计算机研究与发展 DOI:10.7544/issn1000-1239.2016.20150396
Journal of Computer Research and Development 53(11):2556-2566,2016

结合 SE-Tree 结构特征的极小碰集求解算法

刘思光 欧阳丹彤 王艺源 贾凤雨 张立明
(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)
(lsgmliss@163.com)

Algorithm of Computing Minimal Hitting Set Based on the Structural Feature of SE-Tree

Liu Siguang, Ouyang Dantong, Wang Yiyuan, Jia Fengyu, and Zhang Liming
(*College of Computer Science and Technology, Jilin University, Changchun* 130012)
(*Key Laboratory of Symbolic Computation and Knowledge Engineering (Jilin University), Ministry of Education, Changchun* 130012)

Abstract During the process of computing minimal hitting set (MHS) by SE-Tree, it will generate many redundant nodes that cannot be pruned by current SE-Tree based algorithms, which affects the efficiency of these algorithms, i.e., the higher the ratio of redundant nodes is, the greater likely the impact of algorithms has. In this paper, firstly we introduce the definition of redundant nodes by analyzing the characteristic of leaf-node in SE-Tree and the redundant nodes in solution space in existent algorithms. Secondly, on the basis of analyzing the structural feature of SE-Tree and the theory that the subset of non-hitting set is non-hitting set, we propose the concept of assistant pruning tree. Specially, the decision nodes are added into the assistant pruning tree, which can largely reduce the visit of non-solution space. Furthermore, in order to decrease the visit of non-solution space when computing larger problem as much as possible, the algorithm of computing minimal hitting set combining with multi-level assistant pruning tree is proposed. Finally, we set a reasonable termination condition to make our algorithm stop without losing correct solution as early as possible, and then prove its correctness. Results show that the proposed algorithm is easily implemented and efficient. Moreover, our algorithm significantly outperforms a state-of-the-art minimal hitting set algorithm Boolean, even up to one order of magnitude for some instances.

Key words model-based diagnosis; minimal hitting set; SE-Tree; assistant pruning tree; non-solution space pruning

摘 要 在结合 SE-Tree 计算集合簇极小碰集的过程中,现有算法会对大量不会产生碰集的冗余节点进行访问.这无疑将影响算法的效率,冗余节点比例越高,影响越大.通过对 SE-Tree 中叶节点的特殊性质的分析,并结合现有碰集算法有解空间中冗余节点的特征,提出非解冗余节点概念.在对 SE-Tree 的

收稿日期:2015-05-28;**修回日期**:2016-02-16
基金项目:国家自然科学基金项目(61133011,61402196,61272208,61003101,61170092);中国博士后科学基金项目(2013M541302);吉林省科技发展计划基金项目(20140520067JH);浙江师范大学计算机软件与理论省级重中之重学科开放基金项目(ZSDZZZZXK12)
This work was supported by the National Natural Science Foundation of China (61133011, 61402196, 61272208, 61003101, 61170092), the China Postdoctoral Science Foundation (2013M541302), the Science and Technology Development Program of Jilin Province (20140520067JH), and the Provincial Key Disciplines Foundation of Computer Software and Theory of Zhejiang Normal University (ZSDZZZZXK12).
通信作者:张立明(limingzhang@jlu.edu.cn)

结构特征进行深入分析基础上,根据非碰集的子集也不是碰集的特点,提出辅助剪枝的概念,通过在剪枝树上设置剪枝判定节点,减少对极小碰集求解过程中无解空间的访问;针对较大规模问题,还提出结合多级辅助剪枝树的极小碰集求解算法,进而较大程度地减少对非解冗余节点的访问;根据多级辅助剪枝树及 SE-Tree 的结构特征,给出提前终止算法的判定条件,并证明了此算法的正确性. 实验结果表明:与效率较高的 Boolean 算法相比,该算法高效且易于实现,尤其是对规模较大的问题,效率能提升 1 个数量级.

关键词 基于模型诊断;极小碰集;集合枚举树;辅助剪枝树;无解空间剪枝

中图法分类号 TP18

极小碰集问题是人工智能领域的研究热点之一,许多实际和理论问题都可以转化为极小碰集问题. 例如:基于模型诊断(model-based diagnosis)[1]过程一般可以分为冲突识别和候选产生 2 个主要步骤,冲突识别为根据系统描述及观测得到极小冲突集,候选产生则是利用得到的极小冲突集计算极小碰集;极小集合覆盖和极小顶点覆盖等极小覆盖问题可以看作是极小碰集问题的特殊形式;智能规划和软件调试中的核心问题也可以通过转换为极小碰集问题后使用相应算法求解来提高效率[2-3].

迄今为止,国内外许多学者都对极小碰集的求解算法进行了研究和改进. 最早的极小碰集计算算法是由 Reiter[1]于 1987 年提出的 HS-Tree 算法,但此算法可能会因剪枝而丢失正确解;为解决此问题,1989 年 Greiner 等人[4]对此算法进行改进,提出了 HS-DAG 算法;随着对极小碰集问题的深入研究,许多新的求解算法不断被提出:2001 年 Wotawa[5]通过对 HS-Tree 算法的改进提出了 HST-Tree 算法,有效降低了极小碰集求解过程中子集检测的数量;2002 和 2003 年,姜云飞等人提出了 BHS-Tree[6]和 Boolean 算法[7],前者对比 HS-Tree 算法有效减少了节点生成的数量同时解决了 HS-Tree 算法可能会因剪枝而丢解的问题,后者使用 Boolean 变量代表冲突集元素,通过 Boolean 表达式计算求解所有极小碰集,在提高求解效率的同时简化了数据结构;2004 年,欧阳丹彤等人[8]在 HS-DAG 算法的基础上提出了改进后的 New HS-Tree 算法,对比 HS-DAG 算法,搜索节点大幅减少;2006 年,赵相福等人[9]提出了 HSSE-Tree 算法,使用带有终止节点的集合枚举树形式化地表示了极小碰集的求解过程;2010 年,陈晓梅等人[10]提出了 BNB-HSSE 算法,通过使用分支定界法与集合枚举法相结合将问题不断分解,降低求解规模、简化枚举过程,进而提高求解效率;2011 年,张立明等人[11]提出了基于动态极大度的 DMDSE-Tree 算法,该算法通过在扩展集合枚举树(SE-Tree)节点的过程中使用自定义度对待扩充元素进行排序,有效降低了生成节点的数量;2012 年,Pill 等人[12]对 Boolean 算法进行了改进,通过对部分规则进行修改,有效提升了该算法对有界问题的求解效率;2015 年王艺源等人[13]提出了利用 CSP 求解极小碰集的 CSP-MHS 算法,通过将极小碰集问题转换为约束满足问题后调用 CSP 求解器进行求解,并提出 hard-冲突集与 soft-冲突集概念,以此提高算法对于具有某些特征的极小碰集问题的求解效率. 除了这些完备求解算法,随着近似求解策略的不断发展,许多非完备极小碰集求解算法被提出[14-18]. 这些算法大多使用随机搜索策略,即使对于规模很大的问题,也能够在较短时间内完成求解,但同时这也是以牺牲解的完备性为代价的. 此外,近年来也有许多并行及分布式极小碰集求解算法被提出[19-20].

以上极小碰集求解算法的缺点集中表现为:1)因剪枝或使用随机策略可能丢失正确的解[1,14-18];2)需要建立结构复杂的树或图,算法实现繁琐[1,4-6];3)碰集计算由递归实现,效率较差[1,4-5,8];4)先存储计算所得的所有碰集,最后对非极小碰集进行删减,导致空间复杂度较高[7,16].

这些极小碰集求解算法中,大部分完备算法都显式[1,4-6,8-11]或隐式[7,12]地使用了树形结构来进行算法描述及问题求解. 这与树形结构表达清晰、易于根据实际问题特征加入各种剪枝策略等特点密切相关. 其中基于 SE-Tree 的 HSSE-Tree 算法,所用树结构简单且有很强的规律性,算法实现也较容易. 但由于其采用宽度优先的策略对 SE-Tree 中节点进行生成,使得在碰集计算过程中需对当前层中生成的非碰集节点进行存储,用于下一层需访问节点的生成,这就导致了算法运行过程中较高的空间需求,并且无法对一些无需访问的非解冗余节点进行剪枝,影响了算法效率.

本文通过对 SE-Tree 结构进行深度分析,指出 SE-Tree 在深度优先遍历过程中存在一种新型可剪枝节点,给出相应的剪枝算法,并在此基础上提出结合 SE-Tree 结构特征的极小碰集求解算法. 这种算法的主要优点是:1)采用深度优先求解方式,相较于 HSSE-Tree 算法空间复杂度大幅降低;2)可以对无访问价值的非解冗余节点进行大量剪枝,大幅提高算法效率;3)增加了提前终止条件的判定,进一步提高算法效率;4)仅使用 SE-Tree 结构进行描述,实现过程中仅使用链表和数组,实现简单.

1 预备知识

首先介绍模型诊断中与碰集相关的背景知识,以及集合枚举树 SE-Tree 的相关概念及性质.

1.1 基于模型诊断的相关概念

定义 1[1]. 一个系统可定义为一个三元组(SD, $COMPS$, OBS),其中:

1) SD 为系统描述,是一阶谓词公式集合;

2) $COMPS$ 是系统组成部件的集合,是一个有限常量集;

3) OBS 为观测集,是一阶谓词公式的有限集.

定义 2[1]. 冲突集 CS 是一个部件集$\{c_1, c_2, \cdots, c_n\} \subseteq COMPS$,当且仅当 $SD \cup OBS \cup \{AB(c_1), AB(c_2), \cdots, AB(c_n)\}$是不一致的. 其中 AB 为一元谓词,表示"abnormal". $AB(c)$为真,当且仅当 c 异常,且 $c \in COMPS$.

称冲突集是极小冲突集(minimal conflict set, MCS)[21],当且仅当该冲突集的任意真子集都不是冲突集.

定义 3[1]. 设 F 是一个集合簇,$F=\{S_1, S_2, \cdots, S_n\}$,称 H 为 F 的一个碰集 HS,如果 H 满足 2 个条件:

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

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(minimal hitting set, MHS),当且仅当 H 的任意真子集都不是 F 的碰集.

1.2 SE-Tree 的相关概念及性质

SE-Tree 由 Rymon[22] 提出:一个完全的 SE-Tree 是按照一定的顺序(如数字或字母的顺序等),系统地枚举出一个集合的幂集中所有元素的集合. 例如 $S=\{1,2,3,4\}$的完全集合枚举树如图 1 所示:

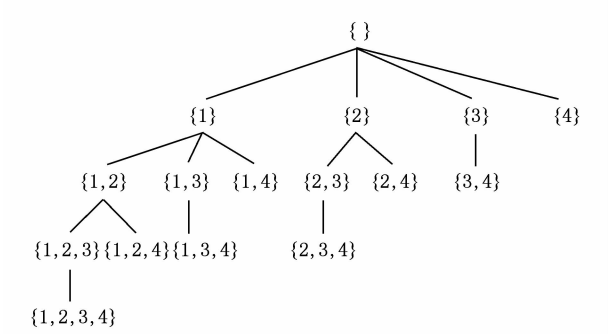

Fig. 1 SE-Tree of $S=\{1,2,3,4\}$.

图 1 $S=\{1,2,3,4\}$对应的 SE-Tree

按上述规则生成的 SE-Tree 具有 2 个性质:

1) 对于所有非叶节点,该节点是其所有子孙节点的真子集,特别地根节点是所有其他节点的真子集;

2) 对于所有非根节点,该节点是其所有祖先节点的真超集,特别地叶节点是其所在分支上所有其他节点的真超集.

根据以上性质和碰集的定义,我们可以得到以下推论:

推论 1. 若一个非叶节点是碰集,则该节点的所有子孙节点都是碰集.

推论 2. 若一个叶节点不是碰集,则该叶节点所在分支上的所有节点都不是碰集.

其中,推论 2 是本文核心算法的基础.

为方便讨论,首先给出一些 SE-Tree 的相关定义:

定义 4. 若一个节点内所有元素都是连续的,则称此节点为纯节点(只含有 1 个元素的节点是纯节点). 若一个叶节点是纯节点,则称该叶节点为纯叶节点.

例如图 1 中,{2,4}不是纯节点,{2,3}是纯节点,{2,3,4}是纯叶节点.

定义 5. 称 SE-Tree 中最左侧的叶节点为头叶节点;最右侧叶节点为尾叶节点.

例如图 1 中{1,2,3,4}为头叶节点,{4}为尾叶节点.

显然,头叶节点和尾叶节点都是纯叶节点. 并且,头叶节点包含了该 SE-Tree 任意节点中可出现的全部元素.

本文集合中的元素在默认情况下都为递增排序.

2 基于 SE-Tree 的深度优先碰集求解算法

首先给出一种基于 SE-Tree 的深度优先极小碰

集求解算法,随后通过 1 个实例对其不足之处进行说明.

基于 SE-Tree 的深度优先极小碰集求解算法,即是在 SE-Tree 上使用深度优先算法求解极小碰集.其一般过程为:对问题对应的 SE-Tree 以左优先进行深度优先遍历.当访问到的节点为碰集时,对该节点的所有子孙节点进行剪枝,对该碰集进行检测:若是极小碰集则加入结果中,否则不加入.继续深度优先遍历,直到 SE-Tree 中所有未被剪枝掉的节点都已被访问.

我们可以发现,以上算法可以看作这样一个过程:每次迭代是对一个叶节点所在分支中的一部分进行遍历.此处,一个分支由根节点到这个叶节点路径上的所有节点构成.迭代开始于本分支与上次进行迭代的分支的最后一个共有节点的下一个节点.在没有遇到碰集节点的情况下,结束于本分支的叶节点;否则结束于第 1 个碰集节点.下面给出基于 SE-Tree 的深度优先极小碰集算法 DF-SE(deep first-SE).

算法 1. DF-SE($F=\{S_1,S_2,\cdots,S_m\}$)

① 初始化:$MHS=\varnothing$,$Node=\varnothing$,$S=\bigcup_{S_i\in F}S_i$,
$Leaf=S$;
② Begin
③ While(1)
④ While(1)
⑤ $Node=next(Node,Leaf)$;
⑥ If ($Node$ 为碰集)
⑦ If ($Node$ 为极小碰集)
⑧ $MHS=MHS\cup Node$;
⑨ EndIf
⑩ Break;
⑪ EndIf
⑫ If ($Node==Leaf$)
⑬ Break;
⑭ EndIf
⑮ EndWhile
⑯ If ($Leaf$ 是尾叶节点)
⑰ Break;
⑱ EndIf
⑲ $Leaf=next\_end(Node)$;
⑳ $Node=next\_begin(Node,Leaf)$;
㉑ EndWhile
㉒ End
㉓ Return MHS.

其中,$Leaf$ 表示本次迭代需遍历的分支上的叶节点,$Node$ 为当前遍历节点,函数 $next$ 用于计算当前分支上下一个需要遍历的节点,函数 $next\_begin$ 用于计算本次迭代遍历分支与下次迭代需遍历分支的最后一个共有节点,函数 $next\_end$ 用于计算下次迭代需遍历分支上的叶节点.

下面给出 1 个结合 SE-Tree 求解极小碰集的深度优先算法过程实例.

例 1. 使用 DF-SE 算法求集合簇 $F=\{\{1,4\},\{2\},\{3\}\}$的极小碰集.

其 SE-Tree 同图 1 给出的一样.求解过程中,需要按顺序遍历{1},{1,2},{1,2,3},{1,2,4},{1,3},{1,3,4},{1,4},{2},{2,3},{2,3,4},{2,4},{3},{3,4},{4}总共 14 个节点,其中{1,2,3},{2,3,4}为极小碰集节点.通过观察可以发现对{1,2,4},{1,3},{1,3,4},{1,4},{2,4},{3},{3,4},{4}这 8 个节点对应分支的遍历没有碰集产生,即这些节点是冗余节点.

定义 6. 在 SE-Tree 中,称非极小的碰集节点为有解冗余节点,节点本身及其所有子孙节点都不为碰集的节点为非解冗余节点.对于不是碰集的叶节点,称之为冗余叶节点.

显然,在极小碰集计算过程中,对非解冗余节点进行剪枝不会对解的正确性及完备性造成影响.接下来我们需要考虑的就是如何最大程度地对非解冗余节点进行剪枝,从而提高算法的效率.

3 结合子集删减的碰集求解算法

在给出具体的算法之前,我们先介绍 SE-Tree 中叶节点在一些结构中的性质.

3.1 SE-Tree 中叶节点的特殊性质及辅助剪枝树

3.1.1 SE-Tree 中叶节点的特殊性质

根据推论 2,若一个叶节点为冗余叶节点,则这个节点对应的分支上不存在碰集,即这条分支没有必要进行遍历.

一种简便的方法就是在对每个分支进行遍历前,先对分支的叶节点进行检测,若叶节点对应的集合是碰集,则对该分支进行遍历,否则将该分支剪掉.注意,此时并不能说明这条分支上所有的节点都是非解冗余节点,但在非冗余节点为根的子树下必有不少于 1 个非冗余叶节点.因此对冗余叶节点所在分支进行剪枝并不会导致分支上的非冗余节点被剪枝,其必然会在对其他分支的遍历中被访问.但在

一棵 SE-Tree 中,叶节点的数量占总节点数的一半,此方法显然十分低效.

定理 1. 设节点 A 为 SE-Tree 中的一个叶节点.若节点 A 不是尾叶节点,则 SE-Tree 中至少存在 1 个叶节点 B,使得叶节点 B 为节点 A 的真子集;若节点 A 不是头叶节点,则 SE-Tree 中至少存在 1 个叶节点 C,使得叶节点 C 为节点 A 的真超集.

显然,对于一个叶节点,若其不是头叶节点,则头叶节点是其真超集;若其不是尾叶节点,则尾叶节点是其真子集.

通过定理 1 我们知道,若节点 A 为冗余叶节点,且节点 A 不是尾叶节点,那么必存在至少 1 个叶节点 B,使得叶节点 B 为节点 A 的真子集.因为节点 A 为非解冗余节点,不是碰集,则其真子集自然也不是碰集,即叶节点 B 同为冗余叶节点.若能通过对节点 A 的判断将节点 B 对应的分支剪掉,必将极大地提高算法效率.此时,我们就需要引入一种新的结构来进行辅助剪枝.

3.1.2 辅助剪枝树

定义 7. 将一棵 SE-Tree 中的尾叶节点作为树根,以剩余叶节点去掉该尾叶节点中的元素后余下的元素作为度量标准,应用与该 SE-Tree 相同的排序规则生成一棵树,称这棵树为该 SE-Tree 的辅助剪枝树.

如图 2 就是图 1 中 SE-Tree 的辅助剪枝树.

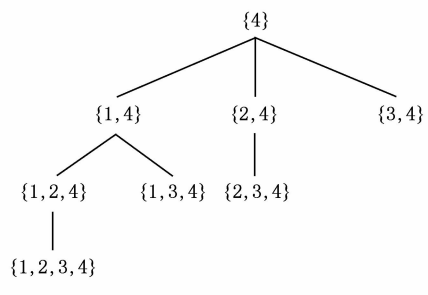

Fig. 2 Assistant pruning tree of example 1.

图 2 例 1 对应的辅助剪枝树

因为辅助剪枝树是应用与 SE-Tree 相似的规则生成的,所以拥有许多与 SE-Tree 一样的性质.如每个节点都是其祖先节点的真超集;若一个节点不是碰集,则其所有的祖先节点都不是碰集.因此,当我们发现辅助剪枝树中的一个节点不是碰集时,其祖先节点在 SE-Tree 中必为冗余叶节点,它们对应的分支也就都不需要进行访问.

通过观察我们还会发现:在 SE-Tree 中进行深度优先算法过程中对叶节点的访问顺序,与在其对应的辅助剪枝树中后序遍历访问节点的顺序是一致的.这样就可以对一些非解冗余节点进行剪枝,减少需访问分支的数量.如在例 1 中,当发现节点{1,3,4}为非解冗余节点后就没有必要对{1,4}节点所在的分支进行遍历,可直接跳转到对{2,3,4}节点所在的分支进行遍历.

下面给出 SE-Tree 中结合辅助剪枝树的深度优先求解算法 DFI-SE(deep first with improvement-SE):

算法 2. DFI-SE($F=\{S_1,S_2,\cdots,S_m\}$)

① 初始化:$MHS=\varnothing$,$Node=\varnothing$,$Prune\_node=\varnothing$,$S=\bigcup_{S_i\in F}S_i$,$Leaf=S$;
② Begin
③ While(1)
④ If (!($Leaf\subseteq Prune\_node$))
⑤ While(1)
⑥ $Node=next(Node,Leaf)$;
⑦ If ($Node$ 为碰集)
⑧ If ($Node$ 为极小碰集)
⑨ $MHS=MHS\cup Node$;
⑩ EndIf
⑪ Break;
⑫ EndIf
⑬ If ($Node==Leaf$)
⑭ $Prune\_node=Node$;
⑮ Break;
⑯ EndIf
⑰ EndWhile
⑱ EndIf
⑲ If ($Leaf$ 是尾叶节点)
⑳ Break;
㉑ EndIf
㉒ If ($Leaf$ 是冗余叶节点)
㉓ $Leaf$ = 辅助剪枝树中 $Leaf$ 右侧的第 1 个叶节点;
㉔ Else
㉕ $Leaf=next\_end(Node)$;
㉖ EndIf
㉗ $Node=next\_begin(Node,Leaf)$;
㉘ EndWhile
㉙ End
㉚ Return MHS.

算法 DFI-SE 与算法 DF-SE 的主要区别在于:1)每次迭代开始前对此次遍历分支的叶节点是否为

记录的冗余叶节点(*Prune_node*)的子集进行判断(行④);2)每次迭代到叶节点时,若是冗余叶节点,则对 *Prune_node* 进行更新(行⑭);3)增加了遇到冗余叶节点时下次需遍历分支的跳转方法(行㉒). *Prune_node* 只用于保存最新遍历到的冗余叶节点,这样做保证了算法具有较低的空间与时间复杂度.

3.1.3 多级辅助剪枝树

结合辅助剪枝树的深度优先算法已经能对部分冗余叶节点所在分支进行剪枝,但剪枝效果十分有限. 为了能够将剪枝最大化,下面引入多级辅助剪枝树的概念.

定义 8. 称定义 7 中得到的辅助剪枝树为 1 级辅助剪枝树,将 1 级辅助剪枝树中的叶节点按照定义 7 中的规则生成的树称为 2 级辅助剪枝树,以此类推可生成级别更高的辅助剪枝树. i 级辅助剪枝树用 Ψ_i 表示. 当辅助剪枝树中只包含 1 个节点时,无法生成更高级别的辅助剪枝树.

图 3 为例 1 中 SE-Tree 的各级辅助剪枝树.

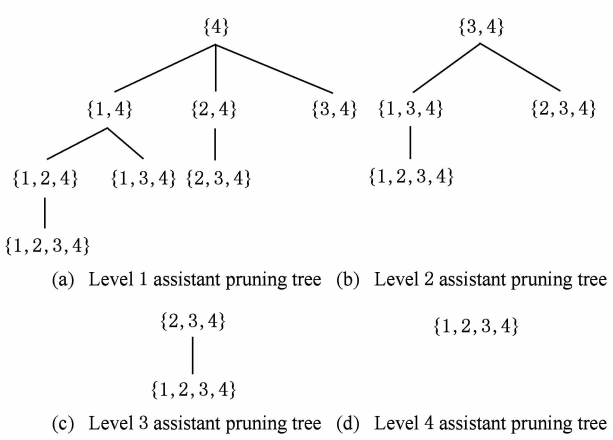

Fig. 3 Multi-level assistant pruning tree of example 1.

图 3 例 1 对应的各级辅助剪枝树

定理 2. 设有集合簇 $F=\{S_1,S_2,\cdots,S_k\}$,$S=\bigcup_{S_i\in F}S_i$,$n=|S|$,则 F 对应的 SE-Tree 最多可以生成 n 级辅助剪枝树.

通过观察图 3,我们可以发现:每个辅助剪枝树的根节点包含的元素为 S 中最后 m 个连续元素,其中 m 为辅助剪枝树的级别. n 级辅助剪枝树的根节点 P 中包含 S 中的全部元素,其无子孙节点,即 n 级辅助剪枝树中只包含 1 个节点. 根据定义 8,此时无法继续生成 $n+1$ 级辅助剪枝树.

多级辅助剪枝树的引入,使得原本在 1 级辅助剪枝树中无法被剪枝的非解冗余节点在满足一些条件的情况下,可以在更高级别的辅助剪枝树中被剪掉. 例如冗余叶节点{3,4}在 1 级辅助剪枝树中为叶节点,因此根据算法 DFI-SE 无法被剪枝,但其在 2 级辅助剪枝树中为{1,3,4}的父节点. 因此当发现{1,3,4}为非解冗余节点后,根据 2 级辅助剪枝树{3,4}必为非解冗余节点,其所在分支没有遍历的必要.

因为在多级辅助剪枝树中,$i+1$ 级辅助剪枝树中的节点为 i 级辅助剪枝树中的所有叶节点,所以级别较高的辅助剪枝树中的节点必然出现在比其级别低的辅助剪枝树中. 这就引出了首次出现的概念.

定义 9. 称包含节点 A 的最高级辅助剪枝树为节点 A 的首次出现辅助剪枝树,记为 Υ_A;该辅助剪枝树的级别为 A 的首次出现级,用 Ω_A 表示.

如例 1 中 $\Upsilon_{\{1,3,4\}}$ 为图 3 中的 Ψ_2,$\Omega_{\{1,3,4\}}=2$.

定义 10. 一个有序集合从前至后顺序地删除一些元素,直至剩余元素都为连续元素,称此时剩余元素为该集合的尾元素.

由多级辅助剪枝树的形成规则我们可以得到以下推论:

推论 3. 一棵 SE-Tree 中某个叶节点 A 的首次出现级即为该节点中最大的尾元素个数.

在此,我们需要先简单阐述在多级辅助剪枝树中发现当前节点为非解冗余节点后,下次算法迭代需遍历分支对应叶节点的选取方法:

当发现叶节点 A 为冗余叶节点后,若叶节点 A 不是 Υ_A 中的根节点,则选取 Υ_A 中节点 A 右侧第 1 个叶节点作为下次检测的叶节点;若节点 A 为 Υ_A 中的根节点,此时为特殊情况,将在 3.2 节中进行说明.

多级辅助剪枝树的结构特点保证了通过此方法被剪枝的叶节点都是叶节点 A 的真子集,即保证了方法的正确性.

3.2 结合 SE-Tree 结构特征的极小碰集求解算法

3.1 节中介绍了如何在基于 SE-Tree 的深度优先算法中结合多级辅助剪枝树对非解冗余节点进行最大化剪枝,其主要是基于非碰集的子集仍为非碰集的思想. 下面给出结合 SE-Tree 结构特征的极小碰集求解算法 SPHS(subset-pruning hitting set algorithm)的终止条件及算法描述.

定义 11. 称一棵 SE-Tree 中任意节点 W 及其所有子孙节点所组成的子树为 K 级子树,K 为 W 中元素的个数.

定理 3. 当以左优先深度优先遍历一棵 SE-Tree 时,若发现一个冗余叶节点为纯节点,则算法可以终止,当前求得的极小碰集即为全部极小碰集.

证明. 设纯叶节点 A 为冗余叶节点. 若纯叶节点 A 为 SE-Tree 中的尾叶节点,则 SE-Tree 中已无未遍历节点,算法应终止;否则纯叶节点 A 所在 1 级子树的右侧必有其他 1 级子树. 从 SE-Tree 的结构可知:

1) SE-Tree 中每棵 1 级子树中有且只有 1 个纯叶节点,该节点会出现在该子树的最左侧分支上,其包含了该子树节点中可能出现的全部元素,因此,这个纯叶节点是该 1 级子树中所有其他叶节点的真超级,若该纯叶节点为冗余叶节点,则该 1 级子树中所有的叶节点都是冗余叶节点;

2) 纯叶节点 A 所在 1 级子树右侧所有 1 级子树的纯叶节点都是纯叶节点 A 的真子集,即若纯叶节点 A 为冗余叶节点,则纯叶节点 A 右侧的所有纯叶节点皆为冗余叶节点.

综合 1)和 2),若纯叶节点 A 不是 SE-Tree 中的尾叶节点,则其右侧的所有叶节点都为冗余叶节点,即纯叶节点 A 右侧的所有分支均不包含碰集节点,算法可终止. 证毕.

算法 SPHS 中需要用到一个名为辅助剪枝表的结构,其主要用来存储各级辅助剪枝树中最新遍历到的冗余叶节点. 辅助剪枝表结构如图 4 所示:

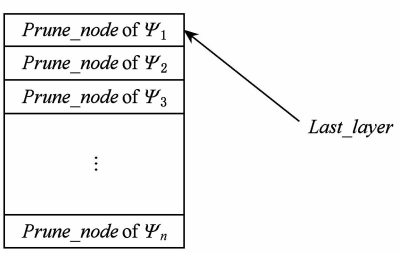

Fig. 4 Structure of assistant pruning table.

图 4 辅助剪枝表结构

图 4 中的 $Last\_layer$ 用来记录最近遍历到的冗余叶节点的首次出现级. 在算法 SPHS 中,会用当前需遍历分支对应叶节点的首次出现级与 $Last\_layer$ 进行比较. 若当前叶节点的首次出现级大于或等于 $Last\_layer$,则使用与当前叶节点首次出现级相对应的表位置存储的信息对该叶节点进行检测;否则,使用与 $Last\_layer$ 相对应的表位置存储的信息进行检测. 这主要是因为首次出现级较低的节点可能是首次出现级较高的节点的子集,而相反的情况则不会出现.

下面给出 SPHS 的算法描述.

算法 3. SPHS($F=\{S_1,S_2,\cdots,S_m\}$)

① 初始化:$MHS=\varnothing$,$Node=\varnothing$,
$Prune\_List[n+1]=\varnothing$,$Last\_layer=0$,
$S=\bigcup_{S_i\in F}S_i$,$Leaf=S$;
② Begin
③ While(1)
④ If (!($Leaf\subseteq Prune\_List[\max(\Omega_{Leaf},Last\_layer)]$))
⑤ While(1)
⑥ $Node=next(Node,Leaf)$;
⑦ If ($Node$ 为碰集)
⑧ If ($Node$ 为极小碰集)
⑨ $MHS=MHS\cup Node$;
⑩ EndIf
⑪ Break;
⑫ EndIf
⑬ If ($Node==Leaf$)
⑭ If ($Leaf$ 为纯叶节点)
⑮ Return MHS;
⑯ EndIf
⑰ $Last\_layer=\Omega_{Node}$;
⑱ $Prune\_List[Last\_layer]=Node$;
⑲ Break;
⑳ EndIf
㉑ EndWhile
㉒ EndIf
㉓ If ($Leaf$ 是尾叶节点)
㉔ Break;
㉕ EndIf
㉖ If ($Leaf$ 是冗余叶节点)
㉗ $Leaf=\Upsilon_{Leaf}$ 中 $Leaf$ 右侧的第 1 个叶节点;
㉘ Else
㉙ $Leaf=next\_end(Node)$;
㉚ EndIf
㉛ $Node=next\_begin(Node,Leaf)$;
㉜ EndWhile
㉝ End
㉞ Return MHS.

其中与算法 DFI-SE 的主要区别在于:1)每次迭代开始前的判断规则变为使用需遍历分支的叶节点与剪枝表中相应项进行比较(行④);2)加入新的提前终止条件(⑭);3)访问到冗余叶节点时需对辅助剪枝表进行更新(行⑰);4)发现冗余的叶节点后,跳转须在其首次出现辅助剪枝树中进行(行㉖).

这些都有效地保证了算法效率.在第 4 节中,我们会以多组实验数据对比的方式,分析说明 SPHS 算法相较于 Boolean 算法的优势.

4 实验结果

本节将使用 SPHS 算法与当前效率较高的 Boolean 算法进行比较,并给出 2 种算法在随机生成测试用例下的实验结果.实验平台如下:Windows XP 操作系统、CPU AMD Athlon™ 64 X2 Dual Core Processor 3600+ 1.9 GHz、2.00 GB RAM.

实验用例使用随机生成器产生,输入参数有元素个数 m、集合簇中集合个数 n 以及元素在一个集合中出现的概率 p.同一个用例中,所有元素的 p 均相等,每个集合包含元素的平均个数约等于 mp.本文使用的实验用例共分为 4 组,每组元素个数固定,分别为 30,25,20,15,各组均包含 p 取值 0.05~0.94 的 19 个用例,所有用例集合个数均为 200.本文所有实验数据均为使用相同参数下独立生成的 10 个用例进行实验所得结果的平均值.

从图 5 我们可以看出,对于耗时较少的用例(2 种算法运行时间均低于 0.2 s),Boolean 算法占优较多;但对于绝大多数其他用例,SPHS 算法拥有较大优势.尤其对于耗时较长的用例(2 种算法运行时间均高于 20 s),多数点已经处于 5 倍对比线甚至 10 倍对比线的上方.不难发现,对比 Boolean 算法,SPHS 算法在耗时越长的用例中优势较为明显.接下来,我们将进一步分析出现这种现象的原因.

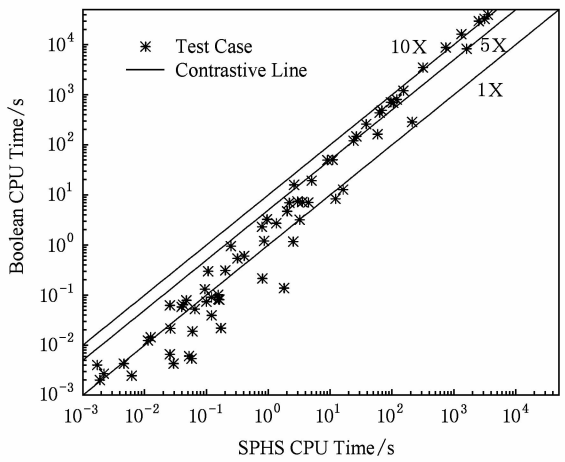

Fig. 5 Experimental comparison of SPHS and Boolean.

图 5 SPHS 算法与 Boolean 算法实验结果比较

图 6 中各坐标系横轴表示元素出现概率 p,MHS 表示对应实例的极小碰集数量,与右侧纵轴相关.通过图 6 我们可以发现,相对于 Boolean 算法,SPHS 算法对于实例对应的极小碰集数量的敏感度

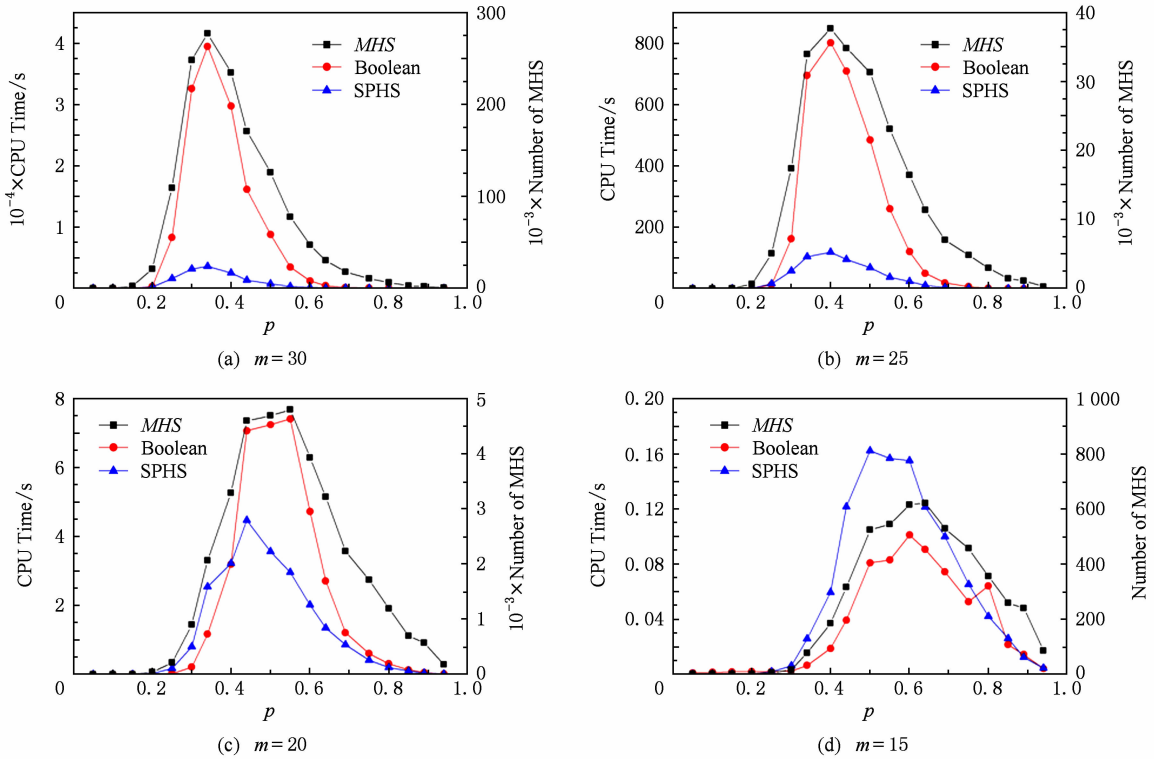

Fig. 6 Experimental comparison of the two algorithms based on diverse number of elements.

图 6 不同元素个数实例下 2 种算法实验结果比较

并不高,尤其在图 6(c)与图 6(d)中更为明显,一些极小碰集数量上升的同时,算法运行时间却发生了下降.我们还可以发现图 6(d)是唯一一个 SPHS 算法效率要差于 Boolean 算法的情况.这说明,相对于 Boolean 算法,SPHS 算法的优势主要集中在规模相对较高的问题上(元素较多).

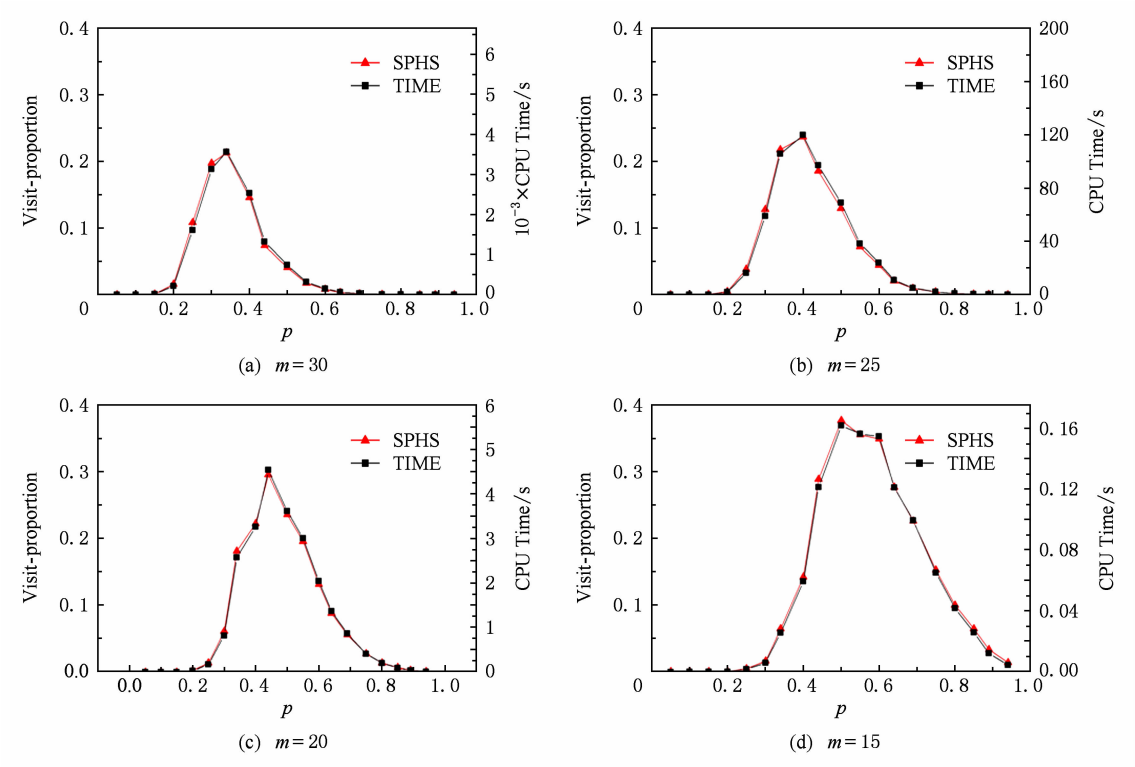

Fig. 7 Visit-proportion of SPHS based on diverse number of elements.

图 7 SPHS 算法访问分支数占总分支数的比例

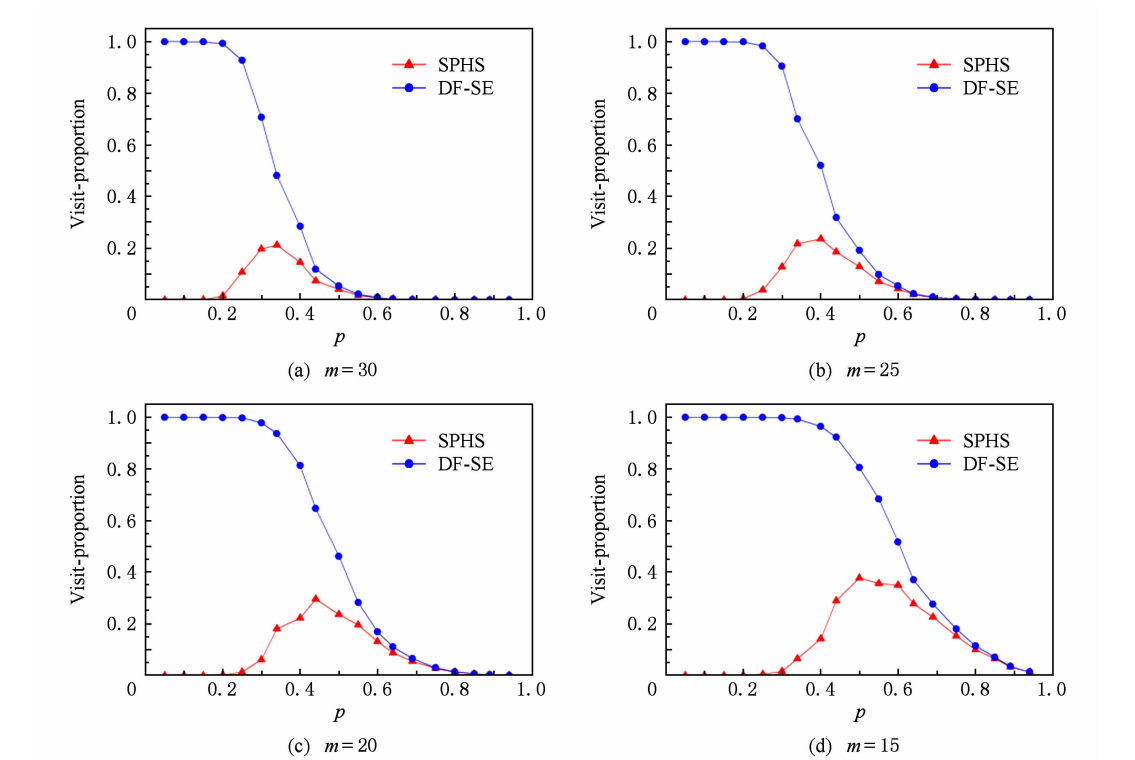

Fig. 8 Experimental comparison of visit-proportion of SPHS and DF-SE.

图 8 SPHS 与 DF-SE 访问分支数对比

图 7 各个坐标系中各包含 2 个折线图.其中 SPHS 表示 SPHS 算法访问分支数占该实例对应的总分支数的比例,与左侧纵轴相关;TIME 为此组用例 SPHS 算法的 CPU 运行时间,与右侧纵轴相关.从图 7 中我们可以发现,各组数据对应的 2 条折线拟合度很高,这说明使用访问分支数与剪枝分支数来衡量此算法的效率是合理的.通过对比各组实验数据我们还能够发现,当用例元素个数增多时,SPHS 算法访问分支数占总分支数的比例有逐渐下降的趋势.也就是说,对于元素数较大的用例,SPHS 算法的剪枝比例较高.如图 7(d)中的最高点已接近 0.4,而图 7(a)中的最高点仅在 0.2 附近.这与实验结果中所表现出来的在较难实例中 SPHS 算法更占优的趋势是相符的.

图 8 给出了 SPHS 与 DF-SE 算法在各组实例中访问分支占总分支数比例.通过对比,我们可以更直观地看出 SPHS 算法中所加策略对其算法效率的提升.我们可以发现,对于各组元素出现概率 p 较小的实例,DF-SE 算法几乎需要访问所有的分支.这主要是未加入提前终止策略所导致的结果.

5 结束语

本文首先给出了基于 SE-Tree 结构的深度优先极小碰集求解算法 DF-SE,通过对 SE-Tree 中叶节点的特殊性质和算法执行过程的分析,并结合现有碰集算法中有解冗余节点的特征,提出非解冗余节点概念.若能在算法执行过程中对此类节点进行剪枝,算法效率将得到提高且不会对算法的正确性及完备性造成影响.针对此问题,本文给出一种新型的辅助剪枝结构——辅助剪枝树.通过将此结构与基于 SE-Tree 结构的深度优先极小碰集求解算法相结合得到的 DFI-SE 算法,能够对部分非解冗余节点进行剪枝.为进一步提高算法效率,本文还通过对辅助剪枝树结构的递归定义得到了多级辅助剪枝树,并且在保证解的完备性的前提下给出了算法的提前终止条件,最终得到 SPHS 算法.该算法容易理解且易于实现,虽然使用了 SE-Tree 以及多级辅助剪枝树,但实现过程中并不需要构造树结构,保证了算法较低的空间复杂度和较高的执行效率.实验结果表明,对比当前效率较高的 Boolean 算法,SPHS 算法对于规模较大的问题具有明显优势,对于部分较难问题的求解效率甚至提高了 1 个数量级以上.

参 考 文 献

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

[2] Bonet B, Helmert M. Strengthening landmark heuristics via hitting sets [C] //Proc of the 19th European Conf on Artificial Intelligence. Amsterdam: IOS Press, 2010: 329-334

[3] Wotawa F. On the relationship between model-based debugging and program slicing [J]. Artificial Intelligence, 2002, 135(1): 125-143

[4] 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

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

[6] Jiang Yunfei, Lin Li. Computing the minimal hitting set with binary HS-tree [J]. Journal of Software, 2002, 13(12): 2267-2274 (in Chinese)
(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)

[7] 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)

[8] Ouyang Dantong, Ouyang Jihong, Cheng Xiaochun, et al. A method of computing hitting set in model-based diagnosis [J]. Chinese Journal of Scientific Instrument, 2004, 25(4): 605-608 (in Chinese)
(欧阳丹彤, 欧阳继红, 程晓春, 等. 基于模型诊断中计算碰集的方法[J]. 仪器仪表学报, 2004, 25(4): 605-608)

[9] Zhao Xiangfu, Ouyang Dantong. A method of combing SE-Tree to compute all minimal hitting sets [J]. Progress in Natural Science, 2006, 16(2): 169-174

[10] Chen Xiaomei, Meng Xiaofeng, Qiao Renxiao. Method of computing all minimal hitting set based on BNB-HSSE [J]. Chinese Journal of Scientific Instrument, 2010, 31(1): 61-67 (in Chinese)
(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)

[11] Zhang Liming, Ouyang Dantong, Zeng Hailin. Computing the minimal hitting set based on dynamic maximum degree [J]. Journal of Computer Research and Development, 2011, 48(2): 209-215 (in Chinese)
(张立明, 欧阳丹彤, 曾海林. 基于动态极大度的极小碰集求解方法[J]. 计算机研究与发展, 2011, 48(2): 209-215)

[12] Pill I, Quaritsch T. Optimizations for the Boolean approach to computing minimal hitting Sets [C] //Proc of the 20th European Conf on Artificial Intelligence. Amsterdam: IOS Press, 2012: 648-653

[13] Wang Yiyuan, Ouyang Dantong, Zhang Liming, et al. A method of computing minimal hitting sets using CSP [J]. Journal of Computer Research and Development, 2015, 52(3): 588-595 (in Chinese)
(王艺源, 欧阳丹彤, 张立明, 等. 利用CSP求解极小碰集的方法[J]. 计算机研究与发展, 2015, 52(3): 588-595)

[14] Lin Li, Jiang Yunfei. Computing minimal hitting sets with genetic algorithm [C/OL] //Proc of the 13th Int Workshop on Principles of Diagnosis. 2002 [2015-01-12]. https://www.researchgate.net/publication/235118684_Thirteenth_International_Workshop_on_Principles_of_Diagnosis_DX-2002

[15] Cincotti A, Cutello V, Pappalardo F. An ant-algorithm for the weighted minimum hitting set problem [C] //Proc of the 2003 IEEE Symp on Swarm Intelligence. Piscataway, NJ: IEEE, 2003: 1-5

[16] Huang Jie, Chen Lin, Zou Peng. Computing minimal diagnosis by compounded genetic and simulated annealing algorithm [J]. Journal of Software, 2004, 15(9): 1345-1350 (in Chinese)
(黄杰, 陈琳, 邹鹏. 一种求解极小诊断的遗传模拟退火算法[J]. 软件学报, 2004, 15(9): 1345-1350)

[17] Abreu R, Gemund A J C V. A low-cost approximate minimal hitting set algorithm and its application to model-based diagnosis [C] //Proc of the 8th Symp on Abstraction, Reformulation, and Approximation, Vol 9. Menlo Park, CA: AAAI, 2009: 2-9

[18] Zhou Gan, Feng Wenquan, Jiang Bofeng, et al. Computing minimal hitting set based on immune genetic algorithm [J]. International Journal of Modelling, Identification and Control, 2014, 21(1): 93-100

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

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

[21] Han B, Lee S J. Deriving minimal conflict sets by CS-tree with mark set in diagnosis from first principles [J]. IEEE Trans on Systems, Man, and Cybernetics: Cybernetics, 1999, 29(2): 281-286

[22] Rymon R. Search through systematic set enumeration [C] //Proc of the 3rd Int Conf on Principles of Knowledge Representation and Reasoning. San Francisco, CA: Morgan Kaufmann, 1992: 539-550

Liu Siguang, born in 1988. Master candidate at Jilin University. His main research interests include model-based diagnosis, SAT problem and automated reasoning.

Ouyang Dantong, born in 1968. Professor and PhD supervisor of Jilin University. Senior member of China Computer Federation. Her main research interests include model-based diagnosis, model checking and automated reasoning (ouyangdantong@163.com).

Wang Yiyuan, born in 1988. PhD candidate at Jilin University. His main research interests include model-based diagnosis, SAT problem and automated reasoning (yiyuanwangjlu@126.com).

Jia Fengyu, born in 1991. Master candidate at Jilin University. His main research interests include model-based diagnosis, SAT problem and automated reasoning (jiafy13@mails.jlu.edu.cn).

Zhang Liming, born in 1980. PhD, post-doctor at Jilin University. His main research interests include model-based diagnosis, model checking and boolean satisfiability.