求解加权偏 MaxSAT 问题的通用子句加权方法

郑迥之 何 琨
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 最大可满足性问题(Maximum Satisfiability Problem,MaxSAT)是著名的可满足性问题(Satisfiability Problem,SAT)的优化形式,也是一个经典的 NP 难组合优化问题. 加权偏 MaxSAT(Weighted Partial MaxSAT,WPMS)是最一般的一类 MaxSAT 问题,其中包含了必须要满足的硬子句,对应了优化问题中的约束条件,以及带权重的软子句,对应了优化问题中的优化目标. WPMS 旨在满足所有硬子句的同时最大化被满足软子句的权重之和. 工业场景中和学术领域中的许多优化问题都能够转化成 WPMS 问题进行求解,因此 WPMS 具有广泛的应用领域和重要的研究意义. 局部搜索方法是求解 WPMS 问题的一种著名且被广泛研究的非完备方法. 子句加权技术是 WPMS 局部搜索算法中常用的一种有效且关键的技术,通过为子句赋予动态权重并在搜索过程中更新它们以引导搜索方向,帮助算法逃离局部最优. 最先进的 WPMS 局部搜索算法都提出或采用了有效的子句加权技术,以帮助它们在不同的解空间中搜索. 然而,现有的子句加权技术仅根据当前局部最优解更新子句动态权重,而未考虑任何历史信息,可能导致子句加权的视野局限,对搜索方向的引导不够准确. 为了解决这一问题,提出了一种新的子句加权技术,称为 Hist-Weighting(Clause Weighting with Historical Information),同时考虑了当前及历史信息来更新子句的动态权重,以改进子句加权机制和局部搜索算法的搜索精度和效率. 具体而言,Hist-Weighting 为那些同时被当前和历史局部最优解所不满足的子句赋予更大的动态权重增量,使算法更倾向于满足那些久未被满足且难以被满足的子句,提高子句加权的准确度. 此外,在 Hist-Weighting 中,子句动态权重的增量能够根据子句中的变元得分自适应地调整,使子句加权更具有灵活性. Hist-Weighting 还为子句动态权重的增量设置了上下限,保证了子句加权的稳定性. 为了评估所提出的 Hist-Weighting 子句加权技术的性能,将其应用于三种最先进的 WPMS 局部搜索算法,即 BandMaxSAT、SATLike3.0 和 CCEHC. 在近五届 MaxSAT 国际算法竞赛 MaxSAT Evaluation 非完备组的所有 WPMS 算例上的实验结果表明,应用 Hist-Weighting 技术的改进算法相比于原算法在获胜算例数上能够提升约 10%至 60%,体现了所提出的 Hist-Weighting 子句加权技术在求解 WPMS 问题时的有效性. 此外,通过将应用了 Hist-Weighting 的改进局部搜索算法与其变体算法对比以进行消融实验,表明了 Hist-Weighting 中限制动态权重增量上下限,以及使动态权重增量根据变元得分自适应调整的机制的有效性.

关键词 最大可满足性问题;局部搜索;子句加权技术;历史信息
中图法分类号 TP301 **DOI 号** 10.11897/SP.J.1016.2024.01341

A General Clause Weighting Method for Solving the Weighted Partial MaxSAT Problem

ZHENG Jiong-Zhi HE Kun
(*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074)

Abstract The Maximum Satisfiability Problem (MaxSAT) is an optimization form of the well-known Satisfiability Problem (SAT) and a classic NP-hard combinatorial optimization problem. The Weighted Partial MaxSAT (WPMS) is the most general type of MaxSAT problem, which includes hard clauses that must be satisfied, corresponding to the constraints in optimization

收稿日期:2023-08-10;在线发布日期:2024-04-08. 本课题得到国家自然科学基金项目(U22B2017)资助.**郑迥之**,博士研究生,主要研究方向为 NP 难度问题求解、组合优化. E-mail: jzzheng@hust.edu.cn. **何 琨**(通信作者),博士,教授,博士生导师,中国计算机学会(CCF)杰出会员,主要研究领域为机器学习、对抗学习、组合优化、图数据挖掘. E-mail: brooklet60@hust.edu.cn.

problems, and weighted soft clauses that correspond to the optimization objective in optimization problems. WPMS aims to satisfy all hard clauses while maximizing the total weight of the satisfied soft clauses. Many optimization problems in industrial scenarios and academic fields can be transformed into WPMS for solving. Therefore, WPMS has a wide range of application domains and is of important research significance. The local search method is a well-known and widely studied incomplete method for solving WPMS. The clause weighting technique is an effective and crucial technique that is commonly used in WPMS local search algorithms, which assigns dynamic weights to clauses and updates them during the search process to guide the search direction, helping the algorithm escape from local optima. The most advanced WPMS local search algorithms have proposed or adopted effective clause weighting techniques to help them search in different solution spaces. However, existing clause weighting techniques only update the dynamic weights of clauses based on the current local optimal solution without considering any historical information, which may limit the perspective of clause weighting and lead to inaccurate guidance of search direction. To address this issue, a new clause weighting technique called Hist-Weighting (Clause Weighting with Historical Information) is proposed, which simultaneously considers the current and historical information to update the dynamic weights of clauses, in order to improve the search accuracy and efficiency of the clause weighting scheme and the local search algorithm. Specifically, Hist-Weighting assigns greater dynamic weight increments to clauses that are not satisfied by both the current and historical local optima, making the algorithm more inclined to satisfy clauses that have not been satisfied for a long time and are difficult to satisfy, so as to improve the accuracy of the clause weighting scheme. In addition, in Hist-Weighting, the increment of clause dynamic weights can be adaptively adjusted to make the clause weighting scheme more flexible. Hist-Weighting also sets upper and lower limits for the increment of dynamic weights in clauses, ensuring the stability of the clause weighting scheme. To evaluate the performance of the proposed Hist-Weighting clause weighting technique, it was applied to three state-of-the-art WPMS local search algorithms, namely BandMaxSAT, SATLike3.0, and CCEHC. The experimental results on all WPMS instances of the incomplete track of the last five years of MaxSAT Evaluations show that the improved algorithm using Hist-Weighting can increase the number of winning instances by about 10% to 60% compared to the baseline algorithm, indicating the effectiveness of the proposed Hist-Weighting clause weighting technique in solving WPMS. In addition, ablation studies are performed by comparing the improved local search algorithms using Hist-Weighting and their variant algorithms, indicating the effectiveness of the mechanisms in Hist-Weighting, including restricting the upper and lower limits of dynamic weight increments and adapting dynamic weight increments.

Keywords MaxSAT problem; local search; clause weighting technique; historical information

1 引 言

给定一个命题逻辑合取范式(Conjunctive Normal Form,CNF),著名的可满足性问题(Satisfiability Problem,SAT)是判断其是否可被满足的一个判定问题.SAT 也是第一个被证明为 NP 完全的问题[1].最大可满足性问题(Maximum Satisfiability Problem,MaxSAT)是 SAT 问题的优化形式,是一个经典的具有 NP 难度的组合优化问题.由于 MaxSAT 问题强大的表达能力,许多学术问题和工业问题都能转化为 MaxSAT 问题求解,例如学术研究中的规划问题[2]、路径问题[3-4]、最大团问题[5-7]、相关聚类问题[8-9]、最小顶点覆盖问题[10]等;以及工业问题中的群组检测[11]、时间表问题[12]、会议安排问题[13-14]、电路设计自动化[15-16]、优化癌症治疗设计[17]、检测

硬件木马[18]、最小校正集枚举问题[19]、最大可满足子集计数问题[20]、线性时序逻辑[21]等.

局部搜索[22-23]是求解 MaxSAT 问题的一类重要的非完备算法(即非精确算法),在近年的 MaxSAT 国际算法竞赛 MaxSAT Evaluation 非完备组中扮演了非常重要的角色,主导与影响了多次冠军的产生.子句加权技术是 MaxSAT 局部搜索算法中的一项关键技术,常作为核心组件被用于各种先进的 MaxSAT 局部搜索算法中.本文针对 MaxSAT 这一基础性问题的局部搜索算法展开研究,首次提出了一种利用历史搜索信息的子句加权技术,能够改进一系列先进的 MaxSAT 局部搜索算法,为 MaxSAT 的子句加权技术提供了新的研究思路与方向,为 MaxSAT 及其相关应用领域提供了高效的求解技术与工具.

MaxSAT 问题是定义在 CNF 公式上的,其基本组件包括布尔变元、文字和子句.给定一个 CNF 公式 F,$V(F)$表示 F 中所有变元的集合$\{x_1,x_2,\cdots,x_n\}$,$x_i\in\{0,1\}(i=1,2,\cdots,n)$,映射 $\alpha:V(F)\to\{0,1\}$表示 $V(F)$的一组赋值.文字是某个变元本身 x_i或者是它的否定形式 $\bar{x}_i$,当且仅当变元 x_i为真时(即 $x_i=1$),正文字 x_i被满足,反之当且仅当 x_i为假时(即 $x_i=0$),负文字 $\bar{x}_i$被满足.子句是文字的析取,例如:$c_i=l_{i1}\vee l_{i2}\vee\cdots\vee l_{ip}$,其中 p 表示子句 c_i所包含的文字数量,$l_{ij}(j=1,2,\cdots,p)$表示子句 c_i中的第 j 个文字.当且仅当一个子句中至少有一个文字被满足时,该子句被满足.CNF 公式是子句的合取,例如:$F=c_1\wedge c_2\wedge\cdots\wedge c_m$,其中 m 表示公式 F 中含有的子句个数.当且仅当公式 F 中所有子句都被满足时,公式 F 被满足.

给定一个 CNF 公式 F,SAT 问题是判断是否存在一组赋值 α 满足公式 F,MaxSAT 问题是要找到一组赋值 α 满足 F 中尽可能多的子句.在更一般的情况下,偏 MaxSAT 问题(Partial MaxSAT,PMS)将子句分为硬子句和软子句两种,其目标是找到一组赋值 α 满足 F 中所有硬子句,并满足尽可能多的软子句.为 PMS 问题中的每个软子句赋予一个正的权重值就得到了加权偏 MaxSAT 问题(Weighted PMS,WPMS),其目标是找到一组赋值 α 满足 F 中所有硬子句,并使被满足的软子句的权重之和最大.在利用 MaxSAT 解决实际生产生活中的约束优化问题时,通常将其中的约束建模为硬子句,将优化目标建模为软子句.PMS 和 WPMS 问题也因其强大的建模和表达能力而越来越受到国内外研究者的广泛关注,例如近年的 MaxSAT 国际算法竞赛 MaxSAT Evaluation 的完备组与非完备组的赛题都是 PMS 和 WPMS 算例.由于 PMS 可视为特殊的 WPMS 算例,因此 WPMS 是上述系列问题中最一般的形式,本文的研究对象也是 WPMS 问题.

求解 WPMS 的局部搜索算法的流程通常是从一个初始解开始,每次迭代将被选择的变元(通常为一个)的布尔值翻转,直至达到停止条件.近年来,国内外研究者从多个角度提出了改进求解 WPMS 的局部搜索算法的策略,例如变元选择策略[24]、初始化方法[25-26]、子句加权技术[27-28]等.其中变元选择策略为算法选择每步迭代翻转的变元;初始化方法为局部搜索提供更高质量的初始解;子句加权技术通过为子句赋予动态权重,并在搜索过程中动态调整这些权重来引导算法的搜索方向.本文主要针对求解 WPMS 问题的局部搜索算法中的子句加权技术展开研究.注意,WPMS 是因为每个软子句被赋予了权重作为不满足它所需的代价而被冠以加权的名字,这一权重值是固定不变的,在后文中称为原始权重.而子句加权技术是为每个子句赋予动态权重,该权重与原始权重无关,且会在搜索过程中根据子句不满足的情况进行调整以引导算法的搜索方向.为了便于区分,在后文中将子句加权技术赋予的权重称为动态权重.

目前,求解 WPMS 问题的最先进的局部搜索算法中都采用了子句加权技术,如 CCEHC[27]、SATLike[28]、SATLike3.0[26]和 BandMaxSAT[24].子句加权技术的主要思想就是在算法陷入局部最优时,增加难以被满足的子句的动态权重,使算法在后续的搜索过程中更倾向于满足那些难以被满足的子句,从而改变算法搜索方向,避免算法陷入局部最优,帮助算法找到更好的解.逃离局部最优是局部搜索算法的一项非常重要的能力,复杂组合优化问题的解空间中可能存在着无数的局部最优陷阱,而全局最优解可能寥寥无几,算法易于跳出局部最优意味着它有更多的机会找到全局最优.

子句加权技术在求解 MaxSAT 问题的局部搜索算法领域取得了很大的成就.但是,现有的子句加权技术仅仅考虑了当前赋值所不满足的子句,而没有结合历史信息综合考虑子句动态权重的调整,可能导致子句加权的视野局限,对搜索方向的引导不够准确.针对这一问题,本文提出一种通用的改进子句加权技术,称为 Hist-Weighting(Clause Weighting

with Historical Information),通过结合当前解和历史信息综合调整子句的动态权重.具体而言,Hist-Weighting 为被当前和历史局部最优解所同时不满足的子句的动态权重赋予更大的增量,引导算法更加倾向于满足这些久未满足的子句.子句久未满足实际上意味着陷入局部最优,因此 Hist-Weighting 能够加速久未满足子句的动态权重增长,使子句加权更快地收敛,从而使算法提前满足这些难以满足的子句,帮助算法更快跳出局部最优,提高算法的搜索效率.

本文将 Hist-Weighting 应用于求解 WPMS 问题最先进的局部搜索算法中,包括 CCEHC、SATLike3.0(SATLike 的改进版本),以及 BandMaxSAT.在最近五届的 MaxSAT Evaluation 非完备组 WPMS 算例集上的实验结果表明,Hist-Weighting 能够显著改进它们的性能.

在后续内容中,首先介绍局部搜索算法中所用到的一些定义,然后介绍求解 WPMS 问题的局部搜索算法的通用框架,并介绍子句加权技术的主要流程,同时介绍目前最先进的局部搜索算法在框架和子句加权技术上的异同;再介绍所提出的 Hist-Weighting 子句加权技术的流程、思想和优点.最后,通过与现有算法的对比实验验证 Hist-Weighting 的通用性和高效性,通过消融实验分析 Hist-Weighting 中各细节设计的作用,并以结论章节总结全文、展望未来.

2 局部搜索相关定义

翻转变元的布尔值是求解 MaxSAT 问题的局部搜索算法中最常见的操作,本文定义 $flip(x)$ 为在当前解中翻转变元 x 的布尔值,即若 x 的值为 1,$flip(x)$ 使 x 的值变为 0,反之亦然.在选择翻转变元时,算法通常用到的一种评估指标是得分.目前采用子句加权技术的局部搜索算法有两种,一种仅对硬子句赋予动态权重,软子句则维持原始权重不变,例如 CCEHC 算法;另一种对硬软子句都赋予动态权重,例如 SATLike(3.0)和 BandMaxSAT 算法.仅对硬子句赋予动态权重的算法分别针对硬软子句计算变元的得分,即变元 x 的硬得分 $hscore(x)$ 为 $flip(x)$ 使公式 F 中不满足硬子句动态权重的减小量,变元的软得分 $sscore(x)$ 为 $flip(x)$ 使公式 F 中不满足软子句原始权重的减小量.对硬软子句都赋予动态权重的算法则统一计算变元的得分,即变元 x 的得分 $score(x)$ 为 $flip(x)$ 使公式 F 中不满足子句动态权重的减小量.

在求解 MaxSAT 问题的局部搜索算法中,通常会维护一个变元集合 $GoodVars$ 以储存一些优质的变元,翻转其中的任何一个都能够改进当前解,使得不满足子句的(动态)权重之和减少.对于仅赋予硬子句动态权重的算法,$GoodVars$ 被定义为 $\{x|hscore(x)>0 \vee (hscore(x)=0 \wedge sscore(x)>0)\}$,对于赋予所有子句动态权重的算法,$GoodVars$ 被定义为 $\{x|score(x)>0\}$.因此,局部搜索算法通常认为当 $GoodVars \neq \varnothing$ 时,算法未陷入局部最优,反之算法陷入局部最优.

举一个简单的样例来描述变元得分的计算方法.假设 CNF 公式 $F=c_1 \wedge c_2 \wedge c_3$,其中 $c_1=x_1 \vee x_2 \vee x_3$,$c_2=\bar{x}_1 \vee x_2$,$c_3=\bar{x}_3 \vee \bar{x}_4$,子句 c_1、c_2、c_3 的动态权重分别为 1、2、3,当前解 α 中变元 x_1、x_2、x_3、x_4 的赋值分别为 1、0、0、1.则此时翻转 x_1 将导致子句 c_1 不满足、子句 c_2 被满足,因此有 $score(x_1)=1$;翻转 x_2 将导致子句 c_2 被满足,因此有 $score(x_2)=2$;翻转 x_3 将导致子句 c_3 不满足,因此有 $score(x_3)=-3$;翻转 x_4 不会对任一子句的满足性造成影响,因此有 $score(x_4)=0$.

此外,本文定义 W 为储存所有子句动态权重的数组,W_c 表示子句 c 的动态权重,并定义 $cost(\alpha)$ 为赋值 α 的代价,当 α 为可行解时,即 α 满足所有硬子句,$cost(\alpha)$ 等于所有不满足软子句的原始权重之和;当 α 为不可行解时,$cost(\alpha)$ 等于 $+\infty$.

3 现有局部搜索与子句加权框架

目前,最先进的求解 WPMS 问题的局部搜索算法,如 CCEHC[27]、SATLike[28]、SATLike3.0[26]、BandMaxSAT[24],都遵循着大致统一的搜索框架,即从一个初始解开始,每步迭代选择并翻转一个变元,在算法陷入局部最优时利用子句加权技术更新子句的动态权重.这些算法的区别主要在于初始化方法、变元选择策略和子句加权技术的细节.本章将首先介绍局部搜索与子句加权的通用框架,再详细介绍各算法的细节区别,以便于理解如何将所提出的 Hist-Weighting 加权技术应用于这些算法,以及分析 Hist-Weighting 应用于这些算法的效果差异.

3.1 局部搜索通用框架

算法 1 展示了求解 WPMS 问题的局部搜索算法的通用框架 $LocalSearch$.算法首先通过 $Init()$ 函

数初始化一个解 α(第 1 行),而后用 α 初始化当前最优解 α^*(第 2 行),并初始化子句的动态权重(第 3 行),随后重复选择与翻转变元操作直至达到截止时间 T_{max}(第 4～13 行),在此期间更新子句动态权重以引导搜索方向(第 8 行),并更新当前最优解 α^*(第 12 行),最后若 α^* 为可行解,则返回 α^*,否则算法未能在截止时间内找到可行解(第 14～15 行).

算法 1. $LocalSearch(F,T_{max})$.

输入:WPMS 算例 F,截止时间 T_{max}

输出:可行解 α 或"*no feasible solution found*"

1. $\alpha \leftarrow Init(F)$;
2. $\alpha^* \leftarrow \alpha$;
3. $W \leftarrow InitClauseWeights()$;
4. WHILE T_{max} is not reached DO
5. IF $GoodVars \neq \varnothing$ THEN /* 算法未达到局部最优 */
6. $v \leftarrow$ a variable in $GoodVars$;
7. ELSE /* 算法达到局部最优 */
8. $W \leftarrow UpdateClauseWeights(F,\alpha,W)$;
9. $v \leftarrow LocalOptEscaping()$;
10. END IF
11. $\alpha \leftarrow flip(v)$ in α;
12. $\alpha^* \leftarrow \alpha$ if $cost(\alpha) < cost(\alpha^*)$;
13. END WHILE
14. IF α^* is feasible THEN RETURN α^*;
15. ELSE RETURN "*no feasible solution found*";

在算法迭代搜索的过程中(第 4～13 行),当 $GoodVars \neq \varnothing$ 时,算法未达到局部最优,此时算法从 $GoodVars$ 中选择将要翻转的变元 v;当 $GoodVars = \varnothing$ 时,算法达到局部最优,即翻转任一变元都不能改进当前解,此时当前解 α 即为局部最优解,算法先利用 $UpdateClauseWeights()$ 函数更新子句的动态权重(第 8 行),再利用 $LocalOptEscaping()$ 函数选择将要翻转的变元 v 以逃离局部最优(第 9 行). 实际上,各种 WPMS 局部搜索算法在选择翻转变元时(算法 1 中第 6 行和第 9 行)都是采用随机和贪心相结合的策略,即在有一定概率随机探索的前提下尽量选择得分最高的变元. 由第 2 节的介绍可知,变元的得分与子句的动态权重紧密相关,因此对动态权重的调整能够影响变元的得分,进而影响搜索方向.

3.2　子句加权通用框架

算法 2 展示了 WPMS 局部搜索算法中常用的子句加权技术的通用框架 $UpdateClauseWeights$. 算法以 sp 的概率平滑(即减小)子句动态权重(第 3～8 行),以 $1-sp$ 的概率增加子句动态权重(第 10～15 行),sp 通常是一个非常小的值(小于等于 0.01). 平滑的目的是维持子句动态权重的稳定,使其可以收敛而非单调增加. 具体地,在平滑阶段,算法将被当前局部最优解 α 所满足的硬、软子句的动态权重分别减小 h_{inc} 和 s_{inc};在增加阶段,算法将被 α 所不满足的硬、软子句的动态权重分别增加 h_{inc} 和 s_{inc}. 算法 2 中的 sp、h_{inc}、s_{inc} 皆为超参数. 此外,局部搜索算法还会为硬软子句的动态权重设置上下限,避免出现子句动态权重为 0 或为负,避免软子句动态权重过大等等. 这些上下限同样是以超参数的形式被设置与调整的,若某次子句动态权重的调整将导致某条子句的动态权重超出上下限,则该子句的动态权重在本次子句加权中保持不变.

算法 2. $UpdateClauseWeights(F,\alpha,W)$.

输入:WPMS 算例 F,当前局部最优解 α,子句动态权重数组 W

输出:更新后的子句动态权重数组 W

1. $r \leftarrow$ a random number in $[0,1]$;
2. IF $r < sp$ THEN /* 平滑子句动态权重 */
3. FOR EACH clause c satisfied by α in F DO
4. IF c is a hard clause THEN
5. $W_c \leftarrow W_c - h_{inc}$;
6. ELSE $W_c \leftarrow W_c - s_{inc}$;
7. END IF
8. END FOR
9. ELSE /* 增加子句动态权重 */
10. FOR EACH clause c falsified by α in F DO
11. IF c is a hard clause THEN
12. $W_c \leftarrow W_c + h_{inc}$;
13. ELSE $W_c \leftarrow W_c + s_{inc}$;
14. END IF
15. END FOR
16. END IF
17. RETURN W;

总之,现有子句加权技术在算法陷入局部最优时,调整被当前局部最优解 α 所满足或不满足的子句的动态权重,使被当前局部最优解 α 所不满足的子句的动态权重增加,或使被当前局部最优解 α 所满足的子句的动态权重减小,进而调整变元的得分,使算法更倾向于满足被当前局部最优解 α 所不满足的子句,达到引导算法搜索方向,帮助算法逃离局部最优,搜索到更好的解等目的.

3.3　各算法细节差异

CCEHC、SATLike、SATLike3.0、BandMaxSAT 算法的细节差异主要在初始化方法,即算法 1 中的 $Init()$ 函数;变元选择策略中的局部最优逃离策略,

即算法 1 中的 *LocalOptEscaping*()函数;以及子句加权技术,即算法 1 中的 *InitClauseWeights*()和 *UpdateClauseWeights*()函数. 由于本文主要关注 WPMS 局部搜索算法中的子句加权模块,本小节主要阐述几个算法在子句加权方面的差异.

具体地,CCEHC 仅对硬子句赋予动态权重,硬子句初始动态权重为 1,软子句则维持原始权重不变,可将其视为软子句动态权重等于原始权重,且 s_{inc} 参数为 0 的算法 2 的特殊版本;SATLike、SATLike3.0、BandMaxSAT 都对所有子句赋予了动态权重,并设置了 h_{inc} 与 s_{inc} 参数以分别更新硬软子句的动态权重.

4 Hist-Weighting 通用子句加权方法

从第 3 节可知,在现有的求解 WPMS 问题的局部搜索算法中,子句加权技术在更新子句的动态权重时都仅着眼于当前局部最优解,而没有考虑历史信息,即没有考察各子句在历史搜索过程中是否都难以满足. 针对这一问题,本文提出一种通用的子句加权技术,称为 Hist-Weighting(Clause Weighting with Historical Information),能够结合当前局部最优解与历史局部最优解来综合调整子句的动态权重.

本节将首先介绍 Hist-Weighting 子句加权技术的流程,再阐述和分析其主要思想和优点.

4.1 Hist-Weighting 的流程

Hist-Weighting 主要调整的是子句加权技术中更新子句动态权重的部分,确切地说是增加子句动态权重的部分,而不改变各局部搜索算法初始化子句动态权重的部分;各算法的平滑率参数 sp;以及各算法平滑子句动态权重的部分.

算法 3 展示了 Hist-Weighting 增加子句动态权重的流程 *IncreaseByHistWeighting*(). 相比于算法 2,算法 3 的输入参数中增加了上一局部最优解 α' 以及控制子句动态权重增量的参数 h_{min}、h_{max}、s_{min}、s_{max}. 如第 3.1 节和第 3.2 节所述,局部搜索算法通常认为当 $GoodVars=\varnothing$ 时算法陷入局部最优,当前解 α 即为局部最优解,子句加权技术将根据当前局部最优解 α 对子句动态权重进行调整. 在 Hist-Weighting 中,子句加权同样作用于算法陷入局部最优时,即 $GoodVars=\varnothing$ 时,但 Hist-Weighting 不仅会根据当前使 $GoodVars=\varnothing$ 的局部最优解 α 调整子句动态权重,还会考虑搜索过程中上一次使 $GoodVars=\varnothing$ 的局部最优解 α'.

算法 3. *IncreaseByHistWeighting*(F, α, α', W, h_{min}, h_{max}, s_{min}, s_{max}).

输入:WPMS 算例 F,当前局部最优解 α,上一局部最优解 α',子句动态权重数组 W,控制子句动态权重增量参数 h_{min}、h_{max}、s_{min}、s_{max}

输出:更新后的子句动态权重数组 W

```
1.  FOR EACH clause c falsified by α in F DO
2.    score*←the maximum score of variables in c;
3.    IF c is a hard clause THEN /* 统一参数 */
4.      inc←h_inc, min←h_min, max←h_max;
5.    ELSE
6.      inc←s_inc, min←s_min, max←s_max;
7.    END IF
8.    IF c is falsified by α' THEN /* c 同时被 α 和 α' 所不满足 */
9.      IF 1−score*>max·inc THEN
10.       W_c←W_c+max·inc;
11.     ELSE IF 1−score*>min·inc THEN
12.       W_c←W_c+1−score*;
13.     ELSE W_c←W_c+min·inc;
14.     END IF
15.   ELSE W_c←W_c+inc; /* c 仅被 α 所不满足 */
16.   END IF
17. END FOR
18. RETURN W;
```

算法 3 首先遍历 F 中被当前局部最优解 α 所不满足的子句(第 1 行),分别调整每个子句的动态权重. 算法将不满足子句 c 中得分最高的变元的得分记作 $score^*$(第 2 行),由于当前解 α 为局部最优解,这意味着 $GoodVars=\varnothing$,即不存在得分为正的变元,因此通常有 $score^*\leq 0$(某些被禁忌而未加入 $GoodVars$ 的变元可能有正的得分). 而后,根据子句 c 的硬软属性采用不同的参数 min、max、inc 来增加子句动态权重(第 3~7 行).

算法 3 第 8~16 行为子句动态权重的增加过程,具体细节如下:若子句 c 同时被赋值 α 和 α' 所不满足,说明连续的两个局部最优解都未能满足子句 c,因此相比于仅被赋值 α 所不满足的子句,子句 c 更加难以被满足. 针对这一类型的子句,本文所提出的 Hist-Weighting 方法尝试将它的动态权重增加 $1-score^*$,目的是使子句 c 中出现得分为正的变元,该变元将被加入到 $GoodVars$ 中,从而更可能在后续的迭代中被选择翻转,进而使子句 c 被满足. 为了处理极端情况,即 $1-score^*$ 的值过小或过大,本文设置了参数 h_{min}、h_{max}、s_{min}、s_{max} 来分别限制硬软子句动态权重增加的下限和上限. 具体地,若子句 c 同时

被赋值 α 和 α' 所不满足，则子句 c 的动态权重增量至少为 $min \cdot inc$，至多为 $max \cdot inc$. 若子句 c 仅被赋值 α 所不满足，则其动态权重增量为原始增量 inc.

Hist-Weighting 实际上是一种调整子句动态权重增加量的技术，算法 3 的整体过程就是遍历被当前局部最优解 α 所不满足的子句并确定每条子句动态权重的增量. 在 CCEHC、SATLike(3.0)、BandMaxSAT 等求解 WPMS 的局部搜索算法中，增加子句动态权重的过程即对应于算法 2 中的第 10～15 行. 因此，将 Hist-Weighting 应用于这些算法只要用算法 3 中的操作替换掉算法 2 中的第 10～15 行即可.

4.2　Hist-Weighting 的思想与优点

Hist-Weighting 的主要思想就是综合考虑当前局部最优解以及历史(上一)局部最优解来决定子句动态权重的增加值. 对于同时被当前局部最优解 α 与上一局部最优解 α' 所不满足的子句，Hist-Weighting 尝试增加该子句动态权重使得子句中至少有一个变元的得分为正，从而使该子句尽快地被满足，加速子句动态权重的收敛过程，提高算法的搜索效率，并通过 h_{min}、h_{max}、s_{min}、s_{max} 等参数来限制子句动态权重增量的上下限，避免子句动态权重无法收敛.

由于求解 MaxSAT 问题的局部搜索算法的搜索算子 $flip$ 非常简单，算法的搜索速度非常快，搜索范围也非常广，因此子句满足与不满足的状态变更情况是非常频繁的，仅根据当前局部最优解来判断子句是否难以被满足是不够准确的. Hist-Weighting 结合了当前与历史局部最优解，能够对子句的难满足程度有更准确的把握，从而更准确地调整子句的动态权重，引导算法搜索到更好的解.

此外，在 Hist-Weighting 的设计中既有多样性又有公平性. 多样性体现在每个同时被赋值 α 和 α' 所不满足子句的动态权重增量不是一个固定不变的值，而是根据子句中变元得分的最大值来自适应地确定的，这样的设计机制增加了算法的灵活性. 另一方面，虽然每个这样的子句的动态权重增量不同，但它们的目标是相同的，即使得子句中至少出现一个得分为正的变元. 实际上，这样的机制会使得公式中出现若干得分为 1 的变元，从这样的角度来看算法又具有公平性，并且在后续的迭代过程中，对于这些得分同为 1 的变元，局部搜索算法[24,26-28]都会选择上一次翻转距当前更久远的变元，即选择不被满足的时间更久的子句去满足，具备合理性与直觉性. 实现这一选择的做法为使用一个时间戳数组 $timestamp$ 在每个变元被翻转时记录当前迭代次数，若 $timestamp[x_1] < timestamp[x_2]$ 则说明 x_1 的上一次翻转比 x_2 距当前更久远.

总之，Hist-Weighting 结合了当前与历史信息，兼顾了算法的多样性、灵活性、公平性和合理性. 实验结果表明，Hist-Weighting 是一种能够改进各种最先进的求解 WPMS 问题的局部搜索算法的通用子句加权方法.

5　实验结果与分析

本文选择的基线算法包括三个求解WPMS问题最先进的局部搜索算法，即 CCEHC[27]、SATLike3.0[26] 和 BandMaxSAT[24]. 将应用 Hist-Weighting 于上述三个算法所得到的改进版本分别称为 CCEHC-HW、SATLike3.0-HW、BandMaxSAT-HW. 本节首先介绍实验设置，然后将各基线算法与其改进版本进行对比，最后在各改进版本上做消融实验以分析 Hist-Weighting 中各细节设计的作用.

5.1　实验设置

本文实验中所有算法均由 C++语言实现，所有算法均在 Ubuntu 16.04 Linux 系统，CPU 型号为 Intel Xeon E5-2650 v3@2.30 GHz 的机器上运行. 实验所用算例集为近五年 MaxSAT Evaluation① (MSE)即 MSE2018～MSE2022 的非完备组的所有 WPMS 算例. 将 MSE2022 的非完备组 WPMS 算例集称为 WPMS2022，以此类推. 各算法的截止时间设置为 300 s，与 MSE 非完备组的设置一致. 各算法求解各算例一次，与 MSE 及各基线算法的设置一致.

Hist-Weighting 中所包含的参数包括四个限制子句动态权重增量上下限的参数 h_{min}、h_{max}、s_{min}、s_{max}，它们的调试范围为 $h_{min} \in [1,6]$，$h_{max} \in [2,25]$，$s_{min} \in [1,6]$，$s_{max} \in [2,25]$，采用自动配置工具 SMAC3[29] 进行调参，该调参工具同样被广泛用于基线算法 CCEHC 和 SATLike3.0 中，最终调试出的最优参数设置为 $h_{min}=1$，$h_{max}=22$，$s_{min}=1$，$s_{max}=23$，实验中各基线算法采用原代码中的默认参数运行，各改进版本中除 Hist-Weighting 中的四个参数之外其余参数，如 sp、h_{inc}、s_{inc} 等，与各基线算法一致.

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

5.2 基线算法与改进版本的对比

本小节将各基线算法与其改进版本对比，BandMaxSAT 与 BandMaxSAT-HW、SATLike3.0 与 SATLike3.0-HW、CCEHC 与 CCEHC-HW 在数据集 WPMS2018～WPMS2022 上的对比结果分别如表 1～表 3 所示. 各表中各算法的“获胜个数”为该算法输出解的 *cost* 值(即被该解所不满足的子句的原始权重之和，详见第 2 节)，小于或等于该表中另一算法输出解的 *cost* 值的算例个数，“平均时间”表示求得获胜算例结果的平均计算时间. “获胜个数”这一评价指标也被广泛用于 CCEHC、SATLike3.0、BandMaxSAT 等 MaxSAT 局部搜索算法中. 表中最优的结果以粗体标出.

表 1 BandMaxSAT 与 BandMaxSAT-HW 在各数据集上的结果对比

数据集	算例数	BandMaxSAT		BandMaxSAT-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	93	77.86	**103**	100.78
WPMS2019	297	155	101.19	**191**	109.01
WPMS2020	253	129	112.91	**156**	118.03
WPMS2021	151	63	134.19	**86**	136.64
WPMS2022	197	86	112.61	**102**	125.23

表 2 SATLike3.0 与 SATLike3.0-HW 在各数据集上的结果对比

数据集	算例数	SATLike3.0		SATLike3.0-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	73	80.95	**116**	108.16
WPMS2019	297	143	86.74	**185**	103.25
WPMS2020	253	114	84.49	**160**	111.57
WPMS2021	151	65	122.48	**77**	99.05
WPMS2022	197	77	109.32	**107**	126.93

表 3 CCEHC 与 CCEHC-HW 在各数据集上的结果对比

数据集	算例数	CCEHC		CCEHC-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	69	91.86	**98**	93.53
WPMS2019	297	117	115.60	**152**	119.60
WPMS2020	253	93	112.22	**139**	137.13
WPMS2021	151	47	136.35	**73**	148.90
WPMS2022	197	67	140.84	**97**	148.00

从表 1～表 3 的结果可以看出，Hist-Weighting 子句加权技术能显著改进各种最先进的求解 WPMS 的局部搜索算法即 BandMaxSAT、SATLike3.0 和 CCEHC 的性能，体现出 Hist-Weighting 的通用性和有效性. 在各算例集上，BandMaxSAT-HW 所求得的获胜算例个数约超过 BandMaxSAT 的 11%～37%；SATLike3.0-HW 所求得的获胜算例个数约超过 SATLike3.0 的 18%～59%；CCEHC-HW 所求得的获胜算例个数约超过 CCEHC 的 30%～55%. 可以看出 Hist-Weighting 对 CCEHC、SATLike3.0、BandMaxSAT 算法的改进程度依次递减，这是由于 SATLike3.0 采用的子句加权机制比 CCEHC 更好，而 BandMaxSAT 在 SATLike3.0 的基础上提出了更优秀的初始化方法和变元选择策略，因此 CCEHC、SATLike3.0、BandMaxSAT 三种算法的性能和鲁棒性依次递增，Hist-Weighting 对它们的改进程度也就自然依次递减.

为了更加清晰地对比 Hist-Weighting 对各基线算法在解的优度上的改进程度，本小节选取了 35 个代表性的 WPMS 算例，于表 4 中详细对比了三种基线算法以及对应的 Hist-Weighting 改进版本求解这 35 个算例的 *cost* 值，表中“—”表示算法未在截止时间内求出可行解. 从结果中可以看出，在求解这些算例时，Hist-Weighting 在解优度上对基线算法有显著的提升，对 BandMaxSAT 的提升最多可达 48.9%，对 SATLike3.0 的提升最多可达 37.8%，对 CCEHC 的提升最多可达 42.6%. 此外，在求解许多算例时，Hist-Weighting 甚至能够帮助基线算法找到可行解，说明 Hist-Weighting 确实能够显著提升 WPMS 局部搜索算法的搜索能力.

表 4　各基线算法与对应的 Hist-Weighting 改进版本的详细 *cost* 结果对比

算例名	BandMaxSAT	BandMaxSAT-HW	SATLike3.0	SATLike3.0-HW	CCEHC	CCEHC-HW
aus.formula_0.8_2021_atleast_31_max-4_reduced_incomplete_adaboost_4.wcnf	416	**371**	395	**391**	464	**384**
aus.formula_0.8_2021_atleast_63_max-5_reduced_incomplete_adaboost_3.wcnf	300	**276**	298	**282**	542	**311**
aus.formula_0.8_2021_atleast_63_max-5_reduced_incomplete_adaboost_5.wcnf	578	**445**	461	**428**	—	**504**
aus.formula_0.8_2021_atleast_63_max-5_reduced_incomplete_adaboost_6.wcnf	557	**441**	525	**466**	—	**512**
BrazilInstance6.xml.wcnf	784	**706**	—	**688**	—	—
car.formula_0.8_2021_atleast_15_max-3_reduced_incomplete_adaboost_6.wcnf	1066	**959**	1062	**998**	1290	**1166**
car.formula_0.8_2021_atleast_63_max-5_reduced_incomplete_adaboost_4.wcnf	602	**476**	602	**454**	590	**536**
comp02.wcnf	285	**264**	268	**242**	936	**799**
comp12.wcnf	1085	**908**	1109	**1005**	1407	**1092**
comp15.wcnf	263	**232**	276	**266**	710	**573**
comp16.lp.sm-extracted.wcnf	1003	**917**	1207	**1073**	—	—
comp18.wcnf	247	**223**	252	**241**	405	**362**
instance10.wcnf	6420	**6275**	6719	**6583**	—	**13388**
instance7.wcnf	1150	**1142**	1384	**1169**	3709	**3300**
instance8.wcnf	3973	**3042**	4474	**3658**	11545	**9616**
lisbon-wedding-10-18.wcnf	673	**647**	669	**558**	650	**519**
lisbon-wedding-4-19.wcnf	—	**1480**	—	—	—	**1320**
lisbon-wedding-7-17.wcnf	617	**610**	693	**612**	—	**597**
metro_8_8_5_20_10_6_500_1_3.lp.sm-extracted.wcnf	248	**205**	280	**205**	—	—
MinWidthCB_mitdbsample_300_32_1k_15s_2t_17.wcnf	28060	**27105**	28155	**27225**	31755	**30090**
MinWidthCB_mitdbsample_300_43_1k_15s_1t_15.wcnf	41735	**39210**	41995	**40080**	46630	**45550**
polysite-bloat.wcnf	194	**140**	216	**205**	—	—
role_university_0.55_2.cnf	235364	**222869**	226567	**224581**	588321	**552825**
Rounded_BTWBNSL_insurance_100_1_3.scores_TWBound_2.wcnf	176183588	**174543675**	225421190	**173672379**	179609452	**177042999**
Rounded_CorrelationClustering_Protein4_BINARY_N360.wcnf	17451640	**17243101**	18195509	**16591666**	113351860	**102789595**
Rounded_CorrelationClustering_Protein4_BINARY_N400.wcnf	20817173	**18851644**	20534362	**18937210**	148973794	**145595861**
Rounded_CorrelationClustering_Vowel_BINARY_N760_D0.200.wcnf	110894235	**84220519**	90631116	**83703783**	—	—
simNo_10-s_5-m_100-n_100-fp_0.01-fn_0.20.wcnf	1.87E+15	**9.55E+14**	5.63E+14	**5.57E+14**	2.96E+14	**2.78E+14**
simNo_10-s_5-m_100-n_500-fp_0.01-fn_0.05.wcnf	1.47E+16	**1.47+16**	3.14E+15	**1.95E+15**	1.35E+15	**1.24E+15**
tcp_students_98_it_11.wcnf	2880	**2805**	2844	**2808**	—	**3255**
tcp_students_98_it_14.wcnf	3138	**3102**	3102	**3066**	—	**3327**
test3.lp.sm-extracted.wcnf	664	**410**	657	**547**	—	**1837**
test4.wcnf	—	**452**	—	**478**	—	—
toms_test_1_CNF_3_10.wcnf	278	**202**	271	**259**	384	**360**
twitter_test_7_CNF_4_5.wcnf	566	**466**	502	**480**	691	**652**

为了进一步评估所提出的 Hist-Weighting 子句加权技术在各类 WPMS 算例上的效果，本小节合并了 WPMS2018～WPMS2022 的所有算例，并将重复算例去除，按照算例类别进行分类，且去掉了所有基线算法和改进版本都无法在截止时间内求出类别中任一算例的可行解的算例类别，剩余算例共 27 类、566 个. BandMaxSAT 与 BandMaxSAT-HW、SATLike3.0 与 SATLike3.0-HW、CCEHC 与 CCEHC-HW 在这 27 类算例上的对比结果分别如表 5、表 6、表 7 所示. 在求解某类算例时，获胜个数更多或获胜个数相同，平均时间更少的算法表现更优，这样的评价标准与各基线算法和 MSE 中的规则一致.

表 5 BandMaxSAT 与 BandMaxSAT-HW 在各算例类别上的结果对比

算例类别名	算例数	BandMaxSAT		BandMaxSAT-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
abstraction-refinement	10	5	248.90	5	**227.88**
af-synthesis	33	12	109.38	**25**	149.51
BTBNSL-Rounded	28	**20**	26.00	12	102.15
causal-discovery	24	21	39.52	21	**37.79**
cluster-expansion	20	20	11.90	20	**11.51**
correlation-clustering	46	17	157.42	**30**	179.97
decision-tree	36	12	186.23	**32**	153.03
hs-timetabling	13	2	229.26	**3**	154.89
lisbon-wedding	21	7	**178.06**	7	210.74
maxcut	29	**29**	11.06	28	9.20
max-realizability	13	10	28.10	10	**26.10**
MaxSATQIC	37	13	143.21	**25**	75.59
metro	2	0	0	**2**	234.68
MWDSP	7	4	153.47	4	**129.27**
min-width	45	17	135.83	**28**	162.14
mpe	22	16	99.63	**21**	55.19
railroad_reisch	6	**5**	143.31	2	148.44
railway-transport	4	1	185.72	**2**	146.06
ramsey	12	11	99.44	**12**	96.62
RBAC	79	**48**	167.62	34	129.12
relational-inference	2	0	0	**2**	145.58
scSequencing_Mehrabadi	14	9	167.98	9	**62.66**
set-covering	13	13	109.32	13	**93.25**
spot5	7	3	151.94	**5**	122.87
staff-scheduling	11	2	108.68	**9**	144.56
tcp	13	6	211.74	**9**	145.09
timetabling	19	1	192.84	**14**	185.16
合计	566	304	108.07	**384**	110.61

表 6 SATLike3.0 与 SATLike3.0-HW 在各算例类别上的结果对比

算例类别名	算例数	SATLike3.0		SATLike3.0-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
abstraction-refinement	10	4	185.31	**7**	185.67
af-synthesis	33	15	78.54	**24**	112.81
BTBNSL-Rounded	28	7	26.23	**21**	58.18
causal-discovery	24	22	**19.89**	22	22.73
cluster-expansion	20	16	81.90	**19**	111.66
correlation-clustering	46	11	117.23	**37**	130.91
decision-tree	36	12	142.40	**29**	136.69
hs-timetabling	13	0	0	**5**	132.34
lisbon-wedding	21	**7**	154.93	6	186.41
maxcut	29	28	12.81	28	**6.73**
max-realizability	13	**9**	80.00	8	38.10
MaxSATQIC	37	**21**	114.40	20	153.02
metro	2	1	170.31	1	**50.55**
MWDSP	7	4	139.65	**5**	156.75
min-width	45	14	156.32	**31**	150.99
mpe	22	**12**	96.87	7	162.29
railroad_reisch	6	**4**	41.63	3	23.22
railway-transport	4	**2**	171.03	1	138.99
ramsey	12	12	45.35	12	**35.84**
RBAC	79	**47**	124.57	34	133.64
relational-inference	2	1	182.91	1	**134.06**
scSequencing_Mehrabadi	14	5	171.24	**10**	47.52
set-covering	13	11	77.05	**13**	90.59
spot5	7	1	40.93	**7**	151.14
staff-scheduling	11	5	129.36	**6**	181.10
tcp	13	7	**137.63**	7	174.44
timetabling	19	3	130.20	**12**	143.70
合计	566	281	93.70	**376**	108.25

表 7　CCEHC 与 CCEHC-HW 在各算例类别上的结果对比

算例类别名	算例数	CCEHC		CCEHC-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
aabstraction-refinement	10	0	0	0	0
af-synthesis	33	31	143.27	**33**	121.94
BTBNSL-Rounded	28	7	188.18	**21**	131.24
causal-discovery	24	0	0	0	0
cluster-expansion	20	20	0.27	20	**0.20**
correlation-clustering	46	5	172.31	**22**	221.28
decision-tree	36	5	145.18	**29**	150.03
hs-timetabling	13	0	0	**1**	70.63
lisbon-wedding	21	0	0	**12**	184.58
maxcut	29	28	0.18	**29**	7.79
max-realizability	13	7	54.35	**10**	40.74
MaxSATQIC	37	**25**	75.31	20	83.68
metro	2	0	0	0	0
MWDSP	7	0	0	**1**	299.08
min-width	45	16	125.11	**28**	154.21
mpe	22	14	140.26	**16**	130.41
railroad_reisch	6	2	196.55	**6**	268.37
railway-transport	4	1	129.64	1	**86.18**
ramsey	12	11	50.42	**12**	57.77
RBAC	79	37	190.71	**42**	206.29
relational-inference	2	0	0	0	0
scSequencing_Mehrabadi	14	**8**	192.01	7	150.43
set-covering	13	12	137.14	**13**	130.86
spot5	7	3	49.32	**4**	68.03
staff-scheduling	11	5	**173.75**	5	186.08
tcp	13	1	125.64	**8**	130.72
timetabling	19	0	0	**8**	183.13
合计	566	238	109.42	**348**	128.79

表 5～表 7 的结果显示，在所有 27 类算例中，有 22/5 类BandMaxSAT-HW比BandMaxSAT表现更优/差；有 18/9 类 SATLike3.0-HW比 SATLike3.0 表现更优/差；有 20/3 类 CCEHC-HW 比 CCEHC 表现更优/差. 在所有 566 个算例上，BandMaxSAT-HW 获胜个数超过 BandMaxSAT 的 26%，SATLike3.0-HW 获胜个数超过 SATLike3.0 的 34%，CCEHC-HW 获胜个数超过 CCEHC 的 42%. 以上结果再次表明 Hist-Weighting 优秀的通用性和鲁棒性，能够在求解大部分算例类别时改进三种基线算法的性能.

Hist-Weighting 结合当前局部最优解和历史局部最优解帮助子句加权技术做出更准确的子句动态权重调整，另外使子句中出现得分为正的变量这一机制也能够加快算法的搜索进程，契合现有局部搜索算法中 *GoodVars* 结构的设计. 以上实验结果也表明了 Hist-Weighting 优秀的鲁棒性和通用性，进一步验证了上述思想的正确性.

5.3　消融实验

本小节将各改进版本与一些它们的变体对比进行消融实验，以分析 Hist-Weighting 中各细节设计的合理性和有效性.

首先，为了证明限制子句加权增量参数的有效性，本小节设计了BandMaxSAT-Nolimit、SATLike3.0-Nolimit、CCEHC-Nolimit 三种变体. 它们可分别通过将三种改进算法中的 h_{max} 与 s_{max} 参数设为正无穷大得到（由于 h_{min} 和 s_{min} 的默认参数值为 1，为该参数理论最小值，故未对它们做消融实验）. BandMaxSAT-Nolimit 与 BandMaxSAT-HW、SATLike3.0-Nolimit 与 SATLike3.0-HW、CCEHC-Nolimit 与CCEHC-HW 在各数据集上的对比结果分别如表 8～表 10 所示.

表 8　BandMaxSAT-Nolimit 与 BandMaxSAT-HW 在各数据集上的结果对比

数据集	算例数	BandMaxSAT-Nolimit		BandMaxSAT-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	**114**	85.23	109	90.78
WPMS2019	297	177	95.40	**198**	97.41
WPMS2020	253	**157**	114.94	156	108.79
WPMS2021	151	78	136.16	**84**	134.41
WPMS2022	197	98	107.66	**116**	114.68

表 9 SATLike3.0-Nolimit 与 SATLike3.0-HW 在各数据集上的结果对比

数据集	算例数	SATLike3.0-Nolimit		SATLike3.0-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	101	97.43	**110**	104.72
WPMS2019	297	154	93.68	**187**	104.25
WPMS2020	253	140	101.48	**159**	108.30
WPMS2021	151	72	106.88	**87**	100.24
WPMS2022	197	91	95.18	**113**	110.16

表 10 CCEHC-Nolimit 与 CCEHC-HW 在各数据集上的结果对比

数据集	算例数	CCEHC-Nolimit		CCEHC-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	77	97.03	**98**	97.38
WPMS2019	297	128	105.07	**162**	109.58
WPMS2020	253	103	115.47	**140**	129.15
WPMS2021	151	59	132.03	**68**	136.67
WPMS2022	197	76	137.78	**97**	139.54

从表 8～表 10 的结果可以看出，相比于对应的 Nolimit 变体，Hist-Weighting 的改进版本在各个数据集上都几乎有更好的表现，说明利用 h_{max} 与 s_{max} 参数限制 Hist-Weighting 子句加权增量的上限是有效的，能够使 Hist-Weighting 的子句加权过程更稳定，避免因子句动态权重增量过大而导致的算法性能下降.

另外，为了表明 Hist-Weighting 中子句动态权重增量根据子句中的变元得分而自适应变化这一机制的好处，本小节还设计了 BandMaxSAT-Fix、SATLike3.0-Fix、CCEHC-Fix 三种变体. 它们可分别通过将三种改进算法中的 h_{min} 和 s_{min} 参数设置为与 h_{max} 和 s_{max} 一致得到. 即在 BandMaxSAT-Fix、SATLike3.0-Fix、CCEHC-Fix 中，同时被当前和上一局部最优解所不满足的硬软子句的动态权重增量为固定的值，$h_{max}\cdot h_{inc}$、$s_{max}\cdot s_{inc}$. BandMaxSAT-Fix 与 BandMaxSAT-HW、SATLike3.0-Fix 与 SATLike3.0-HW、CCEHC-Fix 与 CCEHC-HW 在各数据集上的对比结果分别如表 11～表 13 所示.

表 11 BandMaxSAT-Fix 与 BandMaxSAT-HW 在各数据集上的结果对比

数据集	算例数	BandMaxSAT-Fix		BandMaxSAT-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	83	90.74	**112**	101.91
WPMS2019	297	128	101.24	**201**	111.49
WPMS2020	253	108	117.64	**162**	121.04
WPMS2021	151	56	143.20	**90**	136.05
WPMS2022	197	70	150.57	**113**	120.90

表 12 SATlike3.0-Fix 与 SATlike3.0-HW 在各数据集上的结果对比

数据集	算例数	SATlike3.0-Fix		SATlike3.0-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	57	84.93	**124**	112.45
WPMS2019	297	98	85.33	**206**	112.74
WPMS2020	253	88	105.56	**164**	119.42
WPMS2021	151	53	106.74	**87**	107.04
WPMS2022	197	67	109.57	**108**	131.13

表 13 CCEHC-Fix 与 CCEHC-HW 在各数据集上的结果对比

数据集	算例数	CCEHC-Fix		CCEHC-HW	
		获胜个数	平均时间/s	获胜个数	平均时间/s
WPMS2018	172	67	85.21	**87**	83.63
WPMS2019	297	99	111.97	**156**	103.45
WPMS2020	253	85	132.15	**131**	128.90
WPMS2021	151	35	151.04	**75**	141.77
WPMS2022	197	59	143.52	**92**	139.73

从表11～表13的结果中可以看出，Hist-Weighting 的改进版的性能显著优于各对应的 Fix 变体，说明 Hist-Weighting 中子句加权增量通过其变元得分自适应变化的设计是合理且有效的，相比于固定子句加权增量，这样的设计更具灵活性和多样性，能够显著提升算法的性能，帮助算法找到更好的解.

6 结　论

本文提出了一种结合当前与历史信息的通用子句加权技术 Hist-Weighting，以改进求解 WPMS 问题的局部搜索算法. Hist-Weighting 综合考虑了当前与历史局部最优解，能够根据子句中变元的得分自适应地调整子句动态权重的增量，并通过参数限制子句动态权重增量的范围. 在近五届 MaxSAT Evaluation 非完备组所有 WPMS 算例上的实验结果表明，Hist-Weighting 能够显著改进最先进的求解 WPMS 问题的局部搜索算法 BandMaxSAT、SATLike3.0 和 CCEHC，体现出 Hist-Weighting 子句加权技术在求解 WPMS 问题时的有效性.

目前 Hist-Weighting 仅考虑了历史信息中的冰山一角，即上一个局部最优解的信息，但如此简单的设计与改进却带来了很好的改进效果，这也表明考虑历史信息调整子句加权技术是很有潜力的. 未来将进一步思考如何更好、更全面地利用历史信息来设计更优秀的通用子句加权技术.

参 考 文 献

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

[2] Muise C, Beck J C, Mcilraith S A. Optimal partial-order plan relaxation via MaxSAT. Journal of Artificial Intelligence Research, 2016, 57: 113-149

[3] Li Y L, Lin S T, Nishizawa S, et al. MCell: Multi-row cell layout synthesis with resource constrained MAX-SAT based detailed routing//Proceedings of the International Conference on Computer-Aided Design. San Diego, USA, 2020: 1-8

[4] Fu Z, Malik S. On solving the partial MAX-SAT problem//Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing. Seattle, USA, 2006: 252-265

[5] Fang Z, Li C M, Xu K. An exact algorithm based on MaxSAT reasoning for the maximum weight clique problem. Journal of Artificial Intelligence Research, 2016, 55: 799-833

[6] Jiang H, Li C M, Liu Y, et al. A two-stage MaxSAT reasoning approach for the maximum weight clique problem//Proceedings of the AAAI Conference on Artificial Intelligence. New Orleans, USA, 2018: 1338-1346

[7] He Kun, Zou Sheng-Hao, Zhou Jian-Rong. Enumerating maximal cliques in large sparse graphs. Journal of Huazhong University of Science and Technology (Natural Science Edition), 2017, 45(12): 1-6(in Chinese)
(何琨，邹晟昊，周建荣. 大规模稀疏图的极大团枚举算法. 华中科技大学学报(自然科学版)，2017，45(12)：1-6)

[8] Bansal N, Blum A, Chawla S. Correlation clustering. Machine Learning, 2004, 56(1-3): 89-113

[9] Berg J, Järvisalo M. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. Artificial Intelligence, 2017, 244: 110-142

[10] Dinur I, Safra S. On the hardness of approximating minimum vertex cover. Annals of Mathematics, 2005, 162(1): 439-485

[11] Ciampiconi L, Ghosh B, Scarlett J, et al. A MaxSAT-based framework for group testing//Proceedings of the 34th AAAI Conference on Artificial Intelligence. New York, USA, 2020: 10144-10152

[12] Demirovic E, Musliu N. MaxSAT-based large neighborhood search for high school timetabling. Computers & Operations Research, 2017, 78: 172-180

[13] Bofill M, Garcia M, Suy J, et al. MaxSAT-based scheduling of B2B meetings//Proceedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Barcelona, Spain, 2015: 65-73

[14] Bofill M, Giráldez-Cru J, Suy J, et al. A study on implied constraints in a MaxSAT approach to B2B problems//Proceedings of the 22nd International Conference of the Catalan Association for Artificial Intelligence. Mallorca, Spain, 2019: 183-192

[15] Chen Y, Safarpour S, Veneris A G, et al. Spatial and temporal design debug using partial MaxSAT//Proceedings of the 19th ACM Great Lakes Symposium on VLSI. Boston, USA, 2009: 345-350

[16] Chen Y, Safarpour S, Marques-Silva J, et al. Automated design debugging with maximum satisfiability. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2010, 29(11): 1804-1817

[17] Lin P C K, Khatri S P. Application of Max-SAT-based ATPG to optimal cancer therapy design. BMC Genomics, 2012, 13(6): 1-10

[18] Saikko P, Malone B M, Järvisalo M. MaxSAT-based cutting planes for learning graphical models//Proceedings of the International Conference on Integration of Constraint Programming, Artificial Intelligence, and Operations Research. Barcelona, Spain, 2015: 347-356

[19] Morgado A, Liffiton M H, Marques-Silva J. MaxSAT-based MCS enumeration//Proceedings of the 8th International Haifa Verification Conference. Haifa, Israel, 2012: 86-101

[20] Jaroslav B, Kuldeep S M. Counting maximal satisfiable subsets//Proceedings of the 35th AAAI Conference on Artificial Intelligence. New York, USA, 2021: 3651-3660

[21] Gaglione J R, Neider D, Roy R, et al. Learning linear temporal properties from noisy data: A MaxSAT-based approach//Proceedings of the 19th International Symposium on Automated Technology for Verification and Analysis. Gold Coast, Australia, 2021: 74-90

[22] Morris P. The breakout method for escaping from local minima //Proceedings of the 8th National Conference on Artificial Intelligence. Washington, USA, 1993: 40-45

[23] Cha B, Iwama K, Kambayashi Y, et al. Local search algorithms for partial MAXSAT//Proceedings of the 14th National Conference on Artificial Intelligence. Providence, USA, 1997: 263-268

[24] Zheng J, He K, Zhou J, et al. BandMaxSAT: A local search MaxSAT solver with multi-armed bandit//Proceedings of the 31st International Joint Conference on Artificial Intelligence. Vienna, Austria, 2022: 1901-1907

[25] Cai S, Luo C, Zhang H. From decimation to local search and back: A new approach to MaxSAT//Proceedings of the 26th International Joint Conference on Artificial Intelligence. Melbourne, Australia, 2017: 571-577

[26] Cai S, Lei Z. Old techniques in new ways: Clause weighting, unit propagation and hybridization for maximum satisfiability. Artificial Intelligence, 2020, 287: 103354

[27] Luo C, Cai S, Su K, et al. CCEHC: An efficient local search algorithm for weighted partial maximum satisfiability. Artificial Intelligence, 2017, 243: 26-44

[28] Lei Z, Cai S. Solving (weighted) partial MaxSAT by dynamic local search for SAT//Proceedings of the 27th International Joint Conference on Artificial Intelligence. Stockholm, Sweden, 2018: 1346-1352

[29] Lindauer M, Eggensperger K, Feurer M, et al. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. Journal of Machine Learning Research, 2022, 23(54): 1-9

ZHENG Jiong-Zhi, Ph. D. candidate. His research interests include NP-hard problem and combinatorial optimization.

HE Kun, Ph. D., professor. Her research interests include machine learning, adversarial learning, combinatorial optimization, and graph data mining.

Background

Maximum Satisfiability problem (MaxSAT) is an important generalization of a classic NP-complete problem, Satisfiability problem (SAT), in theoretical computer science. MaxSAT aims to finding an assignment that satisfies as many clauses as possible. Weighted Partial MaxSAT (WPMS) is a practical extension of MaxSAT, where clauses are divided into hard and soft, and soft clauses are associated with positive weights. WPMS aims to finding an assignment that satisfies all the hard clauses and maximizes the total weight of satisfied soft clauses. WPMS has a powerful expressive ability and a wide range of applications, such as circuit design, maximum clique, timetabling, planning, etc.

Local search methods are a kind of famous and widely-studied incomplete method for solving the WPMS. Clause weighting techniques are a kind of effective and key technique in WPMS local search, that associates dynamic weights to clauses and update them during the searching process to guide the search directions and help the algorithms escape from local optima. The state-of-the-art WPMS local search algorithms all proposed or applied effective clause weighting techniques, helping them search in different solution spaces. However, existing clause weighting techniques only update the dynamic clause weights according to the current local optimal solution, without considering any historical information.

To handle this issue, this paper proposes a new clause weighting technique, called Hist-Weighting (Clause Weighting with Historical Information), that considers both current and historical information to update the clause dynamic weights, so as to improve the efficiency of the clause weighting schemes and local search algorithms. To evaluate the performance of Hist-Weighting, this paper applies it to three of the state-of-the-art WPMS local search algorithms, BandMaxSAT, SATLike3.0, and CCEHC. Experimental results on all WPMS instances from the incomplete tracks of the last five MaxSAT Evaluations show that compared to the baseline algorithms, the algorithms using Hist-Weighting can increase the number of winning instances by about 10%～60%, demonstrating the effectiveness of the proposed Hist-Weighting clause weighting technique in solving WPMS.

This research was supported by the National Natural Science Foundation of China (No. U22B2017).