软件学报 ISSN 1000-9825, CODEN RUXUEW
2025,36(12):5537−5553 [doi: 10.13328/j.cnki.jos.007397] [CSTR: 32375.14.jos.007397]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

HSDiag: 变种碰集算法求解诊断*

欧阳继红[1,2], 黄 森[1,2]

[1](吉林大学 计算机科学与技术学院, 吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室 (吉林大学), 吉林 长春 130012)

通信作者: 黄森, E-mail: hs_huangs@163.com

摘 要: 在基于模型诊断领域中, 首先对系统描述进行编码, 利用成熟的 SAT 求解器获得所有极小冲突集, 最后计算极小冲突集的极小碰集, 即待诊断设备的候选诊断. 然而这种策略花费大量的时间, 相当于计算两个 NP-hard 问题, 即计算极小冲突集和极小碰集. 对电路系统描述重新编码, 提出一种变种碰集算法 HSDiag, 该算法可以直接对编码进行计算来获得诊断. 在与目前最先进的求解冲突集再求解碰集的诊断算法相比, 效率最高提升 5–100 倍. 随着电路组件的增多, 编码子句呈线性增加, 诊断数量呈指数级增加. 因为求解大规模电路 (ISCAS-85) 的所有冲突集不切实际, 所以在设置相同的截止时间内, 提出的 HSDiag 算法与基于冲突集的诊断算法相比多求出 1 倍以上的解集. 除此之外, 提出一种专属求解诊断的等价类优化策略, 就算在初始冲突集不可分割的情况下, 利用新提出的集合分裂规则能够对冲突集进一步分解. 在标准的 Polybox 和 Fulladder 电路中, 使用等价类优化后的 HSDiag 算法, 效率进一步提升 2 倍以上.

关键词: 基于模型诊断; 系统描述; 极小冲突集; 极小碰集; 等价类

中图法分类号: TP181

中文引用格式: 欧阳继红, 黄森. HSDiag: 变种碰集算法求解诊断. 软件学报, 2025, 36(12): 5537–5553. http://www.jos.org.cn/1000-9825/7397.htm

英文引用格式: Ouyang JH, Huang S. HSDiag: Variant Hitting Set Algorithm for Solving Diagnosis. Ruan Jian Xue Bao/Journal of Software, 2025, 36(12): 5537–5553 (in Chinese). http://www.jos.org.cn/1000-9825/7397.htm

HSDiag: Variant Hitting Set Algorithm for Solving Diagnosis

OUYANG Ji-Hong[1,2], HUANG Sen[1,2]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)

Abstract: In the field of model-based diagnosis, the system description is first encoded, and all minimal conflict sets are obtained using a mature SAT solver. Finally, the minimal hitting set of the minimal conflict sets is computed as the candidate diagnosis for the equipment to be diagnosed. However, this strategy consumes a significant amount of time, as it is equivalent to solving two NP-hard problems: computing the minimal conflict set and the minimal hitting set. This study re-encodes the description of the circuit system and proposes a novel variant hitting set algorithm, HSDiag, which can directly compute the diagnosis from the encoding. Compared to state-of-the-art diagnosis algorithms that first solve conflict sets and then hitting sets, the efficiency improves by a factor of 5 to 100. As the number of circuit components increases, the encoding clauses increase linearly, while the number of diagnoses increases exponentially. Since solving all conflict sets of large-scale circuits (ISCAS-85) is impractical, the proposed HSDiag algorithm, within the same cutoff time, yields more than twice the number of solutions compared to conflict-set-based diagnosis algorithms. In addition, this study proposes an equivalence class optimization strategy, which further decomposes the conflict set by using the newly proposed set splitting rule, even if the initial

* 基金项目: 吉林省煤炭洁净燃烧技术的环境保护装备项目 (3D516L921421)
收稿时间: 2023-11-06; 修改时间: 2024-05-15; 采用时间: 2025-01-13; jos 在线出版时间: 2025-06-11
CNKI 网络首发时间: 2025-06-11

conflict set is inseparable. The efficiency of the HSDiag algorithm optimized by equivalence class is improved by more than 2 times in standard Polybox and Fulladder circuits.

Key words: model-based diagnosis; system description; minimal conflict set; minimal hitting set; equivalence class

基于模型诊断 (model-based diagnosis, MBD) 是人工智能的一个重要领域, MBD 专家 Reiter 提出的开创性工作是根据系统描述利用基于推理的方法来解释系统观测产生不一致的原因[1]. MBD 被广泛应用于芯片检测[2]、国家配电网[3,4]、软件故障定位[5]等众多域[6,7]. 图 1 是模型诊断框架, 首先对真实的环境系统进行建模编码, 然后通过系统观测对实际值和理论值进行一致性检测, 若不一致则产生冲突, 利用 SAT/ATMS 求解器找到所有的极小冲突集[8,9]. 最后计算极小冲突集的极小碰集就是待诊断系统的故障元件. 这里需要注意的是, 因为极小冲突集和极小不可满足子集在模型诊断中是相同的概念, 所以本文统一称为极小冲突集[10,11].

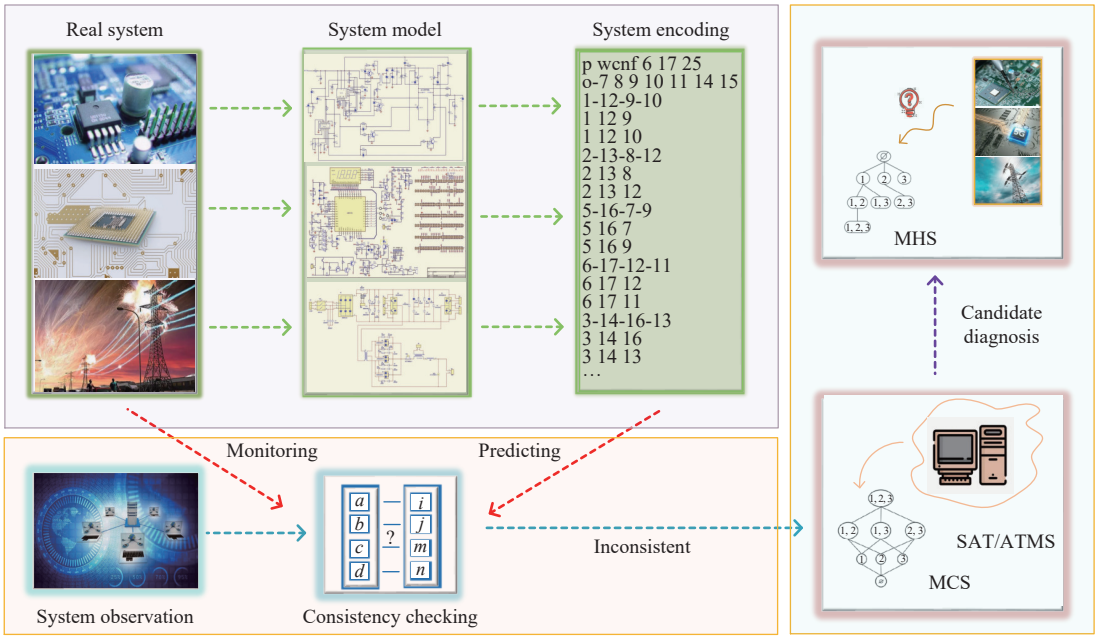


图 1 MBD 框架

计算极小冲突集主要基于枚举树和哈斯图[12]. 2009 年赵相福等人[13]提出了 CSISE 算法, 该算法首先将待诊断系统描述及观测用合取范式编码给出, 最后调用成熟的 SAT 求解器判断组件子集是否是冲突集[14], 然而该算法需要遍历枚举树中大部分节点, 求解花费往往较高. 2013 年 Liffiton 等人[15]结合哈斯图提出了 MARCO 算法, 该算法是求解极大化模型中枚举极小冲突集效率较高的算法, 但是它没有进一步对有解空间进行剪枝. 2016 年 Liffiton 等人[16]结合 eMUS 对该算法进行了优化, 在哈斯图未求解的空间中利用启发式策略优先选取极大节点, 进而快速得到一个极小冲突集. 2019 年欧阳丹彤等人[12]在极大化模型 MARCO 算法基础上增加了中间模型, 利用中间节点对冗余解进行剪枝, 缩小了求解范围. 2024 年蒋璐宇等人[11]提出了 MARCO-ABC 算法, 有效的利用极小冲突集和极小碰集二元性关系, 大幅度减少了求解时间, 是目前求解极小冲突集最快的算法.

计算碰集方法主要是基于树, 2003 年姜云飞等人[17]提出 Boolean 算法, 通过深度优先规则, 把冲突集递归的分成左右分支, 很多情况下都优于其他集中式算法, 后续有很多基于 Boolean 算法的优化策略被提出[18]. 2015 年 Zhao 等人[19]提出了等价类概念, 通过连接关系把冲突集分成若干等价类, 缩减了问题规模的同时也省去了解集合并时间. 同年 Jannach 等人[20]利用分布式思想把碰集算法并行化, 提高了求解效率. 2018 年刘思光等人[21]提出了布尔结合增量独立覆盖策略 (boolean with increment independent coverage checking, BWIICC) 算法, 利用独立覆盖策略保证生成解的都是极小的, 大幅度减少解集极小化的时间. 2022 年赵相福等人[22]结合独立覆盖及增量策略提

出了增量 BWIICC 算法, 利用左分支是右分支的真子集关系, 在计算左分支解集后, 通过增量的形式生成右分支, 是目前求解极小碰集效率较高的算法. 除此之外, 还有一些进化算法被提出[23,24], 但是这些算法大都是不完备的, 在获得部分解时花费较少, 当获得全部解时, 耗时也不容乐观[25].

随着系统软件的复杂性不断增加, 对诊断算法的要求也越来越高. 当系统是单观测时[26], 在某些情况下离线求解冲突集是可以接受的, 但是只通过一次观测, 很难获得具体的故障位置. 获得多次观测数据是容易且廉价的, 许多专家提出了多观测算法, 即对故障组件进行多次观测, 获得多对输入输出数据, 通过计算聚合诊断来缩小待诊断组件的范围[27]. 在多观测系统及实时观测系统中, 求解极小诊断的时间是求解极小冲突集和极小碰集之和, 因此求解极小冲突集在这种情况下已不适合离线进行[28].

近年来, 把系统描述编码成合取范式, 利用 SAT/MaxSAT 求解器获得极小冲突集或极小诊断. 不幸的是, 随着电路规模的增大, 编码子句呈线性增长, 在求解所有冲突集或极小诊断时大部分情况是耗时严重或者是不现实的[29−31]. 为了能够在有限时间返回有用的诊断, 不完备的算法被提出. 2007 年 Siddiqi 等人[32]提出了一种顶层诊断概念, 把电路中的组件分为统治节点和被统治节点, 并且默认被统治节点是健康的, 在获得所有极小顶层诊断后, 再通过统治节点替换被统治节点策略求出所有极小诊断. 2015 年 Mencia 等人[33]提出过滤边和过滤节点概念, 默认过滤节点中的组件健康, 进一步缩小故障组件的范围, 能够快速获得一个诊断. 2019 年 Ignatiev 等人[28]计算多观测诊断时, 利用 MARCO 算法收集极小冲突集, 利用极小碰集算法得到所有的诊断, 但是由于求解极小冲突集耗时严重, 在处理多故障系统时, 效率有待进一步提高. 2022 年 Zhou 等人[34]提出了压缩策略, 利用第 1 次观测得到的变量尽可能地共享其余观测的变量, 但是当其他观测与第 1 次观测共享的变量较少时优化效率不明显, 并且压缩模型在单观测诊断上不起作用.

计算所有极小冲突集和极小碰集都是 NP-hard 问题, 为了缩短过程提高求解速度, 本文基于碰集思想, 定义组件变量和传播变量, 修改系统编码, 提出一种集合分裂规则及变种碰集算法直接计算诊断. 换句话说, 把系统编码当成冲突集, 在计算诊断的过程中, 因碰集中只需要保留组件变量, 故其余变量都相当于传播约束. 为了进一步加快求解速度, 本文在基于等价类的基础上分解冲突集. 当冲突集合簇不可分时, 通过调用集合分裂规则, 对传播变量进行删除, 减少集合簇之间的连接关系, 当对传播变量进行赋值时, 因求得的诊断解只含有组件变量, 故碰集中不需要扩展传播变量.

本文第 1 节介绍基于模型诊断的相关概念以及基于冲突集计算诊断. 第 2 节首先介绍集合分裂规则, 然后结合碰集算法提出了 HSDiag 算法, 最后为了缩减编码规模, 提出了一个专属计算诊断的等价类优化策略. 第 3 节将所提出的算法与目前最快的基于冲突集计算诊断的算法进行对比, 实验数据采用 ISCAS-85, Fulladder, Polybox 这 3 种标准的数据集, 验证了所提算法的有效性. 最后总结全文.

1 基础知识

1.1 基于模型诊断的相关概念

待诊断系统可定义为一个三元组<*SD*, *Comps*, *Obs*>. 其中, *SD* 为系统描述, 为一阶谓词公式的集合; *Comps* 为系统组成部件的集合, 是一个有限常量集; *Obs* 为观测集, 是一阶谓词公式的有限集. 假定所有组件状态是正常的情况下, 当系统模型描述和观测出现不一致时, 我们称存在一个诊断问题, 即 $SD \wedge Obs \wedge \{\neg A(c)|c \in Comps\}$是不一致的. 其中, $A(c)$ 为真代表组件 c 故障, 值为假代表组件 c 正常.

如果系统输出的真实值和理论值不同, 我们称为是不一致的, 反之是一致的.

定义 1. 诊断 (diagnosis)[28]. 给定一个诊断问题 *D*=<*SD*, *Comps*, *Obs*>.一个诊断被定义为一组组件Δ 的集合, 其中$\Delta \subseteq Comps$, 当 $D \wedge Obs \wedge \{A(c)|c \in \Delta\} \wedge \{\neg A(c)|c \in Comps-\Delta\}$是一致的, 其中Δ 是一个极小子集诊断, 当且仅当它的任意真子集都不是诊断.

为了判定一个部件集是否为冲突集即判定该组件集中所有组件的正常行为描述与相关的系统描述及观测描述是否逻辑一致.

定义 2. 极小冲突集 (minimal conflict set, MCS)[1]. 组件集 f 是系统组件集 $Comps$ 的冲突集, 当且仅当 $f\subseteq Comps$, $\{SD\wedge Obs\wedge\{\neg A(c)|c\in f\}$是不一致. 若 f 的任意真子集都不是冲突集, 则称 f 为极小冲突集.

定义 3. 极小碰集 (minimal hitting set, MHS)[1]. 冲突集合簇 $F=\{f_1,f_2,\ldots,f_n\}$, 集合 S 是 F 的一个碰集, 当且仅当 $S\cap f_i\neq\varnothing$, $\forall f_i\in F$. 如果 S 的任意真子集都不是 F 的碰集, 那么 S 是 F 的一个极小碰集.

下面通过例 1 对以上 3 个定义进一步说明.

例 1: 图 2 描述了一个乘法加法器实例, M_1–M_3 和 A_1–A_2 分别代表乘法器和加法器. i_1–i_6、o_1 和 o_2 分别代表系统的输入和输出.

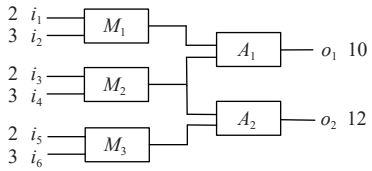


图 2　乘法加法器电路实例

假设通过观测, 输入值 i_1–i_6 是{2, 3, 2, 3, 2, 3}, 输出值 o_1、o_2 是{10, 12}. 通过加法和乘法的原则, 若 M_1–M_3、A_1 和 A_2 都正常工作, 则 o_1 和 o_2 的输出值应都为 12, 然而 o_1 的输出值却为 10, 通过一致性原理[1], 显然至少有一个组件出现故障. 基于当前的观测和模型理论, 调用 MCS 求解算法[11], 我们能够得到{M_1, M_2, A_1}和{M_1, M_3, A_1, A_2}这两个极小冲突集, 即每个冲突集合中的所有组件都正常工作是不可能的. 对于每一个 MCS 里面的组件, 至少有一个出现故障, 计算所有极小冲突集的所有极小碰集, 就能够解释观测情况. 根据定义 3, 我们能够找到所有的极小碰集是{M_1}, {A_1}, {M_2, M_3}, {M_2, A_2}. 例如: {M_1}是一个候选诊断, 假设 M_1 出现故障 (输出值是 4), 其余组件都正常, 则 o_1 的输出值可以是 10. 可见计算所有冲突集的极小碰集是获得故障诊断的重要步骤, 故提高极小冲突集和极小碰集的求解效率能够快速定位诊断.

1.2　基于冲突集计算诊断

定义 4. 文字 (literal)[11]. 给定布尔变量集合 $X=\{x_1, x_2,\ldots, x_n\}$, 文字 l_i 是变量 x_i 或变量 x_i 的否定–x_i, 二者互为补集.

定义 5. 子句 (clause)[11]. 子句 C 是若干个文字的析取组成, $C=l_1\vee l_2\vee\ldots\vee l_m$.

定义 6. 合取范式 (conjunctive normal form, CNF)[11]. 由多个不同子句合取所组成的公式称为合取范式, $F=C_1\wedge C_2\wedge\ldots\wedge C_m$.

定义 7. 命题可满足性问题 SAT[11]. 给定一组布尔变量, 若能找到一组对变量的真值指派, 使得给定命题公式中的每一个子句都为真, 则称该问题是可满足的; 反之, 若不存在这样的真值指派, 则称该问题是不可满足的.

例 2: 给定一个与非门, 假设有两个输入 x_1, x_2, 一个输出 o_1, 一组输入输出观测 $Obs_1=\{x_1, -x_2, o_1\}$, 用 CNF 公式进行描述该组件的逻辑关系, 则公式 $F=\{\{-x_1\vee -x_2\vee -o_1\}\wedge\{x_1\vee o_1\}\wedge\{x_2\vee o_1\}\wedge\{x_1\}\wedge\{-x_2\}\wedge\{o_1\}\}$, 因可以找到一组赋值{$x_1$, $-x_2$, o_1}使得 CNF 中每一个子句都为真, 故该 SAT 问题是可满足的. 若再给定一组观测 $Obs_2=\{-x_1, -x_2, -o_1\}$, 则此时的 $F=\{\{-x_1\vee -x_2\vee -o_1\}\wedge\{x_1\vee o_1\}\wedge\{x_2\vee o_1\}\wedge\{-x_1\}\wedge\{-x_2\}\wedge\{-o_1\}\}$, 因无法找到一组赋值使得所有子句都为真, 故该问题是不可满足的.

已知与非门存在一个低电平的输入, 输出为高电平. $Obs_1=\{x_1, -x_2, o_1\}$, 输入分别是高、低电平, 输出是高电平, 符合逻辑值, 产生的 CNF 编码是可满足的. 而对于 $Obs_2=\{-x_1, -x_2, -o_1\}$, 输入是两个低电平, 输出却是低电平, 显然逻辑值错误, 对应的 CNF 编码是不可满足的. 也就是说, 若 CNF 编码不可满足, 则对应不满足子句的组件就是待诊断故障组件. SAT 子句是由布尔变量给出, 布尔变量中含有的是文字, 相当于本文所说的变量.

SAT 求解器的输入是以 CNF 公式形式, 因此组合电路的相应描述需要以子句集形式表示. 根据图 3 中 c17 电路进行进一步说明基于冲突集求解诊断. 图 3(a) 和图 3(b) 描述了 c17 电路中组件门的性质及连接关系, 图 3(c) 是 c17 电路的 CNF 编码及求诊断算法[26]. 在国际基准电路 ISCAS-85 中[24,28,29], c17 电路的所有组件都是与非门

(N_1–N_6), 系统输入 (i_7–i_{11}) 和输出 (o_{14}、o_{15}) 是可观测的, 其余都是不可观测的. 在编码时用正文字表示高电平, 负文字表示低电平. 对于组件, 用正文字表示组件故障, 负文字表示组件正常 (使用 1–17 分别表示电路中对应的下标变量). 如: 变量 9 表示 i_9 为高电平, −9 表示 i_9 为低电平, 变量 1 表示组件 N_1 故障. 假设观测到系统的输入 i_7–i_{11} 的值分别是{−7, 8, 9, 10, 11}, 系统的输出 o_{14} 和 o_{15} 值是{14, 15}. 显然, 如果组件都正常工作, 得到的理论值应为{−14, −15}, 故该电路必然存在故障组件. 基于冲突集求诊断主要分为两个阶段, 计算冲突集和计算极小碰集.

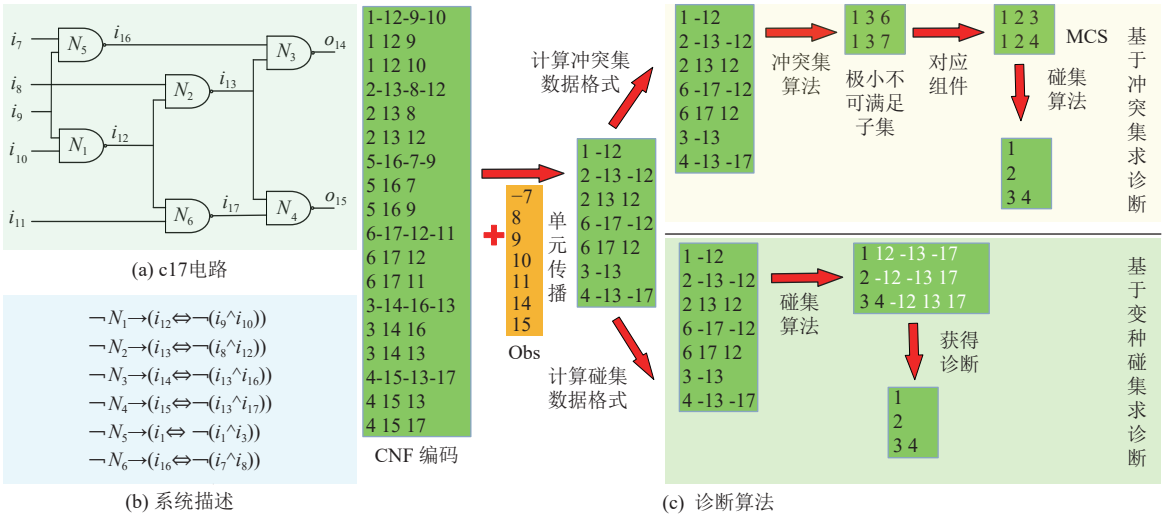


图 3 两种算法计算 c17 电路的候选诊断

第 1 阶段: 计算冲突集. 该阶段一般是基于哈斯图, 首先对编码中的子句生成一个序列, 根据哈斯图中的节点顺序调用 SAT 求解器判断组成的编码是否可满足, 若不满足则生成一个冲突集合, 通过修剪规则遍历所有的节点后, 就能够得到所有极小冲突集.

对于图 3(c), 首先对观测值 *Obs* 进行单元传播, 转化成计算冲突集数据格式的合取范式编码, 白色字体变量在编码中不会出现, 它对应不可满足子句的组件. 假设调用 SAT 求解器, 获得两个极小冲突集{1, 3, 6}, {1, 3, 7}, 它们对应的组件分别是{1, 2, 3}, {1, 2, 4}.

第 2 阶段: 计算诊断. 调用碰集算法对第 1 阶段获得的极小冲突集进行求解. 一共获得 3 个极小碰集{1}, {2}, {3, 4}.

可见, 最终得到的极小碰集就是待诊断的故障组件. 在模型诊断中, 部分解的丢失很有可能由于最后一步的故障检测, 而错过最有可能的故障位置[25], 然而要得到所有的极小碰集需要先计算所有的极小冲突集, 否则求出的解集可能不完备或者不是诊断解. 如图 3, 若调用 SAT 求解器只得到一个极小冲突集{1, 2, 3}, 则极小碰集{3}就不是一个诊断解, 因为就算组件 N_3 故障也不能解释系统观测.

2 基于变种碰集直接求诊断

基于冲突集计算诊断, 在检查子句可满足时, 实际上就是在对子句集进行碰集求解. 基于冲突集计算诊断在编码生成方面时, 组件变量不出现在编码中. 在图 3(c) 中, 首先对所有子句进行一个排序, 计算出不可满足子句{1, 3, 6}, {1, 3, 7}后, 需要对组件进行一个映射, 对应的组件变量才是模型诊断中需要的极小冲突集{1, 2, 3}, {1, 2, 4}. 若组件变量出现在编码中, 则所有的子句都是可满足的, 如一个可满足解{1, 2, 3, 4, 6}.

2.1 碰集算法求解诊断

为了保证所提出策略的正确性, 我们对编码中的变量进行定义.

定义 8. 组件变量和 (单) 传播变量 (component variable and (single) propagation variable). 在编码中, 诊断解中

的变量为组件变量, 其余变量为传播变量. 若编码中传播变量不存在补集, 则为单传播变量.

在图 3(c) 的 CNF 编码中, 变量 7–17 是传播变量, 1–6 是组件变量. 如编码中仅存在 9 而不存在−9, 则 9 为单传播变量. 为了方便描述, 用 e_c、e_p、e_s 分别代表组件变量、传播变量及单传播变量. 利用碰集算法直接计算诊断时需要返回待诊断的组件, 所以把组件变量放入编码中.

定义 9. 极大可满足子集 (maximal satisfiable subset, MSS)[11]. 给定子句集 U, 若 $F \subseteq U$, F 称作 U 的极大可满足子集, 当且仅当 F 是可满足的, F 再添加一个子句是不可满足的.

定义 10. 极小修正集 (minimal correction set, MCS′)[11]. 给定子句集 U, 若 $F \subseteq U$, F 称作 U 的极小修正集, 当且仅当 $U\backslash F$ 是可满足的, $U\backslash F$ 再添加一个子句是不可满足的.

由图 4 可知, 在基于模型诊断中, 极小不可满足子集和极小冲突集是等价的, 而计算他们的极小碰集, 即极小修正集和极小碰集也是等价的[11]. MSS 和极小诊断互为补集, 利用碰集算法, 在不改变碰集算法的规则前, 优先选取传播变量, 选取所有可以满足的子句 (相当于求解极大可满足子集), 对于不可满足的子句选取组件变量进行碰集求解 (相当于求解补集), 最终的碰集解只需要保留组件变量.

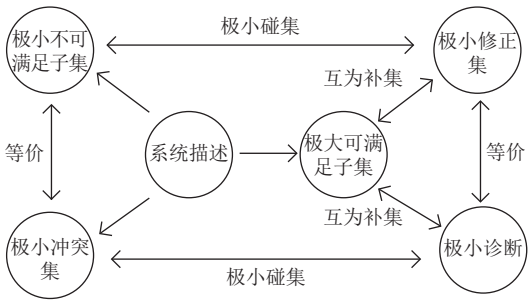


图 4 系统描述到候选诊断关系图

命题 1. 碰集算法计算带有组件变量的 CNF 编码可以获得所有极小诊断.

下面分别从正确性和完备性来对命题 1 进行证明.

正确性: 由诊断定义 1 可知, 需要找到一个极小诊断Δ 来满足系统所产生的所有子句. 利用碰集算法首先找到不含组件的最大可满足的子集 MSS, 剩余不可满足的子句 (MSS 的补集) 用组件变量代替, 即极小诊断.

完备性: MCS′的补集是 MSS (具有一一对应关系), 根据已有算法可知 MCS′是完备性的, 那么可以推出 MSS 的完备性, 故 MSS 的补集也是完备的.

证毕.

例 3: 利用碰集计算图 3(c) 中 F={{1, −12}, {2, −13, −12}, {2, 13, 12}, {6, −17, −12}, {6, 17, 12}, {3, −13}, {4, −13, −17}}的极小诊断.

假设利用碰集算法, 在不选取组件变量的前提下, 碰集 HS={12, −13, −17}对应的一个 $MSS(F)$={{2, −13, −12}, {2, 13, 12}, {6, −17, −12}, {6, 17, 12}, {3, −13}, {4, −13, −17}}, 则剩下一个集合{1, −12}, 需要选取组件变量 1 (选取变量−12 会形成不可行解), 因碰集中只需要保留组件变, 故碰集 HS 只需要返回诊断解{1}.

直接调用碰集算法得到的解中含有大量的非组件变量. 为了避免产生过多的冗余解, 在变量选取的时候, 优先选取传播变量, 并删除含有单传播变量的子句. 我们提出一种集合分裂策略, 该策略结合碰集算法, 大幅度减少左右分支的冗余子句.

2.2 集合分裂

在求解诊断时, 因获得的解是极小的, 故可以删除含有单传播变量 e_s 的子句. 我们用引理 1 给出, 并用碰集知识进行证明.

引理 1. F 和 F'有相同的极小诊断, 其中 $F'=\{F-\{S\}|S\in F\wedge e_s\in S\}$.

证明: 首先证明Δ是 F'的极小诊断也是 F 的极小诊断. 因 e_s 可以满足子句集 S, 当Δ对 e_s 进行传播时, e_s 已经

与 S 有交集, 不会改变Δ的值, 故Δ也是 F 的极小诊断. 下面证明Δ是 F 的极小诊断解也是 F'的极小组件诊断解.

反证法: 若Δ是 F 的极小诊断, 因 F'是 F 的真子集, 故Δ也是 F'的诊断. 接着证明极小性, 根据极小性原理, 若Δ不是 F'的极小诊断, 则必存在Δ的真子集Δ′是 F'的极小诊断, 由上面证明可得Δ′也是 F 的极小诊断, 这与假设Δ是 F 的极小诊断矛盾, 故Δ是 F 的极小诊断, 也是 F'的极小诊断.

证毕.

综上, 含有 e_s 的子句为冗余子句可以直接删除.

布尔算法是一种效率比较快的碰集算法, 它在对子句集 F 选取变量 (e) 时, 会按照递归规则分裂成两个分支 F_l 和 F_r, 其中, $MHS(F)=e\wedge MHS(F_l)\vee MHS(F_r)$, $F_l=\{S_i|S_i\in F\wedge e\notin S_i\}$, $F_r=\{S_i-\{e\}|S_i\in F\}$[18]. 我们把该规则应用于求解候选诊断, 并基于引理 1 进一步优化.

命题 2. 集合分裂. 利用碰集算法计算候选诊断当选取 e_p 时, 那么子句集 F 分裂成两个分支 F_l 和 F_r, 其中, F_l 是删除含有 e_p 的子句及$-e_p$, $F_l=\{S_i-\{\vee e_p\}|S_i\in F\wedge e_p\notin S_i\}$, 右分支是删除含有$-e_p$ 的子句及 e_p, $F_r=\{S_i-\{e_p\}|S_i\in F\wedge -e_p\notin S_i\}$.

证明: 根据极小碰集算法选取变量的规则, 当选取 e_p 时, 能获得 $F_l=\{S_i|S_i\in F\wedge e_p\notin S_i\}$, $F_r=\{S_i-\{e_p\}|S_i\in F\}$. 对于 $e_p\wedge F_l$, 由于引理 1, 含有$-e_p$ 的候选解都是不可满足的, 所以需要删除左分支中$-e_p$, $F_l=\{S_i-\{-e_p\}|S_i\in F\wedge e_p\notin S_i\}$.

而对于右分支, 此时$-e_p$ 相当于单传播变量, 根据引理 1, 可以删除含有$-e_p$ 的集合, $F_r=\{S_i-\{e_p\}|S_i\in F\wedge -e_p\notin S_i\}$.

证毕.

下面用例 4 对命题 2 进一步说明.

例 4: 继续计算图 3(c) 生成的碰集格式的编码 F={{1, −12}, {2, −13, −12}, {2, 13, 12}, {−17, −12}, {17, 12}, {3, −13}, {4, −13, −17}}.

图 5 中是采用目前求解效率较高的碰集算法生成树[17,18,22], 其中右侧树比左侧树多添加了集合分裂策略. 在图 5(a) 中, 每次选取一个变量时, 左分支删除含有选取变量的集合, 右分支删除选取的变量, 不断地递归, 直到节点为空或者只剩一个集合. 而对于图 5(b), 虽然也生成两个分支 (存在单集合变量时生成一个分支), 不同的是, 每次选取变量后, 左分支进行一个判断, 若选取的变量是传播变量, 则需要额外的删除其补集 (单元传播), 这是因为一个解中含有一对互补变量是不可行的; 而对于右分支, 删除左分支选取的变量后, 需要额外删除含有单传播变量的集合. 可见右侧树生成的节点明显少于左侧树. 算法 1 的伪代码是添加分裂策略后的变种碰集算法 HSDiag.

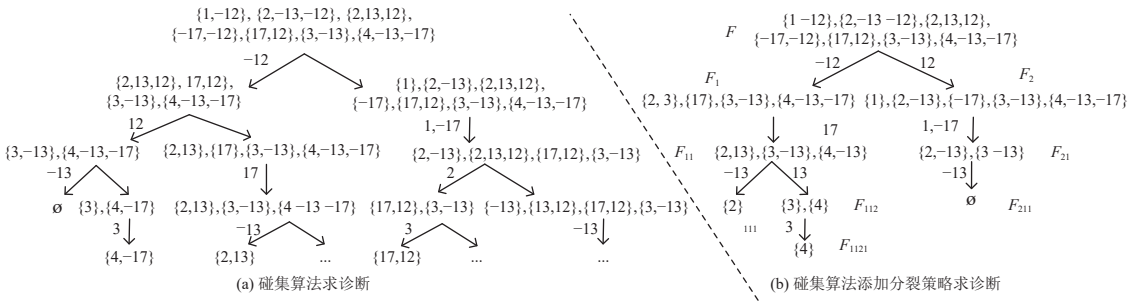


图 5　碰集算法及添加分裂策略求诊断

算法 1. *HSDiag*(F, D).

输出: CNF 编码 $F=\{S_1, S_2,\ldots, S_n\}$, 诊断 D;

输出: 诊断 D.

1. **IF** ($F==\varnothing$) **return**;
2. **IF** ($F==\{S\}$){ //只有一个集合
3. **IF** ($\exists e_p$ ($e_p\in S$)) **return**; //存在传播变量

4. **return** $\{\{e_i\}|e_i \in S\}$; //返回每一个组件变量
5. }
6. **ELSE IF** $(\exists e\,(\{e\} \in F))$ { //存在单变量集合
7. **IF** (e 是 e_c) { //是组件变量
8. $HSDiag(\{S_i - \{e_c\}|S_i \in F \wedge e_c \notin S_i\}, D)$;
9. $D \leftarrow \{d \cup \{e_c\}|d \in D\}$;
10. }
11. **ELSE** $HSDiag(\{S_i - \{-e\}|S_i \in F \wedge e \notin S_i\}, D)$; //删除含有变量 e 的集合并删除变量–e
12. }
13. **ELSE** {
14. $e \leftarrow Select\_element\left(\bigcup_{S_i \in F} S_i\right)$; //从子句集中选取变量
15. **IF** (e 是 e_p) { //是传播变量
16. **IF** $(\exists e \forall S_i\,(S_i \in F \rightarrow e \in S_i))$ **return;** //所有集合都含有传播变量 e
17. **IF** (e 不是 e_c) //不是单传播变量
18. $HSDiag(\{S_i - \{e\}|S_i \in F \wedge -e \notin S_i\}, RightD)$; //删除含有变量-$e$ 的集合并删除变量 e
19. $HSDiag(\{S_i - \{-e\}|S_i \in F \wedge e \notin S_i\}, LeftD)$; //删除含有变量 e 的集合并删除变量-e
20. }
21. **ELSE** { //是组件变量
22. $HSDiag(\{S_i|S_i \in F \wedge e \notin S_i\}, LeftD)$; //把含有变量 e 的集合删除
23. $HSDiag(\{S_i - \{e\}|S_i \in F\}, RightD)$; //把变量 e 删除
24. $D \leftarrow \{ld \cup \{e\}|ld \in LeftD\} \cup RightD$;
25. }
26. }

LeftD、*RightD* 分别存储左分支、右分支中间过程生成的解集, 初始化为空, *ld* 是 *LeftD* 的子集.

在求解诊断时, 若编码子句为空, 则直接返回 (第 1 行); 若集合个数为 1 (第 2 行), 判断子句中是否存在传播变量 e_p, 若存在 e_p, 说明该子句是可满的则直接返回, 否则返回子句中每个组件变量 e_c (第 3, 4 行); 若子句个数大于 1 且存在单变量集合 (第 6 行), 若该单字是单传播变量 e_c, 删除含有该单字的集合并继续递归, 候选诊断中的每个解扩展变量 e (第 7–9 行), 否则删除含有 e_p 的集合并且删除该变量的补集 (第 11 行); 若子句个数大于 1 且没有单变量集合, 从子句集中选择一个变量用来递归 (第 14 行), 若该变量是 e_p (第 15 行), 所有子句都含有该 e_p, 说明所有的子句是可满足的, 则直接返回 (第 16 行), 若该变量不是单传播变量, 根据引理 1, 则把子句中所有含有变量–e 的子句删除并删除变量 e, 继续递归 (第 17, 18 行), 之后生成左分支子句集接着递归 (第 19 行). 若选取的变量 e 是 e_c, 则生成左右两个分支进行递归, 其中左分支是删除含有变量 e 的集合, 右分支只删除变量 e, 最后左分支解集扩展变量 e 后合并右分支解集 (第 22–24 行).

算法 1 每次选取变量的时候调用集合分裂规则, 即删除选取的含有变量的集合并删除该变量的补集. 对于图 5(b) 右侧树, 根节点用 F 表示, 左右分支分别用下标 1、2 表示, 只有一个分支用 1 表示, 如 F 的左右分支为 F_1、F_2, F_1 的左右分支为 F_{11}, F_{12}, 依次递推. 这里需要注意的是, 为了使算法与枚举树对应, 我们把选取的变量放在路径上 (箭头), 生成的中间集合簇放在节点上.

F 不是一个集合且不存在单变量集合 (第 1–12 行), 从 F 中选取变量–12 进行递归 (第 14 行), 因为是传播变量且不是单传播变量 (第 17 行), 生成右分支 F_2 和左分支 F_1 (第 18, 19 行), 左分支 F_1 继续算法递归; F_1 存在单变量集合且不是组件变量 (第 11 行), 生成 F_{11} 继续算法递归; F_{11} 选取变量−13 生成左右分支 F_{111}, F_{112} (第 17–19 行),

此时 F_{111} 只有一个集合且是组件变量, 返回组件变量并扩展路径上的组件变量. 其余节点依次递归直到算法截止.

2.3 等价类优化

把电路编码成 CNF 公式, 当组件不断增多时, 编码子句成线性增长, 求解诊断难度成指数增长. 若能够缩减子句规模, 求解时间将大幅度缩减. 原始冲突集要是能够分成若干个互不相交的子句集, 利用等价类策略只需要求解每个子集的诊断, 最好的情况下能使求解时间成指数级下降. 不幸的是, 因电路的每一个组件都有输入和输出, 导致整个电路的组件通过传播变量连接在一起, 故无法进行分割. 现有的等价类策略无法对电路的 CNF 编码进行分类, 但是考虑到诊断中只存在组件变量, 则可以进一步地分解. 接下来介绍等价类策略, 再提出一种专属求解诊断的优化等价类策略.

定义 11. 冲突集等价类 (equivalence class, EC)[19]. 如果冲突集 F 能够分成 k 个没有交集的子句集, 则称每个子句集为冲突集等价类, 其中满足 $F=\bigcup\limits_{1\leqslant i\leqslant k} EC_i$, $MHS(F)=\mathop{\&}\limits_{1\leqslant i\leqslant k} MHS(EC_i)$.

例 5: 利用等价类求解冲突集 F={{1, 2}, {2, 3}, {4, 5, 6}, {6, 7}, {8, 9}}.

等价类策略可以把 F 分成不相交的 3 个等价类, EC_1={{1, 2}, {2, 3}}, EC_2={{4, 5, 6}, {6, 7}}, EC_3={{8, 9}}. 分别计算每个等价类的极小碰集 $MHS(EC_1)$={{2}, {1, 3}}, $MHS(EC_2)$={{6}, {4, 7}, {5, 7}}, $MHS(EC_3)$={{8}, {9}}. $MHS(F)$={{2, 6, 8}, {2, 6, 9}, {2, 4, 7, 8}, {2, 4, 7, 9}, {2, 5, 7, 8}, {2, 5, 7, 9}, {1, 3, 6, 8}, {1, 3, 6, 9}, {1, 3, 4, 7, 8}, {1, 3, 4, 7, 9}, {1, 3, 5, 7, 8}, {1, 3, 5, 7, 9}}. 从而能够得到这样一个关系式 $MHS(F)=MHS(EC_1)$ & $MHS(EC_2)$ & $MHS(EC_3)$.

其中符号& 是笛卡尔乘积. 把问题集合簇分成等价类可以减少解集的生成, 最好情况下可以使问题的复杂度从 2^k 降低到 $2^{k/n}$ 次方, n 是等价类的个数.

在计算一个冲突集合簇的所有极小碰集时, 若一个冲突集合簇可以分成若干个等价类, 则只需计算每个等价类的所有极小碰集. 当子句集不可分成等价类时, 对于计算碰集而言是无法继续使用等价类策略的, 而对于计算诊断却是可以的. 这是由于编码子句集是通过传播变量 e_p 进行连接 (交集), 若我们对 e_p 调用分裂规则后, 剩余的编码子句则没有任何交集, 生成的诊断解又无需扩展 e_p, 可以对初始不能分等价类的编码子句间接调用等价类策略.

推论 1. 诊断等价类. 若诊断编码 F 通过选取 e_p 后生成 m 个节点, 每个节点有 k_j ($1\leqslant j\leqslant m$) 个互相没有交集的子句集 EC_i ($1\leqslant i\leqslant k_j$), 则称每个子句集为诊断等价类, 其中满足 $HSDiag(F)=\bigcup\limits_{1\leqslant j\leqslant m}\mathop{\&}\limits_{1\leqslant i\leqslant k_i} HSDiag(EC_i)$.

下面用例 6 进一步解释在诊断编码初始不可分成等价类时, 通过选取传播变量间接调用等价类策略.

例 6: 基本电路 Polybox_5 诊断编码间接调用等价类策略.

图 6 是 Polybox_5 电路示例图, 该电路转化成 CNF 编码子句时, 子句 F 是无法分成若干等价类, 但是可以通过选取变量对子句集 F 进行分裂. 在进行分割子句集之间的连接关系时, 每个组件是由传播变量 (在电路里面充当导线) 连接, 并且传播变量不会出现在诊断解中, 这里优先选取传播变量.

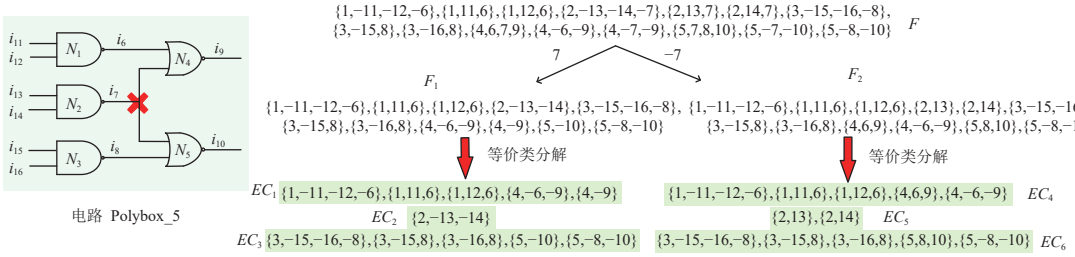


图 6 Polybox_5 电路示例图

当一个组件被设计出来时, 内部结构是已知的, MBD 编码的过程及如何把编码子句集合理分割是可以离线获得的. 对于图 5, 显然选取 i_7 (相当于切断 i_7 这根导线) 是一个合理的选择. 根据分裂规则, 选取传播变量 7 和−7 后,

左分支 $F_1=\bigcup_{1\leqslant i\leqslant 3} EC_i$, 右分支 $F_2=\bigcup_{4\leqslant i\leqslant 6} EC_i$, 每个 EC_i 分别调用 HSDiag 算法, 可以间接获得所有解, 根据推论 1 得 $HSDiag(F)=\bigcup_{1\leqslant j\leqslant m}\ \&_{1\leqslant i\leqslant k_i} HSDiag(EC_i)$, 即 $HSDiag(F)=\&_{1\leqslant i\leqslant 3} HSDiag(EC_i)\cup\ \&_{4\leqslant i\leqslant 6} HSDiag(EC_i)$. 而碰集算法需要归并所有的变量, 并且合并的解集需要复杂的极小化, 即 $MHS(f)=\min\{7\cup\ \&_{1\leqslant i\leqslant 3} MHS(EC_i)\}\cup\{-7\cup\ \&_{4\leqslant i\leqslant 6} MHS(EC_i)\}\}$.

可见优化的变种碰集算法在原始问题子句集不可分的情况下, 对传播变量进行分裂仍然可以使用等价类策略. 而传统的碰集算法在对不可分的子句集使用等价类策略时, 会额外增加对全集解集的极小化时间.

算法 2 是 HSDiag 算法结合等价类策略求解诊断, 初始时利用待诊断组件结构得到要分割子句的所有变量集合 E. 根据系统描述对观测进行单元传播, 生成的子句集做一个整体存入 *CluseF* 中, 此时大小为 1 (第 1 行). *CluseF* 对选取的变量进行集合分裂, 生成子句集 F_l, F_r 存入临时变量 *Temp_F* 中, 继续集合分裂 (第 2–10 行). *CluseF* 中所有的子句集进行等价类分解存入 *EC_Arr* 中 (第 12, 13 行), 其中等价类分解采用文献 [15] 中 *Distribution* 算法. 接着对每一个等价类调用 HSDiag 算法, 并对等价类生成的解集笛卡尔乘积 (第 14, 15 行), 最后合并每一个子句集中的解集 (第 16 行).

算法 2. HSDiag with equivalence class.

输入: 系统描述 *SD*, 系统观测 *Obs*, 组件 *Comps*, 变量集合 $E=\{e_1, e_2,\ldots, e_m\}$;
输出: 诊断 *D*.

1. *CluseF* ← *SD*. 单元传播 (*Obs*);
2. **FOR** (i=1; $i\leqslant m$; ++i){
3. **FOR** (j=1; $j\leqslant$ *CluseF.size*; ++j){
4. **IF** ($\exists e_i\in F_j, F_j\in$ *CluseF*){ //如果子句集中含有该变量, 调用集合分裂规则
5. $F_l\leftarrow\{S_i-\{-e_i\}|S_i\in F_j\wedge e_i\notin S_i\}$;
6. $F_r\leftarrow\{S_i-\{e_i\}|S_i\in F_j\wedge -e_i\notin S_i\}$;
7. *Temp_F* ← *Temp_F* ∪ $F_l\cup F_r$; //*Temp_F* 是一个临时变量
8. }
9. }
10. *CluseF.clear*; *CluseF* ← *Temp_F*; *Temp_F.clear*;
11. }
12. **FOR** ($\exists F_j, F_j\in$ *CluseF*){
13. *EC_Arr* ← *Distribution*(F_j); //等价类分解
14. **FOR** ($\exists EC_i, EC_i\in$ *EC_Arr*) //对等价类进行笛卡尔乘积
15. *D_Arr* ← *D_Arr* & *HSDiag*(EC_i);
16. *D* ← *D* ∪ *D_Arr*;
17. }

在分成等价类的过程中, 合理的分断点能够快速提升优化效率. 而在分割电路时, 若选取的传播变量较多, 那么相当于枚举大部分的解集, 反而牺牲了碰集算法本身的启发式策略带来的优化. 对待诊断组件进行诊断编码是已知组件内部结构, 先合理的离线分割等价类是可行的. 算法 2 的第 14, 15 行时间复杂度分析如下.

设 n 代表编码子句中集合的个数, 没有使用等价类策略时生成诊断解 $O(2^n)$. 等价类分解时需要选取 m 个变量, 最坏情况下生成 2^m 个子句集, 每个子句集分成 k 个等价类. 则通过等价类分解需要求出诊断解为 $O(2^{m+(n-m)/k})$. 最坏情况下 $2^n/2^{m+(n-m)/k}\leqslant 1$, 得出 $(n-m)(1/k-1)\geqslant 0$, 当 k=1 时, 选取 m 个变量没有分割成等价类, 时间复杂度 $T=O(2^n)$; 最好情况下 m=0, k=n, 子句集中所有子句都可以独立为等价类, $T=O(n)$. 一些其他情况:

m=2, k=2, $T=O(2\times 2^{n/2})$;

m=4, k=4, $T=O(2^3\times 2^{n/4})$;

m=6, k=6, T=O($2^5\times 2^{n/6}$);

…

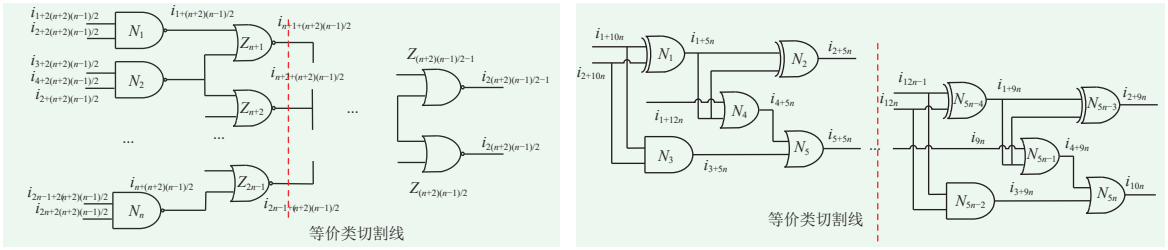
由时间复杂度分析, 当解集 $2^n>>2^m$ 时, 选取变量越少, 分成的等价类越多, 则优化越明显; 反之, 选取变量越多, 分成的等价类越少, 则等价类带来的收益就越小.

3 实验分析

在本节中我们对新提出的 HSDiag 算法和使用等价类优化后的 HSD-EC 算法进行实验分析, 对比算法选取目前求解极小冲突最先进的 MARCO-ABC 算法[7]以及求解极小碰集的 LBX 算法, 命名 MCS-MHS 算法. 实验平台如下: Ubuntu 16.04 Linux Intel Xeon E5-1607@3.00 GHz 16 GB RAM C++, 实验测试用标准的 Polybox, Fulladder 电路和国际基准电路 ISCAS-85[35,36].

3.1 HSDiag 算法与 MCS-MHS 算法实验对比

图 7 是 n 个组件的 Polybox_n 型和 n 位全加器 Fulladder_n 型电路示例图, 这两种类型电路数据来自于文献 [1,8,9], 对于 Polybox_n 电路, 除了第一列组件为与非门, 其余组件都是或非门. 全加器 Fulladder_n 电路是由 n 个 Fulladder_1 电路组成, 其中前一个全加器的输出是后一个的输入. 图 7 还为使用优化等价类策略标记了分割(选取变量) 位置, 具体信息看第 3.2 节.



(a) Polybox_$(n$+2$)(n$–1$)$/2 circuit (b) Fulladder_n circuit

图 7 Polybox_n 和 Fulladder_n 电路

表 1 记录了测试数据中电路的基本信息, |*Comps*|、|*Obs*|、|*SD*|、|*VAR*|分别代表组件个数、观测个数、子句个数、变量个数. 其中 Polybox 和 Fulladder 电路随着 n 的变化而变化. 对于 ISCAS-85 电路, 我们随机设置了 20–40 个故障组件, 并使用文献 [28] 中的 3 种故障类型. 每个电路共有 $2^{|Obs|}$种观测值, 我们随机生成 100 组不同观测变量, 表 1 的右 3 列统计了两种不同算法在 1 000 s 内能够解决的实例数占比 (如第 3 行, HSDiag 算法解决实例的占比是 1, MCS-MHS 算法解决实例的占比也是 1, 但是在运行时间上来说, HSDiag 算法在能够解决实例的前提下, 比 MCS-MHS 算法解决实例用时都短). 对于一些规模相对较小的电路 (c499, c1355, c880, c1908), HSDiag 算法和 MCS-MHS 算法都能很好地解决大部分实例, 其中只有 c1355 电路, MCS-MHS 算法少量数据好于 HSDiag 算法, 其余数据 HSDiag 算法要更有效.

表 1 电路基本信息及测试用例比较

名称	电路基本信息				解决实例的占比		获胜占比
	|*Comps*|	|*Obs*|	|*SD*|	|*VAR*|	HSDiag	MCS-MHS	HSDiag-WIN
c499	202	73	714	445	1	1	1
c880	383	86	1 112	826	1	0.7	1
c1355	546	73	1 610	1 133	1	1	0.8
c1908	880	58	2 378	1 793	1	1	1
c3540	1 669	72	4 608	3 388	0.8	0.2	0.8
c6288	2 416	64	7 216	4 864	0.5	0	0.5
Polybox_n	n	$\sqrt{8n+9}+1$	3n	$2n-1/2+\sqrt{2n+9/4}$	0.9	0.7	0.9
Fulladder_n	5n	3n+3	17n	12n+1	1	0.8	1

图 8 是 Polybox_*n* 和 Fulladder_*n* 电路示例图, 电路规模随着 *n* 的增大而增大. 当 *n* 增大时, 基本上所有电路的求解时间都增加. 电路组件少, 生成的变量和子句个数少, HSDiag 和 MCS-MHS 算法基本上都能在 1 s 内返回所有的解. 对于 Fulladder_6, Fulladder_10, Fulladder_14 电路, 优化效率在 1–3 个数量级之间, 而相对复杂的 Polybox 电路, 平均优化效率也在 5 倍以上.

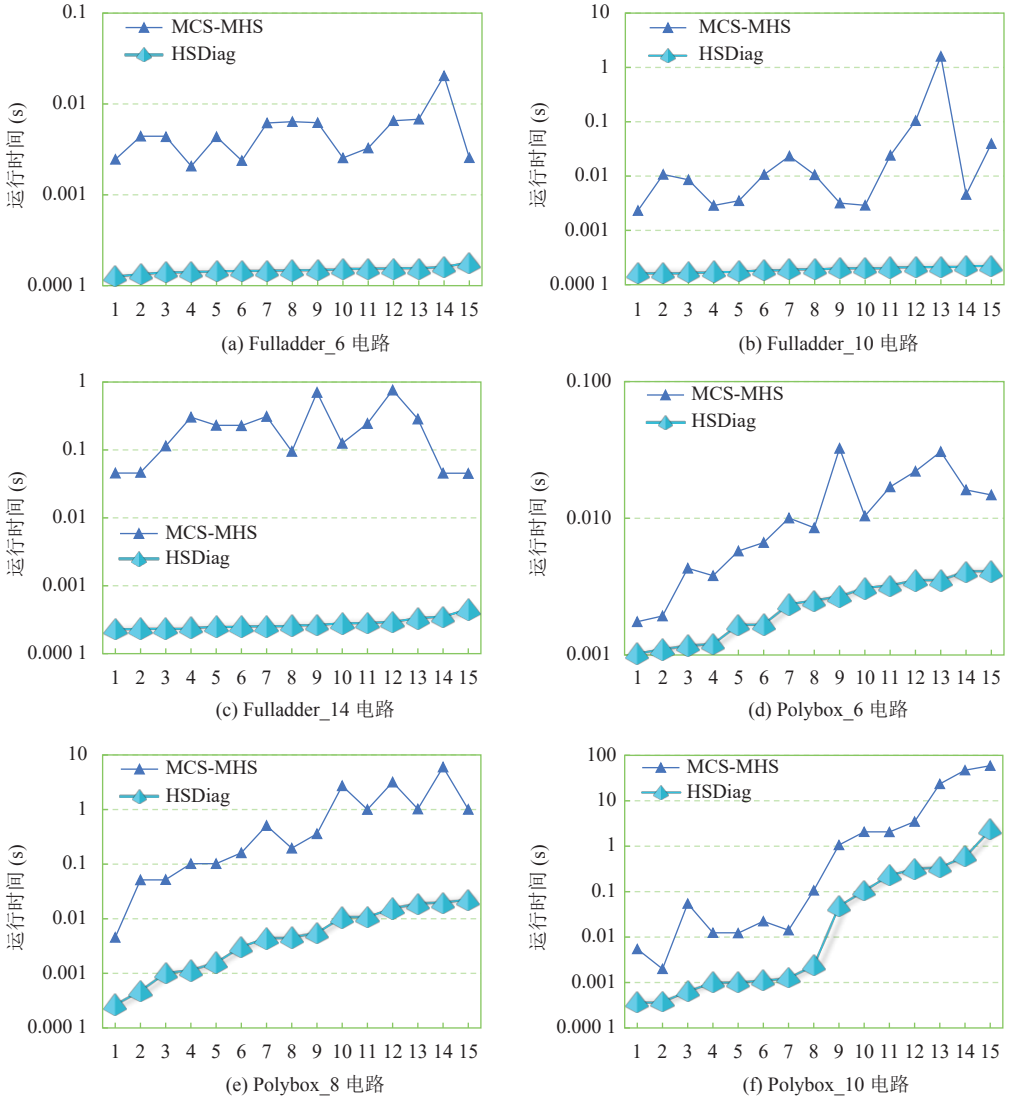


图 8 HSDiag 与 MCS-MHS 算法在 Polybox, Fulladder 电路实验比较

图 9 是在国际基准 ISCAS-85 电路[24,28,29]进行求解, 对于该大规模电路, 因在有限的时间内很难求出所有的解, 故算法截止时间设计为 1 000 s, 横坐标表示电路实例数, 纵坐标表示求解的诊断数. 在所有的电路中, HSDiag 算法求解诊断个数明显多于 MCS-MHS 算法, 对于更大的电路 c880, c3540, c6288, 基于冲突集计算所有诊断的 MCS-MHS 算法在某些情况下求出的解集更少, 甚至求不出来任何解. 这是求解大规模电路的所有极小冲突集是耗时严重的, 而计算出部分冲突集的极小碰集不一定能够满足多故障系统观测. 除了 c6288 电路外, 基本上所有的电路都求到 2×10^6 左右个解. c6288 电路中 88.1% 的组件都是异或门, 在对系统进行编码时, 每个电路门都会生成 4 个子

句, 并且该电路的输入输出也较少, 对观测进行单元传播时, 删除的子句少, 剩余的子句多, 导致整体求解困难, 仅求出约 3.5×10^4 个解.

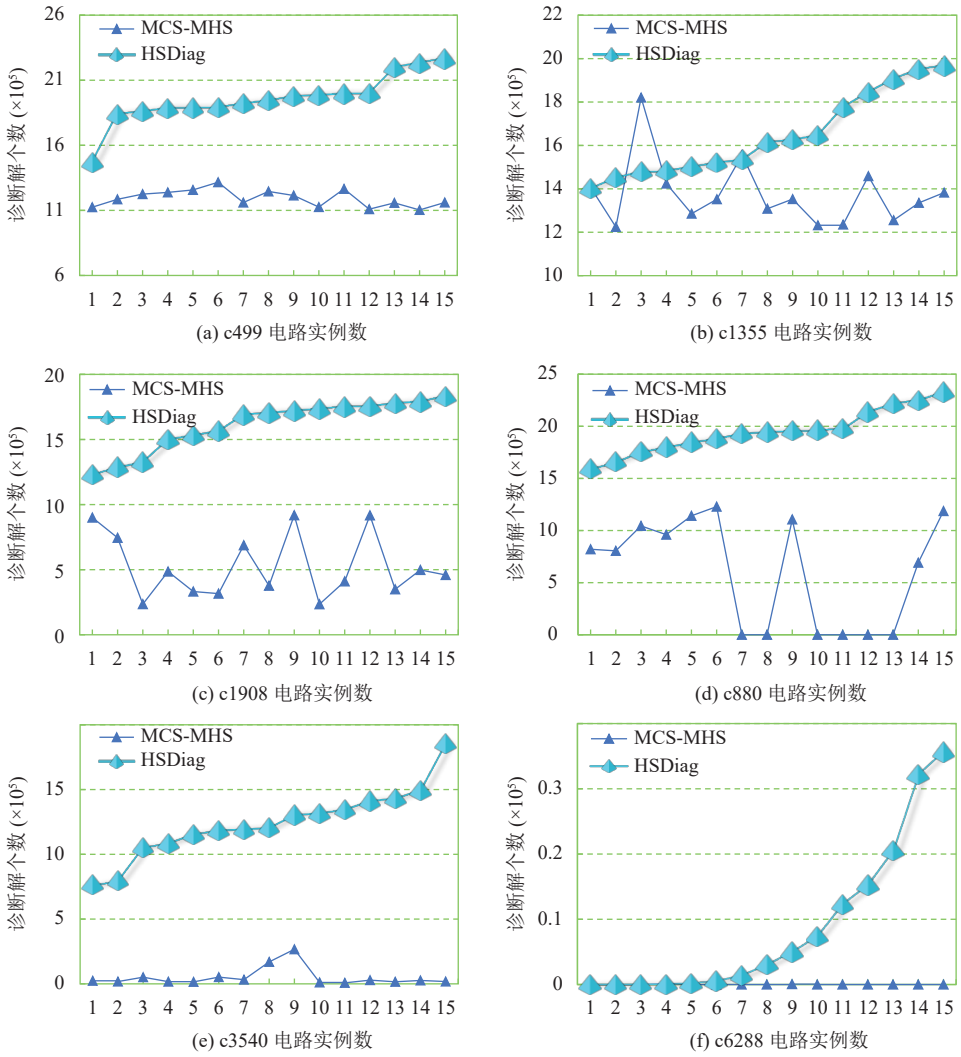


图 9 HSDiag 与 MCS-MHS 算法在 ISCAS-85 电路实验比较

图 10 比较不同截止时间内这两种算法求出的诊断解数, 其中横坐标表示截止的时间 (s), 纵坐标表示诊断解个数. 随着求解时间的增长, HSDiag 算法和 MCS-MHS 算法求解诊断的个数都在增加, 然而二者求出的解集增长速度越来越缓慢, 这是在每求出一个解时, 都需要进行极小化来保证解集的极小性, 随着解集的增大, 极小化时间增加, 耗时也越来越多. c499、c1908、c1355、c880 这 4 个电路, 在求解时间 100–300 s 时, 二者求出的解集相差不大, HSDiag 比 MCS-MHS 算法多求出 30% 左右解集, 而随着求解时间的增长, 二者求出解集之差多达 2 倍. MCS-MHS 算法在求解诊断时需要找到关键的极小冲突集, 在遍历哈斯图时, 因生成的节点个数较多, 求出的冲突集往往不是极小的. 随着计算时间的增加, 计算出的冲突集越多, 极小碰集中的冗余解就越多, 故求出的解集增长越缓慢. HSDiag 算法可以通过极大可满足子集直接获得一个诊断, 因在获得一个诊断时也间接的访问其他诊断解, 故可以快速获得更多诊断. c3540、c6288 电路, 生成的子句较多, 求解冲突集和求解极大可满足子集耗时都增加, 二者求出的解集明显减少, 但是 HSDiag 算法仍能够给出部分诊断.

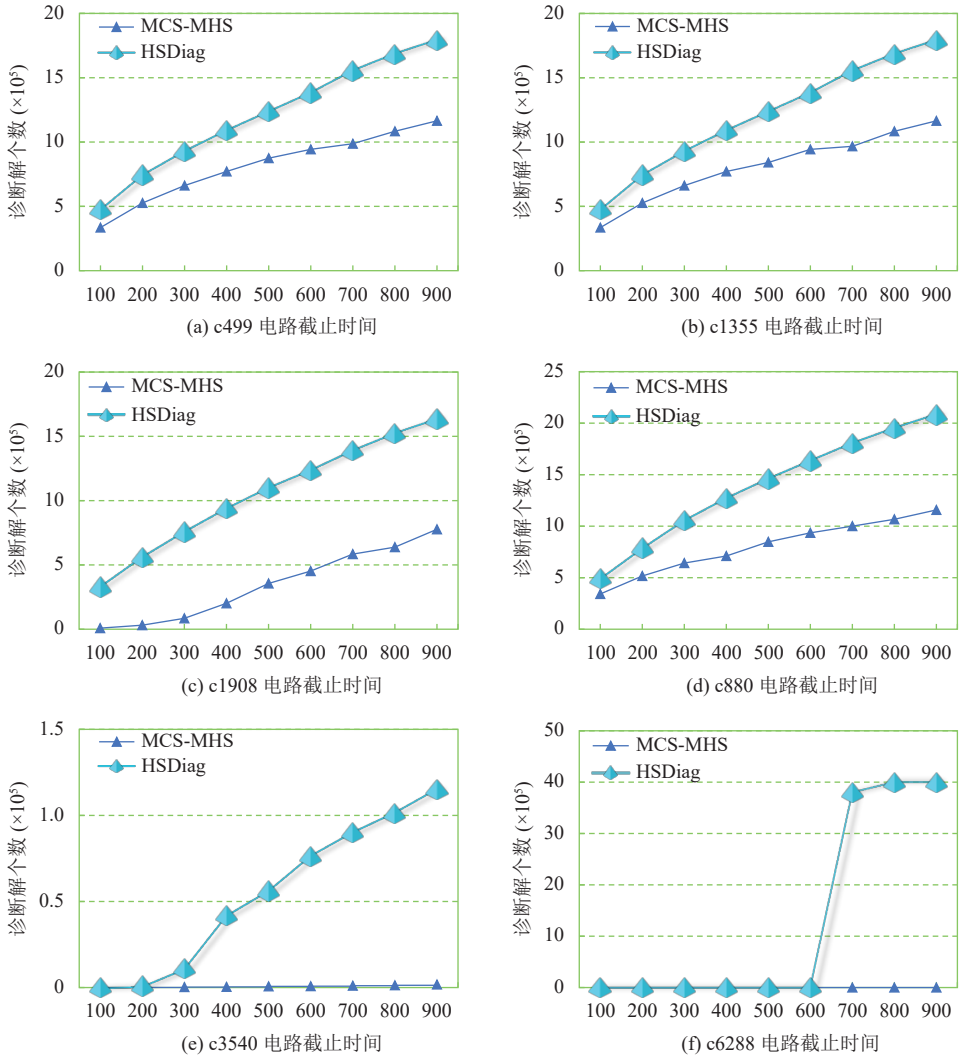


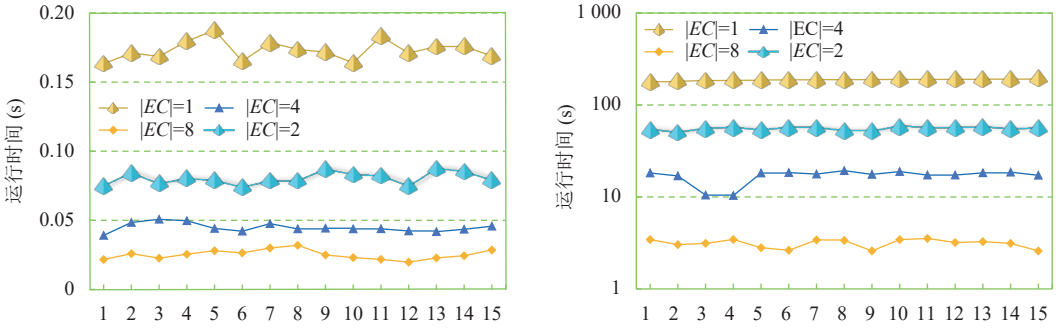
图 10 HSDiag 与 MCS-MHS 算法在 ISCAS-85 电路实验比较

3.2 等价类优化

在图 7 中, 虚线给出了等价类的切割位置, 对于 Polybox_(n+2)(n–1)/2 电路, 从左往右第 i 列组件个数为 n+1–i, 第 i 列与第 i+1 列之间变量 (导线) 的个数为 n+1–i. 也就是说, 如果从第 i 列和第 i+1 列切割电路, 那么需要选取 n+1–i 个变量. Fulladder_n 电路可以看成是由 n 个 Fulladder_1 组成, 只不过第 i 个电路的输出是第 i+1 个电路其中的一个输入, 可见分割 Fulladder_n 电路只需要选取中间的连接变量.

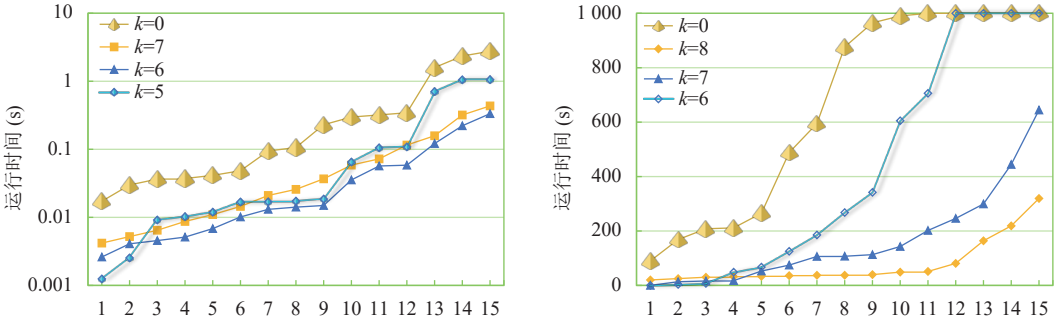
图 11 是 HSDiag 算法在 Fulladder_n 电路进行等价分类, 其中$|EC|$=1 代表没有使用等价类策略, $|EC|$=8 代表把电路编码均分成 8 个等价类. Fulladder 电路结构简单, 能够容易的划分等价类, 在每次分割时, 分割后等价类没有交集, 相当于对称分割, 即$|EC|=2^k$ 时, 只需要选取 k 个传播变量 (从图 7 也可以看出). 在分解等价类时, 需要对选取的变量进行赋值, 然赋值过程中就已经相当于求解. 我们把分解等价类选取变量阶段放到离线, 即获取分割点是在对系统编码时获取, 在算法开始阶段, 直接当作参数直接调用. 在 n=1000 时, 编码的变量及诊断解的个数较少, 所以整体求解效率较快, 平均 0.2 s 内就能够返回所有的解. $|EC|$=8 时, 比没有调用等价类策略的 HSDiag 算法, 效率提升约 5 倍. 而 n=16000 时, 编码整体规模较大, 求解时间都在 100 s 以上, 使用等价类策略后, 大幅度缩减编码规模, 效率最高提升 40 倍.

图 12 显示在复杂电路 Polybox 中生成两个等价类, k 代表选取变量的个数, 等于 0 时表示没有使用等价类策略. 在分割小规模的电路时, 虽然使用等价类策略对电路进行缩减, 但是也分成了 2^k 个集合簇, 而每个集合簇的求解时间相似, 当等价类获得收益较小时, 即部分集合簇解集较少甚至无解, k 值越大反而求解的时间不一定越小. 在 Polybox_54 中, k=6 时整体求解效率要好于 k=5, 7, 比 k=0 时平均提升一个数量级. 在 Polybox_90 电路中, 使用等价类策略的 HSDiag 算法求解时间都明显缩减, 因随着 k 的增大等价类分解的组件个数也逐渐均衡, 故求解时间也随着降低. 特别是对于第 7–15 个实例, k=8 比 k=0 平均提升一个数量级以上. 同样的, 在第 1–4 个实例中并不是 k=8 最好, 一些系统观测值大部分满足了理论值, 使得生成的解集较少, 甚至大量子句集产生的解都是冗余的, 整体求解时间相对较慢.



(a) Fulladder_1000 等价类分类　　(b) Fulladder_16000 等价类分类

图 11　等价类比较在 Fulladder 电路



(a) Polybox_54 选取 k 个变量分成等价类　　(b) Polybox_90 选取 k 个变量分成等价类

图 12　等价类比较在 Polybox 电路

4 总　结

基于模型诊断是根据待诊断的系统的内部结构编码成 CNF 子句, 通过一致性检测来确定故障组件的位置. 利用 SAT 求解器先求出极小冲突集再求出极小碰集. 这种策略虽然可以获得诊断, 但是耗时严重, 并且因求出所有极小冲突集是不切实际的, 故获得的诊断并不精确. 本文利用极大可满足子集与极小修正集的互补性, 提出一种变种碰集算法 HSDiag. 该算法能够直接获得候选诊断, 和目前性能最优间接求诊断的算法相比, 在求解效率及诊断的数量上都有明显的优化, 运行时间有 5–100 倍的提升, 诊断数量能多获得 1 倍以上. 为了在更大规模的电路上有效, 本文又提出一种用于求解候选诊断的等价类策略, 它能够大幅度缩减编码子句的数量, 在本文所提算法的基础上, 运行时间进一步提升 2 倍以上.

随着科技的不断进步, 集成电路的复杂度不断增加, 使得测试电路成本不断提高, 当集成电路在测试集的激励下得到与预期不同的输出响应时, 使用基于模型的诊断方法进行分析, 从而得到集成电路的候选故障信息, 可以大幅度地减少测试的成本. 除此之外, 配电网的故障诊断也尤为重要, 国家电网面积大且错综复杂, 由于硬件老化、

气候、人为等因素, 出现故障频率较高, 若能够在短时间内找到故障点, 防止大面积停电等事故, 可减少大量的经济损失.

References:

[1] Reiter R. A theory of diagnosis from first principles. Artificial Intelligence, 1987, 32(1): 57–95. [doi: 10.1016/0004-3702(87)90062-2]

[2] Mordoch A, Juba B, Stern R. Learning safe numeric action models. In: Proc. of the 37th AAAI Conf. on Artificial Intelligence. Washington: AAAI, 2023. 12079–12086. [doi: 10.1609/aaai.v37i10.26424]

[3] Wang QJ, Jin T, Wang MQ. A hierarchical minimum hitting set calculation method for multiple multiphase faults in power distribution networks. IEEE Trans. on Industrial Electronics, 2021, 68(1): 4–14. [doi: 10.1109/tie.2020.2967691]

[4] Torlak E, Chang FSH, Jackson D. Finding minimal unsatisfiable cores of declarative specifications. In: Proc. of the 15th Int'l Symp. on Formal Methods. Turku: Springer, 2008. 326–341. [doi: 10.1007/978-3-540-68237-0_23]

[5] Wang QJ, Jin T, Mohamed MA. A fast and robust fault section location method for power distribution systems considering multisource information. IEEE Systems Journal, 2022, 16(2): 1954–1964. [doi: 10.1109/JSYST.2021.3057663]

[6] Jose M, Majumdar R. Cause clue clauses: Error localization using maximum satisfiability. In: Proc. of the 32nd ACM SIGPLAN Conf. on Programming Language Design and Implementation. San Jose: ACM, 2011. 437–446. [doi: 10.1145/1993498.1993550]

[7] Jannach D, Schmitz T. Model-based diagnosis of spreadsheet programs: A constraint-based debugging approach. Automated Software Engineering, 2016, 23(1): 105–144. [doi: 10.1007/s10515-014-0141-7]

[8] Luo C, Xing WQ, Cai SW, Hu CM. NuSC: An effective local search algorithm for solving the set covering problem. IEEE Trans. on Cybernetics, 2024, 54(3): 1403–1416. [doi: 10.1109/TCYB.2022.3199147]

[9] Cai SW, Li BH, Zhang XD. Local search for satisfiability modulo integer arithmetic theories. ACM Trans. on Computational Logic, 2023, 24(4): 32. [doi: 10.1145/3597495]

[10] Jiang LY, Ouyang DT, Dong BW, Zhang LM. Enhanced pruning scheme for enumerating MUS. Ruan Jian Xue Bao/Journal of Software, 2024, 35(4): 1964–1979 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/6845.htm [doi: 10.13328/j.cnki.jos.006845]

[11] Jiang LY, Ouyang DT, Zhang Q, Zhang LM. DeciLS-PBO: An effective local search method for pseudo-boolean optimization. Frontiers of Computer Science, 2024, 18(2): 182326. [doi: 10.1007/s11704-023-3018-8]

[12] Ouyang DT, Gao H, Tian NY, Liu M, Zhang LM. MUS enumeration based on double-model. Journal of Computer Research and Development, 2019, 56(12): 2623–2631 (in Chinese with English abstract). [doi: 10.7544/issn1000-1239.2019.20180852]

[13] Zhao XF, Ouyang DT. Deriving all minimal conflict sets using satisfiability algorithms. Acta Electronica Sinica, 2009, 37(4): 804–810 (in Chinese with English abstract). [doi: 10.3321/j.issn:0372-2112.2009.04.024]

[14] Ignatiev A, Morgado A, Marques-Silva J. RC2: An efficient MaxSAT solver. Journal on Satisfiability, Boolean Modeling and Computation, 2019, 11(1): 53–64. [doi: 10.3233/SAT190116]

[15] Liffiton MH, Malik A. Enumerating infeasibility: Finding multiple MUSes quickly. In: Proc. of the 10th Int'l Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems. Yorktown Heights: Springer, 2013. 160–175. [doi: 10.1007/978-3-642-38171-3_11]

[16] Liffiton MH, Previti A, Malik A, Marques-Silva J. Fast, flexible MUS enumeration. Constraints, 2016, 21(2): 223–250. [doi: 10.1007/s10601-015-9183-0]

[17] Jiang YF, Lin L. The computation of hitting sets with Boolean formulas. Chinese Journal of Computers, 2003, 26(8): 919–924 (in Chinese with English abstract). [doi: 10.3321/j.issn:0254-4164.2003.08.004]

[18] Pill I, Quaritsch T. Optimizations for the boolean approach to computing minimal hitting sets. In: Proc. of the 20th European Conf. on Artificial Intelligence. Montpellier: IOS, 2012. 648–653. [doi: 10.3233/978-1-61499-098-7-648]

[19] Zhao XF, Ouyang DT. Deriving all minimal hitting sets based on join relation. IEEE Trans. on Systems, Man, and Cybernetics: Systems, 2015, 45(7): 1063–1076. [doi: 10.1109/TSMC.2015.2400423]

[20] Jannach D, Schmitz T, Shchekotykhin K. Parallelized hitting set computation for model-based diagnosis. In: Proc. of the 29th AAAI Conf. on Artificial Intelligence. Austin: AAAI, 2015. 1503–1510. [doi: 10.1609/aaai.v29i1.9389]

[21] Liu SG, Ouyang DT, Zhang LM. Method of minimality-checking of candidate solution for minimal hitting set algorithm. Ruan Jian Xue Bao/Journal of Software, 2018, 29(12): 3733–3746 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/5311.htm [doi: 10.13328/j.cnki.jos.005311]

[22] Zhao XF, Huang S, Tong XR, Ouyang DT, Zhang LM, Zhang XL. IBWIICC: Incrementally computing minimal hitting-sets combing local independence cover checking. Acta Electronica Sinica, 2022, 50(11): 2722–2729 (in Chinese with English abstract). [doi: 10.12263/DZXB.20211356]

[23] Huang J, Chen L, Zou P. A compounded genetic and simulated annealing algorithm for computing minimal diagnosis. Ruan Jian Xue Bao/Journal of Software, 2004, 15(9): 1345–1350 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1345.htm

[24] Gainer-Dewar A, Vera-Licona P. The minimal hitting set generation problem: Algorithms and computation. SIAM Journal on Discrete Mathematics, 2017, 31(1): 63–100. [doi: 10.1137/15M1055024]

[25] Abreu R, van Gemund AJC. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis. In: Proc. of the 8th Symp. on Abstraction, Reformulation, and Approximation. Lake Arrowhead: AAAI, 2009. 2–9.

[26] Marques-Silva J, Janota M, Ignatiev A, Morgado A. Efficient model based diagnosis with maximum satisfiability. In: Proc. of the 24th Int'l Joint Conf. on Artificial Intelligence. Buenos Aires: AAAI, 2015. 1966–1972.

[27] Lamraoui SM, Nakajima S. A formula-based approach for automatic fault localization of multi-fault programs. Journal of Information Processing, 2016, 24(1): 88–98. [doi: 10.2197/ipsjjip.24.88]

[28] Ignatiev A, Morgado A, Weissenbacher G, Marques-Silva J. Model-based diagnosis with multiple observations. In: Proc. of the 28th Int'l Joint Conf. on Artificial Intelligence. Macao: AAAI, 2019. 1108–1115. [doi: 10.24963/ijcai.2019/155]

[29] Rodler P. Sequential model-based diagnosis by systematic search. Artificial Intelligence, 2023, 323: 103988. [doi: 10.1016/j.artint.2023.103988]

[30] Köhl MA, Hermanns H. Model-based diagnosis of real-time systems: Robustness against varying latency, clock drift, and out-of-order observations. ACM Trans. on Embedded Computing Systems, 2023, 22(4): 68. [doi: 10.1145/3597209]

[31] Stern R, Kalech M, Rogov S, Feldman A. How many diagnoses do we need? Artificial Intelligence, 2017, 248: 26–45. [doi: 10.1016/j.artint.2017.03.002]

[32] Siddiqi S, Huang JB. Hierarchical diagnosis of multiple faults. In: Proc. of the 20th Int'l Joint Conf. on Artificial Intelligence. Hyderabad: ACM, 2007. 581–586.

[33] Mencia, C, Previti A, Marques-Silva J. Literal-based MCS extraction. In: Proc. of the 24th Int'l Joint Conf. on Artificial Intelligence. Buenos Aires: ACM, 2015. 1973–1979.

[34] Zhou HS, Ouyang DT, Zhao XF, Zhang LM. Two compacted models for efficient model-based diagnosis. In: Proc. of the 36th AAAI Conf. on Artificial Intelligence. AAAI, 2022. 3885–3893. [doi: 10.1609/aaai.v36i4.20304]

[35] Ouyang JH, Huang S, Zhang LM, Zhao XF. Model-based diagnosis with low-cost fault identification. Frontiers of Computer Science, 2025, 19(5): 195333. [doi: 10.1007/s11704-024-40393-y]

[36] Kurtoglu T, Narasimhan S, Poll S, Garcia D, Kuhn L, de Kleer J, van Gemund A, Feldman A. First international diagnosis competition—DXC'09. In: Proc. of the 20th Int'l Workshop on Principles of Diagnosis. Stockholm: NASA, 2009. 383–396.

附中文参考文献:

[10] 蒋璐宇, 欧阳丹彤, 董博文, 张立明. 针对 MUS 求解问题的加强剪枝策略. 软件学报, 2024, 35(4): 1964–1979. http://www.jos.org.cn/1000-9825/6845.htm [doi: 10.13328/j.cnki.jos.006845]

[12] 欧阳丹彤, 高菡, 田乃予, 刘梦, 张立明. 基于双模型的 MUS 求解方法. 计算机研究与发展, 2019, 56(12): 2623–2631. [doi: 10.7544/issn1000-1239.2019.20180852]

[13] 赵相福, 欧阳丹彤. 使用 SAT 求解器产生所有极小冲突部件集. 电子学报, 2009, 37(4): 804–810. [doi: 10.3321/j.issn:0372-2112.2009.04.024]

[17] 姜云飞, 林笠. 用布尔代数方法计算最小碰集. 计算机学报, 2003, 26(8): 919–924. [doi: 10.3321/j.issn:0254-4164.2003.08.004]

[21] 刘思光, 欧阳丹彤, 张立明. 极小碰集求解中候选解极小性判定方法. 软件学报, 2018, 29(12): 3733–3746. http://www.jos.org.cn/1000-9825/5311.htm [doi: 10.13328/j.cnki.jos.005311]

[22] 赵相福, 黄森, 童向荣, 欧阳丹彤, 张立明, 章星林. IBWIICC: 结合局部独立覆盖检测策略增量求解极小碰集的算法. 电子学报, 2022, 50(11): 2722–2729. [doi: 10.12263/DZXB.20211356]

[23] 黄杰, 陈琳, 邹鹏. 一种求解极小诊断的遗传模拟退火算法. 软件学报, 2004, 15(9): 1345–1350. http://www.jos.org.cn/1000-9825/15/1345.htm

欧阳继红(1964－), 女, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为基于模型的故障诊断, 机器学习.

黄森(1995－), 男, 博士生, 主要研究领域为基于模型的故障诊断, SAT.