# 基于二进制标签的改进 HSSE-tree 算法

杜 敏 冯文全 王 冬 赵 琦

(北京航空航天大学 电子信息工程学院, 北京 100191)

**摘 要**: 随着对航天器自主生存能力要求的提高,基于模型的故障诊断成为国内外的研究热点. 计算全体极小碰集是基于模型的故障诊断中的关键步骤,以 HSSE-tree 算法为基础,结合二进制位标记,提出一种 HSSE-tree 的高效改进算法——Binary-label HSSE. 改进算法采用二进制位标记来代替实际节点元素,并采用了有效的剪枝策略及节点扩展方式,避免了 HSSE-tree 算法中存在的节点个数及超集个数随着问题规模增大而产生的爆炸式增长的问题;此外,改进算法采用二进制位运算,避免了判断碰集及判断是否超集时的元素遍历,使算法的运行时间进一步减少. 仿真结果表明,与 HSSE-tree 算法相比,改进算法的消耗时间及占用内存均有了大规模减少. 这为航天器系统的故障诊断及实时诊断提供了理论依据和应用基础.

**关 键 词**: 极小碰集; 基于模型的故障诊断; 二进制标签

**中图分类号**: TP 306

**文献标识码**: A **文 章 编 号**: 1001-5965(2012)11-1547-05

## Improved HSSE-tree method based on binary label

Du Min Feng Wenquan Wang Dong Zhao Qi

(School of Electronics and Information Engineering, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

**Abstract**: Because of the increase of the demand for the autonomy of spacecrafts, model-based diagnosis has been a hot research spot both at home and abroad. Computing all minimal hitting sets is a key step of model-based diagnosis. An effectively improved method of HSSE-tree called Binary-label HSSE based on HSSE-tree and combining binary labels was put forward. The improved method used binary digits to mark the real elements of the nodes, and used effectively pruning and expanding strategies, to avoid the main problem of HSSE-tree, the explosive growth of the expanded nodes and supersets along with the dimension of the problems. Additionally, computing between binary digits can avoid the traverse of every element in a node when judging whether the node is a minimal hitting set (MHS), which also contributes to the significant decrease of the run time. Simulation results show the improved method costs much less space and time than the HSSE-tree method, which provides both theoretical and applicative foundation for fault diagnosis and Real-time diagnosis of spacecraft system.

**Key words**: minimal hitting set; model-based diagnosis; binary label

随着我国进入深空探测阶段,更多不可预知和不确定的因素,妨碍了地面的直接操控,这对航天器的可靠性提出了更高的要求,航天器的自主生存能力受到越来越多的关注. 基于模型的故障诊断作为人工智能领域的一个重要分支,以其不存在知识获取瓶颈、诊断过程可解释、复杂度低的优势,成为国内外研究的热点,并已在美国多次空间任务中得到应用[1-2].

基于模型的故障诊断一般可由冲突识别和候选生成两个步骤组成,候选生成为求解最小诊断解,即冲突集的极小碰集. 在计算全体极小碰集的算法中,基于 SE-tree[3](集合枚举树)的 HSSE-

**收稿日期**: 2011-06-17; **网络出版时间**: 2012-11-16 11:08

**网络出版地址**: www.cnki.net/kcms/detail/11.2625.V.20121116.1108.010.html

**作者简介**: 杜 敏(1987-),女,山东肥城人,硕士生,dodolonny@gmail.com.

tree[4]方法既不像 HS-TREE[5]、HST-TREE[6]一样会因为剪枝而丢失正确解,也不像布尔代数[7]方法一样算法设计复杂,是一种完备的、效率较高而又易于实现的诊断方法. 但是,HSSE-tree 方法仍然具有以下几个问题:①冲突集簇中各冲突集之间元素个数、内容均相差较大时,HSSE-tree 扩展节点数目随问题规模而急剧增加;②HSSE-tree 方法没有任何避免产生极小碰集的真超集的措施,节点扩展中产生了大量超集;③HSSE-tree 方法中,每一层的节点均比上一层多一个元素,随着节点数目的急剧增加,算法占用内存也有了爆炸式增长;④HSSE-tree 在计算某节点是否为冲突集簇的碰集时,需要遍历该节点元素及冲突集簇中的所有元素,这个过程消耗了大量时间.

针对 HSSE-tree 算法求解极小碰集的缺点,本文结合二进制标记方法对 HSSE-tree 算法进行了改进. 同时,改进算法还采用了有效的剪枝策略及节点扩展方式,进一步提高了算法效率.

## 1 问题描述

**定义 1**[5] 设 $F$ 是集合簇,称 $H$ 为 $F$ 的一个碰集,当且仅当 $H\subseteq \bigcup_{S\in F} S$,且对于每一个 $S\in F$,都有 $H\cap F\neq\varnothing$. 若 $F$ 的一个碰集的任意真子集均不是 $F$ 的碰集,则称该碰集为 $F$ 的极小碰集(MHS,Minimal Hitting Sets),否则称该碰集为非极小的碰集(MHS 的超集).

SE-tree(集合枚举树,见图 1)由 Rymon R[8]提出,可以方便地枚举出一个集合中所有可能的组合.

**推论 1** 若某一节点元素及其之后可能扩展出的每个元素中,都不含有集合簇 $F$ 中某一集合 $S_i$ 的元素,则该节点及其子节点都不是碰集.

**证明** 若某一节点扩展到最后也不会与集合簇 $F$ 中某集合 $S_i$ 有交集,由定义 1,扩展该节点不会产生集合簇 $F$ 的碰集,可将该节点剪枝. **证毕**

推论 1 在改进算法中作为重要的剪枝规则.

**推论 2** SE-tree 第一层为空集,从第二层的单元素节点开始,每个节点的扩展都可以通过取自身节点与其每个右亲兄弟节点的并集得到.

**证明** SE-tree 在扩展每个节点时,是通过取该节点及其之后的每一个元素的并集来得到该节点的子节点,而位于该节点之后的每一个元素均存在于该节点的右亲兄弟节点中,因此可以通过取该节点与其右亲兄弟节点的并集来扩展它. 如图 1 所示,节点$\{a, b\}$的右亲兄弟节点为$\{a, c\}$,$\{a, d\}$,分别取$\{a, b\}$与$\{a, c\}$,$\{a, b\}$与$\{a, d\}$的并集,$\{a, b\}$可以扩展出节点$\{a, b, c\}$,$\{a, b, d\}$. 通过这种扩展方式,若节点$\{a, d\}$已被判定为 MHS,那么在扩展$\{a, b\}$和$\{a, c\}$时就可以避免取$\{a, d\}$的并集,从而避免了 MHS 的超集节点$\{a, b, d\}$和$\{a, c, d\}$的产生. **证毕**

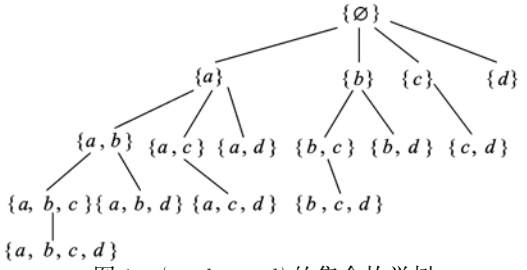

图 1 $\{a, b, c, d\}$的集合枚举树

在改进算法中,采用推论 2 所述的扩展方式,以避免产生已是碰集的亲兄弟节点的超集.

## 2 Binary-label HSSE 算法描述

二进制是计算技术中广泛采用的一种数制. 二进制数据是使用 0 和 1 两个数码来表示的数.

改进算法中,并不将节点元素实际扩展出来,而是采用二进制位标记节点中是否含有枚举集合 COMPS 中的对应位元素,0 为不含有,1 为含有,从低位到高位依次表示节点中每一个元素在集合 COMPS 中的出现情况. 如枚举集合 COMPS = $\{a, b, c, d\}$,节点$\{a, b, d\}$的二进制位表示为 1011. 此外,对于集合簇 $F$ 中的每个集合,在初始化时也做如此标记.

下面说明算法中的其他规则及初始化操作.

存储[4]:链表表达枚举集合链 enumSetLink、待扩展集合链 needExLink 及极小碰集链 pHS.

节点:包括 3 个二进制标签及一个扩展标识:ElesInCOMPS 标记当前节点元素;ParentEles 标记当前节点的父节点元素;剪枝标记 PruneSign 标记当前节点元素与该节点可能扩展出的每个元素的并集;NoExpandSign 为布尔型,取真值时表示该节点不需扩展.

初始化:enumSetLink = needExLink = pHS = $\{\}$,设 $q$ 指向 needExLink 的第一个集合节点;令 $m$ = COMPS 中元素个数,$n$ = 集合簇 $F$ 中集合个数,$k=\min(n, m)$,$k$ 作为算法终止的参考条件,原理详见文献[4]. 将集合簇 $F$ 根据元素个数由少到多排序后,利用排序后的 $F$ 中单元素的出现次序生成 COMPS:COMPS = $\bigcup_{S\in F} S = S_1\cup S_2\cup\cdots\cup S_n = \{c_1, c_2, \cdots, c_m\}$,并将 $F$ 中的每个集合分别按

照其元素在 COMPS 集合中出现的位置用二进制位标记. 标记完成后,集合簇 $F$ 中的 $n$ 个集合变为 $n$ 个数字:$F=\{S_1,S_2,\cdots,S_n\}=>\{fmS_1,fmS_2,\cdots,fmS_n\}$($fmS_i$ 表示集合 $S_i$ 经过二进制位标记格式化后的数字). 对集合簇 $F$ 进行排序可以减少节点扩展. 此外,该排序操作仅在初始化时针对集合簇长度 $n$ 进行一次,排序时间相对于整个算法时间几乎可以忽略不计.

### 2.1　IsHS 算法的实现

输入:长度为 $n$ 的数字集合 $F=\{fmS_1,fmS_2,\cdots,fmS_n\}$,节点元素标记 ElesInCOMPS 以及同一节点剪枝标记 PruneSign.

输出:数字 0,1 或 2. 返回 0 表示该节点是碰集;返回 1 表示该节点不是碰集,且需剪枝;返回 2 表示该节点不是碰集,且不需剪枝.

Step 1:$i=0$,从 $F$ 中第一个数字开始;

Step 2:若 $fmS_i$&ElesInCOMPS=0,转 Step 4;

Step 3:$i=i+1$,若 $i=n$ 返回 0,否则转 Step 2;

Step 4:若 $fmS_i$&PruneSign=0,返回 1;否则返回 2.

该算法最坏情况下需要遍历 $F$ 中的 $n$ 个元素,时间复杂度为 $o(n)$.

### 2.2　Binary-label HSSE 算法的实现

输入:集合簇 $F$.

输出:集合簇 $F$ 的所有极小碰集.

Step 1:初始化 enumSetLink 为所有单元素集合链,设指针 $p$ 指向 enumSetLink 的第一个节点,设 $i=2$,表示当前的 SE-tree 层次;

Step 2:若 $i>k$,则算法终止;

Step 3:若 $p=$NULL,将 $q$ 指向 needExLink 的第一个节点,转 Step 8;

Step 4:调用算法 IsHS 判断 $p$ 所指节点,若返回值为 1 转 Step 7,返回值为 2 则将该节点加入 needExLink 后转 Step 6,返回值为 0 则继续;

Step 5:判断 pHS 中是否存在 $p$ 所指节点的非空真子集.(判断方法:依次将 pHS 中节点的 ElesInCOMPS 与 $p$ 所指节点的 ElesInCOMPS 相与,看结果是否与 pHS 中节点的 ElesInCOMPS 相等.)若不存在,则将其加入 pHS;

Step 6:释放 $p$ 所指节点,$p$ 指向 enumSetLink 中下一邻节点,转 Step 3;

Step 7:将该节点的 NoExpandSign 置为 true 后加入 needExLink,转 Step 6;

Step 8:若 $q=$NULL,转 Step 12;

Step 9:若 $q$ 所指节点的 NoExpandSign 为 true,则释放该节点,$q$ 指向 needExLink 中下一邻节点,转 Step 8,否则设指针 iter 指向 needExLink 中 $q$ 所指节点的下一个节点;

Step 10:若 iter=NULL,或 iter 所指节点与 $q$ 所指节点的 ParentEles 不同,$q$ 指向 needExLink 中下一邻节点,转 Step 8;

Step 11:以如下方式扩展出 $q$ 所指节点的一个子节点:子节点 ElesInCOMPS=$q$ 所指节点的 ElesInCOMPS | iter 所指节点 ElesInCOMPS,子节点 ParentEles= $q$ 所指节点 ElesInCOMPS,子节点 PruneSign= iter 所指节点 | $q$ 所指节点 ElesInCOMPS,子节点 NoExpandSign= false,将扩展出的新节点加入 enumSetLink. iter 指向 needExLink 下一节点,转 Step 10;

Step 12:$p$ 指向 enumSetLink 的第一个集合节点,$i=i+1$,转 Step 2.

产生的节点总数最多为 $\sum_{i=2}^{k}C_m^i(k=\min(n,m))$,因此该算法最坏情形的时间复杂度仍为 $o(2^m)$.

## 3　举例说明

求 $F=\{\{a,b,c,d\},\{a,b\},\{c,d\},\{a,c\},\{b,d\},\{b\}\}$ 的极小碰集(以下节点表示依次为 ElesInCOMPS, ParentEles, PruneSign, NoExpandSign).

1) 初始化:集合簇 $F$ 排序后变为$\{\{b\},\{a,b\},\{c,d\},\{a,c\},\{b,d\},\{a,b,c,d\}\}$,按照该次序生成 COMPS 集合为$\{b,a,c,d\}$,标记后,其二进制位表示为 $F=\{fmS_1,fmS_2,\cdots,fmS_n\}=\{0001,0011,1100,0110,1001,1111\}$.

2) $m=4,n=6,k=\min(n,m)=4$.

3) 生成第二层树节点,节点元素为单元素:{(0001, 0000, 1111, false), (0010, 0000, 1110, false), (0100, 0000, 1100, false), (1000, 0000, 1000, false)},依次加入 enumSetLink. 为表述方便,分别用 $p_{11},p_{12},p_{13},p_{14}$ 指代这 4 个节点.

4) $i=2\leqslant k$;调用 IsHS 逐一判断 enumSetLink 节点. $p_{11}$:IsHS 返回 2,将该节点加入 needExLink 后释放;$p_{12},p_{13}$ 和 $p_{14}$ 均由于 IsHS 返回 1,将这 3 个节点的 NoExpandSign 置为 true,分别加入 needExLink 后释放. 判断完成后,needExLink 链变为:{(0001,0000,1111,false),(0010,0000,1110,true),(0100, 0000, 1100, true),(1000, 0000, 1000,true)},用 $q_{11},q_{12},q_{13},q_{14}$ 指代.

5) needExLink 不为空,逐一判断并扩展:$q_{11}$ 的 NoExpandSign=false,需继续扩展. $q_{11}$ 和 $q_{12}$ 的

ParentEles 相同,通过取 $q_{12}$ 的并集,$q_{11}$ 扩展出的节点为:(0011, 0001, 1111, false),加入 enumSetLink. 同理,$q_{11}$ 通过取亲兄弟节点 $q_{13}$,$q_{14}$ 的并集,得到的扩展节点分别为:(0101,0001,1101, false),(1001,0001,1001,false),分别加入 enumSetLink 后释放 $q_{11}$. $q_{12}$:NoExpandSign = true,不需要扩展,释放. 同理 $q_{13}$,$q_{14}$ 都不需要扩展,分别释放. needExLink 变为空,enumSetLink 链变为只有 3 个节点,分别用 $p_{21}$,$p_{22}$,$p_{23}$ 来指代.

6) $i+1=3\leqslant k$. 现有 enumSetLink:$\{p_{21},p_{22},p_{23}\}$. $p_{21}$:IsHS 返回 2,将 $p_{21}$ 加入 needExLink 后释放. $p_{22}$:IsHS 返回 0,此时 pHS 为空,因而 $p_{22}$ 是极小碰集,直接加入 pHS 后释放. $p_{23}$:IsHS 返回 1,将该节点的 NoExpandSign 置为 true,加入 needExLink 后释放. 检查完毕,needExLink 中有 2 个节点:{(0011,0001,1111,false),(1001,0001,1001, true)},分别用 $q_{21}$,$q_{22}$ 指代.

7) needExLink 中有 2 个节点 $q_{21}$ 和 $q_{22}$,$q_{21}$ 扩展得节点:(1011,0011,1011,false),加入 enumSetLink 后释放 $q_{21}$. $q_{22}$ 不扩展,释放. 此时 enumSetLink 中节点用 $p_{31}$ 指代.

8) $i+1=4\leqslant k$,enumSetLink 中只有一个节点 $p_{31}$:IsHS 返回 0,此时 pHS 链中只有节点 $p_{22}$ 且不是碰集 $p_{31}$ 的子集,将 $p_{31}$ 加入 pHS,释放 $p_{31}$. 此时 needExLink 为空,不再扩展.

9) $i+1=5>k$,故算法终止. 所有极小碰集为 pHS 中的节点:$p_{22}$ 和 $p_{31}$;根据其 ElesInCOMPS 值,对应于枚举元素集合 COMPS = $\{b,a,c,d\}$,可得这两个节点元素分别为 $\{b,c\}$,$\{b,a,d\}$,容易验证它们就是所求集合簇 $F$ 的全体极小碰集.

改进算法执行过程如图 2 所示.

而 HSSE-tree 算法的执行过程如图 3 所示.

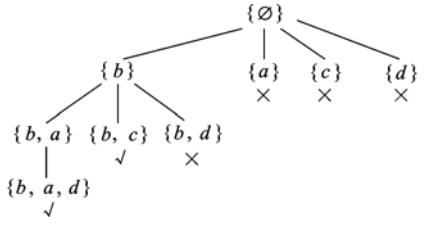


图 2 使用 Binary-label HSSE 算法求解所有 MHS

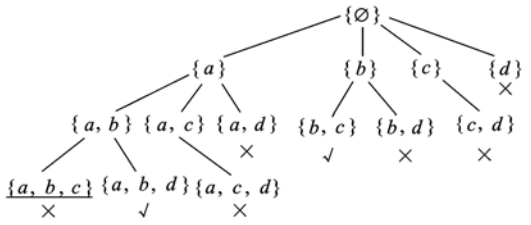


图 3 使用 HSSE-tree 求解所有 MHS

最终 Binary-label HSSE 只扩展了 8 个节点,且在扩展 $\{b, a\}$ 时避免了产生其右亲兄弟节点 $\{b, c\}$ 的超集;而 HSSE-tree 算法扩展的节点数有 13 个,包括一个非极小的碰集节点 $\{a, b, c\}$. 此外,由于改进算法在计算某节点是否碰集以及已是碰集的节点是否为极小时,将逐个元素比较改为二进制位与运算,使得算法的消耗时间及占用内存均有了大量减少.

## 4 统计分析

为进行算法效率的验证比较,首先假设集合簇 $F=\{S_i,i=1,2,\cdots,n\}$ 的集合数目 $n$ 与元素总数 $m$ 相等,并随机生成 $m=n=11,12,\cdots,35$ 的集合簇[9](每个集合中的元素个数不大于 $n$),然后在计算机(Intel(R) Core(TM)2 Duo CPU 2.66 GHz,1.95 GB 内存)Windows XP 操作系统下,采用 VC6.0 进行编程,来比较 HSSE-tree 与本文改进算法的计算效率.

随着状态空间增大,两种算法的运行时间(包括初始化时间如集合簇排序时间等)、扩展节点个数及扩展出的非极小碰集个数见表 1.(集合簇随机生成一次,算法各执行一次.)

表 1 运行时间的折线图如图 4 所示.

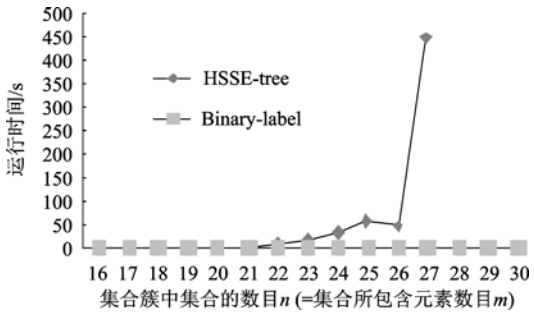


图 4 HSSE-tree 算法和改进算法的运行时间比较

由以上可知,改进算法明显抑制了当状态空间增大时,扩展节点数目及非极小的碰集个数的爆炸式增长,同时算法运行时间也有了大量减少. 特别是当状态空间大于 27 时,在当前的计算机内存上已无法计算 HSSE-tree,然而改进算法仍能以几百毫秒至几秒的时间,继续诊断大得多的状态空间. 这主要得益于以下几点改进:

1) 采用二进制位标记每个节点元素,而不是实际扩展出来. 这样做不止显著减少了 HSSE-tree 算法存储节点元素的内存,同时配合改进 4) 中对集合簇 $F$ 的二进制标记,使用二进制运算来计算是否碰集及是否极小碰集的超集,对于算法效率的提升是非常明显的.

2) 增加了推论 1 所述的剪枝规则. 该规则可以避免一些无用的扩展,特别是在集合簇 $F$ 中含有元素个数极少的集合时,配合改进 4)中对集合簇 $F$ 排序的操作,这种剪枝非常明显.

3) 改进了节点扩展方式,如推论 2 所述. 这样做可以避免产生已是碰集的亲兄弟节点的超集,这种超集在 HSSE-tree 算法所有最终被检查为超集的碰集中占了绝大部分.

4) 将集合簇 $F$ 按照元素个数从少到多的顺序排序,并用与标记节点相同的方式标记 $F$. 前者可使 1)所述剪枝规则更有效,后者将计算是否碰集改为二进制运算,提升了算法效率.

**表 1　两种算法执行性能对照表**

| $m,n$ | MHS 个数 | 运行时间/ms | | 总的扩展节点个数 | | 非极小的碰集个数 | |
|---|---|---|---|---|---|---|---|
| | | HSSE-tree | Binary-label HSSE | HSSE-tree | Binary-label HSSE | HSSE-tree | Binary-label HSSE |
| 16 | 28 | 78.214 721 | 0.407 773 | 23 323 | 139 | 3 994 | 4 |
| 17 | 21 | 244.041 975 | 0.700 702 | 65 535 | 256 | 25 099 | 0 |
| 18 | 60 | 130.607 065 | 3.018 796 | 36 565 | 926 | 2 544 | 77 |
| 19 | 42 | 599.517 623 | 5.128 005 | 162 857 | 1 391 | 4 976 | 128 |
| 20 | 44 | 1 182.936 8 | 2.418 337 | 318 027 | 785 | 20 160 | 51 |
| 21 | 63 | 3 608.295 5 | 2.437 518 | 882 233 | 808 | 4 893 | 31 |
| 22 | 46 | 8 342.266 44 | 3.237 082 | 1 643 327 | 1123 | 151 362 | 51 |
| 23 | 36 | 19 696.269 6 | 1.566 006 | 3 820 671 | 557 | 197 116 | 18 |
| 24 | 108 | 33 163.040 8 | 13.575 417 | 6 106 463 | 4 183 | 24 724 | 302 |
| 25 | 1024 | 56 197.464 8 | 1 565.660 9 | 4 353 735 | 270020 | 927 636 | 18 342 |
| 26 | 661 | 44 650.682 8 | 353.260 72 | 7 140 813 | 74 530 | 253 981 | 5 816 |
| 27 | 680 | 445 102.134 | 494.204 6 | 55 585 599 | 109 968 | 7 364 270 | 7 706 |
| 28 | 560 | — | 108.315 26 | — | 132 854 | — | 5 550 |
| 29 | 468 | — | 55.590 033 | — | 65 902 | — | 5 730 |
| … | | | | | | | |

综上所述,改进算法主要针对 HSSE-tree 算法的诸多缺点进行改进,因此除了像 HSSE-tree 算法一样特别适合于计算单双故障中所有的极小碰集之外,在集合簇中含有元素数目极少的集合时,相对于 HSSE-tree 算法更能体现出其优势.

## 5　结　论

本文提出一种计算全体极小碰集的 HSSE-tree 算法的改进算法. 仿真结果显示,计算随机生成的冲突集簇时,改进算法的运行时间及扩展节点数目均大大优于 HSSE-tree 算法,算法效率有了明显提高. 航天器系统部件数量众多且内存资源有限,本文提出的改进算法以其算法运行时间及占用内存上的优势,为实现大规模系统的故障诊断及实时诊断提供了理论依据和应用基础.

### 参考文献 (References)


[1] Hayden S, Sweet A, Shulman S. Lessons learned in the livingstone 2 on earth observing one flight experiment[C]//Proc AIAA 1st Intelligent Systems Tech Conf. Arlington, Virginia: Infotech@ Aerospace, 2005: 1-15

[2] Narasimhan S, Brownston L. HyDE-a general framework for stochastic and hybrid model-based diagnosis [C]//Proc of the 18th Int Workshop on Principles of Diagnosis. Nashville: TN, 2007: 162-169

[3] Hamscher W C. Modeling digital circuits for troubleshooting[J]. ArtificialIntelligence, 1991, 51(13): 223-272

[4] 赵相福. 基于模型诊断中的关键算法研究[D]. 长春:吉林大学计算机科学与技术学院,2006
Zhao Xiangfu. Research on key algorithms in model-based diagnosis[D]. Changchun: College of Computer Science and Technology, Jilin University, 2006(in Chinese)

[5] Reiter R. A theory of diagnosis from first principles[J]. Artificial Intelligence, 1987, 32(1): 57-96

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

[7] 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 Yunfei, Lin Li. The computation of hitting sets with boolean formulas[J]. Chinese Journal of Computers, 2003, 26(8): 919-924(in Chinese)

[8] Rymon R. Search through systematic set enumeration[C]//Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning. Cambridge, MA: Morgan Kaufmann, 1992, 539-550

[9] 陈晓梅,孟晓风,乔仁晓. 基于 BNB-HSSE 计算全体碰集的方法[J]. 仪器仪表学报,2010,31(1):61-67
Chen Xiaomei, Meng Xiaofeng, Qiao Renxiao. Method of computing all minimal hitting set based on BNB-HSSE[J]. Chinese Journal of Science Instrument, 2010, 31(1): 61-67(in Chinese)


(编　辑:娄　嘉)