计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(8): 1850−1861, 2025

基于模型诊断中隐式碰集的获取与判断

欧阳继红　黄　森
（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）
（jihongouyang@yahoo.com）

Acquisition and Judgement of Implicit Hitting Sets for Model-Based Diagnosis

Ouyang Jihong and Huang Sen
（*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　Model-based diagnosis (MBD) mainly models the behavior of the system, and once the abnormal behavior is observed, a diagnosis algorithm is run on the system model to return a possible explanation. The existing diagnosis algorithm computes a minimal hitting set (MHS) each time a conflict set is identified, and then verifies whether this MHS satisfies the system observations. While this approach reduces the generation of redundant solution sets, the difficulty of computing the MHSs of conflict sets increases exponentially with the number of conflict sets. Since computing the MHS of a partial conflict set is not necessarily a diagnosis, it is also time-consuming to check whether the MHS satisfies the system observations. We design a filtering function to remove low-quality conflict sets based on the diagnosis cardinality and quantity, while ensuring that the obtained hitting sets are as diagnosis as possible. In addition, to facilitate the rapid verification of hitting sets for diagnosis, we propose an efficient decision algorithm based on the logical relationships of the circuit. In the experimental section, we conduct a detailed analysis to compare the runtime and diagnosis yield under varying numbers of fault conditions. Compared with state-of-the-art algorithms, our approach shows efficiency improvements up to 2−40 times in runtime and diagnosis yield enhancements ranging from 5−200 times.

Key words　model-based diagnosis; system observation; minimal conflict set (MCS); minimal hitting set (MHS); cardinality-minimal diagnosis

摘　要　基于模型诊断主要是根据系统的行为进行建模，一旦观察到异常行为就在系统模型上运行一个诊断算法来返回可能的解释. 现有的诊断算法是每求出一个冲突集就计算一次极小碰集，然后再检验该极小碰集是否满足系统观测. 这样虽然能够减少冗余解集的生成，但是计算冲突集的极小碰集难度随冲突集数量的增加呈指数级增长，而计算部分冲突集的极小碰集不一定是诊断，检验极小碰集是否满足系统观测也是十分耗时的. 针对以上问题，设计了一个筛选函数，在保证所得的碰集尽可能是诊断的情况下，分别从诊断的势和数量上来删除低质量的冲突集. 除此之外，为了能够快速检验碰集是否是诊断，还根据电路的逻辑关系提出了一种高效的判定算法. 在实验部分，详细分析了在设置不同数量的故障条件下运行时间和求解诊断个数的比较，与目前最先进的算法相比，效率提升2~40倍，诊断数量多出5~200倍.

关键词　基于模型诊断；系统观测；极小冲突集；极小碰集；极小势诊断

收稿日期：2024-04-03；**修回日期**：2024-12-09
基金项目：国家自然科学基金项目（61876071, 62076108）
This work was supported by the National Natural Science Foundation of China (61876071, 62076108).
通信作者：黄森（hs_huangs@163.com）

中图法分类号 TP391

DOI: 10.7544/issn1000-1239.202440248 **CSTR:** 32373.14.issn1000-1239.202440248

快速寻找一些高科技人造设备的有缺陷部件，如核反应堆、航天器、智能汽车等，对我们社会的安全极其重要[1]. 传统的基于专家的诊断方法可以根据经验帮助人们发现设备的故障组件，然而关于新型高科技人造设备的经验却很少，因此传统的基于经验的诊断方法并不适用[2]. 基于模型诊断（model-based diagnosis, MBD）技术不需要专家的经验，只用考虑设备的行为和结构信息就能够找到故障组件[3]. 图 1 是 MBD 流程图：第①步根据系统的内部结构和行为进行建模；第②步给定系统的一些输入和预期输出，当预期的理论值和观察到的真实值行为存在明显的差异时，利用求解器对建模后的编码求解极小冲突集；第③步计算冲突集的极小碰集，即候选诊断；第④步利用贝叶斯定理，从众多候选诊断中计算出最有可能的故障组件[4]. MBD 最初被用于发现电子电路等硬件工件的故障[5]，后来，由于其通用性，该方法被应用于解决许多不同的问题，包括卫星决策系统[6]、汽车工业[7]、电子制表软件等[8-10].

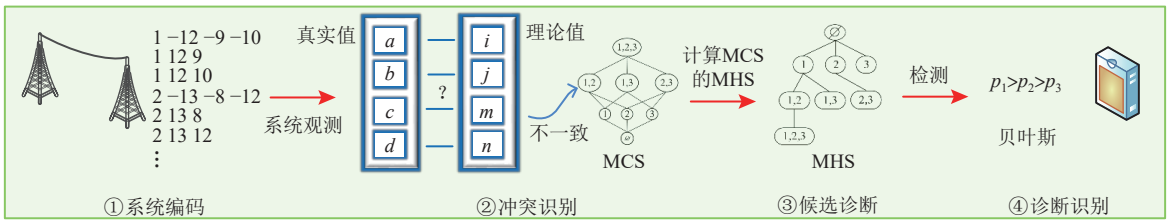


Fig. 1 Workflow diagram of MBD

图 1 MBD 工作流程图

因为被加工出来的组件的内部结构是已知的，系统编码可以在离线阶段获取[11]，诊断识别主要是在获取所有候选诊断后进行定向筛选，所以 MBD 耗时的主要阶段是冲突识别和候选诊断[12-13]. 对于冲突识别阶段一般依靠哈斯图和先进的 SAT 求解器[14]，该阶段能够快速获得大量的冲突集[15-16]. 而计算冲突集的极小碰集是一个 NP-hard 问题，计算出的极小碰集数量随冲突集数量的增加呈爆炸式增长，并且验证碰集是否是诊断所需的时间受极小碰集数量的影响，可见，候选诊断阶段在 MBD 中需要大量的时间开销[1,17].

极小碰集最早算法是由人工智能专家 Reiter[1] 提出，他利用碰集树（hitting set tree, HST）进行宽度优先的方法标记节点，并使用节点剪枝技术减少无解空间的搜索，但是由于剪枝策略可能会破坏算法的完备性. 1989 年 Greiner 等人[18] 在 HS-Tree 算法的基础上提出了 HS-DAG 算法，利用有向无环图结构对相同节点进行重用，保证了算法的完备性. 2001 年 Wotawa[19] 提出了 HST-Tree 算法，通过对元素的唯一标识，不需要有向无环图也能保证算法的完备性，同时大幅度减少了冗余节点的生成. 2003 年姜云飞等人[20] 提出了 Boolean 算法，该算法在递归时的每个节点最多生成 2 个节点，提高了计算效率. 2012 年 Pill 等人[21] 对 Boolean 算法进行改进，利用单元素集合极小化冲突集，减少了树的深度. 由于求出的碰集不一定是极小的，碰集的极小化也阻碍极小碰集的整体求解效率. 2018 年刘思光等人[22] 提出了增量独立覆盖检测（incremental independent coverage checking, IICC）算法，利用独立覆盖思想判断碰集是否是超集，提高了碰集极小化效率. 2019 年田乃予等人[23] 提出基于子集一致性检测去超集算法，缩小了解集极小化的规模. 2022 年赵相福等人[24] 针对 Boolean 算法生成节点较多问题提出了增量布尔算法 IBWIICC（incremental Boolean with IICC），该算法利用左分支增量求解右分支，减少了右分支的重复性计算. 除了基于枚举树的算法，还有一些其他策略的算法被提出[25]，如：基于 Map-Reduce 环境的 MHS^2 算法和多线程的 HS-Tree 算法[17]、基于 MCS 特殊结构的 LinerMerge 和 TreeMerge 算法[2,26]. 这些算法虽然能够快速计算极小碰集，但是由于自身的局限性只能在特定场景工作[27].

本文的主要贡献包括 3 个方面：

1）提出了一种用于筛选冲突集的 Q 函数，利用碰集与冲突集的二元性，删除了大量冗余冲突集，在有限的时间内能够获得更多的极小势碰集.

2）提出了一种数学编码的方式，在验证候选解可满足性时，只需要计算候选诊断的关键组件，减少大量不必要的计算.

3) 在 ISCAS-85 数据中, 新提出的方法能够大幅度提高求解诊断的数量和质量; 在诊断识别阶段, 提高了为获得最终的故障位置的精准度.

1 相关工作

在求解大规模电路时(极小碰集数量大于 10^{10})[11], 由于产生的冲突集数量较多, 求出所有极小碰集已经不现实. 在故障识别阶段, 如果是计算所有冲突集的极小碰集, 那么得到的极小碰集都是该系统的候选诊断, 显然, 每个极小碰集中的组件(用碰集中的元素代替组件)都出现故障的可能性随组件个数的增加而降低, 故同时出现故障组件的个数越少该极小碰集成为最终候选诊断的可能性就越大[28-30]. 2012 年 Pill 等人[21] 通过宽度优先生成树, 仅生成势最小的所有碰集, 但是这种完备算法也只能对规模较小的冲突集有效. 2014 年 Metodi 等人[3] 通过顶层诊断来求解所有极小势碰集(cardinality-minimal hitting set, CMHS), 然后通过一一替换策略扩展其他极小碰集. 2015 年 Marques-Silva 等人[4] 提出了过滤节点和过滤边概念, 在默认一些组件是正常的前提下, 能够快速地求出一个诊断, 然而这样很有可能错过其他重要信息. 2019 年 Ignatiev 等人[31] 提出了碰集二元性(hitting set dualization, HSD)算法, 该算法是每求出一个冲突集就计算已有冲突集的极小碰集, 然后通过 SAT 求解器来判断该碰集是否是诊断, 这样虽然能够快速获得一个诊断, 然而频繁的调用碰集算法, 且验证每个碰集是否是诊断时, 都耗费了大量的求解时间. 2021 年 Kalech 等人[29] 提出了 One-SAT 算法, 该算法虽然把多观测问题的数据编码成单观测算法可计算的格式, 但是随着观测次数的增加编码数量呈线性增长. 2022 年 Zhou 等人[32] 在顶层诊断的基础上提出了压缩模型 CMMO(compacted model with multiple observations)算法, 利用差异输入输出减少变量的生成, 在多观测情况下效率有一定的提升.

在 MBD 中, 获取最有可能的故障组件需要所有极小诊断的后验故障概率, 然而验证所有极小碰集非常耗时[14]. 国内外一些学者通过调用成熟的可满足性(satisfiability, SAT)求解器来提高求解效率, 如 wboinc, RC2 等[33-34], 这些求解器在计算规模较小的数据集时, 因能够求出大部分甚至所有的冲突集, 验证的极小碰集大都是诊断, 相当于在验证可满足性问题时, 大多数实例都是可满足的, 故求解效率较高. 而处理大规模电路时, 由于时间限制求不出所有的冲突集, 导致大部分极小碰集都不是诊断, 相当于大多数实例都是不可满足的, 故求解效率较低[35-36].

根据以上分析, 我们得到阻碍获得诊断的 2 个关键点, 即计算大规模冲突集的极小碰集和验证极小碰集是否是诊断. 对于前一点, 因获取最终的诊断在很大程度上取决于极小势碰集, 而又根据独立覆盖度思想, 极小势碰集中大多数元素的独立覆盖度都大于 1, 故大量的冲突集对于该极小碰集来说都是冗余的. 为了能够在有限的时间内获得最优价值的极小碰集, 我们设计了一个目标函数来缩小冲突集的规模. 该目标函数筛选冲突集主要有 2 个倾向: 首先是选取含有元素较少的冲突集, 因为在计算所有冲突集的极小碰集时, 碰集需要与所有冲突集都有交集, 所以该集合内的元素尤为重要; 其次是选取冲突集中相对元素不相同较多的集合, 这样是为了保证解集的多样性. 对于后一点, 我们根据电路门的逻辑关系, 设计了相应的数学表达式, 只需要计算候选诊断的后继组件来验证候选解是否是诊断.

2 基于模型的诊断问题

本节主要介绍 MBD 问题的相关定义及概念.

2.1 基本定义

待诊断系统可定义为一个三元组$\langle SD, Comps, Obs\rangle$. 其中, SD 是诊断系统的模型, 为一阶谓词公式的集合; $Comps$ 为系统组件的集合, 是一个有限常量集; Obs 是观测到的系统行为(例如, 观察到的系统输入和输出), 是一阶谓词公式的有限集.

系统模型 SD 为每个组件指定了其行为的一组可能假设, 它至少包含 1 个指定的正常行为模式. 如果 SD 包含一些关于组件 c 在它有故障时的行为信息, 我们说组件 c 有一个强故障模型; 否则, c 有一个弱故障模型. MBD 问题的一个解决方案是一个诊断. 诊断是一组关于系统组件行为的假设, 这些假设构成了对观察到的行为的合理解释. 为了正式定义诊断, 我们为每个组件 c 关联一个变量 $h(c)$, 称为健康变量, 其值是观察到组件 c 的行为模式. 在逻辑术语中, $h(c)=1$ 代表组件 c 故障, $h(c)=0(\neg h(c))$ 代表组件 c 正常. 如果所有组件分配其正常行为模式的 SD 和 Obs 不一致(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), 那么至少有 1 个组件有故障.

定义 1. 诊断(diagnosis)[17]. 当 $SD\wedge Obs\wedge \{h(c)|c\in\Delta, \Delta\subseteq Comps\}\wedge\{\neg h(c)|c\in Comps-\Delta, \Delta\subseteq Comps\}$是一致的,

则Δ被定义为该系统的诊断.

如果Δ的任意真子集都不是诊断，那么Δ是一个极小诊断. 对于该系统的任意一个诊断Δ'，如果$|\Delta'|\geqslant|\Delta|$，那么$\Delta$是一个极小势诊断.

定义 2. 极小冲突集(minimal conflict set, MCS)[1]. 当$SD\wedge Obs\wedge\{\neg h(c)|c\in f, f\subseteq Comps\}$不一致时，则$f$被定义为该系统的冲突集.

如果f的任意真子集都不是冲突集，那么f为极小冲突集. F是冲突集簇，$F=\{f_1, f_2, \cdots, f_n\}$.

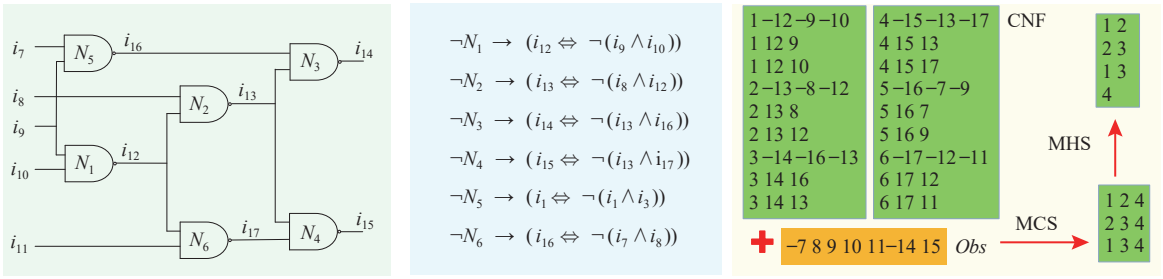
定义 3. 碰集(hitting set, HS)[1]. 当S与任意f_i交集不为空，$f_i\in F$，则S被定义为F的碰集.

如果S的任意真子集都不是F的碰集，那么S是一个极小碰集；对于任意一个极小碰集S，如果$|S'|\geqslant|S|$，那么S是一个极小势碰集.

定义 4. 隐式碰集(implicit hitting set, HIS)[31]. F'是F的真子集，若S是F'的碰集又是F的碰集，那么S是F的一个隐式碰集.

下面通过例 1 对以上定义进一步说明.

例 1. 图 2 描述了 ISCAS-85 数据中的 c17 电路，给定一组观测 *Obs*，计算所有的候选诊断.



(a) c17电路 (b) 系统描述 (c) 诊断算法

Fig. 2 c17 circuit fault identification

图 2 c17 电路故障识别

在模型诊断中，合取范式(conjunctive normal form, CNF)公式可以用子句集表示[31]，子句可以表示文字的集合，每个文字可以用变量表示，每个变量的取值为x或者补集$\neg x$. 在电路的系统描述中，每一个元件的输入输出均可以转化为 CNF 的形式，将转化的 CNF 子句存储到文件后，再使用 SAT 求解器读取此 CNF 文件进行求解，便将一个诊断问题转化为 SAT 问题. 对于图 2 中的示例图，其中图 2(a)是 ISCAS-85 数据中 c17 电路，它由 6 个与非门$\{N_1, N_2, N_3, N_4, N_5, N_6\}$组成，其中只有系统的输入$\{i_7, i_8, i_9, i_{10}, i_{11}\}$和输出$\{i_{14}, i_{15}\}$是可观测的. 图 2(b)是描述电路的门的性质(在该电路中描述的都是对应的与非门)，图 2(c)是 c17 电路转化成 CNF 公式的部分主要数据. 在计算诊断前对图 2(a)中的编码进行了一个映射(用数字代替变量，如$7=i_7$, $5=N_5$). 通过系统观测可知，系统输入$\{i_7, i_8, i_9, i_{10}, i_{11}\}$的值是$\{-7, 8, 9, 10, 11\}$，代表$\{0, 1, 1, 1, 1\}$(0 表示低电平，1 表示高电平)，假设电路中每一个组件都是正常工作的，那么得到系统输出$\{i_{14}, i_{15}\}$是$\{14, -15\}$，而系统观测到的是$\{-14, 15\}$，显然，该系统中必然存在至少 1 个故障组件才能解释观测情况.

首先利用 MCS 算法(每个冲突集中所有组件都正常工作是不可能的)求出所有极小冲突集$\{1, 2, 4\}$, $\{2, 3, 4\}$, $\{1, 3, 4\}$，然后利用 MHS 算法结合每个冲突集最少 1 个故障组件求出所有极小碰集$\{1, 2\}$, $\{2, 3\}$, $\{1, 3\}$, $\{4\}$，其中$\{4\}$是极小势碰集. 如：$\{4\}$(对应N_4组件)组件故障，通过系统输入可以得到和系统观测一样的输出值，说明该碰集是一个诊断. 可见获得系统诊断取决于计算 MCS 的 MHS.

在每个组件被生产出来时，我们是能够获得所有组件的先验故障概率(通过机器学习也可以获得)[11]，并且假设所有组件故障是相互独立的. 例如组件N_1, N_2, N_3, N_4的先验故障概率分别是 0.01, 0.01, 0.02, 0.02，我们通贝叶斯定理得到所有组件的后验故障概率. 从表 1 中可以看出，$\{4\}$的后验故障概率最大，最有可能成为故障点.

Table 1 Posterior Failure Probability of Component

表 1 组件的后验故障概率

诊断	先验故障	组件	后验故障
{1, 2}	0.000 1	1	0.000 3
{2, 3}	0.000 2	2	0.000 3
{1, 3}	0.000 2	3	0.000 4
{4}	0.020 0	4	0.020 0

2.2 HSD 算法回顾

在模型诊断中，如果能够计算出所有的极小冲突集，那么计算所有极小冲突集的极小碰集就是诊断. 不

幸的是, 计算大规模电路的所有极小冲突集是耗时严重的[22], 并且计算所有冲突集的所有极小碰集也是不现实的[11,31]. 为了能够在有限时间内返回尽可能多的解, HSD 算法每计算出一个冲突集, 就计算已有冲突集的极小碰集, 然后再验证极小碰集是否是诊断. HSD 算法是目前求解故障诊断最好的算法之一, 我们给出 HSD 算法的流程图并用例 2 进行说明.

例 2. HSD 算法计算图 2(c)中的诊断.

把系统描述 *SD* 和 *Obs* 编码成 SAT 求解器可用的 *CNF* 数据格式后, 假设 HSD 算法求出的第 1 个极小冲突集 C_1={1, 2, 4}, 那么{1}是 C_1 的一个极小碰集, 也是该系统的一个候选诊断. 利用 SAT 求解器判断子句集 *CNF*\{1}(表示 *CNF* 去掉含有组件 1 后的子句)是否可满足, 因不可满足, 故{1}不是诊断. 继续计算极小冲突集 C_2={2, 3, 4}, 并计算{C_1, C_2}的一个极小碰集是{2}, 因为 *CNF*\{1, 2}是不可满足的, 所以继续计算极小不可满足子集 C_3={1, 3, 4}. 计算{C_1, C_2, C_3}的一个极小碰集是{4}, 因 *CNF*\{1, 2, 4}可满足, 故{4}是一个诊断解. 继续计算{C_1, C_2, C_3}的极小碰集{1, 2}, 因 *CNF*\{1, 2}是可满足的, 故{1, 2}是一个诊断. 继续计算{C_1, C_2, C_3}的极小碰集{2, 3}, 因 *CNF*\{2, 3}是可满足的, 故{2, 3}是一个诊断. 当{C_1, C_2, C_3}已经没有新的碰集生成时算法截止.

值得注意的是, 若先获得 C_1 的一个极小碰集是{4}, 那么{4}也是{C_1, C_2, C_3}的极小碰集, 可见{4}是一个隐式碰集. HSD 虽然有可能快速得到一个诊断, 但是频繁的调用碰集算法且验证得到的碰集是否是诊断, 这都是非常耗时的. 图 3 是 HSD 算法的流程图.

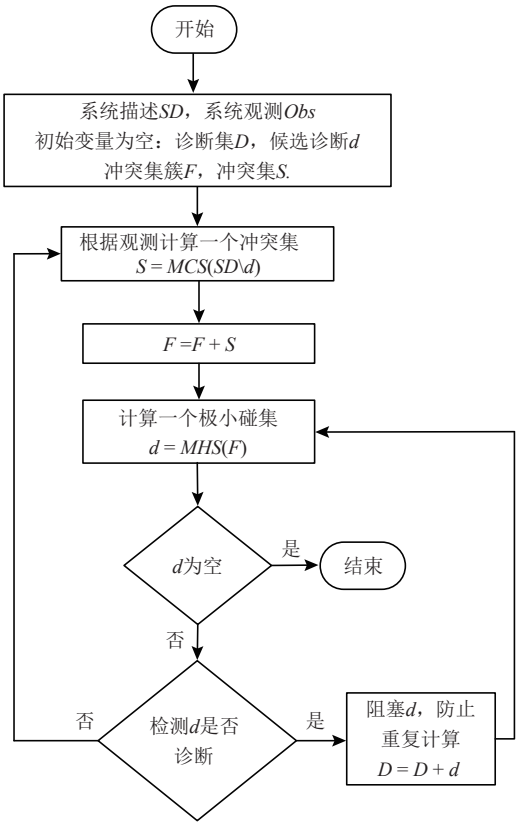


Fig. 3 Flow chart of HSD algorithm

图 3 HSD 算法流程图

3 判断隐式碰集求解诊断

在本节介绍新提出的冲突集选取策略和数学编码方式.

3.1 冲突集的选取

在诊断识别中, 确定最终的故障组件是由所有诊断的后验故障概率获得, 换句话说, 求得的诊断越多, 确定真实故障组件的概率就越大[3,11]. 现有的诊断算法在求解诊断时, 虽然能够快速获得大量的极小冲突集, 但是有些冲突集对于获得精确的故障定位作用较小, 反而会影响诊断的整体运行时间[11]. 根据碰集与冲突集的对偶性, 一些冲突集的获取也是不重要的[31]. 在表 1 中, 显然势最小的{4}在获得最有可能的故障组件中起着关键性作用, 获得一些势比较大的诊断反而会影响算法的整体效率. 为了尽可能地获得故障组件并兼顾诊断算法的运行时间, 我们先求出大量甚至所有的 MCS, 并设计了一个目标函数来删除一些质量较低的 MCS, 最后进行一次碰集算法来获得候选诊断. 设计的目标函数主要从极小势碰集和碰集的数量上评价一个冲突集的质量, 因为现有 MCS 算法从系统编码中求出的 MCS 含有大量的冗余解, 导致求解效率非常缓慢, 甚至在规定时间内都很难求出一个诊断. 为了保证求出足够的候选解数量, 我们只对相对质量较高的 MCS 进行碰集求解. 下面我们给出设计的目标函数并通过实例进一步说明.

$$Q(C_1,C_2)=\begin{cases}\dfrac{|C|}{|C_1|}+\dfrac{|C_2|-|C_1|}{|C_2|}, & |C|\neq 0,\\ 0, & |C|=0,\end{cases} \tag{1}$$

$C=\{e|e\in C_1\wedge e\notin C_2\}$, $|C_1|\leqslant|C_2|$.

$Q(C_1,C_2)$函数(简称 Q 函数)用来计算冲突集 C_2 是否被选取, Q 的值越大被选取的可能性就越大. $|C|$ 表示 C_1 中的元素不在 C_2 中元素的个数, 若$|C|$=0, 则说明 C_1 是 C_2 的子集, C_2 需要删除. 对于$|C|/C_1$, 若 C_1 中的元素不在 C_2 中的占比越高, 则 C_1 不能代表 C_2 的可能性就越大(去掉 C_2 则可能不是整体的碰集概率就越大), 我们选择 C_2 的可能性就越高. 对于($|C_2|$–

$|C_1|)/|C_2|$, 若 C_2 中含有不同于 C_1 中的元素越多, 那么生成的碰集个数就越多, 选择 C_2 的可能性也就越大.

Q 函数中存在一个阈值 k, k 是一个大于 0 的正数, 其根据相应的数据集来判断是否选择该冲突集. 在图 2(c)中, 每一个组件都对应 3 个集合, 我们对每一个集合进行排序标记, 即 $S_1, S_2, \cdots, S_{18}$. 假设通过系统观测及 MCS 算法, 求出 $C_1=\{S_1, S_2, S_5, S_8\}$, $C_2=\{S_1, S_6, S_7\}$, $C_3=\{S_2, S_8, S_9\}$. 虽然每一个 MCS 都是极小且互不相同, 但是映射成组件的冲突集时 $C'_1=\{1, 2, 3\}$, $C'_2=\{1, 2, 3\}$, $C'_3=\{1, 3\}$. 故利用 MCS 算法直接对系统编码计算可能会得到冗余的冲突集, 从而计算碰集时会增加额外的开销.

Q 函数能够很好地避免这种情况的发生. 当利用 Q 函数进行筛选时, $|C'_3|\leqslant|C'_1|\leqslant|C'_2|$, 优先选择 C'_3, 当再添加 C'_1 时, 因为 $Q(C'_1,C'_2)=0<k$, $Q(C'_1,C'_3)=0<k$, 所以删除 C'_1, C'_2.

例 3. 选取 c880 电路中映射成组件后的极小冲突集, $C_1=\{3, 4, 21, 23, 42, 48, 76\}$, $C_2=\{3, 4, 21, 23, 25, 29, 30, 37, 40, 41, 48, 51, 56, 73, 76, 77\}$, $C_3=\{3, 4, 21, 23, 25, 29, 30, 37, 40, 48, 51, 56, 60, 73, 76, 77\}$, $\varepsilon=0.1$. $Q(C_1, C_2)=1/7+9/16=0.705(>k)$, $Q(C_2,C_3)=1/16+ 0/16=0.063(<k)$. $CMHS(\{C_1,C_2\})=CMHS(\{C_1,C_2,C_3\})$, $|\{mhs|mhs\cap C_3\neq\varnothing, mhs\in MHS(\{C_1,C_2\})\}|/|MHS(\{C_1,C_2,C_3\})|=0.938$.

我们通过 Q 函数删除了 C_3, 通过 MHS 算法计算 $\{C_1, C_2\}$与$\{C_1, C_2, C_3\}$含有相同的极小势碰集, 且二者含有 93.8% 相同的极小碰集. 在诊断识别阶段, 我们最终目的是为了获得一个最有可能的故障点(后验故障排名), 而并非精确的概率值. 因此删除低质量的冲突集对诊断识别阶段影响较小(从表 1 也可以看出其主要取决于极小势诊断). 在大规模电路中(c7522, c6288), 直接对所有冲突集求解极小碰集是不现实的, 这是因为 MCS 的个数远远大于 10 万且大部分集合的势大于 500(获取 MHS 是一个 NP-hard 的问题, 这可能导致诊断的数量与组件的个数呈指数级关系), 所以约简它们的规模是非常有必要的. 算法 1 是利用目标函数 Q 选取 MCS 的算法.

算法 1. MCS 选取算法.

输入: 冲突集簇 $MCSs$, 阈值 k;

输出: 冲突集簇 $MCSs$.

① $MCSs\leftarrow Order(MCSs)$; /*升序排序*/

② for(i=2, $i\leqslant|MCSs|$, ++i)

③ for(j=1, $j<i$, ++j)

④ $c=\{e|e\in c_i\wedge e\notin c_j, c_i\in MCSs, c_j\in MCSs\}$;

⑤ if($(|c|/|c_j|+(|c_i|-|c_j|)/|c_i|)<k$)

⑥ $MCSs\backslash c_i$; /*删除 c_i*/

⑦ end if

⑧ end for

⑨ end for

算法 1 首先对所有 MCS 按照势进行一个排序(行①), 然后对每一个 MCS 进行筛选(行②③), 如果通过 Q 函数计算的值小于 k(行⑤), 则删除该 MCS (行⑥).

因为计算冲突集的极小碰集后需要再对其进行验证是否是诊断, SAT 求解器虽然能够高效地判断解集的可满足性, 但是频繁地调用求解器也是一个极耗时的事情. 下面我们给出一种适用于判断候选解是否满足所有观测的数学编码方式, 如式(2)所示, 并提出了一种优化策略大幅度缩减需要计算的组件的规模.

$$\boldsymbol{M}=\begin{pmatrix} \neg N_{\mathrm{BUFFER}} & \Leftrightarrow & o_1=i_1 \\ \neg N_{\mathrm{INVERTER}} & \Leftrightarrow & o_1=1-i_1 \\ \neg N_{\mathrm{AND}} & \Leftrightarrow & o_1=i_1i_2 \\ \neg N_{\mathrm{NAND}} & \Leftrightarrow & o_1=1-i_1i_2 \\ \neg N_{\mathrm{XOR}} & \Leftrightarrow & o_1=|i_1-i_2| \\ \neg N_{\mathrm{NOR}} & \Leftrightarrow & o_1=(1-i_1)(1-i_2) \\ \neg N_{\mathrm{OR}} & \Leftrightarrow & o_1=1-(1-i_1)(1-i_2) \end{pmatrix}. \tag{2}$$

式(2)利用了电路门的性质生成, $\neg N$表示组件正常, o 表示组件的输出, i 表示组件的输入(i, o 取值为 0 或 1). 根据电路的逻辑关系我们进行数学编码, 假设与非门有 2 个输入 $\{i_1, i_2\}$和 1 个输出 $\{o_1\}$, 根据与非门逻辑关系(输入有 0, 输出为 1), 则数学公式可以表示为 $o_1=1-i_1i_2$, (i_1, i_2, o_1 取值为 0 或 1). 同理, 其他电路门也可以用数学公式表示. 式(2)的 7 个公式包含了 ISCAS-85 基准数据中所有的电路类型, 而该数据集包含了现实生活中 1 300 多种故障类型, 在模型诊断中被众多学者认可并使用[2-4,11,22-26,30-34].

我们利用式(2)给出 c17 电路中 组件 N_4 的输出值 i_{15} 的数学编码方式: $i_{15}=1-i_{13}i_{17}=1-(1-i_8i_{12})(1-i_{11}i_{12})=i_8+i_{11}-i_8i_{11}-i_9i_{10}i_{11}-i_8i_9i_{10}+2\times i_8i_9i_{10}i_{11}-i_8i_9^2i_{10}^2i_{11}=i_8+i_{11}-i_8i_{11}-i_9i_{10}i_{11}-i_8i_9i_{10}+i_8i_9i_{10}i_{11}$.

因为 i 的值只能为 0 或 1, 所以在计算等式右侧的值时, 可以去掉指数, 如 $i_9^2=i_9$. 而对于其他的输出组件, 如 N_3, 不必要重新从输入端开始计算, 这是因为在计算 N_4 的值时, N_2 和 N_6 的输出值已经被获取, 所以只需要从它们开始计算即可.

3.2 代数表达式

在 c17 电路中, 因为只有系统观测 $\{i_7, i_8, i_9, i_{10}, i_{11}, i_{14}, i_{15}\}$是已知的, 其余变量及组件是否正常工作是未

知的, 所以要想判断某些组件故障后是否使输入和输出相一致, 则需要重新计算输出组件的输出值. 而计算输出组件的输出值需要先计算其他组件的输出值. 下面通过算法 2 给出每个组件计算的顺序.

算法 2. 诊断的判断算法.

输入: *Obs*, *Comps*, *SD*, *DiagSet*;

输出: *DiagSet*.

① *Order*=∅;

② *Sys_in*←*Obs*; /**Sys_in* 存储已计算变量*/

③ while(*Comps*) /**Comps* 是否为空*/

④ for(*c*∈*Comps*)

⑤ if(*c.input*⊂(*Sys_in*∪*Order*))

⑥ *Order*←*Order*∪*c*;

⑦ *Comps**c*; /*删除 *c**/

⑧ end if

⑨ end for

⑩ end while

⑪ for(*Diag*∈*DiagSet*)

⑫ for(c_i ∈ *Order*)

⑬ n_{c_i}←式(2)(*Diag*); /*计算c_i的输出值n_{c_i}*/

⑭ end for

⑮ if(输出组件的值和系统观测的值不相同)

⑯ *DiagSet**Diag*; /*删除该候选解*/

⑰ end if

⑱ end for

算法 2 先对组件进行排序然后判断候选解是否是诊断. 初始时 *Order* = ∅, *Sys_in* 存储系统的所有输入(行⑫). 若未排序的组件集为空, 则算法结束(行③), 否则循环判断剩余未排序的组件; 若该组件的所有输入都已经选取, 则该组件存入 *Order* 中, 并把该组件从未排序的组件中删除(行⑤~⑩).

例如 c17 电路, 初始时 *Sys_in* 存储{i_7, i_8, i_9, i_{10}, i_{11}}, 因只有组件 N_1 和 N_5 的输入值{i_7, i_9}和{i_9, i_{10}}属于 *Sys_in*, 故 *Order*={N_1, N_5}; 因 N_2 和 N_6 的输入值{i_8, N_1}, {i_{11}, N_1}属于{*Sys_in*, *Order*}, 故 *Order*={N_1, N_5, N_2, N_6}; 因 N_3 和 N_4 的输入值{N_2, N_5}, {N_2, N_6}都属于{*Sys_in*, *Order*}, 故 *Order*={N_1, N_5, N_2, N_6, N_3, N_4}.

算法 2 的行⑪~⑱是对候选解进行检验, 首先计算每一个候选解对应的组件输出值(行⑫~⑭), 然后判断观测值和真实值是否相同, 若不相同, 则从候选解中删除该解(行⑮~⑰). 该算法主要的缺点是对每一个候选解都需要重新计算所有的组件. 为了尽可能的减少计算组件的个数, 我们发现若通过系统的输入, 可以得到所有组件在正常情况下的理论值. 利用这一点, 若检验一个候选解是否是诊断, 我们只需要重新计算故障组件的影响部分.

定义 5. 后继组件. 组件 N_i 到达系统任意输出端所经过的组件, 称为 N_i 的后继组件.

在图 2(a)中, N_2 的所有后继组件为{N_3, N_4}, N_1 的所有后继组件为{N_2, N_6, N_3, N_4}. 我们可以看到, 当组件{N_2}故障时, 组件{N_1, N_5, N_6}不会受到影响. 此时, 判断候选解{N_2}是否是诊断时, 只需要重新计算它的后继组件{N_3, N_4}的输出值. 对于 N_1, N_5, N_6 的输出值只需要用理论值代替, 而 N_2 不在后继组件中, 故只需要翻转 N_2 的正常输出值. 通过诊断的性质可知, 当判断每个候选解是否满足所有观测时, 我们只需要计算候选解的所有后继组件, 对于不在后继组件的候选解组件, 只需要改变该组件的门, 即翻转该值 0 变为 1, 1 变为 0. 翻转策略如式(3)所示:

$$\boldsymbol{M}=\begin{pmatrix} N_{\mathrm{BUFFER}} & \Leftrightarrow & o_1=1-i_1 \\ N_{\mathrm{INVERTER}} & \Leftrightarrow & o_1=i_1 \\ N_{\mathrm{AND}} & \Leftrightarrow & o_1=1-i_1i_2 \\ N_{\mathrm{NAND}} & \Leftrightarrow & o_1=i_1i_2 \\ N_{\mathrm{XOR}} & \Leftrightarrow & o_1=1-|i_1-i_2| \\ N_{\mathrm{NOR}} & \Leftrightarrow & o_1=1-(1-i_1)(1-i_2) \\ N_{\mathrm{OR}} & \Leftrightarrow & o_1=(1-i_1)(1-i_2) \end{pmatrix}. \quad (3)$$

例 4. 继续例 2, 利用算法 2 判断候选解{N_1, N_6}是否是诊断解.

由系统的输入, 可以得到{i_{16}, i_{12}, i_{13}, i_{17}, i_{14}, i_{15}}的理论值分别是{1, 1, 1, 0, 0, 1}. N_1 的后继组件是{N_2, N_6, N_3, N_4}, N_6 的后继组件是{N_4}, 二者的关键组件取交集是{N_2, N_6, N_3, N_4}. 因诊断解 N_2 不在后继组件中, 故只需要利用式(3)反转 N_1 的理论值. N_6 故障, N_2, N_3, N_4 正常, 带入式(2)(3), 可以得到在组件 N_1, N_6 故障的情况下, 得到的理论输出值{0, 1}与系统观测的输出值一致, 可见, 候选解{N_1, N_6}是一个诊断解.

通过计算故障组件的后继组件, 避免了一些不相关组件的重复计算. 显然, 当故障组件越靠近输出端, 重新计算的组件越少. 如: 判断候选解{N_4}是否是一个诊断, 则只需要翻转 N_4 的值即可. 不幸的是, 若判断候选解越靠近输出端, 则后续的组件可能占总体组件的绝大部分. 下面我们通过命题 1, 进一步缩减计算组件的个数.

命题 1. 若故障组件的输出不影响后继组件的输出, 则无需计算该故障组件的输出值.

考虑到图 2(a)的候选解{N_1, N_6}, 只有候选解中的组件故障, 其余组件都是正常的. 因为 i_8 的输入为 0, 则不论 i_{12} 取何值时, i_{13} 的取值都为 1. 可见 N_1 故障并不影响 i_{13} 的取值, 故在求解 N_1 的后续组件时, 不

需要添加 N_2 的后继组件. 而 N_6 的输出值需要 i_{11} 和 i_{12} 共同决定, 故 N_1 的后续组件需要添加 N_6 及其后继组件. 最终 $\{N_1, N_6\}$ 的后继组件有 $\{N_6, N_4\}$. 计算后继组件同例 4, 这里不再赘述,

我们利用式(3)结合命题 1 给出组件 N_4 的输出值 i_{15} 的数学编码方式: $i_{15}=1-i_{13}i_{17}=1-(1-i_8i_{12})(i_{11}i_{12})=1-i_{11}i_{12}+i_8i_{11}i_{12}$.

可见通过命题 1, 我们少计算了一半的后继组件. 算法 3 是添加命题 1 的候选解判定算法.

算法 3. 优化的诊断判断算法.

输入: *Obs*, *Comps*, *SD*, *DiagSet*;

输出: *DiagSet*.

① *Rear*[]=∅;
② for(N_i∈*Comps*)
③ if(N_i.*output*∈N_j.*input*, N_j∈*Comps*)
④ *Rear*[N_i]←N_j; /*N_i 的后续组件 N_j*/
⑤ end if
⑥ end for
⑦ for(N_i←*Order.pop*) /*组件排序*/
⑧ for(N_j∈*Rear*[N_i])
⑨ *Rear*[N_i]←*Rear*[N_j];
⑩ end for
⑪ end for /*递归统计每个组件的后继*/
⑫ for(c_i∈*Comps*)
⑬ n_{c_i}←式(2); /*计算c_i的输出值n_{c_i}*/
⑭ end for /*计算所有组件的理论值*/
⑮ if(理论值与系统观测值一致)
⑯ return;
⑰ end if
⑱ for(*Diag*∈*DiagSet*)
⑲ for(c_i∈*Diag*)
⑳ if(后继组件受 c_i 影响) /*命题 1*/
㉑ *Wait*←*Rear*[c_i];
㉒ end if
㉓ end for
㉔ for(c_i∈*Diag*∧c_i∉*Wait*)
㉕ n_{c_i}←式(3); /*计算c_i的输出值n_{c_i}*/
㉖ end for /*翻转不在后继组件中的组件*/
㉗ for(c_i∈*Wait*)
㉘ n_{c_i}←式(2)(3); /*计算c_i的输出值n_{c_i}*/ /*后继组件*/
㉙ end for
㉚ if(输出组件和系统观测的值不相同)
㉛ *DiagSet*/*Diag*;
㉜ end if
㉝ end for

算法 3 初始化所有组件的后继组件为空(行①); 接着遍历每个组件输入, 如果组件 N_i 的输出属于 N_j 的输入, 则组件 N_j 是 N_i 的后继组件(行②~⑥); 递归地添加每个组件的所有后继组件(行⑦~⑪). 然后利用系统输入及式(2)计算所有组件的理论值(行⑫~⑭). 如果理论值和系统观测值一致, 则不存在故障组件, 直接返回(行⑮~⑰). 依次判断所有的候选解, 通过命题 1 计算所有需要计算的组件(行⑲~㉓). 若诊断解中的组件不在后继的组件中, 则直接翻转该组件的理论值(行㉔㉕). 利用式(2)(3)计算所有后继组件的值(行㉗~㉙), 若存在输出的组件值与系统观测值不相同, 则从解集中删除该候选解(行㉚~㉜).

因为我们最终目标是计算所有的极小诊断, 而计算部分冲突集的极小碰集后需要验证其是否是诊断, 也就是说, 需要验证部分冲突集的极小碰集是否是该系统的隐式碰集, 所以我们对算法 1~3 进行合并, 命名为 JIHS(judgment implicit hitting set)算法. 下面我们给出具体的诊断算法.

算法 4. JIHS 算法求解诊断.

输入: *Obs*, *Comps*, *SD*, *ε*, *MCSs*;

输出: *DiagSet*.

① *DiagSet*=∅, *MHSs*=∅;
② *MCSs*←算法 1(*MCSs*); /*选取 MCS*/
③ *MHSs*←*MHS*(*MCSs*); /*计算极小碰集*/
④ *DiagSet*←算法 3(*MHSs*). /*检验是否是诊断*/

算法的中间变量: *ε*, *MHSs*, *DiagSet* 分别代表阈值、极小碰集的数量、诊断的数量. 算法 4 主要先对冲突集进行筛选, 然后计算冲突集的极小碰集(可以直接调用碰集算法[22]). 注意的是, 这里每求解一个极小碰集就可以直接检验该解是否是诊断, 并非把所有的碰集都计算完毕再验证. 算法 4 中调用碰集算法和对比算法一致, 因算法 1 只是对冲突集进行选取, 运行时间相对于验证候选解忽略不计, 下面我们分析检验碰集是否是诊断的时间复杂度.

假设候选解的个数为 n, 组件个数为 m. 选取每个组件的后继组件时间复杂度为 $O(m^2)$; 计算所有组件的时间复杂度的理论值是 $O(m)$; 在验证一个候选解是否是诊断时, 最坏的情况下, 需要重新计算所有的组件, 时间复杂度是 $O(m)$, 故总体时间复杂度是 $O(mn)$. 因为在重新计算所有组件时, 由于命题 1 的存在, 计算组件的个数要远远小于 m, 最好的情况下, 只翻转 1 个组件的值, 时间复杂度是 $O(n)$.

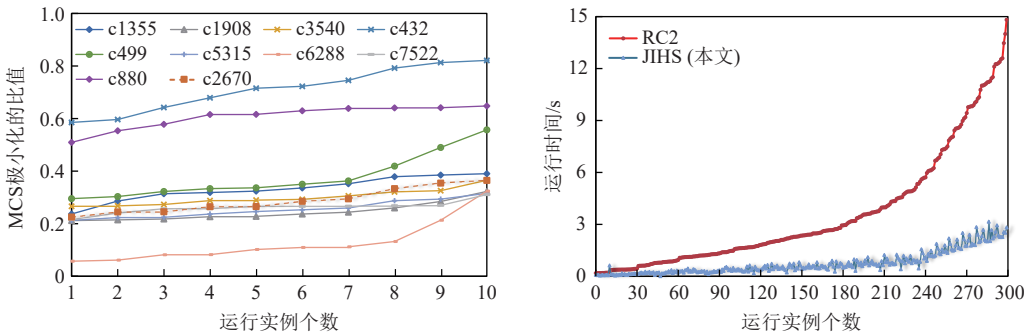
4 实验分析

在本节中我们对新提出的 JIHS 算法进行实验分析, 对比算法采用目前效率较高的 CMMO, HSD, One-SAT 算法, 所有算法都统一使用 RC2 求解器, 这是求解诊断效率最高的求解器之一[33-34]. 实验平台: Ubuntu 16.04 Linux Intel Xeon E5-1607@3.00 GHz 16 GB RAM C++, 实验数据采用国际标准的 ISCAS-85 电路数据, 该数据根据真实的环境系统生成, 由 DXC'09 的研究人员注入故障, 主要用来比较模型诊断算法求解效率.

本文主要从冲突集的选取和诊断验证 2 个方面进行综合考虑, 为了公平性, 在计算极小碰集时, 统一使用目前较快 BWIICC 极小碰集算法[22]. JIHS 算法与对比算法都是一种不完备的算法, 计算规模不同的问题所用的比较的标准不同, 具体看实验分析.

在模型诊断中, 计算诊断的时候需要先求出极小冲突集, 再计算极小冲突集的极小碰集. 在计算 MHS 时, 对 MCS 进行一个极小化后的预处理是一个有效的策略. 图 4 主要从 MCS 极小化和诊断验证 2 个角度进行分析, 横坐标都表示运行实例的个数, 图 4(a)纵坐标表示 MCS 极小化的比值, 图 4(b)纵坐标表示 RC2 求解器和 JIHS 算法判断候选解是否是诊断所花费的时间. 除了规模较小的 c17 电路, 我们对除 ISCAS-85 的其他数据集进行了测试, 每个数据集随机选择 10 组带故障的观测, 对 MCS 约简比进行升序排序. 其中 c432 电路最高约简 82.3%, 平均约简 71.2%, 所有电路平均约简 35.1%. 在候选解判断的过程中, 我们对 300 个实例进行测试, 仅有少数实例使用的求解器比 JIHS 好, 除此之处 JIHS 算法的候选判断要快, 最高达 5 倍以上.

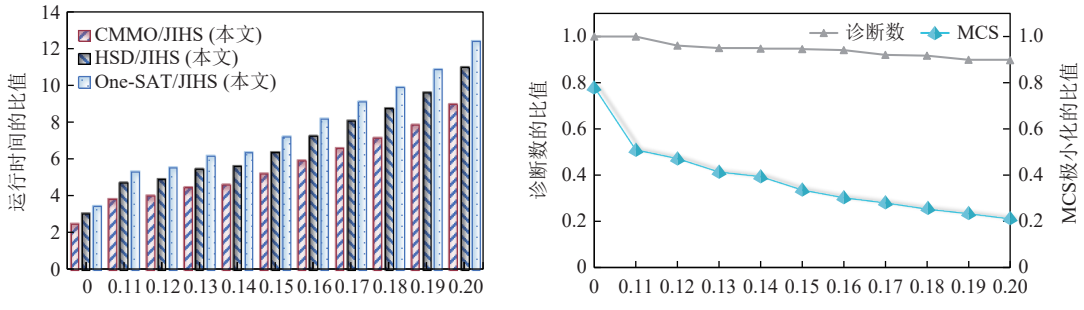
在图 5 中, 我们随机注入了 5 个故障组件, 可以计算出所有的 MCS 极小化的比值, CMMO, HSD, One-SAT 算法的运行时间分别是 89.5 s, 109.7 s, 124.1 s. 图 5 中横坐标表示实例数 k 的取值范围, 图 5(a)纵坐标表示 3 种算法运行时间与 JIHS 运行时间的比值, 图 5(b)纵坐标表示 JIHS 算法相对于对比算法求解冲突集和诊断数量的比值. k=0 时表示只对 MCS 进行一个极小化操作, MCS 的数量减少了 22%, JIHS 算法求解诊断的数量与对比算法的数量一致, 运行时间比对比算法提升 1 倍以上. 在对 MCS 进行极小化后, 随着 k 值的不断增加删除冲突集的数量也在增加,



(a) MCS 极小化 (b) 诊断验证

Fig. 4 Minimization ratio and diagnosis verification runtime of MCS

图 4 MCS 极小化比值和诊断验证运行时间



(a) 算法运行时间比较 (b) 诊断数的比值与MCS极小化的比值

Fig. 5 Diagnosis instances of ten fault components

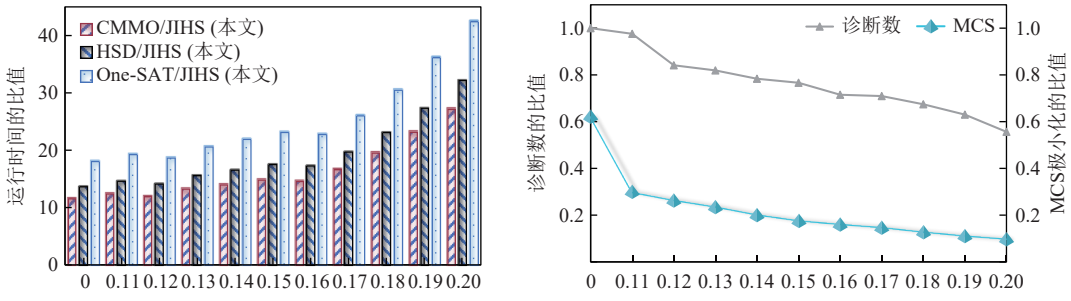
图 5 10 个故障组件诊断实例

从而导致隐式碰集(诊断)的减少. 在 k=0.2 时, JIHS 算法比对比算法快接近 1 个数量级, 而求解诊断数量仍在 85% 以上.

在图 5(a)中, 当 k=0, JIHS 算法求解效率比其他算法至少快 1 倍, 而在图 5(b)中要有 1 个数量级的提升. 在故障组件较少的数据中, 因为在计算出所有 MCS 后, 每一个 MHS 都是可满足的, 所以 SAT 求解器判断 MHS 的时间具有较高的效率.

在图 6 中, 我们随机注入了 15 个故障组件, 由于生成的冲突集数量较多, 规定所有算法计算 10 万解集停止, CMMO, HSD, One-SAT 算法运行时间分别为 995.2 s, 1 178.6 s, 1 554.5 s. 当系统规模较大且故障组件较多时, 由于生成冲突集的数量较多, 导致碰集的求解时间占据总体求解的大部分时间. 而对比算法频繁地调用碰集算法, 求解效率较低, 当 k=0 时, JIHS 算法比其他算法快 1 个数量级. 随着 k 值的增加, JIHS 算法获得诊断的数量也随着下降, 在 k=0.2 时, MCS 的数量是对比算法的 0.1 倍, 诊断数量是对比算法的 0.55 倍, 求解效率提高约 30 倍.

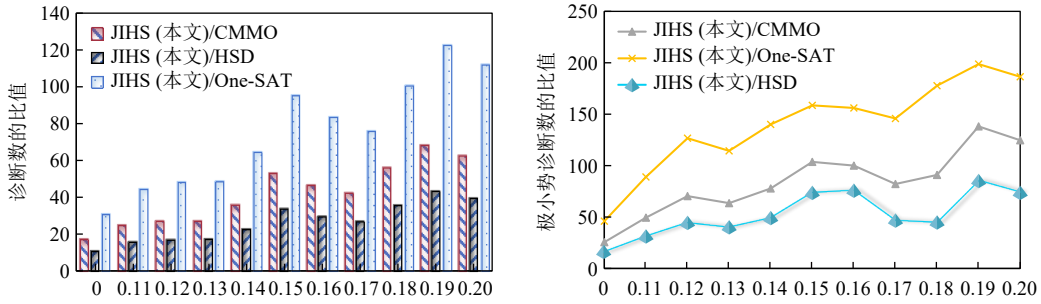
在图 7 中, 我们随机注入了 40 个故障组件, 所有算法的运行时间是 3 600 s. 图 7(a)纵坐标表示求出诊断数的比值, 图 7(b)纵坐标表示求出极小势诊断数的比值. 由于在有限的时间内无法求出所有的冲突集, 使大量的碰集都不是诊断, 对比算法在判断每个碰集是否是诊断时也都耗费了大量的时间. 而我们的检验算法对于非诊断解判断具有一定的优势, 只需要找到系统观测的一个不同组件即可. 而又因为命题 1, 只需要找到候选解影响的组件, 极大减少了组件的不必要计算. 在图 7(a), JIHS 算法比其他算法要多求出 1~2 个数量级的诊断数. 随着 k 值的增大, 求出的诊断数并不是越来越少, 这是因为在规定时间内无法计算出所有的极小碰集, 导致一些诊断数量的丢失(相当于陷入了无解空间), 而随着低质量的冲突集不断减少, JIHS 算法能够获得高质量冲突集更多的极小碰集. 在图 7(b), 随着 k 值的增加, 由于删除了较多相似的冲突集, 留给大量的时间计算极小碰集, 而这些极小碰集都倾向于势较小的解, JIHS 算法求出的极小势诊断数比其他对比算法要多 10~200 倍, 这也大幅度增加我们获得最有可能的故障位置的精准度.



(a) 算法运行时间比较　(b) 诊断数的比值与MCS极小化的比值

Fig. 6　Diagnosis instances of fifteen fault components

图 6　15 个故障组件诊断实例



(a) 算法运行时间比较　(b) 诊断与冲突集数量

Fig. 7　Diagnosis instances of forty fault components

图 7　40 个故障组件诊断实例

5 总 结

基于模型诊断是著名的系统调试方法,它对日益复杂的软件、硬件和网络物理系统的调试尤为重要.在求解多故障诊断问题时,由于庞大的冲突集数量,导致许多诊断算法很难在有限的时间给出一个具体的故障组件解释.本文提出了一个新型高效的诊断算法 JIHS,该算法主要对候选解的获取和检验 2 个方面进行优化.在诊断识别阶段,我们首先设计了一个目标函数,主要倾向于获得势最小的诊断,因为极小势诊断对确定最有可能的故障组件起到至关重要的作用.由于计算出的候选解大部分都不是诊断,基于此,我们根据电路组件的逻辑关系进行编码,通过重新计算关键组件来判断候选解是否是诊断,极大提高了检验效率.最后,我们通过实验,从诊断的效率和数量 2 个方面验证本文所提算法的有效性.

作者贡献声明:欧阳继红给出指导意见并修改论文;黄森提出了算法思路并撰写论文.

参 考 文 献

[1] Reiter R. A theory of diagnosis from first principles[J]. Artificial Intelligence, 1987, 32(1): 57−95

[2] Zhao Xiangfu, Tong Xiangrong, Ouyang Dantong, et al. Treemerge: Efficient generation of minimal hitting-sets for conflict sets in tree structure for model-based fault diagnosis[J]. IEEE Transactions on Reliability, 2021, 70(4): 1596−1610

[3] 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(1): 377−411

[4] Marques-Silva J, Janota M, Ignatiev A, et al. Efficient model based diagnosis with maximum satisfiability[C]//Proc of the 24th Int Joint Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2015: 1966−1972

[5] Dekleer J, Williams B C. Diagnosing multiple faults[J]. Artificial Intelligence, 1987, 32(1): 97−130

[6] Feldman A , De Castro H V, Van Gemund A, et al. Model-based diagnostic decision-support system for satellites[C/OL]//Proc of the 33rd Aerospace Conf. Piscataway, NJ: IEEE, 2013[2024-04-03]. https://ieeexplore.ieee.org/ document/6497427

[7] Struss P, Price C. Model-based systems in the automotive industry[J]. AI Magazine, 2004, 24(4): 17−34

[8] Jannach D, Schmitz T. Model-based diagnosis of spreadsheet programs: A constraint-based debugging approach[J]. Automated Software Engineering, 2016, 23(1): 105−144

[9] Palma J, Juarez J, Campos M, et al. Fuzzy theory approach for temporal model-based diagnosis: An application to medical domains[J]. Artificial Intelligence in Medicine, 2006, 38(2): 197−218

[10] Mordoch A, Juba B, Stern R. Learning safe numeric action models[C]//Proc of the 37th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2023, 12079−12086

[11] Stern R, Kalech M, Rogov S, et al. How many diagnoses do we need[J]. Artificial Intelligence, 2017, 248: 26−45

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

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

[14] 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)

[15] Torlak E, Chang F S, Jackson D. Finding minimal unsatisfiable cores of declarative specifications[C]//Proc of the 15th Int Symp on Formal Methods. Berlin: Springer, 2008: 326−341

[16] Mencia C, Previti A, Marques-Silva J. Literal based MCS extraction[C]//Proc of the 24th Int Joint Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2015: 1973−1979

[17] Jannach D, Schmitz T, Shchekotykhin K. Parallelized hitting set computation for model-based diagnosis[C]//Proc of the 29th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2015: 1503−1510

[18] Greiner R, Smith B. Wilkerson R. A correction to the algorithm in Reiter's theory of diagnosis[J]. Artificial Intelligence, 1989, 41(1): 79−88

[19] Wotawa F. A variant of Reiter's hitting-set algorithm[J]. Information Processing Letters, 2001, 79(1): 45−51

[20] Jiang Yunfei, Lin Li. The computation of hitting sets with Boolean formulas[J]. Chinese Journal of Computers, 2003, 26(8): 919−924 (in Chinese)
(姜云飞, 林笠. 用布尔代数方法计算最小碰集[J]. 计算机学报, 2003, 26(8): 919−924)

[21] Pill I, Quaritsch T. Optimizations for the Boolean approach to computing minimal hitting sets[C]//Proc of the 20th European Conf on Artificial Intelligence. Montpellier: IOS, 2012, 648−653

[22] Liu Siguang, Ouyang Dantong, Zhang Liming. Minimization of candidate solutions in minimal hitting sets solution[J]. Journal of Software, 2018, 29(12): 3733−3746 (in Chinese)
(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[J]. 软件学报, 2018, 29(12): 3733−3746)

[23] 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)

[24] Zhao Xiangfu, Huang Sen, Tong Xiangrong, et al. IBWIICC: Incrementally computing minimal hitting-sets combing local independence cover checking[J]. Acta Electronica Sinica, 2022, 50(11): 2722−2729 (in Chinese)
(赵相福, 黄森, 童向荣, 等. IBWIICC: 结合局部独立覆盖检测策略增量求解极小碰集的算法[J]. 电子学报, 2022, 50(11): 2722−2729)

[25] Zhao Xiangfu, Ouyang Dantong. Deriving all minimal hitting sets based on join relation[J]. Transactions on Systems, Man, and Cybernetics: Systems, 2015, 45(7): 1063−1076

[26] Zhao Xiangfu. Linearmerge: Efficient computation of minimal hitting sets for conflict sets in a linear structure[J]. Engineering Applications of Artificial Intelligence, 2018, 72: 327−339

[27] Wang Qiujie, Tao Jin, Wang Mengqi. A hierarchical minimum hitting set calculation method for multiple multiphase faults in power distribution networks[J]. IEEE Transactions on Industrial Electronics, 2021, 68(1): 4−14

[28] Wang Qiujie, Tao Ji, Mohamed M. A fast and robust fault section location method for power distribution systems considering multisource information[J]. IEEE Systems Journal, 2022, 16(2): 1954−1964

[29] Kalech M, Stern R, Lazebnik E. Minimal cardinality diagnosis in problems with multiple observations[J]. Diagnostics, 2021, 11(5): 780

[30] Siddiqi S. Computing minimum-cardinality diagnoses by model relaxation[C]//Proc of the 24th Int Joint Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2011: 1087−1092

[31] Ignatiev A, Morgado A, Weissenbacher G, et al. Model-based diagnosis with multiple observations[C]//Proc of the 28th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2019: 1108−1115

[32] 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

[33] Ignatiev A, Morgado A, Marques-Silva J. RC2: An efficient MaxSAT solver[J]. Satisfiability Boolean Modeling and Computation, 2019, 11(1): 53−64

[34] Kochemazov S, Kondratiev V, Gribanova I. Empirical analysis of the RC2 MaxSAT algorithm[C]//Proc of the 46th MIPRO ICT and Electronics Convention. Piscataway, NJ: IEEE, 2023: 1027−1032

[35] Gainer-Dewar A, Vera-Licona P. The minimal hitting set generation problem: Algorithms and computation[J]. Siam Journal on Discrete Mathematics, 2016, 31(1): 63−100

[36] Siddiqi S, Huang J. Hierarchical diagnosis of multiple faults[C]//Proc of the 20th Int Joint Conf on Artificial Intelligence. San Francisco, CA: Morgan Kaufmann, 2007: 581−586



Ouyang Jihong, born in 1964. PhD, professor. Member of CCF. Her main research interests include model-based diagnosis and machine learning.

欧阳继红, 1964 年生. 博士, 教授. CCF 会员. 主要研究方向为模型诊断、机器学习.



Huang Sen, born in 1995. PhD candidate. Student member of CCF. His main research interests include model-based diagnosis and SAT problem.

黄　森, 1995 年生. 博士研究生. CCF 学生会员. 主要研究方向为模型诊断、可满足性问题.