CIMHS:基于优化增量策略求解极小碰集的方法

魏 霞[1],赵相福[1],黄 森[2]
(1. 烟台大学计算机与控制工程学院,山东烟台 264005;2. 浙江师范大学计算机系,浙江金华 321000)

摘 要: 在基于模型的诊断推理过程中,极小碰集求解效率是决定诊断快慢的关键一步. 本文在原有极小冲突集合簇与极小碰集簇的基础上,充分考虑它们与新增冲突集中元素的关系,提出一种新型的优化增量策略. 在原有极小冲突集簇中,首先通过启发式策略抽取部分集合进行极小化,从而大幅度缩短求解时间;然后通过优化的增量策略快速补全并更新解集,进而提高整体求解效率. 为进一步提高算法的效率,提出按冲突集的势从小到大增量排序,利用新型优化的增量策略依次递归计算,并最终求得原始问题集合簇所有解集的全增量算法CIMHS(Complete Increment Minimal Hitting Set). 实验结果表明,在许多情形下,CIMHS算法较其他经典的极小碰集求解算法,可减少1至3个数量级的运行时间.

关键词: 基于模型诊断;极小冲突集;极小碰集;增量策略;启发式策略;全增量
基金项目: 国家自然科学基金(No.61972360,No.62072392)
中图分类号: TP306 **文献标识码:** A **文章编号:** 0372-2112(2023)05-1334-07
电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20220482

CIMHS: A Method of Computing all Minimal Hitting Sets for Minimal Conflict Sets Based on an Optimized Incremental Strategy

WEI Xia[1], ZHAO Xiang-fu[1], HUANG Sen[2]
(1. *School of Computer and Control Engineering*, *Yantai University*, *Yantai*, *Shandong* 264005, *China*;
2. *Department of Computer*, *Zhejiang Normal University*, *Jinhua*, *Zhejiang* 321000, *China*)

Abstract: In the process of model-based diagnosis, it is a key step to efficiently generate all minimal hitting sets. In this paper, a novel optimization strategy for incremental diagnosis is proposed. When a new conflict set is added, the relationship between the original minimal hitting sets and the newly added conflict set will be explored thoroughly. The time cost can be greatly reduced by joining the corresponding heuristic strategy to select the specific sets in the course of minimality checking. The computing performance can be improved considerably by introducing the optimized incremental strategy to update the final results. Furthermore, in order to obtain the higher efficiency, an algorithm named CIMHS (Complete Increment Minimal Hitting Set) based on optimized incremental strategy is presented. The proposed algorithm processes set incrementally in turn according to the ascending order of the cardinality of set. Experimental results including both on multiple synthetic and benchmark examples show that the proposed CIMHS algorithm is more efficient than many other state-of-the-art methods, with a reduction of several orders of magnitude runtime.

Key words: model-based diagnosis; minimal conflict set; minimal hitting set; incremental strategy; heuristic strategy; complete increment
Foundation Item(s): National Natural Science Foundation of China (No.61972360, No.62072392)

1 引言

基于模型的诊断(Model-Based Diagnosis,MBD)作为人工智能领域中基于"深知识"的智能化故障诊断推理技术,克服了传统故障诊断中基于专家经验系统存在的诸多局限性,从而在汽车故障诊断、医疗问题诊断以及配电网故障诊断等领域得到广泛的应用[1~3].

求解极小碰集(Minimal Hitting Set,MHS)是MBD过程中的关键步骤之一[4]. 不幸的是,极小碰集的计算已经被证明是一个NP-Hard问题[5]. 1987年智能诊断专家Reiter结合de Kleer提出的冲突集的概念,首次将极

收稿日期:2022-04-29;修回日期:2022-10-15;责任编辑:覃怀银

小碰集算法 HS-Tree (Hitting Set-Tree)[5] 运用到 MBD. 1989 年,Greiner 等提出 HS-DAG (Hitting Set-Directed Acyclic Graph)方法,利用有向无环图结构补全了 HS-Tree 算法的完备性[6]. 近年来,国内外众多学者对极小碰集求解方法进行了大量研究. 姜云飞等提出 BHS-Tree(Binary Hitting Set-Tree)方法,生成节点个数较少,并且可以增量求解[7]. 随后,林笠等提出基于布尔代数的 Boolean 方法,将问题集合簇编码,用布尔代数逻辑知识求解极小碰集,是目前效率最高的算法之一[8]. 但这两种方法在系统规模较大时,碰集极小化需花费大量的时间. Pill 等对 Boolean 方法进行了优化,使用迭代代替了递归,减少空间消耗[9]. 赵相福等提出基于连接关系的增量方法[10],求解过程中只需对同一等价类解集进行极小化操作,提高了求解效率. 此外,欧阳丹彤等提出近似完备算法[11],以牺牲解集完备性换取在可接受的时间内得到部分解集,大幅度提升求解效率.

当前碰集极小化方法,由于存在无法去除相同的冗余极小碰集问题,仍然采用子超集检测极小化算法(Sub-Superset Detecting Minimization,SSDM)[12]. 此方法每次调用时均需将新碰集与原碰集簇中碰集逐一进行比较,导致判定效率会随着问题规模的增大而迅速降低. 为减少子集检测的比较次数,缩小问题集合簇的规模,本文提出优化增量策略以便补全减少的集合. 在增量计算时,根据原冲突集合簇的每一解集与新增集合交集是否为空分成两类,对于交集非空的解集,无需极小化处理;而对于交集为空集的解集,需要与新增集合中元素进行笛卡尔积扩展. 在扩展前,先将新增集合中的元素分为独有与公有两类,一部分解集与独有元素进行笛卡尔积扩展后,无需去超集;另一部分与新增冲突集的公有元素进行笛卡尔积扩展后,通过启发式策略,仅选取部分解集进行极小化.

为进一步提高极小碰集求解效率,提出全增量 CIMHS(Complete Increment Minimal Hitting Set)方法. 将冲突集合簇中的冲突集按照集合的势从小到大依次排序,采用优化增量策略依次递归增量冲突集合,不断更新解集簇,从而求得所有极小碰集.

2　预备知识

2.1　基于模型的诊断相关定义和概念

定义 1　系统[5]. 一个基于模型诊断的系统定义为三元组$<S,C,O>$,其中:

(1) S(System)为系统描述,是一阶谓词公式的集合,描述了待诊断故障系统的相关信息;

(2) C(Component)为系统的组成部件集合,是一个有限的常量集;

(3) O(Observation)为观测集,是一阶谓词公式的有限集.

定义 2　冲突集[5]. 冲突集是系统$<S,C,O>$中C的一个子部件集$\{c_1,c_2,\cdots,c_n\}\subseteq C$,使得$S\cup O\cup\{\neg A(c_1),\neg A(c_2),\cdots,\neg A(c_n)\}$不一致. 其中,使用一阶谓词$A$表示"反常的(abnormal)",$A(c)$为真当且仅当部件$c$反常,$c\in C$. 一个冲突集被称为极小冲突集,当且仅当该冲突集的任意真子集都不是冲突集.

定义 3　碰集[5]. 设冲突集合簇$F=\{S_1,S_2,\cdots,S_n\}$,称集合H为F的一个碰集,如果H满足:(1)$H\in\cup_{S_i\in F}$;(2)对于任意一个$S_i\in F$,都有$H\cap S_i\neq\varnothing$. 一个碰集是极小碰集,当且仅当它的任意真子集都不是F的碰集.

例 1　设冲突集合簇$F=\{\{a,b\},\{b,c\},\{d,e,f\}\}$,则它的所有的极小碰集为:$\{a,c,d\}$,$\{a,c,e\}$,$\{a,c,f\}$,$\{b,d\}$,$\{b,e\}$,$\{b,f\}$. 以集合$\{b,f\}$为例,首先,该集合中所有元素均来自集合簇$F$;其次,集合$\{b,f\}$与$F$中任何集合的交集均非空;因而,集合$\{b,f\}$为$F$的碰集. 此外,若删除元素$b$,则集合$\{f\}$与集合$\{a,b\}$交集为空,因而不是碰集;若删除元素$f$,则集合$\{b\}$与集合$\{d,e,f\}$交集为空,因而不是碰集. 综上,集合$\{b,f\}$为$F$的极小碰集.

定义 4　集合的势. 称集合中系统部件的个数为集合的势,其大小可以用来度量集合的规模大小.

2.2　增量策略的相关定义和概念

定义 5　独有元素和公有元素. 假设向冲突集合簇F中新增的集合为S_s. 称元素e为S_s的独有元素,当且仅当$e\in S_s\wedge e\notin\cup_{S_i\in F}S_i$. 独有元素构成的集合$S_p$称为$S_s$的独有元素集. 称元素$e$为$S_s$的公有元素,当且仅当$e\in S_s\wedge e\cup_{S_i\in F}S_i$. 公有元素构成的集合$S_c$称为$S_s$的公有元素集.

定义 6　集合S与集合X的笛卡尔积$\otimes$. $S\otimes X=\{\{e\}\cup X|e\in S\}$,即,将集合$S$中的每一个元素分别添加至集合$X$.

定义 7　集合S与集合簇F的笛卡尔积$\otimes$. $S\otimes F=\{\{e\}\cup S_i|e\in S,S_i\subseteq F\}$,即,将集合$S$中的每一个元素$e$分别添加至集合簇中的每一个集合中,称为集合与集合簇(或集合簇与集合)的笛卡尔积.

定义 8　集合簇F_1与集合簇F_2的笛卡尔积$\otimes$. $F_1\otimes F_2=\{S_1\cup S_2|S_1\in F_1,S_2\in F_2\}$,即,将集合簇$F_1$中的每一个集合分别与集合簇$F_2$中的每个集合取并集运算产生的所有集合.

例 2　设冲突集合簇F的解集簇为$H=\{\{1,2,3\},\{3,4,5\},\{5,6,7\}\}$,当向冲突集合簇$F$中新增冲突集$S_s=\{1,7,8\}$时,此时$S_p=\{8\}$,$S_c=\{1,7\}$. 当集合$S_c$与解集簇$H$进行笛卡尔积时,$S_c\otimes H=\{\{1,2,3\},\{1,3,4,5\},\{1,5,6,7\},\{1,2,3,7\},\{3,4,5,7\},\{5,6,7\}\}$.

3 极小碰集求解算法

本节首先介绍基于子超集检测极小化算法和原始增量算法，然后给出优化的增量策略以及全增量方法的算法描述，并通过一个具体实例进行说明，最后对算法的时间复杂度进行了分析.

3.1 子超集检测极小化算法

基于子集检测去超集算法主要是对碰集的极小性进行判定. 由定义3可知，若碰集H是集合簇F的一个极小碰集，则H的任意真子集均不在F的解集簇中. 否则，解集簇中存在真超集. 具体过程如算法1所示.

算法1 子超集检测极小化算法

输入:碰集簇$H=\{S_1,S_2,\cdots,S_n\}$.

输出:极小碰集簇M.

1.$M\leftarrow M\cup\{S_1\}$;
2.For S_i in H do
3. For S_j in M do
4. IF($S_j\subseteq S_i$)
5. break;
6. IF($S_i\subseteq S_j$)
7. $M.d(S_j)$;
8. $M\leftarrow M\cup\{S_i\}$;
9. End For
10.End For

3.2 原始增量算法

原始增量算法在增量求解过程中，无需对新冲突集合簇进行重新求解，只需在原解集簇基础上进行增量更新. 将原冲突集合簇的解集根据与新增集合交集是否为空分为两类，对于交集非空的解集无需处理，而交集为空的解集，需要与新增集合中元素进行笛卡尔积扩展. 具体过程如算法2所示.

算法2 原始增量算法

输入:冲突集$S_s=\{e_1,e_2,\cdots,e_m\}$,解集簇$E=\{S_1,S_2,\cdots,S_n\}$.

输出:极小碰集簇M.

1.For S_i in E do
2. IF($S_s\cap S_i==\varnothing$
3. $M\leftarrow M\cup\{\{S_s\otimes S_i|\forall S_i\in M, S_i\not\subset S_s\otimes S_i\}\}$;
4. ELSE
5. $M\leftarrow M\cup\{S_i\}$;
6.End For

例3 设冲突集合簇$F=\{\{1,2,3\},\{3,4,5\},\{5,7\}\}$,新增冲突集$S_s=\{1,4\}$,求解所有的极小碰集.

Step1:由完备极小碰集算法，可以得到F的所有解集为:$\{3,5\}$,$\{3,7\}$,$\{1,4,7\}$,$\{1,5\}$,$\{2,4,7\}$,$\{2,5\}$,共6个集合.

Step2:根据原解集簇中集合与S_s是否有交集，分为$O_1=\{\{1,4,7\},\{1,5\},\{2,4,7\}\}$与$O_2=\{\{3,5\},\{3,7\},\{2,5\}\}$.

Step3:经$S_s\otimes O_2$，得到碰集簇$H=\{1,3,5\},\{1,3,7\},\{1,2,5\},\{3,4,5\},\{3,4,7\},\{2,4,5\}$.

Step4:将H与O_1归并后输入到算法1，得到部分新解集簇h:$\{\{1,3,7\},\{3,4,5\},\{3,4,7\},\{2,4,5\}\}$.

Step5:新冲突集簇的解集簇$M=O_1\cup h$，即:$\{1,4,7\},\{1,5\},\{2,4,7\},\{1,3,7\},\{3,4,5\},\{3,4,7\},\{2,4,5\}$,共7个解.

3.3 优化的增量策略

首先，根据新增集合中元素与解集元素的关系，找到新增集合的独有元素集S_p及公有元素集S_c;根据每一解集与新增集合交集是否为空，将解集簇分成M_1和M_2两部分.

显然，对于交集非空的M_1依然保持为极小解;其次，对于交集为空的M_2与S_p进行笛卡尔积后，得到的解集也为极小解;此外，M_2与$S_c=\{e_1,e_2,\cdots,e_m\}$进行笛卡尔积后，得到的解集仅需与$M_1$中部分解集进行比较去超集. 例如:$M_2$中的各集合$S$扩展$e_i$后，只需与$M_1$中含有元素$e_i$，且不含有$S_c$中其他元素的解集进行比较去超集后，得到的解集也为极小解.

以上三部分解集进行合并，即为最终解集.

命题1 M_2中的集合S扩展元素e后，若其不是超集，则为极小解，不可能为其他解集的真子集.

证明 反证法. 如果S扩展元素e后是其他解集的子集，说明S在扩展前已是其他解集的子集，显然不满足候选解都是极小解的前提. 若S扩展后成为超集，其子集必然不会含有e以外的其他共有元素，故只需与含有e的解集进行超集比较.

具体实现过程如算法3所示.

算法3 优化增量算法

输入:$S_s=\{e_1,e_2,\cdots,e_m\}$,解集簇$E=\{S_1,S_2,\cdots,S_n\}$.

输出:极小碰集簇M.

1.$M_e=\cup_{i=1}^{n}S_i\in E$;
2.$S_p=\{e|e\in S_s\wedge e\notin M_e\}$;
3.$S_c=\{e|e\in S_s\wedge e\in M_e\}$;
4.For S_i in E do
5. IF($S_s\cap S_i==\varnothing$
6. $M\leftarrow M\cup\{\{S_p\otimes S_i\}\}$;
7. $M\leftarrow M\cup\{\{S_c\otimes S_i|\forall S_i\in M, S_i\not\subset S_s\otimes S_i\}\}$;
8. ELSE
9. $M\leftarrow M\cup\{S_i\}$;
10.End For

例4 设冲突集合簇$F=\{\{1,2,3\},\{3,4,5\},\{5,6,7\}\}$,新增冲突集$S_s=\{1,7,8\}$,求解所有的极小碰集.

Step1:F的所有解集为:$\{1,5\}$,$\{1,4,6\}$,$\{1,4,7\}$,

{2,4,7},{3,7},{2,4,6},{2,5},{3,5},{3,6},共9个集合.

Step2:O_1={{1,5},{1,4,6},{1,4,7},{2,4,7},{3,7}},O_2={{2,4,6},{3,5},{2,5},{3,6}}. S_p={8},S_c={1,7}.

Step3:经$S_p \otimes O_2$,可得解集簇h_1={{2,4,6,8},{3,5,8},{2,5,8},{3,6,8}}.

Step4:引入集合簇h_2初始为空. 循环将S_c中的元素与O_2进行笛卡尔积运算,首先元素1进行笛卡尔积,得到碰集H={{1,2,4,6},{1,3,5},{1,2,5},{1,3,6}}. 此时,挑选O_1中仅含有元素1且不含有元素7的部分解集O_1'={{1,5},{1,4,6}},将H和O_1'归并后输入算法1,即可得到部分新解集簇h. 令$h_2=h_2 \cup h$. 循环结束时,转到Step5.

Step5:新冲突集合簇的极小碰集簇为:$M=h_1 \cup h_2 \cup O_1$,即M={{2,4,6,8},{3,5,8},{2,5,8},{3,6,8},{1,3,6},{2,5,7},{1,5},{1,4,6},{1,4,7},{2,4,7},{3,7}},共11个解.

3.4　全增量算法

受到前述优化的增量策略启发,本文进而提出一种全增量策略计算极小碰集. 在计算极小碰集时,首先将冲突集合簇中的冲突集按照集合的势从小到大排序,挑选势最小冲突集依次进行增量求解. 在每次增量过程中,采用算法3求解,直至冲突集合簇为空为止.

具体的实现过程如算法4所示.

算法4　全增量算法

输入:有序冲突集合簇$F=\{S_1,S_2,\cdots,S_n\}$.

输出:极小碰集簇M.

1.For S_i in F do
2.　$S_s=S_i$;
3.　IF($S_s==S_1$)
4.　　$M \leftarrow M \cup \{\{e_i \mid \forall e_i \in S_1\}\}$;
5.　ELSE
6.　　$E \leftarrow M$;
7.　　$M \leftarrow$算法3;
8.End For

例5　设冲突集合簇F={{1,2,3},{2,4},{1,6}},求解其所有的极小碰集.

Step1:将F中的集合按照势从小到大依次排序,即,F_0={{2,4},{1,6},{1,2,3}}.

Step2:首先设M为空,S_s={2,4},将M和S_s作为参数调用算法3,输出得到M={{2},{4}}.

Step3:接着S_s={1,6},M={{2},{4}},将M和S_s作为参数调用算法3,输出新M={{1,2},{1,4},{2,6},{4,6}}.

Step4:最后将S_s={1,2,3}与M={{1,2},{1,4},{2,6},{4,6}}作为参数调用算法3,输出最终M={{3,4,6},{1,2},{1,4},{2,6}},共4个解.

3.5　复杂度分析

在全增量算法中,为简化起见,设冲突集簇共有n个集合,每个集合有z个元素,每次增量时有$q/z(0 \leqslant q \leqslant z)$部分解集与增量集合交集非空. 在最坏情况下,每次笛卡尔积生成的解都是极小解. 增量算法的时间消耗由两部分组成,一部分是取交集时间,另一部分是去超集时间. 设增量第i个集合共产生a_i个解,则:$a_1=z$;$a_2=a_1*(q/z)+a_1*(z-q)$;$a_3=a_2*(q/z)+a_2*(z-q)$;$\cdots$;$a_n=a_{n-1}*(q/z)+a_{n-1}*(z-q)$.

增量n个集合共生成S_n个解集,则$S_n=\sum_{i=1}^{n} a_i=\frac{(q+z^2-qz)^n z^{2-n}-z^2}{q+z^2-qz-z} \approx z^{2-n}(q+z^2-qz)^{n-1}$. 取交集次数为$T_n=S_{n-1}=z^{3-n}(q+z^2-qz)^{n-2}$;去超集次数为$R_n=\left(\frac{z-q}{z}\right)\sum_{i=2}^{n} a_{i-1}^2=\left(\frac{z-q}{z}\right)\frac{(q+z^2-qz)^{2n-2}z^{6-2n}-z^4}{(q+z^2-qz)^2-z^2}$,即,$R_n \approx z^{5-2n}(q+z^2-qz)^{2n-4}(z-q)(n \geq 2, R_1=0)$. 因此,全增量算法的时间复杂度可以表示为$O(T_n+R_n) \approx O\left(z^{5-2n}(q+z^2-qz)^{2n-4}(z-q)\right)$. 而传统去超集算法的时间复杂度为$O\left(z^{4-2n}(q+z^2-qz)^{2n-2}\right)$,后者与前者的比值为$\frac{(q+z^2-qz)^2}{z(z-q)}=z^2(z-q)+\frac{q^2}{z(z-q)}+2q \geq 2q\left(\sqrt{z}+1\right)$. 可见,随着$z$、$q$的增大,全增量CIMHS算法更为高效.

需要注意的是:在传统算法去超集时,需要判断两个候选解的相互包含关系,并且候选解都为全局冲突集的碰集. 而全增量算法在去超集时,只需要判断新增解集是否为超集,且新增解集都为局部冲突集的碰集(候选解的元素比传统去超集算法要少). 除此之外,若新增集合存在独有元素,则会求出更多不需要去超集的解集,因而全增量算法的效率会更高.

4　实验分析

实验平台如下:Windows 10操作系统,CPU AMD Ryzen 7 5800H 3.20 GHz 4核16 GB RAM,C++. 实验的测试用例来源分为2组:人工数据冲突集和国际标准测试电路ISCAS-85(International Symposium on Circuits And Systems-85)数据冲突集. 每组用例测试10次取其平均值作为最终记录结果.

4.1 人工数据冲突集

假设每一组数据为$D_{k_c_m_i}$,其中k代表集合簇中集合的势,c代表相邻集合间相交元素的个数,m代表集合簇中集合的个数,i代表调节集合中元素的个数. 数据类型可以表示为如下形式:$\{1,\cdots,k\},\{k-(c-1),\cdots,k-(c-1)+k-1\},\{2[k-(c-1)]-1,\cdots,2[k-(c-1)]+k-2\},\{3[k-(c-1)]-2,\cdots,3[k-(c-1)]+k-3\},\cdots,\{(m-1)[k-(c-1)]-(m-2),\cdots,(m-1)[k-(c-1)]+k-(m-1)\},\{m[k-(c-1)]-(m-1),\cdots,m[k-(c-1)]-(m-1)+i-1\}$. 譬如$D_{6_2_4_3}$对应的冲突集簇为:$\{\{1,2,3,4,5,6\},\{5,6,7,8,9,10\},\{9,10,11,12,13,14\},\{13,14,15,16,17,18\},\{17,18,19\}\}$.

下面,从冲突集合簇中冲突部件集的个数、长度以及集合间相交元素的个数,三个不同角度对算法性能进行验证与分析. 设置实验组:全增量算法*vs*经典极小碰集求解算法. 选取HS-Tree、BHS-Tree、Boolean三种经典算法作为实验对比对象.

(1) 冲突集簇中冲突部件集个数不同时.

从表1和图1可以看出,随着冲突集簇中冲突部件集个数m增加,各算法在求解所有极小碰集的计算耗时均在增加. 但全增量算法相较于其他三种算法增幅最小,且在每组测试用例下用时均最少. 随着极小碰集解集个数增多,全增量算法求解效率较其他三种算法具有比较明显的提升效果,尤其当极小碰集解集个数超过100 000时,全增量算法比其他三种算法耗时降低两个数量级.

表1 集合个数不同时的运行时间表 单位:s

$D_{7\_1\_m\_i}$	$D_{7\_1\_5\_0}$	$D_{7\_1\_5\_2}$	$D_{7\_1\_5\_4}$	$D_{7\_1\_5\_6}$	$D_{7\_1\_5\_8}$	$D_{7\_1\_6\_0}$	$D_{7\_1\_6\_2}$	$D_{7\_1\_6\_4}$	$D_{7\_1\_6\_6}$
Boolean	0.284 9	0.314 8	2.527 4	6.698 6	8.827 3	9.617 3	10.619 8	87.388 9	337.496 6
CIMHS	0.004 6	0.010 0	0.035 1	0.069 9	0.037 9	0.035 5	0.264 3	1.262 4	2.674 5
HS-Tree	0.492 6	0.839 0	4.201 0	10.767 6	19.972 9	14.703 0	27.312 8	164.401 6	570.815 0
BHS-Tree	0.429 7	0.460 5	3.480 3	9.330 3	18.041 6	13.306 6	14.147 9	104.375 4	434.923 8
极小碰集个数	5 275	5 437	14 293	23 149	32 005	27 577	28 424	74 722	121 020

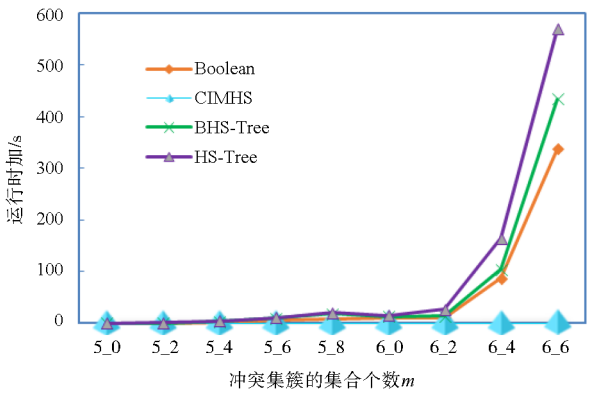

图1 $D_{7_1_m_i}$变化下的求解时间结果图

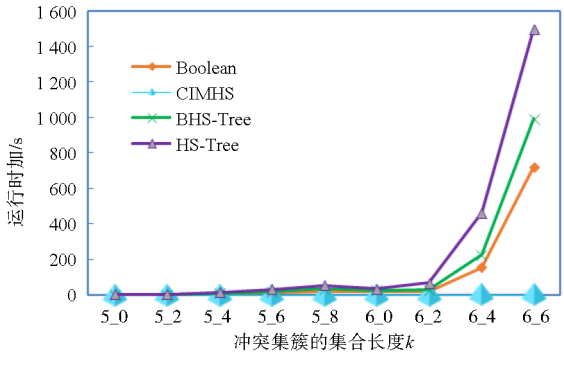

图2 $D_{k_1_7_i}$变化下的求解时间结果图

(2) 冲突集簇中冲突部件集的长度不同时.

从表2和图2可以看出,随着冲突集簇中冲突部件集长度k的增加,全增量算法的计算用时上升幅度较小且耗时最短. 反观其他三种算法,计算耗时急剧增加,运行效率显著下降. 随着解集个数的增加,全增量算法的执行时间明显较低,且在解集个数超过160 000时,全增量算法较其他三种算法可以降低2至3个数量级.

(3) 冲突集簇中集合间相交元素个数不同时.

从表3和图3可以看出,随着冲突集簇中集合间相交元素个数c增多,全增量算法较其他三种算法的运行时间没有大幅度增加,具有较为高效稳定的性能. 随着极小碰集个数的增加,容易看出在解集个数超过100 000时,全增量算法的执行时间较其他三种算法可以降低2到3个数量级.

上述实验验证表明,基于优化增量策略的全增量算法,对于规模较大及解集较多的问题具有显著优势. 这是由于全增量算法较HS-Tree、BHS-Tree、Boolean算法,大幅度降低了子集检测的调用次数,缩小了子集检测时的规模,从而加快了求解速度. 此外,全增量算法首先将原始冲突集簇中的各个集合按照集合的势从小到大依次排序,每次仅增量求解势较小集合,快速补全更新旧解,相较于传统的极小碰集求解算法,进一步提高了求解效率.

表2　集合势不同时的运行时间表　　单位:s

$D_{k\_1\_7\_i}$	$D_{5\_1\_7\_0}$	$D_{5\_1\_7\_2}$	$D_{5\_1\_7\_4}$	$D_{5\_1\_7\_6}$	$D_{5\_1\_7\_8}$	$D_{6\_1\_7\_0}$	$D_{6\_1\_7\_2}$	$D_{6\_1\_7\_4}$	$D_{6\_1\_7\_6}$
Boolean	0.560 2	0.659 6	4.212 9	10.689 2	20.142 7	16.642 0	18.449 0	155.284 9	721.988 0
CIMHS	0.011 9	0.044 7	0.216 4	0.064 3	0.072 2	0.075 2	0.762 1	3.604 8	1.533 2
HS-Tree	1.557 8	3.679 1	13.646 8	29.688 0	53.598 3	32.369 2	68.648 5	461.512 9	1 500.715 7
BHS-Tree	0.922 8	1.103 9	7.009 1	17.408 7	33.126 8	25.390 9	27.334 5	224.298 7	991.381 0
极小碰集个数	7 473	7 977	19 513	31 049	42 585	37 521	39 176	100 026	160 876

表3　集合间相交元素个数不同时的运行时间表　　单位:s

$D_{7\_c\_6\_i}$	$D_{7\_2\_6\_2}$	$D_{7\_2\_6\_4}$	$D_{7\_2\_6\_6}$	$D_{7\_2\_6\_8}$	$D_{7\_1\_6\_0}$	$D_{7\_1\_6\_2}$	$D_{7\_1\_6\_4}$	$D_{7\_1\_6\_6}$	$D_{7\_1\_6\_8}$
Boolean	0.076 8	1.217 2	3.663 1	7.306 4	9.405 2	10.055 9	80.823 6	309.491 3	764.889 6
CIMHS	0.005 3	0.125 6	0.353 5	0.064 2	0.035 2	0.274 1	1.297 5	2.714 4	0.720 2
HS-Tree	1.722 3	7.062 5	15.382 6	27.550 9	14.495 6	26.910 7	163.555 1	555.399 9	1 362.071 3
BHS-Tree	0.172 6	2.600 3	7.919 9	15.850 4	13.383 9	14.221 6	107.641 3	409.576 2	954.395 1
极小碰集个数	2 886	10 072	17 258	24 444	27 577	28 424	74 722	121 020	167 318

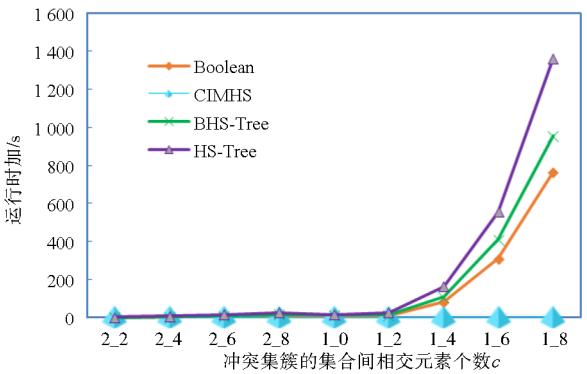

图3　$D_{7_c_6_i}$变化下的求解时间结果图

4.2　国际标准测试电路ISCAS-85冲突集

在MBD领域中,目前许多极小碰集求解算法都在国际标准测试电路ISCAS-85数据集上进行了测试. 为了进一步验证分析全增量算法在系统规模较大时的性能,选取在大规模系统下求解极小碰集效率较为优良的Boolean算法作为实验比较对象.

本文的测试包含了电路c880、c2670、c5315以及c7552的冲突集簇数据. 其中,η代表CIMHS算法较Boolean算法在计算极小碰集时所节省的时间比率.

从表4中可以看出,在部分国际基准电路冲突集簇数据上,相较于目前求解效率优良的Boolean算法,本文提出的全增量算法在系统规模较大的电路冲突集簇中运行效率依然较高. 尤其当解集个数高于10 000时,其求解效果具有显著优势,相较于Boolean算法时间节省达到95%以上. 这主要是由于Boolean算法在最终求得所有碰集之前需要去超集. 然而当系统很大时,极小化过程子集比较

表4　测试结果

电路冲突集簇	极小碰集个数	Boolean/s	CIMHS/s	η/%
c880:10	1 942	0.409 3	0.016 0	96.10
c880:40	3 312	0.497 4	0.004 9	99.01
c880:44	5 568	1.227 8	0.007 1	99.42
c880:51	11 136	4.254 9	0.014 8	99.65
c880:59	9 936	3.734 5	0.023 3	99.37
c2670:11	1 603	0.552 7	0.009 7	98.24
c2670:22	32 349	192.109 8	0.111 2	99.94
c2670:21	17 671	70.756 7	1.229 3	98.26
c2670:42	4 960	3.187 2	0.018 6	99.42
c2670:45	26 768	125.928 6	3.704 4	98.15
c2670:52	39 984	241.759 5	0.235 4	99.90
c2670:54	9 920	10.845 1	0.035 7	99.67
c2670:61	39 990	213.019 0	0.181 8	99.91
c2670:64	49 600	325.661 9	0.196 9	99.94
c2670:75	19 680	25.013 0	0.063 8	99.74
c2670:79	79 980	791.733 3	0.432 9	99.95
c2670:83	22 360	53.510 7	0.071 0	99.87
c5315:10	14 272	112.276 3	0.103 4	99.91
c5315:18	13 025	81.341 1	0.086 0	99.89
c5315:34	23 400	92.483 3	0.320 1	99.65
c5315:40	5 670	9.737 0	0.205 3	97.89
c5315:53	38 610	465.498 3	4.199 1	99.10
c7552:2	7 323	41.211 9	1.430 4	96.53
c7552:3	1 043	0.401 3	0.018 9	95.29
c7552:4	17 272	144.810 0	1.796 9	98.76
c7552:28	4 185	13.296 0	0.050 9	99.62

的规模十分庞大且比较次数较多,因而较全增量算法而言耗时严重. 因此,本算法更加适合系统规模较大且部件间联系较少的故障诊断情形.

5 结束语

本文提出的优化增量策略,在增量求解极小碰集时,根据新增冲突集与原始解集簇元素的关系,将二者进行细化分类. 新碰集只需与原始解集簇中的部分解集进行去超集甚至不需要去超集,减少了子集检测规模及比较次数,缩小了极小化规模,增量时的求解效果较好.

基于优化增量策略,本文提出全增量算法,按照集合势的大小,从小到大依次增量冲突集,直到F为空. 在每次增量过程中,由于去超集的规模降低,且生成的节点个数较少,仅为问题集合簇中集合的个数,因此计算效率比基于子集检测的求解算法明显提升. 特别地,对于冲突集合簇中集合与集合之间交集为空的数据,全增量算法求解效率提升更加明显.

参考文献

[1] FRIEDRICH G, STUMPTNER M, WOTAWA F. Model-based diagnosis of hardware designs[J]. Artificial Intelligence, 1999, 111(1-2): 3-39.

[2] DE K J, WILLIAMS B C. Diagnosing multiple faults[J]. Artificial Intelligence, 1987, 32(1): 97-130.

[3] ZHAO Xiang-fu, TONG Xiang-rong, OUYANG Dan-tong, et al. TreeMerge: Efficient computation of minimal hitting-sets for conflict sets in a tree structure for model-based fault diagnosis[J]. IEEE Transactions on Reliability, 2021, 70(4): 1596-1610.

[4] 何嫱君, 赵相福, 欧阳丹彤, 等. 极小碰集求解算法的性能分析与比较[J]. 电子学报, 2019, 47(5): 1101-1110.
HE Qiang-jun, ZHAO Xiang-fu, OUYANG Dan-tong, et al. Performance analysis and comparison of algorithms for generating minimal hitting sets[J]. Acta Electronica Sinica, 2019, 47(5): 1101-1110. (in Chinese)

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

[6] GREINER R, SMITH B A, WILKERSON R W. A correction to the algorithm in Reiter's theory of diagnosis[J]. Artificial Intelligence, 1989, 41(1): 79-88.

[7] 姜云飞, 林笠. 用对分HS-树计算最小碰集[J]. 软件学报, 2002, 13(12): 2267-2274.
JIANG Yun-fei, LIN Li. Computing the minimal hitting set with binary HS-tree[J]. Journal of Software, 2002, 13 (12): 2267-2274. (in Chinese)

[8] 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.
JIANG Yun-fei, LIN Li. The computation of hitting sets with Boolean formulas[J]. Chinese Journal of Computers, 2003, 26(8): 919-924. (in Chinese)

[9] PILL I, QUARITSCH T. Optimizations for the Boolean approach to computing minimal hitting sets[C]//Proceedings of the 20th European Conference on Artificial Intelligence (ECAI). Montpellier, France: IOS Press, 2012: 648-653.

[10] ZHAO Xiang-fu, OUYANG Dan-tong. Deriving all minimal hitting sets based on join relation[J]. IEEE Transactions on Systems Man Cybernetics: Systems, 2015, 45(7): 1063-1076.

[11] 刘娟, 欧阳丹彤, 王艺源, 等. 结合特征学习的粒子群求解极小碰集方法[J]. 电子学报, 2015, 43(5): 841-845.
LIU Juan, OUYANG Dan-tong, WANG Yi-yuan, et al. Computing minimal hitting sets with particle swarm optimization combined characteristics learning[J]. Acta Electronica Sinica, 2015, 43(5): 841-845. (in Chinese)

[12] 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.
TIAN Nai-yu, OUYANG Dan-tong, 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)

作者简介

魏 霞 女,1995年出生于安徽省亳州市. 烟台大学计算机与控制工程学院研究生,主要研究方向为基于模型的故障诊断.
E-mail: weixia233@163.com

赵相福(通讯作者) 男,1981年出生于山东省济宁市. 烟台大学教授、研究生导师,研究方向为基于模型的故障诊断、智能诊断推理、区块链等. 中国电子学会会员编号:E190035519M.
E-mail: xiangfuzhao@163.com

黄 森 男,1995年出生于山东省枣庄市. 浙江师范大学计算机学院研究生,主要研究方向为基于模型的故障诊断.
E-mail: hs_huangs@163.com