用对分 HS-树计算最小碰集*

姜云飞[1], 林 笠[1,2]

[1](中山大学 软件研究所,广东 广州 510275);
[2](暨南大学 数学系,广东 广州 510632)
E-mail: lncsri05@zsu.edu.cn; douglass@public.guangzhou.gd.cn
http://linlionline.126.com

摘要: 在基于模型的诊断中,利用冲突集计算最小碰集是其关键的步骤,因为所有冲突集的最小碰集就是所考察系统的诊断.在 Reiter 的方法中,要用 HS-树(图)来计算最小冲突集的最小碰集.HS-树的计算量比较大,且又会因为剪枝的问题而剪掉真实解.提出了用对分 HS-树(binary hitting set-树,简称 BHS-树)计算最小碰集的方法.这种方法的优点是:(1) 产生的树的节点数明显少于 HS-树,因而效率较高;(2) 解决了因为剪枝而产生的最小碰集丢失的问题;(3) 在新增加冲突集时不必完全重新计算,只需在原 BHS-树的基础上增加新的分支即可,这种性质对实际诊断问题是特别有用的.对利用 BHS-树的算法从理论上进行了分析和论证,并通过实际编写程序进行了检验.

关 键 词: 基于模型诊断;最小冲突集;最小碰集;对分 HS-树
中图法分类号: TP18 文献标识码: A

基于模型的诊断(model-based diagnosis)[1~3]是人工智能领域近年来发展起来的一个十分活跃的研究分支.基于模型的诊断的主要思想是,根据系统的组成元件和元件之间的连接建立起系统模型,这种模型通常用一阶逻辑语句来描述.根据系统的逻辑模型以及系统的输入,通过逻辑的推理理论,人们可推导出系统在正常情况下预期的行为,如果系统的实际行为与系统预期的行为有差别(discrepancy),就说明系统存在故障.利用逻辑的推理理论,我们能够确定引发故障的元件集合,这就是诊断.基于模型的诊断一般由 3 个步骤组成,即诊断的产生(generation)、诊断的测试(test)和诊断的鉴别(discrimination).诊断的产生是根据系统实际行为与预期行为的差别,利用系统模型,通过逻辑推导得出引发故障的元件集合.在通常情况下,这种元件集合有多个.诊断的测试则是用推导排除那些不合理的集合,仅留下那些能够解释实际观察的集合.如果留下的集合仍有多个(通常情况下是如此),则再通过诊断的鉴别,使用附加的信息去鉴别这些诊断.

所以,要想提高诊断的效率,便可以通过提高计算冲突集或碰集的效率来实现,现在,有许多改进的计算碰集算法[4~7],可以提高计算碰集的效率.

1 预备知识

下面介绍一些基于模型诊断的相关概念和定理[3].

定义 **1**. 一个系统定义为一个三元组(*SD*,*COMPS*,*OBS*),其中

· *SD* 是系统描述,是一阶谓词公式的集合;

· *COMPS* 为系统的组成部件,是一个有限的常量集;

* 收稿日期: 2001-03-13; 修改日期: 2001-05-15
基金项目: 国家自然科学基金资助项目(69873047;60173039);广东省自然科学基金资助项目(980260)

作者简介: 姜云飞(1945－),男,吉林长春人,教授,博士生导师,主要研究领域为智能诊断,智能规划;林笠(1968－),男,吉林敦化人,博士,讲师,主要研究领域为智能诊断,计算机图形学.

· *OBS* 为观测集,是一阶谓词公式的有限集.

一元谓词 *AB* 意味着“abnormal”(反常),当部件 $c\in COMPS$ 反常时,*AB*(*c*)为真.有些系统用~*OK*(·)代替 *AB*(·).

定义 **2**. 冲突集(*CS*)是一个部件集$\{c_1,c_2,\ldots,c_n\}\subseteq COMPS$,使得

$SD\cup OBS\cup\{\sim AB(c_1),\sim AB(c_2),\ldots,\sim AB(c_n)\}$不一致,冲突集也称为冲突部件集,记为⟨·⟩;若冲突集的任意子集都不是冲突集,则称该冲突集为最小冲突集(MCS).

定义 **3**. 设 *C* 是集合簇(set cluster,由集合组成的集合).*C* 的一个碰集(HS)为 $H\subseteq\cup_{s\in C}S$,使得对每一个 $S\in C$ 都有 $H\cap S\neq\varnothing$,记为[·].称 *C* 的一个碰集是最小的(MHS)当且仅当它的任意子集都不是碰集.

定理 **1**. $D(\Delta,COMPS-\Delta)$是(*SD*,*COMPS*,*OBS*)的一个基于一致性的诊断,当且仅当Δ是(*SD*,*COMPS*,*OBS*)的最小冲突集的最小碰集.

由定理 1 可知,若知道了全部最小冲突集,则可以通过计算出最小冲突集的最小碰集得到诊断.Reiter 提出了用 HS-树来计算,但是某些情况下会因为剪枝丢失最小碰集.Greiner 等人给出了修改算法 HS-DAG 图,而不是 HS-树.

这两种方法构造的 HS-树(图),树(图)的节点一般比较多.见例 1.

例 1:*CS*={⟨2,4,5⟩,⟨1,2,3⟩,⟨1,3,5⟩,⟨2,4,6⟩,⟨2,4⟩,⟨2,3,5⟩,⟨1,6⟩},计算 *CS* 的最小碰集是:[1,2],[2,3,6],[2,5,6],[1,3,4],[1,4,5],[3,4,6].

用 Reiter 的算法构造的 HS-树如图 1 所示.其中,用√标记出计算最小碰集所用到的终端节点,从根节点到此终端节点的边上标记的所有元素构成最小碰集,“)(”为剪枝的边.

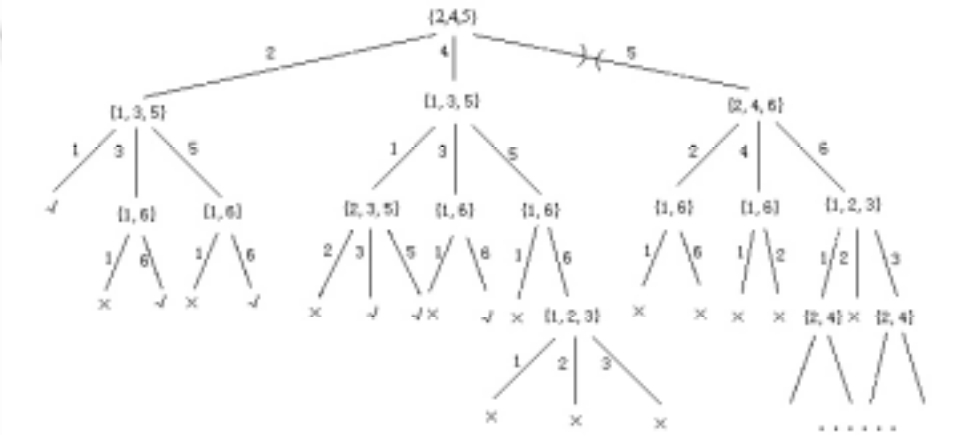


Fig.1 The HS-tree of conflict sets {⟨2,4,5⟩,⟨1,2,3⟩,⟨1,3,5⟩,⟨2,4,6⟩,⟨2,4⟩,⟨2,3,5⟩,⟨1,6⟩}

图 1 冲突集为{⟨2,4,5⟩,⟨1,2,3⟩,⟨1,3,5⟩,⟨2,4,6⟩,⟨2,4⟩,⟨2,3,5⟩,⟨1,6⟩}时的 HS-树

2 用 BHS-树计算最小碰集

定义 **4**(**BHS**-树). 给定最小冲突集 $MCS=\{C_1,C_2,\ldots,C_n\}$,BHS-树是如下递归定义的二叉树,每个节点的数据是两个集合簇,记为 *C* 和 *H*.根节点的 *C*=*MCS*,*H*={}.左、右子树根节点的数据分别记作 C_l,H_l 和 C_r,H_r.

(1) 若 *C*={},则 BHS-树就是该节点本身;

(2) 否则,任取一元素 $a\in\cup C_i$,则 $C_l=\{C_i-\{a\}|a\in C_i\}$,$H_l=\{a\}$和 $C_r=\{C_i|a\notin C_i\}$,$H_r=\{\}$.

BHS-树节点的 *C* 记为⟨·⟩,*H* 记为[·].如果初始给定的冲突集 CS 不是最小的,则要将其变为最小冲突集 MCS,即将某个冲突集的超集全部删除.

例 1 的 BHS-树如图 2 所示.其中,BHS-树边上的值即为每次任取的元素 *a*,用⟨·⟩表示 *C*,用[·]表示 *H*,右边有“*”的集合表示被删除的集合,⟨ ⟩表示 *C*=∅,[ ]表示 *H*=∅,边上的标记表示每次任取的元素 $a\in\cup C_i$.

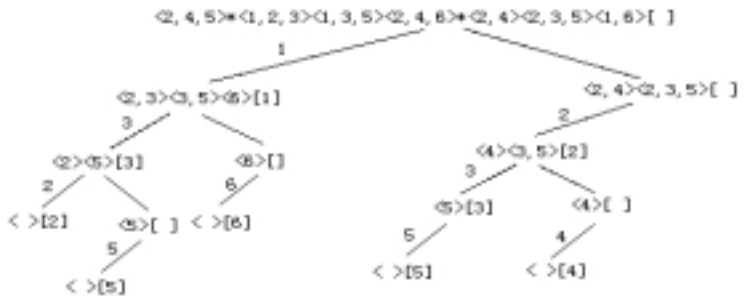


Fig.2 A BHS-tree of {〈2,4,5〉,〈1,2,3〉,〈1,3,5〉,〈2,4,6〉,〈2,4〉,〈2,3,5〉,〈1,6〉}

图 2 {〈2,4,5〉,〈1,2,3〉,〈1,3,5〉,〈2,4,6〉,〈2,4〉,〈2,3,5〉,〈1,6〉}的 BHS-树

已知最小冲突集 MCS,我们可以通过算法 1 来计算它的最小碰集 MHS.

算法 **1**. 利用 BHS-树递归计算最小冲突集 MCS 的最小碰集 MHS.

对 BHS-树的每一个节点,建立该节点的 BHS-树的最小冲突集的候选集簇 M(真正的最小冲突集簇比这个候选集要小,只要删除其他集合的超集,就可以得到最小冲突集簇),然后按照如下算法从叶节点出发自底向上递归计算.

(1) 若 BHS-树本身就是叶节点(即 C 为空集),则该 BHS-树的最小冲突集的候选集 $M=H$;否则自底向上递归计算(2)、(3)至根节点,

(2) BHS-树包含左子树和右子树,$M=\{H,\{m_l\cup m_r|m_l\in M_l,m_r\in M_r\}\}$,$H$ 与 M 取自同一个节点,其中 M_l 为左子树的候选集,M_r 为右子树的候选集.

(3) 将 BHS-树根节点的 M 最小化,删除覆盖其他集合的集合.根节点的 M 即为所要求的最小碰集 MHS.

例 2 (继续例 1):利用 BHS-树计算最小碰集,生成 BHS-树如图 2 所示.利用 BHS-树递归计算最小碰集,如图 3 所示.其中,[·]表示 M.右边有“*”表示删除,“→”表示递归计算 M 时的顺序,从终端节点开始.

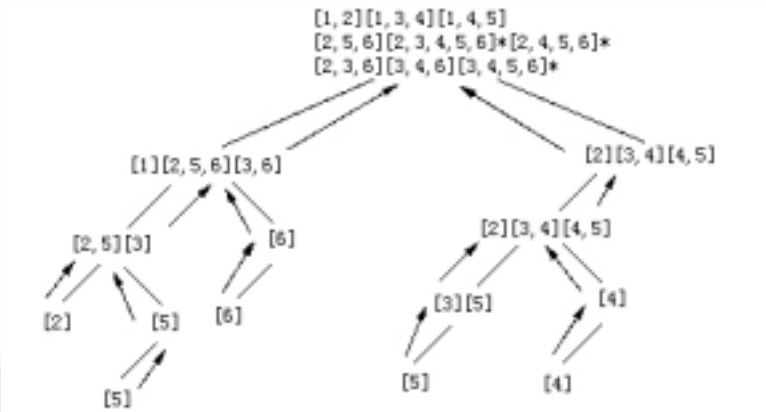


Fig.3 The minimal hitting sets computed by M recursively

图 3 利用 M 递归计算最小碰集

用 BHS-树算法得到的最小碰集是:[1,2],[2,3,6],[2,5,6],[1,3,4],[1,4,5],[3,4,6].与 Reiter 方法的结果一致.

定理 **2**. 如果最小冲突集 $MCS=\{C_1,C_2,...,C_n\}$,用算法 1 得到的 M 一定是最小冲突集 MCS 的最小碰集 MHS.

证明:对 BHS-树的深度 k 用数学归纳法.

(1) 当 k=1 时,即 BHS-树深度=1,这样的情况只有一种:最小冲突集和最小碰集都是空集,显然成立.

(2) 当 k=2 时,即最小冲突集只有一个集合,且是单元素集合,不妨设 $CS=\langle a\rangle$,则根节点 $C=\langle a\rangle$,$H=[\ ]$,由算法 1 可得:左子树 $C_l=\langle\ \rangle$,$H_l=[a]$,右子树为 $C_r=\langle\ \rangle$,$H_r=[\ ]$,得到的最小碰集为 $M=\{H,\{a\cup\varnothing\}\}=\{\varnothing,\{a\}\}=\{a\}$,假设成立.

(3) 假设当 $k\le n$ 时结论成立,则当 $k=n+1$ 时:

假设 BHS-树根节点 $C=\{C_1,C_2,...,C_n\}$,则 $M=\{H,\{m_l\cup m_r|m_l\in M_l,m_r\in M_r\}\}$,($m_l,m_r$ 是集合,M_l,M_r 是集合的集合,且已将 M 中为其他集合超集的集合删除).

任取 $a\in\cup C_i$,可以把 C 分成两个不相交的划分:Π_l,Π_r:

$$\Pi_l=\{C_i|a\in C_i\}.\Pi_r=\{C_i|a\notin C_i\}.$$

显然$\Pi_l\cap\Pi_r=\varnothing,C=\Pi_l\cup\Pi_r$.

构造左子树,根节点 $C_l=\{C_i-\{a\}|C_i\in\Pi_l\},H_l=\{a\}$.构造右子树,根节点 $C_r=\{C_i|C_i\in\Pi_r\},H_r=\varnothing$.显然左右子树的深度≤$n$,所以 M_l 是 C_l 的最小碰集,M_r 是 C_r 的最小碰集,则 $M=\{H,m_l\cup m_r|m_l\in M_l,m_r\in M_r\}$是 $C_l\cup C_r=C$ 的最小碰集.即 BHS-树根节点的 M 即是 CS 的最小碰集 MHS. □

说明:(1) 用 BHS-树计算最小碰集不必进行剪枝,因而不会丢失正确解.

(2) 在建立 BHS-树时,由根节点的 C 和 a 决定它的左、右子树根节点的 C_l,H_l 与 C_r,H_r,与根节点的 H 无关.取不同的 a 值,左、右子树根节点的 C_l,H_l 与 C_r,H_r 会有所不同,但不影响最终结果的正确性.

(3) 在建立 BHS-树时,右子树根节点的 $H_r=\varnothing$,左子树根节点的 H_l 是单元素单集合$\{a\}$.

(4) 建立 BHS-树时,终端节点的 $C=\varnothing,H$ 为单元素单集合或为$\varnothing$.

(5) 递归计算根节点的 M 时,由建立 BHS-树时得到的 H 及左、右子树根节点的 M_l,M_r 决定,与 C,C_l,C_r 无关,所以,得到 BHS-树之后,可以不再保留每个节点的 C.

(6) 该算法不能直接产生出最小碰集,但能产生全部最小碰集,需要将得到的碰集最小化,即删除为其他集合的超集的集合.

例 3:进行最小化处理,计算⟨1,2,4⟩,⟨1,3,4⟩的最小碰集,如图 4 所示.

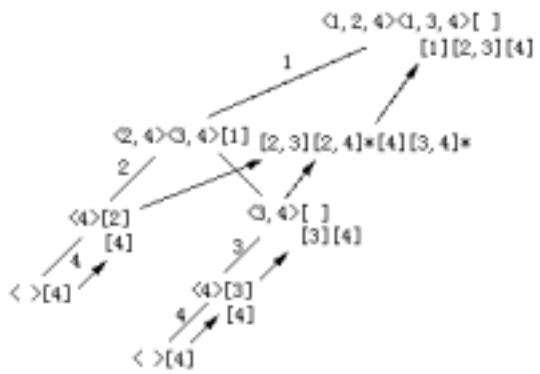


Fig.4 The BHS-tree of {⟨1,2,3⟩⟨1,3,4⟩}, the results are minimalized, "*" stands for deleted

图 4 {⟨1,2,3⟩⟨1,3,4⟩}的 BHS-树,对结果进行最小化处理,有"*"表示删除

设最小冲突集合簇 MCS 共有 n 个集合,记为$\{C_1,C_2,...,C_n\}$,第 i 个冲突集 C_i 含有 m_i 个元素,$m_a=\left(\sum_i^n m_i\right)\Big/n$,冲突集含元素个数最多为 $m_\lambda=\max(m_i)$,则建立的 BHS-树有以下性质:

性质 **1**. BHS-树的深度(depth)≤$|\cup C_i|+1,i=1,2,...$其中 C_i 是冲突集,$|\cdot|$是集合中元素的个数.

任取$\cup C_i$ 中的一个元素 a,按算法 1 构造 BHS-树,显然,左子树与右子树根节点的 C 均不含元素 a,所以,左、右子树根节点的$|\cup C_l|,|\cup C_r|\le|\cup C_i|-1$,所以,BHS-树的深度≤$|\cup C_i|+1$.

性质 **2**. BHS-树的深度≤$n+m_\lambda+1$.

因为每建立一层子树,子树根节点 C 中的集合个数就比根节点的集合个数少 1,或子树根节点集合 C 中的元素个数比根节点 C 中的元素个数少 1.终端节点的 C 中集合个数是 1,所含元素个数为 0,所以 BHS-树的深度≤$n+m_\lambda+1$.

由性质 1 和性质 2 易得性质 3.

性质 **3**. BHS-树的深度≤$\min(|\cup C_i|+1,n+m_\lambda+1)$.

性质 **4**. BHS-树的终端节点个数=n.则左子树的集合个数+右子树的集合个数=根节点的集合个数,左子树

的空集记集合个数为 1,右子树的空集不记数.所以终端节点个数=n.

性质 5. BHS-树的节点总数≤$2\times(\sum|C_i|)-n+1=2m_an-n+1=(2m_a-1)n+1$.

任意消去 BHS-树根节点 C 的元素 a,最多可生成两个新的节点,左子树根节点由原来含 a 的集合构成,右子树根节点由原来不含 a 的集合构成.且在生成终端节点时,只有左子树,没有右子树(少 n 个右子树).所以,BHS-树的节点总数≤$2\times(\sum|C_i|)-n+1=(2m_a-1)n+1$.

由以上性质可知,该算法可以在有限步骤内完成,且可知 BHS-树的空间复杂度 $O(nm)$,而 HS-树的空间复杂度可能为 $O(m^n)$.时间复杂度与空间复杂度成正比关系,当然,它们还与集合中元素的具体取值情况有关.

3 用 BHS-树计算导出冲突集的最小碰集

在诊断系统中,得到的诊断有时不满能足要求,需要增加探测(probe),得到新的冲突集,再重新计算新的最小碰集.用 BHS-树计算新的最小碰集,无须对原 BHS-树作任何改动,只需在原 BHS-树的基础上,增加新的分支即可.

算法 2. 利用导出最小冲突集 $D_i(i=1,2,...,n)$与原 BHS-树,构造新 BHS-树来计算新的最小碰集.

步骤如下:

(1) 用算法 1 根据全部导出最小冲突集 D_i,构造一棵新 BHS-树,树根 $C=\{D_1,D_2,...,D_n\}$;

(2) 用算法 1 计算导出 BHS-树根节点的 M.将新 BHS-树作为左子树,原 BHS-树作为右子树,合并成一棵新的 BHS-树,用算法 1 计算出新的 BHS-树根节点的 M 即为所求最小碰集.

例 4(继续例 3):在原有冲突集合簇的情况下增加一个新集合⟨5,6⟩.生成的 BHS-树如图 5 所示,方框中的部分是原 BHS-树,不必重新计算,⟨·⟩表示 C,[·]表示 H,边上的标记表示每次任取的元素 $a\in\cup C_i$.递归生成最小碰集如图 6 所示,增加⟨5,6⟩之后的最小碰集是:[1,2,5],[1,4,5],[2,5,6],[1,2,6],[2,3,6],[3,4,6].

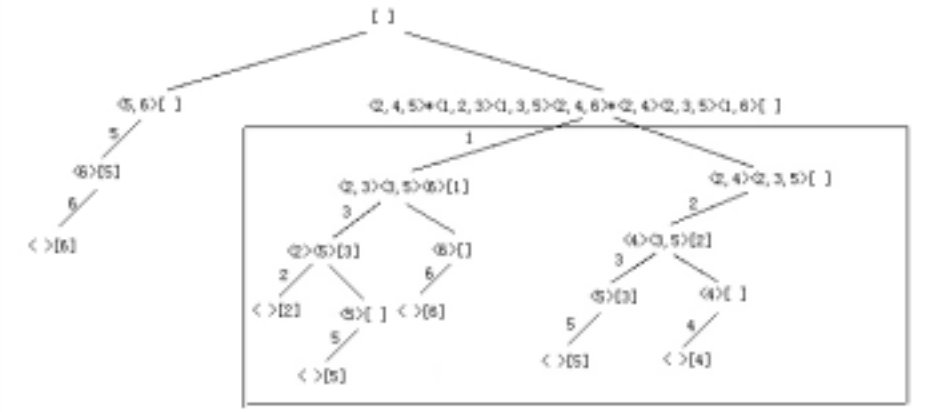


Fig.5 The BHS-tree after inserted ⟨5,6⟩

图 5 增加⟨5,6⟩之后的 BHS-树

定理 3. 导出的最小冲突集 $DS=\{D_i\}(i=1,2,...)$构成的 BHS-树根节点 M_l,原最小冲突集 $CS=\{C_j\}(j=1,2,...)$构成的 BHS-树根节点是 M_r,$M=\{m_l\cup m_r|m_l\in M_l,m_r\in M_r\}$是由 DS 和原冲突集 CS 构成的冲突集{CS,DS}的最小碰集.

证明:根据定理 2,M_l 是 DS 的最小碰集,M_r 是 CS 的最小碰集,所以,由 BHS-树算法可知,BHS-树根节点 $C=\{DS,CS\}$,$H=\varnothing$,$M=\{m_l\cup m_r\ |m_l\in M_l,m_r\in M_r\}$是$\{DS,CS\}$的最小碰集.即由算法 2 得到的最小碰集一定是所求最小碰集. □

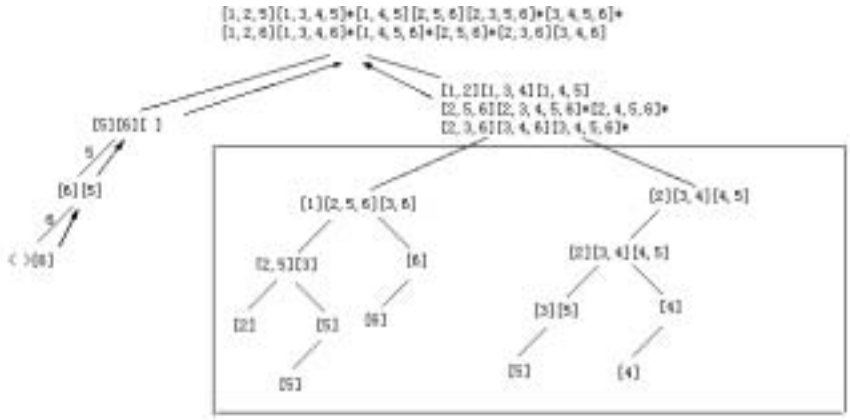


Fig.6 The hitting sets after inserted ⟨5,6⟩. "*" stands for deleted,[·] stands for M

图 6 增加⟨5,6⟩之后的递归计算最小碰集.右边有"*"号表示删除,[·]表示 M

4 比较与结论

(1) 在生成 BHS-树时,因为每生成一层,则$|\cup C_i|-1$,即$|\cup C_i|$少一个元素,而$|\cup C_i|\le nm$ (n 是最小冲突集的个数,m 是冲突集平均元素个数),即 BHS-树是一个深度$\le nm+1$ 的二叉树,所以该算法一定能够停止,即 BHS-树能够在有限步骤内建立.且在多数情况下,BHS-树的空间复杂度优于 HS-树(BHS-树空间复杂度为 $O(nm)$,HS-树的空间度为 $O(m^n)$).

(2) 在建立 BHS-树时,不必进行剪枝,因而不会丢失解.

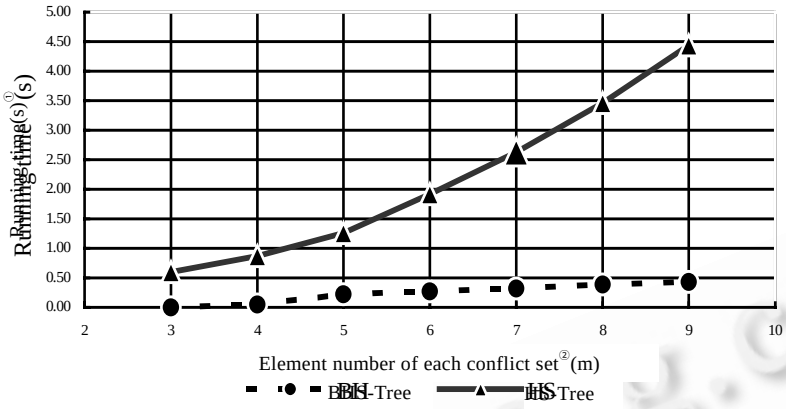
(3) 计算碰集时,不能直接得到"最小"碰集,这样需要在根节点处将最终得到的碰集进行最小化,删除某些碰集的超集，或在每个中间节点处,将得到的中间结果进行比较,删除"非最小碰集",这样可以提高计算效率.最小冲突集个数均为 n=9,取值为⟨1,2,...,m⟩⟨2,3,...,m+1⟩,...,⟨n,n+1,...,n+m−1⟩(Intel Celeron 667M,128M 内存,中文 Windows98,Borland C++ Builder 5.5).见表 1 和图 7.

Table 1 Run-Time comparison between BHS-tree and HS-tree methods

表 **1** BHS-树与 HS-树运行时间比较

Number of conflict sets (n)①	Element number of each conflict set (m)②	Node number of BHS-tree③	Running time of BHS-tree (s)④	Node number of HS-tree⑤	Running time of HS-tree (s)⑥
9	3	36	<0.01	531	0.60
9	4	45	0.05	896	0.87
9	5	54	0.22	1220	1.26
9	6	63	0.27	1512	1.92
9	7	72	0.32	1785	2.63
9	8	81	0.39	2048	3.46
9	9	90	0.43	2304	4.44

①最小冲突集包含集合个数(n),②每个冲突集均含元素个数(m),③BHS-树节点个数,
④BHS-树运行时间(秒),⑤HS-树节点个数,⑥HS-树运行时间(秒).



①运行时间(秒),②每个冲突集包含的元素个数.

Fig.7 Run-Time comparison between BHS-tree and HS-tree methods

图 7 BHS-树与 HS-树运行时间比较图

说明:(1) 如果冲突集不变,仅是排列次序不同,当用 BHS-树或 HS-树计算时,所花费的时间及树的总节点数都可能会有很大的不同.当 n=9,m=8 时,把前 8 个集合分成 4 对,相邻的两个集合互换先后位置,其他都不改变,则结点数或运行时间都有可能不同.换位置前/换位置后 BHS-树的节点数为 81/77,运行时间为 0.39/0.22 秒;HS-树的节点数为 2048/7880,运行时间为 3.46/4.28.

(2) 节点数量与冲突集的元素个数有关,也与冲突集的交集的元素个数有关,若冲突集的交集元素较多,则 BHS-树的节点数量相对较少.

(3) 运行时间不但与节点数量有关,同时也与 BHS-树的深度有关.

(4) 在某些特殊情况下,HS-树的节点数量可能会更少,例如:当最小冲突集是单集合单元素时,BHS-树有 3 个节点,而 HS-树只有 2 个节点.

Franz Wotawa[8]提出的 HST-树的算法是改进的 HS-树.在建立 HST-树之前,需要对冲突集中的元素进行比较、判定,选择性地生成子树节点,总的节点数比 HS-树要少.所以,该算法可以提高计算效率,但仍需要剪枝,且插入新的冲突集需要重新建立 HST-树.而 BHS-树算法则在新插入冲突集时不必重新建立 BHS-树,这一点在实际诊断问题中是非常有效的.

综上所述,用 BHS-树计算最小碰集,在空间复杂性与时间复杂性上一般都比较好.同时,也有较好的通用性.

References:

[1] de Kleer, J., Mackworth, A.K., Reiter, R. Characterizing diagnoses and systems. Artificial Intelligence, 1992,56(2-3):197~222.

[2] de Kleer, J., Williams, B.C. Diagnosing multiple faults. Artificial Intelligence, 1987,32(1):97~130.

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

[4] Greiner, R, Smith, B.A, Wilkerson, R.W. A correction to the algorithm in Reiter's theory of diagnosis (research note). Artificial Intelligence, 1989,41(1):79~88.

[5] Lin, Li, Jiang, Yun-fei. Computing minimal hitting sets with Genetic Algorithm. In: Proceedings of the 13th International Workshop on Principles of Diagnosis. 2002. 77~80.

[6] Benjamin, H., Lee, Shie-Jue. Deriving minimal conflict sets by CS-tree with mark set in diagnosis from first principles. IEEE Transactions on System, Man and Cybernetics—Part B: Cybernetics, 1999,29(2):281~286.

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

Computing the Minimal Hitting Sets with Binary HS-Tree *

JIANG Yun-fei[1], LIN Li[1,2]

[1](*Institute of Software*, *Zhongshan University*, *Guangzhou* 510275, *China*);

[2](*Department of Mathematics*, *Jinan University*, *Guangzhou* 510632, *China*)

E-mail: lncsri05@zsu.edu.cn; douglass@public.guangzhou.gd.cn

http://linlionline.126.com

Abstract: In the model-based diagnosis, a key step is to compute the minimal hitting sets from the minimal conflict sets, because the minimal hitting sets of all the conflict sets are the faults of an investigated system. In Reiter's method, HS-tree is used for computing the minimal hitting sets. However, it will need a heavy work, and will result in the fact that the correct hitting sets are probably pruned away. In this paper, a new method is put forword for computing minimal hitting sets by the "binary hitting set tree", in brief, BHS-tree. This method will improve Reiter's approaches in the aspects as follows. (1) The number of BHS-tree nodes is apparently less than that of HS-tree; (2) As a result of being pruned, the loss of the minimal hitting sets will not produce; (3) When the new conflict sets are inserted, it is unnecessary to rebuild BHS-tree, instead, just adding the new branches to the BHS-tree. This property is extraordinarily useful for the actual diagnosis. Thus, the BHS-tree method is analyzed and verified in theory and the given programs are tested.

Key words: model-based diagnosis; minimal conflict set; minimal hitting set; binary HS-tree

---

* Received March 13, 2001; accepted May 15, 2001

Supported by the National Natural Science Foundation of China under Grant Nos.69873047, 60173039; the Natural Science Foundation of Guangdong Province of China under Grant No.980260