分类号：　　密级：公开

UDC　　学号：2020023056

华南师範大學

South China Normal University

硕士学位论文

（专业学位）

PMS 的局部搜索算法改进策略研究

学 位 申 请 人：于瀚一

专 业 学 位 名 称：工程硕士

专 业 学 位 领 域：计算机技术

所 在 院 系：计算机学院

导师姓名及职称：陈寅　副教授

2023 年 5 月 25 日

PMS 的局部搜索算法改进策略研究

专业名称：计算机技术
申请者：于瀚一
导师：陈寅 副教授

摘 要

部分最大可满足性问题 (Partial MaxSAT, PMS) 及其加权版本 (Weighted Partial MaxSAT, WPMS) 是可满足性问题 (Boolean Satisfiability Problem, SAT) 的重要变体。它们将问题实例的约束分为两类：硬约束和软约束，并要求所有硬约束必须被满足，同时满足尽可能多的软约束。(W)PMS 可以对广泛的现实问题进行建模，如时间表、带取送货的车辆选路和无线传感器优化等问题都可以编码为 (W)PMS 进行求解。(W)PMS 具有非多项式完全 (NPC) 的难度，因此，如何在合理的时间内求解 (W)PMS 成为亟需解决的难题。

完备算法与不完备算法是求解 (W)PMS 的主流方法。完备算法保证问题的最优解，但求解时间无法确定；不完备算法通常能在规定时间内返回一组高质量解，越来越得到研究者关注。随机局部搜索求解器 (Stochastic Local Search, SLS) 是不完备算法的一种实现，通常包括初始解生成、变量选择启发式和权重机制等三个重要组件。从近些年来看，SLS 的研究大多针对搜索深度和广度方面改进，对初始解生成算法方面的研究甚少。

本文梳理 SLS 的关键技术发展，探讨各个改进策略的目的和实现方式，挖掘当下最先进的 SLS: SATLike3.0 的改进方向。首先提出了一种优先传播未满足硬子句中未赋值变量的方法，使得该方法更擅长处理 WPMS 实例。在上述工作的基础上，本文提出了基于分析硬子句矛盾动态配置子句初始权重的方法，得到的算法可以解决更多的 PMS 实例。随后将两个方法的思想进行了融合，提出了新颖的

权重初始化机制，全面提升了 SATLike3.0 处理 (W)PMS 实例的综合性能。本文通过实验的方式验证改进方法的有效性，并最终将得到的方法应用于求解组合拍卖、菜品规划和 FPGA 布局优化问题，取得了良好的效果。

关键词：随机局部搜索; 部分最大可满足性问题; 初始解; 权重机制; SATLike3.0

Research on Improvement Strategies for Local Search Algorithms in PMS

Major: Computer Technology

Name: HanYi Yu

Supervisor: Yin Chen

ABSTRACT

The Partial MaxSAT (PMS) problem and its weighted version (Weighted Partial MaxSAT, WPMS) is an important variant of the Boolean Satisfiability Problem (Boolean Satisfiability Problem, SAT). They divide the constraints of a problem instance into two categories: hard and soft constraints, and requires that all hard constraints must be satisfied, while satisfying as many soft constraints as possible. (W)PMS can model a wide range of real-world problems, such as timetabling, vehicle routing with pickup and delivery, and wireless sensor optimization, which can be coded as (W)PMS for solution. The (W)PMS is non-polynomially complete (NPC) in difficulty, therefore, how to solve (W)PMS in a reasonable time has become a difficult problem that needs to be solved urgently.

Complete and incomplete algorithms are the dominant methods for solving (W)PMS. A complete algorithm guarantees an optimal solution to the problem, but the solution time cannot be determined; an incomplete algorithm usually returns a set of high quality solutions in a specified time and is gaining increasing attention from researchers. Stochastic Local Search Solver (SLS) is an implementation of an incomplete algorithm, which usually includes three important components such as initial solution generation, variable

selection heuristic and weighting mechanism. In recent years, most of the research on SLS has focused on improving the search depth and breadth, and little research has been done on the initial solution generation algorithm.

This thesis reviews the key technical developments in SLS, explores the purpose and implementation of each improvement strategy, and explores the current state-of-the-art SLS: SATLike 3.0. Firstly, a method to preferentially propagate unassigned variables in unsatisfied hard clauses is proposed, making the method more adept at handling WPMS instances. Based on the above work, this thesis presents a method based on analysing hard clause contradictions to dynamically configure the initial weights of clauses, resulting in an algorithm that can solve more PMS instances. We merge the ideas of the two methods and propose a novel weight initialisation mechanism that comprehensively improves the comprehensive performance of SATLike 3.0 in handling (W)PMS instances. This thesis verifies the effectiveness of the improved method through experiments, and finally applies the obtained method to solve the combination auction, dish planning and FPGA placement optimization problems, and achieves good results.

KEY WORDS: Stochastic Local Search Solver; Partial MaxSAT; Initial Solution; Weighting Scheme; SATLike3.0

目 录

符号列表

SAT	最大可满足性问题
PMS	部分最大可满足性问题
WPMS	加权部分最大可满足性问题
UP	单元传播
UPDeci	一项初始解生成技术 (Unit Propagation Based Decimation)
SLS	随机局部搜索 (Stochastic Local Search)
MSE	一项国际 MaxSAT 比赛 (MaxSAT Evaluaion)
BMS	一项概率采样技术
Modularity	模块化程度，用以衡量图的社区结构
FPGA	一种可以重构电路的芯片，中文名是现场可编程门阵列 (Field Programmable Gate Array)

第 1 章　绪论

1.1　研究背景与意义

可满足性问题 (Boolean Satisfiability Problem, SAT) 是第一个被证明的非多项式完全问题，即如果 SAT 问题可以在多项式时间内求解，那么所有非多项式时间问题都可以在多项式时间内求解[1]。SAT 问题用于判断逻辑命题公式是否存在一组使得整个公式为真的指派。如频谱分配[2]、电路设计自动化[3] 等相关问题等都可以归结为 SAT 问题进行求解。SAT 问题的实例通常使用合取范式 (Conjunctive Normal Form, CNF) 表示：问题公式由一组子句的合取构成，子句由一组文字的析取构成，文字包含正文字（变量本身）和负文字（变量的非）。当存在为真的文字时子句被满足，当所有子句被满足时整个命题公式被满足。SAT 问题的本质即探求繁多的布尔变量之间的逻辑推理关系是否成立。最大可满足性问题 (Maximum Satisfiability Problem, MaxSAT) 是 SAT 问题的变体，目标是寻找一组满足最多子句的指派，相关问题包括最大割[4]、汽车再配置[5] 和群组测试等[6]。尽管 MaxSAT 已经具有很强的表达力（SAT 问题是它的子集），但在同时处理必要条件和非必要条件时无法给出优雅的解决方案。部分最大可满足性问题 (Partial MaxSAT, PMS) 引入了硬约束和软约束的概念[7]，将子句分为必须要满足的硬子句集合，以及需要尽可能多地满足的软子句集合，进一步提升了问题公式的表达力，可方便地对最优规划问题[8] 和高校课程表安排[9] 等问题进行编码。特别地，加权的部分最大可满足性问题 (Weighted Partial MaxSAT, WPMS) 为每个软子句分配一个正整数作为权重，WPMS 的目标为满足所有硬子句的同时，使被满足的软子句权重达到最大。因此，作为 SAT 问题和 MaxSAT 问题的超集，(W)PMS 进一步扩大了命题公式的可表示范围，能够对更广泛的现实问题进行建模和求解。

目前主要有两种用于解决 (W)PMS 问题的方法：完备方法和不完备方法。完备方法会根据搜索树来探索优化问题的所有可行解以证明解的最优性。早期的完备方法包含基于分支定界算法[10–12] 和 DPLL 范式[13–16]。由于完备方法可以保证最优解的特点，在解决小中型问题实例时效果颇佳。然而，当面对现代许多应用

场景时，尤其是工程和工业应用都涉及大规模问题实例，完备算法难以在人们能接受的时间窗口内给出满意的解答。不完备方法可以快速响应现实需求，而随机局部搜索求解器 (Stochastic Local Search, SLS) 是不完备方法的一种高效实现，通常能在合理的时间内返回一组高质量解。近年来 SLS 在某些关键技术上取得了突破性进展，2018 年由雷震东和蔡少伟提出的 SATLike[17] 使 SLS 首次在工业实例上达到媲美完备方法的水准，SLS 也逐渐成为求解 (W)PMS 及其子集的常规引擎之一。自 2006 年 MaxSAT Evaluation (MSE) 举办以来，公开提供了大量标准数据集和求解器源码，成为研究使用 SLS 解决 (W)PMS 问题的重要推动力。

诚然 SLS 通常可以在时限内快速响应一组高质量解，但其仍然存在无法解决某些问题实例的缺陷。究其原因，SLS 无法像基于 SAT 求解器的算法一样有效利用问题实例的结构而在搜索的方向上呈现一种盲目性[18–21]。已有研究指出 SAT 求解器使用 VSIDS、LRB 等机制促使子句间矛盾的发生[22, 23]，而这些经常发生矛盾的变量、子句变得格外重要，往往蕴含着具体问题实例的结构信息，对求解问题实例十分关键；此外，Dist[24] 等 SLS 由于开发了优先硬子句的策略取得成功。因此，为 SLS 开发识别冲突变量与优先满足硬子句的技术可能是提升其求解性能的解决方案。期间可能面临的挑战如下：（1）改进思路的实现需要依托于一个具体的求解器，怎样选择一个合适的基准求解器？（2）如何在基准求解器中定义矛盾，即如何高效地识别和分析矛盾？（3）以何种形式嵌入本文的改进思路？

不难想到，挑战（1）描述的基准求解器应当具有挑战（2）和挑战（3）规定的属性：使用 SAT 求解器的相关技术，并建立了引导算法搜索方向的唯一标准。SATLike 系列求解器契合上述需求，且其本身的性能得到世界范围研究者的认可[25–27]。特别地，选取 SATLike3.0[28] 作为基准求解器。其一，SATLike3.0 使用单元传播技术构建初始解，该项技术一定能监测到子句间矛盾的发生，具体地，当单元传播得到空子句时算法即遇到了矛盾。其二，可以粗粒度地修改求解器工作流程以改变搜索方向，也可以细粒度地调整 SATLike3.0 动态维护的子句权重来影响变量的得分，进而引导搜索方向。综上，本文选取 SATLike3.0 作为基准求解器，主要实现分析子句矛盾和优先满足硬子句的策略。此外，需要考虑赋予算法从先前解中吸取经验的特性，从而在策略没有命中时为变量选择合适的极性和

提取可能对求解问题有益的子句。本文的策略也应提供轻量级的实现，原因在于SLS 需要及时交付一组答案，如果引入了复杂的计算，则会大量占用搜索解的时间（SLS 通常在一定时限内运行）。

基于以上背景和挑战，本文首先考虑了强制优先满足硬子句策略对SATLIke3.0 的性能影响，改造后的求解器记作 HFCRP-F (Hard Falsified Clause Random Propagation With Feedback)。这部分工作粗粒度地修改了 SATLIke3.0 生成初始解的流程：在当前数据状态下不存在单元子句时优先处理尚未满足的硬子句中的未赋值变量；同时，当传播硬单元子句遇到矛盾时参考目前找到的局部最优解进行赋值。这部分工作的不足之处在于策略的实现方式过于强硬，难以与其他策略配合，且在参考局部最优解时没有考虑该解的质量，可能使算法受到低质量解的误导。基于上述考量，在实现分析子句矛盾的策略时采用了细粒度的方式，即通过调整冲突变量所属子句的权重使算法倾向于优先满足此类重要的子句；还通过简单的统计学方式为可能的“高质量”子句[29] 增加权重，得到的算法记为 SATLC (SATLike3.0 Based On Confilcts)。最后，将 HFCRP-F 和 SATLC 的策略融合，得到的方法 HRP-C (Hard Falsified Clause Random Propagation With Feedback Based On Confilcts) 在 (W)PMS 实例上均表现优异。本文还将 HRP-C 应用于解决现实中常见的组合拍卖、菜品规划和 FPGA 布局优化等问题，取得了良好的效果。

1.2 研究现状

SLS 用于解决困难的组合优化问题，通常从一个初始的候选解出发，不断搜索其邻域以逼近最优解，潜在的主要问题包括容易陷入局部最优以及变量的循环翻转。具备游走策略的 GSAT[30]、模拟退火法[31]、禁忌搜索[32] 和动态局部搜索[33] 等方法均属于 SLS，有些方法在初始解邻域空间内只执行单一的搜索策略，有些方法则集成多种搜索技术。SLS 通常涉及构造初始解的启发式方法、局部搜索过程和候选解的扰动策略，以及用于衡量是否接受当前解的目标函数。

已有的 (W)PMS 局部搜索求解算法可依据年代大致归为两类。第一类为早期的研究，通过赋予硬子句相当大的权重以与软子句相区分，但实质上将两类子句均当作软子句进行计算[34, 35]。这种不区分软硬子句的方法需要为硬子句分配合适

的权重：如果与软子句的权重差别不明显，则算法无法自发满足所有硬子句；如果与软子句权重差别明显，则算法只关注满足硬子句，忽视软子句的满足情况[35]。事实上，这种算法的实际应用效果往往很差，因为虽然满足所有硬子句才能输出一个可行解，但有些硬子句总是被满足的，在搜索过程中没必要始终保持很高的权重，而即使动态调整权重，也显然无法解决这个问题，因此引入类别参数区分硬子句和软子句是更好的选择。

第二类研究为近期的 (W)PMS 局部搜索算法。在传统的 SLS 框架上引入了得分的概念，即通过评估翻转变量对目标函数的影响来决定是否翻转变量。通过计算各类得分的方法，不同的 SLS 也设计了不同的变量选择启发式策略。具有代表性的如 Dist[24]，通过不同种类的得分机制选择被翻转的变量，但仍保留了优先硬子句的思想：先比较硬得分，如果相等则再比较软得分。这种方式与第一类研究中通过指定合适权重优先硬子句的方式相比更加灵活，性能也有很大的提升。美中不足是得分机制略显复杂，效果仍然无法达到完备方法的水准。

目前国内的 SLS 研究取得了领先地位。主流的方法主要包括三种：基于区分硬子句的方法，基于格局检测的方法，以及动态（随机）局部搜索方法。基于区分硬子句的方法主要有 Dist 和 DistUP[36] 等求解器，在求解的过程中保留了优先满足硬子句的思想，主要的缺点是难以在“容易满足的硬子句”和“不容易满足的软子句”间权衡，这个问题在 NuDist 中得以改善；基于格局检测的方法主要由 CCASat[37] 和 CCLS[38] 相关求解器组成，在评估变量时会综合考虑翻转变量对目标函数的影响和变量翻转前后的配置信息是否发生改变，该项技术有效缓解了 SLS 的变量循环翻转问题；动态局部搜索方法主要包括 SATLike 系列求解器[28]，主要的技术创新在于开发了高效的初始解生成机制和权重机制，在算法陷入局部最优时同时调整硬子句和软子句的权重，事实证明这种权重机制可以有效地引导算法脱离局部最优。这些方法主要依赖独特的子句加权机制和某些启发式算法。从初始解的构造角度来看，DistUP 中提出的 PrioUP 技术改变了 SLS 用完全随机的方式构造初始解的常态，转而利用 SAT 求解中常用的单元传播技术进行数据的预处理和初始解的构造，大大提升了 SLS 解决 (W)PMS 实例的能力；从权重机制方面来看，Dist 系列和 SATLike 系列分别使用了严格优先硬子句（区分不

同种类的变量得分机制）和仅依靠权重引导搜索方向（硬子句的权重调整幅度更大，且为软子句权重设限）的加权机制，前者对探索可行解很有帮助（这种思想在 SATLike3.0 的初始解构造算法中也能见到[28]），后者则由于能更有效地脱离局部最优从而提升解的质量；从避免无效搜索的角度来看，CCLS 等算法使用的格局检测和变量年龄等机制缓解了 SLS 常见的循环翻转问题。上述多种技术的合理组合最终达到局部搜索求解系统最先进的效果。

1.3 主要研究内容

本文主要开展了 (W)PMS 的局部搜索算法改进策略研究。首先，本文梳理了 SLS 各项关键技术的发展过程和解决的问题；其次，在基准求解器 SATLike3.0 上分别独立实现了优先满足硬子句和矛盾时参考局部最优解的机制，以及为矛盾子句和高质量子句增加额外权重的机制，并在国际竞赛 MaxSAT Evaluation (MSE) 提供的标准数据集上进行了测试和分析；再次，本文将上述两个机制进行了轻巧的融合，在基本不修改原算法结构的前提下结合了两种机制各自的优势，得到了求解器 HRP-C，同样在 MSE 的数据集上进行了测试；最后，本文将 HRP-C 应用到解决组合拍卖、菜品规划和 FPGA 布局优化问题中，取得了良好的效果。

本文的主要贡献总结如下：

(1) 本文梳理了局部搜索求解器的关键技术，讨论了现有研究存在的不足之处。

(2) 本文在 SATLike3.0 的基础上开发了两种求解器，分别在处理 WPMS 和 PMS 实例上取得了明显的优势。

(3) 本文在以上工作的基础上，将提出的两个求解器中应用的机制进行了轻巧的融合，得到的方法 HRP-C 在 PMS 和 WPMS 实例上相比 SATLike3.0 均取得了显著优势。

(4) 本文将得到的算法应用于组合拍卖、菜品规划和 FPGA 布局优化问题中，取得了良好的效果。

1.4 本文的组织结构

第 1 章绪论。介绍使用 SLS 解决 (W)PMS 的研究背景和意义，讨论改进 SLS 的必要性，对 SLS 的研究现状进行概述，分析当下研究成果的不足，提出本文的工作内容。

第 2 章背景知识。给出 (W)PMS 的定义，介绍了求解此类问题需要掌握的技术概念和数据集的来源。

第 3 章随机局部搜索求解器的优化方向及先进技术。从最基本的随机局部搜索求解器出发，梳理 SLS 发展的路线图；介绍近年来 SLS 的关键技术突破和目前最先进的随机局部搜索求解器 SATLike3.0；讨论可能的算法优化方向。

第 4 章优先随机传播硬变量的初始解生成过程：HFCRP-F。介入 SATLike3.0 的初始解生成过程，使其在同等情况下优先传播未满足的硬子句中的未赋值变量，提升解的质量。

第 5 章基于子句矛盾的初始权重配置方法：SATLC。基于第 4 章工作中存在的不足，提出在初始解生成阶段分析子句矛盾信息的方法，重新配置子句初始权重，进而帮助 SATLike3.0 解决更多问题实例。

第 6 章提取潜在关键子句：HRP-C。结合第 4 章和第 5 章的改进思路，在初始解生成过程中分析子句矛盾的同时，细粒度地引导算法优先传播未满足硬子句中的未赋值变量，从而提升算法的综合性能；将组合拍卖、菜品规划和 FPGA 布局优化问题编码为 (W)PMS 实例，使用 HRP-C 进行求解。

第 7 章总结与展望。总结了本文的工作成果，给出作者对 SLS 发展方向的看法，以及对基准求解器 SATLike3.0 的期待。

第 2 章　背景知识

2.1　算法复杂性

2.1.1　计算复杂性

计算复杂性[39] 是对算法运行时消耗资源数量的衡量，包括时间复杂度和空间复杂度。在编写算法时，通常需要预测其执行时间和占用的内存，以确保该程序可以在现实世界中执行完成。算法的空间复杂度是容易计算的，而时间复杂度往往难以估计。有些算法仅需多项式时间即可执行完毕，而有些算法需要指数时间才能完成。研究者通常为不存在多项式时间解法的问题开发启发式算法，牺牲解的质量，换取在合理的时间内“试”出问题的解决方案。

2.1.2　问题难度

问题按照难度可分为 P 类问题、NP 类问题、$NP-hard$ 问题和 NPC 问题[40]，它们之间的关系如图 2.1所示。

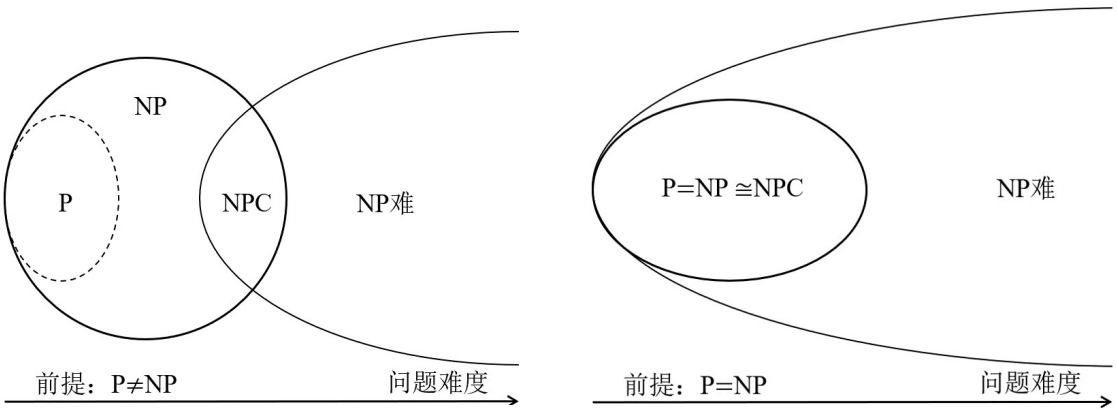

图 2.1　P、NP、$NP-hard$、NPC 之间的关系

P 类问题可以在多项式时间内求解，即存在一个解决该类问题的算法，该算法执行步骤的数量以 n 的多项式函数为界，其中 n 为问题输入数据的规模。P 类问题是容易求解的，如线性规划问题属于 P 类问题[41]。

对于一个问题，如果无法确定是否存在多项式时间的算法，但可以保证在多

项式时间内验证该问题的解的正确性，那么这个问题属于 NP 问题。

如果一个问题可以规约为任何 NP 问题，则称这个问题是 $NP-hard$ 的。

一个既属于 NP 又属于 $NP-hard$ 的问题被称为 NPC 问题。具备 NPC 难度的问题意味着仍未找到有效解决该问题的算法程序，诸如 TSP 问题、可满足性问题和点覆盖问题等重要的计算机科学问题都属于这一类。再例如大多数现代密码学依赖于构造具有高计算复杂度的加密方案，这使得正向生成公钥和私钥的速度很快，但逆向破解的时间成本极高，甚至穷尽当代的计算资源也无法求解[42]。当必须解决一项 NPC 问题时，通常采用不完备的算法在多项式时间内得到近似解。

容易得出这样的结论：为任意一个 NPC 问题找到有效算法意味着可以为所有 NP 问题找到有效的算法，因为这些困难问题都是 NPC 问题的子集。而一旦找到了这样的算法，则会直接证明 $P=NP=NPC$，彻底颠覆计算机科学和数学的许多领域。目前科学家们仍无法证明 $P=NP$，本文是以 $P\neq NP$ 为前提展开的。

2.2 部分最大可满足性问题和相关定义

2.2.1 问题概述

SAT 问题是计算机科学和人工智能的一个基本问题，也是被科学家证明的第一个 NPC 问题。SAT 问题即建立一组变量的指派，使得整个布尔公式的值为真。目前，SAT 求解器在调度、模型检查等实际应用中取得了巨大的成功。PMS 和其加权版本 WPMS 是 SAT 问题的一个优化变体。通过引入硬约束（必须满足）、软约束（尽可能多地满足）等概念实现了比 SAT 更强的表达力，因此被研究者广泛关注。著名的最大团、最小点覆盖等 $NP-hard$ 问题都可以通过构造点与边的关系编码为 (W)PMS 实例进行求解。

2.2.2 形式化定义

给定布尔变量的集合 $V=\{v_1,v_2,\ldots,v_n\}$，一个文字 l_i 可以是变量本身 v_i（正文字）或它的否定形式 $\neg v_i$（负文字）。一个子句 c_i 是一组文字的析取，即 $\{c_i=l_{i1}\vee l_{i2}\vee\ldots\vee l_{ij}\}$。特别地，当子句的长度恰好为 1 时称它为**单元子句**。最后，一个用合取范式 (Conjunctive Normal Form, CNF) 表示的命题公式 F 为一组子句的

合取，即 $F = \{c_1 \wedge c_2 \wedge \ldots \wedge c_m\}$。容易观察到一组 CNF 范式的公式具有以下特征：当子句中任意文字的值为真时，该子句被满足；当命题公式的所有子句被满足时，该公式才被满足。如果用符号“1”和符号“0”分别表示文字极性的正负，那么一个映射 $\alpha : V \to \{0, 1\}$ 被称为命题公式 F 的一组指派，当所有布尔变量都被赋值时，该指派被称为一组**完整的指派**。给定 CNF 范式的命题公式 F，(W)PMS 将所有子句分为硬子句和软子句两个子句集合，其目标为找到一组完整指派 α，使得所有硬子句被满足，且满足权重（数量）尽可能大的软子句。当所有硬子句都被满足时，称 α 为该命题公式的一组**可行解**。显然，当 (W)PMS 实例中各个子句的权重为 1 时即等价于不加权版本的 PMS 实例，因此在不做特殊说明的情况下，本文以 WPMS 的角度进行讨论。

下面给出本文可能使用的相关定义：

- **硬子句**。必须被满足的子句，通常通过足够大的权重进行标识。
- **软子句**。非硬子句的子句称为软子句。
- **硬变量**。出现在任意硬子句中的变量称为硬变量。
- **软变量**。非硬变量的变量称为软变量。
- *make_score*(v)。考虑将变量赋值为真或假，*make_score*(v) 为将当前变量赋值后，新增被满足子句的数量或权重之和。
- *break_score*(v)。考虑将变量赋值为真或假，*break_score*(v) 为将当前变量赋值后，旧有被满足子句变得不满足的数量或权重之和。
- **翻转变量的得分** *score*(v)。考虑变量 v 已被赋值的情况，翻转这个变量的得分为翻转之后被满足子句数量 (权重) 的增量，即 *score*(v) = *make_score*(v) − *break_score*(v)。
- **好变量** *good_var*。*score*(v) > 0 的变量 v 被称为好变量。
- **软得分** *sscore*(v)。考虑将变量赋值为真或假，软得分为将当前变量赋值后，被满足软子句数量（权重）的增量的最大值。

- **硬得分** $hscore(v)$。考虑将变量赋值为真或假，硬得分为将当前变量赋值后，被满足硬子句数量的增量的最大值。
- **变量的年龄** $age(v)$。表示自上次 v 的值被翻转以来已经进行的翻转次数。
- **变量的邻域** $N(v_i)$。定义为 $N(v_i) = \{v_j \in N(v_i), i \neq j\}$，即当且仅当两个不同的变量同时出现在至少一个子句中时，它们才是相邻的。
- **变量的格局检测结果** $conf(v_i)$。表示自变量 v_i 最后一次被翻转以来，是否有任何变量 $v_j \in N(v_i)$，$i \neq j$ 的值被翻转。如果至少有一个这种变量的值自 v_i 上次被翻转后被翻转，则称 v_i 的配置被改变，$conf(v_i) = 1$；否则，$conf(v_i) = 0$。
- **代价** $cost$。对于 PMS，如果所有硬子句被满足，那么 $cost$ 为未满足软子句的数量，否则为正无穷；对于 WPMS，如果所有硬子句被满足，那么 $cost$ 为未满足软子句的权重之和，否则为正无穷。
- **单元传播 (Unit Propagation, UP)**。仅包含一个文字的子句被称为单元子句。为了满足这种子句，必须使其仅有的文字为真。那么公式中所有包含这个文字的子句都可以被删除（由于这个文字为真而被满足）；公式中所有包含这个文字的非的子句都可以删除这个文字的非（这个文字的非一定为假，所以对子句是否被满足没有影响）。

2.3 局部搜索方法

局部搜索方法被广泛运用于解决困难的组合优化问题，主要由搜索空间、邻域关系、目标函数和移动方法四部分定义[43]。**搜索空间**是给定问题实例的变量中所有可能的真值指派集合，如果问题实例具有 n 个布尔变量，那么搜索空间的大小，即候选集的大小为 2^n；**邻域关系**定义了不同解之间如何连接的拓扑结构，通常将两个只在一个布尔变量的赋值上相异的解定义为相邻解；**目标函数**定义了评价当前解质量的标准，局部搜索方法通常依照目标函数值的变化决定是否接受当前解；**移动方法**规定了怎样从一组可行解移动到另一组可行解，通常涉及变量的选择和翻转。

在使用局部搜索方法求解 (W)PMS 时，通过翻转一个或一组变量的值以得到当前解 α 的一个邻域解 α^*。在每一次实际翻转前都需要进行伪翻转，检查伪翻转后的目标函数值 $cost(\alpha^*)$ 相比 $cost(\alpha)$ 是否有提升，如果 $cost(\alpha^*) < cost(\alpha)$（满足的子句数量或权重更多）则确认翻转相应的变量（集合）并接受 α^*。以上过程会循环执行，直到找到代价为 0 的解（满足所有子句）或算法超时。

SLS 是一种结合了随机性的局部搜索方法。GSAT[30] 首先随机生成一组完整赋值 α，随后不断贪婪地挑选对目标函数提升最大的变量 v 进行翻转，直到达到终止条件；GWSAT[44] 是带有随机游走机制的 GSAT 变种：有概率 p（也称噪声函数）在随机模式（随机挑选未满足子句中的变量进行翻转）和贪心模式间（同 GSAT）切换，以向搜索空间的多样性方面做出让步。WalkSAT[45] 同样基于 GSAT，具备随机模式和贪心模式：首先随机挑选一个未满足的子句 c，并且有 p 的概率随机翻转其中一个变量的值，否则贪心 c 中一个对目标函数提升最大的变量进行翻转。SLS 方法很大的缺陷是容易陷入局部最优和循环翻转的困境。SLS 用于脱离局部最优的策略包括随机游走、大邻域搜索[46] 和权重调整等；为了避免循环翻转，某些 SLS 开发了格局检测和禁忌搜索等机制。此外，每种机制都会引入新的控制参数，如何调整这些参数也成为 SLS 的一大难题。

除了求解技术上难以精进，由于 SLS 需要在有限时间内做大量运算，其难点还在于提出的技术要足够简单，这导致 SLS 的研究主要通过大量繁重的实验来支撑技术的有效性，取得的成果难以得到认可。

2.4 MaxSAT Evaluations

MaxSAT Evaluations (MSE)[1]是一项年度的国际赛事，首次举办于 2006 年，主题是评估目前最先进的使用 MaxSAT 范式解决优化问题的计算系统。MSE 本着真正的评估精神，旨在推广使用 MaxSAT 范式解决现实世界中 $NP-hard$ 优化问题。同时，MSE 向 MaxSAT 社区提供大量标准数据集，并且要求参赛求解器开源和提供技术细节概述，这十分方便于社区成员对求解器的学习和研究。本文实验采用的数据集均来源于 MSE。

[1]https://maxsat-evaluations.github.io

MSE 2020[2]共包括四条赛道：两条完备赛道、一条“Top-k”赛道（列举最优的几个解），和一条不完备赛道。不区分“工业”和“手工”数据集。

MSE 2020[47] 新增的数据集包括硬件漏洞的自动合成、联盟结构的产生、在多状态特征上寻找最兼容的系统生成树、网络分析、消除程序歧义、铁路时间表、点覆盖、单机调度、肿瘤演变、用户授权查询和可重构扫描网络中安全弱点的见证者等问题；参赛的不完备求解器中 Loandra、StableResolver、TT-Open-WBO-Inc、SATLike-c 和 sls-mcs 综合性能比较优异。

MSE 2021[3]共包括四条赛道：两条完备赛道和两条不完备赛道（分别评估解决 MaxSAT 和 Weighted MaxSAT 的不完备求解器）。不区分” 工业” 和” 手工” 数据集。

MSE 2021[48] 新增的数据集包括学习决策树、用学习的二值神经网络进行规划、持续转换活动最大化和大学课程时间表等问题；参赛的不完备求解器中 Loandra、StableResolver、TT-Open-WBO-Inc、Open-WBO-Inc 和 SATLike-c 综合性能比较优异。

MSE 2022[4]共包括五条赛道：两条完备赛道，两条不完备赛道，以及一条增量赛道（评估求解器求解一系列关联的 MaxSAT 实例的性能和适用性）。不区分” 工业” 和” 手工” 数据集。

MSE 2022[49] 新增的数据集包括增强学习和双目标优化等问题；参赛的不完备求解器中 Loandra、DT-HyWalk、TT-Open-WBO-Inc、Open-WBO-Inc、NuWLS-c 和 noSAT-MaxSAT 综合性能比较优异。

从 2020 年起，MSE 中更为优秀的不完备求解器不再使用单一的求解引擎 (采取完备方法与不完备方法结合的方式）。

[2]https://maxsat-evaluations.github.io/2020
[3]https://maxsat-evaluations.github.io/2021
[4]https://maxsat-evaluations.github.io/2022

第 3 章　随机局部搜索求解器的优化方向及先进技术

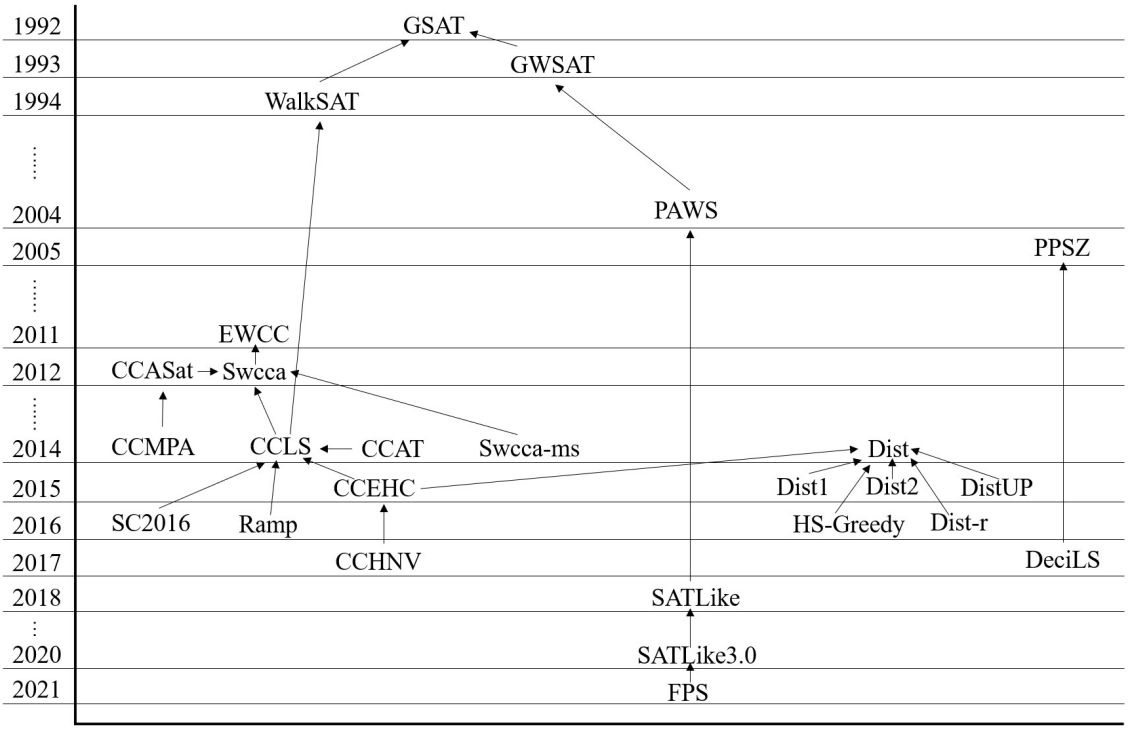

图 3.1　SLS 发展路线图

图 3.1提供了部分 SLS 的发展路线图[1]。经典的 GWSAT 和 WalkSAT 是 GSAT 的变体，在随机模式的基础上加入了贪心模式。此时并未建立权重机制，逃离局部最优的手段也只有依靠随机游走。

沿 WalkSAT 的技术发展方向前进，CCLS 在随机游走的基础上引入了 CCM (Configuration Checking with Make) 启发式方法，通过判断格局检测是否通过以及 *score* 和 *make_score* 的值是否大于 0 将变量区分为不同的优先级，在 Weighted MaxSAT 问题上的效果相比 WalkSAT 具有很大提升。

Dist 只对硬子句应用权重机制，并通过区分硬得分和软得分严格优先满足硬子句，在探索可行解方面效果提升显著。DistUP 使用单元传播技术构造初始解，主要在工业实例上取得性能提升。

沿 GWSAT 的技术发展方向前进，PAWS[50] 建立了子句的加权机制，得以

[1]图中未展示采用完备求解器加不完备求解器的混合求解器，即仅展示使用单一搜索引擎的 SLS。

平滑地引导 SLS 脱离局部最优。SATLike 在此基础上提出了同时处理硬子句和软子句，但对前者调整的幅度更大，同时对后者的权重成长设限的权重机制，使 SATLike 得以同时对硬子句和软子句的重要性进行权衡。SATLike3.0 在此基础上提出了基于单元传播的消解方法用于生成初始解，并进一步引入了 *sscore*、*hscore* 等概念，用于在构造初始解时为变量选择极性。FPS[51] 额外开发了深度邻域搜索机制，在不存在好变量时尝试伪翻转一组变量改善当前解[2]。

下面列举 SLS 中部分具有代表性的技术及相关应用。

3.1 权重调整机制

最典型的应用加权机制的局部搜索方法是动态随机局部搜索求解器。这种方法在搜索开始时，会采取简单的配置策略赋予所有或部分子句不同的权重；在搜索过程中，尤其是在当前子句权重配置下无法改善目标函数时，会为某些它认为特别的子句增加权重（通常是未被满足的子句），从而破坏原有子句权重的差异关系以推进算法的执行。这种权重调整机制符合直觉且经长期实验证明十分有效，然而关键点在于调整哪些子句的权重、如何调整以及调整的幅度，基于这些差别延申出了许多机制和变种权重调整方法。

早期的权重机制包括不区分软硬子句和只调整硬子句权重两类。由于硬子句与软子句的权重差别过大会严重降低搜索效率（软子句权重过大则不易找到可行解，硬子句权重过大则难以逃离局部最优），且不区分软硬子句的加权方法每次会以相同的幅度调整子句权重，因此这种方法通常需要为硬子句手动设置合适的初始权重。在 [24, 52] 等工作中采用了只对硬子句加权的机制，旨在尽可能满足硬子句的同时探索更多的可行解空间，因此这种方案对找到可行解有利，而难以达到较低的 *cost*。

目前最为有效的加权机制采取了区分软硬子句的方案，即引入硬得分和软得分，在多数情况下只参考二者之一推进算法的执行；也有算法将这个概念理解为同时调整硬子句和软子句的权重，但严格区分两者调整的幅度，这种方法取得了目前最先进的效果。以 SATLike 系列求解器提出的 Weighting-PMS 权重机制为

[2]FPS 在本篇工作完成时未正式发表，且与本文的改进方向不相关，因此本文不涉及与 FPS 的探讨和比较。

例[17, 28]，算法将硬子句的权重初始化为 1，将软子句的权重初始化为问题实例中指定的数值，并根据问题实例的规模为软子句的权重设置上限；所有子句的权重下限都为 1；当算法无法通过翻转变量的值降低 *cost* 时有一定概率提升未满足子句的权重，否则降低已满足子句的权重。根据具体问题实例的规模和性质，硬子句权重每次调整的幅度为 1 或 300，软子句权重每次调整的数值为 1。

基于一定概率放大和缩小子句权重可以减少极端权重数值的出现，使各子句权重接近平均权重，这被称作权重的平滑策略。在 SATLike 系列求解器中表现为当算法陷入局部最优时有 *sp* (Smoothing Possibility) 的概率降低已满足子句的权重，有 $(1 - sp)$ 的概率提升未满足子句的权重。根据具体问题实例的规模和性质，SATLike 系列求解器中 *sp* 的数值采用 0.000003 或 0.01。相关技术最早应用于 [53–55]，在平滑阶段使用公式 3.1调整权重，其中 ρ 是 0 到 1 之间的实数，$\bar{w}$ 是平均权重。

$$w := w \cdot \rho + (1 - \rho) \cdot \bar{w}. \tag{3.1}$$

NuWLS-c 针对 SATLike 权重调整的时机和上下限进行了详细配置，主要改善了由于子句权重更新导致的权重差异关系被破坏的问题。NuWLS-c 区分了硬子句和软子句的 *sp* 值，且将软子句的权重上限由固定值更改为与子句平均权重相关[56]，具体如公式 3.2所示。

$$w_{soft\_limit} = \frac{s_{avg} \times w_{org}(c)}{ws_{avg}(I)} + \theta. \tag{3.2}$$

其中 s_{avg} 是软子句的平均权重，$w_{org}(c)$ 是被调整子句的原始权重，$ws_{avg}(I)$ 是当前问题实例所有子句的平均权重，θ 是参数。

3.2 变量重要性的评价

格局检测策略 (Configuration Checking, CC) 是一种简单有效的局部搜索策略[37, 57, 58]，可以有效缓解循环翻转的现象。CC 的基本思想是：如果一个变量翻转前后的配置信息没有发生改变，则不应予以翻转。配置方式为记录当前变量邻居的值是否发生了改变，已有的研究将邻居定义为相邻的变量、相邻的硬变量、

包含当前变量的子句和包含当前变量的硬子句[52, 59]。CC 及其变体中定义了 *CCD*（通过配置检查且 *score* 值为正的变量）和 *CCMP*（通过配置检查且 *make_score* 值为正的变量）等变量集合，从定义上看，*CCD* 最能改善当前解，其次是 *CCMP*，最后是随机选择的变量。

score(*v*) 可以反映改变变量 *v* 的极性对目标函数的影响，该值越大则越能降低 *cost*，因此大部分 SLS 都以此作为挑选变量进行翻转的标准。从 SLS 在同等条件下选择变量的倾向来看，*score* 为正的变量优于 *score* 为负的变量，同样的结论可以推广到 *sscore* 和 *hscore* 上。

由于变量的得分通常由子句的权重和子句间的邻里关系确定，因此子句的属性也会影响变量的重要程度。Dist、SATLike、FPS 等算法均优先满足硬子句或为硬子句赋予更大的权重增量，这使得硬变量比软变量更容易得到关注；此外，单元子句中的文字只能为真，即其所代表的变量只能有一种可能的赋值以使子句满足，单元子句中的变量显然是重要的。

在求解问题实例的过程中有时会发现可行解中部分变量的赋值情况始终不变，在理想情况下这类变量在该问题实例的所有可行解中仅存在一种赋值，这类变量可以极大降低求解当前问题实例的复杂度，也是需要着重关注的。

3.3 参数的自动配置

SLS 的每项技术都会引入额外参数，一款成熟的求解器往往有成百上千个参数可供调整，即使是原始的 SATLike 算法也包含 15 个参数。参数的设定对算法的综合性能影响巨大，一些细微的调整就可导致算法搜索到相异的解。自动配置参数的目标是找到使得算法实现其最优性能的参数配置，有离线和在线两种配置方式[60]。

离线的配置方式会引入额外的预训练过程。早先的训练过程只能调整数值型参数；随后发展出既能调整数值参数，又能调整类别参数的方法，通用性大大增强。

在线的配置方式即在探索问题最优解的过程中动态调整参数。该方法对比离线方法最大的优势在于使求解器适应当前问题实例而非一类问题，因此更适合用

于求解工业场景下大量存在的结构化问题实例。研究的方向大致分为响应式搜索和进化计算。在线配置方法面临的最大问题是无法调整多个参数，甚至通常只针对一个参数进行调整，这使得该项技术的应用者必须深刻理解 SLS 设计和所解决的问题，才能对最关键的参数进行动态调整。

3.4 搜索范围

SLS 通常涉及候选集的计算，并从中挑选最佳元素。这种启发式方法可以帮助求解器快速收敛至局部最优，也因此被广泛应用于各种 NP 难问题的局部搜索求解技术中：如在求解最小点覆盖问题和 SAT 问题时，GSAT、选择最大增益对启发式、最小损失删除启发式、Novelty++ 启发式均采用了这种方式。这类需要准确划分候选集的策略往往具有 $O(N)$ 的时间复杂度，在面对大规模实例时会消耗大量时间。蔡少伟在 FastVC 中提出的 BMS (Best from Multiple Selections) 策略可以近似达到上述“挑选最佳元素”的策略，且时间复杂度仅为 $O(1)$[61]。具体做法为：从集合 S 中带放回地随机选择 k 个元素，然后根据某个比较函数 f 返回最佳的元素，其中 k 是一个参数。BMS 不再需要遍历所有元素以过滤初始候选集，而是直接随机挑选，且 f 可为任意简单的度量标准。经计算验证，对于集合 S 和 0 到 1 之间的实数 ρ, BMS 返回元素的质量不劣于集合 S 中 $\rho|S|$ 个元素的概率大于 $1-\rho^k$。

LNS (Large Neighborhood Search) 首次由 Shaw 提出用于解决车辆选路问题[62]，后被用于解决高校时间表、进化算法、带时间窗的取送货等问题。LNS 包含两个组件：破坏 (destroy) 和插入 (insertion)，前者会取消指定事件的安排，后者会为之前取消的事件安排时间，即破坏和修复当前解。LNS 的有效性在于搜索解的大范围邻域更容易脱离局部最优，但需要解决如何定义邻域及大范围搜索带来的耗时问题。VNS (Variable Neighborhood Search) 和 VDS (Variable Depth Neighborhood Search) 都探索当前解决方案的不同邻域，以找到更好的解决方案。他们首先生成初始解，然后迭代应用一组邻域结构，这些邻域结构逐渐变大或加深，直到找到局部最优解。如果当前解决方案在给定邻域中是局部最优的，则搜索移动到具有相同大小或深度的不同邻域，直到找到更好的解决方案。这两种方

法都可以有效地解决优化问题，选择哪种方法取决于要解决的具体问题和空间的特性。

3.5 初始解生成策略

SLS 的初始解大多采用随机生成的策略。近些年有研究者利用单元传播技术构建初始解，该项技术由蔡少伟等人广泛应用于 DistUP、SATLike3.0 等求解器中，方法名为 UPDeci (UP-based Decimation)。UPDeci 优先传播硬单元子句，其次是软单元子句，再次是任意未被赋值的变量。UPDeci 的优势是实现简单、运算速度快，配合一些简单的启发式策略可以提供一组高质量的完整赋值，对 Dist、SATLike 系列求解器性能提升巨大。

3.6 SATLike3.0

目前最先进的 SLS 为 SATLike 系列求解器。自 SATLike 于 2018 年参加 MSE 以来，始终作为求解引擎之一被集成于不完备赛道的部分最先进混合求解器中：Open-WBO-Inc (inc-bmo-satlike, 2019-2022)、SATLike-c (2020, 2021)、SATLike3.0 (2019)、NuWLS-c (2022)、noSAT-MaxSAT (2022)、DT-HyWalk (2022)。SATLike3.0 是 SATLike 的优化版本，在基础的动态局部搜索框架上引入了贪心 *hscore* 和 *sscore* 的启发式和基于单元传播的初始解生成等技术。SATLike3.0 的工作流程如图 3.2所示，首先使用 UPDeci 算法生成一组初始解，硬单元子句最优先被传播，其次是软单元子句，再次是随机传播；在时限与 max_flips 翻转次数以内时，优先用 BMS 机制挑选一个 *score* 为正的变量进行翻转，否则调整权重后随机满足一个硬子句或软子句；当变量的翻转次数达到 max_flips 时，SATLike3.0 重新生成一组初始解继续搜索，直到超时；搜索期间每当找到更优的局部最优解都汇报此解信息。

SATLike3.0 采用了基本的随机局部搜索框架，足够简洁；集成了 BMS、UPDeci 等优秀的机制，且引入了 *hscore*、*sscore* 等概念，方便扩展。因此本文决定以 SATLike3.0 为基础，探索 SLS 有前途的改进方向。

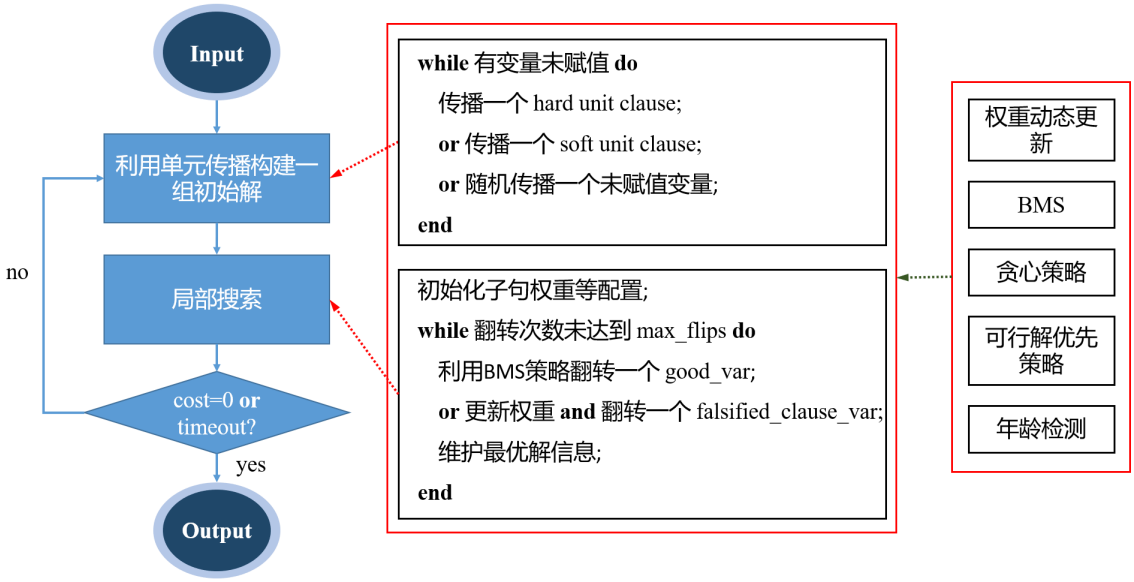

图 3.2　SATLike3.0 工作流程

3.7　小结

本节讨论了 SLS 有效的改进方向，其中对子句权重的调整、对变量重要性的评估和搜索范围的扩大是更为关键的，对提升 SLS 总体求解性能起着至关重要的作用。

本文决定从子句权重和变量评估的方向出发改进 SATLike3.0。可以依靠的结论主要有以下几点：

- Dist、SATLike 等先进的 SLS 中都体现了硬子句优先的思想；
- 研究表明在应用完备方法解决组合优化问题时可以用当前找到的最优解指导 SAT 引擎的状态保存机制[63–65]，因此开发参考局部最优解的反馈机制可能是有效的；
- VSIDS、LRB 和 LBD 等机制是现代 SAT 求解器中非常重要的子句或变量评价机制[22, 23]，它们都是基于子句矛盾进行统计和计算得来的，因此利用好字句间的矛盾一定对求解 (W)PMS 问题有所帮助；
- 研究表明骨干文字占比高的子句十分有利于求解 SAT 问题[29]。

第 4 章　优先随机传播硬变量的初始解生成过程：HFCRP-F

4.1　研究思路

UPDeci 依赖单元传播技术，因此需要监控单元子句的产生。单元子句源于问题实例原有的子句结构，以及 UPDeci 对子句长度的不断缩减。在当前不存在单元子句时，UPDeci 会随机为一个未赋值的变量赋值。对于某些子句长度很长的实例来说，这个过程会频繁执行，因此可以添加启发式方法以快速达到高质量的局部最优。此外，参考局部最优解进行赋值的方法在求解 SAT 相关问题的应用中已被证实有效，因此可以从局部最优解中参考变量的赋值信息。

削减随机性，优先传播硬变量。 UPDeci 存在随机赋值变量的过程，该现象在面对具有大量长子句的问题实例时尤为明显，这会为初始解提供多样性。然而局部搜索过程拥有一定的纠错能力，且算法的执行时间往往是受限的，因此希望可以更快地搜索到高质量的可行解并及时对其进行优化：在不存在单元子句时，强制使算法优先传播未满足的硬子句中的变量。这样既保证了硬子句中的变量较软子句中的变量有更高的优先级，又使得初始解的多样性得到了保证，即当硬子句全部被满足时退化为原策略。

参考当前局部最优解。在传播硬单元子句时如果遇到矛盾，UPDeci 会检查 *sscore*(*v*) 是否为非正数，是则会随机为变量赋值。这里可以加入参考当前的局部最优解的策略, 将此局部最优解中的值赋给变量。这种方法使算法具备一定的学习能力，将该方法的作用范围进行限制又不至于大幅削减初始解的多样性。

为“高质量”子句增加权重。 SATLike3.0 的变量评分机制依赖于子句权重的差别，这将影响算法的初始解生成过程 (UPDeci) 和局部搜索过程。考虑到算法每次都基于不同的初始情况进行搜索，子句的权重理应被丢弃。然而算法仍然可以从已知的局部最优解中提取信息：总是被满足的软子句有更大概率出现在最优解中（可能包含骨干文字），因此可以增加它们的初始权重，以引导算法倾向于优先满足这些“高质量”的子句。

4.2 具体实现

根据上文针对 UPDeci 的改进思路，一方面随机传播尚未满足的硬子句中的未赋值变量，另一方面引入反馈机制，得到的初始解生成算法名为 UPDeci-HFCRP-F。整个动态局部搜索算法名为 HFCRP-F (Hard Falsified Clauses Random Propagation with Feedback)。

初始解生成算法 UPDeci-HFCRP-F。如算法 4.1所示，在传播硬子句遇到矛盾的情况下会首先检查 *sscore*(*v*)，如果小于等于 0 则会参考局部最优解进行赋值；并且在随机选择变量赋值之前插入了优先传播硬变量的过程：先选择一个未满足的硬子句，再选择该子句中一个未赋值的文字，赋予其真值。

初始权重配置。额外开辟一个子句初始权重数组，每当搜索过程中找到一个更优解，则为每个当前解中被满足的软子句初始额外权重加 1。这个操作会影响初始解生成时变量的选值和局部搜索过程初期的决策，进而对局部搜索过程产生深远的影响。

4.3 对比实验

4.3.1 实验设置

数据集。在 MSE 2018-2021 不完备赛道中的 PMS 和 WPMS 实例上测试 HFCRP-F 以及原始算法 SATLike3.0。将对应的数据集记为 pms2018、wpms2018 等。需要注意的是，本部分工作将至少包含一个硬子句的问题实例视作 (W)PMS 实例。

算法实现。 HFCRP-F 使用 C++ 编写，使用 g++ 工具“-O3”参数编译，与 SATLike3.0 一致。

实验环境。所有实验在 Windows 10 的 WSL 子系统运行 (Ubuntu 20.04)，计算机配置为 AMD Ryzen 7 4800U @ 1.80 GHz。

对比指标。两个求解器在每个实例上分别运行 300 秒，记录其找到的最优解信息（最低 *cost*、对应时间）。对于每个数据集，统计两个求解器分别的获胜次数

算法 4.1 UPDeci-HFCRP-F

input : (W)PMS instance F
output: A complete assignment of variables in F

```
while ∃ unassigned variables do
    if ∃ hard unit clauses then
        c := pick a hard unit clause;
        if c has a contradictory hard unit clause then
            x := the related variable;
            assign x according to last best solution, simplify F accordingly;
        else
            perform unit propagation using c;
        end
    else if ∃ soft unit clauses then
        c := pick a soft unit clause with highest sscore using BMS startegy;
        if c has a contradictory soft unit clause then
            x := the related variable;
            assign x randomly, simplify F accordingly;
        else
            perform unit propagation using c;
        end
    else if ∃ falsified hard clauses then
        c := randomly pick a falsified hard clause;
        x := the first unassigned variable in c;
        assign x according to polarity in c, simplify F accordingly;
    else
        x := randomly pick an unassigned variable;
        randomly assign x, simplify F accordingly;
    end
    return the resulting assignment;
end
```

(找到比对方更低的 *cost*) 和求出最终解的平均时间（单位为秒），记为“#win”和“time(s)”；每个数据集包含的实例个数记作“#inst”；使用 MSE 2019 比赛规则中的不完备分数比较两个求解器的求解质量，记作“score”。如果求解器在一个实例上没有找到可行解，则将 time 记为 600 秒、score 记为 0。

- incomplete score = $\sum_i \left(\frac{\text{cost of best solution for i found by any solver+1}}{\text{cost of solution for i found by solver+1}}\right)$, $i \in$ instances.
- For each instance we consider the best solution found by the two incomplete solvers within 300 seconds.
- For an instance i, incomplete score is 0 if no solution was found by that solver.

4.3.2 实验结果与分析

如表 4.1所示，HFCRP-F 在所有数据集的 score 都优于 SATLike3.0；HFCRP-F 在 pms2021 的获胜次数略低于 SATLike3.0，而在其余的数据集上占据优势，尤其在 wpms2019 和 wpms2020 上优势明显。综上，HFCRP-F 在解的质量方面取得了显著进步。

更多的实验细节如图 4.1、图4.2、图 4.3和图 4.4所示。纵轴表示 HFCRP-F 找到更低 *cost* 的频数，横轴表示不同的问题实例类别。PMS 和 WPMS 实例的统计结果分别用蓝色和黄色表示。结果表明 HFCRP-F 和 SATLike3.0 各有自己擅长求解的问题，HFCRP-F 可以大幅优化 WPMS 问题解的质量。然而 HFCRP-F 在某些实例上无法求出可行解，这可能是由于降低了初始解的多样性导致的。

表 4.1 Experimental result

benchmark	#inst	HFCRP-F				SATLike3.0			
		#solved	#win	time(s)	score	#solved	#win	time(s)	score
pms2018	133	**89**	**23**	**256.32**	**0.66**	85	17	274.66	0.63
wpms2018	141	117	**54**	213.83	**0.97**	**118**	51	**196.06**	0.80
pms2019	251	**171**	**39**	**238.56**	**0.67**	166	35	250.61	0.65
wpms2019	253	211	**105**	197.43	**0.98**	**212**	69	**194.76**	0.79
pms2020	223	**145**	**34**	259.11	**0.65**	**145**	30	**256.61**	0.64
wpms2020	233	198	**102**	199.24	**0.98**	**199**	72	**185.15**	0.79
pms2021	124	**65**	16	**327.79**	**0.52**	**65**	**17**	328.75	**0.52**
wpms2021	139	110	**55**	237.78	**0.97**	**111**	52	**216.62**	0.77

4.4 主要改进机制工作流程示例

本节将演示仅为 UPDeci 添加优先传播硬变量机制的初始解构造过程，将该过程记作“UPDeci-HRP”。

图 4.5逐步展示了两种方法构造初始解的过程。命题公式用“*F*”表示，硬子句用“*H*”表示，软子句用“*S*”表示，括号内的数字代表文字。其中，□ 表

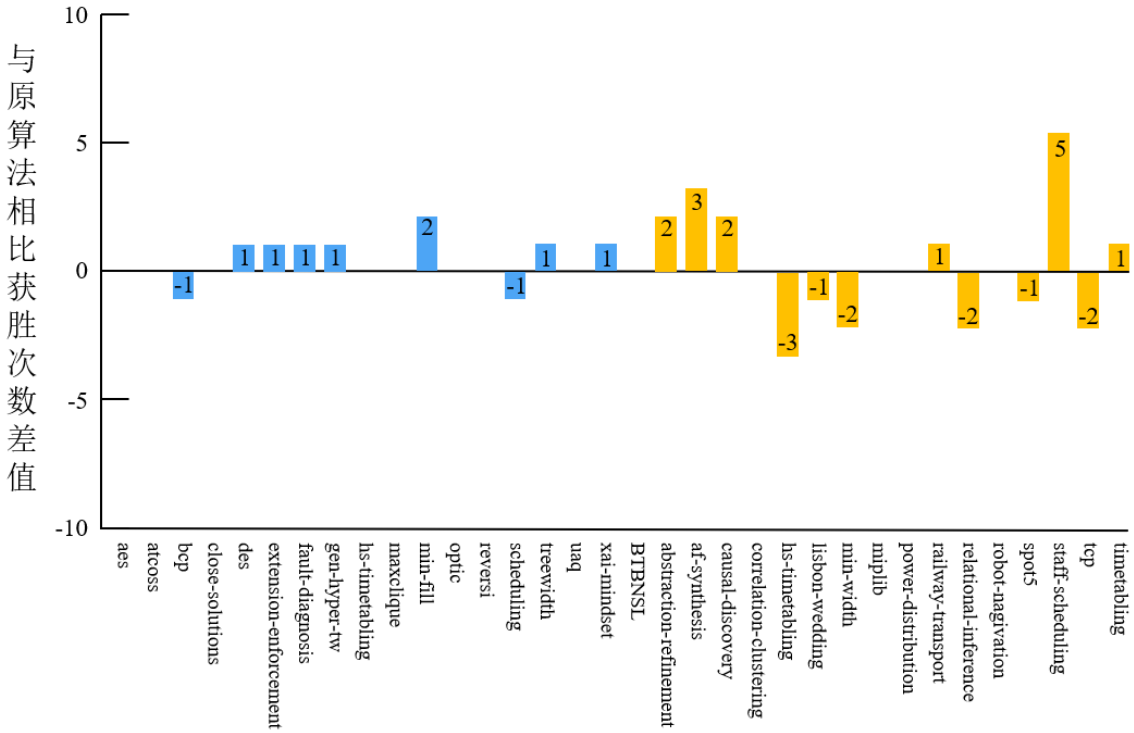

图 4.1　(w)pms2018 实验结果

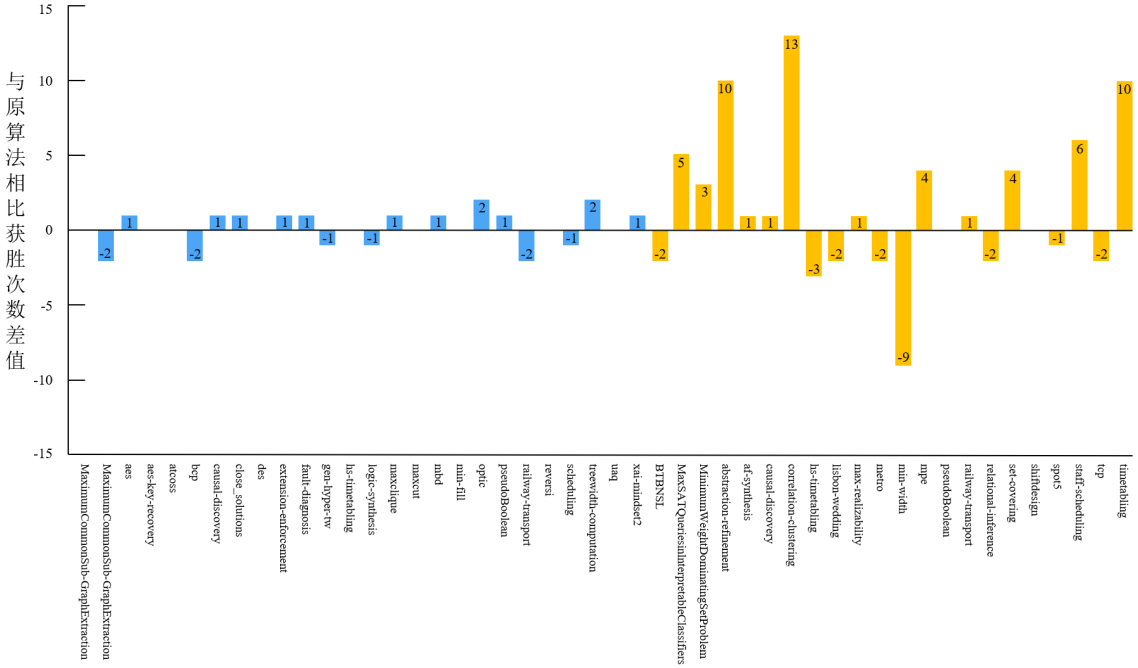

图 4.2　(w)pms2019 实验结果

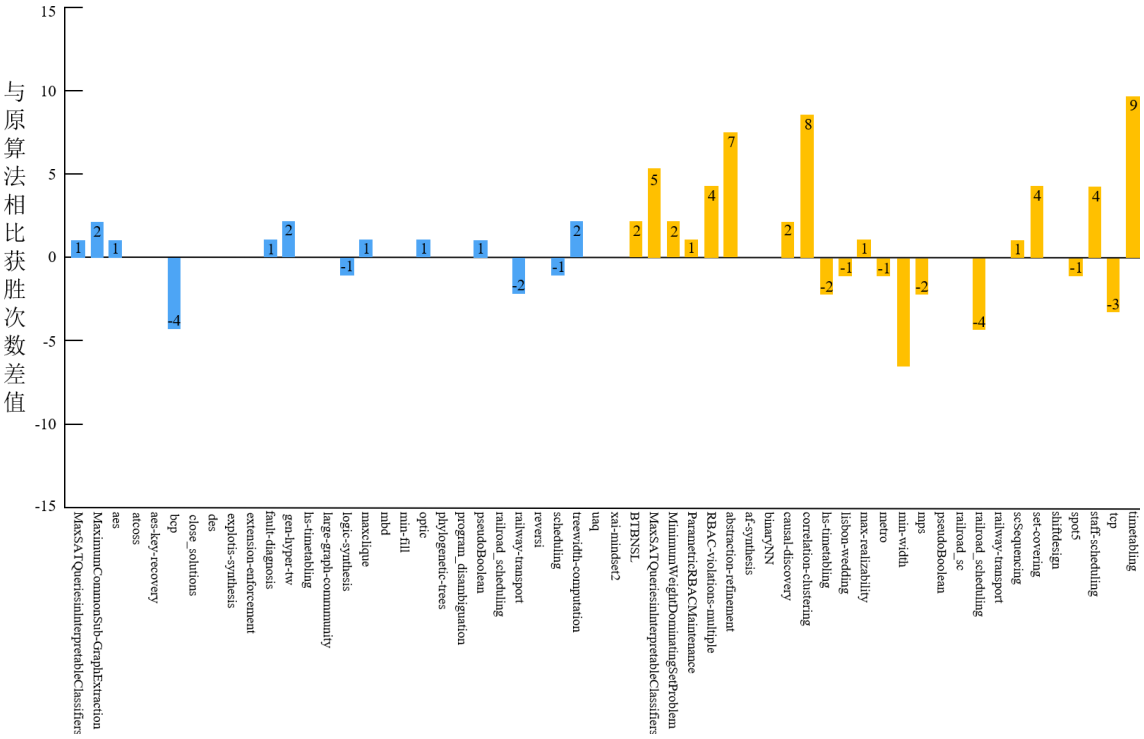

图 4.3　(w)pms2020 实验结果

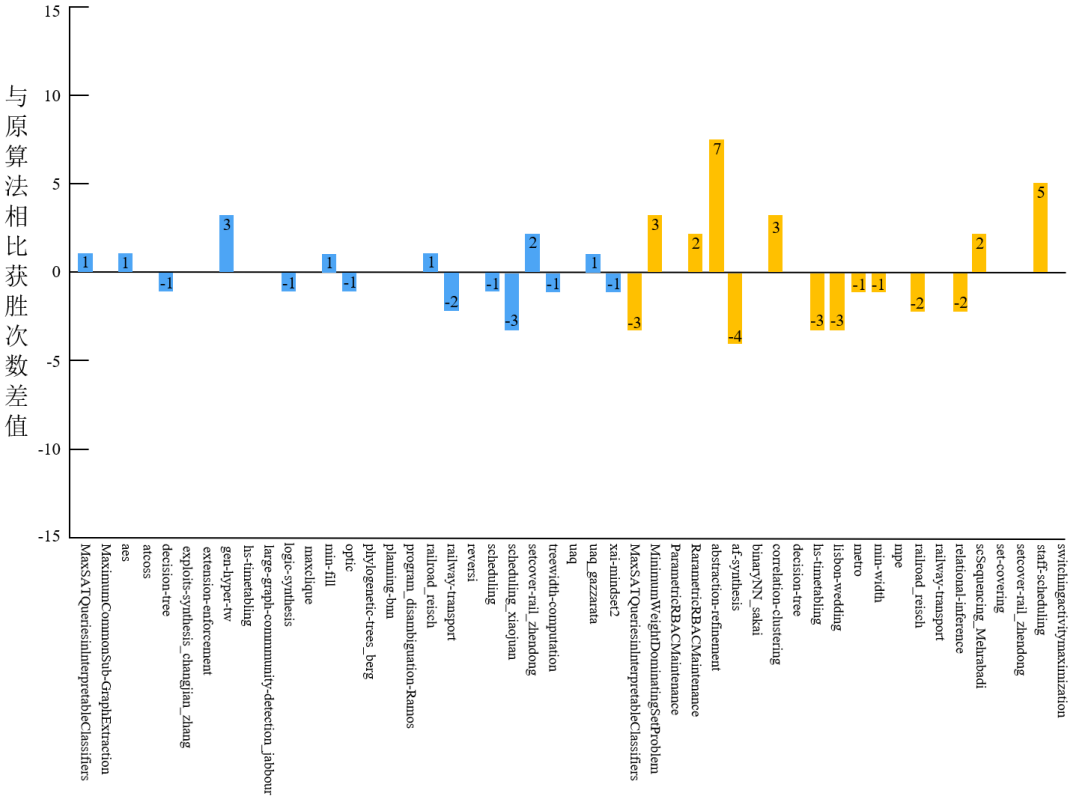

图 4.4　(w)pms2021 实验结果

示空子句，序号的下标表示执行的动作，上标表示操作的对象。例如，$①^{-5}_{hup}$ 表示当前动作为第一步，根据“*hup*”规则传播变量 5，其值为假（以下简称传播“-5”）。“*hup*”(Hard Unit Propagation) 表示传播硬单元子句；“*sup*”(Soft Unit Propagation) 表示传播软单元子句；“*ran*”(Random) 表示随机选择一个未赋值的变量，随机赋值传播。需要注意的是，动作“*ran*”的结果并不固定，图中只是列出了一种可能的计算结果。下面以硬子句 $\{(-2 \vee 4 \vee 5) \wedge (-5) \wedge (-4 \vee 5 \vee -2)\}$, 软子句 $\{(1 \vee 2) \wedge (-2 \vee 3)\}$ 的命题公式为例，说明 UPDeci 与 UPDeci-HRP 工作流程上的差异。

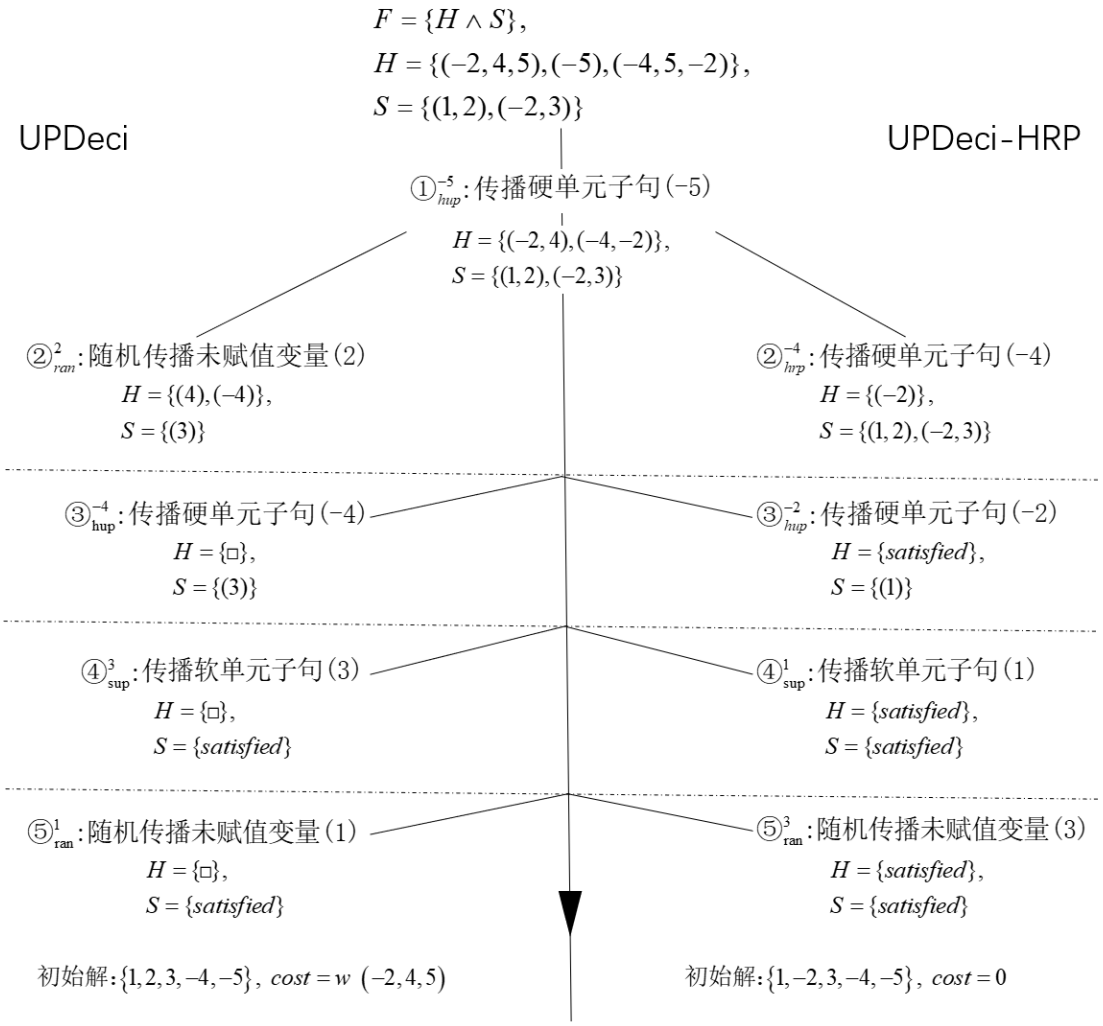

图 4.5　UPDeci 与 UPDeci-HRP 工作流程对比

UPDeci 优先传播硬单元子句，其次是软单元子句，再次是未赋值的变量。因此对于图 4.5中的公式，UPDeci 首先传播“−5”(①)；此时没有单元子句，因此随

机传播一个未赋值变量，假设传播“2”(②)；随后在传播“4”时遇到了矛盾，由于 *sscore*(4) 不为正数，因此随机赋值, 假设赋值为假 (③,④)；最后传播软单元子句“3”并随机为变量“1”赋值 (⑤,⑥)。最终，UPDeci 生成的初始解为 $\{1,2,3,4,5\}$，该解的 *cost* 值为硬子句 $\{(-2 \vee 4 \vee 5)\}$ 的权重。

同样对于该实例，当不存在单元子句时，UPDeci-HRP 优先传播一个未赋值的硬变量 (②)，接下来仅需确定性地执行两轮单元传播即构造了一组 *cost* 值为 0 的可行解。

4.5 改进思路的进一步理解

SATLike3.0 的搜索过程。每个未满足的子句都“有意愿”使自己满足，表现为将自身权重作为 *score* 或 *sscore* 贡献给其中的变量，结果是未满足的子句争夺变量的“赋值权”，其目的是促使算法为其中的变量翻转极性；已满足的子句也会捍卫变量的极性，表现为用自身权重降低变量的 *score* 或 *sscore*。值得注意的是，已满足的子句只会在其中只有一个满足的文字时才会阻止改变这个文字的值，因为一旦将这个文字变为假，那么子句本身将从满足变为不满足状态。

只在遇到矛盾的硬单元子句时参考局部最优解。初始解的多样性是至关重要的。尽管搜索过程具有一定的“纠错”能力，但如果总是围绕相似的初始解搜索，则必然会导致性能降级。此外这个过程可以与 HFCRP 配合：HFCRP 优先传播未赋值的硬变量，如果它带来了好的决策，那么就很有希望将结果留在搜索到的局部最优解中，因此在处理矛盾时参考局部最优解可能是一个好的选择。

只调节软子句权重。软子句的权重有上限，因此为了使整个公式被满足，硬子句的权重会在连续的调整中增长到足够大的数值，以至于只在意硬子句的满足。因此本文没有关注硬子句的权重配置，这也是没有从历史搜索的结果中提取软子句权重的一部分原因：许多软子句的权重达到了上限，无法反映出子句权重的差异。

对于搜索第一个可行解的影响。反馈机制依赖于第一个可行解，因此此时只有 HFCRP 生效。HFCRP 在初始解生成阶段尝试满足硬子句，然而它缩小了初始解的空间，并且影响了翻转变量的得分，因此可能对找第一个可行解有害。不过

从实验结果来看，HFCRP-F 总体受益于该机制。

综上，本部分工作主要的贡献是结合了优先传播硬变量和从局部最优解提取信息两个方面，更加确定性地处理了 SATLike3.0 中严格执行随机策略的部分。

4.6 小结

向大规模的随机领域添加合适的启发式策略可以快速引导算法达到局部最优，但也增加了计算复杂度，进而可能增加求解时间。相比原算法，修改后的算法在 WPMS 上取得了成功，在 PMS 上擅长处理的问题族却与原算法有较大差别，然而本章并没有探讨出这些差异的来源，原因在于通过强制改变算法流程的手段粒度较大，尝试只通过调整权重达成类似的效果或许是更好的解决方案。

第 5 章　基于子句矛盾的初始权重配置方法：SATLC

5.1　研究思路

分析子句矛盾。SLS 一项主要的问题是难以找到第一组可行解，SATLike3.0 的解决方案是在搜索第一组解时仅关注硬子句，将所有软子句的权重初始化为 0，以此缓和对于某些实例无法找到可行解的问题[28]。这种方式等同于解决 (W)PMS 实例所有硬子句组成的 SAT 问题，而子句矛盾的分析对于求解 SAT 问题是至关重要的，因此决定将子句矛盾的分析纳入子句权重的初始化。

为了减小参数设置对实验结果的影响，本部分工作仅在 PMS 实例上完成。由于 PMS 问题仅要求满足所有硬子句，而软子句被满足的数量是无法事先确定的，因此仅优化 UPDeci 处理硬子句的过程。UPDeci 的传播过程会持续缩短硬子句的长度，最终成为硬单元子句；在为单元子句中的唯一变量赋值时，如果发现变量已有值，即说明遭遇了矛盾。因此，仅处理 UPDeci 传播硬单元子句的过程即可捕捉硬子句间的矛盾信息，这极大程度上降低了分析矛盾的难度。

提取高质量子句。含有大量骨干文字（总是为真）的子句对求解 PMS 问题有积极作用，这些子句被称为高质量子句。由于这类子句总是被满足，因此可以通过统计的方式粗略地识别出可能的高质量子句。

5.2　具体实现

统计硬子句间矛盾信息的初始解构造过程 UPDeci-C。每当 UPDeci 传播硬单元子句遇到矛盾时，就向由该子句中唯一剩余的变量连接的邻居子句增加额外权重。方便起见将权重的增量设置为 1。总体思路如图 5.1所示。假设变量 v_1 连接了三个硬子句和一个软子句，且在传播 v_1 时硬单元子句 huc_1 和 huc_2 相互矛盾，那么由 v_1 连接的子句 huc_1、huc_2、hc_3、sc_1 都会获得 1 个单位的额外权重。本文将这种分析子句矛盾的初始解构造过程称作 UPDeci-C。

通过统计子句满足情况提取高质量子句。本文提出一种简单的方法寻找高质量子句：在执行完一轮局部搜索后，为当前局部最优解中被满足的子句增加额外

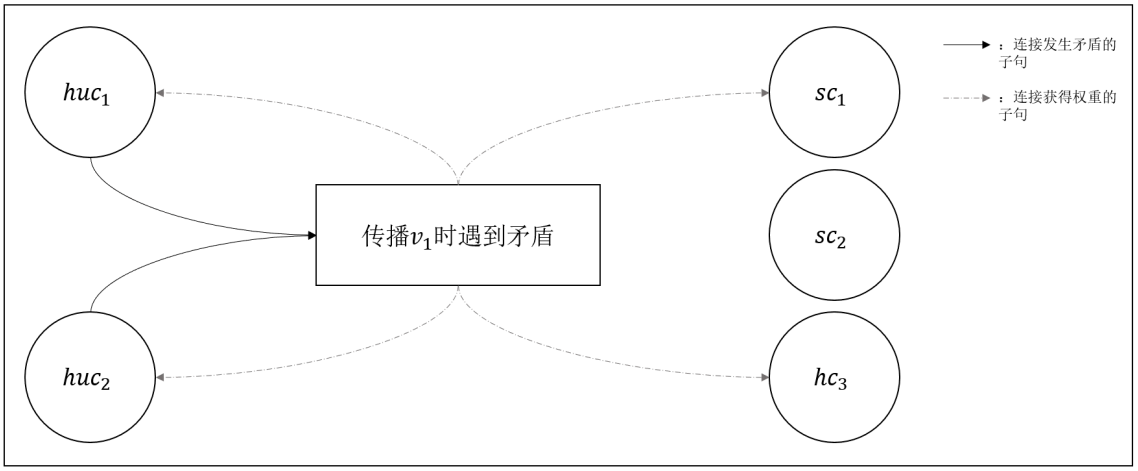

图 5.1　UPDeci-C：利用硬子句间的矛盾

1 个单位的权重。这个方法会使总被满足的子句获得更多额外权重，从而引导算法优先满足其中的文字。

本部分工作组合了上述两个改进方案。为了使方案更适配 SATLike3.0，还在算法的实现上采取如下策略：

延迟生效额外权重。为了不破坏原有的子句权重信息、避免实时额外权重信息误导当前的搜索过程，额外开辟容器来记录每个子句的额外重量，并使当前过程（生成初始解过程或局部搜索过程）收集的信息在 SATLike3.0 搜索过程的下一阶段生效。

限制额外权重生效的范围。由于可行解要求满足所有的硬子句，那么每一轮局部搜索完成后都会增长大部分硬子句的额外权重，因此决定在算法找到第一组可行解之后便不再为硬子句应用额外权重。

最终得到的算法记作 SATLC (SATLike3.0 Based On Confilcts)，其初始解生成算法 UPDeci-C 如算法 5.1所示。

5.3　对比实验

5.3.1　实验设置

数据集。在 MSE 2019-2021 不完备赛道中的 PMS 实例上测试 SATLC 以及原始算法 SATLike3.0，将对应的数据集记为 pms2019 等。

算法实现。 SATLC 使用 C++ 编写，使用 g++ 工具“-O3”参数编译，与

算法 5.1 UPDeci-C

input : (W)PMS instance F
output: A complete assignment of variables in F

figure out *hscore* and *sscore* of variables according to original weights and extra weights;
while ∃ unassigned variables **do**
  if ∃ hard unit clauses **then**
    c := pick a hard unit clause;
    if c has a contradictory hard unit clause **then**
      x := the related variable;
      if $sscore(x) > 0$ **then**
        propagate x with the sense achieves $sscore(x)$;
      else
        propagate x with the sense in the last best solution;
      end
      add extra weights to neighbors connected by x;
    else
      perform unit propagation using c;
    end
  else if ∃ soft unit clauses **then**
    c := pick a soft unit clause with highest *sscore* using BMS startegy;
    if c has a contradictory soft unit clause **then**
      propagate c with a random value;
    else
      perform unit propagation using c;
    end
  else
    randomly propagate an unassigned variable;
  end
  return the resulting assignment;
end

SATLike3.0 一致。

实验环境。所有实验在 Windows 10 的 WSL 子系统运行 (Ubuntu 20.04)，计算机配置为 AMD Ryzen 7 4800U @ 1.80 GHz。

对比指标。两个求解器在每个实例上分别运行 300 秒，记录其找到的最优解信息）最低 *cost*、对应时间）。对于每个数据集，统计两个求解器分别的获胜次数(找到比对方更低的 *cost*）和求出最终解的平均时间（单位为秒），记为“#win”和“time(s)”；每个数据集包含的实例个数记作“#inst”；使用 MSE 2019 规则中的不完备分数比较两个求解器的求解质量，记作“score”。如果求解器在一个实例上没有找到可行解，则将 time 记为 600 秒、score 记为 0。

- incomplete score = $\sum_i \left(\frac{\text{cost of best solution for i found by any solver+1}}{\text{cost of solution for i found by solver+1}}\right)$, $i \in$ instances.

- For each instance we consider the best solution found by the two incomplete solvers within 300 seconds.
- For an instance *i*, incomplete score is 0 if no solution was found by that solver.

5.3.2 实验结果与分析

表 5.1展示了 SATLC 和 SATLike3.0 在 300 秒内解决 PMS 实例的情况。参与比较的标准数据集为 pms2019、pms2020 和 pms2021。实验结果表明 SATLC 在解决问题的数量和求得可行解的质量上都明显优于 SATLike3.0。

需要注意的是在本次实验中并没有区别 PMS 和纯粹的 MaxSAT 实例：PMS 实例包含硬子句和软子句，而 MaxSAT 实例可以看作为仅包含软子句的 PMS 实例。纯 MaxSAT 实例的问题族主要包括“SeanSafarpour”、“maxclique”、“max-cut”、“ramsey”、“scpcyc”和“set-covering”。

更加详细的实验结果在表 5.2、表 5.3和表 5.4中呈现。用“#winDiff”标明 SATLC 和 SATLike3.0 在每个问题族上的获胜次数差值。可以明显发现 SATLC 在“maxclique”和“maxcut”问题族上表现明显劣于 SATLike3.0，这可能是由于 UPDeci-C 在纯 MaxSAT 实例无法完全生效导致的。

表 5.1 Experimental result

benchmark	#inst	SATLG				SATLike3.0			
		#solved	#win	time(s)	score	#solved	#win	time(s)	score
pms2019	299	**244**	**80**	**176.82**	**0.81**	214	53	218.30	0.70
pms2020	262	**206**	**70**	**191.55**	**0.78**	184	36	226.04	0.69
pms2021	155	**116**	**40**	**210.87**	**0.74**	96	29	267.67	0.62

5.4 小结

利用子句矛盾的机制早已在基于 SAT 的求解器中取得成功，如 VSIDS、LRB 和 LBD 都是现代 SAT 求解器的关键技术手段。本章工作中最主要的贡献是将利用子句矛盾的思想迁移到一个先进的动态局部搜索求解器中，并使用了轻量高效

表 5.2　SATLC 与 SATLike3.0 在 pms2019 上的获胜次数差值

instance family	#inst	win differential
MaxSATQueriesinInterpretableClassifiers	15	4
MaximumCommonSub-GraphExtraction	15	0
SeanSafarpour	13	7
aes	5	2
aes-key-recovery	5	0
atcoss	14	1
bcp	8	-1
causal-discovery	3	0
close_solutions	14	1
des	12	0
extension-enforcement	15	-2
fault-diagnosis	8	8
gen-hyper-tw	18	-2
hs-timetabling	1	1
logic-synthesis	1	1
maxclique	17	-4
maxcut	12	-8
mbd	6	1
min-fill	15	5
optic	16	6
pseudoBoolean	6	1
railway-transport	4	-1
ramsey	14	0
reversi	11	0
scheduling	5	-1
set-covering	9	2
treewidth-computation	8	1
uaq	15	0
xai-mindset2	14	5
sum	299	27

表 5.3　SATLC 与 SATLike3.0 在 pms2020 上的获胜次数差值

instance family	#inst	win differential
MaxSATQueriesinInterpretableClassifiers	19	0
MaximumCommonSub-GraphExtraction	11	2
SeanSafarpour	10	6
aes	5	2
aes-key-recovery	3	0
atcoss	14	1
bcp	14	-2
close_solutions	4	1
des	3	0
exploits-synthesis	2	0
extension-enforcement	12	-2
fault-diagnosis	4	4
gen-hyper-tw	8	-2
hs-timetabling	1	1
large-graph-commmunity	3	-1
logic-synthesis	1	1
maxclique	11	-1
maxcut	9	-4
mbd	5	0
min-fill	12	6
optic	15	7
phylogenetic-trees	5	0
program_disambiguation	2	0
pseudoBoolean	6	1
railroad_scheduling	4	4
railway-transport	4	-1
ramsey	11	0
reversi	11	0
scheduling	3	0
set-covering	9	2
treewidth-computation	8	2
uaq	18	0
xai-mindset2	15	7
sum	262	34

表 5.4　SATLC 与 SATLike3.0 在 pms2021 上的获胜次数差值

instance family	#inst	win differential
MaxSATQueriesinInterpretableClassifiers	1	-1
MaximumCommonSub-GraphExtraction	1	0
aes	6	2
atcoss	7	1
decision-tree	23	14
exploits-synthesis_changjian_zhang	2	0
extension-enforcement	2	-1
gen-hyper-tw	8	-1
hs-timetabling	1	1
large-graph-commmunity-detection_jabbour	3	-1
logic-synthesis	1	1
maxclique	3	0
maxcut	11	-11
min-fill	7	4
optic	1	-1
phylogenetic-trees_berg	8	0
planning-bnn	8	0
program_disambiguation-Ramos	1	0
railroad_reisch	8	4
railway-transport	4	-1
ramsey	11	0
reversi	7	0
scheduling	3	0
scheduling_xiaojuan	8	-3
set-covering	9	2
setcover-rail_zhendong	4	0
treewidth-computation	3	0
uaq	1	0
uaq_gazzarata	1	1
xai-mindset2	2	1
sum	155	11

的实现方式。实验结果证明这种利用矛盾的机制帮助原算法解决了更多 PMS 实例，而且 SATLC 通常能找到比原算法更优的可行解。这种利用矛盾的策略具有通用性，也许可以应用到广泛的预处理过程中[66]。

SATLC 目前最明显的问题是参数设置过于简单：权重调整的幅度均为 1，因此没有在 WPMS 实例上进行测试。

第 6 章　提取潜在关键子句：HRP-C

6.1　研究思路

在第 4章和第 5章中分别提升了 SATLike3.0 在 WPMS 和 PMS 实例上的表现，但如何将使用的技术结合仍需考虑。

UPDeci-HFCRP-F 需要维护硬子句的信息以得知其中未满足子句的变量赋值情况，因此需要额外进行较为复杂的计算。考虑 UPDeci 随机传播未赋值变量的情况，由于变量未赋值，其 *hscore* 一定为正（存在包含该变量的未满足硬子句）或零（不存在包含该变量的未满足硬子句），因此传播 *hscore* 为正数的未赋值变量在逻辑上与 UPDeci-HFCRP-F 的区别仅在于可能先选到不在任何未满足硬子句中的变量。如果对未赋值变量进行随机采样并选取其中合适的变量则可以有效增大选到 *hscore* 为正数的未赋值变量的概率，从而将 UPDeci-HFCRP-F 额外引入的时间复杂度降低 $O(N)$。此外，反馈局部最优解时应考虑当前解的质量，在算法执行早期获取的局部最优解的质量无法保证，不应予以采用。

SATLC 统计子句间的矛盾信息用于调整子句的初始权重，对 SATLike3.0 在 PMS 实例上的表现提升明显。但调整的幅度与 WPMS 实例子句的原始权重相比微乎其微，理论上对 WPMS 实例不会有明显影响，那么在 SATLC 的基础上集成 HFCRP-F 使用的技术即可结合两种算法各自的长处。

综上，将 HFCRP-F 和 SATLC 进行结合，组合后的求解器记为 HRP-C，算法流程如 6.1和 6.2所示。

6.2　具体实现

本节工作在 SATLC 的基础上进行开发，利用 BMS 的思想：在 UPDeci 算法随机传播未赋值变量时，最多随机采样 k 个未赋值变量（放回），如果其 *hscore* 大于 0 则传播该变量，否则随机赋值传播。设 S 为变量集合，当前状态下有 $\rho|S|$ 个 *hscore* 为 0 的变量，E = {采样k次得到变量的的*hscore*大于0}，其中 $\rho \in (0,1)$，那么 $Pr(E) = 1 - \rho^k$。MSE 竞赛的不完备算法执行时间通常以 60 秒和 300 秒为界，

在反馈机制中可加入算法执行 60 秒后才可生效的约束。

算法 6.1 UPDeci-C*

input : (W)PMS instance F
output: A complete assignment of variables in F

figure out *hscore* and *sscore* of variables according to original weights and extra weights;
while ∃ unassigned variables **do**
  if ∃ hard unit clauses **then**
    c := pick a hard unit clause;
    if c has a contradictory hard unit clause **then**
      x := the related variable;
      if $sscore(x) > 0$ **then**
        propagate x with the sense achieves $sscore(x)$;
      else if execution time reaches 60 seconds **then**
        propagate x with the sense in the last best solution;
      else
        assign x randomly, simplify F accordingly;
      end
      add extra weights to neighbors connected by x;
    else
      perform unit propagation using c;
    end
  else if ∃ soft unit clauses **then**
    c := pick a soft unit clause with highest *sscore* using BMS startegy;
    if c has a contradictory soft unit clause **then**
      propagate c with a random value;
    else
      perform unit propagation using c;
    end
  else
    v := a random unassigned variable;
    for iteration := 1 **to** k **do**
      v := a random unassigned variable;
      if $hscore(v) > 0$ **then**
        propagate v with the value achieves *hscore*;
        break;
      end
    end
    if $hscore(v) = 0$ **then**
      propagate v with a random value;
    end
  end
  return the resulting assignment;
end

6.3 实验设置与实验结果

数据集。在 MSE 2018-2021 不完备赛道中的 PMS 和 WPMS 实例上测试 HRP-C 以及原始算法 SATLike3.0，将对应的数据集记为 pms2018、wpms2018 等。

算法 6.2 HRP-C

input : (W)PMS instance F
output: A best found feasible assignment α and its cost, or "no feasible assignment found"

```
94  cost* := +∞;
95  α := ∅;
96  α* := ∅;
97  while elapsed time < cutoff time and try times < max tries do
98  |   α := an initial complete assignment by UPDeci-C*;
99  |   Initialization();
100 |   add extra weights to clauses;
101 |   while flip times < max non improve times do
102 |   |   if α is feasible and cost(α) < cost* then
103 |   |   |   α* = α;
104 |   |   |   cost* = cost(α)
105 |   |   end
106 |   |   if D := {x | score(x) > 0} ≠ ∅ then
107 |   |   |   x := a variable picked by BMS;
108 |   |   else
109 |   |   |   update clause weight;
110 |   |   |   if ∃ falsified hard clauses then
111 |   |   |   |   c := a random falsified hard clause;
112 |   |   |   else
113 |   |   |   |   c := a random falsified soft clause;
114 |   |   |   end
115 |   |   |   x := the variable with highest score in c;
116 |   |   end
117 |   |   α := α with x flipped;
118 |   end
119 |   add extra weights to satisfied clauses under α.
120 end
121 if α* is feasible then
122 |   return (cost*, α*);
123 else
124 |   return "no feasible assignment found";
125 end
```

算法实现。 HRP-C 使用 C++ 编写，使用 g++ 工具"-O3"参数编译，与 SATLike3.0 一致。

实验环境。所有实验在 Windows 10 的 WSL 子系统运行 (Ubuntu 20.04)，计算机配置为 AMD Ryzen 7 4800U @ 1.80 GHz。

对比指标。两个求解器在每个实例上分别运行 300 秒，记录其找到的最优解信息（最低 *cost*、对应时间）。对于每个数据集，统计两个求解器分别的获胜次数 (找到比对方更低的 *cost*) 和求出最终解的平均时间（单位为秒），记为"#win"和"time(s)"；每个数据集包含的实例个数记作"#inst"；使用 MSE 2019 规则中

的不完备分数比较两个求解器的求解质量，记作“*score*”。如果求解器在一个实例上没有找到可行解，则将 time 记为 600 秒、score 记为 0。

实验结果见表 6.1、图 6.1、图 6.2、图 6.3和图 6.4。可以发现 HRP-C 虽然在求解的过程中没有对实例是否加权做出严格区分，但继承了两种算法各自在处理 WPMS 和 PMS 实例的优势。同时也应注意到 HRP-C 在数据集 wpms2021 上解决的问题实例少于基准算法，这可能是由于对 UPDeci 的干涉导致缩小了初始解空间。

表 6.1　Experimental result

benchmark	#inst	HRP-C				SATLike3.0			
		#solved	#win	time(s)	score	#solved	#win	time(s)	score
pms2018	153	**121**	**42**	**190.26**	**0.78**	105	30	249.45	0.68
wpms2018	172	**149**	**68**	172.58	**0.98**	**149**	56	**171.93**	0.83
pms2019	299	**242**	**84**	**172.73**	**0.80**	214	52	218.30	0.70
wpms2019	297	**256**	**102**	178.38	**0.97**	**256**	94	**174.63**	0.82
pms2020	261	**203**	**67**	**194.64**	**0.77**	184	36	226.55	0.69
wpms2020	253	**220**	**103**	174.43	**0.98**	219	76	**171.07**	0.81
pms2021	155	**116**	**47**	**208.95**	**0.74**	96	28	267.67	0.61
wpms2021	151	122	**53**	202.53	**0.97**	**123**	52	**200.18**	0.78

6.4　实验结果分析

按照 HRP-C 和 SATLike3.0 在不同问题族中获胜次数差值的绝对值大于等于 5，且该数值占对应问题族实例数量一半以上为条件，选取出测试结果差别显著的数据集。包括“maxcut”、“min-fill”、“SeanSafarpour”、“fault-diagnosis”、“xai-mindset2”、“decision-tree”、“af-synthesis”、“correlation-clustering”、“staff-scheduling”、“abstraction-refinement”和“MaxSATQueriesinInterpretableClassifiers”，各数据集概况见表 6.2，展示了不同问题族的实例数量、两种算法获胜次数差值、平均变量数量、平均子句数量、平均硬子句数量、平均硬子句长度、平均软子句数量和平均软子句长度。根据 [67] 中整理的可能对描述问题难度有帮助的特征值计算得到表 6.3，其中 *c_v* 表示子句数量与变量数量的比值，*hc_v* 表示硬子句数量

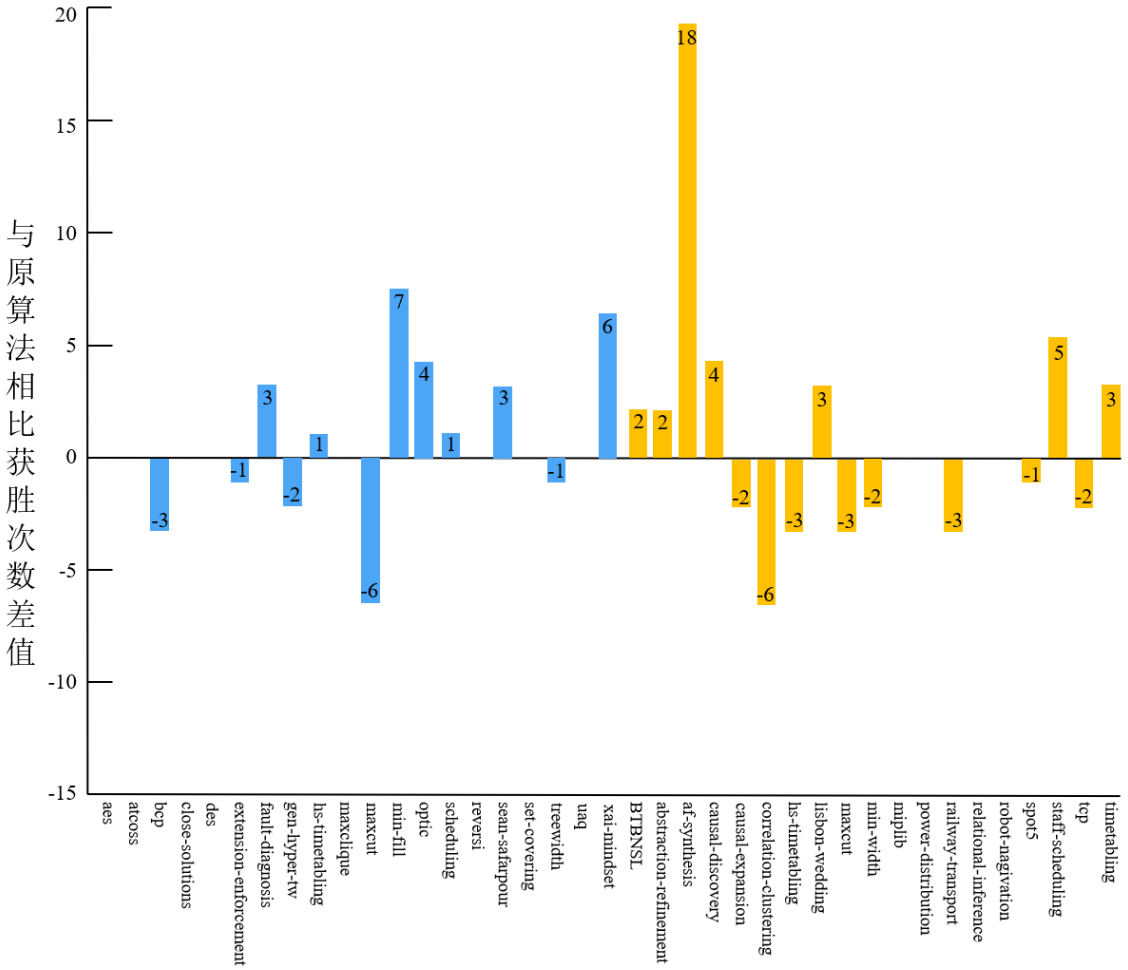

图 6.1　(w)pms2018 实验结果

与变量数量的比值，*hc_sc* 表示硬子句与软子句数量的比值，*hc_c* 表示硬子句对所有子句的占比，*c_len* 表示子句的平均长度，以上所有均取所属问题族全部实例的平均值。另外计算了子句平均长度与硬子句所占比例的乘积。

对于硬子句数量为 0 的实例，HRP-C 新提出的机制除一轮搜索后为所有满足的软子句增加权重外全部失效，这种情况下在 maxcut 问题族中表现出了明显的性能降级，却在 SeanSafarpour 问题族中实现了较为明显的性能提升。直觉上不断为已满足的子句增加额外权重会缩小解空间，因此 maxcut 问题实例的结构也许更为随机，需要扩大搜索范围或提供随机游走机制到达更“远”的邻域。可以支撑这个推断的依据为 Nudelman 等人[67] 通过实验证明 *c_v* 值可以用来预测随机生成的 3-SAT 问题难度，而 pms2018、pms2019 和 pms2021 中 maxcut 的 *c_v* 值分别为 19.07、29.18 和 9.46，远大于 pms2019 和 pms2020 中 SeanSafarpour 的 3.46 和

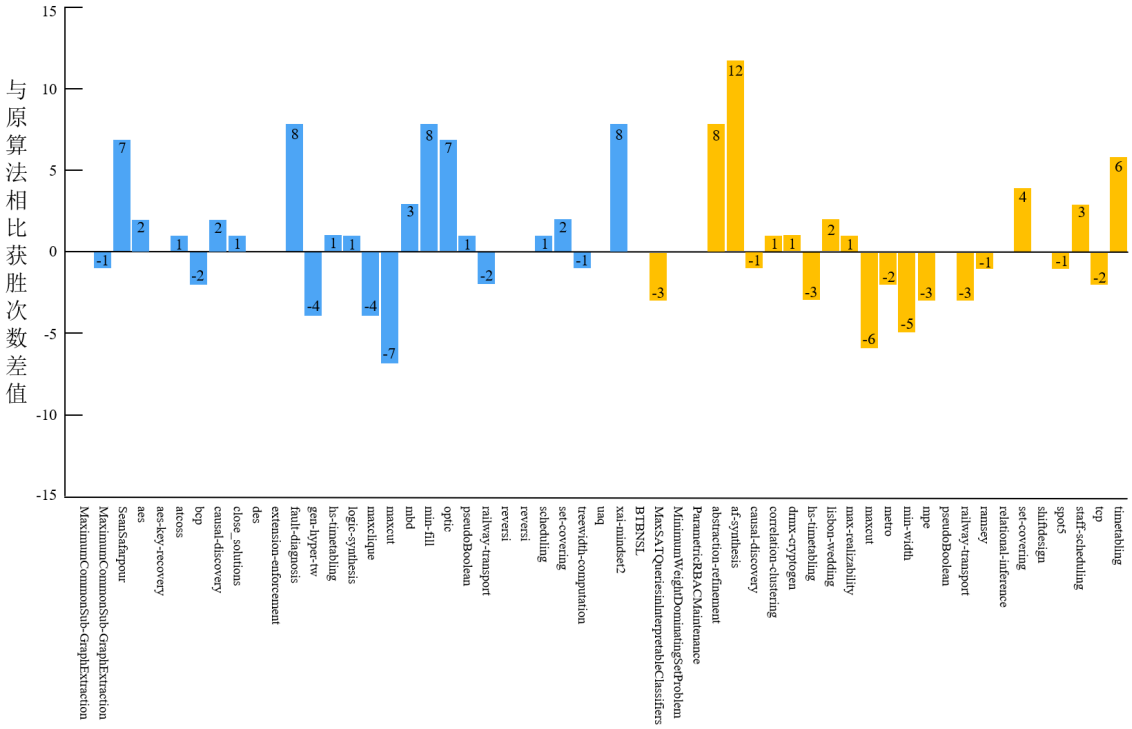

图 6.2　(w)pms2019 实验结果

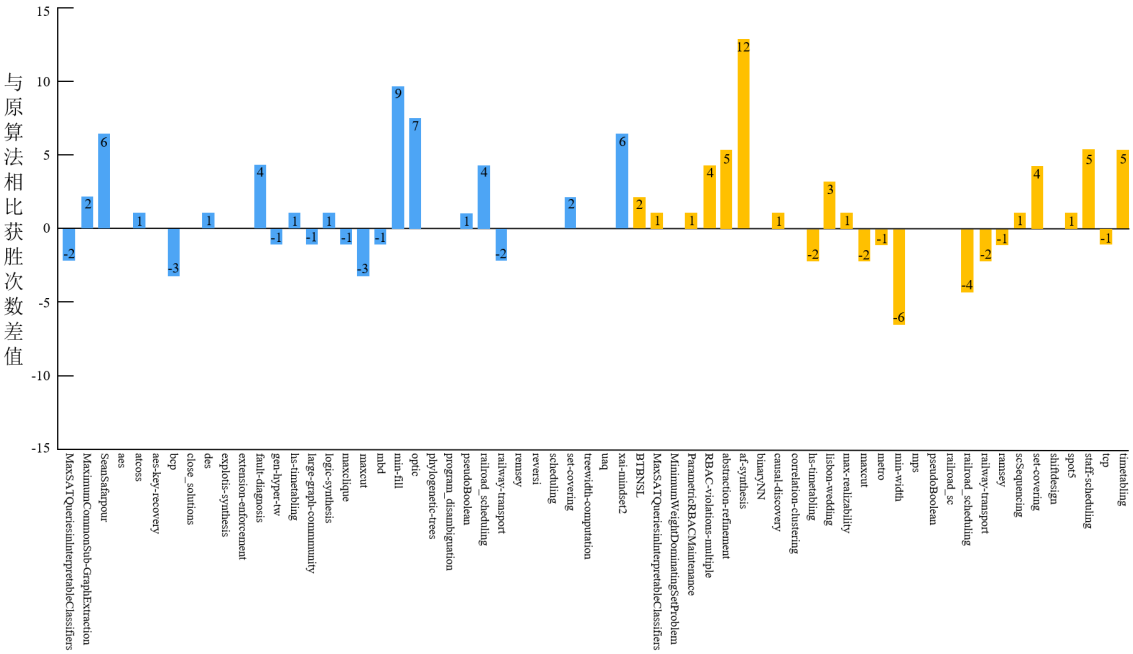

图 6.3　(w)pms2020 实验结果

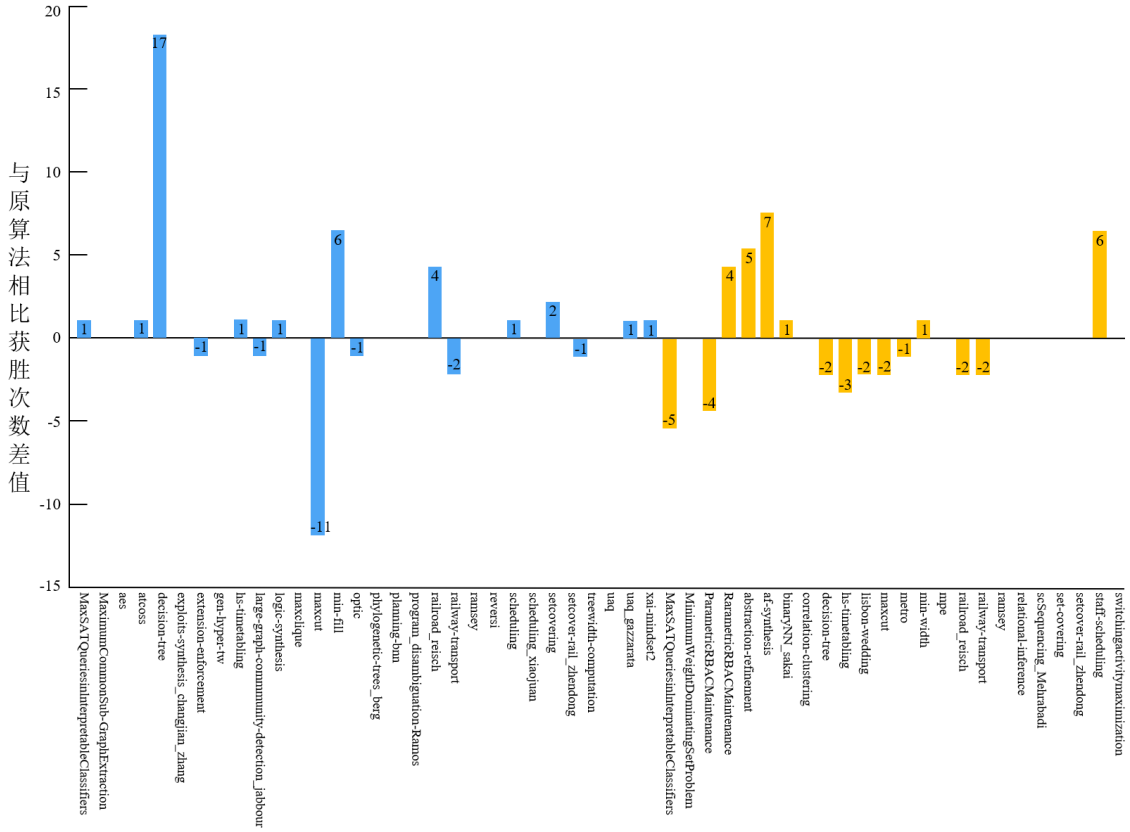

图 6.4　(w)pms2021 实验结果

3.38。进一步通过公式 6.1和公式 6.2计算了 maxcut 和 SeanSafarpour 的模块化程度 (*modularity*)[68] 的平均值。计算结果分别 0.173 和 0.959[1]，说明 maxcut 实例的结构性很弱，而 SeanSafarpour 的结构性很强，这直接验证了上述对 maxsat 实例结构的猜想，证明本文的方法可以帮助算法识别问题实例的结构性。

$$Q(G,P)=\sum_{P_i\in P}\frac{\sum_{x,y\in P_i}w(x,y)}{\sum_{x,y\in V}w(x,y)}-\left(\frac{\sum_{x\in P_i}deg(x)}{\sum_{x\in V}deg(x)}\right)^2. \tag{6.1}$$

$$Modularity=Q(G)=max\{Q(G,P)\,|\,P\}. \tag{6.2}$$

显然 *hc_c* 和 *hc_c* 的值越大则 UPDeci-C 越容易执行随机传播未满足硬子句中变量的过程：可以观察到 HRP-C 在表 6.3的 $c\_len\times hc\_c$ 值大于 3 的问题族中性能普遍优越。最为有趣的是 pms2021 数据集的 decision-tree 问题族，HFCRP-F 和 SATLC 各自都无法实现良好的效果（获胜差值分别为 0 和 -2），通过计算得其

[1]最理想的情况下 PMS 应满足所有子句，因此在去掉子句权重信息后将整个 PMS 实例视作 SAT 问题进行 *modularity* 的计算。由于计算资源有限，没有计算物理内存大于 100MB 的实例

modularity 值为 0.38[2]，结构性并不明显，但两者结合得到的 HRP-C 相比原算法却取得了绝对性的优势，这说明随机传播硬变量引起的子句矛盾使算法捕捉到了问题的结构信息，多种机制的相互影响使 HRP-C 在处理该问题族上取得了突破。

6.5 (W)PMS 问题相关应用

6.5.1 组合拍卖

组合拍卖问题是一种在现实商业活动中常见的组合优化问题。在这种问题中，有若干个拍卖品，每个商家都会对商品组合进行竞拍并给出价格。在每个商品只能卖给一个商家的前提下，需要求解卖家所能获得的最大销售额。

设有 $i = 10$ 个商品，对于商品集合 $Goods = \{g_i, i \in N_+, i < 10\}$，$k = 6$ 个商家分别给出的拍卖方案集合为 $Sales = \{[(g_1, g_2, g_3), 101], [(g_2, g_3, g_5, g_6), 150], [(g_1, g_7, g_8), 120], [(g_4, g_9, g_{10}), 100], [(g_1, g_5, g_9), 180], [(g_4, g_6, g_{10}), 200]\}$，其中 $[(g_1, g_2, g_3), 101]$ 表示第一个商家出价 101 元竞拍商品 1、2、3。为每个商家的竞拍方案分配一个变量 x_i，表示这个商家的竞拍结果，即当 $x_i = 1$ 时表示竞拍成功，$x_i = 0$ 时竞拍失败。由此得到变量集合 $V = \{x_1, x_2, x_3, x_4, x_5, x_6\}$。由于每个商品最多只能卖给一个商家，因此购买相同竞品的商家不能同时竞拍成功。比如第一个商家和第五个商家同时购买拍卖品 g_1，因此 x_1 和 x_5 不能同时为真。

拍卖事件之间的互斥是硬性条件，由此为所有竞拍相同商品的变量生成硬子句：

$$\forall m \in k, n \in k, m \neq n, (\overline{x_m} \vee \overline{x_n}). \tag{6.3}$$

卖家希望所有商家都竞拍成功以获取最大利益，由此以每组竞拍方案的价格作为权重生成软子句：

$$\forall j \in k, (x_j), w_j. \tag{6.4}$$

因此对于上述例子，可以生成硬子句集合 $H = \{\overline{x_1} \vee \overline{x_3}, \overline{x_1} \vee \overline{x_5}, \overline{x_1} \vee \overline{x_2}, \overline{x_2} \vee$

[2]同样没有计算占用内存过大的问题实例

表 6.2　显著问题族情况概览

benchmark	family	#inst	#winDiff	nvars	nclauses	nhards	hard_len	nsofts	soft_len
pms2018	maxcut	7	-6	70	1335	0	0	1335	2
	min-fill	8	7	78408	764123	761770	3.39	3548	0.89
pms2019	SeanSafarpour	13	7	1166651	4040390	0	0	4040390	2.47
	fault-diagnosis	8	8	153258	1091832	1042062	2.41	49770	0.89
	maxcut	12	-7	123	3589	2508	0.46	1081	1.62
	min-fill	15	8	119765	1339832	1334691	3.6	5141	0.94
	xai-mindset2	14	8	34046	2104709	2099716	3.72	4993	0.93
pms2020	SeanSafarpour	10	6	1015288	3426647	0	0	3426647	2.41
	min-fill	12	9	93145	1027796	1023482	3.53	4313	0.92
pms2021	decision-tree	23	17	264167	2224735	2223955	5.99	780	0.96
	maxcut	11	-11	131	1239	0	0	1239	1.83
	min-fill	7	6	70246	643418	640518	3.25	2900	0.88
wpms2018	af-synthesis	19	18	19823	283412	283233	3.3	178	0.95
	correlation-clustering	12	-6	128120	588096	575861	2.73	12235	0.92
	staff-scheduling	10	5	125996	507560	503530	2.23	4029	0.91
wpms2019	abstraction-refinement	10	8	7123990	16317592	16312207	2.5	5385	0.91
	af-synthesis	16	12	19279	269431	269260	3.27	171	0.94
wpms2020	abstraction-refinement	7	5	8260175	19000205	18994896	2.43	5309	0.88
	af-synthesis	17	12	19009	262312	262144	3.3	168	0.94
wpms2021	MaxSATQueriesinInterpretableClassifiers	5	-5	68537	3751062	3727945	2.88	23117	0.83
	abstraction-refinement	7	5	7919532	18431857	18426206	2.43	5651	0.88
	af-synthesis	9	7	17799	235249	235096	3.19	152	0.9
	staff-scheduling	10	6	169550	681109	676537	2.24	4572	0.91

表 6.3　显著问题族特征表

benchmark	family	#inst	#winDiff	c_v	hc_v	hc_sc	hc_c	c_len	c_len×hc_c
pms2018	maxcut	7	-6	19.07	0	0	0	2	0
	min-fill	8	7	9.75	9.72	214.7	1	3.37	3.37
pms2019	SeanSafarpour	13	7	3.46	0	0	0	2.47	0
	fault-diagnosis	8	8	7.12	6.8	20.94	0.95	2.34	2.22
	maxcut	12	-7	29.18	20.39	2.32	0.7	1.84	1.29
	min-fill	15	8	11.19	11.14	259.62	1	3.59	3.59
	xai-mindset2	14	8	61.82	61.67	420.53	1	3.68	3.68
pms2020	SeanSafarpour	10	6	3.38	0	0	0	2.41	0
	min-fill	12	9	11.03	10.99	237.3	1	3.51	3.51
pms2021	decision-tree	23	17	8.42	8.42	2851.22	1	5.97	5.97
	maxcut	11	-11	9.46	0	0	0	1.83	0
	min-fill	7	6	9.16	9.12	220.87	1	3.24	3.24
wpms2018	af-synthesis	19	18	14.3	14.29	1591.2	1	3.29	3.29
	correlation-clustering	12	-6	4.59	4.49	47.07	0.98	2.69	2.64
	staff-scheduling	10	5	4.03	4	124.98	0.99	2.21	2.19
wpms2019	abstraction-refinement	10	8	2.29	2.29	3029.19	1	2.5	2.5
	af-synthesis	16	12	13.98	13.97	1574.62	1	3.27	3.27
wpms2020	abstraction-refinement	7	5	2.3	2.3	3577.87	1	2.43	2.43
	af-synthesis	17	12	13.8	13.79	1560.38	1	3.3	3.3
wpms2021	MaxSATQueriesinInterpretableClassifiers	5	-5	54.73	54.39	161.26	0.99	2.79	2.76
	abstraction-refinement	7	5	2.33	2.33	3260.7	1	2.43	2.43
	af-synthesis	9	7	13.22	13.21	1546.68	1	3.18	3.18
	staff-scheduling	10	6	4.02	3.99	147.97	0.99	2.22	2.2

$\overline{x_5}, \overline{x_2} \vee \overline{x_6}, \overline{x_4} \vee \overline{x_5}, \overline{x_4} \vee \overline{x_6}\}$，软子句集合 $S = \{(x_1, 101), (x_2, 150), (x_3, 120), (x_4, 100), (x_5, 180), (x_6, 200)\}$。写为标准的 CNF 格式文件如图 6.5。图的第一行为说明性文字，三个数字分别表示变量的数量、子句的数量和硬子句的权重，硬子句的权重通常记作所有软子句权重之和加 1；以下每一行代表一个子句，第一列为该子句的权重，每行末尾的数字 0 表示结尾。

```
p wcnf 6 20 852
852 -1 -3 0
852 -1 -5 0
852 -1 -2 0
852 -2 -5 0
852 -2 -6 0
852 -4 -5 0
852 -4 -6 0
101 1 0
150 2 0
120 3 0
100 4 0
180 5 0
200 6 0
```

图 6.5　组合拍卖问题 CNF 格式文件

上述例子的求解结果如图 6.6所示，同意 x_3、x_5 和 x_6 的竞拍方案则卖方可以获得最大收益。

```
yuhanyi@yuhanyi:/mnt/d/IdeaProjects/what2eat$ ./bin/HRP-C-printsolu src/main/resources/whatsnew.wcnf
o 351
time limit:60s
351     0
time limit:300s
351     0
v -1 -2 3 -4 5 6
```

图 6.6　HRP-C 求解组合拍卖问题

6.5.2 菜品规划

生活中经常需要为家里囤积的食材能做哪些菜而发愁。在食材有限的情况下，希望得到通过这些食材能做哪些菜品的建议，具体地，如何用有限的食材做种类尽可能多的菜品？这个问题可以归结为组合拍卖问题，即标定每种成品菜需要的主食材后，希望能做出所有已知的成品菜，但所使用的主食材不能相同。将每项成品菜视为商家提供的竞拍方案，将有限的食材作为商品，在不引入偏好的情况下将所有成品菜的权重设置为 1，就可以将本问题规约为组合拍卖问题。可以利用相似的手段将菜品规划问题编码为 PMS 进行求解。为每个成品菜中使用的主食材生成变量，每个成品菜本身作为子句，编码的规则如下：

（1）如果一组成品菜消耗的一种食材超过了该项食材的储备量，则它们不可同时被满足，为硬子句；

（2）为所有成品菜需要的主食材分别建立软子句，在不考虑偏好的情况下子句权重为 1。

将提供固定数额的能量作为一份食材，可完成食材的量化处理[69]，如规定每份食材提供 90 千卡能量，可将不同重量的食物换算成份数。如通过查表得知鸡蛋每百克可食部提供能量 138 千卡[3]，那么 2 斤鸡蛋（假设都是可食部）可换算为 $\frac{2\times138\times5}{90}\approx15.3$ 份。假设现有主要食材鸡肉 20 份、虾 9 份、土豆 5 份、鸡蛋 15 份、娃娃菜 2 份、鲤鱼 9 份，现有食谱[4]拔丝土豆（土豆 2 份）、黄油煎虾（虾 3 份）、蒜蓉虾（虾 4 份）、白灼虾（虾 3 份）、鲤鱼炖白菜（鲤鱼 7 份，娃娃菜 1 份）、糖醋鲤鱼（鲤鱼 21 份）、苏格兰蛋（鸡蛋 1 份，手抓饼 1 份）、上汤娃娃菜（娃娃菜 2 份）。形式化表示食材 $Food=\{f_1,f_2,f_3,f_4,f_5,f_6,f_7\}$，分别代表鸡肉、虾、土豆、鸡蛋、娃娃菜、鲤鱼和手抓饼；食谱 $Dishes=\{(2f_3,1),(3f_2,1),(4f_2,1),(3f_2,1),(1f_5,7f_6,1),(21f_6,1),(1f_4,1f_7,1),(2f_5,1)\}$[5]，编写的 CNF 格式文件如图 6.7所示。求解结果如图 6.9所示，在不区分偏好的情况下储备的食材可以做拔丝土豆、黄油煎虾、蒜蓉虾、鲤鱼炖白菜。

[3]本文中使用的食材营养成分数据来源于 https://www.boohee.com

[4]食谱数据来源于 https://github.com/Anduin2017/HowToCook

[5]每对括号包裹一道菜的菜谱，与前文顺序一致；f 前面的数字代表使用的份数，末尾的数字代表权重。

若引入偏好，如将白灼虾和上汤娃娃菜的权重设置为 2，则新的 CNF 公式如图 6.8所示，再次求解的结果如图 6.10所示。本次算法建议做拔丝土豆、蒜蓉虾、白灼虾、上汤娃娃菜。

```
p wcnf 8 14 9
9 -2 -3 -4 0
9 -5 -8 0
9 -5 -6 0
9 -6 0
9 -7 0
9 -5 -8 0
1 1 0
1 2 0
1 3 0
1 4 0
1 5 0
1 6 0
1 7 0
1 8 0
```

图 6.7　菜品规划问题 CNF 文件

```
p wcnf 8 14 11
11 -2 -3 -4 0
11 -5 -8 0
11 -5 -6 0
11 -6 0
11 -7 0
11 -5 -8 0
1 1 0
1 2 0
1 3 0
2 4 0
1 5 0
1 6 0
1 7 0
2 8 0
```

图 6.8　带偏好的菜品规划问题 CNF 文件

```
yuhanyi@yuhanyi:/mnt/d/IdeaProjects/what2eat$ ./bin/HRP-C-printsolu src/main/resources/dishmenu.wcnf
o 4
time limit:60s
4       0
time limit:300s
4       0
v 1 2 3 -4 5 -6 -7 -8
```

图 6.9　HRP-C 求解菜品规划问题

```
yuhanyi@yuhanyi:/mnt/d/IdeaProjects/what2eat$ ./bin/HRP-C-printsolu src/main/resources/dishmenu.wcnf
o 4
time limit:60s
4       0
time limit:300s
4       0
v 1 -2 3 4 -5 -6 -7 8
```

图 6.10　HRP-C 求解带偏好的菜品规划问题

6.5.3 FPGA 布局

在 FPGA 的布局流程中，需要将电路网表的各个逻辑单元映射到 FPGA 芯片对应类型的物理单元[70]。例如 IO 单元只能映射到芯片四周的 IO 物理单元，有些单元只能映射到 CLB 物理单元或 DSP 物理单元上，且每个物理单元上只能分配一个逻辑单元。布局阶段除了要求逻辑单元类型与物理单元类型一一匹配，还需要考虑布局后的电路延时和总线长等问题[71]。FPGA 的布局流程常采用模拟退火法，但为了保证布局质量，往往需要运行较长时间。本文提出一种将 FPGA 布局优化流程编码为 PMS 问题的方法，由于评估布局质量的各项参数需要参考具体芯片，这里不考虑逻辑单元和物理单元类型匹配的约束，且只考虑尽可能缩短电路单元之间的逻辑连接的优化问题。在上述条件下，FPGA 的布局优化问题简化为：给定具有 rows*cols 数量物理单元的阵列，将 k 个逻辑单元一一映射到阵列中的任意物理单元处，且映射后的逻辑单元排列尽可能紧凑。

将“把第 i 个逻辑单元映射到坐标为 (x,y) 的物理单元”生成变量 v_{ixy}，正文字 v_{ixy} 表示进行这样的映射，负文字反之；将“坐标为 (x,y) 的物理单元已经与逻辑单元映射”生成变量 v_{xy}，其中正文字 v_{xy} 表示该物理单元已经参与映射，负文字反之。

硬子句，每个逻辑单元必须映射到物理单元上：

$$\forall i \in k, \bigvee_{x,y} v_{ixy}. \tag{6.5}$$

硬子句，每个逻辑单元只能映射到一个物理单元上：

$$\forall i \in k, \overline{v_{ixy}} \bigvee_{(x\neq x')\ or\ (y\neq y')} \overline{v_{ix'y'}}. \tag{6.6}$$

硬子句，两个逻辑单元不能映射到同一物理单元上：

$$\forall m, n \in k,\ m \neq n,\ \overline{v_{mxy}} \bigvee_{x,y} \overline{v_{nxy}}. \tag{6.7}$$

硬子句，如果逻辑单元映射到了坐标为 (x,y) 的物理单元上，则该物理单元

上一定存在了映射关系：

$$\forall i \in k, \overline{v_{ixy}} \bigvee_{x,y} v_{xy}. \tag{6.8}$$

硬子句，如果坐标为 (x,y) 的物理单元上存在映射关系，那么一定映射到逻辑单元的其中一个：

$$\forall i \in k, \overline{v_{xy}} \bigvee_{x,y} v_{ixy}. \tag{6.9}$$

软子句，如果坐标为 (x,y) 的物理单元上存在映射关系，那么该物理单元四周的物理单元上也应存在映射关系：

$$\overline{v_{xy}} \bigvee_{|x'-x+y'-y|=1} v_{x'y'}. \tag{6.10}$$

将软子句的权重设置为 1，硬子句权重设置为所有软子句权重之和加 1。对于将 45 个逻辑单元映射到含 8*8 个物理单元的 FPGA 芯片的简化问题，共生成变量 2944 个，子句 248013 个，其中软子句 224 个[6]。将 HRP-C 的求解结果进行可视化，方阵为 FPGA 芯片，单元格为物理单元，单元格为灰色代表该单元格上应该存在映射关系，单元格上的数字代表将第几个逻辑单元映射到该物理单元。图 6.11展示了 HRP-C 生成的初始分布结果，在该解中所有物理单元上都应该存在映射关系，但并不是所有的物理单元上都实际映射了逻辑单元，因此该解并不是可行的[7]；逻辑单元被映射到的位置集合并不紧凑，所以该解的质量是可以大幅优化的。图 6.12展示了 HRP-C 在 300 内搜索到的最优映射结果（实际仅耗时 28.65 秒），灰色的单元格上均存在对应的逻辑单元，且逻辑单元排列得十分紧凑，说明可以使用 (W)PMS 描述 FPGA 布局优化问题进行求解[8]。

[6]不同的编码方式会生成不同规模的问题实例。在本问题的一个硬约束“每个逻辑单元只能映射到一个物理单元上”中存在基数约束，可以采用鸽笼编码、组合编码等方式将基数约束翻译为 CNF 范式[72]，本文采取了组合编码的方式，因此会产生指数级的子句。

[7]没有满足硬约束：“如果逻辑单元映射到了坐标为 (x,y) 的物理单元上，则该物理单元上一定存在了映射关系”。

[8]相关工具会在完成学位答辩后发布在 `https://github.com/whyte-yhy?tab=repositories`，根据具体需求编写约束即可使用。

	4	12	38	21			
31	40		9		45	37	6
44	43	30	20	35		33	13
18				16	41	29	42
19		8	27	14			
22	26	24	15	17	10	36	32
	11	3	39		7	34	
28	2		23	5	25	1	

图 6.11　初始解的映射结果

1	39	10	15	7	25	35	44
14	32	37	23	9	42	4	29
21	8	31	17	41	40	5	19
13	6	36	3	2	11	34	26
24	16	30	43	45	22	27	28
38	12	20	33	18			

图 6.12　局部最优解的映射结果

6.6　小结

本章组合了第四章和第五章的改进思路，且保证了较低的时间复杂度，实验结果表明 HRP-C 在 (W)PMS 实例上的性能普遍优于 SATLike3.0。最后将 HRP-C 应用于组合拍卖、菜品规划和 FPGA 布局优化问题中，取得了良好的效果。

第 7 章　总结与展望

7.1　本文总结

(W)PMS 是组合优化领域的一项重要问题，它的子集包括 MaxSAT 和 SAT，在理论和实践中均有广泛应用。本文主要梳理了使用 SLS 解决 (W)PMS 的关键技术，讨论了可能的改进方向，并基于先进的求解器 SALike3.0 轻量级地实现了本文提出的改进方案，使其在构造初始解阶段具备了分析子句矛盾和优先传播硬变量的特性，得到了算法 HRP-C。HRP-C 在 MSE 提供的 (W)PMS 实例上的性能大幅超过 SATLike3.0。

HRP-C 目前存在的问题包括没有精细调整参数，以及在规定时间内求解 SAT 问题的能力仍无法媲美完备的 SAT 求解器[1]。

本文还将 HRP-C 应用于解决现实中的组合拍卖、菜品规划和 FPGA 布局优化问题中，取得了良好的结果。

7.2　未来工作

经过长期实验发现不断钻研某项技术可以提升算法总体的性能，但在一条技术路线上走得太远也许反倒不如在另一条技术路线上得到希望的效果（如从强制优先硬变量到采取 BMS 策略选取合适的变量）。原因在于求解系统的组件之间会相互影响，找到相互配合的方式可以事倍功半。

SLS 天然存在的问题即如何在局部视角和全局视角中权衡、如何确定重启搜索进程的时机、如何识别具体实例的固有结构。目前这些方面的研究仍然匮乏，甚至不知是否可以解决，因此笔者希望至少从提供更加多样的高质量初始解方面进行研究，如果整个搜索系统可以达到接近 SAT 求解器求解小中规模 SAT 实例的水准，那么对最先进的 SLS 研究就可以从目前复杂的混合 SLS 回归到简单清爽的传统 SLS 模式，对 SAT 求解器与 SLS 求解器混合方法的研究也不会再大多局限于互相提供初始解的范畴。但一个显著问题是 SAT 求解器也许已经是利用具体实

[1]目前最先进的混合不完备求解器将初始解生成算法简单替换为 SAT 求解器，在算法初始阶段只求解由硬子句组成的 SAT 问题，从而找到第一组可行解。

例结构能达到的性能上限，在其后拼接 SLS 求解器只是利用其无规则的特点提供随机性。如果上述猜想成立，那么单纯的 SLS 求解要想最大程度利用问题结构，就一定会发展为结构复杂的 SAT 求解器。

此外，笔者认为 SATLike 的初始解生成过程在某种程度上等同于 SAT 求解器的轻量级实现——区别在于没有子句学习、子句数据库、回溯和复杂的数据结构。两者的共同点在于都使用了单元传播技术，这使得它们都需要利用子句矛盾做出当下唯一正确的选择（一旦选择错误将导致回溯），因此识别可能频繁发生矛盾的变量以及实现简单的回溯策略正是笔者想追求的。

参考文献

[1] Cook S A. The complexity of theorem-proving procedures [C]. In Proceedings of the third annual ACM symposium on Theory of computing. 1971: 151–158.

[2] Milgrom P, Segal I. Clock auctions and radio spectrum reallocation [J]. Journal of Political Economy. 2020, 128 (1): 1–31.

[3] Marques-Silva J P, Sakallah K A. Boolean satisfiability in electronic design automation [C]. In Proceedings of the 37th Annual Design Automation Conference. 2000: 675–680.

[4] Festa P, Pardalos P M, Resende M G, et al. Randomized heuristics for the MAX-CUT problem [J]. Optimization methods and software. 2002, 17 (6): 1033–1058.

[5] Walter R, Zengler C, Küchlin W. Applications of MaxSAT in Automotive Configuration. [C]. In Configuration Workshop. 2013: 21.

[6] Ciampiconi L, Ghosh B, Scarlett J, et al. A MaxSAT-based framework for group testing [C]. In Proceedings of the AAAI Conference on Artificial Intelligence. 2020: 10144–10152.

[7] Li C M, Manya F. MaxSAT, hard and soft constraints [M] // Li C M, Manya F. Handbook of satisfiability. IOS Press, 2021: 2021: 903–927.

[8] Robinson N, Gretton C, Pham D N, et al. Partial weighted MaxSAT for optimal planning [C]. In PRICAI 2010: Trends in Artificial Intelligence: 11th Pacific Rim International Conference on Artificial Intelligence, Daegu, Korea, August 30–September 2, 2010. Proceedings 11. 2010: 231–243.

[9] De Coster A, Musliu N, Schaerf A, et al. Algorithm selection and instance space analysis for curriculum-based course timetabling [J]. Journal of Scheduling. 2022: 1–24.

[10] Borchers B, Furman J. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems [J]. Journal of Combinatorial Optimization. 1998, 2: 299–306.

[11] Heras F, Larrosa J, Oliveras A. MiniMaxSAT: An efficient weighted Max-SAT solver [J]. Journal of Artificial Intelligence Research. 2008, 31: 1–32.

[12] Palubeckis G. A new bounding procedure and an improved exact algorithm for the Max-2-SAT problem [J]. Applied Mathematics and Computation. 2009, 215 (3): 1106–1117.

[13] Fu Z, Malik S. On solving the partial MAX-SAT problem [C]. In Theory and Applications of Satisfiability Testing-SAT 2006: 9th International Conference, Seattle, WA, USA, August 12-15, 2006. Proceedings 9. 2006: 252–265.

[14] Narodytska N, Bacchus F. Maximum satisfiability using core-guided MaxSAT resolution [C]. In Proceedings of the AAAI Conference on Artificial Intelligence. 2014.

[15] Li C-M, Xu Z, Coll J, et al. Combining clause learning and branch and bound for MaxSAT [C]. In 27th International Conference on Principles and Practice of Constraint Programming (CP 2021). 2021.

[16] Berg J, Demirović E, Stuckey P J. Core-boosted linear search for incomplete MaxSAT [C]. In Integration of Constraint Programming, Artificial Intelligence, and Operations Research: 16th International Conference, CPAIOR 2019, Thessaloniki, Greece, June 4–7, 2019, Proceedings 16. 2019: 39–56.

[17] Lei Z, Cai S. Solving (Weighted) Partial MaxSAT by Dynamic Local Search for SAT. [C]. In IJCAI. 2018: 1346–52.

[18] Ansótegui C, Bonet M L, Giráldez-Cru J, et al. Community structure in industrial SAT instances [J]. Journal of Artificial Intelligence Research. 2019, 66: 443–472.

[19] Newsham Z, Ganesh V, Fischmeister S, et al. Impact of community structure on SAT solver performance [C]. In 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 17. 2014: 252–268.

[20] Ansótegui C, Giráldez-Cru J, Levy J, et al. Using community structure to detect relevant learnt clauses [C]. In Theory and Applications of Satisfiability Testing–SAT 2015: 18th International Conference, Austin, TX, USA, September 24-27, 2015, Proceedings 18. 2015: 238–254.

[21] Ansótegui C, Didier F, Gabàs J. Exploiting the structure of unsatisfiable cores in MaxSAT [C]. In Twenty-Fourth International Joint Conference on Artificial Intelligence. 2015.

[22] Liang J H, Ganesh V, Zulkoski E, et al. Understanding VSIDS branching heuristics in conflict-driven clause-learning SAT solvers [C]. In Hardware and Software: Verification and Testing: 11th International Haifa Verification Conference, HVC 2015, Haifa, Israel, November 17-19, 2015, Proceedings 11. 2015: 225–241.

[23] Audemard G, Simon L. On the glucose SAT solver [J]. International Journal on Artificial Intelligence Tools. 2018, 27 (01): 1840001.

[24] Cai S, Luo C, Thornton J, et al. Tailoring local search for partial MaxSAT [C]. In Proceedings of the AAAI Conference on Artificial Intelligence. 2014.

[25] AlKasem H H, Menai M E B. Stochastic local search for Partial Max-SAT: an experimental evaluation [J]. Artificial Intelligence Review. 2021, 54: 2525–2566.

[26] 雷震东. 最大可满足性及其扩展问题的求解 [D]. [S. l.]: 中国科学院软件研究所, 2021.

[27] Lübke O, Schupp S. noSAT-MaxSAT [C]. In MaxSAT Evaluation 2022. 2022: 29–30.

[28] Cai S, Lei Z. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability [J]. Artif. Intell. 2020, 287: 103354.

[29] Lorenz J-H, Wörz F. On the effect of learned clauses on stochastic local search [C]. In Theory and Applications of Satisfiability Testing–SAT 2020: 23rd International Conference, Alghero, Italy, July 3–10, 2020, Proceedings 23. 2020: 89–106.

[30] Selman B, Levesque H, Mitchell D. A New Method for Solving Hard Satisfiability Problems [C]. In Proceedings of the Tenth National Conference on Artificial Intelligence. 1992: 440–446.

[31] Jacobs L W, Brusco M J. Note: A local-search heuristic for large set-covering problems [J]. Naval Research Logistics (NRL). 1995, 42 (7): 1129–1140.

[32] Gao C, Weise T, Li J. A weighting-based local search heuristic algorithm for the Set Covering Problem [J]. 2014 IEEE Congress on Evolutionary Computation (CEC). 2014: 826–831.

[33] Morris P H. The Breakout Method for Escaping from Local Minima [C]. In AAAI Conference on Artificial Intelligence. 1993.

[34] Thornton J, Bain S, Sattar A, et al. A Two Level Local Search for MAX-SAT Problems with Hard and Soft Constraints [C]. In Australian Joint Conference on Artificial Intelligence. 2002.

[35] Luo C, Cai S, Wu W, et al. CCLS: An Efficient Local Search Algorithm for Weighted Maximum Satisfiability [J]. IEEE Transactions on Computers. 2015, 64: 1830–1843.

[36] Cai S, Luo C, Lin J, et al. New local search methods for partial MaxSAT [J]. Artificial Intelligence. 2016, 240: 1–18.

[37] Cai S, Luo C, Su K. CCASat: solver description [J]. Proceedings of SAT challenge. 2012, 2012: 13.

[38] Luo C, Cai S, Wu W, et al. CCLS: an efficient local search algorithm for weighted maximum satisfiability [J]. IEEE Transactions on Computers. 2014, 64 (7): 1830–1843.

[39] Britannica E. computational complexity. [EB/OL]. (2022-10-7)[2022-11-13], https://www.britannica.com/topic/computational-complexity.

[40] Britannica E. NP-complete problem. [EB/OL]. (2022-10-10)[2022-11-13], https://www.britannica.com/science/NP-complete-problem.

[41] Khachiyan L G. A polynomial algorithm in linear programming [C]. In Doklady Akademii Nauk. 1979: 1093–1096.

[42] 结城浩. 图解密码技术 [M]. 人民邮电出版社, 2016.

[43] Kastrati M, Biba M. Stochastic local search: a state-of-the-art review [J]. International Journal of Electrical and Computer Engineering. 2021, 11 (1): 716.

[44] Selman B, Kautz H A, Cohen B. Noise Strategies for Improving Local Search [C]. In Proceedings of the Twelfth AAAI National Conference on Artificial Intelligence. 1994: 337–343.

[45] Kautz H, Selman B, McAllester D. Walksat in the 2004 SAT Competition [C]. In Proceedings of the International Conference on Theory and Applications of Satisfiability Testing. 2004.

[46] Pisinger D, Ropke S. Large neighborhood search [J]. Handbook of metaheuristics. 2019: 99–127.

[47] Bacchus F, Berg J, Järvisalo M, et al. MaxSAT evaluation 2020: solver and benchmark descriptions [J]. 2020.

[48] Bacchus F, Berg J, Järvisalo M, et al. MaxSAT evaluation 2021: solver and benchmark descriptions [J]. 2021.

[49] Bacchus F, Berg J, Järvisalo M, et al. MaxSAT evaluation 2022: solver and benchmark descriptions [J]. 2022.

[50] Thornton J. Clause weighting local search for SAT [C]. In SAT 2005: Satisfiability Research in the Year 2005. 2006: 97–142.

[51] Zheng J, Zhou J, He K. Farsighted probabilistic sampling based local search for (weighted) partial maxsat [J]. arXiv preprint arXiv:2108.09988. 2021.

[52] 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.

[53] Thornton J, Pham D N, Bain S, et al. Additive versus multiplicative clause weighting for SAT [C]. In AAAI. 2004: 191–196.

[54] Hutter F, Tompkins D A, Hoos H H. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT [C]. In Principles and Practice of Constraint Programming-CP 2002: 8th International Conference, CP 2002 Ithaca, NY, USA, September 9–13, 2002 Proceedings 8. 2002: 233–248.

[55] Schuurmans D, Southey F, Holte R C, et al. The exponentiated subgradient algorithm for heuristic boolean programming [C]. In IJCAI. 2001: 334–341.

[56] Chu Y, Cai S, Lei Z, et al. NuWLS-c: Solver Description [J]. MaxSAT Evaluation 2022: 28.

[57] 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.

[58] Cai S, Su K. Configuration checking with aspiration in local search for SAT [C]. In Proceedings of the AAAI Conference on Artificial Intelligence. 2012: 434–440.

[59] Chu Y, Luo C, Huang W, et al. Hard neighboring variables based configuration checking in stochastic local search for weighted partial maximum satisfiability [C]. In 2017 IEEE 29th International Conference on Tools with Artificial Intelligence (ICTAI). 2017: 139–146.

[60] Hoos H H, Stützle T. Stochastic local search algorithms: an overview [J]. Springer Handbook of Computational Intelligence. 2015: 1085–1105.

[61] Cai S. Balance between complexity and quality: Local search for minimum vertex cover in massive graphs [C]. In Twenty-Fourth International Joint Conference on Artificial Intelligence. 2015.

[62] Shaw P. Using constraint programming and local search methods to solve vehicle routing problems [C]. In Principles and Practice of Constraint Programming—CP98: 4th International Conference, CP98 Pisa, Italy, October 26–30, 1998 Proceedings 4. 1998: 417–431.

[63] Nadel A. Anytime weighted MaxSAT with improved polarity selection and bit-vector optimization [C]. In 2019 Formal Methods in Computer Aided Design (FMCAD). 2019: 193–202.

[64] Demirović E, Chu G, Stuckey P J. Solution-based phase saving for CP: A value-selection heuristic to simulate local search behavior in complete solvers [C]. In Principles and Practice of Constraint Programming: 24th International Conference, CP 2018, Lille, France, August 27-31, 2018, Proceedings 24. 2018: 99–108.

[65] Nadel A. Polarity and variable selection heuristics for SAT-based anytime MaxSAT [J]. Journal on Satisfiability, Boolean Modeling and Computation. 2020, 12 (1): 17–22.

[66] Paxian T, Raiola P, Becker B. On preprocessing for weighted MaxSAT [C]. In Verification, Model Checking, and Abstract Interpretation: 22nd International Conference, VMCAI 2021, Copenhagen, Denmark, January 17–19, 2021, Proceedings 22. 2021: 556–577.

[67] Nudelman E, Leyton-Brown K, Hoos H H, et al. Understanding random SAT: Beyond the clauses-to-variables ratio [C]. In Principles and Practice of Constraint Programming–CP 2004: 10th International Conference, CP 2004, Toronto, Canada, September 27-October 1, 2004. Proceedings 10. 2004: 438–452.

[68] Newman M E, Girvan M. Finding and evaluating community structure in networks [J]. Physical review E. 2004, 69 (2): 026113.

[69] 中国营养学会. 中国居民膳食指南 [M]. 人民卫生出版社, 2022.

[70] Chen S-C, Chang Y-W. FPGA placement and routing [C]. In 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD). 2017: 914–921.

[71] 王德奎. FPGA 高效布线方法研究 [D]. [S. l.]: 西安电子科技大学, 2020.

[72] Boudane A, Jabbour S, Raddaoui B, et al. Efficient SAT-Based Encodings of Conditional Cardinality Constraints. [C]. In LPAR. 2018: 181–195.

致 谢

当是穿越冬夏才能修成正果，历经考研和推免的少年来了勤勤，从路边的包子铺去往楼下的肠粉摊。

三年来，我的身份增了又增——我是陈寅老师 2020 级唯一的学生，是陶南 420 宿舍的一员，是 332 实验室的一份子，是计算机学院第十一届研究生会主席团成员，是计算机学院 2021-2022 学年的兼职辅导员，是“龙吟忆往昔”的“奶爸”绾青丝，是“提瓦特大陆”的午时，是武汉博格华纳研发中心李晓伦培养的实习生，亦是宣誓忠诚的中共预备党员。

如果说艾薇儿的声音像颗蓝宝石，Juice Wrld 的声音正如将死之人的呜咽。听懂他的狂欢与落寞，正是我在狂欢与落寞中徘徊。

我的性格很坏，总是与自己怄气，心里总念叨着几句话：事物在发展，好与坏更迭，所以无法提供与人有用的建议，千万不要主动改变他人，因而要保持缄默。可缄默的主要原因终究不来源于此：我担心自己会不会打搅别人，因为自己不喜欢被打搅；担心自己的话在别人看来是冒犯的，因为曾被人误解。到后来，发现自己其实真的喜欢一个人发呆。我不喜欢受到陌生人的羡慕，取得的一切成果都暗自付出了代价，多数时候都让我感到是在利用我抒发他们所谓的苦难。我从不值得称道的环境成长起来，拒绝接受这样的傲慢。

可我总是遇到好人，如同再次从那个躁动而暴力的青春期悠悠穿过。在陈寅老师门下学习的自由让我有机会参与学院建设和“不务正业”；在陈卫东老师、龚征老师的课堂中寻得了感兴趣的研究方向；中国科学院大学的雷震东师兄向我提供了算法的源代码；在罗盼老师的开导下不再执拗于完美策划；王政忠、罗兰苑、陈婧宇、郭月晴和唐婷芳老师教会了我行政技能；与姚、张、郑等室友打造舒适的睡眠环境；在 332 实验室大家庭开朗的学习氛围中顺利完成科研工作；在李晓伦老师的指导下入行汽车电子；杜、孙、邹、薛、王、车、周等最好的兄弟姊妹们给予我敢于迎接失败的底气。

衷心祝福恩师与同窗们身体健康，希望在来日的挫折里总有贵人相助。感谢

我的父母，尽管我们矛盾重重，但能忍受彼此的坏毛病走到现在，已是世上最为难得。还有那个我，无论情绪怎样陷入泥潭，都想办法存续，毕竟拯救自己是项工作，如果没人能帮自己，那总得有人来做。

祝我毕业快乐!

作者攻读学位期间发表的学术论文目录

发表的学术论文

[1] 于瀚一, 陈寅. 基于 SATLike3.0 局部搜索求解器的算法改进. (已被计算机系统应用录用.)