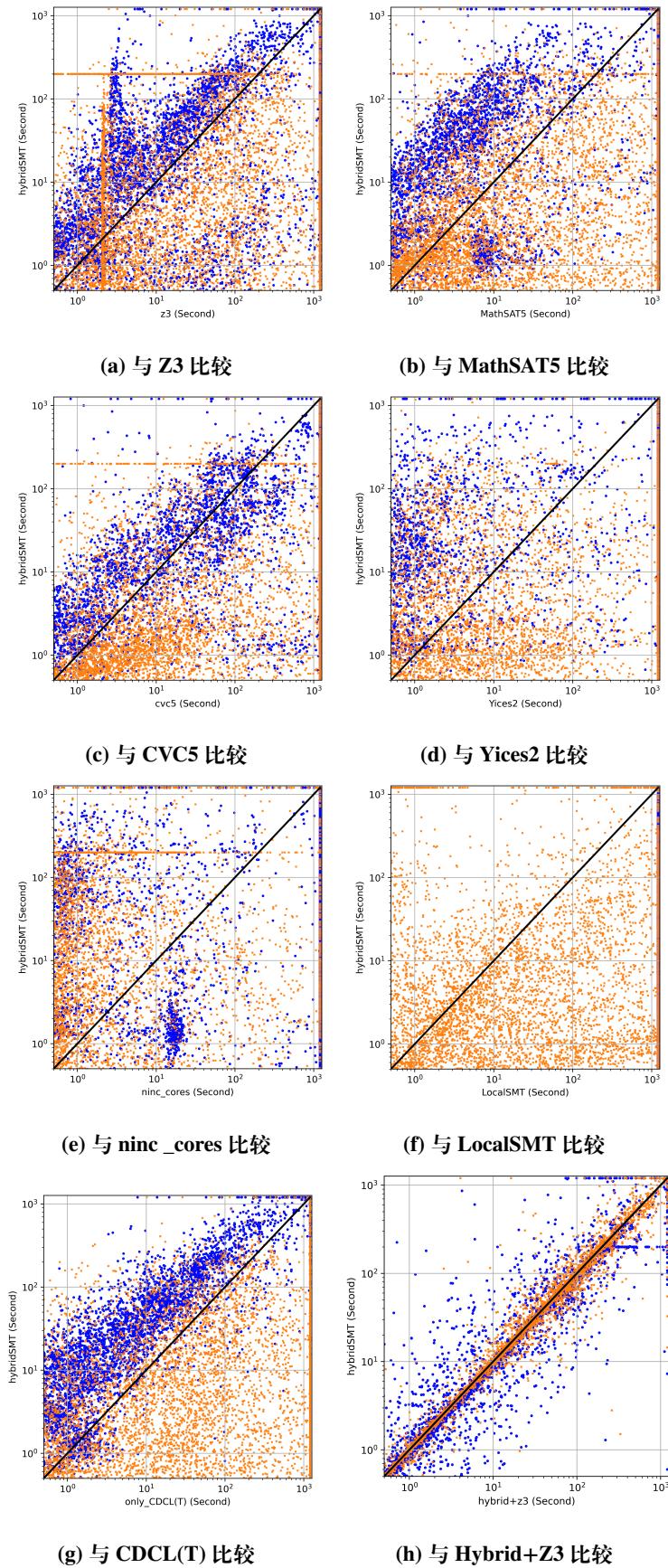


图 5.2 HybridSMT (Hybrid+Z3) 与主流 SMT(IA) 求解器、基求解器间的比较

Figure 5.2 Comparing HybridSMT (Hybrid+Z3) to SOTA SMT(IA) solvers and base solvers

**表 5.4 Hybrid+Z3: 利用 HybridSMT 改进 Z3 求解器****Table 5.4 Hybrid+Z3: Improve Z3 by HybridSMT**

测试集	Hybrid+Z3			HybridSMT			Z3		
	#S	#U	#A	#S	#U	#A	#S	#U	#A
AProVE	1655	<b>697</b>	2352	1652	696	2348	<b>1658</b>	<b>697</b>	<b>2355</b>
CInteger	854	<b>691</b>	<b>1545</b>	<b>857</b>	686	1543	771	510	1281
Dartagnan	13	341	354	13	341	354	13	341	354
ITS	9589	4606	14195	<b>9594</b>	<b>4613</b>	<b>14207</b>	8540	3977	12517
SAT14	<b>1853</b>	<b>73</b>	<b>1926</b>	<b>1853</b>	<b>73</b>	<b>1926</b>	1852	68	1920
Math	<b>677</b>	7	<b>684</b>	100	7	107	659	7	666
MCM	<b>82</b>	0	<b>82</b>	78	0	78	15	<b>1</b>	16
Other	259	247	506	<b>260</b>	<b>248</b>	<b>508</b>	259	245	504
汇总	<b>14982</b>	6662	<b>21644</b>	14407	<b>6664</b>	21071	13767	5846	19613

解决了最多的 #S、#U 和 #A。该研究表明，混合方法不仅提高了 #S，还提高了 #U。一个合理的解释是，更好的分支（顺序和阶段）启发式有助于 CDCL(T) 找到更多高质量的子句，并更快地解决更多的冲突，从而实现更快的剪枝。参见文献<sup>[43]</sup>。

值得提及的是，SMTLIB 标记了 5212 个 SMT(NIA) 实例，这些实例的标签为“未知”，这表明它们具有高度的挑战性。HybridSMT 可以在这些“未知”实例中求解出 260 个可满足实例，1810 个无法满足实例。Hybrid+Z3 可以在这些“未知”实例中求解出 260 个可满足实例，1811 个无法满足实例。这进一步证实了我们方法的有效性。

#### 5.4.2 组件策略有效性分析

本节通过将 HybridSMT 与其基础求解器进行对比，以及执行了大量的消融分析，来分析这些技术的有效性。

##### 5.4.2.1 比较 HybridSMT 与其基础求解器

我们比较了 HybridSMT 与 Z3 的 CDCL(T) 策略和 LocalSMT，结果列于表 5.5 中，运行时的比较在图 5.2f 和 5.2g 中给出。

在两者都能解决的实例上，HybridSMT 的平均速度是 Z3 的 CDCL(T) 策略的 3.4 倍。在可满足的实例上，CDCL(T) 的性能可以显著提升 12.2%。在不可满足的实例上，求解时间和 #U 数量受到局部搜索分配的影响而有一定下降，因为

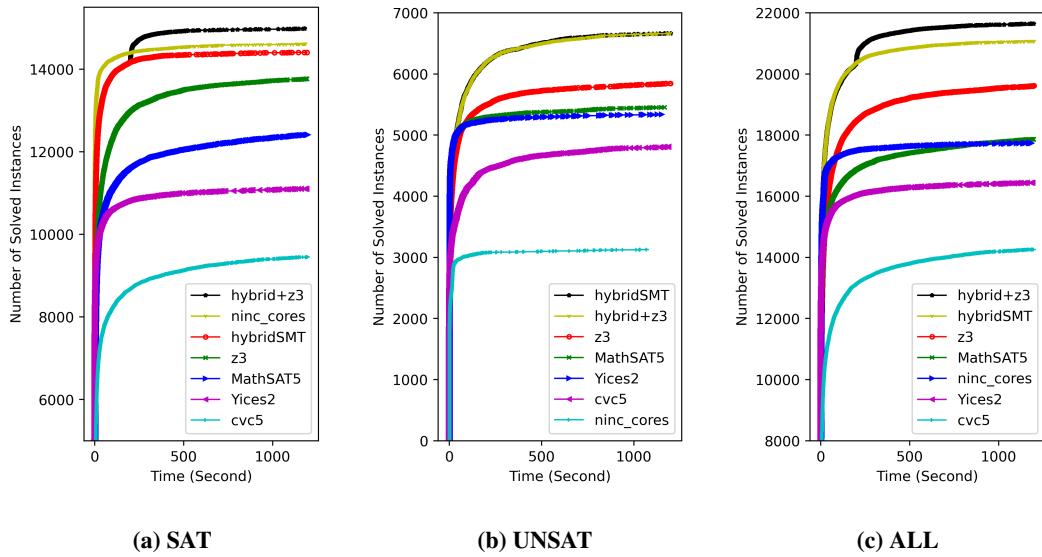


图 5.3 HybridSMT 与 Hybrid+Z3 与其他主流求解器在 SAT、UNSAT、ALL 样例上的 CDF 图

Figure 5.3 CDFs for SAT, UNSAT, ALL instances for comparing HybridSMT and Hybrid+Z3 with SOTA solvers

表 5.5 混合方法的有效性分析

Table 5.5 Effectiveness of the hybrid methods

测试集	HybridSMT			CDCL(T)			LocalSMT
	#S	#U	#A	#S	#U	#A	#S / #A
AProVE	<b>1652</b>	<b>696</b>	<b>2348</b>	1643	695	2338	1620
CInteger	<b>857</b>	<b>686</b>	<b>1543</b>	717	681	1398	790
Dartagnan	<b>13</b>	<b>341</b>	<b>354</b>	<b>13</b>	<b>341</b>	<b>354</b>	0
ITS	9594	4613	<b>14207</b>	8273	<b>4638</b>	12911	<b>9627</b>
SAT14	<b>1853</b>	<b>73</b>	<b>1926</b>	<b>1853</b>	<b>73</b>	<b>1926</b>	1478
Math	<b>100</b>	<b>7</b>	<b>107</b>	<b>100</b>	<b>7</b>	<b>107</b>	0
MCM	78	0	78	15	<b>1</b>	16	<b>86</b>
Other	<b>260</b>	248	<b>508</b>	228	<b>249</b>	477	245
汇总	<b>14407</b>	6664	<b>21071</b>	12842	<b>6685</b>	19527	13846

该策略不太适合处理此类实例。得益于成功的局部搜索调度方法和局部搜索信息，调用局部搜索对不可满足实例的负面影响可以忽略不计。

与 LocalSMT 相比，HybridSMT 可以多解决 561 个可满足实例，且对于可解实例的平均求解时间大约快了 3.12 倍。

表 5.6 组件的高效性分析

Table 5.6 Effectiveness of each component.

测试集	HybridSMT	V1	V2	V3	V4	V5
AProVE	2348	2344	2350	2335	2341	<b>2351</b>
CInteger	<b>1543</b>	1518	1538	1534	<b>1543</b>	1538
Dartagnan	354	354	354	354	354	354
ITS	<b>14207</b>	14081	13894	14047	13748	14109
SAT14	<b>1926</b>	<b>1926</b>	<b>1926</b>	<b>1926</b>	1925	<b>1926</b>
Math	107	107	107	107	107	107
MCM	78	<b>93</b>	14	82	83	82
Other	508	494	<b>509</b>	507	505	506
汇总	<b>21071</b>	20917	20692	20892	20606	20973

#### 5.4.2.2 组件有效性分析

为了分析 HybridSMT 中每一种技术的有效性, 我们修改了 HybridSMT , 并获得了五个变种版本, 具体如下。

- V1: 去除掉 CG-SLS。
- V2: 去除掉 I-SLS。
- V3: 使用 Z3 中默认的相位重置组件。
- V4: 禁用掉所有与相位有关的组件。
- V5: 禁用掉冲突频率相关的组件。

在表 5.6中的比较下, 我们可以了解每个组件的贡献。具体来说, 这五个变体分别比 HybridSMT 少解决了 154、379、179、465 和 98 个实例。而我们的方法主要对名为 “CInteger”、“ITS” 和 “SAT14” 的三个终止验证家族做出了贡献。

我们使用了一个 R 包 (<https://github.com/b0rxa/scmamp>) 来绘制关键差异 (Critical Difference, 简称 CD) 图<sup>[277]</sup>。该图利用 Friedman 检验<sup>[278]</sup>在假设算法具有相同性能的条件下评估了 5 个变体的统计差异。图 5.4显示, 这 5 个变体之间存在关键差异, 其显著性水平为 0.01, 并给出了平均性能排名。排名越低, 贡献越大。如果两个求解器之间的距离超过左上 CD 栏, 则它们之间存在相当大的差异。我们可以得知, I-SLS 和相位重置对解决实例的数量贡献最大, 而 CG-SLS 对排名 (考虑到求解时间) 贡献最大。

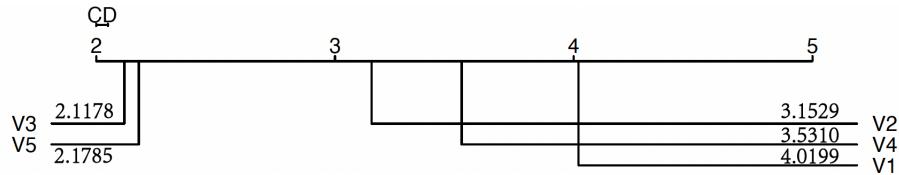


图 5.4 五个不同变种的关键差异图

Figure 5.4 Critical difference diagram about 5 variants.

### 5.4.3 两种引擎间的协作性分析

为了更好地理解局部搜索的作用，我们进行了额外的分析实验，其结果总结于表 5.7 中。

我们观察到有 164 个实例可以被混合后的求解器 HybridSMT 单独求解，而不能通过局部搜索和 CDCL(T) 的简单组合来解决。即便是一个虚拟的最佳求解器（即  $VBS_2$ ），为每个实例选择具有最佳结果的求解器，也会在这些实例上失败。

大约有 66% 的可满足实例是通过内部局部搜索调用解决的。在每次成功运行中，大约进行了 1.8 次 I-SLS 调用和 88.1 次 CG-SLS 调用，分别占运行总时间的 39% 和 16%。在解决不可满足的实例时，局部搜索调用占用了更多的求解时间，但每次局部搜索调用的代价远小于解决可满足实例时的调用代价。在不可满足的实例上，每个 I-SLS（或 CG-SLS）调用的平均运行时间大约是可满足的实例上的 3.6 倍（或 7.0 倍）。

<sup>5</sup> 我们进行了一项额外的统计收集实验，其性能与未进行信息收集的求解器实验结果类似。

表 5.7 分析 HybridSMT 中内部局部搜索求解器的作用  
 Table 5.7 Analysis for the role of internal local search solver in HybridSMT

测试集	#S	VBS <sub>2</sub> (S)	#B(S)	对于 SAT 实例				对于 UNSAT 实例								
				#ByI	#ByG	#Ct(I)	#Ct(G)	T(I)%	T(G)%	#U	VBS <sub>2</sub> (U)	#B(U)	#Ct(I)	#Ct(G)	T(I)%	T(G)%
AproVE	1654	1662	1	893	34	0.59	0.51	5.97	1.72	696	695	4	1.3	21.16	13.5	19.02
CInteger	857	835	25	618	203	1.79	33.74	49.89	9.0	688	681	10	4.02	293.0	43.88	32.36
Dartagnan	13	13	0	0	0	0.0	0.0	0.0	0.0	341	341	0	0.06	0.03	3.75	0.28
ITS	9590	10216	67	5726	1247	1.09	13.25	46.05	5.16	4612	4638	41	3.96	318.47	38.42	33.41
SAT14	1853	1853	0	411	144	1.11	1.98	42.15	37.9	73	73	0	1.62	46.66	29.62	53.27
Math	100	100	0	1	0	1.02	1.01	22.88	22.37	7	7	0	1.29	2.0	12.65	13.98
MCM	81	86	9	81	0	2.19	42.16	68.54	10.36	0	1	-	-	-	-	-
Other	260	257	5	122	50	1.18	13.6	24.18	9.34	247	249	2	0.77	1.48	36.36	6.96
总计	14408	15022	107	7852	1678	1.09	11.63	40.71	9.42	6664	6685	57	3.34	253.44	34.4	29.32

## 5.5 本章小结

本章提出了一种针对 SMT(IA) 问题的双层混合方法，成功的将 SAT 问题的混合求解技术迁移到了整数算术理论 SMT 问题上。该方法以 CDCL(T) 为主体，外层会定期调用一个“独立的局部搜索”，另一方面内层会在找到一个有希望的骨架解时调用“CDCL(T) 指导的局部搜索”。另一方面，局部搜索也被用于改进 CDCL(T) 的启发式。具体地，本章提取并使用了局部搜索变元的冲突频率和最近一次局部搜索调用中的最佳候选解等信息来改进 CDCL(T) 算法的变元分支启发式和相位决策启发式。最终，实验表明，我们的混合求解器在 SMT-LIB 通用测试样例集合上以大幅优势领先了其它主流求解技术，在等价性验证、状态可达性等软件工程领域的应用中发挥了巨大优势，尤其是在程序的终止性与非终止性问题上有巨大的推动作用。相关求解器参与了 SMT 比赛，取得了模型验证总比分“最大领先奖”与“最大贡献奖”，并在算术理论多个细分赛道（NIA、LIA 等）取得冠军。

我们提出的方法所提出的方法可以应用于解决其他算术理论背景的 SMT 问题。在未来，我们计划深入探讨 CDCL(T) 与局部搜索之间的子句交互策略，并研究与其他高效算法间的深度混合方法，例如，NLSat 与局部搜索的深度结合，以及其他背景理论下的混合方法。



## 第 6 章 混合求解技术在 EDA 和密码分析中的应用

前面的章节中介绍了 SAT 问题和 SMT 问题的混合求解技术，并强调了 SMT 混合求解在软件工程领域中的应用价值。本章进一步介绍了 SAT 混合求解技术在 EDA 和密码分析中的应用。

第 6.1 节描述了 SAT 求解与一种概率精确仿真 (EPS) 技术的混合求解策略，并介绍了该技术在组合等价性验证问题 (CEC) 的应用。组合等价性验证 (CEC) 在确保设计优化的正确性方面起着至关重要的作用。主流的 CEC 算法为 SAT-sweeping，它会按照一定顺序，利用 SAT 求解来证明内部节点对的等价性，并将等价节点合并。虽然 SAT 求解器可以用于证明较大规模的电路，但是难以处理 XOR 链密度很高的子电路。为了解决该问题，本章设计了一种 SAT 求解与 EPS 技术的混合 Sweeping 框架。在来自工业界的实例集合中，我们的方法相比目前最好的开源求解器 ABC，求解 100 倍以上。

6.2 节介绍了混合 SAT 求解器在流密码分析中的应用。本章以“强网杯”密码数学专项赛赛题中的“环上前馈序列密码”作为研究案例，详细阐述了其建模和应用混合 SAT 求解器求解的过程。本章提出了一种“轮增量编码”的求解技术，并凭借该技术成功恢复了流密码的初态。相比其他参赛队伍中应用千核心并行求解数周的方法，我们的方法能在单核心上于数个小时完成分析任务，获得了该赛事的全国冠军。

### 6.1 基于 SAT 求解和精确仿真的组合等价性混合验证方法

组合等价性验证 (Combinational Equivalence Checking, 简称 CEC) 是指用于形式化方法证明两个电路是否是功能等价的问题，这是电子设计自动化 (EDA) 和数字 IC 设计领域中最重要的技术之一。CEC 具有广泛的应用，例如移除逻辑功能等价子电路<sup>[279]</sup>、时序等价性验证<sup>[280]</sup>，电路对称性检测<sup>[281]</sup>，工程变更单 (ECO)<sup>[282]</sup> 等。

本节选择的样例主要来自于数据通路电路。数据通路电路是由许多如乘法器、加法器和多路复用器等基本的算术单元组成的一类电路，常在计算密集型应用中出现，如微处理器设计<sup>[283]</sup>。为了获得更好的 PPA (性能、功率和面积)

性能，数据通路电路在设计时会通过逻辑综合或结构重写等技术进行优化，而 CEC 是确保优化正确性的关键工具。

最先进的 CEC 算法是基于 SAT-sweeping 的一类方法<sup>[284,285]</sup>，代表技术有 ABC 中的 ‘&cec’ 命令<sup>[286]</sup>。对于两个待测电路，该方法通过将 PI 和 PO 两两对应连接创建一个 miter 电路，如果 miter 电路的输出始终为 0，则这两个电路是等价的。这可以通过调用 SAT 求解器对转化后的公式进行检查来实现。在进行 SAT-sweeping 之前，使用逻辑仿真（Logic Simulation）来生成一系列内部代测等价点对。在 SAT-sweeping 的过程中，每个内部节点对的等价性均会通过 SAT 求解器验证，一旦验证成功，等价节点将被合并为一个节点，以加速后续的验证过程。

当扫描数据通路电路时，通常会遇到一些特殊的内部节点对，这些点对对应的 miter 电路的传递扇入锥 (TFI-cone) 具有高密度的异或链。验证这样的节点对对于现代 SAT 求解器是一个挑战，即使对于小的 TFI-cone 电路也是如此。我们调研中发现，有一种专门用于验证此类高异或链密度电路的方法，称为精确概率模拟 (EPS) 方法<sup>[287]</sup>。EPS 方法特别适用于此类小而难的电路（最多 24 个主输入）。例如，现代 SAT 求解器不能在 12 小时内证明两个 16 位乘法器之间的等价性，而 EPS 可以在 4 分钟内完成验证。然而，EPS 需要  $O(2^N)$  内存来保存每个节点的概率信号，其中  $N$  是主输入 (PI) 的数量，这极大的限制了 EPS 在实际问题中的推广，因此我们很少看到它作为一个独立的 CEC 工具被应用。

本节，我们将 EPS 加入到了 SAT-sweeping 中，以弥补 SAT 求解器在高异或链密度电路上表现不佳的弱点。此外，为了提高 EPS 的可扩展性，我们提出将过长的概率信号划分为多组较短的信号，以减少内存开销。

我们设计了一种 SAT 求解和 EPS 技术的混合 sweeping 算法。在不同的异或链密度的电路下，SAT 求解器与 EPS 引擎有明显互补性—EPS 引擎在异或链密度高的情况下大概率优于 SAT 求解器，而 SAT 求解器在异或链密度低的情况下则远远优于 EPS。此外，我们设计了一种启发式算法，可以根据节点对的 TFI-cone 的异或链密度，动态的从 SAT 求解器和 EPS 工具中选择证明引擎。

最后，数据通路电路中，通常有许多相同功能的子模块，他们在结构和拓扑顺序上具有相同的实现<sup>[288]</sup>。据此，我们设计了一种相同结构检测 (ISD) 技术来发现具有相同实现结构的子模块，跳过检测，从而减少同一种结构的重复证明，

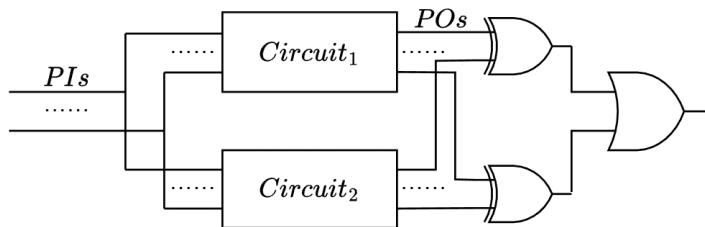


图 6.1 Miter 电路的结构示意图

Figure 6.1 The structure of miter circuits.

这使得我们的算法进一步提速。

### 6.1.1 背景知识

本节中主要介绍了一些 CEC 相关的基础知识和常见方法。

#### 6.1.1.1 基础定义

一个布尔网络 (network) 是一个有向无环图 (DAG)  $G(V, E)$ , 其中  $V$  是一组布尔变元 (节点) 的集合,  $E \subseteq V \times V$  是一组边的集合, 从中可以确定变元拓扑结构顺序。组合门级网表是一种布尔网络, 其中线路 (wire) 对应于  $V$  中的布尔变元, 而门 (gate) 则对应于由  $E$  中的多条边表示的布尔函数。在本文中, 术语“网络”和“网表”都表示电路。

每个布尔网络都有主输入 (primary input, 简称 PI) 和主输出 (primary output, 简称 PO)。给定一条边  $e = (u, v)$ ,  $u$  是  $v$  的扇入 (fan-in), 而  $v$  是  $u$  的扇出 (fan-out)。节点  $v \in V$  的传递扇入 (transitive fan-in, 简称 TFI) 锥 (cone) 是在  $E$  的拓扑关系下终止于  $v$  的所有节点与对应的变元张成的电路, 可以看作是电路的一个子模块。

本文中的所有样例均为与-非图 (AIG) 格式的门级网表, 其中每个节点都是一个双输入与门, 并且该门的每个输入都可能关联一个非门。AIG 是一种通用且整洁的格式, 可以轻松地表示任意布尔网络。

组合等价性验证 (CEC) 是检查两个给定的组合电路是否在功能上等价的问题。在本文中, 我们关注于检查两个数据通路电路的等价性, 这些电路由许多算术单元组成, 如乘法器、加法器、多路复用器等。由于设计中存在大量的“与-异或”项, 因此这类电路的 CEC 通常非常困难<sup>[289]</sup>。

给定两个具有相同数量 PI 和 PO 的电路，一个 *miter* 电路<sup>[290]</sup> 由要比较的两个电路通过成对连接 PI 和 PO 产生。如图6.1所示，相匹配的 PI 直接连接，而相匹配的 PO 则通过 XOR 门连接，并且新插入的 XOR 门的输出由一个 OR 门连接。对于给定的 miter，如果 miter 右侧的 OR 门输出始终为常量 0，则原始的两个电路是等价的。

在证明电路等价性的时候，一般会采用形式化验证工具来辅助证明。一个 AIG 电路可以很自然的编码到命题逻辑 CNF 公式上，其中主要采用的技术为 Tseitin 编码<sup>[60]</sup>。除了 SAT 求解器之外，另外的一类常见的推理工具为二叉决策图（binary decision diagram，简称 BDD）知识编译器，代表工具为 KCBox<sup>[291]</sup>，其输入也为 CNF 公式。BDD 编译器在小规模的样例上有不错的表现，并于 2000 年之前盛行；但是 BDD 编译器对于验证超大规模的算术电路来说，也会受到内存大小的限制。相比之下，SAT 求解器由于其拓展能力强，逐渐成为了验证大规模电路上唯一的工具。此外，也有学者对 SAT 问题和 BDD 问题的结合开展了研究<sup>[292]</sup>。

### 6.1.1.2 基于 Sweeping 的 CEC 工作流

当前 CEC 引擎最流行的方法是基于扫描的方法，该方法的典型流程如图6.2所示。给定两个待验证的电路，第一步是构建一个 miter 电路  $M_o$ ，然后 CEC 主引擎会从检验两个电路等价性的“ $M_o$  证明阶段”和对待测电路进行化简的“ $M_o$  化简阶段”两个阶段间不断的切换。

在  $M_o$  化简阶段， $M_o$  会通过逻辑综合<sup>[293]</sup>、结构 Hash<sup>[292]</sup>、功能性 AIG 化简 (FRAIG)<sup>[294]</sup>、BDD-sweeping<sup>[292]</sup> 或 SAT-sweeping<sup>[295]</sup> 等技术化简。其中 Sweeping 是最出名的技术，相关的示意图6.3如图所示。它会首先通过逻辑仿真（logic simulation）识别初潜在等价点对  $(a_1, b_1), \dots, (a_n, b_n)$ ，然后依次采用逻辑推理工具（SAT 或者 BDD 工具）检测每一组待测等价点对  $(a_i, b_i)$  的等价性。一旦一个等价点对  $(a_i, b_i)$  被证明是等价的，那么这两个节点就会被合并为同一个节点。最终  $M_o$  会被化简为一个规模更小的电路，证明的难度也会相经述减。

逻辑仿真时 Sweeping 的一个预处理程序，用于将可能等价的节点分组到不同组中。它会随机的为所有的 PI 赋值，并将这些值传播到 PO 上。根据仿真过程中每个节点相关联的值，我们可以大致了解内部节点的逻辑行为，并迅速排除非等价的节点对。最终保留下的点对即为待测等价点对。有时，逻辑仿真可以直

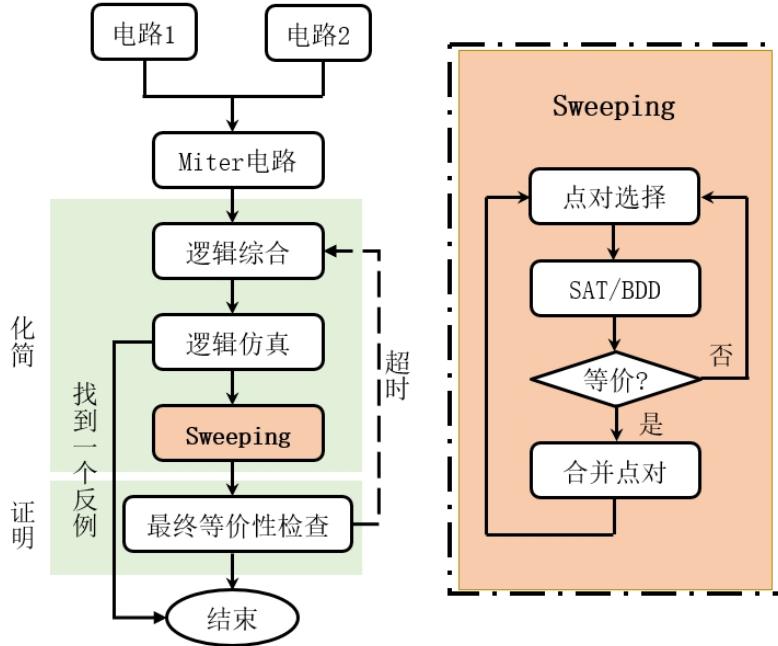


图 6.2 基于 Sweeping 的 CEC 典型算法流程图

Figure 6.2 Framework for a Typical Sweeping Based CEC Algorithm.

接找到两个非等价电路的反例。另外，分配给 PI 的值可以根据 Sweeping 过程中推理引擎的信息启发式地生成<sup>[285]</sup>。

### 6.1.1.3 精确概率仿真

与应用和传播布尔值的逻辑仿真不同，基于概率的仿真方法<sup>[287,296]</sup>会为每个 PI 赋值一个概率信号  $p \in [0, 1]$ ，并在 PO 处捕获该相关的概率信号值。具体的，对于非门来说，如果输入概率为  $p$ ，则输出概率为  $1 - p$ ；对于双输入与门，如果两个输入的概率为  $p_1$  和  $p_2$ ，则输出概率为  $p_1 \times p_2$ 。

输出的概率可以被视为电路逻辑功能的一种体现，可以被用于识别一个网络的逻辑功能是否有区别，即在相同输入信号下，两个逻辑功能等价的电路的输出也应该是相同<sup>[296]</sup>。然而，这个过程有可能会出现一种称为混叠（aliasing）的现象，表示即便是具有不同逻辑函数的电路也有可能会共享相同的输出概率。

存在混叠的信号只可用于快速判断两个算法的不等价性；当两个网络在相同的无混叠信号赋值下输出概率相同时，两个网络才是等价的。Shih-Chieh Wu<sup>[287]</sup>等人提出了一种 PI 概率信号赋值的方法，该方法可以保证不存在上述的“混叠”现象，其提出的基于精确概率仿真（exact probability-based simulation，简称 EPS）

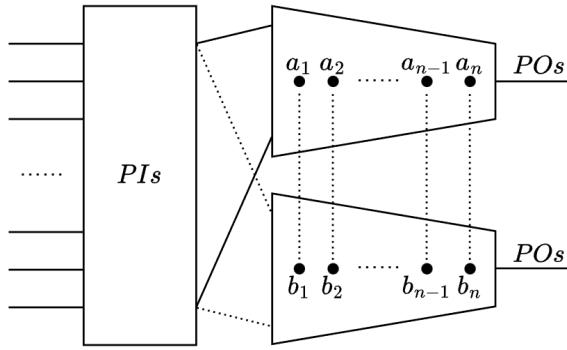


图 6.3 Sweeping 示意图

Figure 6.3 A Schematic Diagram for Sweeping.

方法描述如下：

对于一个有  $N$  个 PI 的电路，第  $i$  个 PI 的概率应该根据公式 6.1 被赋值为  $1/\theta_i$ ，其中  $\theta_1$  被赋值为 3 来更好的节省空间。

$$\theta_{i+1} \leftarrow (\theta_i - 1)^2 + 1, i \in [1, N - 1]; \quad (6.1)$$

在实际应用中，Wu 提出了一种更为有效的计算方法<sup>[287]</sup>，它采用快速位运算代替了时间开销巨大的浮点乘法操作。在为 PI 生成一组无混叠概率后，PI 概率的所有分数均被通分，因此我们可以只关注分子上的数值，该数值是适合执行位运算的长整数。然后对于 AIG 中的与门执行与位运算，对于非门执行取反位运算操作。然而，EPS 的缺点是这种信号赋值的长度会随着 PI 数量的增加呈指数级的增长。本质上，EPS 仍然是一种真值表枚举方法。具体地说，对于一个具有  $N$  个 PI 的电路，采用按位优化的方法，分子的长度为  $2^N$  位。由于内存开销较大，这种方法不适用于 PI 数目较多的电路。针对该问题，本文后续在实现中将输入向量进一步拆分为多组短输入，在减小内存开销的同时，也利用了批处理的特性。

### 6.1.2 混合求解算法设计

本节主要介绍我们的混合 CEC 求解器 hybridCEC。我们从一个例子出发阐述了我们的核心思路。然后，介绍了该方法的主要 CEC 框架，接着介绍了改进的精确仿真、引擎选择启发式和相同结构检测三个主要的创新点。

### 6.1.2.1 案例演示

在微处理器的优化应用中，经常会出现数据通路电路的等价性验证问题。为了阐明我们的动机，我们在本小节中使用了一个名为“ec\_h1”的 miter 电路，它来自于工业界高位宽运算电路设计场景。目前最著名的开源 CEC 工具 ABC 的“&cec”指令无法解决此问题。

通过应用  $2^{20}$  轮纯随机的逻辑仿真操作（小于 0.1s），我们可以选择出 113 个潜在等价点对。按照拓扑顺序，如果去除掉与前一对具有相同结构的点对后，共剩余 48 个节点对。

在典型的 SAT-sweeping 算法中，会逐次将 48 个节点对的 miter 电路转换为 CNF，并通过 SAT 求解器进行检查。我们使用主流的 SAT 求解器 kissat-MAB<sup>[227]</sup> 进行测试。（有关详细的实验设置，请参阅 6.1.4 部分）。我们在表 6.1 中给出了该样例不同点对在采用 SAT 求解器和 EPS 工具两种不同引擎时的求解时间对比。表中汇报了等价点对的拓扑序号（ID），门数目（Gates），TFI-cone 的 PI 数目（PIs）和两个求解器的求解时间。为了节省空间，我们省去了两个求解器均可以在 1 秒内求解的简单样例。其中“TO”表示一小时无法给出结论。

根据表 6.1 中的结果，有 10 个节点对对于 SAT 求解器来说太难了（我们也尝试了 MiniSat<sup>[44]</sup>，但效果更差），而它们可以通过 EPS 工具轻松验证。另一方面，有 16 个节点对无法通过 EPS 工具的验证，可以通过 SAT 求解器快速求解。从表中的结果表明 EPS 工具没有能力证明 PI 数大于 36 的电路的等价性，这主要是由于巨大的内存开销和需要枚举覆盖的搜索真值表过大导致的。

结果表明，SAT 求解器和 EPS 工具在不同的电路上是互补的。进一步观察表明，性能差异主要与异或链的密度有关。这促使我们设计一种基于混合算法的 CEC 工具，可以从 SAT 求解器和 EPS 工具两种引擎中动态的选择合适的引擎。

### 6.1.2.2 混合求解框架

我们的算法 (hybridCEC) 的框架类似于图 6.2 中给出的典型 SAT-sweeping 方法，主要区别是 miter 电路化简阶段的 sweeping 过程不同，如图 6.4 所示。

在 SAT-sweeping 过程中，我们提出了一种启发式方法来选择一个推理工具（SAT 求解器或精确概率仿真工具）来检查两个潜在等效内部点的等价性。此外，当一个节点对合并成功时，我们进一步按照拓扑顺序检查剩余的节点对；如果有

表 6.1 SAT 和 EPS 工具验证 ‘ec\_h1’ 样例的求解时间对比

Table 6.1 Runtime comparison between SAT and EPS for verifying the internal pairs of miter ‘ec\_h1’.

ID	Gates	PIs	推理引擎	
			SAT (秒)	EPS (秒)
17	268	32	<0.01	9.67
18	353	18	1.75	<0.01
19	360	36	<0.01	TO
20	452	20	5.47	<0.01
21	556	22	29.40	0.02
22	468	40	<0.01	TO
23	548	44	<0.01	TO
24	678	24	137.32	0.08
25	658	48	<0.01	TO
26	807	26	903.51	0.65
27	768	52	<0.01	TO
28	1097	30	TO	13.30
29	950	28	TO	2.94
30	1423	32	TO	66.41
31	1310	64	<0.01	TO
32	1022	60	<0.01	TO
33	896	56	<0.01	TO
34	1259	32	TO	59.64
35	2018	32	TO	90.56
36	1734	32	TO	79.45
37	1580	32	TO	73.27
38	1832	64	<0.01	TO
39	1580	64	<0.01	TO
40	1446	64	<0.01	TO
41	1168	64	<0.01	TO
42	2262	32	TO	132.24
43	2144	32	TO	122.90
44	1879	32	TO	84.97
45	2046	64	<0.01	TO
46	1942	64	<0.01	TO
47	1708	64	<0.01	TO
48	4280	96	<0.01	TO

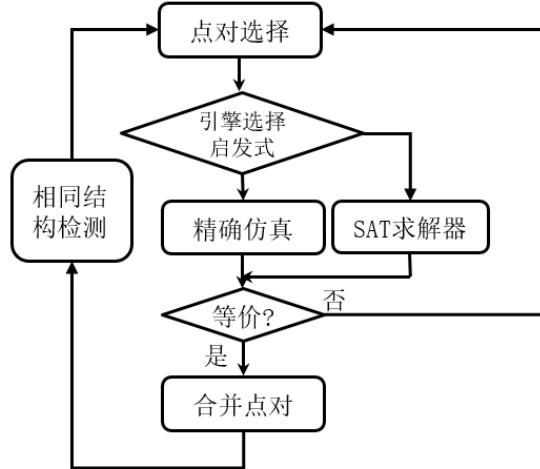
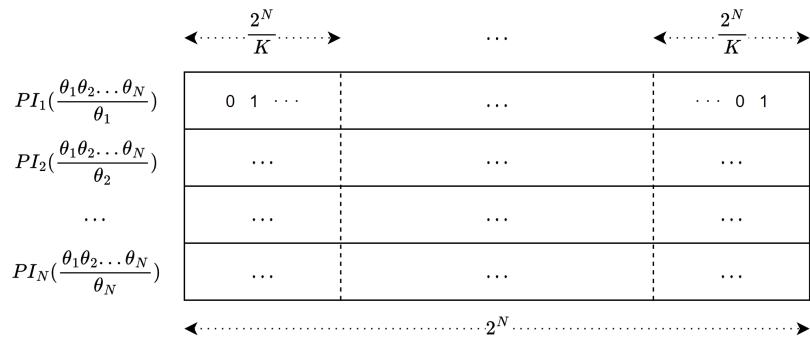


图 6.4 CEC 算法的主框架算法

Figure 6.4 Main Framework for Our CEC Algorithm.

图 6.5 将 PI 概率信号取值分割成  $K$  个小的数Figure 6.5 Truncate the PI values into  $K$  blocks of small values.

一对  $(a_i, b_i)$  与新合并的等价对处于同一实现中，然后可以立即合并  $a_i$  和  $b_i$ 。这种技术称为相同结构检测（identical structure detection，简称 ISD）。因此，提出了一种结合 SAT 求解和精确仿真的高效 CEC 算法，称为 hybridCEC。

### 6.1.2.3 内存优化的 EPS 算法

如上文所述，Wu 等人提出的 EPS<sup>[287]</sup> 算法会将第  $i$  个 PI 赋值为  $1/\theta_i$ 。对于一个包含  $N$  个 PI 的电路来说，通分之后第  $i$  个值会变成一个长整数  $\theta_1\theta_2\dots\theta_N/\theta_i$ 。

通过我们分析，每一个 PI 的二进制表示有很明显的结构化规律。图6.5中每一行表示 PI 的布尔取值下的赋值。总的来说，每行有  $2^N$  可能的赋值模式，也就是说，为了保证无混叠，原始的 EPS 方法为每个节点分配一个宽度至少为  $2^N$  的整数。当 PIs 的数量  $N$  较大时，对每个 PI 执行一个完整整数的仿真会造成巨

**Algorithm 7:** 改进的精确概率仿真 (EPS) 算法

**输入:** Miter 电路  $M$ ,  $PI = \{i_0, i_1, \dots, i_k\}$ ,  $PO = o$

**输出:** 求解结果 ‘等价’ 或者 ‘不等价’

```

1 gate_list  $\leftarrow$  topological_sort( $M$ )
2  $l \leftarrow \min\{|PI|, bits\_limit\}$ 
3  $R \leftarrow 2^{|PI|-l}$ 
4 for  $i$  in  $1, \dots, R$  do
5   for each  $p_j$  in  $PI$  do
6      $V_{p_j} \leftarrow construct\_initial\_value(j, i)$ 
7   for each gate  $G_j$  in gate_list do
8      $propagete\_value(G_j)$ 
9   if  $V_o$  is not 0 then
10    return “不等价”
11 return “等价”

```

大的内存使用开销。因此，在我们的方法中，我们将长整数划分为  $K$  组较小的整数，并一次执行一组模拟，从而减少内存消耗。在  $K$  值的选择上，我们需要考虑到当前操作系统指令的位长和硬件（尤其是 CPU）的指令位宽，尽可能使得  $\frac{2^N}{K}$  的值等于位宽，以提高批处理的执行效率。具体地，若机器支持  $2^k$  bit 位宽的批处理，我们则需要设置  $K = N - k$ ，以最大化批处理的效率。拆分方法如图 6.5 所示。

改进 EPS 的伪码如算法 7 所示。在开始时，该算法将所有的门按拓扑顺序排序（第 1 行）。然后计算每组的长度 ( $l$ ) 和执行模拟所需的轮数 ( $R$ )（第 2-3 行）。 $PI$  的数量不应超过  $bits\_limit$ 。然后，对于每个组，算法会将短整数赋值于  $PI$  上（第 5-6 行），并根据电路的拓扑序列对赋值进行传播（第 7-8 行）。同时，SIMD 可以从指令层面并行的加速值传播的过程。在模拟过程中有可能找到一个反例（第 9-10 行）。在完成所有轮后，测试了所有分组下的  $PI$  可行赋值，即成功验证了 miter 电路（第 11 行）。通过这种截断方法，EPS 的内存开销被限制在一个合理的范围内。改进的 EPS 方法可以在 4 分钟内检查 32 个  $PI$  的电路，而 40 个  $PI$  的电路需要大约半天的时间。

#### 6.1.2.4 引擎选择启发式

在 Sweeping 的过程中，SAT 求解和 EPS 仿真在不同结构的电路上具有不同的性能表现。通过分析实例的内部设计（包括 6.1.2.1一节中提到的 ‘ec\_h1’，数据分析表明，对于 SAT 求解器难以证明的节点对，其 TFI-cone 的 miter 电路都有很大比例的异或链。

因此，对于要验证的 miter 电路  $M$ ，我们根据异或链密度设计了一种启发式算法来确定是使用 SAT 求解器还是使用 EPS。我们提出一个函数，表示为  $score_{XOR}$ ，用于测量 XOR 链密度，定义如下。

$$score_{XOR}(M) = \log_2(\sum_{b \in BS} 2^{|b|})/N \quad (6.2)$$

$score_{XOR}$  是按照以下直觉设计的：EPS 的求解时间随着 PI 的数量 N 呈指数增长，估计为  $2^N$ 。CDCL 在异或链占很大比例的不可满足实例上性能较差，因为异或链的不可满足证明在最坏情况下需要指数数量的子句<sup>[297,298]</sup>。驳斥具有  $b$  个 XOR 门的 XOR 块的运行时可以估计为  $2^b$ ，具有多个 XOR 链的电路的运行时可以估计为  $\sum_{b \in BS} 2^{|b|}$ 。虽然 SAT 问题是一个 NP 难问题，但由于几十年来的各种优化，实际的 CDCL 求解器比理论估计的效率更高。因此我们引入系数  $\rho$  来平衡其中的关系。通过预实验，本文将  $\rho$  设为 0.15。

使用  $score_{XOR}$  函数，选择启发式显示在算法 8 中。对于要检查的给定节点对  $(a_i, b_i)$ ，我们首先获取两个节点的 TFI-cone 的 miter 电路  $M$ （第 1 行）。然后，我们识别  $M$  中的异或门，并根据连接关系将异或门分组为多个异或块  $BS$ （第 2 行）。评分函数是通过访问  $BS$  中的每个 XOR 块来计算的（第 3-6 行）。最后，当异或链密度大于参数  $\rho$  时，启发式算法更倾向于选取 EPS 工具；否则，选择 SAT 求解器。

#### 6.1.3 相同结构检测

在 sweeping 的过程之前，算法通过逻辑仿真会选择出大量的内部潜在等价节点。因为执行的仿真轮数较多，其中大部分是真正的等价节点。数据通路电路的运算单元通常是典型的乘法器和加法器。在实际编码时，相同的单元一般是由同一套代码或者规则生成的，因此许多算术单元共享相同的逻辑函数和实现，即具有一定的“正则性” [288]。

**Algorithm 8:** SAT 与 EPS 工具的选择启发式算法

---

**输入:** 待测等价点对  $(a_i, b_i)$   
**输出:** 推理工具种类 (“SAT” 或者 “EPS” )

```

1  $M \leftarrow miter\_TFI\_cones(a_i, b_i)$ 
2  $BS \leftarrow find\_all\_XOR\_blocks(M)$ 
3  $score_{XOR} \leftarrow 0$ 
4 for  $b$  in  $BS$  do
5   |  $score_{XOR} \leftarrow score_{XOR} + 2^{|b|}$ 
6 end
7  $score_{XOR} \leftarrow \frac{\log_2(score_{XOR})}{|PI|}$ 
8 if  $score_{XOR} > \rho$  then
9   | return EPS
10 end
11 else
12   | return SAT
13 end
```

---

对于每个潜在的等价对，如果具有相同结构和拓扑实现（正则性）的前一对已被证明是等价的，则无需再次使用十分耗时的推理工具再次检查该组节点对的等价性。

原理图如图 6.6 所示，设节点对  $(x, x')$  是功能等价的，且是通过某推理工具的单次调用证明的。如果节点  $x, y$  是具有完全相同实现的两个单元，并且节点  $x', y'$  也是具有相同实现的两个单元，则节点对  $(y, y')$  是功能等价的节点。

对两个节点的 TFI-cone,  $x, y$  进行正则性检验的具体方法是贪心算法。最初，我们维护一个只有一个节点对  $(x, y)$  的队列。如果队列不为空，则从队列中获取节点对  $(a, b)$ ，并比较输出为  $a$  和  $b$  的与门的输入。设  $a$  的输入为  $a_1$  和  $a_2$ ,  $b$  的输入为  $b_1$  和  $b_2$ 。我们检查  $a_1$  和  $b_1$  的极性是否匹配，以及  $a_2$  和  $b_2$  的极性(门的输入变元的符号)是否匹配。如果它们都匹配，我们将两对  $(a_1, b_1), (a_2, b_2)$  推送到队列中。一旦其中一个比较失败，则整个结构匹配失败。此匹配过程迭代运行，直到队列变为空，并完成验证。

本小节中提到的方法称为相同结构检测 (ISD)，它将引擎调用的次数减少了一次，同时增加了两个轻量级的规则检查。

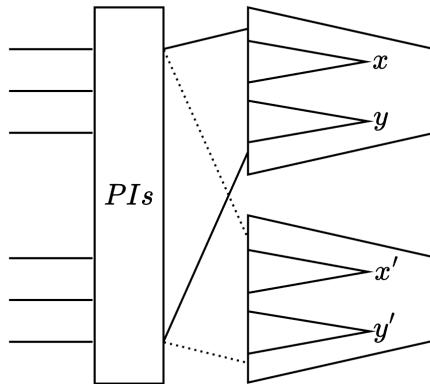


图 6.6 相同结构检测 (IDS) 技术的示意图

Figure 6.6 Schematic Diagram for Identical Structure Detection (IDS).

#### 6.1.4 实验评估

本节介绍了实验环境，展示了对比实验的结果，并对组件执行了有效性分析。

##### 6.1.4.1 实验设置

本节的实验环境与第 2 章保持一致。本文提出的 CEC 求解器 (hybridCEC) 采用 C++ 编写的，并用 GNU g++ (版本 9.4.0) 编译。相关代码可以从我们的 GitHub 仓库获取<sup>1</sup>。所有的对比求解器也都是用 C/C++ 编写的，并在与 hybridCEC 相同的环境中编译。

本文提到的推理工具如下：

- kissat-MAB<sup>[227]</sup>。它是目前性能最强悍的 SAT 求解器之一，也是 2021 年 SAT 比赛主赛道的冠军。
- MiniSat<sup>[44]</sup>。它是基于 CDCL 的 SAT 求解器的一个里程碑意义的求解器，是 ABC 默认的求解引擎，也是应用数量最多的一个求解器。
- KCBox<sup>[291]</sup>。它是一个开源的知识编译工具箱，集成了一些强大的 BDD 相关的工具。

我们使用了 50 个来自工业界的高位宽运算电路优化的样例，均为带有大量加法器和乘法器的数据通路电路。所有 50 个实例都是 AIG 格式的 miter 电路，均由两个功能等价的电路构成，即若转化成 SAT 问题均为不可满足的例子。实

<sup>1</sup><https://github.com/HybridCEC/Hybrid-CEC>

例可以分为 3 类：  $dpm$ 、 $ec$  和  $dp$ 。  $dp$  电路是具有乘加混合运算单元的典型数据通路电路。 $dpm$  是运算单元数量相对较少的电路，主要包含乘法器。这些  $dp$  和  $dpm$  电路是由华为公司生成的实际样例。 $ec$  电路可以看作是  $dp$  和  $dpm$  的混合电路，可以从 EDA 挑战赛，或者我们的 GitHub 仓库下载。此外，测试集中的所有实例都是从实际应用工具中生成的原始样例，没有执行任何简化或逻辑综合操作。

对于每一个电路，每一个证明器的求解时间上限被设置为 3600s，超时样例的求解时间用“TO”表示。`#Solved` 表示被一个 CEC 证明工具成功证明的实例数；`#Best` 是某证明器在所有证明器中用时最短的实例的数目。

#### 6.1.4.2 在数据通路样例集合上的表现

我们将 hybridCEC 与其它的三个主流的求解器和方法进行了对比，包括

- ABC &cec<sup>[285]</sup>. 该求解器是目前最具有代表性的一个基于 SAT-sweeping 方法，也是本文中主要的对比求解器。
- Pure SAT. 使用一个高效的 SAT 求解器, kissat-MAB<sup>[227]</sup>, 一次性的直接验证（不采用 sweeping 框架）最终的 miter 电路  $M_o$  对应的 CNF 公式。
- Pure BDD. 使用一个高效的 BDD 编译器, KCBox<sup>[291]</sup>, 直接验证  $M_o$  对应的公式。

从表6.2中的结果，我们可以了解到 hybridCEC 比它的对比求解器明显有更好的性能。在这 50 个例子上，hybridCEC 可以比 ABC &cec、pure SAT 和 pure BDD 分别多求解 20、19 和 22 个实例。与 ABC &cec 相比，hybridCEC 有 12% 的例子可以至少达到 3 个数量级的提速；且有 30% 的样例能达到 2 个数量级的提速。此外，单次调用 SAT 或者 BDD 的方法与 ABC 有相似的结果，通过对比，可以得到相似的结论。

#### 6.1.4.3 统计数据分析

为了更好的理解 hybridCEC 求解过程中的行为，我们收集了 hybridCEC 求解过程中的统计数据。在表 6.3 中，我们汇报了有可能等价的待测等价点对数目（“Pairs”列）、由 ISD 技术识别出的相同结构（“ISD”列）、分配由 SAT 求解器证明的点对数（“#SAT”列）、SAT 求解器花费的时间（“ $T_{SAT}$ ”列）、分配由 EPS 引擎证明的点对数（“#EPS”列）以及 EPS 求解器花费的时间（“ $T_{EPS}$ ”列）。

从“ISD”列，我们了解到，该组数据通路样例中，有很大比例的子电路的结构出现了重复现象，占比为 56.4%。通过使用 ISD 技术，很多耗时的不必要的引擎调用被直接跳过，因此证明器的证明时间可以被极大的削减。

在数据路径电路中，当存在高密度 XOR 链时，引擎选择启发式选择 EPS。EPS 调用的比例反映了电路的难度。从表中可以看出，EPS 在验证过程中发挥着重要作用。同时，我们可以了解到，数据通路电路的验证困难主要是由于整个验证过程总是卡在一些难以证明的子电路上。

#### 6.1.4.4 求解策略的评估

本节对 hybridCEC 中的每一个子策略进行了有效性的评估。在表 6.4 中，我们将 hybridCEC 与 4 个不同的变种版本进行了对比：

- $V_1$ : 在 Sweeping 的过程中只使用 SAT 求解器。
- $V_2$ : 在 Sweeping 的过程中只使用 EPS 求解器。
- $V_3$ : 禁用 ISD 技术。
- $V_4$ : 将 hybridCEC 中的 SAT 求解器替换为与 ABC 一致的 MiniSat<sup>[44]</sup>。

通过将 hybridCEC 与  $V_1$  和  $V_2$  对比，我们了解到 hybridCEC 的性能会受到选择策略启发式的影响。从 hybridCEC 与  $V_3$  的对比中，我们可以了解到 ISD 能有效的加速验证的时间，但是不会特别明显的影响求解器能力（即某样例的可解性）。从与 ABC 的对比中，为了公平起见，我们将 hybridCEC 中的 SAT 求解器 kissat-MAB<sup>[227]</sup> 替换为 MiniSat。从实验结果中，我们可以了解到，极高性能的 SAT 求解器的对于 hybridCEC 在证明数据通路实例等价性中的性能影响不大，这也从另一方面证明了我们框架和内部策略的有效性。

#### 6.1.4.5 对于引擎选择启发的进一步分析

本节，我们详细阐述了引擎选择启发式的参数选择细节。第 6.1.2.1 节的样例展示了 SAT 求解器和 EPS 工具在不同种类电路上的互补性，第 6.1.2.4 节基于该现象提出了一种基于异或链密度的启发式打分函数  $score_{XOR}$ 。

图 6.7 中展示了求解时间和异或链密度打分  $score_{XOR}$  之间的关系。对于每一个“ec\_h1”中的待测等价点对，我们都会计算其异或链密度打分  $score_{XOR}$ ，并统计了 SAT 求解器与 EPS 工具的求解时间。我们将其对应关系绘制在了散点图上，图中的横坐标表示密度打分，所有待测点对都按照这个密度进行了排序。通过图

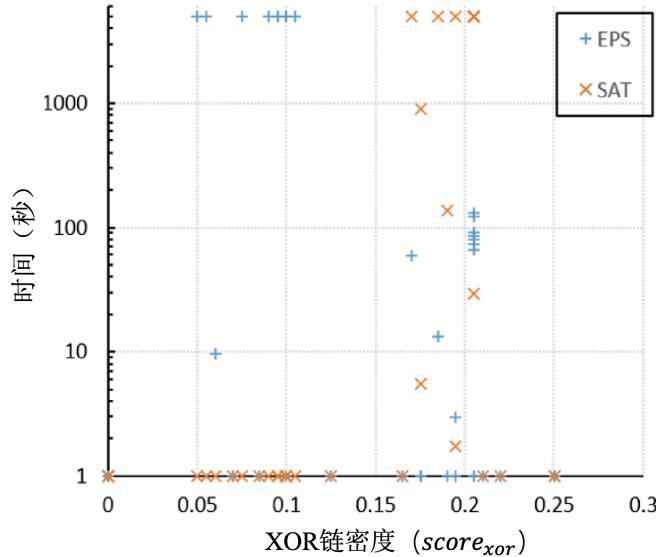


图 6.7 不同异或链密度打分的待测等价点对上 EPS 引擎与 SAT 求解器验证时间的对比

**Figure 6.7 The runtime comparison between EPS and SAT for each potential-equivalent internal node pair according to  $score_{xor}$ .**

中的数据，我们可以明显的观察到，SAT 求解器更适合用于求解  $score_{XOR} \leq 0.15$  的子电路等价性验证问题。相比之下，EPS 引擎更适合于验证  $score_{XOR} > 0.15$  的子电路等价性问题。此外，本文中其他的数据通路电路与“sec\_h1”样例的结构相似，因此我们采用了相同的启发式选择参数。

表 6.2 hybridCEC 与其它方法的对比

Table 6.2 Comparing hybridCEC with competitors.

实例名称	Gates	证明器名称			
		hybridCEC	ABC &ccec	Pure SAT	Pure BDD
dpm_1_1	386	<b>0.01</b>	0.18	0.14	0.46
dpm_2_1	867	<b>0.02</b>	1.46	0.86	1.16
dpm_3_1	696	<b>0.01</b>	5.44	3.07	11.24
dpm_3_2	975	<b>0.02</b>	13.15	5.77	15.32
dpm_4_1	877	<b>0.02</b>	24.77	19.08	88.98
dpm_4_2	1333	<b>0.04</b>	60.11	21.97	81.06
dpm_4_3	1628	<b>0.08</b>	4.88	2.82	8.67
dpm_5_1	703	<b>0.01</b>	6.08	6.02	17.33
dpm_5_2	1319	<b>0.34</b>	1576.8	834.81	TO
dpm_5_3	2068	<b>0.84</b>	2198.41	491.28	2207.7
dpm_6_1	963	64.02	116.79	<b>57.55</b>	252.5
dp1_1	681	82.45	<b>13.93</b>	70.67	215.29
dp2_1	460	1.14	2.55	<b>0.59</b>	1.19
dp3_1	2116	<b>0.05</b>	10.98	218.8	TO
dp3_2	2647	<b>0.09</b>	TO	538.21	TO
dp3_3	7118	<b>25.7</b>	TO	TO	TO
dp3_4	8574	<b>47.98</b>	TO	TO	TO
dp3_5	10182	<b>42.63</b>	TO	TO	TO
dp4_1	1646	<b>0.05</b>	2.52	16.93	1951.42
dp4_2	5332	<b>24.32</b>	TO	TO	TO
dp4_3	10448	<b>171.28</b>	TO	TO	TO
dp4_4	11256	<b>267.02</b>	TO	TO	TO
dp4_5	12360	<b>487.97</b>	TO	TO	TO
dp5_1	18	<0.01	0.02	<0.01	0.18
dp5_2	1646	<b>0.03</b>	2.56	12.35	459.65
dp5_3	9798	<b>424.6</b>	TO	TO	TO
dp5_4	11484	<b>541.12</b>	TO	TO	TO
dp5_5	13617	<b>937.57</b>	TO	TO	TO
dp6_1	4585	<b>2.41</b>	TO	TO	TO
dp6_2	5332	<b>5.85</b>	TO	TO	TO
dp6_3	6128	<b>26.72</b>	TO	TO	TO
dp6_4	8690	<b>297.69</b>	TO	TO	TO
dp6_5	15787	TO	TO	TO	TO
dp7_1	1238	<b>0.03</b>	0.36	2.41	50.3
dp8_1	2116	<b>0.14</b>	10.11	104.41	2532.35
dp9_1	6128	<b>26.78</b>	TO	TO	TO
dp10_1	14049	TO	TO	TO	TO
dp11_1	20091	TO	TO	TO	TO
dp12_1	24773	TO	TO	TO	TO
dp13_1	378	<0.01	0.02	0.02	0.3
dp14_1	7061	<b>445.1</b>	TO	TO	TO
ec_e1	280	<0.01	0.06	0.03	0.26
ec_e2	492	<b>0.01</b>	0.51	0.35	0.6
ec_m1	612	<0.01	0.04	0.1	0.53
ec_m2	1256	<b>0.02</b>	0.41	2.17	50.37
ec_m3	1664	<b>0.05</b>	2.55	12.54	1314.76
ec_h1	12499	<b>1464.17</b>	TO	TO	TO
ec_h2	13675	<b>3543.39</b>	TO	TO	TO
ec_h3	14152	TO	TO	TO	TO
ec_h4	15604	<b>2497.91</b>	TO	TO	TO
#Solved		<b>45</b>	25	26	23
#Best		<b>42</b>	1	3	0

表 6.3 hybridCEC 的过程中求解器数据

Table 6.3 Statistics behavior of hybridCEC checking process

实例名称	Gates	Pairs	ISD	#SAT	$T_{SAT}$	#EPS	$T_{EPS}$
dpm_1_1	386	3	0	2	<0.01	1	<0.01
dpm_2_1	867	6	1	4	<0.01	1	<0.01
dpm_3_1	696	5	0	4	<0.01	1	<0.01
dpm_3_2	975	6	1	4	<0.01	1	<0.01
dpm_4_1	877	6	1	4	<0.01	1	0.02
dpm_4_2	1333	7	1	5	<0.01	1	0.03
dpm_4_3	1628	8	3	4	<0.01	1	0.05
dpm_5_1	703	5	0	4	<0.01	1	0.02
dpm_5_2	1319	9	4	4	<0.01	1	0.34
dpm_5_3	2068	13	7	5	<0.01	1	0.80
dpm_6_1	963	1	0	1	64.23	0	<0.01
dp1_1	681	1	0	1	82.77	0	<0.01
dp2_1	460	1	0	1	1.13	0	<0.01
dp3_1	2116	40	22	12	0.01	6	<0.01
dp3_2	2647	45	25	13	0.02	7	0.02
dp3_3	7118	83	47	22	0.04	14	25.58
dp3_4	8574	101	59	25	0.06	17	47.81
dp3_5	10182	143	83	37	0.15	23	42.12
dp4_1	1646	35	19	11	0.01	5	<0.01
dp4_2	5332	65	37	17	0.02	11	24.16
dp4_3	10448	114	66	29	0.06	19	170.91
dp4_4	11256	127	75	31	0.21	21	266.83
dp4_5	12360	159	95	39	0.25	25	487.43
dp5_1	18	1	0	1	<0.01	0	<0.01
dp5_2	1646	35	19	11	0.01	5	<0.01
dp5_3	9798	98	57	24	0.16	17	424.23
dp5_4	11484	119	69	30	0.08	20	540.56
dp5_5	13617	171	100	44	43.60	27	893.61
dp6_1	4585	60	34	16	0.01	10	2.23
dp6_2	5332	65	37	17	0.02	11	5.73
dp6_3	6128	70	40	18	0.02	12	26.60
dp6_4	8690	85	48	22	0.04	15	297.50
dp7_1	1238	30	15	11	0.01	4	<0.01
dp8_1	2116	40	22	12	0.01	6	0.01
dp9_1	6128	70	40	18	0.02	12	26.64
dp13_1	378	15	7	6	<0.01	2	<0.01
dp14_1	7061	75	43	19	0.04	13	445.30
ec_e1	280	3	0	2	<0.01	1	<0.01
ec_e2	492	3	0	2	<0.01	1	<0.01
ec_m1	612	20	10	8	0.01	2	<0.01
ec_m2	1256	30	16	10	<0.01	4	<0.01
ec_m3	1664	35	18	11	0.02	6	<0.01
ec_h1	12499	113	65	28	0.06	20	1465.44
ec_h2	13675	129	76	31	0.07	22	3548.04
ec_h4	15604	163	98	38	0.14	27	2497.60

表 6.4 hybridCEC 关键技术的有效性分析

Table 6.4 Analysis of the key techniques used in hybridCEC.

实例名称	Gates	验证器名称				
		hybridCEC	V <sub>1</sub>	V <sub>2</sub>	V <sub>3</sub>	V <sub>4</sub>
dpm_1_1	386	<b>0.01</b>	0.12	0.02	0.03	<b>0.01</b>
dpm_2_1	867	<b>0.02</b>	0.89	0.17	0.07	<b>0.02</b>
dpm_3_1	696	<b>0.01</b>	2.82	0.08	0.04	<b>0.01</b>
dpm_3_2	975	0.02	5.55	0.13	0.07	<b>0.01</b>
dpm_4_1	877	<b>0.02</b>	23.42	0.2	0.14	0.03
dpm_4_2	1333	<b>0.04</b>	27.16	0.07	0.24	0.05
dpm_4_3	1628	<b>0.08</b>	2.9	0.12	0.28	0.09
dpm_5_1	703	<b>0.01</b>	6.43	0.06	0.11	0.03
dpm_5_2	1319	<b>0.34</b>	1334.05	5.04	3.85	0.96
dpm_5_3	2068	0.84	681.02	<b>0.31</b>	4.84	1.09
dpm_6_1	963	64.02	64.77	<b>0.79</b>	70.36	191.45
dp1_1	681	<b>82.45</b>	109.91	1469.39	89.57	307.97
dp2_1	460	1.14	1.15	10.27	<b>1.11</b>	2.51
dp3_1	2116	<b>0.05</b>	1.98	TO	0.22	0.13
dp3_2	2647	<b>0.09</b>	7.21	TO	0.35	0.2
dp3_3	7118	<b>25.7</b>	TO	TO	120.24	64.11
dp3_4	8574	<b>47.98</b>	TO	TO	233.99	98.52
dp3_5	10182	<b>42.63</b>	TO	TO	471.04	50.28
dp4_1	1646	<b>0.05</b>	0.53	TO	0.12	0.08
dp4_2	5332	24.32	TO	TO	103.58	<b>19.81</b>
dp4_3	10448	171.28	TO	TO	1271.06	<b>85.0</b>
dp4_4	11256	267.02	TO	TO	2500.96	<b>113.66</b>
dp4_5	12360	<b>487.97</b>	TO	TO	3387.28	879.55
dp5_1	18	<0.01	<0.01	<0.01	<0.01	<0.01
dp5_2	1646	<b>0.03</b>	0.74	TO	0.15	0.11
dp5_3	9798	<b>424.6</b>	TO	TO	2813.27	1104.36
dp5_4	11484	<b>541.12</b>	TO	TO	TO	1535.88
dp5_5	13617	<b>937.57</b>	TO	TO	TO	2722.96
dp6_1	4585	<b>2.41</b>	1220.27	TO	27.9	3.14
dp6_2	5332	<b>5.85</b>	TO	TO	169.79	21.32
dp6_3	6128	<b>26.72</b>	TO	TO	588.2	232.32
dp6_4	8690	<b>297.69</b>	TO	TO	TO	1765.74
dp6_5	15787	TO	TO	TO	TO	TO
dp7_1	1238	<b>0.03</b>	0.2	TO	0.09	0.08
dp8_1	2116	<b>0.14</b>	2.0	TO	0.47	0.16
dp9_1	6128	26.78	TO	TO	889.63	<b>24.9</b>
dp10_1	14049	TO	TO	TO	TO	TO
dp11_1	20091	TO	TO	TO	TO	TO
dp12_1	24773	TO	TO	TO	TO	TO
dp13_1	378	<0.01	0.02	0.42	0.01	0.02
dp14_1	7061	445.1	TO	TO	3330.51	<b>157.6</b>
ec_e1	280	<0.01	0.04	0.02	0.03	0.01
ec_e2	492	<b>0.01</b>	0.28	0.06	0.03	<b>0.01</b>
ec_m1	612	<0.01	0.02	29.89	0.05	0.01
ec_m2	1256	<b>0.02</b>	0.1	TO	0.1	0.05
ec_m3	1664	<b>0.05</b>	0.44	TO	0.19	<b>0.05</b>
ec_h1	12499	1464.17	TO	TO	TO	<b>1061.78</b>
ec_h2	13675	<b>3543.39</b>	TO	TO	TO	TO
ec_h3	14152	TO	TO	TO	TO	TO
ec_h4	15604	<b>2497.91</b>	TO	TO	TO	3161.23
#Solved		<b>45</b>	27	18	39	44
#Best		<b>35</b>	1	3	2	13

## 6.2 混合 SAT 求解器在流密码分析中的应用

信息安全逐渐成为目前互联网技术、大数据技术和军事技术发展的热点话题，密码技术作为信息安全的核心问题，也日益成为其中的重点研究方向。在密码学领域，如何设计一个安全、高效、轻量的密码算法一直以来都是该领域的热点话题。密码分析（cryptanalysis）问题指的是在不知道通常揭秘所需要的密钥信息时，对信息进行解密的一门学问。密码分析对于密码使用者来说有助于发现存在的漏洞，进而帮助提高自身的安全性；而对于攻击者来说有重要的军事和战略意义。

常见的密码问题一般可以分为对称密码和非对称密码，而流密码（序列密码）算法属于对称密码中十分关键的一类问题，一般具有高效性、结构简单等特点，常被应用于保护数据的完整性和私密性，常见的流密码问题有类如 RC4<sup>[299]</sup>、A5/1<sup>[300]</sup>、Chacha<sup>[301]</sup>、Trivium<sup>[302]</sup>、SNOW<sup>[303]</sup>、Ascon<sup>[304]</sup>、Grain<sup>[305]</sup>、ZUC<sup>[306]</sup>等。

流密码的组成结构简单，一般内部包含一个固定大小的内部状态、作用在该内部状态上的更新函数、以及由内部状态得到的输出函数组成。每一个时钟周期，流密码算法会由输出函数输出  $n$  比特的密钥流，并通过更新函数对内部状态进行更新。加密时，密钥流与明文的异或会生成密文，同理，解密时可以用密文与密钥流执行异或操作恢复明文。

因为流密码问题结构的灵活性，从技术的角度上，存在多种针对流密码分析的方法。理想情况下，一个好的密码算法应该从理论上可以抵抗所有已知的分析方法，即想要获得所需信息的代价不低于穷举密钥的安全强度。但是实际上，密码结构在设计上难免会存在一些漏洞或缺陷，相关人员可以利用发现的特性，设计相关的分析方法，以降低分析难度。除此之外，对密码算法本身的结构和理论特性的分析也有助于帮助我们设计相关的密码分析方法。

根据最近文献的观点<sup>[307]</sup>，目前主流的密码分析方法可以分为四类：基于相关性的方法、基于差分性质的方法、基于代数方程组的分析方法和基于时空折中的方法。其中基于代数方程组的分析方法与本文的方法相关性最高。

基于相关性的方法主要利用函数输入输出之间的统计学差异进行密码分析，并可以进一步分为对高相关性进行查找和利用两类技术。在查找方面，又可以进一步分为以将密钥流序列与随机序列区分开的区分分析方法<sup>[308]</sup> 和以恢复内部

状态或密钥为主要目标的相关分析<sup>[309]</sup>两类方法。基于相关性的方法中最具有代表性的方法为快速相关分析方法<sup>[310,311]</sup>。

基于差分性质的方法主要利用了函数的差分性质对密码问题进行分析，代表性方法有差分分析<sup>[312]</sup>、滑动攻击<sup>[313]</sup>、碰撞攻击<sup>[314]</sup>、近似碰撞分析<sup>[315]</sup>和立方分析<sup>[316-318]</sup>等技术。

基于代数方程组的分析方法主要借助求解方程组来恢复内部的状态，主要包括先猜后定法<sup>[319]</sup>和代数分析两类方法。先猜后定会先对所求目标的（如状态）进行一定的猜测，根据猜测值降低密码分析的难度，然后再根据最后分析的结果判断猜测值的正确性，该类问题的核心研究难点在于如何选取一个较优的猜测集<sup>[320]</sup>。单纯的使用该分析方法往往不能得到一个理想的分析结果，但是该方法在与遗传算法<sup>[321]</sup>、混合整数规划<sup>[322]</sup>等自动化搜索技术相结合时能产生不错的效果。代数分析法则将密码问题建模到代数问题或者对问题进行形式化建模来分析的方法，常见于密码分析的工具有SAT求解器<sup>[323]</sup>、SMT求解器<sup>[324]</sup>、Groebnor基<sup>[325]</sup>和MILP求解器<sup>[326]</sup>等工具。代数免疫度<sup>[327]</sup>相关的分析也属于该范畴。

基于时空折中的方法一般会将密码问题当做一个黑盒，充分利用目前已有的信息，来建立合适的折中曲线<sup>[328]</sup>。

“强网杯”密码数学专项赛中给出了一个“环上前馈序列密码”问题，它是一个人为构造的流密码问题。它的设计结构完整、简洁，包含了主流流密码算法的所有特征。为了方便对参赛队伍的分析方法更好的评估，该算法的密码输出函数也进行了精心的设计，且寄存器状态的不同分位间有一定的独立性。分位越高、相关性分析难度越大、密码分析的难度越高。

**主要贡献** 本节以“环上前馈序列密码问题”为研究案例，介绍了一套密码分析的方法，对研其它流密码分析有一定指导意义。

具体地，我们首先通过分析算法在状态更新阶段的特性，我们发现寄存器内部状态具有相对独立性。因此，我们以寄存器初始状态的分位比特位顺序，根据输出序列每拍的第0、1、2、3分位，依次恢复初始状态每个半字节的第0、1、2、3分位，从而恢复出完整的初始密钥。对于前两分位，我们发现了分位函数上的特殊性质，并利用该技术成功恢复了初态；对于后两分位，分位函数上没有高概

率线性逼近和二次逼近表达式，对此，我们提出了一种“轮增量编码”的分析技术，将问题编码到 SAT 问题，然后采用了混合 SAT 求解器作为密码分析的主要推理工具，成功分析出了后两分位的初态。通过仿真测试，我们验证了求得的初态密钥的正确性。

### 6.2.1 环上前馈序列密码问题

在实际应用中，流密码由于其加解密速度快、吞吐量高、易于硬件实现等优点在资源受限的场景中得到广泛的应用。本案例中涉及的“环上前馈序列密码”本质上是一种基于线性反馈移位寄存器（LFSR）的流密码问题，由于其在数学上具有更加清晰的结构，统计特性、周期特性以及相关性特性也具有更好的理论分析。

模 16 的整数环  $\mathbb{Z}/16$  上前馈序列密码算法如图6.8所示。线性移位寄存器的内部状态被划分成 128 个半字节（4 比特），记作  $(a_{127}, a_{126}, \dots, a_1, a_0)$ 。移位寄存器初始化后，每次先调用非线性函数  $f$  计算并生成 4 比特的输出  $z$ ，然后调用状态更新函数更新寄存器状态。移位寄存器一共运行 8000 拍，且本案例给出了 8000 拍的输出  $z^{(0)}, z^{(1)}, \dots, z^{(7999)}$  和非线性函数的 S 盒。

**寄存器初始化.** 密钥（初态） $K_0, \dots, K_{53}$  共包含 54 个字节，第  $i$  个字节  $K_i$  中第  $j$  个分位的取值表示为  $K_i[j]$ ，其取值具体初始化规则为：

- $a_{2i} \leftarrow K_i[3, 2, 1, 0], a_{2i+1} \leftarrow K_i[7, 6, 5, 4], 0 \leq i \leq 53$
- $a_{2i+108} \leftarrow K_i[5, 4, 3, 2], a_{2i+109} \leftarrow K_i[1, 0, 7, 6], 0 \leq i \leq 9$

**非线性函数.** 若  $\mathbb{F}_2^n$  表示有限域  $\mathbb{F}_2$  上的  $n$  维向量空间，则非线性函数  $f$  是一个定义在  $\mathbb{F}_2^{36} \rightarrow \mathbb{F}_2^4$  上的非线性函数，其输入为当前时刻寄存器中抽取的 9 个半字节  $a_{12}, a_{24}, a_{41}, a_{64}, a_{77}, a_{96}, a_{112}, a_{121}, a_{126}$ ，依次记为  $x_0, \dots, x_8$ 。输出半字节记为  $z$ ，第  $i$  个分位的值为  $z[i]$ <sup>2</sup>。

非线性函数  $f(x) = (f_3, f_2, f_1, f_0)$  的分位函数  $f_i$  定义在  $\mathbb{F}_2^9 \rightarrow \mathbb{F}_2$  上，其输入为  $x_0, \dots, x_8$  的第  $i$  分位<sup>3</sup>，即

$$f_i = f_i(x_8[i], \dots, x_0[i]), 0 \leq i \leq 3$$

<sup>2</sup>前 150 轮的输出为：0x9ee0fd5a73d234ae27bffdaadb3aa0ff39da483df2978ba30d6335cba7aee77453f8dd50bd571782ea5fe628f69930c632e659aeaf9bbe99565e911430ae6f4e052d83a8e344325c56540f。

<sup>3</sup> $f_0, f_1, f_2, f_3$  的真值表（S 盒）的 16 进制表示分别为：0xf996946d9669499696694996698e86699669696699636eb699692699669499406696b926996b6696996b66996717996699696699669c91496696d966996b66b, 0x8bf02fb19b7646d91de46ed98f640ef92eb19d7406d99f6607f89f7427d089e4641db906f20d994ef29bb007e00f

**状态更新函数.** 下一时刻的状态为  $(a_{128}, a_{127}, \dots, a_2, a_1)$ , 其中  $a_{128} = (a_0 + a_{25} + a_{35} + a_{48} + a_{54} + a_{120}) \bmod 16$ 。

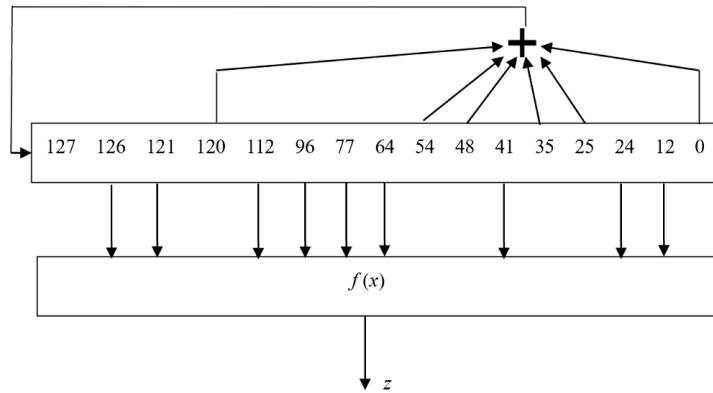


图 6.8 一个精心设计的流密码算法

Figure 6.8 A carefully designed stream cipher algorithm

### 6.2.2 基本求解思路

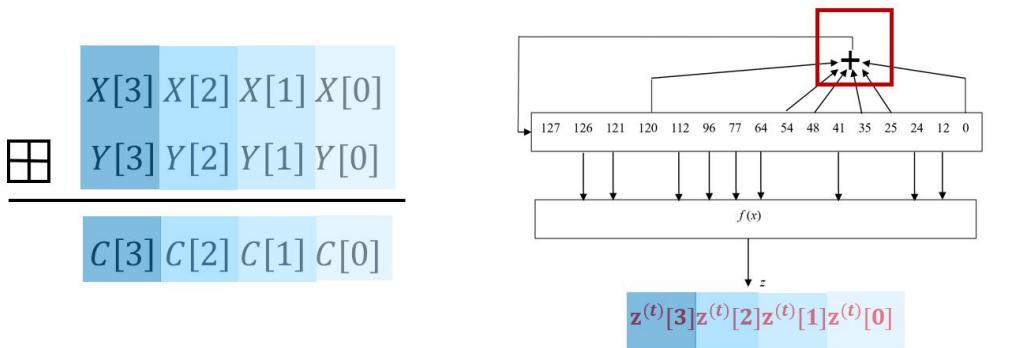


图 6.9  $f_0, f_1, f_2, f_3$  的相对独立性

Figure 6.9  $f_0, f_1, f_2, f_3$  are relatively independent

初始状态下，54字节密钥变量经过初始化后载入寄存器，在非线性函数  $f$  作用下生成4比特的输出  $z$ 。其中， $z$  的第  $i$  分位比特仅与当前寄存器每个半字节的第  $i$  个分位比特有关。然后，在状态更新函数的作用下，寄存器进行更新。我们的攻击目标是根据寄存器的 8000 拍输出值  $z^{(0)}, z^{(1)}, \dots, z^{(7999)}$ ，恢复出初始 54 字节密钥变量，即寄存器的初始状态。我们将寄存器在时刻  $t$  的状态记

9846f946601fd10e608fd806649df047f60b, 0x334fe641569d6b015c8ec6cfbc0a5e31334fe641569d6b015c8ec6cfbc0a5e31334fe641569d6b015c8ec6cfbc0a5e31, 0x9e9b0c1080ee7648a0d9ad875635e7f8a27c8ca5bd7459102b87d2ec66c3e4e81c4c271f6fcab1c6592c59899b298310e107ff945dd752a69b9e5f4f486ae78c。

为  $(s_{127}^{(t)}, s_{126}^{(t)}, \dots, s_0^{(t)})$ 。通过观察寄存器的更新函数，我们发现寄存器的状态具有相对独立性。

**观察 1（寄存器状态的相对独立性）**对于任意时刻  $t$ ，寄存器第  $j$  个半字节的第  $i$  个分位的比特值  $s_j^{(t)}[i]$  都可以表示为只与  $s_{127}^{(0)}[i \sim 0], s_{126}^{(0)}[i \sim 0], \dots, s_0^{(0)}[i \sim 0]$  有关的表达式，其中  $0 \leq i \leq 3, 0 \leq j \leq 127$ 。且进一步，当  $i = 0$  时，寄存器每个半字节的第 0 分位比特可以表示为  $s_{127}^{(0)}[0], s_{126}^{(0)}[0], \dots, s_0^{(0)}[0]$  的线性表达式。

如图 6.9 左侧所示，表示两个半字节上模 16 的加法运算，每个第  $i$  分位上的加法运算的和只会由其涉及的加数中分位小于  $i$  的数值决定。图右侧表示状态更新函数之间的关系。其中 4 个分位的输出非线性函数的输出是相互独立的，且因为新状态的计算只涉及到移位运算与模 16 的加法，所以可知上述观察 1 中的结论。

通过观察 1，我们可以进一步发现，每拍（时刻）的输出序列半字节的第  $i$  个分位比特  $z^{(0)}[i], z^{(1)}[i], \dots, z^{(7999)}[i]$ ，实际上都只与寄存器初始状态中每个半字节的第  $0, \dots, i$  个分位比特有关。而当我们确定了寄存器初态每个半字节的第  $0, \dots, i - 1$  个分位比特，输出序列每拍的第  $i$  个分位比特仅与寄存器初态每个半字节的第  $i$  个分位比特有关。

故我们的方法为，首先根据 8000 拍输出序列每半字节的第 0 个分位比特  $z^{(0)}[0], z^{(1)}[0], \dots, z^{(7999)}[0]$ ，恢复寄存器初始状态每个半字节的第 0 个分位比特  $s_{127}^{(0)}[0], s_{126}^{(0)}[0], \dots, s_0^{(0)}[0]$ ，从而恢复出初态密钥的对应 128 个密钥比特。然后，我们在 0 分位已经取得的 128 比特密钥的基础上，根据输出序列每半字节的第 1 个分位比特  $z^{(0)}[1], z^{(1)}[1], \dots, z^{(7999)}[1]$ ，恢复寄存器初始状态每个半字节的第 1 分位比特，即  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_0^{(0)}[1]$ ，此阶段共计恢复 128 个密钥比特。

由于初态密钥赋值前后 20 比特的值有重复性，因此，第 2、3 分位的密钥中已各自有 40 比特已知，即 2、3 分位的密钥各自有 88 个比特。最后，依次根据输出序列每半字节的第 2 分位比特，恢复寄存器初始状态每个半字节的第 2 个分位比特，此阶段共计恢复  $88+88=176$  个密钥比特。第 3 分位的 88 比特密钥恢复方法也类似，从而我们可以得到完整的初始密钥字节。

### 6.2.2.1 第 0 分位轮求解

我们对非线性函数  $f_0$  进行了线性相关分析，发现  $f_0$  以 93.75% 的概率满足下述线性表达式，违背的概率较小。

$$x_0 \oplus x_1 \oplus x_2 \oplus x_3 \oplus x_4 \oplus x_5 \oplus x_6 \oplus x_7 \oplus x_8 \oplus y = 0 \quad (6.3)$$

根据观察 1 可知， $f_0$  在任意时刻的输入均可以看成  $s_{127}^{(0)}[0], s_{126}^{(0)}[0], \dots, s_0^{(0)}[0]$  的线性表达式。由于  $z^{(0)}[0], z^{(1)}[0], \dots, z^{(7999)}[0]$  已知，我们可以根据 6.3 构建出 8000 条线性表达式，其中，每一个方程均以 93.75% 的概率成立。

如果能从这 8000 个方程组中找到一组正确的方程组，其中每一个方程均成立，且秩为 128，则我们可以采用高斯消元法求解器出  $s_{127}^{(0)}[0], s_{126}^{(0)}[0], \dots, s_0^{(0)}[0]$  的赋值。事实上，从 8000 个线性方程中选择 128 个成立的方程，可能的组合方法远大于 100 种。

$$C = \binom{8000}{128} \times 0.9375^{128} \gg 100 \quad (6.4)$$

而  $\mathbb{F}_2$  上随机  $n$  维矩阵满秩的概率为

$$G_n = \frac{(2^n - 1)(2^n - 2) \cdots (2^n - 2^{n-1})}{2^{n^2}} = (1 - \frac{1}{2^n})(1 - \frac{1}{2^{n-1}}) \cdots (1 - \frac{1}{2}) > 0.288 \quad (6.5)$$

综上，必定存在一组秩为 128 的正确方程组。

在实际求取中，我们随机选取  $\frac{1}{p^{128}} \times G_n \approx 13400$  组 128 维方程组，平均可以得到一个秩为 128 的正确方程组。通过求解该方程组，可得到  $s_{127}^{(0)}[0], s_{126}^{(0)}[0], \dots, s_0^{(0)}[0]$  的具体取值。求解过程仅用时几秒钟。

### 6.2.2.2 第 1 分位轮求解

此时，我们已知  $s_{127}^{(0)}[0], s_{126}^{(0)}[0], \dots, s_0^{(0)}[0]$  的取值。根据观察 1，输出序列每拍的第 1 个分位比特  $z^{(0)}[1], z^{(1)}[1], \dots, z^{(7999)}[1]$  仅与  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_0^{(0)}[1]$  有关。 $z^{(0)}[1], z^{(1)}[1], \dots, z^{(7999)}[1]$  是非线性函数  $f$  的分位函数  $f_1(x_8, \dots, x_1, x_0)$  的输出。仿照第 0 分位的求解思路，我们同样想构造  $f_1$  的线性逼近表达式，从而构造出一组秩为 128 的正确方程组，恢复 128 比特初始状态。但  $f_1$  并不存在高概率线性逼近式，因此第 0 分位求解的策略并不适用。但在搜索高概率线性逼近式的过程中，我们发现，表达式  $x_2 \oplus x_3 \oplus x_8 \oplus y = 1$  和表达式  $x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_7 \oplus x_8 \oplus y = 0$  同时成立的概率为 0。自然的，我们构造概率为 1 的二次逼近表达式如下：

$$(x_2 \oplus x_3 \oplus x_8 \oplus y) \cdot (x_0 \oplus x_1 \oplus x_3 \oplus x_4 \oplus x_7 \oplus x_8 \oplus y) = 0 \quad (6.6)$$

根据 8000 拍输出序列，总共可以构造 8000 个关于  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_0^{(0)}[1]$  的二次方程。

我们希望可以通过这 8000 个二次方程，构造出一组秩为 128 的线性方程组，通过高斯消元法求解出  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_0^{(0)}[1]$  的值。但， $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_0^{(0)}[1]$  一共可以形成  $\binom{128}{2} = 8128$  个二次项，且经过实际测试这 8000 个二次方程的任意线性组合都无法产生线性方程。同时，由于二次项的数目超过 8000 个，也无法采用线性化策略进行求解。为了降低二次项数目，我们采取先猜后定（Guess-and-determine）策略，通过预先猜测部分变量值，将二次项转化为线性项。在变量  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_0^{(0)}[1]$  中，我们任意选取  $d$  个变量进行猜测，此时变量数目减少为  $128 - d$  个，总共可以形成  $\binom{128-d}{2}$  个二次项。则这 8000 个二次方程至少可以生成  $8000 - \binom{128-d}{2}$  个线性方程。对任意一组猜测，根据，那么我们平均能够得到能够得到

$$G_{128-d} \times \binom{8000 - \binom{128-d}{2}}{128-d} \quad (6.7)$$

组秩为  $128-d$  的方程组，从而求解出  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_0^{(0)}[1]$ 。

由

$$128 - d < 8000 - \binom{128-d}{2} \quad (6.8)$$

和

$$G_{128-d} \times \binom{8000 - \binom{128-d}{2}}{128-d} \geq 1 \quad (6.9)$$

我们可知  $d \geq 3$ 。

在实际求取中，我们选取  $s_2^{(0)}[1], s_1^{(0)}[1], s_0^{(0)}[1]$  进行猜测。对于每个猜测，此时变量个数减少到  $128-3=125$  个，一共可以形成  $\binom{125}{2} = 7750$  个二次项。这 8000 个方程至少可以构造出  $8000-7750=250$  个关于  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_3^{(0)}[1]$  的 125 元线性方程，且平均至少有一组满秩方程组。通过该求解方程组，即可得到  $s_{127}^{(0)}[1], s_{126}^{(0)}[1], \dots, s_3^{(0)}[1]$  的所有可能取值。错误的取值可以通过重新加密排除。该求取过程耗时数分钟。

### 6.2.3 轮增量分析与 SAT 求解

我们尝试对 2、3 分位上采用与 0、1 分位上类似的方法进行求取，但是发现在  $f_2$  和  $f_3$  上没有明显的线性性质和差分性质，代数免疫度也很高，因此我们采用了 SAT 求解的方法来恢复该两轮的密钥。

### 6.2.3.1 SAT 编码

环上前馈序列密码中只包含模加和移位操作，进而可以拆分为与、或、异或与等价操作，对问题中出现的各种结构，我们采用了典型的 Tseitin 编码<sup>[60]</sup>对密码结构进行建模：

- 等价操作  $eql(a, b)$  可以编码为  $(\neg a \vee b); (a \vee \neg b)$  两条 CNF 子句。
- 与操作  $p = and(a_1, a_2, \dots, a_n)$  可以编码为  $n + 1$  条子句： $(\neg p \vee a_1); \dots; (\neg p \vee a_n); (\neg a_1, \dots, \neg a_n, p)$ 。
- 或操作  $p = or(a_1, a_2, \dots, a_n)$  可以编码为  $n + 1$  条子句： $(p \vee \neg a_1); \dots; (p \vee \neg a_n); (a_1, \dots, a_n, \neg p)$ 。
- 异或操作  $p = xor(a_1, a_2, \dots, a_n)$  可以通过引入  $n - 2$  个中间变元  $m_1, \dots, m_{n-2}$  的形式，将其转化为  $n - 1$  个长度为 3 的异或短表达式，进一步可以转化为  $4(n - 1)$  条子句： $m_1 = xor(a_1, a_2); m_2 = xor(m_1, a_2); \dots; m_{n-2} = xor(m_{n-3}, a_{n-2}); p = xor(m_{n-2}, a_n)$ 。其中每一个长度为 3 的 xor 表达式  $a = xor(b, c)$  均可以表示为 4 条 CNF 子句  $(\neg a \vee \neg b \vee \neg c); (\neg a \vee b \vee c); (a \vee \neg b \vee c); (a \vee b \vee \neg c)$ 。

由于我们无法直接用 Tseitin 编码直接编码非线性输出函数，因此我们首先需要将真值表转化为标准的代数表达式形式，然后才可以进一步套用 Tseitin 编码来生成 CNF。首先，对于非线性函数的 S 盒，我们采用了 LogicFriday 软件<sup>4</sup>将其转化为合取范式（CNF）表达式（外层析取，内层合取）和析取范式（DNF）表达式（外层合取，内层析取）；采用 sageMath 软件<sup>5</sup>通过莫比乌斯反演计算了其代数正规形（ANF）表达式（外层异或，内层与）。对于  $f_2$ ，我们可以将其转化为分别包含 25、25 和 74 项的 CNF、DNF 和 ANF；对于  $f_3$ ，我们可以将其转化为分别包含 92、90 和 261 项的 CNF、DNF 和 ANF；现在，我们拥有了一组关于  $f_2$ （或  $f_3$ ）的表达式，分别用  $f_2^{CNF}, f_2^{DNF}$  和  $f_2^{ANF}$ （或  $f_3^{CNF}, f_3^{DNF}$  和  $f_3^{ANF}$ ）表示标准化的输出函数。

因为第 2、3 分位初态的求取过程一致，我们以第 2 分位为例展开讨论。函数的每一拍的输出是给定的，下面，我们需要建模输出函数的输入变量  $(x_0, x_2, \dots, x_8)$  与初态变元  $s_{127}^{(0)}[2], s_{126}^{(0)}[2], \dots, s_3^{(0)}[2]$  之间的关系。

为了得到第 2 分位所有时刻的内部状态关于第 0 时刻初态的线性表示，我

<sup>4</sup>[https://download.cnet.com/Logic-Friday/3000-20415\\_4-75848245.html](https://download.cnet.com/Logic-Friday/3000-20415_4-75848245.html)

<sup>5</sup><https://www.sagemath.org/>

**Algorithm 9:** 第 2 分位所有时刻内部状态表达式的求取

**输入:** 第 0、1 分位所有时刻内部状态的赋值  
**输出:** 第 2 分位所有时刻内部状态关于第 0 时刻的线性表达式

```

1 for  $t$  in {1, 2, ..., 7999} do
2   for  $i$  in {0, 1, ..., 126} do
3      $s_i^{(t)}[2] \leftarrow s_i^{(t-1)}[2];$ 
4      $carry \leftarrow 0;$ 
5     for  $i$  in {0, 25, 35, 48, 54, 120} do
6        $carry \leftarrow carry + s_i^{(t-1)}[0];$ 
7      $carry \leftarrow carry >> 1;$ 
8     for  $i$  in {0, 25, 35, 48, 54, 120} do
9        $carry \leftarrow carry + s_i^{(t-1)}[1];$ 
10     $s_{127}^{(t)}[2] \leftarrow (carry \bmod 2 == 1 ? "1" : "0");$ 
11    for  $i$  in {0, 25, 35, 48, 54, 120} do
12      for  $s_0$  in  $s_i^{(t)}[2]$  do
13        if  $s_0$  in  $s_{127}^{(t)}[2]$  then
14          Remove  $s_0$  from  $s_{127}^{(t)}[2];$ 
15        else
16          Add  $s_0$  to  $s_{127}^{(t)}[2];$ 

```

们采用了算法9的方法进行计算。对于第 2 分位每一个时刻  $t$  下第  $i$  个内部状态  $s_i^{(t)}[2]$ ，我们均维护了一个关于 0 时刻内部状态的初态的集合，集合中所有元素的模 2 加法（即异或）组成了一个线性表达式。集合中“1”表示是否有常数位 1，“0”可以视为一个补位。

经过上述的计算之后，我们得到了所有时刻下第 2 分位状态的表达式，因此可以用于建立等式，并编码相关的 CNF 公式。如图6.10所示，对于每一个时刻  $t$  和其输出  $z^{(t)}$  都可以对应于一条命题逻辑表达式，并可以按照内外层共拆成 10 条小表达式，该表达式的外层可以看作是在一组在输入为  $x_8^{(t)}, x_7^{(t)}, \dots, x_0^{(t)}$ ，输出为  $z^{(t)}$  时  $f_2$  的表达式，该表达式可以有 CNF、DNF、ANF 三种表示形式。另外， $f_2$  的表达式中的 9 个参数，也可以对应的表示为 9 条约束，均可以表示为第 2 分位初态的线性表达式。这十条表达式，可以分别通过 Tseitin 编码的方式编码到 CNF 的数条子句。

如果直接将以上方法直接应用到 CNF 子句的生成中，会生成大量的冗余变

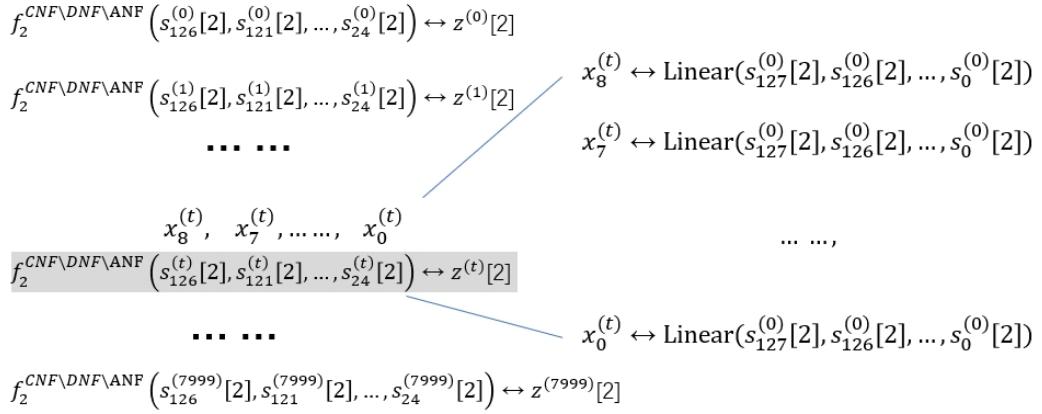


图 6.10 轮增量编码示意图

Figure 6.10 Round incremental encoding

量和子句。每一轮的输出  $z$  是一个确定的常量，且由于 0、1 分位已经求取成功，也可以视作一个常量，因此在计算第二分位所有时刻的线性表达式时会引入大量的 0、1 常量，针对这种情形，我们采用了布尔约束传播（BCP）的技术来处理其中的单元子句；针对编码中存在大量由于移位操作和中间变量引入的等价语句，采用了等价文字替换技术（ELS）技术。通过引入两种预处理技术，平均可以删除 36.4% 的冗余子句。

### 6.2.3.2 轮增量编码技术

通过前面的方法，我们已经可以建立 8000 组 SAT 公式约束，每一组 SAT 公式约束均代表一组非线性等式，直接将全部的非线性等式组成方程组，编码到 CNF 公式交给 SAT 求解器求解，求解难度太大，且有很多冗余的约束，因此需要对非线性方程组中的等式做筛选。

为了求取第 2、3 分位的初态，我们采取了一种轮增量编码的技术，其思想类似于模型检测领域的 Bounded Model Checking<sup>[329]</sup> 算法的思想。图6.11给出了轮增量编码求解的示意图。

**实现** 我们首先根据第 0 时刻到  $k$  时刻对应的线性方程组，构建 CNF 表达式，并交给 SAT 求解器求解。如果 SAT 求解器在规定的时间内可以求得解到一组解，则会从中提取出来初态变元的赋值，并模拟是否符合所有 8000 轮次的输出。如果符合，则找到一组解；否则，则增加  $k$  的值，按照相同的方法继续建模

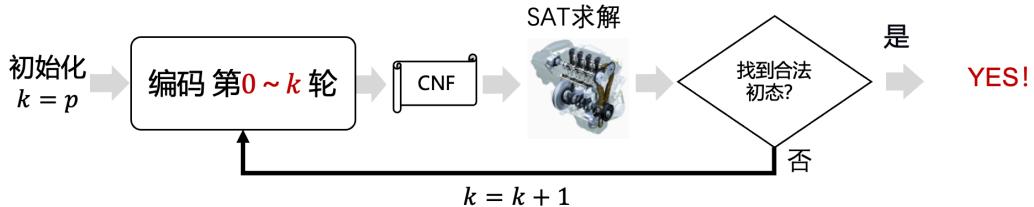


图 6.11 轮增量编码 SAT 求解技术

Figure 6.11 Round incremental encoding SAT solving technique

求解，直到找到一组正确的初态赋值为止。

实际上，我们可以随机抽取任意轮次的函数，组成方程组并建模到 SAT 问题来求解验证。我们发现，时刻越小，轮次越低，输出函数的输入变量对应的表达式中涉及的初态变量的数量越少。因此，我们更倾向于从低轮次到高轮次添加新的非线性约束表达式。

此外，在实际求解时，可以有如下的优化策略：1) 根据  $k$  相对较小时生成的 CNF 调参以提高当  $k$  较大时的求解能力。2) 根据第 3 章的建议，可满足样例求解时间有很大的不确定性，因此可以同时生成多个不同  $k$  取值的 CNF，或者将一个 CNF 的变元子句进行打乱重排，并行求解，若其中有一个 CNF 的求解速度较快且被证明不是假解，则可完成恢复工作。该方法可大幅提升证明效率。

#### 6.2.4 实验评估

本节的实验环境与第 2 章保持一致。

本文的所有默认 SAT 求解器在 kissat<sup>[50]</sup> 求解器基础上采用了我们的混合策略改进的 SAT 求解器，kissat\_inc<sup>6</sup>，进行求解。该求解器在求解该类例子上有不错的性能，相比之下，经典的 SAT 求解器 MiniSat<sup>[44]</sup>、密码学常用的求解器 CryptoMiniSat<sup>[270]</sup> 和移除掉局部搜索算法的 kissat 单一求解算法均无法在规定时间内找到合法的初态。

表6.5和表6.6中分别给出了求解第 2、3 分位时，求解由不同  $k$  值编码来的 CNF 的变量数 (|VI|)、子句数 (|CI|)、求解状况 (state) 和求解时间 (time)。求解状况分位成功求取密钥 “Verified” 和求取失败 “Failed” 两种。当规定的求解时间内不能找到解时，标记为 “TO”。我们给出了三种 (CNF、DNF、ANF) 不同输出

<sup>6</sup>相关系列求解器获得 SAT 比赛 2022 年 NoLimits 赛道冠军

表6.5 不同条件的CNF编码的第2分位求解情况

Table 6.5 Statistics for CNF encoded from solving class-2 keys with different conditions

k	ANF				CNF				DNF			
	V	Cl	state	time	V	Cl	state	time	V	Cl	state	time
50	11850	23880	Failed	0.03	3500	6131	Failed	0.0	3500	5786	Failed	0.0
55	13161	26799	Failed	0.08	3976	7249	Failed	0.01	3976	6934	Failed	0.01
60	14512	29931	Failed	0.07	4492	8590	Failed	0.07	4492	8223	Failed	0.01
65	15908	33472	Failed	0.63	5053	9995	Failed	0.16	5053	9867	Failed	0.12
70	17360	37670	Failed	3.15	5670	11739	Failed	2.5	5670	11682	Failed	1.73
75	18853	41772	Failed	12.89	6328	13602	Failed	11.13	6328	13611	Failed	4.93
80	20390	45486	Failed	21.16	7030	15707	Failed	30.29	7030	15662	Failed	126.96
81	20707	46414	Failed	11.45	7180	16132	Failed	95.52	7180	16137	Failed	266.02
82	21024	47154	Failed	146.4	7330	16607	Failed	104.56	7330	16563	Failed	160.29
83	21339	48080	Failed	854.37	7478	17030	Failed	172.34	7478	17036	Failed	283.44
84	21659	49023	Failed	592.29	7631	17470	Failed	196.14	7631	17526	Failed	312.1
85	21980	49922	Failed	1281.1	7785	17912	Failed	5759.63	7785	17990	Failed	805.05
86	22301	50655	Failed	1668.71	7939	18358	Failed	440.48	7939	18453	Failed	523.98
87	22623	51383	Failed	7355.32	8094	18823	Failed	1211.52	8094	18890	Failed	2194.72
88	22947	52336	Failed	839.28	8251	19323	Failed	4008.82	8251	19347	Failed	10739.33
89	23280	53322	Verified	1865.66	8417	19814	Failed	3040.48	8417	19882	Failed	TO
90	23611	54112	Failed	TO	8581	20300	Failed	TO	8581	20413	Failed	6957.27
91	23944	55101	Verified	3764.01	8747	20836	Failed	TO	8747	20906	Failed	TO
92	24281	56106	Failed	TO	8917	21388	Verified	3778.87	8917	21415	Failed	TO
93	24619	57115	Failed	TO	9088	21944	Failed	TO	9088	21928	Failed	TO
94	24955	57931	Failed	TO	9257	22500	Failed	TO	9257	22440	Failed	TO
95	25294	58744	Failed	TO	9429	23009	Failed	TO	9429	22994	Failed	TO
96	25634	59573	Failed	TO	9602	23534	Verified	1318.62	9602	23564	Failed	TO
97	25983	60406	Failed	TO	9784	24095	Failed	TO	9784	24112	Failed	TO
98	26330	61404	Verified	2808.13	9964	24651	Failed	TO	9964	24653	Failed	TO
99	26674	62229	Verified	1780.67	10141	25188	Failed	TO	10141	25217	Failed	TO
100	27021	63230	Verified	17169.56	10321	25732	Failed	TO	10321	25784	Verified	8573.3
101	27368	64234	Failed	TO	10501	26279	Failed	TO	10501	26354	Verified	3468.63
102	27716	65092	Failed	TO	10682	26849	Failed	TO	10682	26903	Failed	TO
103	28063	65920	Verified	125.19	10862	27405	Failed	TO	10862	27446	Failed	TO
104	28414	66767	Verified	12110.5	11046	27964	Failed	TO	11046	28032	Failed	TO
105	28779	67880	Failed	TO	11244	28614	Failed	TO	11244	28646	Failed	TO
106	29139	68811	Failed	TO	11437	29212	Failed	TO	11437	29291	Failed	TO
107	29505	69766	Failed	TO	11636	29872	Failed	TO	11636	29914	Failed	TO
108	29872	70728	Failed	TO	11836	30539	Failed	TO	11836	30544	Failed	TO
109	30240	71694	Verified	1519.31	12037	31210	Failed	TO	12037	31178	Failed	TO
110	30613	72845	Failed	TO	12243	31860	Failed	TO	12243	31881	Failed	TO
111	30984	73823	Failed	TO	12447	32543	Failed	TO	12447	32527	Verified	1553.07
112	31361	74987	Verified	14428.25	12657	33244	Failed	TO	12657	33192	Failed	TO
113	31743	76174	Failed	TO	12872	33930	Failed	TO	12872	33931	Verified	10871.1
114	32122	77349	Failed	TO	13084	34604	Failed	TO	13084	34658	Verified	12964.93
115	32501	78356	Failed	TO	13296	35278	Failed	TO	13296	35379	Failed	TO
116	32881	79367	Verified	4967.1	13509	35956	Failed	TO	13509	36104	Verified	14593.3
117	33265	80599	Failed	TO	13726	36737	Failed	TO	13726	36804	Verified	13887.52
118	33649	81831	Failed	TO	13943	37518	Failed	TO	13943	37504	Verified	17799.29
119	34030	83051	Failed	TO	14157	38287	Verified	5568.99	14157	38192	Failed	TO
120	34413	84279	Verified	2771.75	14373	38983	Failed	TO	14373	38975	Failed	TO

表 6.6 不同条件的 CNF 编码的第 3 分位求解情况

Table 6.6 Statistics for CNF encoded from solving class-3 keys with different conditions

k	ANF				CNF				DNF			
	V	C	state	time	V	C	state	time	V	C	state	time
50	41404	83501	Failed	1.31	11354	23860	Failed	0.31	11354	23860	Failed	0.34
51	42282	85231	Failed	1.08	11631	24422	Failed	0.27	11631	24422	Failed	0.27
52	43165	87923	Failed	3.91	11913	25023	Failed	1.11	11913	25023	Failed	1.14
53	44050	89692	Failed	1.37	12197	25633	Failed	0.86	12197	25633	Failed	0.77
54	44936	92393	Failed	5.86	12482	26243	Failed	1.39	12482	26243	Failed	1.62
55	45822	94166	Failed	7.99	12767	27099	Failed	2.33	12767	27099	Failed	2.37
56	46713	95950	Failed	5.3	13057	27966	Failed	0.45	13057	27966	Failed	0.45
57	47609	97760	Failed	5.01	13352	28859	Failed	5.32	13352	28859	Failed	5.33
58	48506	99571	Failed	30.46	13648	29511	Failed	8.41	13648	29511	Failed	7.76
59	49405	101390	Failed	57.61	13946	30413	Failed	1.88	13946	30413	Failed	1.79
60	50309	103229	Failed	5.58	14249	31335	Failed	19.57	14249	31335	Failed	19.38
61	51214	105075	Failed	108.13	14553	32264	Failed	21.97	14553	32264	Failed	21.86
62	52118	106917	Failed	78.85	14856	32947	Failed	68.09	14856	32947	Failed	67.53
63	53022	108759	Failed	175.9	15159	33630	Failed	9.84	15159	33630	Failed	10.6
64	53927	110608	Failed	466.17	15463	34320	Failed	22.12	15463	34320	Failed	22.11
65	54840	113554	Failed	417.94	15775	35076	Failed	242.91	15775	35076	Failed	244.66
66	55752	116499	Failed	608.84	16086	35831	Failed	69.19	16086	35831	Failed	68.26
67	56668	119454	Verified	<b>147.38</b>	16401	37064	Failed	186.45	16401	37064	Failed	189.05
68	57587	122283	Failed	586.32	16719	38069	Failed	79.48	16719	38069	Failed	80.35
69	58511	125123	Failed	1141.17	17042	39085	Failed	498.0	17042	39085	Failed	501.83
70	59432	127957	Failed	13605.42	17362	40095	Failed	6989.38	17362	40095	Failed	7881.51
71	60352	130787	Failed	10915.3	17681	40844	Failed	14676.26	17681	40844	Failed	14735.2
72	61276	132788	Failed	23693.73	18004	41603	Failed	1943.73	18004	41603	Failed	1952.37
73	62210	135668	Failed	77051.58	18337	42659	Failed	134274.61	18337	42659	Failed	135138.75
74	63142	138546	Failed	120412.18	18668	43713	Failed	89561.7	18668	43713	Failed	89372.12
75	64076	141432	Failed	145774.26	19001	44518	Failed	TO	19001	44518	Failed	TO
76	65013	143488	Failed	TO	19337	45332	Failed	106158.67	19337	45332	Failed	105558.47
77	65954	146399	Failed	20599.83	19677	46162	Verified	24793.0	19677	46162	Verified	24959.21
78	66894	148464	Failed	TO	20016	46985	Verified	60632.47	20016	46985	Verified	60324.78
79	67831	150523	Failed	TO	20352	48052	Failed	TO	20352	48052	Failed	TO
80	68771	152594	Failed	TO	20691	48881	Failed	TO	20691	48881	Failed	TO
81	69718	154690	Failed	TO	21037	49985	Failed	TO	21037	49985	Failed	TO
82	70668	156795	Failed	TO	21386	51098	Failed	TO	21386	51098	Failed	TO
83	71617	158899	Verified	43452.98	21734	51960	Failed	TO	21734	51960	Failed	TO
84	72570	161016	Failed	TO	22086	53085	Failed	TO	22086	53085	Failed	TO
85	73524	163066	Verified	33409.89	22439	54076	Failed	TO	22439	54076	Failed	TO
86	74478	165110	Failed	TO	22792	54931	Failed	TO	22792	54931	Failed	TO
87	75430	167146	Failed	TO	23143	55778	Failed	TO	23143	55778	Failed	TO
88	76386	170112	Failed	TO	23498	56919	Failed	TO	23498	56919	Failed	TO
89	77347	172244	Failed	TO	23858	58077	Failed	TO	23858	58077	Failed	TO
90	78307	175229	Failed	TO	24217	58981	Failed	TO	24217	58981	Failed	TO

函数非线性表达式编码下的求解情况。因为第2分位的输出函数 $f_2$ 缺失了2位的输入， $f_2$ 的表达式较为简单，相对第3分位的求取较易，我们设置的第2分位的截止时间为5小时，第3分位的截止时间为2天。

我们可以根据数据得到如下的结论：

- 非线性函数以ANF表示时，求解效果最好，其次为以DNF表示的非线性函数的方法。随着 $k$ 的增加，变量和子句线性增加，找到真解的可能性越高，但是求解器难度也会发生明显的增加。
- $k$ 值较小时求解难度较小，但是随着接近可求解出初态时的前几轮时，求解难度会发生明显变化。此时可行解数量较少，但是也有可能会产生一组真解。
- 如果采用ANF编码的方法，串行单核心求取第2分位的时间大约为5个小时，串行单核心求取第3分位的时间大约为50分钟。
- 如果采用上一节提到的第2种优化策略，一次性生成不同 $k$ 值的CNF，并采用并行求解的方法，在本文提及的服务器上可以同时求解128个CNF，故2、3分位初态求解均可以在百秒时间量级恢复初态。

赛题主办方邀请了国内来自国防科大、上海交通大学、山东大学、中科院信工所、软件所、数学所等密码学研究的顶尖团队来共同挑战该赛题，经过长达半年的攻坚，进入决赛的队伍只有两支队伍可以完整恢复所有分位初态。其他队伍采用的密码分析方法大多采用了MILP（混合整数线性规划）等传统分析方法，在上千核心的超级计算机上进行密码分析，最快仍需要数周的时间才可完成整个初态恢复流程。相比之下，我们的方法可以在单核计算机上数小时完成密码分析，成功恢复了初态。因此，获得了该赛事的全国“冠军”。

**讨论** Guess and Determine（猜测确定，也称先猜后定）算法是流密码分析中常用的一种分析方法。在猜测（Guess）部分，密码分析算法会基于密码分析者对密码算法的了解、对密钥生成的规律，尝试猜测一部分密钥或者算法内部状态。在确定（Determine）部分，密码分析者会利用已知的明密文和算法的其他信息来验证猜测的正确性。如果正确则可以利用该信息进一步破解密码或者恢复更多的密钥信息；如果猜测错误，那么就需要回到猜测阶段，重新进行猜测验证。我们的方法不存在猜测的阶段，但是存在验证的阶段。我们的方法会通过逐渐增加约束的方法来使得求解的结果变的更加精确。类似于猜测确定，本文方法

同样也有可能产生错误，但是错误不是来源于猜测，而是来自于缺少部分约束。此外，使用本文提出的方法时，当约束过少时会产生假解，当约束过多则会使得求解过难，因此当约束增加时，可以引入猜测确定的方法来加速验证的速度，以利用两种方法的互补性。

### 6.3 本章小结

在本章第一节中，我们实现了一个 CEC 证明器 hybridCEC，它通过在 SAT-sweeping 框架中集成了精确概率仿真引擎（EPS）技术，设计了一种 SAT 求解器与 EPS 引擎的混合算法。由于内存限制，以前的 EPS 引擎只能处理 24 个 PI 内的电路，我们通过截断输入信号值来优化内存开销，使 EPS 引擎适用于任何数量 PI 的逻辑电路。为了在 sweeping 过程中使 EPS 引擎和 SAT 求解器两类方法达到更好的协作效果，我们设计了一种基于异或链密度的选择启发式算法。最后，我们也基于数据通路电路中的“正则性”对 hybridCEC 进行了改进。我们在工业数据通路样例集上进行了大量的实验，证明了 hybridCEC 的有效性。在未来，我们计划开发 hybridCEC 的并行版本，并尝试将本文提出的技术迁移到时序等价性检查问题中。

本章第二节以“环上前馈序列密码”问题为案例，讲解了一套使用混合 SAT 求解器解决实际问题的工作流程。求解过程中，主要采用了一种轮增量的密码学编码方法，并成功的在合理的资源条件下快速恢复了流密码的初态。凭借该结果，相关技术在与多组国内顶尖的密码学团队的竞技中，取得了“强网杯”密码数学专项赛的全国冠军。这启示我们将该技术迁移到更多相关的流密码问题上去，例如 Trivium 算法。此外，我们也考虑将先猜后定和 Groebnor 基等技术应用到密码学的求解过程中，设计一个混合算法，来提高求解能力，将 SAT 混合求解器适配更多的密码学问题。

## 第7章 总结与展望

本章对本文中的所有工作的算法进行了总结，并给出了未来的研究方向。

### 7.1 工作总结

可满足性 (SAT) 问题是计算机科学的核心问题，在芯片设计和密码学中有重要应用价值；可满足性模理论 (SMT) 问题拓展了一阶逻辑背景理论，有更好的建模、表述能力，更适合建模软件工程问题。SAT 问题和 SMT 问题的算法可以分为系统搜索算法和局部搜索算法两类，本文主要研究了 SAT 问题和 SMT 问题的混合求解算法以及相关的应用问题，取得了系列成果，总结如下：

第三章中，本文针对“重启之后学习信息是否需要保留”这一关键问题开展了研究，将传统保留变元顺序、相位和学习子句的重启策略称为热重启，遗忘部分信息的重启策略称为冷重启，并设计了不同策略的冷热重启的混合重启策略。本文在分析之后总结了不同重启策略的优劣并用于提高主流的求解器。我们也利用主流求解器中仍存在求解时间的重尾分布现象，利用互补性设计了并行的冷重启策略，相关算法在国际 SAT 比赛的并行赛道中以领先亚军  $1/4$  性能的优势取得了并行组的冠军。

第四章中，本文设计了一种 SAT 问题的 CDCL 与局部搜索间的深度混合算法。本文创新性地提出了一种以 CDCL 为主体，并在希望分支上调用 SLS 算法进行搜索空间采样的方法。且 CDCL 与局部搜索算法之前会进行深度的信息交流。CDCL 会通过进入一种松弛的非回退过程，构造一个高一致性的完全赋值，然后作为局部搜索算法的初始解。另一方面，局部搜索会搜集过程中遇到的高一致性赋值和变元冲突频率，用于改进 CDCL 算法的变元分支启发式与相位决策启发式。该方法大幅提高了基求解器的性能，解决了大量基求解器无法求解的实例，首次解决的 Bart Selman 命题逻辑十大挑战的第七项，并依次取得了 SAT 会议“最佳论文奖”。相关求解器在国际 SAT 比赛中取得了冠军。

第五章中，本文将第四章中的技术迁移到了 SMT(IA) 问题中，并优化了在求解软件验证问题上的求解效果。本文提出了一种双层的混合策略。外层为 CDCL(T) 与局部搜索算法的交替调用。内层则是一种以 CDCL(T) 为主，并在遇

到希望分支时，调用局部搜索算法进行空间采样的方法。CDCL(T) 会传递给局部搜索算法高质量的子公式，而局部搜索算法也会利用过程中收集到的高一致性赋值和冲突频率来改进 CDCL(T) 的启发式，且两者间的信息传递会经过一定的策略转化，以适配 SMT 问题的抽象命题逻辑骨架。大量的实验证明了我们的方法的有效性，且实验表明本文的方法在终止性和非终止性验证领域有重要的应用价值。此外，相关求解器参与了 SMT 比赛，获得了“最大贡献奖”、“最大领先奖”和数个细分赛道的冠军。

第六章中，本文阐述了 SAT 问题在组合等价性验证（CEC）和密码分析问题的应用。首先，针对包含大量数据通路的 CEC 问题，本文提出将一种精确概率仿真的策略嵌入到以 SAT-sweeping 为主体的 CEC 算法中，形成了一种专用于求解数据通路 CEC 问题的混合求解算法。该求解器可以在一部分来自工业界的实例上，相较主流算法，提速超过一千倍。另一方面，本文以“环上前馈序列密码”作为研究案例，设计了一套“轮增量编码”的流密码分析算法，我们的方法可以在单核心数小时内完成初态恢复的任务，凭借该成果，我们取得了“强网杯”密码数学专项赛的全国冠军。

## 7.2 未来工作

未来我们拟从如下的几个角度开展工作。

**1. 传统组件策略的回顾：**第三章中通过对重启策略的一个固有思维范式进行挑战，并提出了一种改进策略，这启发我们去针对特定的应用环境下的实例集合，开发特定的专用求解策略。最近 ChatGPT 等大语言模型技术在大量实际问题上得到了应用，起到了类似于强化学习的作用，这启发我们采用深度强化学习和大语言模型来改进 SAT 和 SMT 求解器的组件组合策略。

**2. 混合算法的进一步拓展：**第四、五章中的方法是通过混合技术改进求解性能的一次成功的尝试。该系列的混合策略有希望于迁移到 QBF、MaxSAT、CSP、MILP 等其他类似的问题中去。针对 SMT 问题，本文的工作目前仅限于 IA 理论上，CDCL(T) 与局部搜索的混合算法研究，缺少对 SMT 其它背景理论的混合算法研究，也缺少了针对 SMT 问题的其他求解器策略间的混合算法研究，例如 NLSat 与局部搜索的混合算法的研究。

**3. 应用的拓展研究：**第六章中，本文介绍了 SAT 问题在 EDA 问题和密码分

析中的应用。未来，本人将继续研究 SAT、SMT 求解技术在时序等价性验证、逻辑综合、模型检查等领域的应用，也将继续探索如何将我们的密码分析方法应用于真实场景下的密码算法中。



## 参考文献

- [1] Biere A, Heule M, van Maaren H. Handbook of satisfiability: volume 185 [M]. IOS press, 2009.
- [2] Church A. The calculi of lambda-conversion: number 6 [M]. Princeton University Press, 1985.
- [3] Turing A M, et al. On computable numbers, with an application to the entscheidungsproblem [J]. J. of Math, 1936, 58(345-363): 5.
- [4] Kleene S C. Recursive predicates and quantifiers [J]. Transactions of the American Mathematical Society, 1943, 53(1): 41-73.
- [5] Henkin L. The completeness of the first-order functional calculus [J]. The Journal of Symbolic Logic, 1949, 14(3): 159-166.
- [6] Davis M, Putnam H. A computing procedure for quantification theory [J]. Journal of the ACM (JACM), 1960, 7(3): 201-215.
- [7] Davis M, Logemann G, Loveland D W. A machine program for theorem-proving [J]. Commun. ACM, 1962, 5(7): 394-397.
- [8] Shannon C E. A symbolic analysis of relay and switching circuits [J]. Electrical Engineering, 1938, 57(12): 713-723.
- [9] Cook S A. The complexity of theorem-proving procedures [C]//Proceedings of the third annual ACM symposium on Theory of computing. 1971: 151-158.
- [10] Weber T. Smt solvers: New oracles for the hol theorem prover [J]. International Journal on Software Tools for Technology Transfer, 2011, 13(5): 419-429.
- [11] Paulson L C. Isabelle: A generic theorem prover [M]. Springer, 1994.
- [12] Peng Y, Greenstreet M. Extending acl2 with smt solvers [J]. arXiv preprint arXiv:1509.06082, 2015.
- [13] Armand M, Faure G, Grégoire B, et al. A modular integration of sat/smt solvers to coq through proof witnesses [C]//International Conference on Certified Programs and Proofs. Springer, 2011: 135-150.
- [14] Heule M J, Kullmann O, Marek V W. Solving and verifying the boolean pythagorean triples problem via cube-and-conquer [C]//International Conference on Theory and Applications of Satisfiability Testing 2016. 2016: 228-245.
- [15] Heule M. Schur number five [C]//Proceedings of AAAI Conference on Artificial Intelligence 2018: volume 32. 2018.

- [16] Brakensiek J, Heule M, Mackey J, et al. The resolution of keller' s conjecture [J]. Journal of Automated Reasoning, 2022, 66(3): 277-300.
- [17] Bright C, Cheung K K, Stevens B, et al. A sat-based resolution of lam's problem [C]// Proceedings of the AAAI Conference on Artificial Intelligence: volume 35. 2021: 3669-3676.
- [18] 黄沛. 逻辑公式可满足性判定中的推理技术研究 [D]. 中国科学院软件研究所, 2022.
- [19] 刘明昊. 约束求解的符号与神经方法 [D]. 中国科学院软件研究所, 2023.
- [20] Newman N, Fréchette A, Leyton-Brown K. Deep optimization for spectrum repacking [J]. Commun. ACM, 2018, 61(1): 97-104.
- [21] Stachowiak S, Kurkowski M, Soboń A. Sat-based cryptanalysis of salsa20 cipher [C]// Progress in Image Processing, Pattern Recognition and Communication Systems: Proceedings of the Conference (CORES, IP&C, ACS)-June 28-30 2021 12. Springer, 2022: 252-266.
- [22] 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.
- [23] Brayton R, Jiang J H R, Jang S. Sat-based logic optimization and resynthesis [C]//Proceeding of International Workshop on Logic Synthesis. 2007: 358-364.
- [24] Haaswijk W J. Sat-based exact synthesis for multi-level logic networks [R]. EPFL, 2019.
- [25] Huang J, Zhen H L, Wang N, et al. Neural fault analysis for sat-based atpg [C]//2022 IEEE International Test Conference (ITC). IEEE, 2022: 36-45.
- [26] Goldberg E I, Prasad M R, Brayton R K. Using sat for combinational equivalence checking [C]//Proceedings Design, Automation and Test in Europe. Conference and Exhibition 2001. IEEE, 2001: 114-121.
- [27] Possani V N, Mishchenko A, Ribas R P, et al. Parallel combinational equivalence checking [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39(10): 3081-3092.
- [28] Sagahyroon A, Aloul F A. Using sat-based techniques in power estimation [J]. Microelectronics Journal, 2007, 38(6-7): 706-715.
- [29] Huang P, Wu H, Yang Y, et al. Towards efficient verification of quantized neural networks [Z]. 2023.
- [30] Heizmann M, Hoenicke J, Podelski A. Software model checking for people who love automata [C]//Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25. Springer, 2013: 36-52.
- [31] Beyene T, Chaudhuri S, Popescu C, et al. A constraint-based approach to solving games on infinite graphs [C]//Proceedings of the 41st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. 2014: 221-233.

- 
- [32] Tiwari A, Gascón A, Dutertre B. Program synthesis using dual interpretation [C]// International Conference on Automated Deduction. Springer, 2015: 482-497.
  - [33] Heizmann M, Dietsch D, Leike J, et al. Ultimate automizer with array interpolation: (competition contribution) [C]//Tools and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21. Springer, 2015: 455-457.
  - [34] Colón M A, Sankaranarayanan S, Sipma H B. Linear invariant generation using non-linear constraint solving [C]//Computer Aided Verification: 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003. Proceedings 15. Springer, 2003: 420-432.
  - [35] Leike J, Heizmann M. Geometric series as nontermination arguments for linear lasso programs [J]. arXiv preprint arXiv:1405.4413, 2014.
  - [36] Cook B, Podelski A, Rybalchenko A. Termination proofs for systems code [J]. ACM Sigplan Notices, 2006, 41(6): 415-426.
  - [37] De Moura L, Bjørner N. Z3: An efficient smt solver [C]//International conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2008: 337-340.
  - [38] Barbosa H, Barrett C, Brain M, et al. cvc5: A versatile and industrial-strength smt solver [C]//International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2022: 415-442.
  - [39] Dutertre B. Yices 2.2 [C]//International Conference on Computer Aided Verification. Springer, 2014: 737-744.
  - [40] Henzinger T A, Jhala R, Majumdar R, et al. Software verification with blast [C]//Model Checking Software: 10th International SPIN Workshop Portland, OR, USA, May 9–10, 2003 Proceedings 10. Springer, 2003: 235-239.
  - [41] Cadar C, Dunbar D, Engler D R, et al. Klee: unassisted and automatic generation of high-coverage tests for complex systems programs. [C]//USENIX Symposium on Operating Systems Design and Implementations: volume 8. 2008: 209-224.
  - [42] Moskewicz M W, Madigan C F, Zhao Y, et al. Chaff: Engineering an efficient sat solver [C]// Proceedings of the 38th annual Design Automation Conference. 2001: 530-535.
  - [43] Oh C. Between sat and unsat: the fundamental difference in cdcl sat [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2015: 307-323.
  - [44] Eén N, Sörensson N. An extensible sat-solver [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2003: 502-518.
  - [45] Biere A, Heule M. The effect of scrambling cnfs [C]//Pragmatics of SAT: volume 59. 2019: 111-126.

- [46] Huang J, et al. The effect of restarts on the efficiency of clause learning. [C]//IJCAI: volume 7. 2007: 2318-2323.
- [47] Biere A, Fleury M, Froleyks N, et al. The sat museum [J]. 2023.
- [48] Cai S, Luo C, Su K. Ccanr: A configuration checking based local search solver for non-random satisfiability [C]//Theory and Applications of Satisfiability Testing—SAT 2015: 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings 18. Springer, 2015: 1-8.
- [49] Biere A. Yet another local search solver and lingeling and friends entering the sat competition 2014 [J]. Proceedings of SAT Competition, 2014, 2014(2): 65.
- [50] Biere A, Fazekas K, Fleury M, et al. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020 [C]//51-53.
- [51] Biere A. Cadical, lingeling, plingeling, treengeling and yalsat entering the sat competition 2018 [J]. Proceedings of SAT Competition, 2017, 14: 316-336.
- [52] Soos M, Nohl K, Castelluccia C. Extending sat solvers to cryptographic problems [C]// International Conference on Theory and Applications of Satisfiability Testing. Springer, 2009: 244-257.
- [53] Behrmann K, Dyck A, Emmes F, et al. Bit-blasting for smt-nia with aprove [J]. Proceedings of SMT Competition, 2014, 14.
- [54] Jovanovic D, Barrett C, De Moura L. The design and implementation of the model constructing satisfiability calculus [C]//2013 Formal Methods in Computer-Aided Design. IEEE, 2013: 173-180.
- [55] Cimatti A, Griggio A, Irfan A, et al. Incremental linearization: A practical approach to satisfiability modulo nonlinear arithmetic and transcendental functions [C]//2018 20th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). IEEE, 2018: 19-26.
- [56] Cai S, Li B, Zhang X. Local search for smt on linear integer arithmetic [C]//International Conference on Computer Aided Verification. Springer, 2022: 227-248.
- [57] Jia F, Han R, Huang P, et al. Improving bit-blasting for nonlinear integer constraints [C]// Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis. 2023: 14-25.
- [58] Li H, Xia B, Zhao T. Local search for solving satisfiability of polynomial formulas [C]// International Conference on Computer Aided Verification. Springer, 2023: 87-109.
- [59] Wang Z, Zhan B, Li B, et al. Efficient local search for nonlinear real arithmetic [C]// International Conference on Verification, Model Checking, and Abstract Interpretation. Springer, 2023: 326-349.

- 
- [60] Tseitin G S. On the complexity of derivation in propositional calculus [J]. Automation of reasoning: 2: Classical papers on computational logic 1967–1970, 1983: 466-483.
  - [61] Barrett C, Stump A, Tinelli C. The SMT-LIB Standard: Version 2.0 [R]. Department of Computer Science, The University of Iowa, 2010.
  - [62] Kroening D, Strichman O. Decision procedures [M]. Springer, 2016.
  - [63] Impagliazzo R, Paturi R. On the complexity of k-sat [J]. Journal of Computer and System Sciences, 2001, 62(2): 367-375.
  - [64] Buss S, Nordström J. Proof complexity and sat solving. [J]. Handbook of Satisfiability, 2021, 336: 233-350.
  - [65] Järvisalo M, Matsliah A, Nordström J, et al. Relating proof complexity measures and practical hardness of sat [C]//International Conference on Principles and Practice of Constraint Programming. Springer, 2012: 316-331.
  - [66] 周俊萍. 自动推理与规划问题最小上界和相变规律研究 [D]. 吉林大学, 2011.
  - [67] 符祖峰. d-正则 (k,s)-SAT 问题的临界现象与结构性质 [D]. 贵州大学, 2020.
  - [68] Audemard G, Simon L. Predicting learnt clauses quality in modern sat solvers [C]//Twenty-first international joint conference on artificial intelligence. Citeseer, 2009.
  - [69] Pretolani D, et al. Efficiency and stability of hypergraph sat algorithms [M]//Cliques, coloring, and satisfiability: second DIMACS implementation challenge: volume 26. American Mathematical Society, 1996: 479-498.
  - [70] Biere A, Fröhlich A. Evaluating cdcl variable scoring schemes [C]//Theory and Applications of Satisfiability Testing—SAT 2015: 18th International Conference, Austin, TX, USA, September 24–27, 2015, Proceedings 18. Springer, 2015: 405-422.
  - [71] Ryan L. Efficient algorithms for clause-learning sat solvers [J]. 2004.
  - [72] Shaw A, Meel K S. Designing new phase selection heuristics [C]//Theory and Applications of Satisfiability Testing—SAT 2020: 23rd International Conference, Alghero, Italy, July 3–10, 2020, Proceedings 23. Springer, 2020: 72-88.
  - [73] Liang J H, Ganesh V, Zulkoski E, et al. Understanding vsids branching heuristics in conflict-driven clause-learning sat solvers [C]//Hardware and Software: Verification and Testing: 11th International Haifa Verification Conference, HVC 2015, Haifa, Israel, November 17-19, 2015, Proceedings 11. Springer, 2015: 225-241.
  - [74] Liang J H, Ganesh V, Poupart P, et al. Learning rate based branching heuristic for sat solvers [C]//Theory and Applications of Satisfiability Testing—SAT 2016: 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings 19. Springer, 2016: 123-140.
  - [75] Pipatsrisawat K, Darwiche A. A lightweight component caching scheme for satisfiability solvers [C]//Proceedings of SAT 2007. 2007: 294-299.

- [76] Goldberg E, Novikov Y. Berkmin: A fast and robust sat-solver [J]. Discrete Applied Mathematics, 2007, 155(12): 1549-1561.
- [77] Biere A. Lingeling and friends at the sat competition 2011 [J]. Proceedings of SAT Competition, 2011.
- [78] QUEUE S D. Cadical at the sat race 2019 [J]. Proceedings of SAT Race, 2019, 2019: 8.
- [79] Liang J, Ganesh V, Poupart P, et al. Exponential recency weighted average branching heuristic for sat solvers [C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 30. 2016.
- [80] Sleator D D, Tarjan R E. Amortized efficiency of list update and paging rules [J]. Communications of the ACM, 1985, 28(2): 202-208.
- [81] Gomes C P, Selman B, Crato N. Heavy-tailed distributions in combinatorial search [C]// International Conference on Principles and Practice of Constraint Programming. Springer, 1997: 121-135.
- [82] Biere A, Fröhlich A. Evaluating cdcl restart schemes [J]. Proceedings of Pragmatics of SAT, 2015: 1-17.
- [83] Luby M, Sinclair A, Zuckerman D. Optimal speedup of las vegas algorithms [J]. Information Processing Letters, 1993, 47(4): 173-180.
- [84] Audemard G, Simon L. Refining restarts strategies for sat and unsat [C]//Principles and Practice of Constraint Programming: 18th International Conference, CP 2012, Québec City, QC, Canada, October 8-12, 2012. Proceedings. Springer, 2012: 118-126.
- [85] Jabbour S, Lonlac J, Sais L, et al. Revisiting the learned clauses database reduction strategies [J]. arXiv preprint arXiv:1402.1956, 2014.
- [86] Één N, Biere A. Effective preprocessing in sat through variable and clause elimination [C]// International Conference on Theory and Applications of Satisfiability Testing. Springer, 2005: 61-75.
- [87] Sörensson N. Minisat 2.2 and minisat++ 1.1 [J]. A short description in SAT Race, 2010, 2010.
- [88] Järvisalo M, Biere A, Heule M. Blocked clause elimination [C]//Tools and Algorithms for the Construction and Analysis of Systems: 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings 16. Springer, 2010: 129-144.
- [89] Bacchus F. Enhancing davis putnam with extended binary clause reasoning [J]. Proceedings of AAAI Conference on Artificial Intelligence, 2002, 2002: 613-619.
- [90] Bacchus F, Winter J. Effective preprocessing with hyper-resolution and equality reduction

- [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2003: 341-355.
- [91] Järvisalo M, Biere A, Heule M J. Simulating circuit-level simplifications on cnf [J]. Journal of Automated Reasoning, 2012, 49(4): 583-619.
- [92] Han H, Somenzi F. Alembic: An efficient algorithm for cnf preprocessing [C]//Proceedings of the 44th annual Design Automation Conference. 2007: 582-587.
- [93] Piette C, Hamadi Y, Sais L. Vivifying propositional clausal formulae [M]//Proceedings of the European Conference on Artificial Intelligence 2008. IOS Press, 2008: 525-529.
- [94] Luo M, Li C M, Xiao F, et al. An effective learnt clause minimization approach for cdcl sat solvers [C]//Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI-17). 2017: 703-711.
- [95] Järvisalo M, Heule M J, Biere A. Inprocessing rules [C]//International Joint Conference on Automated Reasoning. Springer, 2012: 355-370.
- [96] Biere A. Preprocessing and inprocessing techniques in sat. [C]//Haifa Verification Conference: volume 1. 2011.
- [97] Wotzlaw A, van der Grinten A, Speckenmeyer E. Effectiveness of pre-and inprocessing for cdcl-based sat solving [J]. arXiv preprint arXiv:1310.4756, 2013.
- [98] Beame P, Kautz H, Sabharwal A. Towards understanding and harnessing the potential of clause learning [J]. Journal of artificial intelligence research, 2004, 22: 319-351.
- [99] Sörensson N, Biere A. Minimizing learned clauses [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2009: 237-243.
- [100] Zhang H, Stickel M. Implementing the davis–putnam method [J]. Journal of Automated Reasoning, 2000, 24(1-2): 277-296.
- [101] Brafman R I. A simplifier for propositional formulas with many binary clauses [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004, 34(1): 52-59.
- [102] Zhang L. On subsumption removal and on-the-fly cnf simplification [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2005: 482-489.
- [103] Biere A. Resolve and expand [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2004: 59-70.
- [104] Biere A, Järvisalo M, Kiesl B. Preprocessing in sat solving. [J]. Handbook of satisfiability, 2021, 336: 391-435.
- [105] Hoos H H, Stützle T. Stochastic local search: Foundations and applications [M]. Elsevier, 2004.
- [106] Li C M, Li Y. Satisfying versus falsifying in local search for satisfiability [C]//Theory and Ap-

- plications of Satisfiability Testing—SAT 2012: 15th International Conference, Trento, Italy, June 17-20, 2012. Proceedings 15. Springer, 2012: 477-478.
- [107] Balint A, Schöning U. Choosing probability distributions for stochastic local search and the role of make versus break [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2012: 16-29.
- [108] Cai S, Luo C, Lin J, et al. New local search methods for partial maxsat [J]. Artificial Intelligence, 2016, 240: 1-18.
- [109] Luo C, Cai S, Su K, et al. Ccehc: An efficient local search algorithm for weighted partial maximum satisfiability [J]. Artificial Intelligence, 2017, 243: 26-44.
- [110] Cai S, Luo C, Zhang X, et al. Improving local search for structured sat formulas via unit propagation based construct and cut initialization (short paper) [C]//27th International Conference on Principles and Practice of Constraint Programming (CP 2021). Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.
- [111] Audemard G, Lagniez J M, Mazure B, et al. Boosting local search thanks to cdcl [C]//Logic for Programming, Artificial Intelligence, and Reasoning: 17th International Conference, LPAR-17, Yogyakarta, Indonesia, October 10-15, 2010. Proceedings 17. Springer, 2010: 474-488.
- [112] Mazure B, Sais L, Grégoire É. Boosting complete techniques thanks to local search methods [J]. Annals of mathematics and artificial intelligence, 1998, 22: 319-331.
- [113] Zhang W, Sun Z, Zhu Q, et al. Nlocalsat: Boosting local search with solution prediction [J]. arXiv preprint arXiv:2001.09398, 2020.
- [114] Cai S. Novel local search methods for satisfiability [D]. Griffith University, Australia, 2014.
- [115] Gent I P, Walsh T, et al. Towards an understanding of hill-climbing procedures for sat [C]// Proceedings of AAAI Conference on Artificial Intelligence: volume 93. 1993: 28-33.
- [116] Mitchell D, Selman B, Leveque H. A new method for solving hard satisfiability problems [C]//Proceedings of the tenth national conference on artificial intelligence (AAAI-92). 1992: 440-446.
- [117] Selman B, Kautz H A, Cohen B, et al. Noise strategies for improving local search [C]// Proceedings of AAAI Conference on Artificial Intelligence: volume 94. 1994: 337-343.
- [118] Cai S, Su K. Local search for boolean satisfiability with configuration checking and subscore [J]. Artificial Intelligence, 2013, 204: 75-98.
- [119] Schuurmans D, Southey F. Local search characteristics of incomplete sat procedures [J]. Artificial Intelligence, 2001, 132(2): 121-150.
- [120] Schuurmans D, Southey F, Holte R C, et al. The exponentiated subgradient algorithm for heuristic boolean programming [C]//IJCAI. 2001: 334-341.
- [121] Hutter F, Tompkins D A, Hoos H H. Scaling and probabilistic smoothing: Efficient dynamic

- local search for sat [C]//Principles and Practice of Constraint Programming-CP 2002: 8th International Conference, CP 2002 Ithaca, NY, USA, September 9–13, 2002 Proceedings 8. Springer, 2002: 233-248.
- [122] Thornton J, Pham D N, Bain S, et al. Additive versus multiplicative clause weighting for sat [C]//Proceedings of AAAI Conference on Artificial Intelligence: volume 4. 2004: 191-196.
- [123] Cai S, Luo C, Su K. Ccasat: solver description [J]. Proceedings of SAT challenge, 2012, 2012: 13.
- [124] Gableske O, Heule M J. Eagleup: Solving random 3-sat using sls with unit propagation [C]// Theory and Applications of Satisfiability Testing-SAT 2011: 14th International Conference, SAT 2011, Ann Arbor, MI, USA, June 19-22, 2011. Proceedings 14. Springer, 2011: 367-368.
- [125] Cai S, Su K, Sattar A. Local search with edge weighting and configuration checking heuristics for minimum vertex cover [J]. Artificial Intelligence, 2011, 175(9-10): 1672-1696.
- [126] Luo C, Cai S, Su K, et al. Clause states based configuration checking in local search for satisfiability [J]. IEEE transactions on Cybernetics, 2014, 45(5): 1028-1041.
- [127] McAllester D, Selman B, Kautz H, et al. Evidence for invariants in local search [C]// Proceedings of AAAI Conference on Artificial Intelligence. Rhode Island, USA, 1997: 321-326.
- [128] Li C M, Wei W, Zhang H. Combining adaptive noise and look-ahead in local search for sat [C]//Theory and Applications of Satisfiability Testing–SAT 2007: 10th International Conference, Lisbon, Portugal, May 28-31, 2007. Proceedings 10. Springer, 2007: 121-133.
- [129] Li C M, Huang W Q. Diversification and determinism in local search for satisfiability [C]// Theory and Applications of Satisfiability Testing: 8th International Conference, SAT 2005, St Andrews, UK, June 19-23, 2005. Proceedings 8. Springer, 2005: 158-172.
- [130] Selman B, Kautz H, et al. Domain-independent extensions to gsat: Solving large structured satisfiability problems [C]//IJCAI: volume 93. 1993: 290-295.
- [131] Balint A, Fröhlich A. Improving stochastic local search for sat with a new probability distribution [C]//Theory and Applications of Satisfiability Testing–SAT 2010: 13th International Conference, SAT 2010, Edinburgh, UK, July 11-14, 2010. Proceedings 13. Springer, 2010: 10-15.
- [132] Manthey N. Parallel sat solving-using more cores [J]. Pragmatics of SAT, 2011.
- [133] 吴贯锋. 关于并行 SAT 求解器和并行自动演绎推理系统的研究 [D]. 西南交通大学, 2019.
- [134] Wieringa S, Heljanko K. Concurrent clause strengthening [C]//International Conference on Theory and Applications of Satisfiability Testing 2013. 2013: 116-132.

- [135] Gebhardt K, Manthey N. Parallel variable elimination on cnf formulas [C]//Annual Conference on Artificial Intelligence 2013. 2013: 61-73.
- [136] Osama M, Wijs A. Parallel sat simplification on gpu architectures [C]//TACAS 2019. 2019: 21-40.
- [137] Osama M, Wijs A. Gpu acceleration of bounded model checking with para frost [C]// International Conference on Computer Aided Verification. Springer, 2021: 447-460.
- [138] Hamadi Y, Jabbour S, Piette C, et al. Deterministic parallel dpll [J]. Journal on Satisfiability, Boolean Modeling and Computation, 2011, 7(4): 127-132.
- [139] Singer D, Monnet A. Jack-sat: A new parallel scheme to solve the satisfiability problem (sat) based on join-and-check [C]//PPAM 2007. 2007: 249-258.
- [140] Audemard G, Lagniez J M, Szczechanski N, et al. An adaptive parallel sat solver [C]// International Conference on Principles and Practice of Constraint Programming 2016. 2016: 30-48.
- [141] Heule M J, Kullmann O, Wieringa S, et al. Cube and conquer: Guiding cdcl sat solvers by lookaheads [C]//Haifa Verification Conference 2011. 2011: 50-65.
- [142] Nejati S, Newsham Z, Scott J, et al. A propagation rate based splitting heuristic for divide-and-conquer solvers [C]//International Conference on Theory and Applications of Satisfiability Testing. 2017: 251-260.
- [143] Nejati S, Le Frioux L, Ganesh V. A machine learning based splitting heuristic for divide-and-conquer solvers [C]//International Conference on Principles and Practice of Constraint Programming 2020. 2020: 899-916.
- [144] Chrabakh W, Wolski R. Gridsat: A chaff-based distributed sat solver for the grid [C]// Proceedings of SAT Competition 2003. 2003: 37-37.
- [145] Hyvärinen A E, Junttila T, Niemelä I. Partitioning sat instances for distributed solving [C]// International Conference on Logic for Programming Artificial Intelligence and Reasoning. 2010: 372-386.
- [146] Hyvärinen A E J, Junttila T A, Niemelä I. Incorporating clause learning in grid-based randomized SAT solving [J]. Journal on Satisfiability, Boolean Modeling and Computation, 2009, 6(4): 223-244.
- [147] Heule M, van Maaren H. Look-ahead based sat solvers. [J]. Handbook of satisfiability, 2009, 185: 155-184.
- [148] Eén N, Sörensson N. Temporal induction by incremental sat solving [J]. Electronic Notes in Theoretical Computer Science, 2003, 89(4): 543-560.
- [149] Heisinger M, Fleury M, Biere A. Distributed cube and conquer with paracooba [C]//Theory

- and Applications of Satisfiability Testing–SAT 2020: 23rd International Conference, Alghero, Italy, July 3–10, 2020, Proceedings 23. Springer, 2020: 114-122.
- [150] Heule M J H, Kullmann O, Marek V W. Solving and verifying the boolean pythagorean triples problem via cube-and-conquer [C]//Creignou N, Berre D L. Lecture Notes in Computer Science: volume 9710 Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings. Springer, 2016: 228-245.
- [151] Le Frioux L, Baarir S, Sopena J, et al. Painless: a framework for parallel sat solving [C]// International Conference on Theory and Applications of Satisfiability Testing 2017. 2017: 233-250.
- [152] Balyo T, Sanders P, Sinz C. Hordesat: A massively parallel portfolio sat solver [C]// International Conference on Theory and Applications of Satisfiability Testing. 2015: 156-172.
- [153] Hamadi Y, Jabbour S, Sais L. Manysat: a parallel SAT solver [J]. *J. Satisf. Boolean Model. Comput.*, 2009, 6(4): 245-262.
- [154] Manthey N. The mergesat solver [C]//International Conference on Theory and Applications of Satisfiability Testing 2021. 2021: 387-398.
- [155] Tchinda R K, Djamegni C T. Hkis, hcad, pakis and painless exmaplecmdistchronobt in the sc21 [J]. Proceedings of SAT Competition 2021: 26.
- [156] Sonobe T, Inaba M. Portfolio with block branching for parallel sat solvers [C]//Learning and Intelligent OptimizatioN Conference 2013. 2013: 247-252.
- [157] Sonobe T, Kondoh S, Inaba M. Community branching for parallel portfolio sat solvers [C]// International Conference on Theory and Applications of Satisfiability Testing 2014. 2014: 188-196.
- [158] Audemard G, Simon L. Lazy clause exchange policy for parallel SAT solvers [C]//Sinz C, Egly U. Lecture Notes in Computer Science: volume 8561 Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings. Springer, 2014: 197-205.
- [159] Heule M, Manthey N, Philipp T. Validating unsatisfiability results of clause sharing parallel SAT solvers [C]//Berre D L. EPiC Series in Computing: volume 27 POS-14. Fifth Pragmatics of SAT workshop, a workshop of the SAT 2014 conference, part of FLoC 2014 during the Vienna Summer of Logic, July 13, 2014, Vienna, Austria. EasyChair, 2014: 12-25.
- [160] Hamadi Y, Jabbour S, Piette C, et al. Deterministic parallel DPLL [J]. *Journal on Satisfiability, Boolean Modeling and Computation*, 2011, 7(4): 127-132.

- [161] Nabeshima H, Inoue K. Reproducible efficient parallel SAT solving [C]//Pulina L, Seidl M. Lecture Notes in Computer Science: volume 12178 Theory and Applications of Satisfiability Testing - SAT 2020 - 23rd International Conference, Alghero, Italy, July 3-10, 2020, Proceedings. Springer, 2020: 123-138.
- [162] Xu L, Hoos H H, Leyton-Brown K. Hydra: Automatically configuring algorithms for portfolio-based selection [C]//Fox M, Poole D. Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010. AAAI Press, 2010.
- [163] Barrett C, Tinelli C. Satisfiability modulo theories [M]. Springer, 2018.
- [164] Sebastiani R. Lazy satisfiability modulo theories [J]. Journal on Satisfiability, Boolean Modeling and Computation, 2007, 3(3-4): 141-224.
- [165] Nieuwenhuis R, Oliveras A, Tinelli C. Solving sat and sat modulo theories: From an abstract davis–putnam–logemann–loveland procedure to dpll (t) [J]. Journal of the ACM (JACM), 2006, 53(6): 937-977.
- [166] Cimatti A, Griggio A, Schaafsma B J, et al. The mathsat5 smt solver [C]//International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2013: 93-107.
- [167] Armando A, Castellini C, Giunchiglia E. Sat-based procedures for temporal reasoning [C]// European Conference on Planning. Springer, 1999: 97-108.
- [168] Audemard G, Bertoli P, Cimatti A, et al. A sat based approach for solving formulas over boolean and linear mathematical propositions [C]//International Conference on Automated Deduction. Springer, 2002: 195-210.
- [169] Ganzinger H, Hagen G, Nieuwenhuis R, et al. Dpll (t): Fast decision procedures [C]// Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings 16. Springer, 2004: 175-188.
- [170] Nieuwenhuis R, Oliveras A, Tinelli C. Abstract dpll and abstract dpll modulo theories [C]// International Conference on Logic for Programming Artificial Intelligence and Reasoning. Springer, 2005: 36-50.
- [171] Nieuwenhuis R, Oliveras A. Dpll (t) with exhaustive theory propagation and its application to difference logic [C]//Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005. Proceedings 17. Springer, 2005: 321-334.
- [172] Bozzano M, Bruttomesso R, Cimatti A, et al. An incremental and layered procedure for the satisfiability of linear arithmetic logic [C]//Tools and Algorithms for the Construction and Analysis of Systems: 11th International Conference, TACAS 2005, Held as Part of the Joint

- European Conferences on Theory and Practice of Software, ETAPS 2005, Edinburgh, UK, April 4-8, 2005. Proceedings 11. Springer, 2005: 317-333.
- [173] Barrett C, Nieuwenhuis R, Oliveras A, et al. Splitting on demand in sat modulo theories [C]//Logic for Programming, Artificial Intelligence, and Reasoning: 13th International Conference, LPAR 2006, Phnom Penh, Cambodia, November 13-17, 2006. Proceedings 13. Springer, 2006: 512-526.
- [174] Katz G, Huang D A, Ibeling D, et al. The marabou framework for verification and analysis of deep neural networks [C]//Computer Aided Verification: 31st International Conference, CAV 2019, New York City, NY, USA, July 15-18, 2019, Proceedings, Part I 31. Springer, 2019: 443-452.
- [175] Schupp S, Ábrahám E, Rossmanith P, et al. Interval constraint propagation in smt compliant decision procedures [J]. Master's thesis, RWTH Aachen, 2013.
- [176] Dantzig G. Linear programming and extensions [M]. Princeton university press, 1963.
- [177] Weispfenning V. Quantifier elimination for real algebra—the quadratic case and beyond [J]. Applicable Algebra in Engineering, Communication and Computing, 1997, 8: 85-101.
- [178] Collins G E. Quantifier elimination for real closed fields by cylindrical algebraic decomposition: a synopsis [J]. ACM SIGSAM Bulletin, 1976, 10(1): 10-12.
- [179] Barrett C, Conway C L, Deters M, et al. Cvc4 [C]//Computer Aided Verification: 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings 23. Springer, 2011: 171-177.
- [180] Bouton T, Caminha B. de Oliveira D, Déharbe D, et al. verit: an open, trustable and efficient smt-solver [C]//International Conference on Automated Deduction. Springer, 2009: 151-156.
- [181] Kim H, Somenzi F, Jin H. Efficient term-ite conversion for satisfiability modulo theories [C]// Theory and Applications of Satisfiability Testing-SAT 2009: 12th International Conference, SAT 2009, Swansea, UK, June 30-July 3, 2009. Proceedings 12. Springer, 2009: 195-208.
- [182] Dutertre B, De Moura L. A fast linear-arithmetic solver for dpll (t) [C]//International Conference on Computer Aided Verification. Springer, 2006: 81-94.
- [183] Schrijver A. Theory of linear and integer programming [M]. John Wiley & Sons, 1998.
- [184] Williams H P. Fourier-motzkin elimination extension to integer programming problems [J]. Journal of combinatorial theory, series A, 1976, 21(1): 118-123.
- [185] Pugh W. The omega test: a fast and practical integer programming algorithm for dependence analysis [C]//Proceedings of the 1991 ACM/IEEE conference on Supercomputing. 1991: 4-13.
- [186] Davis M. Hilbert's tenth problem is unsolvable [J]. The American Mathematical Monthly, 1973, 80(3): 233-269.

- [187] Kremer G, Corzilius F, Ábrahám E. A generalised branch-and-bound approach and its application in sat modulo nonlinear integer arithmetic [C]//Computer Algebra in Scientific Computing: 18th International Workshop, CASC 2016, Bucharest, Romania, September 19-23, 2016, Proceedings 18. Springer, 2016: 315-335.
- [188] Borralleras C, Lucas S, Navarro-Marsé R, et al. Solving non-linear polynomial arithmetic via sat modulo linear arithmetic [C]//International Conference on Automated Deduction. Springer, 2009: 294-305.
- [189] Fränzle M, Herde C, Teige T, et al. Efficient solving of large non-linear arithmetic constraint systems with complex boolean structure [J]. Journal on Satisfiability, Boolean Modeling and Computation, 2006, 1(3-4): 209-236.
- [190] Tung V X, Van Khanh T, Ogawa M. rasat: an smt solver for polynomial constraints [J]. Formal Methods in System Design, 2017, 51: 462-499.
- [191] Bradley A R, Manna Z, Sipma H B. Linear ranking with reachability [C]//International Conference on Computer Aided Verification. Springer, 2005: 491-504.
- [192] Barrett C, Stump A, Tinelli C, et al. The smt-lib standard: Version 2.0 [C]//Proceedings of the 8th international workshop on satisfiability modulo theories (Edinburgh, UK): volume 13. 2010: 14.
- [193] Barrett C, Dill D, Levitt J. Validity checking for combinations of theories with equality [C]// Formal Methods in Computer-Aided Design: First International Conference, FMCAD'96 Palo Alto, CA, USA, November 6–8, 1996 Proceedings 1. Springer, 1996: 187-201.
- [194] Filliâtre J C, Owre S, Rue\* B H, et al. Ics: Integrated canonizer and solver? [C]//Computer Aided Verification: 13th International Conference, CAV 2001 Paris, France, July 18–22, 2001 Proceedings 13. Springer, 2001: 246-249.
- [195] Stump A, Barrett C W, Dill D L. Cvc: A cooperating validity checker [C]//Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27–31, 2002 Proceedings 14. Springer, 2002: 500-504.
- [196] Barrett C, Berezin S. Cvc lite: A new implementation of the cooperating validity checker: Category b [C]//Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings 16. Springer, 2004: 515-518.
- [197] Ganesh V. Decision procedures for bit-vectors, arrays and integers: volume 68 [M]. Citeseer, 2007.
- [198] Brummayer R, Biere A. Boolector: An efficient smt solver for bit-vectors and arrays [C]// Tools and Algorithms for the Construction and Analysis of Systems: 15th International Conference, TACAS 2009, Held as Part of the Joint European Conferences on Theory and Practice

- of Software, ETAPS 2009, York, UK, March 22-29, 2009. Proceedings 15. Springer, 2009: 174-177.
- [199] Niemetz A, Preiner M. Bitwuzla [C]//International Conference on Computer Aided Verification. Springer, 2023: 3-17.
- [200] Brummayer R, Biere A. Lemmas on demand for the extensional theory of arrays [C]// Proceedings of the Joint Workshops of the 6th International Workshop on Satisfiability Modulo Theories and 1st International Workshop on Bit-Precise Reasoning. 2008: 6-11.
- [201] Preiner M, Niemetz A, Biere A. Lemmas on demand for lambdas [J]. Program Proceedings, 2013: 28.
- [202] Fuhs C, Giesl J, Middeldorp A, et al. Sat solving for termination analysis with polynomial interpretations [C]//Theory and Applications of Satisfiability Testing–SAT 2007: 10th International Conference, Lisbon, Portugal, May 28-31, 2007. Proceedings 10. Springer, 2007: 340-354.
- [203] Corzilius F, Kremer G, Junges S, et al. Smt-rat: an open source c++ toolbox for strategic and parallel smt solving [C]//Theory and Applications of Satisfiability Testing–SAT 2015: 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings 18. Springer, 2015: 360-368.
- [204] Jovanović D, De Moura L. Solving non-linear arithmetic [J]. ACM Communications in Computer Algebra, 2013, 46(3/4): 104-105.
- [205] Jovanović D. Solving nonlinear integer arithmetic with mcsat [C]//Verification, Model Checking, and Abstract Interpretation: 18th International Conference, VMCAI 2017, Paris, France, January 15–17, 2017, Proceedings 18. Springer, 2017: 330-346.
- [206] Hader T, Kaufmann D, Irfan A, et al. Mcsat-based finite field reasoning in the yices2 smt solver [J]. arXiv preprint arXiv:2402.17927, 2024.
- [207] Cimatti A, Griggio A, Irfan A, et al. Invariant checking of nra transition systems via incremental reduction to lra with euf [C]//Tools and Algorithms for the Construction and Analysis of Systems: 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part I 23. Springer, 2017: 58-75.
- [208] Aniva L, Barbosa H, Barrett C, et al. Cvc5 at the smt competition 2023 [J].
- [209] Cimatti A, Griggio A, Irfan A, et al. Experimenting on solving nonlinear integer arithmetic with incremental linearization [C]//International Conference on Theory and Applications of Satisfiability Testing. Springer, 2018: 383-398.
- [210] Reynolds A, Tinelli C, Jovanović D, et al. Designing theory solvers with extensions [C]//

- Frontiers of Combining Systems: 11th International Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings 11. Springer, 2017: 22-40.
- [211] Borralleras C, Lucas S, Oliveras A, et al. Sat modulo linear arithmetic for solving polynomial constraints [J]. *Journal of Automated Reasoning*, 2012, 48(1): 107-131.
  - [212] Borralleras C, Larraz D, Rodríguez-Carbonell E, et al. Incomplete smt techniques for solving non-linear formulas over the integers [J]. *ACM Transactions on Computational Logic (TOCL)*, 2019, 20(4): 1-36.
  - [213] Griggio A, Phan Q S, Sebastiani R, et al. Stochastic local search for smt: combining theory solvers with walksat [C]//International Symposium on Frontiers of Combining Systems. Springer, 2011: 163-178.
  - [214] Fröhlich A, Biere A, Wintersteiger C, et al. Stochastic local search for satisfiability modulo theories [C]//Proceedings of the AAAI Conference on Artificial Intelligence: volume 29. 2015.
  - [215] Niemetz A, Preiner M, Biere A, et al. Improving local search for bit-vector logics in smt with path propagation [C]//Proceedings of the Fourth International Workshop on Design and Implementation of Formal Tools and Systems, Austin, TX, USA. 2015: 1-10.
  - [216] Niemetz A, Preiner M, Biere A. Precise and complete propagation based local search for satisfiability modulo theories [C]//International Conference on Computer Aided Verification. Springer, 2016: 199-217.
  - [217] Cai S, Li B, Zhang X. Local search for satisfiability modulo integer arithmetic theories [J]. *ACM Transactions on Computational Logic*, 2022.
  - [218] Gaschnig J. A general backtrack algorithm that eliminates most redundant tests. [C]//IJCAI. 1977: 457.
  - [219] Stallman R M, Sussman G J. Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis [J]. *Artificial intelligence*, 1977, 9(2): 135-196.
  - [220] Marques-Silva J. Search algorithms for satisfiability problems in combinational switching circuits [D]. University of Michigan, 1995.
  - [221] Marques-Silva J P, Sakallah K A. Grasp: A search algorithm for propositional satisfiability [J]. *IEEE Transactions on Computers*, 1999, 48(5): 506-521.
  - [222] Gu J. Efficient local search for very large-scale satisfiability problems [J]. *ACM SIGART Bulletin*, 1992, 3(1): 8-12.
  - [223] Gomes C P, Selman B, Kautz H, et al. Boosting combinatorial search through randomization [J]. *Proceedings of AAAI Conference on Artificial Intelligence* 1998, 1998, 98: 431-437.
  - [224] 罗茂. 基于子句化简的 SAT 完备性算法研究 [D]. 华中科技大学, 2022.
  - [225] 肖凡. SAT 及其相关问题的精确算法研究 [D]. 华中科技大学, 2019.

- [226] Nadel A, Ryvchin V. Chronological backtracking [C]//Proceedings of SAT 2018. 2018: 111-121.
- [227] Cherif M S, Habet D, Terrioux C. Combining vsids and chb using restarts in sat [C]//International Conference on Principles and Practice of Constraint Programming 2021. 2021: 1-19.
- [228] Zheng J, He K, Chen Z, et al. Combining hybrid walking strategy with kissat mab, radical, and lstechn-maple [J]. Proceedings of SAT Competition 2022, 2022: 20.
- [229] Haberlandt A, Green H, Heule M J. Effective auxiliary variables via structured reencoding [J]. arXiv preprint arXiv:2307.01904, 2023.
- [230] Habet D, Li C M, Devendeville L, et al. A hybrid approach for SAT [C]//Proceedings of CP 2002. 2002: 172-184.
- [231] Letombe F, Marques-Silva J. Improvements to hybrid incremental SAT algorithms [C]//Proceedings of SAT 2008. 2008: 168-181.
- [232] Balint A, Henn M, Gableske O. A novel approach to combine a SLS- and a dpll-solver for the satisfiability problem [C]//Proceedings of SAT 2009. 2009: 284-297.
- [233] Selsam D, Lamm M, Bünz B, et al. Learning a sat solver from single-bit supervision [J]. arXiv preprint arXiv:1802.03685, 2018.
- [234] Han J M. Enhancing sat solvers with glue variable predictions [J]. arXiv preprint arXiv:2007.02559, 2020.
- [235] Hamadi Y, Sais L. Handbook of parallel constraint reasoning [M]. Springer, 2018.
- [236] Knuth D E. The art of computer programming, volume 4, fascicle 6: Satisfiability [M]. Addison-Wesley Professional, 2015.
- [237] Gomes C P, Selman B, Crato N, et al. Heavy-tailed phenomena in satisfiability and constraint satisfaction problems [J]. Journal of Automated Reasoning, 2000, 24(1/2): 67-100.
- [238] Gomes C P, Sabharwal A. Exploiting runtime variation in complete solvers. [J]. Handbook of Satisfiability, 2009, 185: 271-288.
- [239] Liang J H, Oh C, Mathew M, et al. Machine learning-based restart policy for CDCL SAT solvers [C]//SAT 2018. 2018: 94-110.
- [240] Biere A, Fröhlich A. Evaluating cdcl restart schemes [J]. Pragmatics of SAT, 2015: 1-17.
- [241] Marques-Silva J, Lynce I, Malik S. Conflict-driven clause learning sat solvers [M]//Handbook of satisfiability. 2021: 133-182.
- [242] Biere A. Picosat essentials [J]. Journal on Satisfiability, Boolean Modeling and Computation, 2008, 4(2-4): 75-97.
- [243] Biere A. Adaptive restart strategies for conflict driven SAT solvers [C]//Büning H K, Zhao X. Proceedings of SAT 2008. 2008: 28-33.

- [244] Biere A, Fleury M. Chasing target phases [C]//Workshop on the Pragmatics of SAT. 2020.
- [245] Baptista L, Marques-Silva J P. Using randomization and learning to solve hard real-world instances of satisfiability [C]//International Conference on Principles and Practice of Constraint Programming 2000: volume 1894. 2000: 489-494.
- [246] Ramos A, van der Tak P, Heule M. Between restarts and backjumps [C]//International Conference on Theory and Applications of Satisfiability Testing 2011: volume 6695. 2011: 216-229.
- [247] Kochemazov S, Zaikin O, Semenov A, et al. Speeding up cdcl inference with duplicate learnt clauses [M]//ECAI 2020. IOS Press, 2020: 339-346.
- [248] Manthey N. Radical modification-watch sat [J]. Proceedings of SAT Competition 2021, 2021: 28.
- [249] Selman B, Kautz H A, McAllester D A. Ten challenges in propositional reasoning and search [C]//Proceedings of IJCAI 97. 1997: 50-54.
- [250] Audemard G, Lagniez J, Mazure B, et al. Integrating conflict driven clause learning to local search [C]//Proceedings of LSCS 2009. 2009: 55-68.
- [251] Cha B, Iwama K. Adding new clauses for faster local search [C]//Proceedings of AAAI 96. 1996: 332-337.
- [252] Anbulagan, Pham D N, Slaney J K, et al. Old resolution meets modern SLS [C]//Proceedings of AAAI 2005. 2005: 354-359.
- [253] Lorenz J, Wörz F. On the effect of learned clauses on stochastic local search [C]//Proceedings of SAT 2020. 2020: 89-106.
- [254] Balint A, Manthey N. SparrowToRiss 2018 [C]//Proceedings of SAT Competition 2018: Solver and Benchmark Descriptions. 2018: 38-39.
- [255] Cai S, Luo C, Su K. CCAnr+glucose in SAT Competition 2014 [C]//Proc. of SAT Competition 2014: Solver and Benchmark Descriptions. 2014: 17.
- [256] Li C M, Habet D. Description of RSeq2014 [C]//Proc. of SAT Competition 2014: Solver and Benchmark Descriptions. 2014: 72.
- [257] Kroc L, Sabharwal A, Gomes C P, et al. Integrating systematic and local search paradigms: A new strategy for maxsat [C]//Proceedings of IJCAI 2009. 2009: 544-551.
- [258] Gershman R, Strichman O. Haifusat: A new robust SAT solver [C]//Ur S, Bin E, Wolfsthal Y. Proceedings of Haifa Verification Conference 2005. 2005: 76-89.
- [259] Biere A. P{re,i}cosat@sc'09 [C]//SAT 2009 Competitive Event Booklet. 2009: 42-43.
- [260] Fleury A B M. Gimsatul, isasat, kissat [J]. Proceedings of SAT competition 2022: 10.
- [261] Kochemazov S, Zaikin O, Kondratiev V, et al. Maplelcmdistchronobt-dl, duplicate learnnts heuristic-aided solvers at the sat race 2019 [J]. Proceedings of SAT Race, 2019: 24-24.

- [262] Christ J, Hoenicke J, Nutz A. Smtinterpol: An interpolating smt solver [C]//International SPIN Workshop on Model Checking of Software. Springer, 2012: 248-254.
- [263] Bryant R E, Kroening D, Ouaknine J, et al. Deciding bit-vector arithmetic with abstraction [C]//Tools and Algorithms for the Construction and Analysis of Systems: 13th International Conference, TACAS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007 Braga, Portugal, March 24-April 1, 2007. Proceedings 13. Springer, 2007: 358-372.
- [264] Talupur M, Sinha N, Strichman O, et al. Range allocation for separation logic [C]//Computer Aided Verification: 16th International Conference, CAV 2004, Boston, MA, USA, July 13-17, 2004. Proceedings 16. Springer, 2004: 148-161.
- [265] Strichman O, Seshia S A, Bryant R E. Deciding separation formulas with sat [C]//Computer Aided Verification: 14th International Conference, CAV 2002 Copenhagen, Denmark, July 27-31, 2002 Proceedings 14. Springer, 2002: 209-222.
- [266] Seshia S A, Lahiri S K, Bryant R E. A hybrid sat-based decision procedure for separation logic with uninterpreted functions [C]//Proceedings of the 40th annual Design Automation Conference. 2003: 425-430.
- [267] Hadarean L, Bansal K, Jovanović D, et al. A tale of two solvers: Eager and lazy approaches to bit-vectors [C]//International Conference on Computer Aided Verification. Springer, 2014: 680-695.
- [268] Niemetz A, Preiner M. Bitwuzla at the smt-comp 2020 [J]. arXiv preprint arXiv:2006.01621, 2020.
- [269] Jia F, Dong Y, Liu M, et al. Suggesting variable order for cylindrical algebraic decomposition via reinforcement learning [C]//Thirty-seventh Conference on Neural Information Processing Systems. 2023.
- [270] Soos M. The cryptominisat 5 set of solvers at sat competition 2016 [J]. Proceedings of SAT Competition, 2016: 28.
- [271] Barrett C, De Moura L, Stump A. Smt-comp: Satisfiability modulo theories competition [C]//Computer Aided Verification: 17th International Conference, CAV 2005, Edinburgh, Scotland, UK, July 6-10, 2005. Proceedings 17. Springer, 2005: 20-23.
- [272] Borralleras C, Larraz D, Oliveras A, et al. Verymax: tool description for termcomp 2016 [C]//15th International Workshop on Termination. 2016: 18.
- [273] Leike J, Heizmann M. Geometric nontermination arguments [C]//International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2018: 266-283.
- [274] Beyer D. Software verification and verifiable witnesses: (report on sv-comp 2015) [C]//Tools

- and Algorithms for the Construction and Analysis of Systems: 21st International Conference, TACAS 2015, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2015, London, UK, April 11-18, 2015, Proceedings 21. Springer, 2015: 401-416.
- [275] Ponce-de León H, Haas T, Meyer R. Dartagnan: Smt-based violation witness validation (competition contribution) [C]//International Conference on Tools and Algorithms for the Construction and Analysis of Systems. Springer, 2022: 418-423.
- [276] Bofill M, Nieuwenhuis R, Oliveras A, et al. The barcelogic smt solver: tool paper [C]// International Conference on Computer Aided Verification. Springer, 2008: 294-298.
- [277] Demšar J. Statistical comparisons of classifiers over multiple data sets [J]. The Journal of Machine learning research, 2006, 7: 1-30.
- [278] Friedman M. The use of ranks to avoid the assumption of normality implicit in the analysis of variance [J]. Journal of the American Statistical Association, 1937, 32(200): 675-701.
- [279] Amarú L, Marranghello F, Testa E, et al. Sat-sweeping enhanced for logic synthesis [C]// 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020: 1-6.
- [280] Mishchenko A, Case M, Brayton R, et al. Scalable and scalably-verifiable sequential synthesis [C]//2008 IEEE/ACM International Conference on Computer-Aided Design. IEEE, 2008: 234-241.
- [281] Zhang J S, Mishchenko A, Brayton R, et al. Symmetry detection for large boolean functions using circuit representation, simulation, and satisfiability [C]//Proceedings of the 43rd annual Design Automation Conference. 2006: 510-515.
- [282] Jarratt T, Eckert C M, Caldwell N H, et al. Engineering change: an overview and perspective on the literature [J]. Research in engineering design, 2011, 22: 103-124.
- [283] Nowka K J, Galambos T. Circuit design techniques for a gigahertz integer microprocessor [C]//Proceedings International Conference on Computer Design. VLSI in Computers and Processors (Cat. No. 98CB36273). IEEE, 1998: 11-16.
- [284] Zhu Q, Kitchen N, Kuehlmann A, et al. Sat sweeping with local observability don't-cares [C]//Proceedings of the 43rd Annual Design Automation Conference. 2006: 229-234.
- [285] Mishchenko A, Chatterjee S, Brayton R, et al. Improvements to combinational equivalence checking [C]//Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design. 2006: 836-843.
- [286] Mishchenko A, et al. Abc: A system for sequential synthesis and verification [J]. URL <http://www.eecs.berkeley.edu/alanmi/abc>, 2007, 17.
- [287] Wu S C, Wang C Y, Hsieh J A. The potential and limitation of probability-based combinational equivalence checking [C]//2006 15th Asian Test Symposium. IEEE, 2006: 103-108.

- [288] Chowdhary A, Kale S, Saripella P K, et al. Extraction of functional regularity in datapath circuits [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 1999, 18(9): 1279-1296.
- [289] Lv J, Kalla P. Formal verification of galois field multipliers using computer algebra techniques [C]//2012 25th International Conference on VLSI Design. IEEE, 2012: 388-393.
- [290] Brand D. Verification of large synthesized designs [C]//Proceedings of 1993 International Conference on Computer Aided Design (ICCAD). IEEE, 1993: 534-537.
- [291] Lai Y, Meel K S, Yap R H. The power of literal equivalence in model counting [C]// Proceedings of the AAAI Conference on Artificial Intelligence: volume 35. 2021: 3851-3859.
- [292] Kuehlmann A, Paruthi V, Krohm F, et al. Robust boolean reasoning for equivalence checking and functional property verification [J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2002, 21(12): 1377-1394.
- [293] Bjesse P, Boralv A. Dag-aware circuit compression for formal verification [C]//IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004. IEEE, 2004: 42-49.
- [294] Mishchenko A, Chatterjee S, Jiang R, et al. Fraigs: A unifying representation for logic synthesis and verification [R]. ERL Technical Report, 2005.
- [295] Kuehlmann A. Dynamic transition relation simplification for bounded property checking [C]//IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004. IEEE, 2004: 50-57.
- [296] Agrawal V D, Lee D. Characteristic polynomial method for verification and test of combinational circuits [C]//Proceedings of 9th International Conference on VLSI Design. IEEE, 1996: 341-342.
- [297] Dudek J M, Meel K S, Vardi M Y. The hard problems are almost everywhere for random cnf-xor formulas [J]. arXiv preprint arXiv:1710.06378, 2017.
- [298] Gwynne M, Kullmann O. On sat representations of xor constraints [C]//Language and Automata Theory and Applications: 8th International Conference, LATA 2014, Madrid, Spain, March 10-14, 2014. Proceedings 8. Springer, 2014: 409-420.
- [299] Siahaan A P U. An overview of the rc4 algorithm [J]. 2017.
- [300] Anderson D P, Herrtwich R G. Internet communication with end-to-end performance guarantees [C]//Telekommunikation und multimediale Anwendungen der Informatik: GI-21. Jahrestagung, Darmstadt, 14.-18. Oktober 1991. Springer, 1991: 246-258.
- [301] Bernstein D J, et al. Chacha, a variant of salsa20 [C]//Workshop record of SASC: volume 8. Citeseer, 2008: 3-5.

- [302] De Canniere C, Preneel B. Trivium [M]//New Stream Cipher Designs: The eSTREAM Finalists. Springer, 2008: 244-266.
- [303] Ekdahl P, Johansson T. Snow-a new stream cipher [C]//Proceedings of first open NESSIE workshop, KU-Leuven. 2000: 167-168.
- [304] Dobraunig C, Eichlseder M, Mendel F, et al. Ascon v1. 2: Lightweight authenticated encryption and hashing [J]. Journal of Cryptology, 2021, 34: 1-42.
- [305] Hell M, Johansson T, Meier W, et al. An aead variant of the grain stream cipher [C]// International Conference on Codes, Cryptology, and Information Security. Springer, 2019: 55-71.
- [306] Yang Y, Zhao W, Xiong L, et al. Optimized implementations for zuc-256 on fpga [J]. Wireless Personal Communications, 2021, 116(3): 2615-2632.
- [307] 周照存. 基于相关性的流密码分析方法研究 [D]. 中国科学院软件研究所, 2023.
- [308] Coppersmith D, Halevi S, Jutla C. Cryptanalysis of stream ciphers with linear masking [C]// Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22. Springer, 2002: 515-532.
- [309] Siegenthaler. Decrypting a class of stream ciphers using ciphertext only [J]. IEEE Transactions on computers, 1985, 100(1): 81-85.
- [310] Meier W, Staffelbach O. Fast correlation attacks on certain stream ciphers [J]. Journal of Cryptology, 1989, 1: 159-176.
- [311] Chepyzhov V V, Johansson T, Smeets B. A simple algorithm for fast correlation attacks on stream ciphers [C]//Fast Software Encryption: 7th International Workshop, FSE 2000 New York, NY, USA, April 10–12, 2000 Proceedings 7. Springer, 2001: 181-195.
- [312] Biham E, Shamir A. Differential cryptanalysis of des-like cryptosystems [J]. Journal of Cryptology, 1991, 4: 3-72.
- [313] Biryukov A, Wagner D. Slide attacks [C]//International Workshop on Fast Software Encryption. Springer, 1999: 245-259.
- [314] Stevens M. Fast collision attack on md5 [J]. Cryptology EPrint Archive, 2006.
- [315] Zhang B, Li Z, Feng D, et al. Near collision attack on the grain v1 stream cipher [C]// International Workshop on Fast Software Encryption. Springer, 2013: 518-538.
- [316] Dinur I, Shamir A. Cube attacks on tweakable black box polynomials [C]//Advances in Cryptology-EUROCRYPT 2009: 28th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cologne, Germany, April 26-30, 2009. Proceedings 28. Springer, 2009: 278-299.

- [317] Sun Y. Automatic search of cubes for attacking stream ciphers [J]. IACR Transactions on Symmetric Cryptology, 2021: 100-123.
- [318] Li Z, Bi W, Dong X, et al. Improved conditional cube attacks on keccak keyed modes with milp method [C]//Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part I 23. Springer, 2017: 99-127.
- [319] Hawkes P, Rose G G. Guess-and-determine attacks on snow [C]//Selected Areas in Cryptography: 9th Annual International Workshop, SAC 2002 St. John's, Newfoundland, Canada, August 15–16, 2002 Revised Papers 9. Springer, 2003: 37-46.
- [320] Jiao L, Li Y, Hao Y. A guess-and-determine attack on snow-v stream cipher [J]. The Computer Journal, 2020, 63(12): 1789-1812.
- [321] Cao C, Cen Z, Feng X, et al. Straightforward guess and determine analysis based on genetic algorithm [J]. Journal of Systems Science and Complexity, 2022, 35(5): 1988-2003.
- [322] Fu K, Wang M, Guo Y, et al. Milp-based automatic search algorithms for differential and linear trails for speck [C]//Fast Software Encryption: 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers 23. Springer, 2016: 268-288.
- [323] Semenov A, Zaikin O, Bespalov D, et al. Sat-approach for cryptoanalysis of some stream ciphering systems [J]. Vychisl. Tekhnol, 2008, 13(6): 134-150.
- [324] Shi Z, Jin C, Zhang J, et al. A correlation attack on full snow-v and snow-vi [C]//Annual International Conference on the Theory and Applications of Cryptographic Techniques. Springer, 2022: 34-56.
- [325] Cid C, Weinmann R P. Block ciphers: algebraic cryptanalysis and groebner bases [M]// Groebner bases, coding, and cryptography. Springer, 2009: 307-327.
- [326] Funabiki Y, Todo Y, Isobe T, et al. Several milp-aided attacks against snow 2.0 [C]// International Conference on Cryptology and Network Security. Springer, 2018: 394-413.
- [327] Meier W, Pasalic E, Carlet C. Algebraic attacks and decomposition of boolean functions [C]//Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23. Springer, 2004: 474-491.
- [328] Hellman M. A cryptanalytic time-memory trade-off [J]. IEEE Transactions on Information Theory, 1980, 26(4): 401-406.
- [329] Biere A, Cimatti A, Clarke E M, et al. Bounded model checking. [J]. Handbook of Satisfiability, 2009, 185(99): 457-481.



## 致 谢

时光荏苒，六年的博士生涯已经到了尾声。接下来，我将继续从事我热爱的学术事业。这篇论文是我多年研究的成果，也是我对所有帮助过我的人的致敬。在此，我衷心感谢所有在我博士学习期间给予我支持和帮助的人，谢谢你们！

首先，我要向我的导师蔡少伟研究员表示最深的感谢！我与蔡老师第一次见面是在 2017 年，当时我还是大学三年级，有对科研的憧憬，但也有对未来的迷惘。正是与蔡老师的交流，让我坚定的踏上了学术研究的道路。在相处的 7 年多时光里，蔡老师不仅在科研上悉心指导，在生活和心理上对我提供了莫大的照顾和帮助。他的严谨治学，深厚的专业素养，悉心指导，使我在学术道路上不断成长；他的言传身教，使我深刻理解了科研的真谛，并激励我以更高的标准要求自己；他的耐心和热情，使我在面对困难和挫折时，都能找到前进的动力！

感谢罗川老师、林锦坤老师、李勇坚老师。当我遇到科研和学术上的难题时，他们总是无私的答疑解惑。感谢国科大和软件所提供的优秀学术环境和丰富资源，使我能够顺利完成我的研究工作。感谢实验室主任陈海波老师、副主任王文成老师、詹乃军老师，感谢吴鹏老师在奖学金申请等方面的工作，感谢孙守云老师、张鑫老师、张丽老师、费腾老师尽心地管理实验室的各项事务，为学生的科研工作保驾护航。感谢研究生部李彩丽老师、张欢老师、肖祎晨老师、杜慧雯老师、唐诗豪老师在学生工作上的付出。感谢党支部薛白老师等人用心的组织各种学习和参观活动。感谢所有参与我论文评审和答辩的老师们，你们的宝贵意见使我的研究工作更加完善。

感谢师兄师姐、师弟师妹们，让我的博士生活变得丰富多彩。感谢李元杰、雷震东、王浩然、傅英杰、何兵、侯雯颖、李博涵、陈志翰、贺翔、姜滔、赵梦宇、李延、马锟、朱凌峰、钱宇航、林鹏、李瑾媛、陈少皇、张聪懿、初一、叶馥榕、陶悦、邹蒙川、胡皓、吴豪的帮助。

感谢我的发小们，感谢朱鹏飞、孙康宁、贾云珂、曾帅、孙明哲、韩凤平、杨明格，我们相识二十余年，感谢人生路上有你们的陪伴，感谢当我踌躇不前时为我排忧解难、排空我的所有坏心情。

感谢我的父母张运毅先生与题露晓女士，感谢你们给予我生命，带我认识这

个世界，为我漫漫求学路遮风挡雨。更是他们无私的爱与支持，让我能够坚定地走完这段博士求学之路。在我成长和求学的每一个阶段，父母都给予了我最坚定的支持和鼓励。父母的爱如同灯塔，指引我不断前进，不断追求卓越。感谢干妈徐珺女士对本人及爱人的照顾。感谢家人朋友们，是你们的无私支持和深深理解，让我能够全心投入到博士学习中。你们的爱是我走过这段艰难路程的最大动力。

在此，我要特别感谢我的妻子牟林女士，感谢她十年来的陪伴和照顾，感谢她陪我走过了所有离乡求学的日子。她始终如一地支持我，鼓励我，给予我无尽的力量。我衷心感谢她的理解、耐心和付出，并愿以此论文作为我们共同努力和坚持的见证！

最后，我要感谢这段博士生涯中的每一次失败和成功，它们都让我更加深刻地理解了科研的本质，也使我更加坚定了自己的学术追求。这段博士生涯，是我人生中宝贵的财富。在未来的学术道路上，我将带着所有人的期望与厚爱，继续努力，勇往直前！

致以最深的敬意！

## 作者简历及攻读学位期间发表的学术论文与研究成果

### 基本信息：

姓名：张昕荻

性别：男

出生日期：1996 年 12 月 15 日

籍贯：山东省聊城市

联系方式：zhangxd@ios.ac.cn

### 教育经历：

2014.09–2018.06, 吉林大学, 软件学院, 工学学士; 经济学院, 金融学学士。

2018.09–2024.06, 中国科学院软件研究所, 基础软件与系统重点实验室（前计算机科学国家重点实验室）, 工学博士。

### 已发表的学术论文（时间排序）：

1. Shaowei Cai, and **Xindi Zhang**. “Pure MaxSAT and Its Applications to Combinatorial Optimization via Linear Local Search”. In: *Principles and Practice of Constraint Programming: 26th International Conference, CP 2020* (CCF-B 会议), 学生一作
2. Bohan Li, **Xindi Zhang**, Shaowei Cai, Jinkun Lin, Yiyuan Wang, and Christian Blum. “NuCDS: An Efficient Local Search Algorithm for Minimum Connected Dominating Set”. In: *Proceedings of the 29th International Conference on International Joint Conferences on Artificial Intelligence, IJCAI 2020* (CCF-A 会议), 第二作者
3. **Xindi Zhang**, Bohan Li, Shaowei Cai, and Yiyuan Wang. “Efficient local search based on dynamic connectivity maintenance for minimum connected dominating set”. *Journal of Artificial Intelligence Research* 71 (2021): 89-119. **JAIR** (CCF-B 期刊), 2021, 第一作者
4. Shaowei Cai, Chuan Luo, **Xindi Zhang**, and Jian Zhang. “Improving local search

- for structured SAT formulas via unit propagation based construct and cut initialization (short paper)”. In: *27th International Conference on Principles and Practice of Constraint Programming, CP 2021* (CCF-B 会议), 第三作者
5. Shaowei Cai, and **Xindi Zhang**. “Deep Cooperation of CDCL and Local Search for SAT”. In: *Theory and Applications of Satisfiability Testing: 24th International Conference, SAT 2021* (CCF-B 会议), 共同一作 (最佳论文奖)
  6. Shaowei Cai, Bohan Li, and **Xindi Zhang**. “Local Search for SMT on Linear Integer Arithmetic”. In: *International Conference on Computer Aided Verification, CAV 2022* (CCF-A 会议), 共同一作
  7. Shaowei Cai, **Xindi Zhang**, Mathias Fleury, and Armin Biere. “Better decision heuristics in CDCL through local search and target phases”. *Journal of Artificial Intelligence Research* 74 (2022): 1515-1563, **JAIR** (CCF-B 期刊), 2022, 学生一作
  8. Shaowei Cai, and **Xindi Zhang**. “Deep cooperation of CDCL and local search for SAT (Extended Abstract)”. In: *Proceedings of the 31th International Conference on International Joint Conferences on Artificial Intelligence, IJCAI 2022* (CCF-A 会议), 学生一作
  9. Shaowei Cai, Bohan Li, and **Xindi Zhang**. “Local Search For Satisfiability Modulo Integer Arithmetic Theories”, *ACM Transactions on Computational Logic* 24.4 (2023): 1-26, **ToCL** (CCF-B 期刊), 共同一作
  10. Zhihan Chen, **Xindi Zhang**, Yuhang Qian, and Shaowei Cai. “Integrating Exact Simulation into Sweeping for Datapath Combinational Equivalence Checking”. In: *2023 IEEE/ACM International Conference on Computer Aided Design, IC-CAD 2023* (CCF-B 会议), 第二作者
  11. **Xindi Zhang**, Bohan Li, and Shaowei Cai. “Deep Combination of CDCL(T) and Local Search for Satisfiability Modulo Non-Linear Integer Arithmetic Theory”. In: *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE 2024* (CCF-A 会议), 第一作者
  12. Zhiteng Chao, **Xindi Zhang**, Junying Huang, Jing Ye, Shaowei Cai, Huawei Li, Xiaowei Li. “A Fast Test Compaction Method for Commercial DFT Flow Us-

ing Dedicated Pure-MaxSAT Solver”, In: *29th Asia and South Pacific Design Automation Conference, ASP-DAC 2024* (CCF-C 会议), 第二作者

#### 参加的研究项目情况:

1. “可满足性问题求解”, 国家自然科学基金优秀青年科学基金项目, 2022-2024
2. “面向 RISCV 的先进 EDA 技术”, 中科院战略先导 A 专项课题, 2022-2025
3. “信息安全关键领域核心算法研究及算法库构建”, 国家重点研发计划, 2023-2028

#### 申请或已获得的专利:

1. 基于增量 SMT 问题求解的软件断言验证方法、系统及设备
2. 一种基于过程中分治的布尔可满足性问题并行求解方法
3. 一种基于属性指导可达性的增强限界模型检查系统
4. 基于布尔差分的乘法复杂度优化方法及装置
5. 一种基于完备仿真的组合运算电路等价性验证方法及系统

#### 学科竞赛获奖:

1. 2018 年, 国际 SAT 协会, 国际 SAT 竞赛, NoLimit-Track 冠军
2. 2018 年, LIACS(Leiden University), Sparkle SAT Challenge, 亚军
3. 2020 年, 国际 SAT 协会, 国际 SAT 竞赛, Main-Track-SAT 冠军
4. 2021 年, 国际 SAT 协会, 国际 SAT 竞赛, Main-Track-SAT/UNSAT 亚军
5. 2021 年, 国际 SAT 协会, 国际 SMT 竞赛, QF-IDL 冠军 (国内首次获得 SMT 比赛冠军)
6. 2022 年, 国际 SAT 协会, 国际 SAT 比赛, Main-Parallel-Track 冠军 \*2 (国内首次获得 SAT 比赛并行主赛道冠军); NoLimits-Track 冠军
7. 2022 年, 国际逻辑学联合会议, 奥林匹克竞赛 金牌 \*3
8. 2022 年, 集成电路 EDA 设计精英挑战赛, 海思企业特别奖; 二等奖
9. 2023 年, 国际 SAT 协会, 国际 SAT 比赛, Main-Parallel-Track 冠军 \*3; Cloud-Track 亚军

10. 2023 年, 国际 SAT 协会, 国际 SMT 比赛, 模型验证“最大领先奖”与“最大贡献奖”金牌
11. 2023 年, 中央网信办、河南人民政府、信息工程大学等, “强网杯”全国网络安全挑战赛密码数学专项赛, 冠军

#### 荣誉与奖励:

1. 2020 年, 中国科学院大学, 硕士国家奖学金
2. 2022 年, 中国科学院大学, 三好学生标兵
3. 2022 年, 中国科学院大学, 博士国家奖学金
4. 2023 年, 中国教育发展基金会, 奋进奖学金-集成电路人才培养
5. 2023 年, 中国科学院大学, 朱李月华优秀博士奖
6. 2023 年, 中国软件大会 (CCF ChinaSoft), 优秀博士生