IBWIICC:结合局部独立覆盖检测策略增量求解极小碰集的算法

赵相福[1],黄 森[2],童向荣[1],欧阳丹彤[3],张立明[3],章星林[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;
3. 吉林大学计算机科学与技术学院,吉林长春 130012)

摘 要: 基于模型的诊断推理是人工智能领域的一个重要分支. 其中由冲突部件集产生所有极小碰集是基于模型诊断推理的重要一步. 根据布尔算法的特征,所有的冲突集可以划分为左右两个子集合簇,且左分支集合簇恰好为右分支集合簇的子集,这为由左分支增量产生右分支极小碰集提供了理论基础;另外,在自底向上增量归并元素的过程中,结合局部独立覆盖策略可以直接增量产生所有的极小碰集,从而避免非极小碰集和相同极小碰集的冗余产生. 理论上,右分支解集可由左分支解集增量方法产生,避免了碰集中大量元素的重复计算. 大量实验结果表明:本文提出的算法比之前的布尔算法及相关改进算法都具有显著的效率提升,最高可达约5倍.

关键词: 基于模型诊断; 诊断推理; 极小碰集; 布尔算法; 冲突集; 增量方法
中图分类号: TP306 **文献标识码:** A **文章编号:** 0372-2112(2022)11-2722-08
电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20211356

IBWIICC:Incrementally Computing Minimal Hitting-Sets Combing Local Independence Cover Checking

ZHAO Xiang-fu[1], HUANG Sen[2], TONG Xiang-rong[1], OUYANG Dan-tong[3],
ZHANG Li-ming[3], ZHANG Xing-lin[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*;
3. *College of Computer Science and Technology, Jilin University, Changchun, Jilin* 130012, *China*)

Abstract: Model-based diagnostic reasoning is an important research branch in the field of artificial intelligence. Generating all minimal hitting-sets(MHSs) from conflict sets is a crucial step in the process of model-based diagnostic reasoning. To solve this problem, all conflict sets are divided into left and right parts according to the Boolean algorithm, and the left cluster is just a subset of the right one, which provides a theoretical foundation for incrementally generating the right MHSs from the left ones. In addition, in the process of incrementally merging branch elements from bottom to top, with the help of the local independence cover checking strategy, all MHSs can be directly produced without generating any redundant non-MHSs and duplicated MHSs. Theoretically, the right branch solution can be incrementally generated by the left one. Experimental results show that the efficiency of the proposed algorithm is significantly improved by about five times, compared with the classical algorithm based on Boolean algorithm and its related improved algorithms.

Key words: model-based diagnosis; diagnostic reasoning; minimal hitting-set; Boolean algorithm; conflict sets; incremental method

1 引言

基于模型的诊断推理为人工智能领域中一个重要的研究分支,尤其适用于新兴人造设备的故障诊断[1]. 在基于模型的诊断推理过程中,首先根据待诊断设备的描述和实际观测产生极小冲突集,然后再求解极小冲突集簇的极小碰集即为该设备的候选诊断[2]. 由此,产生极小碰

收稿日期:2021-10-08;修回日期:2022-01-20;责任编辑:覃怀银
基金项目:国家自然科学基金面上项目(No.61972360,No.62076108,No.62072392)

集为基于模型诊断推理的一个重要步骤,其效率将影响整体的诊断效率. 许多学者对求解极小碰集的算法进行了深入研究. 其中,人工智能专家Reiter在1987年最早提出HS-Tree(Hitting Set-Tree)[1]算法,但是该算法由于剪枝策略可能会丢失部分正确解,后来的学者在HS-Tree算法的基础上提出了HS-DAG(Hitting Set Directed Acyclic Graph)[3]、NHST(New Hitting Set-Tree)[4]、BHS-Tree(Binary Hitting Set Tree)[5]等算法,不仅使算法的完备性得到保证,还提高了求解效率. 除了基于树形结构的算法,姜云飞等提出了基于布尔代数公式求解极小碰集的算法[6](后文也称Boolean算法),该算法将极小冲突集合簇编码为布尔表达式,通过布尔代数规则来产生所有极小碰集,极大提高了求解效率,在很多情况下都优于当前其他集中式的求解算法. 后续许多研究人员对布尔代数方法进行了优化研究. 比如,Pill等人使用迭代思想代替递归策略对布尔代数方法进行改进,使冲突集簇中的集合最快缩减到单元素[7]. 刘思光等提出BWIICC(Boolean with Increment Indenpent Coverage Checking)算法[8],在求解过程中对候选解进行极小性判定,得到的结果不会产生超集,但该算法在每选取一个元素时都进行耗时较多的全局独立覆盖判断,从而影响了求解的效率.

目前大多算法除了根据启发式策略进行优化之外,还有部分学者利用分治策略[9]进行并行计算,但他们都没有考虑算法运行过程中冲突集簇本身的结构,导致对部分冲突集簇重复求解[10,11]. 针对该问题,本文深度挖掘了布尔代数算法求解过程中的冲突集簇的结构,发现左分支集合簇恰是右分支的子集,因此在求解完毕左分支时,只需在此基础上增量求解右分支,从而减少对冲突集簇的重复计算. 同时,因增量独立覆盖策略只能区分极小碰集和超集,无法区分相同的极小碰集,而局部独立覆盖策略则可以避免产生相同的极小碰集,保证算法求出且仅求出全部的极小碰集,从而进一步提高求解效率.

2　相关基础

定义1　待诊断系统可定义为一个三元组(D,P,O). 其中,D为系统描述,为一阶谓词公式的集合;P为系统组成部件的集合,是一个有限常量集;O为观测集,是一阶谓词公式的有限集.

定义2　称部件集$\{c_1,c_2,\cdots,c_n\}\subseteq P$是冲突集,当且仅当$D\cup O\cup\{\sim A(c_1),\sim A(c_2),\cdots,\sim A(c_n)\}$是不一致的. 其中,$A$为一元谓词,表示反常的(abnormal),即,$A(c)$为真,当且仅当$c$异常,$c\in P$. 称冲突集是极小冲突集,当且仅当该冲突集的任意真子集都不是冲突集.

定义3　设$F=\{S_1,S_2,\cdots,S_n\}$为集合簇,称H为F的碰集,如果H满足以下条件:

(1)$H\subseteq\bigcup_{S_i\in F}$;

(2)对于任意一个$S_i\in F$,都有$H\cap S_i\neq\varnothing$.

称F的一个碰集H为极小碰集,当且仅当H的任意真子集都不是F的碰集. 若去掉定义3中条件2的约束,则称H为F的候选解.

下面通过例1对以上提到的概念进行说明.

例1　给定集合簇$F=\{\{1,2,3\},\{3,4,5\},\{5,6,7\},\{1,4,7\}\}$. 可知,$\{1,5\},\{1,3,6\},\{1,3,7\},\{1,4,6\},\{2,5,7\},\{3,7\}$及$\{3,4,6\}$都是$F$的碰集,其中虽然$\{1,3,7\}$与$\{3,7\}$都是碰集,但$\{3,7\}$为极小碰集. 特别的,碰集解也是全局问题集合簇的候选解.

3　结合局部独立覆盖的布尔增量求解算法

本节首先介绍文献[8]提出的BWIICC算法,然后针对该算法结合增量策略产生超集的问题提出了局部独立覆盖算法,最后提出IBWIICC(Increment BWIICC)算法.

3.1　BWIICC算法

BWIICC算法是布尔代数求解算法与增量独立覆盖检测IICC(Increment Indenpent Coverage Checking)算法[8]的结合,布尔代数求解算法在每次候选解向上归并元素时,首先进行独立覆盖检测,若候选解中所有元素的独立覆盖度均不为0,则添加该解,否则舍去该候选解. 下面是BWIICC算法的伪代码.

初次调用该算法需对一些参数初始化,把F赋值给J,M置为空. BWIICC算法利用独立覆盖策略,求出的所有解都是极小碰集,避免布尔代数求解算法在计算

算法1　BWIICC(F,J,M)

输入:原始问题集合簇$F=\{S_1,S_2,\cdots,S_n\}$,中间问题集合簇J,候选解集合簇M,其中,J是F的子集;

输出:极小碰集簇M

```
1. IF(J==∅)
2.    RETURN;
3. IF(J=={S})  /*J中只有一个集合*/
4.    RETURN{{e_i}|e_i∈S,IICC(∅,e_i,F)==1};
5. ELSE IF(∃e({e}∈J )){
6.    BWIICC(J,{S_i|S_i∈J∧e∉S_i},M);
7.    M←{m∪{e}|m∈M,IICC(m,e,F)==1};
8.  }//end_ELSE_IF
9. ELSE IF(∃e∀S_i(S_i∈J→e∈S_i)){
10.   BWIICC(F,{S_i−{e}|S_i∈J∧e∈S_i},M);
11.   M←{m⊎{e}|m∈M,IICC(m,e,F)==1};
12.   }//end_ELSE_IF
13.  ELSE{
14.   e←Select_element(J);  /*从J的集合中选取元素*/
15.   BWIICC(F,{S_i|S_i∈J∧e∉S_i},L);
16.   BWIICC(F,{S_i|S_i∈J∧e∉S_i}∪{S_i−{e}|S_i∈J∧e∈S_i},R);
17.   L←{l∪{e}|l∈L,IICC(l,e,F)==1};
18.   M←L⊎R;
19. }//end_ELSE
```

极小碰集时解的极小化过程，从而大大缩短了求解时间.

3.2 冗余碰集的产生

由BWIICC算法的第15、16行可知，该算法可分为左右两个分支，左分支F_1是将集合簇J去掉含有元素e的集合($\{S_i|S_i\in J\wedge e\notin S_i\}$)，右分支$F_2$则只是将$J$中基本元素$e$去掉($\{S_i|S_i\in J\wedge e\notin S_i\}\cup\{S_i-\{e\}|S_i\in J\wedge e\in S_i\}$). 因此，我们得到如下一个有趣的结论：若BWIICC算法能够运行到第15、16行，则左分支集合簇为右分支集合簇的真子集.

基于上述结论和增量算法[9]的原理，若我们先求出左分支的极小碰集，则可以通过增量算法来直接求解对应右分支的极小碰集，且增量的所有集合为$\{S_i-\{e\}|S_i\in J\wedge e\in S_i\}$. 左分支增量求解右分支节，加快了求解速度，然而在某些情况下，增量的过程中会产生相同的冗余极小碰集，具体如下：

左分支问题集合簇$\{\{3,4,5\},\{5,6,7\}\}$会产生$\{3,6\}$，$\{4,6\}$这两个候选解. 当增量$\{2,3\}$，$\{4,7\}$时：$\{3,6\}\xrightarrow{\text{与}\{2,3\}\text{有交集}}\{3,6\}\xrightarrow{\text{与}\{4,7\}\text{没有交集}}\{3,4,6\}$，$\{4,6\}\xrightarrow{\text{与}\{2,3\}\text{没有交集}}\{3,4,6\}\xrightarrow{\text{与}\{4,7\}\text{有交集}}\{3,4,6\}$.

显然，两个候选解在增量后都会求出$\{3,4,6\}$这个极小碰集，这是因为独立覆盖思想只能区分极小碰集和超集(候选解中有无独立覆盖为0的元素)，所以BWIICC算法直接调用增量策略时，最终可能会产生相同的极小解，这样就需要最后一步复杂的去重处理，从而影响了求解的效率，由增量算法思路可以得到命题1：

命题1 若求出原有问题集合簇的所有极小碰集，则增量一个集合后不会产生相同的极小碰集.

证明 根据极小碰集的定义，每两个基础解之间至少有一个元素不相同. 在增量前每个解集根据与增量集合交集是否为空可划分为M_1和M_2两部分. 若交集非空则M_1无需处理即为极小碰集，若交集为空则M_2需逐一扩展增量集合的元素，显然，M_1中没有相同的解集.

由于M_2中每个解集与增量集合交集为空，则对M_2中每个解集来说都是新元素，增量完毕后也无相同的解；反之，若M_2中新生成的某解集m_2与M_1中的某解集m_1相同，则说明原极小碰集m_2为m_1的真子集，与前提“根据极小碰集的定义，每两个基础解之间至少有一个元素不相同”相矛盾，故新生成的解与M_1的解集相比也无相同的解. 证毕

由命题1可知，如果候选解在每次增量计算前，能够保证所有的解都是当前所关注问题集合簇的极小候选解，那么增量计算后不会产生相同的解. 注：命题1只保证不会产生相同的极小碰集，而不保证不会产生非极小的碰集. 当然，BWIICC算法保证了不会产生非极小碰集.

命题2 若左分支某个候选解与右分支增量集合簇中所有集合都有非空交集，则该候选解为超集.

证明 若左分支某个候选解与所有增量集合有非空交集，则在增量计算时该解不需添加任何元素即为当前大集合簇的极小碰集；则，当该左分支解集归并上层分支元素时，显然将成为超集. 证毕

根据命题2，当右分支存在至少一个增量集合与某左分支候选解交集为空时，该候选解归并上层元素后即为当前的极小碰集，因为上层扩展元素至少局部独立覆盖了该增量集合，接下来介绍本文提出的局部位置标记算法(Local Position Marked Algorithm，LPMA).

3.3 LPMA算法

在BWIICC算法中，相关变量及其含义为：原始问题集合簇为F，中间问题集合簇为J，且初始化为：$J\leftarrow F$，增量集合簇$F_2=\{S_i-\{e\}|S_i\in J\wedge e\in S_i\}$因为BWIICC算法体现的是原问题集合簇$F$的特征，而不是对中间局部问题集合簇的特征，所以在进行左分支增量求解右分支时有可能出现如下情形：增量产生的解虽不是原始问题集合簇F的某极小碰集的超集，但却是中间(局部)问题集合簇F_1的超集，从而导致产生相同冗余极小碰集. 基于上述分析，为避免最终产生相同的冗余解，需确保每次增量产生的解都是中间局部问题集合簇的极小候选解. 为此，我们为问题集合簇新增一维数组C，仅标记中间局部问题集合簇的特征，具体如算法2所示.

算法2 LPMA(J,F_1,F_2,e,C)

输入：中间问题集合簇J，中间问题集合簇递归时生成的左、右分支F_1、F_2，$F_1=\{\{S_i|S_i\in J\wedge e\notin S_i\}\}$，$F_2=\{\{S_i-\{e\}|S_i\in J\wedge e\in S_i\}\}$，$e$为$J$中重复度最高的元素，$C$为中间问题集合簇的一维数组属性，用来标记$J$中集合在原始问题集合簇中的位置；

输出：C

```
1. FOR (i=1; i ≤ N; i++){  /*N为全局集合的个数*/
2.     IF(e∈S_i (S_i∈J)){
3.         F_1.C[i]=1;
4.         F_2.C[i]=1;
5.     }
6.     IF(J.C[i]==1)
7.         F_1.C[i]=1;
8. }
```

算法2简要解释如下：初始化问题集合簇J、F_1、F_2的C每一位的值均为0，C数组的长度为原始问题集合簇中集合的个数. 因为元素独立覆盖的集合是覆盖初始问题集合簇中集合的位置，所以C也只标记集合在原始问题集合簇中的位置.

BWIICC算法在每一次向下递归时F_1变为J，当J选

取元素e进行递归分支时,如果S_i含有元素e,则左分支$F_1.C[i]$=1(i代表含有元素e的集合,此时左分支不含有该集合),而$F_2.C[i]$=1(右分支含有该集合),即每次向下递归处理时$F_1.C[i]$含有S_i,在J含有S_i的前提下F_2不含有S_i位置集合. 算法2的时间复杂度简要分析如下:LPMA算法只需要遍历一遍问题集合簇(算法2第1行),判断一个元素是否在集合里面(第2行),因此时间复杂度为$O(N)$. 接下来介绍本文提出的增量产生右分支的IRB(Increment Right Branch)算法.

3.4 IRB算法

IRB算法的基本思想:通过独立覆盖策略对右分支增量求解,并通过LPMA算法定位左分支与右分支集合的位置进行局部独立覆盖检测. 若候选解与增量集合的交集为空,则将候选解扩展归并元素并进行独立覆盖检测,若扩展结果中不存在独立覆盖为0的元素(候选解不是全局集合簇某极小碰集的超集),则进一步判断该集合是否为中间问题集合簇的极小候选解,伪代码见算法3.

算法3　IRB(C,F_1,F_2,L)

输入:左分支集合簇F_1、右分支集合簇F_2及数组C,L为左分支极小候选解;

输出:右分支极小候选解R

```
1. R←L;  /*在左分支的基础上增量求解右分支*/
2. FOR (i∈{1,2,···,N},∃F2.c[i]==1){
3.   F1.C[i]=0,F2.C[i]=0;
4.   FOR (m∈R,m∩Si==∅,Si∈F2){
5.     FOR (e'∈Si,∃IICC(m,e',F)==1){
6.       y=1;
7.       FOR (j∈{1,2,···,N},∃F1.C[j]==1){
8.         IF(集合j被m的元素独立覆盖)
9.           IF(该元素独立覆盖度减1后为0){
10.            y=0; break; }
11.          } /*7~11行局部去超集*/
12.        IF(y==1){    /*是局部极小候选解*/
13.          恢复7~11行的操作;
14.          R←m∪{e'};
15.        }
16.    }
17.  }
18. }
19. 删除L中与F2都有交集的解;
```

IRB算法通过LPMA算法获取了C属性,利用左分支已求出的解集增量求解右分支(算法3第1行). 遍历待增量扩展的集合(第2行),并把该集合位置加入到左分支局部问题集合簇中(第3行). 若候选解与增量集合交集非空,则该解为极小候选解无需处理;而若候选解与增量集合交集为空,则对增量集合中的每一个元素进行增量计算(第4~17行). 算法3的第5行为候选解归并元素前进行全局独立覆盖判断,若返回0则说明候选解添加该元素后将变为超集因而需要删除;而若返回1则进一步判断该解是否为局部极小候选解(第7~11行).

因为BWIICC算法思想是对全局问题集合簇的覆盖检测,所以我们需要屏蔽中间问题集合簇不含有的集合覆盖. 如果$F_1.C[i]$值为1,说明此时局部问题集合簇不含有集合S_i,则需将独立覆盖该集合的元素独立覆盖度减1. 若减1后该元素的独立覆盖度变为0,则说明该候选解是局部问题集合簇的超集,需删除(第9行);若在整个FOR循环(第7~11行)结束后元素的独立覆盖度没有变为0,则恢复减1的操作并归并该元素(第13行). 算法的第19行删除左分支与F_2中所有集合交集均非空的候选解,为左分支减少了向上归并解的数量,进一步优化了求解效率,算法3的时间复杂度简要分析如下:

一方面,算法的第19行用于判断左分支解集与F_2中的集合是否交集非空,因算法的第4行已经判断,故只需扫描一遍之前的比较结果,因此时间复杂度只与L中候选解的个数成线性比例.

另一方面,左分支候选解在增量求解右分支时,为简化起见,设左分支解集个数为n,候选解元素个数为m,增量集合个数为k,每个增量集合中有$h(h>1)$个元素,全局问题集合簇中集合个数为N. 在最坏的情况下,每个候选解都与增量集合交集为空且生成的解都需全局及局部独立覆盖去超判断,增量k个集合后共求出nh^k个解,因而总的时间复杂度可表示为:$O(\frac{mn}{h-1}h^k+\frac{Nn}{h-1}h^{k+1}+\frac{Nn}{h-1}h^k)=O(\frac{n}{h-1}h^k(m+Nh+N))$,而BWIICC算法对应时间复杂度为$O(nh^kN(m+k))$.

BWIICC算法与IRB算法的时间复杂度之比为$\frac{(h-1)N(m+k)}{m+Nh+N}$,对$k$、$N$、$h$、$m$分别求偏导得$\frac{(h-1)N}{m+Nh+N}$、$\frac{(h-1)(m+k)m}{(m+Nh+N)^2}$、$\frac{N(m+k)(2N+m)}{(m+Nh+N)^2}$、$\frac{(h-1)N(Nh+N-k)}{(m+Nh+N)^2}$,$k$、$N$、$m$大于0,因$N$是全局问题集合簇中集合的个数,$k$是局部问题集合簇中集合的个数,所以$N>k$,且$Nh+N>k$.

$h>1$,前述4个导数均恒大于0,且比值随着k、N、h、m增大而增大,其中当k增大时增长率为定值,N、h、m增长率越来越小,且最后趋于一定值. 此时,BWIICC算法与IRB算法的时间复杂度之比变为$\frac{(h-1)N(m+k)}{m+Nh+N}$. 当$k=m=h=2$时,该比值为$\frac{4N}{2+3N}>1$. 当$k$、$m$、$h$均大于2时比值增大,IRB算法比BWIICC算法增量求解具有更

高的效率.

3.5 IBWIICC算法

基于前述分析,在BWIICC算法的基础上,有机结合IRB增量算法,产生结合局部独立覆盖的布尔代数增量求解算法IBWIICC,只需将BWIICC算法的第16行改为以下两行伪代码:

1. C←LPMA(J,F_1,F_2,e,C);
2. R←IRB(C,F_1,F_2,L);

其中,LPMA算法用于标记局部问题集合簇中的集合在全局集合簇中的位置,与独立覆盖检测去超集相比,时间花费基本可以忽略.

例3 求解集合簇F={{1,2,3},{3,4,5},{5,6,7},{1,4,7}}的极小碰集.

如图1所示,假设已经求出左分支问题集合簇{{3,4,5},{5,6,7}}所有候选解为{5},{3,6},{3,7},{4,6}和{4,7},且LPMA已经标记各个集合对应原始问题集合簇的位置,分别为:$\begin{cases}F_1.C[i]=1, i=1,4\\F_1.C[i]=0, i=2,3\end{cases}$,$\begin{cases}F_2.C[i]=1, i=1,4\\F_2.C[i]=0, i=2,3\end{cases}$.此时,首先复制一份左分支候选解集用于增量计算,$F_2.C[1]$=1(IRB算法第2行),对右分支的集合{2,3},增量前$F_1.C[1]$的值置为0,代表左分支含有S_1,同时$F_2.C[1]$的值置为0(IRB算法第4行),代表右分支不再含有S_1.

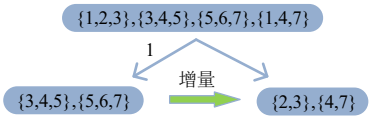


图1 IBWIICC算法计算极小碰集

因为集合{3,6},{3,7}与待增量的集合{2,3}交集非空,所以不需要操作;而{5},{4,6},{4,7}与{2,3}交集为空,因而需要逐一增量扩展,得到{2,5},{3,5},{2,4,6},{3,4,6},{2,4,7},{3,4,7}.注意:对于{3,4,7}这个解,实际上在全局独立覆盖去超集时(IRB算法第7行)就已经去掉了.而对于{2,4,6}与{2,4,7}:候选解{2,4,6}中元素4的独立覆盖为2,独立覆盖了集合{3,4,5}与{1,4,7},在做局部去超集判断时元素4的独立覆盖度只减少1,不会变为0,所以不是超集;虽然{2,4,7}这个集合中元素4的独立覆盖度为1,但元素4没有独立覆盖S_4,所以也不需减1.需要注意的是,此处的去超集检测只是针对当前局部集合簇以外的集合的独立覆盖度相减,但若局部以外的集合没有被独立覆盖则不需减1.具体来说,当前局部集合簇以外的集合为{1,4,7}(S_4),被元素4、7同时覆盖,所以没有任何元素独立覆盖该集合,去超集时覆盖这个集合元素的独立覆盖度则不需减1.

第一次增量集合{2,3}得到了候选解{3,6},{3,7},{2,5},{3,5},{2,4,6},和{2,4,7},左右分支集合簇的C值分别为:$\begin{cases}F_1.C[i]=1, i=4\\F_1.C[i]=0, i=1,2,3\end{cases}$,$\begin{cases}F_2.C[i]=1, i=4\\F_2.C[i]=0, i=1,2,3\end{cases}$.接着对集合{4,7}增量扩展,使$F_1.C[4]$=0,此时局部问题集合簇即为全局问题集合簇($F_1.C[i]$值全为0,无法进入IRB算法第10~15行),所以只需对候选解做全局独立覆盖判断(IRB算法第7行).

同理,集合{3,7},{2,4,6},{2,4,7}与集合{4,7}交集非空,即为极小碰集,而集合{3,6},{2,5},{3,5}分别逐一扩展元素4、7并做独立覆盖检测只能得到{3,4,6},{2,4,5},{3,4,5},{2,5,7},这是因为其中的集合{3,6,7},{3,5,7}因含有独立覆盖度为0的元素,所以被全局独立覆盖策略删除了,最终右分支共得到7个极小碰集解.左分支问题集合簇F_1的候选解再添加元素1并进行独立覆盖检测得到{1,5},{1,3,6},{1,4,6},{1,4,7}.至此,产生所有11个极小碰集.

4 实验与结果

本文前述提到的算法均已实现,实验测试平台如下:Windows 10操作系统,CPU Intel Core i7-1065G7 1.30 GHz 4核 16 GB RAM,C++.

4.1 BWIICC、IBWIICC与其他算法比较与分析

为了测试本文所提算法的有效性,设置数据集合簇Dk_c_m_i,其中k代表集合中元素的个数,c代表两个相邻集合公共的元素个数,m代表集合的个数,i表示额外增加的元素.一般的集合簇中数据可表示为:{0,1,2,…,k−1},{k−c,k−c+1,…,(k−c)+k−1},{2×(k−c),2×(k−c)+1,…,2×(k−c)+k−1},…,{m×(k−c),m×(k−c)+1,…,m×(k−c)+k−1},{(m+1)×(k−c),(m+1)×(k−c)+1,…,(m+1)×(k−c)+i−1}.例如D6_2_4_4表示的集合簇为:{0,1,2,3,4,5},{4,5,6,7,8,9},{8,9,10,11,12,13},{12,13,14,15,16,17},{16,17,18,19}.

图2展示了BWIICC、IBWIICC算法与目前效率较高的Boolean、BHS-Tree算法的比较结果.由图2可以看出,IBWIICC算法仍占据最优地位,其次是BWIICC算法.Boolean算法一般优BHS-Tree算法.使用独立覆盖检测策略的BWIICC算法和IBWIICC算法比使用子集检测的Boolean和BHS-Tree算法求解效率有显著的提升,且当解集的数目超过十万时,BWIICC与IBWIICC两种算法的效率一般高于后面两种算法约达三个数量级.

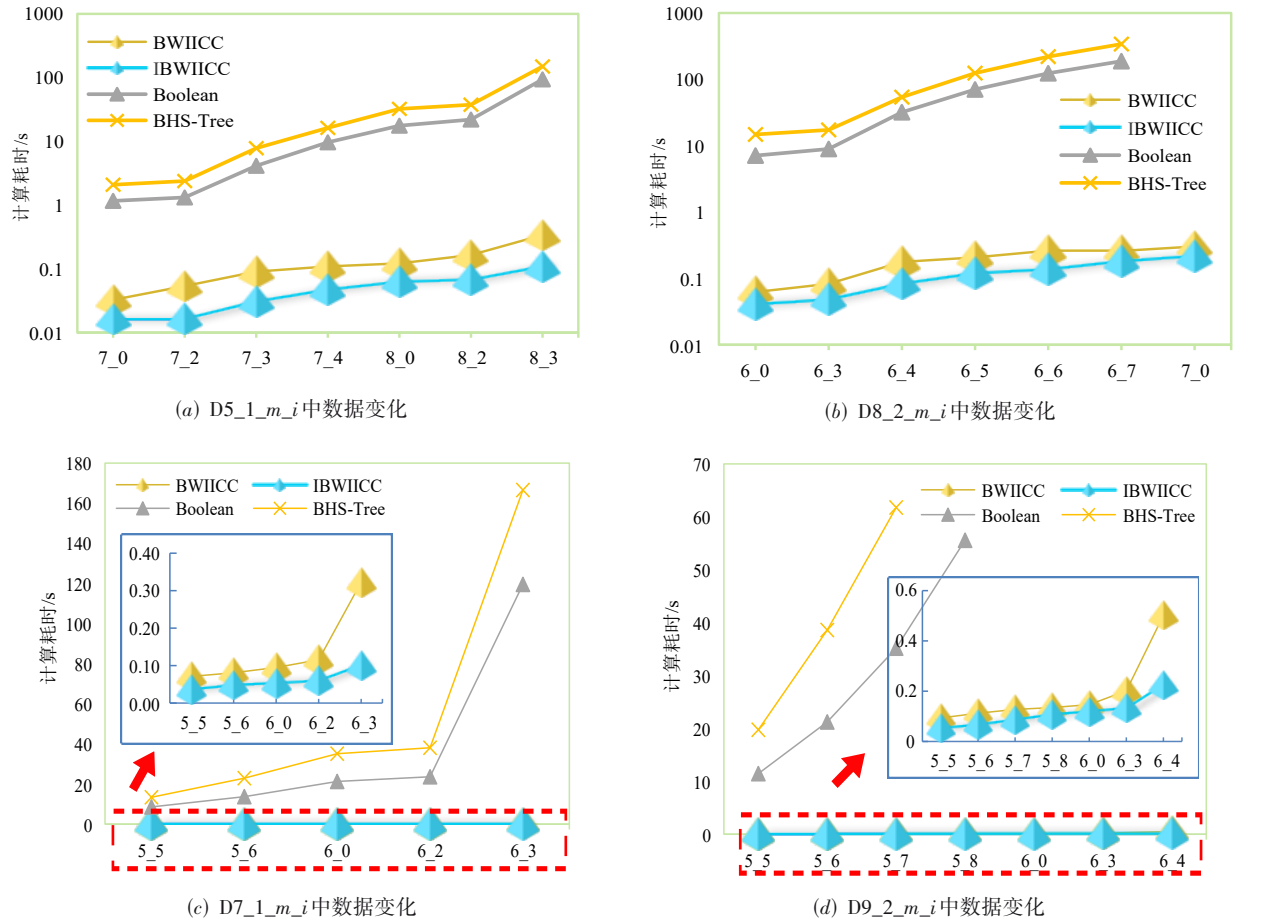
4.2 增量策略的影响情况分析

为进一步分析增量策略的影响,设置第二组数据序列Em_k_h,其中k代表本集合簇内除第一个元素相同外其余元素均不同的集合个数,对应于IRB算法复杂

度分析的增量集合个数;m 代表 k 的种类数,可近似作为候选解中元素的个数;h 代表每个集合中元素的个数,对应增量集合中有 $h-1$ 个元素.

集合簇 Em_k_h 数据集合可表示为:$\{1,m+1,m+2,\cdots,m+h\},\{1,m+h+1,m+h+2,\cdots,m+2\times h\},\cdots,\{1,m+(c-1)\times h+1,m+c\times h+2,\cdots,m+c\times h\},\{2,m+k\times h+1,\cdots,m+(k+1)\times h\},\cdots,\{m,m+(k-1)\times k\times h+1,\cdots,m+m\times k\times h\}$. 例如 E4_2_4 包含的集合为:$\{1,5,6,7\},\{1,8,9,10\},\{2,11,12,13\},\{2,14,15,16\},\{3,17,18,19\},\{3,20,21,22\},\{4,23,24,25\},\{4,26,27,28\}$.

图 3 展示了相关的实验结果,其中 B/I 表示 BWIICC 与 IBWIICC 算法运行时间之比,对每一组数据只设置一个变量. 由图可以看出,随着 h、k、m 的动态增加,B/I 值也相应地逐渐增大.



(a) D5_1_m_i 中数据变化　　(b) D8_2_m_i 中数据变化

(c) D7_1_m_i 中数据变化　　(d) D9_2_m_i 中数据变化

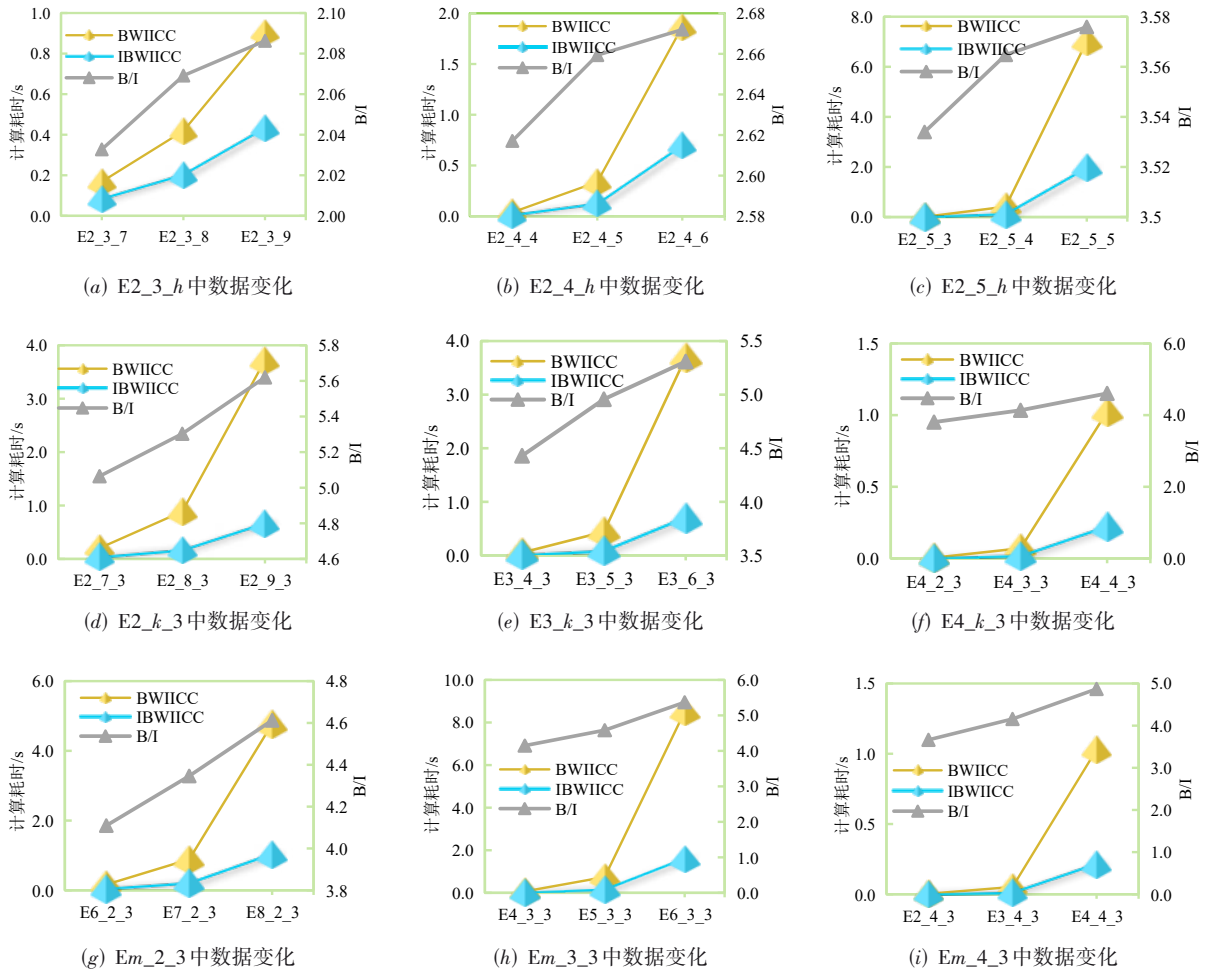
图 2　IBWIICC、BWIICC、Boolean、BHS-Tree 算法运行时间比较

(1) 在数据集 E2_3_h、E2_4_h、E2_5_h 中,当增量集合中元素的个数 h 不断增大时,B/I 变化分别为:1.93~2.08,2.62~2.67,3.53~3.57,平均增加 0.031,0.018,0.014. 由此可见,增量集合元素的增加对效率影响较小,且增长趋势越来越小,BWIICC 算法花费的时间约是 IBWIICC 算法的 2.77 倍.

(2) 在数据集 E2_k_3、E3_k_3、E4_k_3 中,当增量集合的个数 k 不断增大时,B/I 变化分别为:4.72~5.62,4.43~5.31,3.80~4.60,平均增加 0.225,0.293,0.267,由此可见,增量集合的增加对效率影响较大,BWIICC 花费的时间约是 IBWIICC 算法的 5. 18 倍.

(3) 在数据集 Em_2_3、Em_3_3、Em_4_3 中,当候选解中元素的个数 m 不断增大时,B/I 变化分别为:3.98~4.61,4.15~5.37,3.66~4.87,平均增加 0.159,0.407,0.402,由此可见,借用左分支元素越多,无需重新计算元素独立覆盖的次数也就越多,BWIICC 花费的时间约为 IBWIICC 算法的 4.95 倍.

根据人工数据的设计数据规则可知 $N=k\times c$,其中当问题集合簇的个数 N 为 6~10 时,B/I 值分布在 1.9~3.5 之间;N 为 10~15 时,B/I 值分布在 3.5~4.9 之间;N 为 16~18 时,B/I 值分布在 4.6~5.6 之间. 由此可见,问题集合簇中集合的个数越多,BWIICC 算法的候选解元素去超集时遍历的集合个数也越多,而 IBWIICC 算法只需对增量的元素进行独立判断,故节省的时间也越多.



(*a*) E2_3_*h* 中数据变化 (*b*) E2_4_*h* 中数据变化 (*c*) E2_5_*h* 中数据变化

(*d*) E2_*k*_3 中数据变化 (*e*) E3_*k*_3 中数据变化 (*f*) E4_*k*_3 中数据变化

(*g*) E*m*_2_3 中数据变化 (*h*) E*m*_3_3 中数据变化 (*i*) E*m*_4_3 中数据变化

图 3 IBWIICC、BWIICC 算法运行时间及其比率

5 结束语

极小碰集问题在许多重要领域都有着广泛的应用. 本文深入研究了目前效率较高的布尔求解算法的结构, 提出了增量策略对 BWIICC 算法进行增量求解的 IBWIICC 算法, 在增量的过程中提出了局部独立覆盖检测的策略, 有效避免了相同极小候选解的生成; 在候选解极小化时减少了右分支候选解的独立覆盖检测次数, 使得大部分解集在左分支解的基础上得以增量地求出, 大大提高了计算效率. 所提算法在大量的人工数据上进行了较广泛而充分的测试, 实验结果表明, IBWIICC 算法可比 BWIICC 算法效率最高可达五倍以上.

参考文献

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

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

[3] 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.

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

[5] 姜云飞, 林笠. 用对分 HS-树计算最小碰集[J]. 软件学报, 2002, 13(12): 2267-3374.
JIANG Yun-fei, LIN Li. Computing the minimal hitting set with binary HS-Tree[J]. Journal of Software, 2002, 13 (12): 2267-3374. (in Chinese)

[6] 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)

[7] PILL I, QUARITSCH T. Optimizations for the Boolean approach to computing minimal hitting sets[C]//Proceedings of the 20th European Conference on Artificial Intelligence,

2012: 648-653.

[8] 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.
LIU Si-guang, OUYANG Dan-tong, ZHANG Li-ming. Minimization of candidate solutions in minimal hitting sets solution[J]. Journal of Software, 2018, 29(12): 3733-3746. (in Chinese)

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

[10] 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)

[11] 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)

作者简介

赵相福　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.
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

童向荣　男,1975年出生于山东省烟台市. 烟台大学教授、研究生导师,研究方向为计算机科学、智能信息处理、社交网络等.
E-mail: xr_tong@163.com

欧阳丹彤　女,1968年出生于吉林省长春市. 吉林大学教授、博士生导师,研究方向为基于模型的故障诊断、自动推理等.
E-mail: ouyd@lu.edu.cn

张立明　男,1980年出生于吉林省长春市. 吉林大学教授、研究生导师,研究方向为基于模型的故障诊断、自动推理等.
E-mail: limingzhang@jlu.edu.cn

章星林　女,1997年出生于安徽省安庆市. 浙江师范大学计算机学院研究生,研究方向为基于模型的故障诊断.
E-mail: xinglinzhang2022@163.com