计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(5): 1226−1234, 2025

基于增量信息交互的极小不可满足子集求解算法

蒋璐宇　欧阳丹彤　张　奇　太　然　张立明
（吉林大学计算机科学与技术学院　长春　130012）
（符号计算与知识工程教育部重点实验室（吉林大学）　长春　130012）
（jiangluyu1998@163.com）

Incremental Information Interaction-Based Algorithm for Enumerating MUSes

Jiang Luyu, Ouyang Dantong, Zhang Qi, Tai Ran, and Zhang Liming
（*College of Computer Science and Technology*, *Jilin University*, *Changchun* 130012）
（*Key Laboratory of Symbol Computation and Knowledge Engineering* (*Jilin University*), *Ministry of Education*, *Changchun* 130012）

Abstract　Enumerating minimal unsatisfiable subsets (MUS) is an important issue in theoretical computer science. Given an unsatisfiable Boolean formula, the number of its MUS is exponentially related to the formula's scale. Contemporary methods aim to identify MUSes as many as possible within appropriate time limits. Dealing with the huge search space and choosing a suitable node to expand can markedly reduce the time consumption on shrink and grow operations, thereby the algorithm could obtain better performance. We introduce an incremental information interaction-based MUS solving algorithm, denoted as MARCO-MSS4MUS, which utilizes the duality and complementary relationships among MUS, minimal correction set (MCS), and maximal satisfiable subset (MSS). Based on the framework of MARCO algorithm, the proposed algorithm selects a more suitable node to expand via intersection and union information of previously identified MSSes during the search, i.e., the incremental MSS information is employed as a heuristic for node selection to accelerate the enumeration of MUS. This process also benefits in identifying more MSSes, in turn, the incremental MSS information helps select a better node for next exploration, thereby achieving an interaction of incremental information. Two theorems and two corollaries regarding interactive incremental information are proposed and the feasibility of our MARCO-MSS4MUS algorithm is theoretically analyzed. Experiments on standard MUS benchmark instances show the superiority of the proposed algorithm over state-of-the-art algorithms. Both the enumeration efficiency and the number of enumerated wins of the proposed algorithm are significantly improved compared with existing algorithms.

Key words　minimal unsatisfiable subset (MUS); maximal satisfiable subset (MSS); infeasibility analysis; hitting set; duality

摘　要　极小不可满足子集（minimal unsatisfiable subset, MUS）的求解是理论计算机科学的重要问题. 由于MUS的个数随问题规模呈指数级增长，现有算法致力于在合适的时间限制内求解出尽可能多的MUS. 在庞大的搜索空间中，选择合适的节点来扩展可以大幅减小收缩和扩充操作的时间开销，从而提高算法的求解效率. 提出一种基于增量信息交互的MUS求解算法MARCO-MSS4MUS，利用MUS、极小修正集（minimal correction set, MCS）和极大可满足子集（maximal satisfiable subset, MSS）之间的对偶和互补关

收稿日期：2024-05-21；**修回日期**：2024-09-29
基金项目：国家自然科学基金项目（62076108, 61872159）；吉林省教育厅科学研究项目（JJKH20211106KJ, JJKH20211103KJ）
This work was supported by the National Natural Science Foundation of China (62076108, 61872159) and the Scientific Research Project of Education Department of Jilin Province (JJKH20211106KJ, JJKH20211103KJ).
通信作者：张立明（limingzhang@jlu.edu.cn）

系，在采用 MARCO 算法框架增量求解 MSS 和 MUS 的过程中，根据已求解的 MSS 的交集和并集信息辅助选择节点来扩展，即通过增量的 MSS 信息启发用于扩展节点选择以加速 MUS 枚举，这一过程同时利于算法找到更多的 MSS，在枚举过程中新识别出的 MSS 又能辅助下一轮扩展节点的选择，从而实现了增量信息的有效交互. 针对交互的增量信息提出 2 个定理及 2 个推论，从理论角度分析了 MARCO-MSS4MUS 算法的可行性，并通过 MUS 标准测试用例上的实验验证了所提算法相较于当前先进算法的优越性，在部分测试用例上的结果显示所提算法的枚举效率和枚举获胜个数较已有算法均有显著的提高.

关键词 极小不可满足子集；极大可满足子集；不可行分析；碰集；对偶性

中图法分类号 TP391

DOI: 10.7544/issn1000-1239.202440326 **CSTR:** 32373.14.issn1000-1239.202440326

极小不可满足子集(minimal unsatisfiable subset, MUS)的求解是布尔可满足性问题(satisfiability, SAT)[1-4]的重要扩展问题之一. 给定一个布尔逻辑公式，当公式被判定为不可满足时，则需要进一步分析出哪些子句之间起了冲突导致公式不可满足. 而这些“冲突”正是该布尔逻辑公式所对应的 MUS. 由于 MUS 是基于数理逻辑表达的，因此该问题广泛应用于采用逻辑语言描述且含有冲突的系统中，其作用在于定位冲突进而解决冲突，使得系统恢复一致性. 在计算机领域，MUS 求解对于电子设计自动化、形式化验证等方向具有重要意义. 例如，在软件工程领域，对一个正在开发的系统进行需求分析时，需要首先将需求使用逻辑语言进行形式化描述，再进行需求的一致性检查，即确保所有需求间具有一致性(可满足)且没有冗余，若检查出不一致(不可满足)或冗余，则最好以最小化的形式向用户展示，以暴露需求中的核心问题，从而对其做出相应的修改使得所有需求可以同时被满足，将需求形式化描述所得公式的 MUS 就是不一致需求的极小集合；在基于模型的诊断问题[5-7]中，当电路的实际输出与期望输出不一致时，就需要对电路中的组件进行故障诊断，将电路的系统描述、组件和输入输出信息形式化描述为一个不可满足的逻辑公式，对该公式求解其 MUS，即对应于电路的极小冲突集，代表电路中存在冲突的组件集合的最小单位，而该公式的极小修正集(minimal correction set, MCS)对应于电路的诊断解，意味着去掉 MCS 中的组件后电路的输出正常，电路的剩余部分具有一致性. 除此之外，MUS 的枚举在很多领域都有广泛的应用，如描述逻辑中的公理定位问题[8-9]、本体中的不一致检测[10-11]、软硬件验证问题[12-13]等.

本文的主要贡献为 3 个方面：

1)所提算法采用了极大可满足子集(maximal satisfiable subset, MSS)的交集和并集作为启发信息，定义了新的启发式帮助指导选择下一次扩展的节点.

2)提出了 2 个定理及其推论. 从理论层面分析了增量式求解 MSS 和 MUS 的过程中，MSS 的交集和并集信息对于枚举 MUS 的效率提升的可行性.

3)提供多种时间限制下的对比实验，在 MUS 标准测试用例上的实验结果表明，本文所提算法 MARCO-MSS4MUS 可以有效利用 MSS 信息以加速对 MUS 的枚举过程，在部分用例上的枚举效率有数倍的提升效果.

1 相关工作

MUS 的枚举算法目前主要分为 2 类：基于碰集对偶性的方法和基于种子缩减的方法. 第 1 类方法主要利用了 MUS 和 MCS 互为碰集的对偶性特点，以及 MSS 与 MCS 互为补集的特点，首先计算出全部的 MSS，再求其补集得到全部的 MCS，最后对全部的 MCS 构成的集合簇求碰集得出全部的 MUS，代表算法有 CAMUS[14-15]，DAA[16]. 第 2 类方法主要依赖对搜索空间的剪枝来识别 MUS，代表算法有 MARCO[17]及其变体[18-20]、TOME[21]算法、ReMUS[22]算法和 UNIMUS[23]算法等. 其中，MARCO 算法利用哈斯图结构，每找到一个节点必会将其收缩成一个 MUS 或扩充成一个 MSS，再对搜索空间中的无解空间进行剪枝；TOME 算法则将哈斯图结构看作纵向的链式结构，找寻每一条链的可满足节点和不可满足节点的交界处，收集这些节点来识别出真正的 MSS 和 MUS；ReMUS 算法则以递归的方式计算当前节点所表示集合的 MUS；UNIMUS 算法则是在原有搜索空间的基础上定义了一个动态的子搜索空间，使用识别出的 MUS 的并集的幂集与未探索空间的交集作为新的搜索空间来选择节点进行扩展. 尽管 ReMUS 和 UNIMUS 展现出良好的求解效率，但这 2 种算法致力于极致的 MUS 枚举，无法识别出任何的 MSS. 尽管 MSS 的计算较 MUS 更容易，但 MSS 的枚举同样有着不可替代的意

义[24-26]. 因此, 本文将继续针对 MARCO 类算法框架提出新的枚举算法, 充分利用搜索过程中识别出的 MSS 交集和并集等信息作为选择扩展节点的启发信息, 辅助算法选择更合适的节点进行扩展, 在此过程中利用增量的 MSS 信息不断优化节点选择, 使得 MSS 和 MUS 之间的信息进行有效交互, 进一步促进 MUS 的枚举.

2 预备知识

本节将首先介绍不可满足的布尔逻辑公式中的 3 种特殊集合——MUS, MSS, MCS 的概念, 以及三者之间的对偶和互补关系. 之后介绍在枚举此类集合的算法中普遍采用的对搜索空间的刻画方法.

2.1 3 种特殊子集及关系

定义 1. MUS[15]. 给定一个不可满足的布尔逻辑公式 F, 称集合 U 是 F 的一个 MUS, 当且仅当 $U \subseteq F$, 且 U 不可满足, $\forall\, c \in U$, $U\backslash\{c\}$可满足.

定义 2. MSS[15]. 给定一个不可满足的布尔逻辑公式 F, 称集合 S 是 F 的一个 MSS, 当且仅当 $S \subseteq F$, 且 S 可满足, $\forall\, c \in F\backslash S$, $S \cup \{c\}$不可满足.

定义 3. MCS[15]. 给定一个不可满足的布尔逻辑公式 F, 称集合 C 是 F 的一个 MCS, 当且仅当 $C \subseteq F$, 且 $F\backslash C$ 可满足, $\forall\, c \in C$, $F\backslash\{C\backslash\{c\}\}$不可满足.

定义 4. 碰集[4]. 称集合 H 是集合簇 Q 的碰集, 当且仅当 H 同时满足: $H \subseteq \bigcup\limits_{Q' \in Q} Q'$; 对于 $\forall Q' \in Q$, $H \cap Q' \neq \varnothing$, 其中 Q'代表集合簇 Q 中的集合.

定义 5. 极小碰集[4]. 称集合簇 Q 的一个碰集 H 是极小的, 当且仅当 H 是 Q 的碰集且 H 的任意真子集都不是 Q 的碰集.

从问题定义中可以看出, MSS 和 MCS 的定义是互补的, 因此布尔逻辑公式 F 的每个 MSS 在 F 下的补集都是 F 的一个 MCS. 此外, MUS 代表了导致公式 F 不可满足的子句的极小集合, 而 MCS 代表从公式 F 中去除该部分后不一致性被修复的子句集合, 故 MCS 和 MUS 都对应着与公式的不可满足性相关的子句集合. 已有文献 [27] 证明了 MUS 与 MCS 之间互为碰集的对偶关系.

2.2 搜索空间的表示

针对 MUS 枚举问题, 给定一个含有 n 个子句的布尔逻辑公式 F, 其幂集就构成了算法的整个搜索空间, 用 1 个节点表示 F 的 1 个子集, 那么搜索空间中就有 2^n 个节点. MARCO 算法采用子句映射的方式, 将每个子句映射为 1 个新的变量, 从而形成 1 个新的布尔逻辑公式 *map* 来刻画搜索空间. 向上剪枝操作 *BlockUp* 和向下剪枝操作 *BlockDown* 就是向 *map* 中添加相应的子句从而对搜索空间进行剪枝. 判断一个节点是否被扩展过只需要对 *map* 中该节点所含子句所对应的映射变量赋值后进行可满足性检查. 公式 *map* 可被可视化为哈斯图(图示为含有 4 个子句的布尔逻辑公式 F 所对应的搜索空间), 如图 1 所示. 可以看出, 相比于传统方式存储 2^n 个节点及节点之间的联系, 使用逻辑公式 *map* 刻画的搜索空间大大减小了内存消耗又保证了搜索效率.

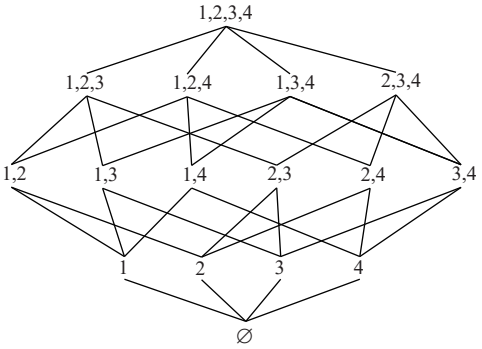


Fig. 1 Hasse diagram

图 1 哈斯图

3 MARCO 算法及已有变体

MARCO 算法作为 MUS 枚举问题的经典算法, 近年来已有多个改进版本. 经典 MARCO 算法的具体描述如算法 1 所示. 首先, 将给定的公式进行子句到变量的映射, 形成一个新的公式 *map* 用于表示搜索空间(行①). 之后开始在搜索空间中寻找 MUS, 执行一个 while 循环, 当哈斯图中还有未探索的节点时, *map* 为可满足的, 循环终止的条件是 *map* 不可满足, 即搜索空间中所有的节点都已探索过(或已被剪枝)(行②). 在循环中, 算法通过调用函数 *getUnexplored*(*map*) 在 *map* 中找到一个未探索的节点 *seed*, 检查其可满足性. 当 *seed* 可满足时, 说明 *seed* 所表示的集合是公式 F 的一个可满足子集, 于是将 *seed* 扩展为一个 MSS, 并将 *seed* 所表示集合的全部子集对应的节点从哈斯图中剪枝; 当 *seed* 不可满足时, 意味着 *seed* 所表示的集合是 F 的一个不可满足子集, 于是将 *seed* 收缩为一个 MUS, 并将 *seed* 所表示集合的全部超集对应的节点剪枝, 重复此过程直到所有节点均被探索或剪枝, 算法停止, 所有的 MUS 和 MSS 被输出.

MARCO 的现有变体主要有 3 种: 第 1 种[18], 对算法 1 的行③进行改进, 更改为选择一个极大的未探

索节点，这样算法更偏向于先搜索不可满足的 *seed*，且这种方式下找到的 *seed* 若可满足则一定是 MSS，可以省去 *Grow* 操作；第 2 种[19]，对循环中每轮选择的 *seed* 个数进行改进，增加一个基数为 *seed* 所代表集合一半的子集对应的节点作为第 2 个 *seed* 参与搜索，这样的改进使得算法偏向于中上层的节点优先探索；第 3 种[20]，找到一类特殊的 MSS 并根据其特性计算出当前可确定的全部 MUS 的子集，将其对应的节点剪枝，这类改进可以进一步对搜索空间进行剪枝，在一定程度上加速对 MUS 的枚举效率.

算法 1. MARCO 算法.

输入：给定的布尔逻辑公式 F；

输出：已识别的 MSS 集合 S_{MSS} 和 MUS 集合 S_{MUS}.

① *map*←*F* 中子句映射为变量形成的新公式；
② while *map* 可满足 do
③ *seed*←*getUnexplored* (*map*);
④ if *seed* 是可满足的
⑤ S_{MSS}←*Grow* (*seed*); /*扩充操作*/
⑥ 输出 S_{MSS};
⑦ *map*←*map* ∧ *BlockDown* (*seed*);
⑧ else /**seed* 不可满足*/
⑨ S_{MUS}←*Shrink* (*seed*); /*收缩操作*/
⑩ 输出 S_{MUS};
⑪ *map*←*map* ∧ *BlockUp* (*seed*);
⑫ end if
⑬ end while

4 MARCO-MSS4MUS 算法

基于第 3 节对 MARCO 已有改进的分析，本文在结合上述改进的基础上提出新的算法 MARCO-MSS4MUS. 本节首先介绍所提算法的思想与理论依据，之后给出算法的详细描述.

由于 MUS, MSS, MCS 这 3 种集合之间均存在紧密的联系，在算法枚举的过程中可以充分利用 MSS 的信息来促进 MUS 的枚举. 具体而言，由于 MSS 与 MCS 互为补集，而 MCS 又与 MUS 互为碰集，于是可以利用 MCS 建立起 MSS 与 MUS 之间信息传递的桥梁. 基于上述思想，结合增量式求解碰集的特点，给出以下定理、推论及证明过程. 其中，定理 1 和定理 2 对应于问题的全部 MSS 和全部 MUS 均被找出时的情况，而推论 1 和推论 2 则对应于部分 MSS 和部分 MUS 被找出时的情况.

定理 1. 令集合 S_{IMSS} 表示所求出的 MSS 的交集，当全部的 MSS 被求出时，出现在集合 S_{IMSS} 中的元素一定不会出现在任何 MUS 中.

证明. 反证法. 假设存在一个元素 $a \in S_{\text{IMSS}}$ 且出现在 MUS 集合 U_1 中. 已知 $a \in U_1$，由于 MUS 与 MCS 互为极小碰集，根据极小碰集的定义，则必有一个 MCS 集合 C_1，满足 $a \in C_1 \cap U_1$，于是一定有 $a \in C_1$. 而已知 $a \in S_{\text{IMSS}}$，于是对任意的 MSS 集合 S，都有 $a \in S$，根据 MSS 与 MCS 互补的性质，对任意的 MCS 集合 C，都有 $a \notin C$，与 $a \in C_1$ 相矛盾. 故原假设不成立，不存在这样的元素 a 既出现在 S_{IMSS} 中又出现在某一 MUS 中. 证毕.

推论 1. 令集合 S_{IMSS} 表示所求出的 MSS 的交集，在增量式得到 MSS 的过程中，当前集合 S_{IMSS} 中的元素一定不会出现在部分 MUS 中.

证明. 反证法. 类似地，假设存在一个元素 $a \in S_{\text{IMSS}}$ 且 a 出现在全部 MUS 中. 根据 $a \in S_{\text{IMSS}}$，有 a 属于当前找到的所有 MSS，又由于 MSS 与 MCS 互补，有 a 不属于当前找到的任意 MCS. 根据 a 出现在全部 MUS 中及 MUS 与 MCS 互为极小碰集的关系，有 $\{a\}$必是一个 MCS，与前面根据 $a \in S_{\text{IMSS}}$ 的条件推断出 a 不属于当前找到的任意 MCS 的结论相矛盾，故原假设不成立. 证毕.

定理 2. 令集合 S_{UMSS} 表示所求出的 MSS 的并集，令集合 S_{CoUMSS} 为 S_{UMSS} 的补集. 当全部的 MSS 被求出时，对任意元素 $u \in S_{\text{CoUMSS}}$，一定存在一个 MUS 集合 U 满足 $u \in U$.

证明. 对于任意元素 $u \in S_{\text{CoUMSS}}$，必有 $u \notin S_{\text{UMSS}}$，于是有 u 不属于任意 MSS 集合 S. 由于 MSS 与 MCS 互为补集，于是有 u 属于所有 MCS 集合 C. 由于 MCS 与 MUS 互为极小碰集，则根据极小碰集的定义，对任意 $u \in C$，必存在一个 MUS 集合 U_1，满足 $u \in U_1$. 证毕.

推论 2. 令集合 S_{UMSS} 表示所求出的 MSS 的并集，令集合 S_{CoUMSS} 为 S_{UMSS} 的补集. 在增量式得到 MSS 的过程中，当前集合 S_{CoUMSS} 中的元素一定出现在部分 MUS 中.

证明. 反证法. 假设存在一个元素 $a \in S_{\text{CoUMSS}}$ 且 a 不会出现在任何 MUS 中. 根据 $a \in S_{\text{CoUMSS}}$，有 $a \notin S_{\text{UMSS}}$，有 a 不属于当前找到的任意 MSS，又由于 MSS 与 MCS 互为补集，有 a 属于当前找到的全部 MCS，又由于 MUS 与 MCS 互为极小碰集，根据碰集定义可知，一定存在一个 MUS 集合 U' 满足 $a \in U'$，与假设中 a 不会出现在任何 MUS 中相矛盾. 故假设不成立. 证毕.

基于定理 1 及其推论 1 对 MSS 交集的理论分析

以及定理 2 和其推论 2 对 MSS 并集的补集的理论分析, 在算法 2 每轮选择扩展节点 *seed* 时, 引入一个新的启发式在原本的 *seed* 基础上做出相应调整: 将 *seed* 与 MSS 的交集即集合 S_{IMSS} 做差集后, 再与 MSS 的并集的补集即集合 S_{CoUMSS} 做并集, 得到新的扩展节点 *seed*′. 判断新节点是否未被探索过: 若未探索, 则扩展 *seed*′而非 *seed*; 否则, 扩展 *seed*. 在实验中发现, 如果 *seed* 与集合 S_{IMSS} 直接做差集, 得到的 *seed*′节点会过小, 位于哈斯图的靠底部位, 使得算法将过于偏向对 MSS 的枚举, 于是调整为 *seed* 与集合 S_{IMSS} 的子集做差集, 子集选择基数为集合 S_{IMSS} 基数 1/3 的任意子集. 引入启发式后的 *seed*′更加符合 MUS 的特点, 从一定程度上减小收缩操作的时间消耗, 更快收缩为一个 MUS.

算法 2 是本文所提算法 MARCO-MSS4MUS 的伪代码描述.

算法 2. MARCO-MSS4MUS 算法.

输入: 给定的布尔逻辑公式 *F*;

输出: 已识别的 MSS 集合 S_{MSS} 和 MUS 集合 S_{MUS}.

① *map*←*F* 中子句映射为变量形成的新公式;
② while *map* 可满足 do
③ *seed*←*getUnexploredMax* (*map*);
④ if *seed* 是可满足的
⑤ S_{MSS}←*seed*;
⑥ 输出 S_{MSS};
⑦ S_{IMSS}←$S_{\mathrm{IMSS}} \cap S_{\mathrm{MSS}}$;
⑧ S_{UMSS}←$S_{\mathrm{UMSS}} \cup S_{\mathrm{MSS}}$;
⑨ *map*←*map* ∧ *BlockDown* (*seed*);
⑩ if *seed* 是关键 MSS
⑪ S_{MCS}←*co* (*seed*);
⑫ *subU*←*subU*∪S_{MCS};
⑬ *map*←*map* ∧ *BlockDown* (*subU*);
⑭ end if
⑮ else /**seed* 不可满足*/
⑯ *seed*′ ←*seed* – *sub* (S_{IMSS})∪*co* (S_{UMSS});
⑰ if *seed*′ 未被探索
⑱ if *seed*′ 可满足
⑲ S_{MSS}←*Grow* (*seed*′);
⑳ 输出 S_{MSS};
㉑ S_{IMSS}←$S_{\mathrm{IMSS}} \cap S_{\mathrm{MSS}}$;
㉒ S_{UMSS}←$S_{\mathrm{UMSS}} \cup S_{\mathrm{MSS}}$;
㉓ *map*←*map* ∧ *BlockDown* (*seed*′);
㉔ else /* *seed*′ 不可满足*/
㉕ S_{MUS}←*Shrink* (*seed*′);
㉖ 输出 S_{MUS};
㉗ *map*←*map* ∧ *BlockUp* (*seed*′);
㉘ end if
㉙ else /**seed*′已被探索*/
㉚ S_{MUS}←*Shrink* (*seed*);
㉛ 输出 S_{MUS};
㉜ *map*←*map* ∧ *BlockUp* (*seed*);
㉝ end if
㉞ end if
㉟ end while

在经典 MARCO 算法的基础上, 行③表示 *seed* 节点的选择方式采用的是已有改进中的第 1 种, 即选择极大未探索的 *seed* 版本; 行⑩~⑭使用的是已有改进中的第 3 种, 找到一类特殊的 MSS 即关键 MSS, 由于关键 MSS 对应的 MCS 为单元素集合, 将关键 MSS 的补集元素加入集合 *subU* 中, 并将 *subU* 对应的节点剪枝. 其中, *co* (*seed*) 指 *seed* 的补集, *sub* (*seed*) 指 *seed* 的子集. 本文的改进主要体现在:

1)算法每找到一个新的 MSS 时, 更新 MSS 交集 S_{IMSS} 和 MSS 并集 S_{UMSS}(行⑦⑧㉑㉒); 2)算法找到的待探索节点 *seed* 不可满足时, 不直接执行下一步收缩操作 *Shrink* (*seed*), 而是通过新的启发式在 *seed* 的基础上先与 S_{IMSS} 的子集作差集, 再与 S_{UMSS} 的补集作并集, 得到新的探索节点 *seed*′(行⑯). 若 *seed*′已探索, 则还是选择 *seed* 来探索; 否则, 使用 *seed*′代替 *seed* 作为探索节点. MARCO-MSS4MUS 算法没有改变经典 MARCO 算法及其变体的剪枝方式, 而是进一步利用了搜索过程中获取的 MSS 信息来改进所选择的探索节点, 从减少收缩和扩充操作的执行时间的角度加速了 MUS 的枚举效率.

5 实 验

本节将所提出的新算法 MARCO-MSS4MUS 与当前最先进的算法 MARCO+[①], MARCO-MAM[②], MARCO-ABC[②]进行对比. 实验设备为一台 60 GB RAM、Intel Xeon E5-2690v4 2.6 GHz 的 CPU、64 位 Ubuntu 18.04 版本 Linux 操作系统的主机. 算法所用的编程语言均

① http://www.iwu.edu/~mliffito/marco/

② https://github.com/jiangluyu1998/ABC-scheme-for-MUS

为 Python 3.7 版本，可满足性检查过程调用的 MiniSAT 求解器[28] 为 C++语言编写. 采用标准测试用例 SAT11 中的 MUS 实例①对算法进行测试，并对每组实验设置了 3 种时间限制，分别为 0.5 h, 1 h, 2 h.

表 1、表 2、表 3 分别对应 0.5 h, 1 h, 2 h 时间限制下各个算法所枚举出的 MUS 个数的对比结果.实验结果显示，在算法执行的前期，所提算法枚举 MUS 的效率在部分实例上并没有已有的先进算法表现好，但随着运行时间的增加，所提算法对 MUS 的枚举效

Table 1 Comparative Results on the Number of MUS of Each Algorithm Under 0.5 h Time Limit

表 1 0.5 h 时间限制下各算法枚举的 MUS 个数对比结果

测试用例	本文算法	MARCO+[18]	MARCO-MAM[19]	MARCO-ABC[20]
fdmus_b14_134	**491**	158	181	173
fdmus_b14_141	53	82	83	**84**
fdmus_b14_210	1	**80**	78	**80**
fdmus_b15_291	**54**	51	52	53
fdmus_b15_294	**51**	47	44	47
fdmus_b15_304	1	32	**33**	31
fdmus_b15_421	1	151	142	**157**
fdmus_b15_463	2	174	**191**	189
fdmus_b17_1 037	1	338	**539**	500
fdmus_b17_1 324	2	58	**77**	57
fdmus_b17_1 456	2	**195**	178	186
fdmus_b17_420	**197**	158	154	161
fdmus_b20_141	1	16	17	**18**
fdmus_b20_171	1	20	19	**21**
fdmus_b20_238	**145**	44	48	45
fdmus_b20_341	**945**	188	190	176
fdmus_b20_349	1	**33**	31	**33**
fdmus_b20_381	1	29	**32**	31
fdmus_b20_492	**215**	38	40	40
fdmus_b20_498	**520**	61	65	65
fdmus_b21_111	1	**14**	13	**14**
fdmus_b21_112	1	14	13	**15**
fdmus_b21_167	1	**35**	31	34
fdmus_b21_481	**115**	31	28	31
fdmus_b21_96	94	115	**118**	**118**
fdmus_b22_102	1	34	**36**	**36**
fdmus_b22_113	1	**12**	**12**	**12**
fdmus_b22_133	1	46	40	**54**
fdmus_b22_172	1	21	22	**23**
fdmus_b22_241	**61**	29	27	30
fdmus_b22_356	1	**14**	**14**	**14**
fdmus_b22_488	**480**	51	47	51
fdmus_b22_600	1	15	14	**20**
fdmus_b22_662	1	**23**	**23**	20

注：黑体数值表示 MUS 枚举个数的最优结果.

Table 2 Comparative Results on the Number of MUS of Each Algorithm Under 1 h Time Limit

表 2 1 h 时间限制下各算法枚举的 MUS 个数对比结果

测试用例	本文算法	MARCO+[18]	MARCO-MAM[19]	MARCO-ABC[20]
fdmus_b14_134	**1 367**	513	574	584
fdmus_b14_141	**484**	199	194	203
fdmus_b14_210	4	**205**	183	204
fdmus_b15_291	**345**	102	109	108
fdmus_b15_294	**190**	100	93	100
fdmus_b15_304	**175**	59	53	63
fdmus_b15_421	**395**	294	305	327
fdmus_b15_463	2	369	**420**	405
fdmus_b17_1 037	938	338	1 082	**1 085**
fdmus_b17_1 324	**188**	117	149	116
fdmus_b17_1 456	2	**386**	383	374
fdmus_b17_420	**537**	318	323	327
fdmus_b20_141	1	32	35	**36**
fdmus_b20_171	1	**38**	35	**38**
fdmus_b20_238	**500**	93	92	92
fdmus_b20_341	**2 424**	629	600	639
fdmus_b20_349	1	66	**68**	63
fdmus_b20_381	1	66	**71**	67
fdmus_b20_492	**806**	88	89	86
fdmus_b20_498	**1 528**	179	176	200
fdmus_b21_111	1	27	**29**	**29**
fdmus_b21_112	1	26	26	**27**
fdmus_b21_167	1	**70**	63	**70**
fdmus_b21_481	**924**	64	63	65
fdmus_b21_96	**987**	300	309	311
fdmus_b22_102	1	63	67	**70**
fdmus_b22_113	1	23	23	**25**
fdmus_b22_133	3	100	98	**108**
fdmus_b22_172	1	**42**	41	**42**
fdmus_b22_241	**313**	60	60	60
fdmus_b22_356	1	**28**	27	27
fdmus_b22_488	**1 524**	110	106	122
fdmus_b22_600	1	26	**34**	29
fdmus_b22_662	1	**47**	44	38

注：黑体数值表示 MUS 枚举个数的最优结果.

① http://www.cril.univ-artois.fr/SAT11/

Table 3 Comparative Results on the Number of MUS of Each Algorithm Under 2 h Time Limit

表 3 2 h 时间限制下各算法枚举的 MUS 个数对比结果

测试用例	本文算法	MARCO+[18]	MARCO-MAM[19]	MARCO-ABC[20]
fdmus_b14_134	**2 852**	1 566	1 486	1 664
fdmus_b14_141	**1 422**	642	598	642
fdmus_b14_210	467	**637**	611	609
fdmus_b15_291	**587**	231	252	253
fdmus_b15_294	**762**	224	208	218
fdmus_b15_304	**468**	143	124	144
fdmus_b15_421	**1 398**	581	689	673
fdmus_b15_463	2	764	787	**823**
fdmus_b17_1 037	**3 593**	338	2 362	2 356
fdmus_b17_1 324	**473**	228	290	229
fdmus_b17_1 456	2	**766**	746	750
fdmus_b17_420	**1 082**	738	664	673
fdmus_b20_141	1	67	70	**71**
fdmus_b20_171	1	**74**	**74**	**74**
fdmus_b20_238	**952**	222	219	236
fdmus_b20_341	**5 424**	1 562	1 496	1 623
fdmus_b20_349	3	141	**157**	149
fdmus_b20_381	3	162	**170**	158
fdmus_b20_492	**3 043**	352	375	322
fdmus_b20_498	**3 091**	552	510	518
fdmus_b21_111	1	55	57	**60**
fdmus_b21_112	1	**53**	52	**53**
fdmus_b21_167	24	**153**	151	148
fdmus_b21_481	**2 343**	224	152	175
fdmus_b21_96	**1 837**	715	710	738
fdmus_b22_102	4	**136**	135	133
fdmus_b22_113	1	45	47	**49**
fdmus_b22_133	**259**	226	223	229
fdmus_b22_172	1	88	**89**	86
fdmus_b22_241	**898**	145	136	142
fdmus_b22_356	1	**63**	58	62
fdmus_b22_488	**3 426**	428	390	476
fdmus_b22_600	1	62	62	**69**
fdmus_b22_662	1	**81**	76	75

注：黑体数值表示 MUS 枚举个数的最优结果.

率出现了显著的提升，在大部分实例上的结果远远优于对比算法. 这样的现象应用本文的理论分析也是可解释的：由于算法是基于极大化版本的 MARCO 算法，因此在搜索初期，算法优先从哈斯图的顶层选择节点来扩展，因此在算法初期找到的 MSS 往往偏大，这就使得 MSS 的交集可能偏大，MSS 的并集也偏大，MSS 并集的补集则偏小. 于是，所提算法的启发式即在原先选择的待探索节点基础上先与 MSS 交集做差集，再与 MSS 并集的补集做并集得到新的待扩展节点就会偏小，使得算法在初期更倾向于对 MSS 的枚举，当算法找到的 MSS 数量逐渐增加后，结合其交集和并集得到的启发式的信息才会更偏向于 MUS，此时，算法将大量搜索出 MUS，这也与本文的实验结果相吻合. 综上，本文所提算法的特点是在搜索初期会更偏向于搜索 MSS，在搜索到的 MSS 逐渐积累后，算法对 MUS 的枚举效率会呈现显著提升.

为更清晰地可视化实验结果，对表 1、表 2、表 3 的结果分别计算获胜个数并绘制成柱状图，如图 2 所示. 每个算法在每种时间限制下的获胜个数即在各个测试用例下，若该算法枚举 MUS 个数最多则记为获胜 1 次，统计所有测试用例下的总获胜次数. 从图 2 可以看出，所提算法 MARCO-MSS4MUS 的获胜个数仅在 0.5 h 的时间限制下没有超过对比算法 MARCO-ABC，但在 1 h 和 2 h 的时间限制下的实验结果中，MARCO-MSS4MUS 的获胜个数均优于其他对比算法. 这也印证了在前文中的分析，MARCO-MSS4MUS 算法的启发式引导它在搜索的前期更偏向于收集更多的 MSS，从而导致 MUS 的枚举个数极少，但在其收集到足够多的 MSS 后，充分利用已有 MSS 的信息将引导算法逐渐偏向于搜索 MUS，使得算法枚举 MUS 的效率呈现出显著的增长.

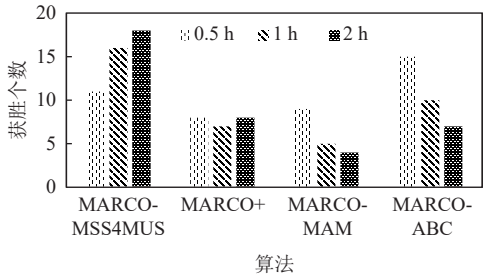


Fig. 2 Comparative results on win number of each algorithm under different time limits

图 2 各算法在不同时间限制下的获胜个数对比结果

6 结 论

本文提出了一种针对 MUS 枚举问题的新算法 MARCO-MSS4MUS. 基于经典 MARCO 框架及已有改进，在增量式求解 MUS 和 MSS 的过程中，进一步利用了搜索过程中枚举出的 MSS 的交集和并集作为启发式信息，指导算法对于待扩展节点的选择，从而

引导搜索逐渐朝着偏向 MUS 的方向进行，减少收缩和扩充操作的耗时以达到加速 MUS 枚举的目的. 提出 2 个新的定理及其推论为 MARCO-MSS4MUS 的有效性提供了理论依据，从理论角度说明了在 MSS 交集中的元素不一定出现在 MUS 中，在 MSS 并集的补集中的元素一定出现在部分 MUS 中，因此基于以上信息对待扩展节点进行相应的集合运算操作得到新的待扩展节点将更倾向于找到一个 MUS. 最后，标准测试用例的实验表明，MARCO-MSS4MUS 在枚举获胜个数和枚举效率 2 方面均较已有算法有显著的提高.

作者贡献声明：蒋璐宇提出了算法思路和实验方案，负责代码编写、撰写并修改论文；欧阳丹彤确定选题及指导论文写作；张奇负责文献资料的查阅和整理；太然负责整理实验数据；张立明负责对论文提出修改意见.

参 考 文 献

[1] Cai Shaowei, Zhang Xindi. Deep cooperation of CDCL and local search for SAT[C]//Proc of the 24th Int Conf on Theory and Applications of Satisfiability Testing. Berlin: Springer, 2021: 64−81

[2] Cai Shaowei, Zhang Xindi, Fleury M, et al. Better decision heuristics in CDCL through local search and target phases[J]. Journal of Artificial Intelligence Research, 2022, 74: 1515−1563

[3] Zheng Jiongzhi, He Kun, Zhou Jianrong, et al. BandMaxSAT: A local search MaxSAT solver with multi-armed bandit[C]//Proc of the 31st Int Joint Conf on Artificial Intelligence. Freiburg: IJCAI, 2022: 1901–1907

[4] Wang Rongquan, Ouyang Dantong, Wang Yiyuan, et al. Solving minimal hitting sets method with SAT based on DOEC minimization[J]. Journal of Computer Research and Development, 2018, 55(6): 1273–1281 (in Chinese)
(王荣全, 欧阳丹彤, 王艺源, 等. 结合 DOEC 极小化策略的 SAT 求解极小碰集方法[J], 计算机研究与发展, 2018, 55(6): 1273–1281)

[5] Metodi A, Stern R, Kalech M, et al. A novel SAT-based approach to model based diagnosis[J]. Journal of Artificial Intelligence Research, 2014, 51: 377−411

[6] Tian Naiyu, Ouyang Dantong, Liu Meng, et al. A method of minimality-checking of diagnosis based on subset consistency detection[J]. Journal of Computer Research and Development, 2019, 56(7): 1396–1407 (in Chinese)
(田乃予, 欧阳丹彤, 刘梦, 等. 基于子集一致性检测的诊断解极小性判定方法[J], 计算机研究与发展, 2019, 56(7): 1396–1407)

[7] Zhou Huisi, Ouyang Dantong, Zhao Xiangfu, et al. Two compacted models for efficient model-based diagnosis[C]//Proc of the 36th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2022: 3885–3893

[8] Ouyang Dantong, Liao Mengting, Ye Yuxin. Lightweight axiom pinpointing via replicated driver and customized SAT-solving[J]. Frontiers of Computer Science, 2023, 17(2): 172315

[9] Arif M F, Mencía C, Marques-Silva J. Efficient MUS enumeration of Horn formulae with applications to axiom pinpointing[C]//Proc of the 18th Int Conf on Theory and Applications of Satisfiability Testing. Berlin: Springer, 2015: 324–342

[10] Hunter A, Konieczny S. On the measure of conflicts: Shapley inconsistency values[J]. Artificial Intelligence, 2010, 174(14): 1007−1026

[11] Xiao Guohui, Ma Yue. Inconsistency measurement based on variables in minimal unsatisfiable subsets[C]//Proc of the 20th European Conf on Artificial Intelligence. Ohmsha: IOS, 2012: 864–869

[12] Shazli S Z, Tahoori M B. Using Boolean satisfiability for computing soft error rates in early design stages[J]. Microelectronics Reliability, 2010, 50(1): 149−159

[13] Andraus Z, Liffiton M, Sakallah K. Reveal: A formal verification tool for Verilog designs[C]//Proc of the 15th Int Conf on Logic for Programming Artificial Intelligence and Reasoning. Berlin: Springer, 2008: 343–352

[14] Liffiton M, Sakallah K. On finding all minimally unsatisfiable subformulas[G]//LNCS 3569: Proc of the 8th Int Conf on Theory & Applications of Satisfiability Testing. Berlin: Springer, 2005: 173–186

[15] Liffiton M, Sakallah K. Algorithms for computing minimal unsatisfiable subsets of constraints[J]. Journal of Automated Reasoning, 2008, 40(1): 1−33

[16] Bailey J, Stuckey P. Discovery of minimal unsatisfiable subsets of constraints using hitting set dualization[C]//Proc of the 7th Int Workshop on Practical Aspects of Declarative Languages. Berlin: Springer, 2005: 174–186

[17] Liffiton M, Malik A. Enumerating infeasibility: Finding multiple MUSes quickly[C]//Proc of the 10th Int Conf on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems. Berlin: Springer, 2013: 160–175

[18] Liffiton M, Previti A, Malik A, et al. Fast, flexible MUS enumeration[J]. Constraints, 2016, 21(2): 223−250

[19] Ouyang Dantong, Gao Han, Tian Naiyu, et al. MUS enumeration based on double-model[J]. Journal of Computer Research and Development, 2019, 56(12): 2623–2631 (in Chinese)
(欧阳丹彤, 高菡, 田乃予, 等. 基于双模型的 MUS 求解方法[J], 计算机研究与发展, 2019, 56(12): 2623−2631)

[20] Jiang Luyu, Ouyang Dantong, Dong Bowen, et al. Enhanced pruning scheme for enumerating MUS[J]. Journal of Software, 2024, 35(4): 1964−1979 (in Chinese)
(蒋璐宇, 欧阳丹彤, 董博文, 等. 针对 MUS 求解问题的加强剪枝策略[J], 软件学报, 2024, 35(4): 1964−1979)

[21] Bendík J, Benes N, Cerna I, et al. Tunable online MUS/MSS enumeration[G]//LIPIcs 65: Proc of the 36th IARCS Annual Conf on

Foundations of Software Technology and Theoretical Computer Science. Dagstuhl, Germany: Schloss Dagstuhl--Leibniz-Zentrum für Informatik, 2016: 50: 1–50: 13

[22] Bendík J, Cerna I, Benes N. Recursive online enumeration of all minimal unsatisfiable subsets[G]//LNCS 11138: Proc of the 16th Int Symp on Automated Technology for Verification and Analysis. Berlin: Springer, 2018: 143–159

[23] Bendík J, Cerna I. Replication-guided enumeration of minimal unsatisfiable subsets[C]//Proc of the 26th Int Conf on Principles and Practice of Constraint Programming. Berlin: Springer, 2020: 37–54

[24] Bendík J, Cerna I. Rotation based MSS/MCS enumeration[C]//Proc of the 23rd Int Conf on Logic for Programming, Artificial Intelligence and Reasoning. Stockport, UK: EasyChair, 2020: 120–137

[25] Bendík J, Meel K. Counting maximal satisfiable subsets[C]//Proc of the 35th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2021: 3651–3660

[26] Bendík J. On decomposition of maximal satisfiable subsets[C]//Proc of the 21st Conf on Formal Methods in Computer Aided Design. Piscataway, NJ: IEEE, 2021: 212−221

[27] Previti A, Marques-Silva J. Partial MUS enumeration[C]//Proc of the 27th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2013: 818–825

[28] Een N, Sorensson N. An extensible SAT-solver[G]//LNCS 2919: Proc of the 6th Int Conf on Theory and Applications of Satisfiability Testing. Berlin: Springer, 2004: 502–518



Jiang Luyu, born in 1998. PhD candidate. Student member of CCF. Her main research interests include SAT, combinatorial optimization, and model-based diagnosis problem.

蒋璐宇, 1998 年生. 博士研究生. CCF 学生会员. 主要研究方向为可满足性问题、组合优化问题、基于模型的诊断问题.



Ouyang Dantong, born in 1968. PhD, professor, PhD supervisor. Senior member of CCF. Her main research interests include automatic reasoning and model-based diagnosis problem. (ouyd@jlu.edu.cn)

欧阳丹彤, 1968 年生. 博士, 教授, 博士生导师. CCF 高级会员. 主要研究方向为自动推理、基于模型的诊断问题.



Zhang Qi, born in 1998. PhD candidate. Student member of CCF. His main research interests include SAT and model-based diagnosis problem.

张　奇, 1998 年生. 博士研究生. CCF 学生会员. 主要研究方向为可满足性问题和基于模型的诊断问题.



Tai Ran, born in 1997. PhD candidate. Student member of CCF. His main research interests include combinatorial optimization and model-based diagnosis problem.

太　然, 1997 年生. 博士研究生. CCF 学生会员. 主要研究方向为组合优化、基于模型的诊断问题.



Zhang Liming, born in 1980. PhD, senior engineer. Senior member of CCF. His main research interests include SAT and model-based diagnosis problem.

张立明, 1980 年生. 博士, 正高级工程师. CCF 高级会员. 主要研究方向为可满足性问题和基于模型的诊断问题.