

基于动态子句加权策略的MaxSAT问题局部搜索算法研究

张 彬, 赵凯文*

南宁师范大学数学与统计学院, 广西 南宁

收稿日期: 2026年9月11日; 录用日期: 2026年10月13日; 发布日期: 2026年10月26日

摘 要

最大可满足性问题 (Maximum Satisfiability Problem, MaxSAT) 是布尔可满足性问题 (Satisfiability Problem, SAT) 的优化延伸, 是NP难组合优化问题。在理论研究以及工程应用中的诸多优化问题都可以转化并编码为MaxSAT问题, 从而借助其策略开展相关的研究与求解, 是一种具有极其强大表达能力的逻辑约束优化模型。MaxSAT问题主要研究的是, 给定一个命题逻辑合取范式公式 (Conjunctive Normal Form, CNF), 通过局部搜索算法寻找是否存在使其被满足的可行赋值。本研究主要聚焦于非完备算法中的随机局部搜索算法, 并用其对最大可满足性问题进行求解。首先, 详细介绍了SATLike3.0的子句加权方案, 分析SATLike3.0子句加权策略中的硬惩罚增量和软惩罚增量, 该加权策略均为设定的定值较为固化。其次, 基于SATLike3.0子句加权阶段的加权部分, 提出了动态惩罚增量调整的子句加权策略SR-Weighting()。最后, 将硬子句满足率以及本组满足加权条件的软子句原始权重占比, 融合到硬惩罚增量和软惩罚增量中, 使其呈现灵活动态的调整, 形成了高效的随机局部搜索求解器WrSATLike3.0。实验结果表明, 在国际MaxSAT竞赛 (MaxSAT Evaluation, MSE) 2023和2024年的实例中。求解器WrSATLike3.0在成功求解实例个数以及求解质量上显著优于原始求解器SATLike3.0。在WPMS问题中进行对比时, 成功求解实例个数上对比NuWLS以及BandMaxSAT也取得了一定的优势。

关键词

最大可满足问题, 子句加权, 组合优化, 局部搜索

Research on Local Search Algorithm for MaxSAT Problem Based on Dynamic Clause-Weighting Strategy

Bin Zhang, Kaiwen Zhao*

School of Mathematics and Statistics, Nanning Normal University, Nanning Guangxi

*通讯作者。

文章引用: 张彬, 赵凯文. 基于动态子句加权策略的 MaxSAT 问题局部搜索算法研究[J]. 计算机科学与应用, 2026, 16(10): 85-95. DOI: 10.12677/csa.2026.1610313

Abstract

The Maximum Satisfiability Problem (MaxSAT) is an optimized extension of the Boolean Satisfiability Problem (SAT). It is an NP-hard combinatorial optimization problem. Many optimization problems in theoretical research and engineering applications can be transformed and encoded into MaxSAT problems. Relevant research and solving can be carried out using its strategies. It is a logical-constraint optimization model with extremely strong expressive power. For the MaxSAT problem, we are given a propositional logic formula in Conjunctive Normal Form (CNF). Local search algorithms are used to find feasible assignments to satisfy this formula. This study mainly focuses on stochastic local search algorithms among incomplete algorithms. It employs such algorithms to solve the Maximum Satisfiability Problem. First, this paper describes the clause-weighting scheme of SATLike3.0 in detail. It analyzes the hard penalty increment and soft penalty increment in the clause-weighting strategy of SATLike3.0. These values are preset fixed numbers. The weighting strategy is rather rigid. Second, based on the weighting component in the clause-weighting stage of SATLike3.0, this paper proposes a clause-weighting strategy with dynamic penalty increment adjustment. Finally, this work integrates the satisfaction rate of hard clauses and the proportion of original weights of soft clauses that satisfy the weighting conditions in the current group into the hard penalty increment and soft penalty increment. These two increments can realize flexible and dynamic adjustment. An efficient stochastic local search solver called WrSATLike3.0 is developed. The experimental results show that the test instances come from MaxSAT Evaluation (MSE) 2023 and 2024. The solver WrSATLike3.0 significantly outperforms the original solver SATLike3.0 in the number of successfully solved instances and solution quality. For WPMS problem benchmarks, WrSATLike3.0 also achieves certain advantages in the number of solved instances. The comparison targets include NuWLS and BandMaxSAT.

Keywords

Maximum Satisfiability Problem, Clause-Weighting, Combinatorial Optimization, Local Search

Copyright © 2026 by author(s) and Hans Publishers Inc.

This work is licensed under the Creative Commons Attribution International License (CC BY 4.0).

<http://creativecommons.org/licenses/by/4.0/>



Open Access

1. 引言

最大可满足性问题(Maximum Satisfiability Problem, MaxSAT) [1]是约束满足问题(Constraint Satisfaction Problem, CSP) [2]中备受关注的研究邻域之一。CSP 的一般形式为: 给定一组含有 n 个变元的集合 $D = \{x_1, x_2, \dots, x_n\}$ 以及 m 个约束条件 $\{C_1, C_2, \dots, C_m\}$, 判断是否存在一组赋值使得所有约束条件同时成立。最大可满足性问题的核心在于判断命题逻辑合取范式公式(Conjunctive Normal Form, CNF) [3]是否存在使其成立的可行赋值 α 。

将子句划分为硬子句与软子句, 寻找一个赋值在满足所有硬子句的前提下, 最大程度满足软子句的数量, 这类问题被称为部分最大可满足性问题(Partial Maximum Satisfiability Problem, PMS)。将权重与子句进行结合, 寻找一个赋值旨在满足所有的硬子句并最大化满足软子句的权重之和, 这类问题被称为加权部分最大可满足性问题(Weighted Partial Maximum Satisfiability Problem, WPMS)。其中, PMS 是 WPMS 的一种特殊形式, 即软子句权重均被设置为 1 的情形。MaxSAT 问题有着强大的表达能力, 现实世界中

的组合问题通常蕴含着硬子句与软子句, 将此类问题编码为(W)PMS 问题是自然且灵活的。

MaxSAT 问题在组合优化问题[4] [5]、顶点覆盖问题[6]、最大团[7]等问题中有着重要的应用和深远的学术价值。近年来, 对 MaxSAT 问题的研究聚焦在(W)PMS 求解算法中的核心引导方法[8]和 MaxSMT [9]等方面, 相关的研究成果为设计高效的 MaxSAT 求解算法奠定了坚实的理论与实践基础, 并提供了关键的方法指导。

2. 相关工作

完备算法与非完备算法是当前求解 MaxSAT 问题的主流算法。完备算法能够遍历整个解空间找到问题所有的解, 若存在解, 则一定能够确保解的最优性, 当问题的实例规模过大或对时效性具有严苛要求时, 此时性能就会受到制约。相较而言, 非完备算法中的随机局部搜索(Stochastic Local Search, SLS) [10], 虽然无法保证在限定的时间内提供问题的最优解, 但在问题有解的情况下往往能够在较短的时间内提供高质量的可行解, 有效改进了求解的效率, 因此近年来深受学界的关注。

目前对 MaxSAT 问题的研究主要集中于初始化(子句惩罚权重初始化[11]或变元初始赋值[12]), 子句加权[13]以及变元选择[14]这三个核心阶段。SLS 算法在 MaxSAT 问题的求解上虽已取得显著进展, 但现有局部搜索中, 子句加权的研究仍是一大挑战。本研究对子句的结构特征以及子句的原始权重进行分析, 并设计出基于 MaxSAT 问题的 SLS 求解算法 WrSATLike3.0。

3. 贡献

(1) 本研究针对 SATLike3.0 [13]子句加权方案中的加权部分做出了进一步的改进。提出了子句加权动态惩罚增量调整策略 SR-Weighting(), 当局部搜索算法陷入局部最优时, 我们调用新的子句加权方案 SR-Weighting()。

(2) 其次, 我们依托 2023 和 2024 年 MaxSAT Evaluation 不完备赛道的 4 组基准集, 将子句加权方案 SR-Weighting()嵌入 SATLike3.0 中, 得出了新的改进求解器 WrSATLike3.0。

(3) 实验对比结果表明该求解器成功求解的实例个数以及求解质量, 显著优于原始求解器 SATLike3.0。并且, 在与优于原始求解器 SATLike3.0 的求解器 NuWLS [11]和 BandMaxSAT [15]进行 WPMS 问题的对比时, 也得到了一定的改进。

4. 预备知识

若 $V = \{x_1, x_2, \dots, x_j, \dots, x_n\}$ 为一组 n 个布尔变元的集合, 那么变元 x_j 及其否定形式 $\neg x_j$ 构成了这组布尔变元的文字集 $L = \{x_1, \neg x_1, x_2, \neg x_2, \dots, x_n, \neg x_n\}$ 共包含 $2n$ 个元素。每个子句 C_i 则是若干文字的析取构成, 即 $C_i = x_{i1} \vee x_{i2} \vee \dots \vee x_{ik}$ 。若子句 C_i 的总个数为 m 个, 那么 $i \in \{1, 2, \dots, m\}$, $1 \leq k \leq 2n$ 为子句包含文字的个数, 或为子句的长度。命题逻辑合取范式公式 CNF, 则是 m 条子句的合取构成, 即 $F = C_1 \wedge C_2 \wedge \dots \wedge C_m$ 。一个赋值 α 表示为每个布尔变元赋予一个布尔值, 即映射 $\alpha: x_j \rightarrow \{0, 1\}$, $j \in \{0, 1, 2, \dots, n\}$ 。若在该赋值 α 下, 子句 $C_i = x_{i1} \vee x_{i2} \vee \dots \vee x_{ik}$ 中至少存在一个文字为真, 则称该子句被满足, 否则该子句不被满足。

在(W)PMS 问题中, 若赋值 α 使得 CNF 公式中所有的硬子句 C^H 被满足, 即 $\forall C_i \in C^H, \alpha(C_i) = 1$, 则称该赋值为可行赋值或可行解, 否则为不可行解。衡量一个赋值的质量通常使用该赋值的代价 cost 进行评估。若赋值 α 是一个可行解, 那么该赋值的代价为不被满足软子句 C^S 的原始权重 w_{ori} 之和, 如下所示。

$$\text{cost}(\alpha) = \sum_{C_i \in C^S, \alpha(C_i)=0} w_{\text{ori}}(C_i) \quad (1)$$

若 α 是不可行解, 则 $\text{cost}(\alpha) = +\infty$ 。如果两个赋值 α_1 和 α_2 的代价大小比较为 $\text{cost}(\alpha_1) < \text{cost}(\alpha_2)$, 则表示赋值 α_1 的质量要优于赋值 α_2 。

局部搜索过程中, 翻转变元指的是改变变元 x_j 的布尔值, 它是生成邻域解的基础操作, 用 $\text{flip}(x_j)$ 表示。若当前变元 x_j 的布尔值为 1, 那么 $\text{flip}(x_j)=0$; 若当前变元 x_j 的布尔值为 0, 那么 $\text{flip}(x_j)=1$ 。在翻转变元时, 通常使用得分 score 来衡量翻转的质量, 即 $\text{score}(x_j)=\text{make}(x_j)-\text{break}(x_j)$, 其中 $\text{make}(x_j)$ 表示翻转变元 x_j 后, 子句由不满足变为满足的总权重之和; $\text{break}(x_j)$ 表示翻转变元 x_j 后, 子句由满足变为不满足的总权重之和。这表明选择得分较大的变元进行翻转, 能够使得 CNF 公式中不被满足子句的权重之和减小, 进一步降低解的代价。

在搜索过程中, 变元集合 GoodVars 则是用于存储得分大于 0 的优质变元, 即 $\text{GoodVars}=\{x_j \mid \text{score}(x_j)>0\}$ 。在该集合中随机选取一个变元进行翻转, 都能够不同程度的降低子句的权重之和, 并使得算法往有利的方向进行搜索。当集合 $\text{GoodVars}=\emptyset$ 时, 表明算法陷入局部最优。

5. SATLike3.0 子句加权方案

当算法陷入局部最优时, 通常采用子句加权策略进行惩罚权重的加权或平滑处理, 此操作过程有助于提升算法的求解效率。目前已有多种有效且不同的子句加权方案用于求解(W)PMS 问题。本小节主要针对 SATLike3.0 [13]的子句加权方案进行改进, 并通过实验验证改进思想以及方案的有效性。因此, 接下来主要详细介绍 SATLike3.0 的子句加权方案, 其主要操作过程与步骤如下算法 1 所示。

(1) 在 WPMS 问题中, 每个硬子句的惩罚权重初始化为 1, 即 $\forall C_i \in C^H, p(C_i)=1$; 其次对于每个软子句的惩罚权重, 初始化为它自身所带的原始权重, 即 $\forall C_i \in C^S, p(C_i)=w_{\text{ori}}(C_i)$ (算法 1 第 1~3 行)。在 PMS 问题中, 所有的硬子句和软子句, 它们的惩罚权重初始化为 1, 即 $\forall C_i \in C^H \cup C^S, p(C_i)=1$ (算法 1 第 4~5 行)。

(2) 对不被满足的硬子句和软子句, 以概率 sp 增加他们的惩罚权重。首先, 对每个不被满足的硬子句, 在它们现有的硬惩罚权重基础之上增加硬惩罚增量 δ_h ; 其次, 对每个不被满足的软子句, 如果它们的软惩罚权重值小于设定的软惩罚阈值, 即 $p(C_i)<\varpi, C_i \in C^S, \text{st } \alpha(C_i)=0$, 那么在它们现有的软惩罚权重基础之上增加软惩罚增量 δ_s (算法 1 第 7~12 行)。

(3) 对已被满足的硬子句和软子句, 以概率 $1-sp$ 平滑它们的惩罚权重。首先, 对于每个已满足的硬子句, 如果对应的硬惩罚权重大于硬惩罚增量, 即 $p(C_i)>\delta_h$, 那么平滑该硬子句的惩罚权重为 $p(C_i)=p(C_i)-\delta_h$; 其次, 对每个已满足的软子句, 如果对应的软惩罚权重大于软惩罚增量, 即 $p(C_i)>\delta_s$, 那么平滑该软子句的惩罚权重为 $p(C_i)=p(C_i)-\delta_s$ (算法 1 第 13~19 行)。

在该子句加权的方案中, sp 、 δ_h 、 δ_s 、 t 和 ϖ 等参数, 均为 SATLike3.0 [13]所设定。

算法 1 SATLike3.0 子句加权策略

input : (W)PMS 实例 F , 当前赋值 α , 概率 sp , 参数 ϖ , 增量 δ_h, δ_s
output: 更新过后的子句惩罚权重 p
 // 初始化惩罚权重
 1 **if** $F \in \text{WPMS}$ **then**
 2 $p(C_i) \leftarrow 1, \forall C_i \in C^H$;
 3 $p(C_i) \leftarrow w_{\text{ori}}(C_i) \forall C_i \in C^S$;
 4 **else**
 5 $p(C_i) \leftarrow 1, \forall C_i \in C^H \cup C^S$;
 6 $r \leftarrow$ 生成随机概率;

```

7 if  $r < sp$  then
    // 以概率  $sp$  增加不满足子句的权重
8 foreach  $C_i \in C^H, st \alpha(C_i) = 0$  do
9      $p(C_i) \leftarrow p(C_i) + \delta_h$ ;
10 foreach  $C_i \in C^S, st \alpha(C_i) = 0$  do
11     if  $p(C_i) < \varpi$  then
12          $p(C_i) \leftarrow p(C_i) + \delta_s$ ;
13 else
    // 以概率  $1-sp$  平滑满足子句的权重
14 foreach  $C_i \in C^H, st \alpha(C_i) = 1$  do
15     if  $p(C_i) > \delta_h$  then
16          $p(C_i) \leftarrow p(C_i) - \delta_h$ ;
17 foreach  $C_i \in C^S, st \alpha(C_i) = 1$  do
18     if  $p(C_i) > \delta_s$  then
19          $p(C_i) \leftarrow p(C_i) - \delta_s$ ;
20 return  $p$ ;

```

6. 动态惩罚增量调整策略

SATLike3.0 [13]子句加权策略中的子句加权部分, 对每一个不满足的硬子句进行加权, 均为在该硬子句惩罚权重的基础上增加硬惩罚增量 δ_h ; 对每一个满足条件的软子句进行加权也是单一的增加软惩罚增量 δ_s 。其中硬惩罚增量 δ_h 和软惩罚增量 δ_s 均为设定的定值, 在搜索过程中一定程度上不具备动态灵活性。通过以上观察, 我们发现在子句加权策略中的加权部分, SATLike3.0 [13]未考虑到当前硬子句的满足率以及当前不被满足且 $p(C_i) < \varpi$ 的软子句原始权重占比。若使得 δ_h 和 δ_s 能够根据当前硬子句的满足率, 以及当前不被满足且 $p(C_i) < \varpi$ 的软子句原始权重占比进行动态的调整。这在一定程度上使得 δ_h 和 δ_s 增量更具有搜索的动态灵活性, 也可减少算法不必要的计算过程, 并提高搜索效率。

通过上述分析与思考, 我们首先引入了硬子句的满足率 h_{sr} 以及针对当前不被满足且 $p(C_i) < \varpi$ 的软子句原始权重占比 λ 。在当前赋值 α 下, 所有已被满足的硬子句数量在所有硬子句数量中的占比, 记为硬子句满足率 h_{sr} 。其次, 在当前赋值 α 下, 针对不被满足的软子句且惩罚权重小于阈值 ϖ , 计算所有这些软子句的原始权重之和为 $\sum_{p(C_i) < \varpi} w_{ori}(C_i)$, $C_i \in C^S, st \alpha(C_i) = 0$, 那么每个满足加权条件的软子句原始权重占比, 为该软子句的原始权重与这部分软子句的原始权重之和的比值, 记为该软子句的原始权重占比 λ , 如下所示。

$$\lambda = \frac{w_{ori}(C_i)}{\sum w_{ori}(C_i)}, C_i \in C^S \text{ st } \alpha(C_i) = 0 \text{ 且 } p(C_i) < \varpi \quad (2)$$

接下来, 分别将 h_{sr} 和 λ 作用于硬惩罚增量 δ_h 以及软惩罚增量 δ_s 。若在硬惩罚增量 δ_h 的基础上直接乘以 h_{sr} , 这会使得硬惩罚增量在硬子句满足率较低的时候变得更低, 在硬子句满足率较高时变得更高, 这使得惩罚权重增量的波动比较大, 且不利于局部搜索。因此, 采用如下方式将 h_{sr} 作用于硬惩罚增量 δ_h , 即 $\delta_h \times \xi$, 其中 $\xi = 1 + (1 - h_{sr})$ 。这使得硬子句满足率较低或较高时, 都能够赋予相对温和的惩罚权重, 并且根据硬子句的满足情况, 实现了惩罚增量的动态调整。

在对软惩罚增量 δ_s 进行调整时, 考虑到硬子句对于整个局部搜索的重要性, 因此取 ξ 和 λ 的最小值 $\min(\xi, \lambda)$, 再作用于软惩罚增量 δ_s , 即 $\delta_s + \min(\xi, \lambda)$, 令 $\beta = \delta_s + \min(\xi, \lambda)$ 。这样的软惩罚权重赋予, 不仅使得算法在陷入不同局部最优下软惩罚增量的动态调整, 而且在局部搜索过程中有助于优先满足硬子句后, 再尽可能的去满足软子句原始权重之和最大化。

对于初始化惩罚权重部分以及子句加权的平滑阶段, 均保持与 SATLike3.0 [13] 的操作一致。下面, 具体给出基于 SATLike3.0 子句加权策略在加权阶段的改进。当局部搜索算法陷入局部最优时, 以概率 sp 增加不被满足子句的惩罚权重, 分为不被满足的硬子句以及不被满足的软子句, 子句加权过程如下所示。

(1) 对不被满足的硬子句 $C_i \in C^H$ st $\alpha(C_i) = 0$ 按如下方式进行子句加权:

$$p(C_i) = p(C_i) + \delta_h \times \xi \quad (3)$$

(2) 对不被满足的软子句 $C_i \in C^S$ st $\alpha(C_i) = 0$ 且 $p(C_i) < \varpi$ 按如下方式进行子句加权:

$$p(C_i) = p(C_i) + \beta \quad (4)$$

上述过程为我们提出的子句加权思想方法, 并命名为 SR-Weighting()子句加权策略。当局部搜索算法陷入局部最优时, 执行 SR-Weighting()子句加权策略。

7. 子句加权改进方案 SR-Weighting()

算法 2 展示了 SR-Weighting()子句加权策略的整体流程, 当局部搜索算法陷入局部最优时, 在子句加权策略中的子句加权部分, 分别对硬子句以及满足加权条件的软子句进行新的惩罚权重增量调整。

算法 2 SR-Weighting()子句加权策略

input : (W)PMS 实例 F , 当前赋值 α , 参数 ϖ , 增量 δ_h , δ_s , 概率 sp

output: 更新后的子句惩罚权重 p

// 初始化惩罚权重;

1 初始化惩罚权重与 SATLike3.0 的操作一致; // 见算法 1 第 1-5 行

// 针对加权部分进行改进;

2 $r \leftarrow$ 生成随机概率;

3 **if** $r < sp$ **then**

 // 以概率 sp 增加不被满足子句的惩罚权重;

4 **foreach** $C_i \in C^H$, st $\alpha(C_i) = 0$ **do**

5 $p(C_i) \leftarrow p(C_i) + \delta_h \times \xi$; // 公式(3)

6 **foreach** $C_i \in C^S$, st $\alpha(C_i) = 0$ **do**

7 **if** $p(C_i) < \varpi$ **then**

8 计算该部分软子句的原始权重之和 $\sum w_{ori}(C_i)$ 以及 λ 和 β

9 $\lambda \leftarrow \frac{w_{ori}(C_i)}{\sum w_{ori}(C_i)}$, $\beta \leftarrow \delta_s + \min(\xi, \lambda)$;

10 $p(C_i) \leftarrow p(C_i) + \beta$; // 公式(4)

11 **else**

 // 以概率 $1-sp$ 平滑被满足子句的惩罚权重;

12 平滑阶段与 SATLike3.0 的操作一致; // 见算法 1 第 13-19 行

13 **return** p ;

首先, 针对子句初始化惩罚权重 $\text{SR-Weighting}()$ 保持与 SATLike3.0 [13] 的操作一致, 引用于算法 1 第 1~5 行的操作过程(算法 2 第 1 行)。

其次, 给定生成随机概率 r (算法 2 第 2 行); 若 $r < sp$ 则以概率 sp 增加不被满足子句的惩罚权重, 对每个不被满足的硬子句按照公式(3)进行加权处理, 对满足加权条件的软子句, 计算出该部分软子句的原始权重之和 $\sum w_{\text{ori}}(C_i)$ 以及 λ 和 β , 并按照公式(4)进行加权处理(算法 2 第 3~10 行)。

$\text{SR-Weighting}()$ 子句加权算法的最后, 以概率 $1-sp$ 平滑已经满足硬子句和软子句的惩罚权重, 平滑阶段 $\text{SR-Weighting}()$ 保持与 SATLike3.0 的操作过程是一致的, 引用于 SATLike3.0 的子句加权算法 1 第 13~19 行的操作过程(算法 2 第 11~12 行)。

8. 创新算法 WrSATLike3.0 核心框架

我们将 SATLike3.0 [13] 中的原始子句加权方案, 替换为新提出的动态惩罚增量调整, 子句加权策略 $\text{SR-Weighting}()$ (算法 2)。具体的 WrSATLike3.0 局部搜索算法, 如算法 3 所示。

算法 3 WrSATLike3.0

input : (W)PMS 实例 F , 截止时间 cutoff

output : 可行赋值 α^* 和该赋值的代价值 cost^* 或 “未找到可行赋值”

```

1  $\alpha^* = \emptyset; \text{cost}^* \leftarrow +\infty$ 
2  $\alpha \leftarrow$  一个完整的初始赋值
3 while 运行时间  $< \text{cutoff}$  do
4   if  $\forall C_i \in C^H, \alpha(C_i) = 1$  并且  $\text{cost}(\alpha) < \text{cost}^*$  then
5      $\alpha^* \leftarrow \alpha; \text{cost}^* \leftarrow \text{cost}(\alpha)$ 
6   if  $\text{GoodVars} \neq \emptyset$  then
7      $x \leftarrow$  使用 BMS 策略从  $\text{GoodVars}$  中选取变元  $x$ ;
8   else
9     执行  $\text{SR-Weighting}()$  子句加权策略; // 见算法 2(本文创新点)
10     $x \leftarrow$  根据变元的得分或年龄, 从随机选择未满足子句中选取一个变元;
11     $\alpha \leftarrow \text{flip}(x)$  翻转选取变元的布尔值以获取新的赋值;
12     $\alpha^* \leftarrow \alpha$ , 如果  $\text{cost}(\alpha) < \text{cost}(\alpha^*)$ ;
13  if  $\alpha^*$  是可行赋值 then
14    return  $(\alpha^*, \text{cost}^*)$ ;
15  else
16    return “未找到可行赋值”

```

首先将迄今为止找到的最优赋值 α^* 设置为空集, 以及它的代价值 cost^* 设置为正无穷, 利用单元传播 [16] 生成初始赋值 α (算法 3 第 1~2 行)。当运行的时间小于截止时间时, 算法进入随机局部搜索(算法 3 第 3~12 行)。在搜索过程中, 最优可行赋值 α^* 及其代价值 cost^* 的更新, 包括 $\text{GoodVars} \neq \emptyset$ 时的变元挑选与 SATLike3.0 [13] 的方法一致。

当局部搜索算法陷入局部最优时(即 $\text{GoodVars} = \emptyset$ 时), 此时调用子句加权策略 $\text{SR-Weighting}()$ 见算

法 2 (本章的创新点)。通过 SR-Weighting() 子句加权策略调整子句惩罚权重后, 根据 BMS (Best from Multiple Selections) 采样策略, 从 GoodVars 集合中随机选取 t 个变元 (t 是 BMS 的参数), 然后从中选择得分最高的变元 (算法 3 第 8~10 行)。然后将选取的变元进行翻转 $\text{flip}(x)$ 获得新赋值 α , 此时若 $\text{cost}(\alpha) < \text{cost}(\alpha^*)$, 那么更新最优赋值 α^* (算法 3 第 11~12 行)。

最后当运行达到截止时间后, 若 α^* 是可行赋值则返回 α^* 和 cost^* ; 否则返回 “未找到可行赋值” (算法 3 第 13~16 行)。

9. 实验结果与分析

我们将改进求解器 WrSATLike3.0 分别与 NuWLS [11]、BandMaxSAT [15] 以及原始求解器 SATLike3.0 [13] 三款主流求解器进行了实验的对比与分析。这三个求解器代表了 (W)PMS 领域中当前最先进的局部搜索求解器。其中, 在相关的评测场景中, BandMaxSAT 求解器的求解性能要优于 SATLike3.0 求解器, 而 NuWLS 的性能要优于 BandMaxSAT。NuWLS 在 2022 年 MaxSAT 评测 (MSE) 的不完备赛道中获得了 (W)PMS 冠军。

9.1. 实验设置

本文实验中所涉及到的所有算法均由 C++ 语言实现, 并利用 g++ (版本 11.4.0) 进行编译。实验均在一台配备 Intel Core i7-10875H 八核 CPU 和 8 GB 内存、操作系统为 Ubuntu 22.04.3 LTS 的服务器上运行。每个实例所测评的要求均与 MSE 竞赛非完备组的设置要求一致, 设定 60 秒以及 300 秒的截止时间。求解器在每个实例上运行 1 次, 与 MSE 的各基线算法设置要求是一致的, 每次运行后记录运行找到的最优赋值和此时的运行时长。通过三项不同的指标, 对比此四个求解器在不同基准上的性能。

(1) 求解器在测试实例集上成功求解实例个数, 在这里简记为 #win;

(2) 在找到最优赋值的实例中, 记录寻找到最优赋值的平均时间, 即所有最优赋值的总时长与最优赋值个数的比值, 这里简记为 #time;

(3) 求解出最优赋值的个数, 即在求解的同一个实例集下, 求解出代价值最小的实例个数, 用 #qa 进行表示。

所测试的基准集总实例个数记为 “#ins”, 对比表格中所展示加粗的数值结果表示对应基准的最优结果。所有测试的基准集均来自 MaxSAT 竞赛网, 分别为 PMS_2023、WPMS_2023、PMS_2024、WPMS_2024, 分别代表 2023、2024 年的 (W)PMS 实例集 (已剔除只含软子句的实例)。

WrSATLike3.0 的参数设置与 SATLike3.0 [13] 保持一致, 且没有额外的参数设定。其中 sp 、 δ_h 、 t 和 ϖ 的参数设定如下所示。

(1) 对于 PMS 的测试基准集: 若变元数量 $|V| < 1100$, 则 $sp = 0.01$ 、 $\delta_h = 1$ 、 $t = 15$ 、 $\varpi = 1$; 否则 $sp = 3 \times 10^{-6}$ 、 $\delta_h = 1$ 、 $t = 42$ 、 $\varpi = 400$;

(2) 对 WPMS 的测试基准集: 若软子句的平均惩罚权重值 $\bar{w}(C^S) < 1 \times 10^4$, 则 $sp = 0.01$ 、 $\delta_h = 3$ 、 $t = 5$ 、 $\varpi = 1$; 否则 $sp = 0.01$ 、 $\delta_h = 300$ 、 $t = 15$ 、 $\varpi = 500$ 。

基于动态惩罚增量调整策略, 是针对 SATLike3.0 子句加权过程中加权阶段的惩罚增量 δ_h , δ_s , 通过新定义的 λ 和 h_{sr} 进行动态调整。没有改变 SATLike3.0 的参数设置, 沿用 SATLike3.0 的参数设置旨在通过实验验证动态惩罚增量策略的可靠性。

9.2. WrSATLike3.0 与主流 SLS 求解器的对比分析

表 1 将 WrSATLike3.0 与原始求解器 SATLike3.0 [13] 单独的进行了实验对比分析, 以直观展示改进

后的性能提升, 即在成功求解实例个数#win 和求解质量#qa 上的大幅提升。结果显示改进后的求解器 WrSATLike3.0, 在实例成功求解个数#win 上以及求解质量#qa 上, 较原始求解器 SATLike3.0 均取得显著优势。与此同时, 实验对比结果进一步验证了动态惩罚增量调整策略的可行性和有效性。

由于 NuWLS 的求解性能要优于 BandMaxSAT, 而 BandMaxSAT 的求解性能要优于 SATLike3.0, WrSATLike3.0 则是本章针对 SATLike3.0 进行的改进求解器。基于求解器性能的差异, 表格 2 给出了 WrSATLike3.0 求解器与这两款主流 SLS 求解器 NuWLS 和 BandMaxSAT, 针对 WPMS 问题在成功求解实例个数#win 上的对比结果。结果显示改进求解器 WrSATLike3.0, 要优于比原始求解器 SATLike3.0 更好的求解器 NuWLS 和 BandMaxSAT, 即在实例成功求解个数#win 上取得优势。详细分析过程如下所示。

在表 1 中, 当运行截止时间设置为 60 s 时, WrSATLike3.0 在 PMS_2023、PMS_2024、WPMS_2023 和 WPMS_2024 四个基准集上的成功求解实例个数#win 均优于原始求解器 SATLike3.0, 分别多出 2 个、2 个、1 个和 5 个; 其次, 分别保持有 53 个、81 个、46 个和 86 个求解质量#qa 一致的前提下, 同时分别提升求解质量#qa 个数为 26 个、25 个、4 个和 43 个, 较 SATLike3.0 求解质量提升了 49.06%、30.86%、8.70%和 50%。特别地, 在 PMS_2023 这个测试基准集上, WrSATLike3.0 不仅提高了成功求解实例个数以及求解质量, 而且在平均时间#time 上达到最低。

Table 1. Comparison between WrSATLike3.0 and the original solver SATLike3.0
表 1. WrSATLike3.0 与原始求解器 SATLike3.0 的比较

ins_class	#ins	SATLike3.0			WrSATLike3.0		
		#win	#qa	#time	#win	#qa	#time
60 s							
PMS_2023	172	96	53	26.393	98	79	25.114
PMS_2024	200	126	81	15.174	128	106	16.549
WPMS_2023	140	78	46	23.404	79	50	25.058
WPMS_2024	218	147	86	14.405	152	129	21.888
300 s							
PMS_2023	172	107	74	139.188	104	68	118.846
PMS_2024	200	133	96	77.927	135	103	85.127
WPMS_2023	140	80	56	134.159	80	41	127.122
WPMS_2024	218	150	104	113.547	155	115	106.251

当运行截止时间设置为 300 s 时, WrSATLike3.0 在 WPMS_2023 这个测试基准集上的成功求解实例个数#win 与 SATLike3.0 均为 80 个, 在 PMS_2023 基准集上略低于 SATLike3.0。但在(W)PMS_2024 这两个基准集上, WrSATLike3.0 表现出了成功求解个数#win 以及求解质量#qa 上的优势。其中, 在 PMS_2024 和 WPMS_2024 两个基准集上, 成功求解实例个数#win 较 SATLike3.0 分别提升了 2 个和 5 个; 求解质量#qa 保持 96 个和 104 个一致的前提下, 分别提升了 7 个和 11 个, 较 SATLike3.0 求解质量#qa 提升 7.29%、10.58%。特别地, 在 WPMS_2023 这个测试基准集上, WrSATLike3.0 不仅提高了成功求解实例个数以及求解质量, 而且在平均时间#time 上达到最低。

Table 2. Comparison between WrSATLike3.0 and other SLS solvers
表 2. WrSATLike3.0 与其他 SLS 求解器的比较

ins_class	#ins	NuWLS		BandMaxSAT		WrSATLike3.0	
		#win	#time	#win	#time	#win	#time
60 s							
WPMS_2023	140	83	28.381	77	25.383	79	25.821
WPMS_2024	218	150	28.882	143	18.819	152	19.609
300 s							
WPMS_2023	140	84	114.977	79	137.208	80	124.002
WPMS_2024	218	152	100.660	151	80.659	155	85.254

在表 2 中, 当运行截止时间设置为 60 s 时, WrSATLike3.0 在 WPMS_2024 基准集上的成功求解实例个数#win 以 2 个的优势优于 NuWLS, 在 WPMS_2023 基准集上低于 NuWLS 的成功求解个数; 其次, 与 BandMaxSAT 求解器进行对比时, 在 WPMS_2023 和 WPMS_2024 两个基准集上的成功求解实例个数#win, WrSATLike3.0 均优于求解器 BandMaxSAT, 分别多出 2 个和 9 个。当运行截止时间设置为 300 s 时, WrSATLike3.0 在 WPMS_2024 基准集上的成功求解实例个数#win 以 3 个的优势优于 NuWLS, 在 WPMS_2023 基准集上低于 NuWLS 的成功求解个数; 其次, 与 BandMaxSAT 求解器进行对比时, 在 WPMS_2023 和 WPMS_2024 两个基准集上的成功求解实例个数#win, WrSATLike3.0 均优于求解器 BandMaxSAT, 分别多出 1 个和 4 个。在 WPMS 问题共四个测试基准集中, 改进的 WrSATLike3.0 求解器与 NuWLS 进行对比时, 取得了两个基准集的获胜; 与 BandMaxSAT 进行对比时, 在四个基准集上均取得了获胜。

10. 总结与展望

本文进一步研究了 SATLike3.0 子句加权策略中的加权阶段, 针对原有硬惩罚增量和软惩罚增量进行了动态的调整, 提出了动态惩罚增量调整的子句加权策略 SR-Weighting(), 以及新的局部搜索算法 WrSATLike3.0。当搜索陷入局部最优时, 将当前赋值下的硬子句满足率以及满足加权条件的软子句原始权重占比, 与原有硬惩罚增量和软惩罚增量进行结合。使之在不同局部最优下, 根据不同硬子句满足率和软子句原始权重占比, 进行动态灵活的调整。最后进行实验的对比与分析, 结果表明在 2023 和 2024 年 MaxSAT 评测中, 该策略有效改进了原始求解器 SATLike3.0, 在成功求解的实例个数以及求解质量上相对于原算法取得显著的提升。其次, 在与优于 SATLike3.0 的主流局部搜索求解器 NuWLS 和 BandMaxSAT 进行对比时, 取得了针对 WPMS 问题求解的获胜。这在一定程度上, 对未来 MaxSAT 问题的求解提供了方案。

未来我们将继续深入探讨如下问题。基于 BandMaxSAT 算法, 将我们的子句加权方案 SR-Weighting() 应用于 BandMaxSAT 算法中, 并继续分析 BandMaxSAT 算法中关于子句选择策略对解质量的影响, 设计出基于权重影响子句选择的有效算法, 并应用于解决实际问题。其次, 利用变元与变元之间、变元与子句之间的结构信息, 若某一个变元存在于多个子句中, 则优先选择该变元, 以此设计出局部搜索算法中的变元选择方案。释放 SR-Weighting()子句加权策略的潜力。

基金项目

有关特殊交换群上 Davenport 常数与广义 EGZ 常数的研究 (0300001348)。

参考文献

- [1] 何琨, 郑迺之. 最大可满足性问题的算法研究综述[J]. 华中科技大学学报(自然科学版), 2022, 50(2): 82-95.
- [2] 牛鹏飞, 王晓峰, 芦磊, 等. 随机约束满足问题相变研究综述[J]. 计算机工程与科学, 2022, 44(7): 1321-1330.
- [3] 赵星宇, 王晓峰, 庞立超, 等. 可满足性问题研究进展[J]. 计算机应用与软件, 2025, 42(10): 13-23+52.
- [4] 朱叶, 丁苍峰, 曹博浩, 等. 面向组合优化问题的图神经网络研究进展[J]. 计算机科学与探索, 2026, 20(2): 367-385.
- [5] Wen, J., Li, Y., Selman, B. and He, K. (2025) LocalEscaper: A Weakly Supervised Framework with Regional Reconstruction for Scalable Neural TSP Solvers. arXiv:2502.12484.
- [6] Cai, S., Su, K. and Chen, Q. (2010) EWLS: A New Local Search for Minimum Vertex Cover. *Proceedings of the AAAI Conference on Artificial Intelligence*, **24**, 45-50. <https://doi.org/10.1609/aaai.v24i1.7539>
- [7] Fan, Y., Li, N., Li, C., Ma, Z., Latecki, L.J. and Su, K. (2017) Restart and Random Walk in Local Search for Maximum Vertex Weight Cliques with Evaluations in Clustering Aggregation. *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, Melbourne, 19-25 August 2017, 622-630. <https://doi.org/10.24963/ijcai.2017/87>
- [8] Pan, S., Wang, Y. and Cai, S. (2025) An Efficient Core-Guided Solver for Weighted Partial MaxSAT. *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence*, Montreal, 16-22 August 2025, 2647-2656. <https://doi.org/10.24963/ijcai.2025/295>
- [9] He, X., Li, B., Zhao, M. and Cai, S. (2024) A Local Search Algorithm for MaxSMT(Lia). In: Platzer, A., Rozier, K.Y., Pradella, M. and Rossi, M., Eds., *Lecture Notes in Computer Science*, Springer, 55-72. https://doi.org/10.1007/978-3-031-71162-6_3
- [10] 徐振兴, 何琨, 李初民, 等. 基于搜索信息反馈策略的 MaxSAT 不完备求解算法[J]. 计算机学报, 2023, 46(4): 711-726.
- [11] Chu, Y., Cai, S. and Luo, C. (2023) NuWLS: Improving Local Search for (Weighted) Partial MaxSAT by New Weighting Techniques. *Proceedings of the AAAI Conference on Artificial Intelligence*, **37**, 3915-3923. <https://doi.org/10.1609/aaai.v37i4.25505>
- [12] Zhang, Z., Zhou, J., Wang, X., Yang, H. and Fan, Y. (2022) Initial Solution Generation and Diversified Variable Picking in Local Search for (Weighted) Partial MaxSAT. *Entropy*, **24**, Article 1846. <https://doi.org/10.3390/e24121846>
- [13] Cai, S. and Lei, Z. (2020) Old Techniques in New Ways: Clause Weighting, Unit Propagation and Hybridization for Maximum Satisfiability. *Artificial Intelligence*, **287**, Article 103354. <https://doi.org/10.1016/j.artint.2020.103354>
- [14] Zheng, J., He, K. and Zhou, J. (2022) Farsighted Probabilistic Sampling: A General Strategy for Boosting Local Search MaxSAT Solvers. arXiv:2108.09988. <http://arxiv.org/abs/2108.09988>
- [15] Zheng, J., He, K., Zhou, J., Jin, Y., Li, C.M. and Many'a, F. (2022) BandMaxSAT: A Local Search MaxSAT Solver with Multi-Armed Bandit. arXiv:2201.05544. <https://arxiv.org/abs/2201.05544>
- [16] Cai, S., Luo, C., Lin, J. and Su, K. (2016) New Local Search Methods for Partial MaxSAT. *Artificial Intelligence*, **240**, 1-18. <https://doi.org/10.1016/j.artint.2016.07.006>