软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2018,29(12):3733−3746 [doi: 10.13328/j.cnki.jos.005311]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

极小碰集求解中候选解极小性判定方法*

刘思光[1,2], 欧阳丹彤[1,2], 张立明[1,2]

[1](吉林大学 计算机科学与技术学院,吉林 长春 130012)

[2](符号计算与知识工程教育部重点实验室(吉林大学),吉林 长春 130012)

通讯作者: 张立明, E-mail: limingzhang@jlu.edu.cn

摘 要: 极小碰集问题是人工智能中的重要问题,应用广泛.碰集极小性判定,作为极小碰集求解过程中的关键步骤,效率的高低会对极小碰集求解算法的耗时产生直接影响.现有的极小碰集求解算法主要使用子集检测方法进行碰集极小性判定.针对子集检测方法在极小碰集簇规模较大时效率较低的问题,提出了基于元素独立覆盖度检测的碰集极小性判定方法——ICC 方法,剥离了碰集极小性判定耗时与极小碰集簇大小的相关性;通过深入分析增量求解过程中非极小碰集的产生原因,给出了 ICC 方法的增量判定形式 IICC 方法,使其可以尽早发现并丢弃非极小候选解,为使用其增量极小碰集求解算法带来额外的剪枝效果,进一步提升算法的效率.实验结果表明:该方法易于实现,可扩展性强,对于当前效率较高的 Boolean 算法,使用 IICC 方法后,算法可求解问题的规模和整体效率均有明显提升,效率提升最高达 4 个数量级以上.

关键词: 基于模型诊断;极小碰集;碰集极小性判定;预剪枝;增量方法

中图法分类号: TP18

中文引用格式: 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.软件学报,2018,29(12):3733−3746. http://www.jos.org.cn/1000-9825/5311.htm

英文引用格式: Liu SG, Ouyang DT, Zhang LM. Method of minimality-checking of candidate solution for minimal hitting set algorithm. Ruan Jian Xue Bao/Journal of Software, 2018,29(12):3733−3746 (in Chinese). http://www.jos.org.cn/1000-9825/5311.htm

Method of Minimality-Checking of Candidate Solution for Minimal Hitting Set Algorithm

LIU Si-Guang[1,2], OUYANG Dan-Tong[1,2], ZHANG Li-Ming[1,2]

[1](College of Computer Science and Technology, Jilin University, Changchun 130012, China)

[2](Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education (Jilin University), Changchun 130012, China)

Abstract: Minimal hitting set (MHS) problem is an important problem with widely applications in artificial intelligence. During computing all minimal hitting set, how to check the minimality of hitting sets is a key issue. Most of exhaustive MHS algorithms ensure the minimality of results by using subset checking method where its effectiveness is mainly relevant to the scale of minimal hitting sets, i.e., the larger the scale of the MHSs is, the lower the effectiveness of this method has. Consequently, when coming to solve the massive cases, the effectiveness of these algorithms is not high. To overcome this problem, this paper proposes independent coverage checking (ICC) and then develops it into an incremental method named IICC. The effectiveness of this method is irrelevant to the scale of minimal hitting sets. Moreover, when computing all MHS by the incremental MHS algorithm with IICC, it can incrementally test the minimality of

* 基金项目: 国家自然科学基金(61133011, 61402196, 61272208, 61003101, 61170092); 中国博士后科学基金(2013M541302); 吉林省科技发展计划基金(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(ZSDZZZZXK12)

Foundation item: National Natural Science Foundation of China (61133011, 61402196, 61272208, 61003101, 61170092); China Postdoctoral Science Foundation (2013M541302); Jilin Province Science and Technology Development Plan (20140520067JH); Provincial Key Disciplines Foundation of Computer Software and Theory of Zhejiang Normal University (ZSDZZZZXK12)

收稿时间: 2016-03-17; 修改时间: 2016-09-17, 2016-11-06; 采用时间: 2017-04-22

candidate solutions, resulting in cutting down many unnecessary non-minimal hitting sets. IICC is used to optimize the Boolean algorithm which is an incremental MHS algorithm using subset checking. The results show that IICC method significantly outperforms the subset checking methods in terms of running time.

Key words: model-based diagnosis; minimal hitting set; minimality-checking of hitting set; pre-pruning; incremental method

极小碰集问题一直是人工智能领域的热点问题,在许多理论和实际问题中有着广泛应用.如:基于模型诊断中,候选产生阶段是通过计算所有极小冲突集的极小碰集得到极小诊断解[1];溯因推理问题、老师与课程问题,以及智能规划、软件调试中的核心问题,都可以通过转换为极小碰集问题后使用成熟的极小碰集求解算法来提高求解效率[2−4];极小顶点覆盖、极小集合覆盖等极小覆盖问题,也可以视为极小碰集问题的特殊形式.

迄今为止,国内外众多研究人员都对极小碰集的求解方法进行了研究和改进.1987 年,著名的人工智能专家 Reiter 完成了该领域的开创性工作,提出了最早的极小碰集计算方法 HS-Tree 方法[1],并将其应用在基于模型的诊断中.但此方法存在一些缺陷,可能会因剪枝而丢失正确解.针对此问题,1989 年,Greiner 等人对此方法进行改进后提出了 HS-DAG 方法[5].2001 年,Wotawa 提出了 HS-Tree 的改进算法——HST-Tree 方法[6],理论上,很大程度地降低了极小碰集求解过程中子集检测的数量.2002 年和 2003 年,姜云飞等人提出了 BHS-Tree[7]和布尔代数方法[8],前者在解决了 HS-Tree 方法可能因剪枝而丢解问题的同时,有效减少了生成节点的数量;后者将问题集合簇编码为布尔表达式,通过布尔代数计算来求解所有极小碰集,不仅提高了求解的效率,而且简化了数据结构. 2004 年,欧阳丹彤等人对 HS-DAG 方法进行了改进,提出了 New HS-Tree 方法[9],相对于 HS-DAG 方法,大幅减少了搜索节点的数量.2006 年,赵相福等人提出了 HSSE-Tree 方法[10],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.近年来,随着对极小碰集问题的深入研究,一些新的求解方法及改进算法不断被提出[11−15].除了以上这些完备求解方法,随着近似求解策略的不断发展,学者们提出了许多非完备极小碰集求解方法[16−20].这些方法大多使用随机搜索策略,对比传统的完备方法求解,效率大幅提升,即使对于规模很大的问题,也能够在可接受的时间内进行求解,但同时,这也是以牺牲解的完备性为代价的.此外,也有一些并行及分布式极小碰集求解算法被提出[21,22].

碰集极小性判定作为极小碰集求解过程中的关键步骤,其任务是保证算法最终求得问题解的极小性,重要性不言而喻.同时,所用判定方法效率的高低也会对极小碰集求解算法效率产生直接影响.但是,当前对于极小碰集求解算法的研究主要集中在搜索过程的改进上,未见专门针对碰集极小性判定方法进行优化的相关工作.

现有的极小碰集求解算法普遍采用子集检测方法来进行碰集极小性判定,每次调用该方法,都需要将新得到的碰集与极小碰集簇中部分甚至全部碰集进行比较.这使得一般情况下,该方法的调用耗时往往与极小碰集簇中碰集的个数正相关.随着问题难度的增加,算法调用子集检测方法进行碰集极小性判定的耗时会很快超过搜索耗时,在算法总耗时中占据较大比例.若能对此进行优化改进,降低碰集极小性判定耗时,将会大幅提高现有算法的求解效率.针对此问题,本文提出了一种基于元素独立覆盖度检测的碰集极小性判定方法,剥离了碰集极小性判定耗时与极小碰集簇大小的相关性.并根据增量求解过程中非极小碰集的产生原因,对所提方法进行优化,使其能够快速对新产生候选解的极小性给出判定结果,通过尽早发现并丢弃非极小候选解,为使用其的极小碰集增量求解算法带来额外的剪枝收益.

本文将应用所提方法对当前完备求解算法中效率较高[23]的 Boolean 算法进行优化.通过优化前后算法对相同实例求解效率的对比实验,对优化效果进行说明.

1 预备知识

1.1 基于模型诊断及极小碰集的相关概念

定义 1[1].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]. 称一个部件集$\{c_1,c_2,\ldots,c_n\}\subseteq COMPS$ 是冲突集,当且仅当 $SD\cup OBS\cup\{AB(c_1),AB(c_2),\ldots,AB(c_n)\}$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(MCS)[24],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,$F=\{S_1,S_2,\ldots,S_n\}$,称 H 为 F 的一个碰集 *HS*,如果 H 满足以下条件:

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(MHS),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,我们通过例 1 对以上提到的概念进行说明.

例 1:对于集合簇 $F=\{\{1,2,8\},\{1,2,9\},\{1,3,10\},\{1,3,11\},\{1,4,12\},\{2,13\},\{2,14\},\{3,15\},\{3,16\},\{4,17\},\{4,18\},\{5,19\},\{6,20\},\{7,21\}\}$.{1,2,3,4,5,6,7},{2,3,4,5,6,7}及{1,2,3,4}都是该集合簇的候选解,其中,{1,2,3,4,5,6,7}与{2,3,4,5,6,7}是碰集,并且{2,3,4,5,6,7}为极小碰集.

碰集的极小性判定,即是对 F 的一个碰集 H 是否为该集合簇的极小碰集给出验证结果.现有的极小碰集求解算法一般是先求出碰集,再对其极小性进行判定.对这类算法来说,候选解的极小性判定等价于碰集的极小性验证.但就于一般情况而言,两者存在一定差别,极小候选解和极小碰集也是不同的概念,具体定义将在第 2.1 节中给出.

1.2 基于子集检测的碰集极小性判定方法

根据定义 3,我们可以得到以下推论.

推论 1. 设 H 是集合簇 F 的一个碰集,若 F 存在碰集 L 且 L 为 H 的真子集,则 H 不是 F 的极小碰集.

基于子集检测的碰集极小性判定方法就是根据推论 1,当极小碰集求解算法得到一个新碰集时,将其与极小碰集簇中的碰集进行比较,根据比较结果对极小碰集簇进行一些操作.具体过程见算法 1:

算法 1. Subset_Checking.

Function: *Subset_Checking*(*H*, *allMHS*)

Input: 碰集 H,极小碰集簇 $allMHS=\{MHS_1,MHS_2,\ldots,MHS_n\}$;

Output: 极小碰集簇 *allMHS*.

```
Begin
1.  for (i=1; i≤n; i++)
2.  {
3.    if (H⊊MHS_i)
4.      allMHS.Remove(MHS_i);
5.    else if (MHS_i⊆H)
6.      return;
7.  }
8.  allMHS←allMHS∪{H};
End
```

在这一过程中,若发现新得到碰集的真超集,则说明该超级必不为集合簇的极小碰集,将其从极小碰集簇中删去(第 3 行);若发现新得到碰集的子集,则将新得到碰集丢弃,结束比较过程(第 5 行);若比较过程结束时都未发现新得到碰集的子集,则将该碰集暂时加入极小碰集簇中(第 8 行).

当极小碰集求解算法结束时,极小碰集簇中保留的碰集即是问题集合簇的所有极小碰集.需要注意的是:在极小碰集算法运行的过程中,极小碰集簇中保留的碰集并不一定是极小的.

这种方法存在的主要问题是:极小碰集簇的规模会随着极小碰集求解算法的进行而不断变大,新得到碰集的极小性判定时间也不断增加.这使得当问题难度较高时,使用此种碰集极小性判定方法的极小碰集求解算法很难在可接受的时间内进行完备求解.而本文提出的基于元素独立覆盖度的碰集极小性判定方法则不存在这种问题.

Boolean 算法正是使用了本节介绍的基于子集检测的碰集极小性判定方法.使用本文所提方法改进后的 Boolean 算法及两种算法的比较将在第 2.3 节中给出.

1.3 Boolean算法简介

Boolean 方法是将问题集合簇编码为布尔表达式,然后应用给定的规则对表达式进行计算,得到碰集,再应用布尔代数中的吸收律得到所有极小碰集(与子集检测原理一致).具体规则和布尔表达式计算原理见文献[8],在此不做介绍.其算法实现可以递归表示为如下过程.

算法 2. Boolean.

Function: *Boolean*(F',*E*,*allMHS*)

Input: 集合簇 F',候选解 *E*,极小碰集簇 *allMHS*,其中,F'是问题集合簇 $F=\{S_1,S_2,\ldots,S_n\}$的子集;

Output: *allMHS*.

Begin

1. **if** ($F'==\varnothing$)
2. {
3. *Subset_Checking*(*E*,*allMHS*);
4. **return**;
5. }
6. **if** ($\exists e(\{e\}\in F')$)
7. *Boolean*($\{S_i|S_i\in F'\wedge e\notin S_i\}$,$E\cup\{e\}$,*allMHS*);
8. **elseif** ($\exists e\forall S_i(S_i\in F'\rightarrow e\in S_i)$)
9. {
10. *Subset_Checking*($E\cup\{e\}$,*allMHS*);
11. *Boolean*($\{S_i|S_i\in F'\wedge e\notin S_i\}\cup\{S_i-\{e\}|S_i\in F'\wedge e\in S_i\}$,*E*,*allMHS*);
12. }
13. **else**
14. {
15. $e\leftarrow Select\_element\left(\bigcup_{S_i\in F'}S_i\right)$;
16. *Boolean*($\{S_i|S_i\in F'\wedge e\notin S_i\}$,$E\cup\{e\}$,*allMHS*);
17. *Boolean*($\{S_i|S_i\in F'\wedge e\notin S_i\}\cup\{S_i-\{e\}|S_i\in F'\wedge e\in S_i\}$,*E*,*allMHS*);
18. }

End

首次调用该算法前需对传入参数进行如下设置.

1. $F'\leftarrow F$;
2. 将 *E* 及 *allMHS* 置为空;

为叙述清晰,我们对原 Boolean 方法做了一些小的改动,如将子集检测放入了递归过程中(第 3 行、第 10 行),但算法整体原理是保持不变的.

若 F'为空,则说明参数传入的候选解 *E* 已经是碰集,对其进行检测并返回(第 1 行,也是递归出口).否则,就需

要选择元素对候选解进行扩充:若 F'中含有单元素构成的集合,则将该元素加入 E 中,同时从 F'中删除包含该元素的集合,继续递归计算(第 6 行);否则,若 F'中所有集合都含有某一元素 e,则说明 $E\cup\{e\}$是碰集,对其进行检测,之后从 F'包含的所有集合中删去元素 e,继续递归计算(第 8 行);若前两个条件都不满足,则从 F'包含的所有集合的并集中挑选一个元素,以此将 F'表示的问题分解为 2 个解空间不重叠的子问题,继续递归求解(第 13 行).其中,第 15 行挑选元素所使用的规则可以自定义.最常使用的挑选规则是选择被 F'中被最多集合包含的元素,本文的算法 2 和算法 5 都依此对元素进行选择.

2 基于元素独立覆盖度的候选解极小性判定方法

本节将首先给出极小候选解的相关概念及定义,并提出一种基于元素独立覆盖度检测的碰集极小性判定方法.在此基础上,深入分析增量求解过程中非极小碰集的产生原因,指出可将极小性判定提前到候选解产生阶段,并给出相关方法.最后,将所提方法与 Boolean 方法相结合.

2.1 相关定义

设有集合簇 $F=\{S_1,S_2,\dots,S_n\}$,$E=\{e_1,e_2,\dots,e_m\}$为 F 的候选解,其中,$e_k\in\bigcup_{S_i\in F}S_i$ $(1\leqslant k\leqslant m)$.现给出如下 3 个定义.

定义 4. 若 $E\cap S_j=\{e_k\}(1\leqslant j\leqslant n,1\leqslant k\leqslant m)$,则称在 E 中,元素 e_k 独立覆盖了集合 S_j.e_k 独立覆盖 F 中集合的个数,称为该元素在候选解 E 中的独立覆盖度.

定义 5. 设 $T_E=\{S_i|S_i\in F\wedge E\cap S_i\neq\varnothing\}$,称 T_E 为 E 对于 F 的覆盖集合簇.

定义 6. 若$\{W|W\subsetneq E\wedge T_W=T_E\}=\varnothing$,则称 E 为极小候选解.

对于例 1 中给出的集合簇 F,候选解{1,2,3,4}中,元素 2~元素 4 的独立覆盖度都为 2,元素 1 的独立覆盖度为 0.$T_{\{1,2,3,4\}}$={1,2,8},{1,2,9},{1,3,10},{1,3,11},{1,4,12},{2,13},{2,14},{3,15},{3,16},{4,17},{4,18}}.同时有候选解{2,3,4},使得 $T_{\{2,3,4\}}=T_{\{1,2,3,4\}}$成立,且{2,3,4}是{1,2,3,4}的真子集,所以{1,2,3,4}不为极小候选解.

下面给出定理 1,对候选解是否极小的判定条件进行说明.

定理 1. 设 E 是集合簇 F 的候选解,E 不是极小候选解的充分必要条件为:E 中含有独立覆盖度为 0 的元素.

证明:

- 充分性:假设 e 在 E 中独立覆盖度为 0,则 e 覆盖的集合簇必被 E 中其他元素覆盖,即 $T_{\{e\}}\subseteq T_{E-\{e\}}$,因此存在 $W=E-\{e\}$,使得 $T_W=T_E$ 成立,根据定义 6,E 不为极小候选解.
- 必要性(反证法):假设 E 中不含有独立覆盖度为 0 的元素.由于 E 不为极小候选解,其必有真子集 W 使得 $T_W=T_E$ 成立,设 $e\in E-W$,去掉 e 并未改变候选解的覆盖集合簇,因此有 $T_{\{e\}}\subseteq T_{E-\{e\}}$,即,$e$ 的独立覆盖度为 0.与假设矛盾,假设不成立.

证毕. □

2.2 基于元素独立覆盖度检测的候选解极小性判定方法

本节将给出一种基于元素独立覆盖度检测的候选解极小性判定方法.为方便理解以及与第 1.2 节中介绍的子集检测方法进行比较,首先给出这种方法对给定碰集极小性进行判定的过程,然后通过分析增量求解过程中非极小碰集产生的原因,对该过程进行优化,使其在增量求解过程中保持较高效率.

2.2.1 基于元素独立覆盖度的碰集极小性判定方法

碰集显然也是候选解,因此由定理 1,我们可以得到一种判别给定碰集极小性的新方法:对给定碰集中所有元素的独立覆盖度进行计算,若其中含有独立覆盖度为 0 的元素,则说明该碰集不是极小碰集.下面给出这种方法的伪码表示.

算法 3. ICC (independent coverage checking).

Function: *ICC*(*H*,*F*,*allMHS*)

Input: 碰集 $H=\{e_1,e_2,\dots,e_m\}$,问题集合簇 $F=\{S_1,S_2,\dots,S_n\}$,极小碰集簇 *allMHS*;

Output: *allMHS*.

Begin

```
1.  for (i=1; i≤n; i++)
2.      configuration[i]←|H∩Si|;   //统计 Si 被 H 中多少个元素覆盖
3.  for (i=1; i≤m; i++)
4.  {
5.      for (j=1; j≤n; j++)
6.      {
7.          if (ei∈Sj∧configuration[j]==1)   //说明 ei 独立覆盖 Sj
8.              break;
9.          if (j==n)   //说 ei 的明独立覆盖度为 0
10.             return;
11.     }
12. }
13. allMHS←allMHS∪{H};
```

End

算法中首先对集合簇 F 中各集合被 H 中元素覆盖情况进行统计,结果记录在数组 *configuration* 中(第 1 行).然后对 H 中各元素的独立覆盖度进行检测:若 H 中某一元素至少独立覆盖集合簇 F 中的 1 个集合,则说明该元素的独立覆盖度不为 0(第 5 行);反之,则说明该元素独立覆盖度为 0,H 不是极小碰集(第 7 行).若 H 中所有元素的独立覆盖度都不为 0,则 H 为极小碰集,将其加入到中 *allMHS*.

我们可以发现:算法 3 的执行过程中无需对极小碰集簇进行遍历比较,因此其效率与极小碰集簇的大小无关.下面,本文将通过实例分析,指出增量求解过程中非极小碰集产生的原因,并以此对算法 3 进行优化,提出一种增量的候选解极小性判定方法.

2.2.2 增量求解过程中非极小碰集的产生原因

现有的完备方法中普遍使用增量求解,即是按照一定规则,将候选解中不包含的元素加入其中,直到候选解满足一定条件再对其进行下一步操作.

现有的极小碰集增量求解算法是将条件设置为候选解成为碰集,之后再使用子集检测方法保证最终结果的极小性.在候选解成为碰集前,一般不对其极小性进行检测.算法中所用的优化策略,通常也都是以通过选择适当的扩充元素,使候选解尽快成为碰集为目的.

对于例 1 中给出的集合簇 F,若使用算法 2 介绍的 Boolean 方法对其所有极小碰集进行求解,便会在求解过程中首先得到碰集{1,2,3,4,5,6,7},并将其暂时保存在极小碰集簇中(此前极小碰集簇为空,不包含该碰集的子集).通过第 2.1 节中给出的实例,我们知道{1,2,3,4}并不是极小候选解,因此{1,2,3,4,5,6,7}也不是极小碰集.

定理 2. 在增量求解过程中,非极小候选解的产生不滞后于碰集的产生.

证明:反证法.假设存在问题集合簇 F,在使用增量方法对其进行求解的过程中,可得到极小碰集 *MHS*,并会在得到 *MHS* 后继续向其中加入元素生成非极小候选解 E,即非极小候选解的产生滞后于碰集的产生.这与增量求解规则相矛盾,增量求解方法不会在得到一个碰集后继续向其中加入元素,假设不成立.证毕. □

而非极小候选解通过加入元素生成的碰集显然也不是极小碰集.由此可见,增量求解过程中非极小碰集的产生原因主要有两种:(1) 最后加入的元素使得候选解的极小性发生了改变;(2) 该碰集是某个已产生的非极小候选解的超集.若能在极小碰集增量求解算法运行过程中尽早地对非极小候选解进行丢弃,防止其超级的生成,必将减少很多不必要的计算.

一种简单的方法是:将 *configuration* 数组作为候选解的一个属性保存下来,每当有新元素加入到候选解中,便对数组进行相应的修改,之后再使用算法 3 中的第 3 行~第 9 行对当前候选解的极小性进行判定,对于非极小

候选解直接丢弃.但通过进一步分析我们会发现:加入新元素后,并不需要重新检测候选解中所有元素的独立覆盖度,只需要对其中一部分元素的独立覆盖度进行操作,即可对新生成候选解的极小性给出判定结果.

2.2.3 增量求解过程中候选解极小性的判定方法

定理 3. 设问题集合簇 F 有极小候选解 E,将某元素 e 加入 E 后得到 E',$F'=\{S_i|S_i\in F\wedge S_i$ 被 E 中元素独立覆盖$\}$,$P=\{e_j|e_j\in E\wedge e_j\in S_i\wedge S_i\in F'\}$.当且仅当存在属于 P 的元素在 e 加入候选解 E 后,其独立覆盖度变为 0 时,E'为非极小候选解.

也就是说,将元素 e 加入极小候选解 E 后,只需对原候选解中独立覆盖的集合中包含 e 的元素进行独立覆盖度检测(对于不能保证新加入元素独立覆盖度非 0 的算法,也需要对新加入元素的独立覆盖度进行检测),便能够对新候选解的极小性进行判定.

证明:根据定理 1,若 E'为非极小候选解,则说明有元素的独立覆盖度由于元素 e 的加入变为了 0.同时,根据定义 4,若一个元素的独立覆盖度发生了变化,则说明该元素在 E 中独立覆盖的部分或全部集合中包含元素 e.而对于那些独立覆盖的全部集合中都不包含 e 的元素,独立覆盖度不会发生变化.故,只需对独立覆盖度发生变化元素进行检测的即可确定加入元素 e 后候选解 E'的极小性.证毕. □

对于例 1 中给出的集合簇 F,{1,2,3}是 F 的一个极小候选解.其中:元素 1 独立覆盖的集合簇为{{1,4,12}},独立覆盖度为 1;元素 2 独立覆盖的集合簇为{{2,13},{2,14}},独立覆盖度为 2;元素 3 独立覆盖的集合簇为{{3,15},{3,16}},独立覆盖度为 2.当元素 4 加入此候选解时,由于元素 2、元素 3 各自独立覆盖的集合内不包含元素 4,这 2 个元素的独立覆盖度自然不会发生变化;而{1,4,12}内包含元素 4,其已不被元素 1 独立覆盖,需对元素 1 的独立覆盖度减 1,得到新的独立覆盖度为 0,此时候选解非极小.可见,元素 4 的加入使得元素 1 的独立覆盖度变为 0,进而导致了候选解极小性的改变.算法 4 给出了这种方法的伪码表示以及其中使用数据结构的说明.

算法 4. IICC (incremental independent-coverage-checking).

Function: *IICC*(E,e_{m+1},F,*CS_attribute*)

Input: 极小候选解 $E=\{e_1,e_2,\ldots,e_m\}$,新加入元素 e_{m+1},问题集合簇 $F=\{S_1,S_2,\ldots,S_n\}$,候选解属性 *CS_attribute*,其包含 *set_coverage*,*independently_covered_with*,*element_independent_coverage* 这 3 个一维数组;

Output: *trueorfalse*, *CS_attribute*.

```
Begin
1.   for (i=1; i≤n; i++)
2.     if (e_{m+1}∈S_i)
3.       if (set_coverage[i]==0)
4.       {
5.         independently_covered_with[i]←m+1;
6.         set_coverage[i]←1;
7.         element_independent_coverage[m+1]++;
8.       }
9.       else if (set_coverage[i]==1)
10.      {
11.        set_coverage[i]←2;
12.        element_independent_coverage[independently_covered_with[i]]--;
13.        if (element_independent_coverage[independently_covered_with[i]]==0)
14.          return false;
15.      }
16. return true;
End
```

首先对算法 4 中使用的一个新数据结构——候选解属性 *CS_attribute* 进行说明.其包含 3 个一维数组,分别是 *set_coverage*,*independently_covered_with* 以及 *element_independent_coverage*.*set_coverage*[*i*]用来记录集合 S_i 被候选解 E 中的元素覆盖的情况,为 0 表示未被覆盖,为 1 表示被独立覆盖,为 2 表示被多次覆盖.若 *set_coverage*[*i*]为 1,则 *independently_covered_with*[*i*]记录了候选解 E 中哪个元素独立覆盖了 S_i,比如:*set_coverage*[*i*]为 1 的同时 *independently_covered_with*[*i*]的值为 *j*,表示 S_i 被元素 e_j 独立覆盖.*element_independent_coverage*[*j*]则表示在候选解 E 中,元素 e_j 独立覆盖集合簇 F 中集合的个数,即 e_j 的独立覆盖度.

算法 4 在运行过程中,会根据 *set_coverage*[*i*]中记录的不同分别进行如下操作:若此前 S_i 未被 E 中的元素覆盖(第 3 行),则将其记录为被 e_{m+1} 独立覆盖(第 5 行、第 6 行),同时,将 e_{m+1} 的独立覆盖度加 1(第 7 行);若 S_i 被 E 中元素 e_j 独立覆盖(第 9 行),则加入 e_{m+1} 后,S_i 不被 E 中任何元素独立覆盖(第 11 行),需将 e_j 的独立覆盖度减 1(第 12 行),此过程中如果发现 e_j 的独立覆盖度变为 0(第 13 行),则说明在 E 中加入元素 e_{m+1} 形成的新候选解非极小,直接返回.

在极小碰集增量求解过程中使用算法 4 给出的 IICC 方法,会在每次对候选解进行扩充时,都对其加入新元素后的极小性进行快速判定,以期尽快发现并丢弃非极小候选解,减少非极小碰集的生成.

从算法 4 中我们不难看出:IICC 算法对一个极小碰集中某个元素进行独立覆盖度检测的上限为在其之后加入该候选解的元素个数,对于包含 n 个元素的极小碰集就是$(n-1)n/2$.其只与自身有关,与整个解集的结构无关.而对于子集检测方法而言,当一个碰集产生时,需要将其与当前解集中所有碰集进行比较,最坏情况下的比较次数是整个解集的大小.通常情况下,后者都要比前者高出数个数量级.

在随后的第 2.3 节中,会给出结合 IICC 的 Boolean 方法,并通过实例对比说明优化效果.

2.3 结合元素独立覆盖度检测的Boolean算法

算法 5. BWIICC (Boolean with IICC).

Function: *BWIICC*(F,F',E,e',*CS_attribute*,*allMHS*)

Input: 原始问题集合簇 $F=\{S_1,S_2,\ldots,S_n\}$,中间过程集合簇 F',候选解 E,新加入元素 e',候选解属性 *CS_attribute*,极小碰集簇 *allMHS*,其中,F'是 F 的子集;

Output: *allMHS*.

Begin

1. **if** ($e'\neq NULL$)
2. {
3. **if** (*IICC*(E,e',F,*CS_attribute*)==false)
4. **return**;
5. **else**
6. $E\leftarrow E\cup\{e'\}$;
7. }
8. **if** ($F==\varnothing$)
9. {
10. $allMHS\leftarrow allMHS\cup\{E\}$;
11. **return**;
12. }
13. **if** ($\exists e(\{e\}\in F')$)
14. *BWIICC*(F,$\{S_i|S_i\in F'\wedge e\notin S_i\}$,$E$,$e$,*CS_attribute*,*allMHS*);
15. **elseif** ($\exists e\forall S_i(S_i\in F'\rightarrow e\in S_i)$)
16. {
17. *BWIICC*(F,$\{S_i|S_i\in F'\wedge e\notin S_i\}\cup\{S_i-\{e\}|S_i\in F'\wedge e\in S_i\}$,$E$,*NULL*,*CS_attribute*,*allMHS*);

18. **if** (*IICC*(*E*,*e*,*F*,*CS_attribute*)==false)
19. **return**;
20. **else**
21. *allMHS*←*allMHS*∪{*E*∪{*e*}};
22. }
23. **else**
24. {
25. $e \leftarrow Select\_element\left(\bigcup_{S_i \in F'} S_i\right)$;
26. *BWIICC*(*F*,{S_i|S_i∈*F*′∧*e*∉S_i},*E*,*e*,*CS_attribute*,*allMHS*);
27. *BWIICC*(*F*,{S_i|S_i∈*F*′∧*e*∉S_i}∪{S_i−{*e*}|S_i∈*F*′∧*e*∈S_i},*E*,*NULL*,*CS_attribute*,*allMHS*);
28. }

End

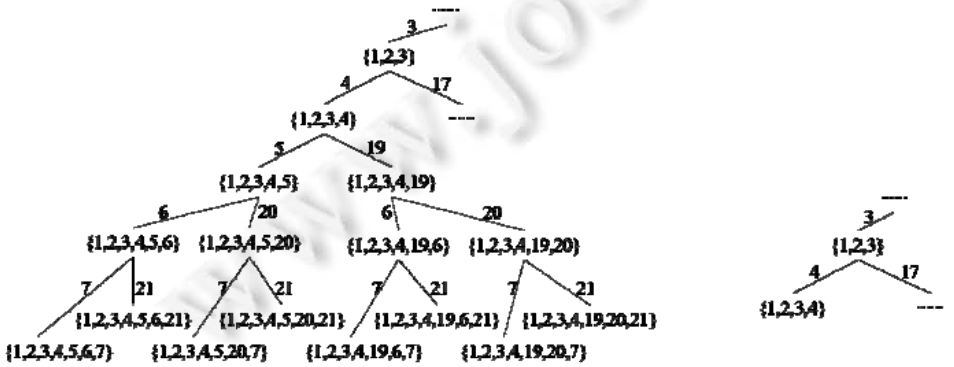
首次调用该算法前需对传入参数进行如下设置.

1. *F*′←*F*;
2. 将 *E* 及 *allMHS* 置为空;
3. 将 *e*′置为 *NULL*;
4. 将 *CS_attribute* 中各数组所有位都置为 0;

该算法与算法 2 的主要不同在于:算法 5 每次对候选解 *E* 进行扩充前,都会对扩充后新候选解的极小性进行检测(第 2 行、第 16 行),以此保证算法执行过程中只生成极小候选解.这样做不仅剥离了碰集极小性判定耗时与极小碰集簇大小间的关系,使得在极小碰集簇较大时算法仍能以较高的效率对候选解的极小性进行判定,而且由于 IICC 方法能够及早地发现非极小候选解并将其丢弃,也为算法 5 提供了额外的剪枝效果.

下面通过图 1 给出在对例 1 集合簇 *F* 的极小碰集进行求解过程中,算法 2 和算法 5 对部分候选解空间访问情况的对比.

图 1 中,各节点表示会被访问到的候选解,边表示自上而下新加入候选解的元素.图 1(a)为算法 2 访问的部分候选解空间,图 1(b)是算法 5 对于相同空间的访问情况.算法 5 在发现{1,2,3,4}不是极小候选解后,会立将其丢弃,停止对其超集的访问;而算法 2 会继续向其中添加元素,直到其成为碰集.但图 1 所示的候选解空间实际上是没有极小碰集产生的,对于类似情况,算法 5 提前剪枝所带来的收益是显而易见的.



(a) Boolean (b) BWIICC

Fig.1 Visit-Situation-Comparison of the two methods forpart of candidate-solution-spaceinexample one

图 1 2 种方法对例 1 部分候选解空间的访问情况比较

由于 BWIICC 方法在搜索阶段仍使用 Boolean 方法的规则对节点进行扩充,只有在节点所代表的候选解非极小时才会停止对当前结点的继续扩充而返回上一层,因此,BWIICC 方法在整个求解过程中所遍历的全部节点是 Boolean 方法的子集,其正确性由 Boolean 方法保证.同时,由于非极小候选解的超集不可能是极小碰集,因此对比 Boolean 方法,使用 IICC 的 BWIICC 方法所剪枝掉的节点中不可能包含极小碰集,即,使用 IICC 后,Boolean 方法的完备性不会被破坏.

在第 3 节中,我们将会以多组实验数据对比的方式,分析说明算法 5 相较于算法 2 的优势.

3 实验分析

首先,将使用算法 5 给出的 BWIICC 方法与算法 2 介绍的 Boolean 方法进行比较,给出两种方法在随机生成测试用例下的实验结果.然后,会单独使用 BWIICC 方法对另外几组难度较高的测试用例进行求解,以测试这种方法对较难问题的求解能力.

实验平台如下:Windows 10 操作系统,CPU Intel Core i5-3470 3.2Ghz,8.00GB RAM,C++.

实验所用测试用例为随机生成器产生,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 8 小组,分为对比组和较难问题组两部分.每小组元素个数固定,对比组分别为 15,20,25,30,较难问题组分别为 35,40,45,50,各小组均包含 p 取值 0.05~0.94 的 19 个用例,所有用例集合簇中集合个数均为 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.

图 2 给出了两种方法对于对比组中所有实验用例运行时间的对比情况.

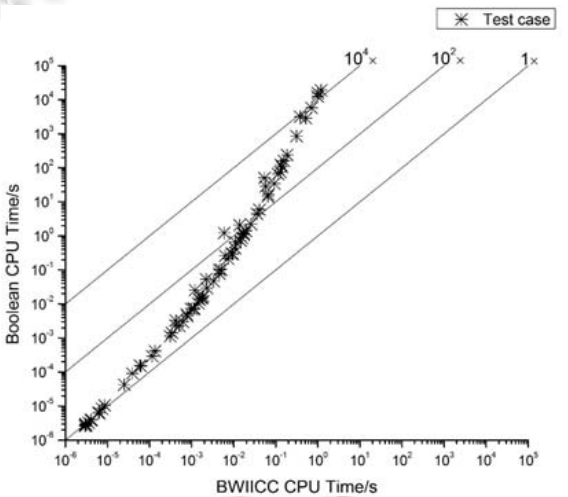


Fig.2 Experimental comparisons of BWIICC and Boolean

图 2 BWIICC 方法与 Boolean 方法实验结果比较

我们可以看出:对于耗时较少的用例,两种方法所用时间十分接近,用例点基本都落在 1 倍对比线附近;但随着用例耗时的上升,BWIICC 方法的优势逐步显现;尤其对于那些 Boolean 方法耗时在 1s 以上的用例,多数用例点已经处于 10^2 倍对比线上方,部分耗时较高的用例点甚至位于 10^4 倍对比线之上.接下来,我们将会通过实验数据的分组比较,进一步分析产生这种现象的原因.

图 3 是将两种方法的实验数据按用例元素数分组后得到的结果,其中,各子图的横坐标轴表示元素出现的概率 p,*MHS* 表示对应实例的极小碰集数量,与右侧纵轴相关.从图中我们可以看出:各子图中,对于大多数测试用例,产生极小碰集的数量越多,两种方法在时间效率上的差距也就越大;而对于产生极小碰集较少的用例,差距则并不明显.这主要是由于 Boolean 算法中使用的子集检测方法对极小碰集簇的大小较为敏感导致.同时,产生极小碰集的数量越多,一般也意味着用例的难度较高,算法耗时也会相应增加,这与图 2 中体现的趋势是一致

的.除此之外我们还可以发现,各子图中 Boolean 与 *MHS* 两条折线图拟合的并不是很好.这说明对元素个数相同的一组用例,出现概率 p 的不同会导致 Boolean 算法对于单一极小碰集的平均求解时间产生较大波动.

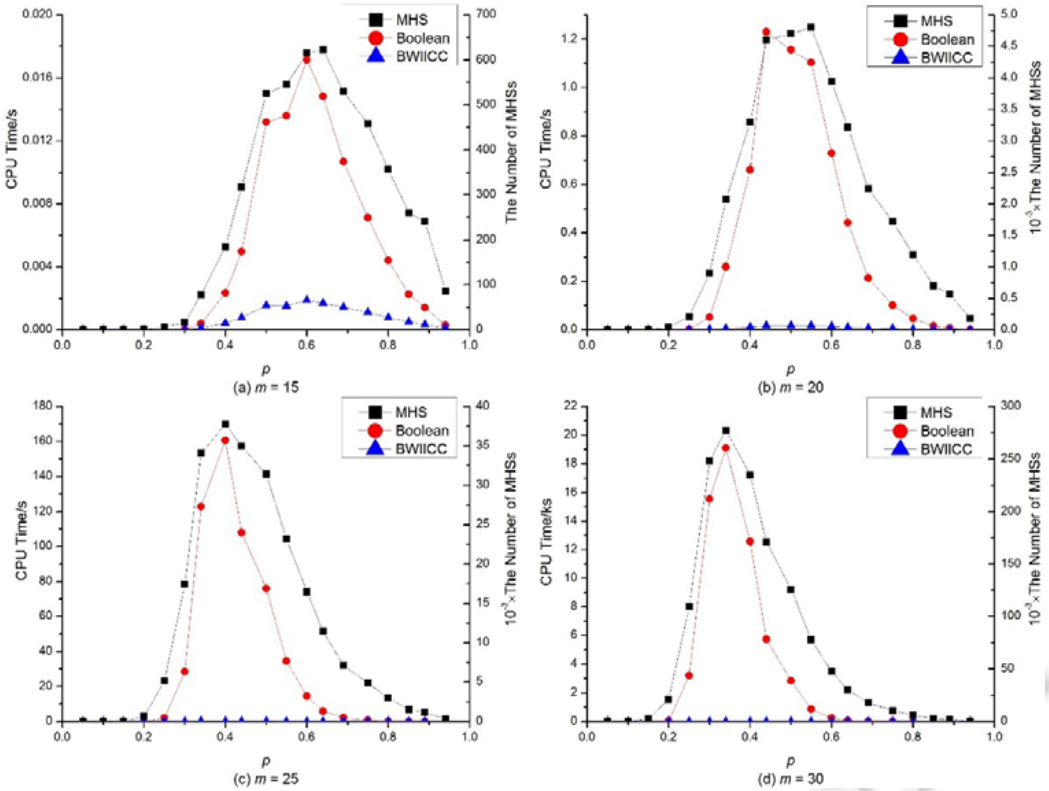


Fig.3 Experimental comparisons of the two methods based on diverse number of elements

图 3 不同元素数实例分组下 2 种方法实验结果比较

图 4 为两种方法对于各实例求解过程中访问总分支数情况的对比.分支数即是在形同图 1 的树形表示下,算法访问总节点对应的二叉树中所包含叶节点数量.其中,各子图横坐标轴表示含义与图 3 一致,纵坐标轴表示算法对于各实例求解过程中访问总分支数与该实例解中实际包含的极小碰集数的比值.以此,我们可以直观地看出使用 IICC 为 BWIICC 所带来的剪枝收益以及两种方法实际访问总分支数与理论下界(极小碰集数)的比较情况.同样是对于较难的实例,IICC 所带来的剪枝效果更加明显.同时我们还能发现:虽然对于各组实例,BWIICC 访问的总分支数都明显少于 Boolean,但两者间的差距并没有图 3 中体现出的时间差那样巨大.这说明除去剪枝收益,IICC 方法在候选解极小性判定上为使用其的 BWIICC 算法带来了更为巨大的时间效率收益.

图 5 给出了 BWIICC 方法对于较难问题组测试用例的实验结果,其中,各坐标轴表示含义与图 3 中一致.

从图 5(a)中我们可以看到,算法对其中难度较高的测试用例求得的极小碰集数已达到 1.8×10^6 个以上.而随着元素数的增加,图 5(d)中难度最高测试用例产生的极小碰集数更是超过 6×10^8 个.但 BWIICC 方法仍可以将求解时间控制在 3000s 以内,可见,BWIICC 方法对于较高难度的问题也有着不错的求解效率.

同时,对比图 3 各子图中 Boolean 与 *MHS* 折线图的拟合情况,图 5 中 BWIICC 与 *MHS* 折线图的拟合度显然更高.这说明对于含有相同元素个数但出现概率 p 不同的一组用例,BWIICC 方法的单一极小碰集平均求解时间较 Boolean 方法更为稳定,随问题难度的波动幅度更小.

通过以上实验对比分析,我们得到以下结论:使用 IICC 的 BWIICC 方法相较使用子集检测的 Boolean 方法,求解效率有着显著的提升;同时,对于不同难度实例的单一极小碰集平均求解时间,BWIICC 方法也更为稳定.

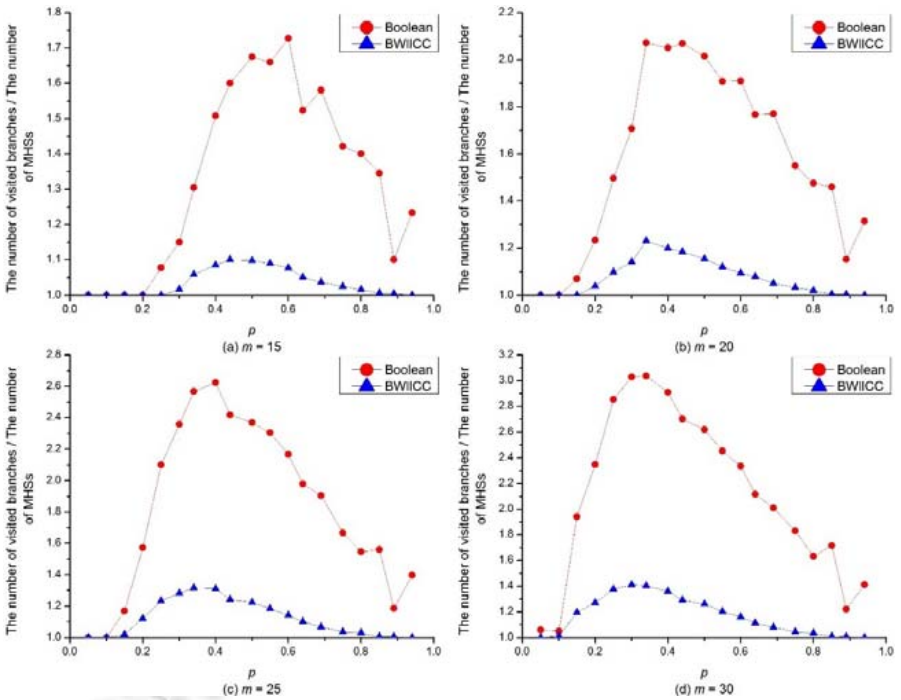


Fig.4 Experimental comparisons of the number of visited branches of the two methods

图 4 两种方法访问分支数对比

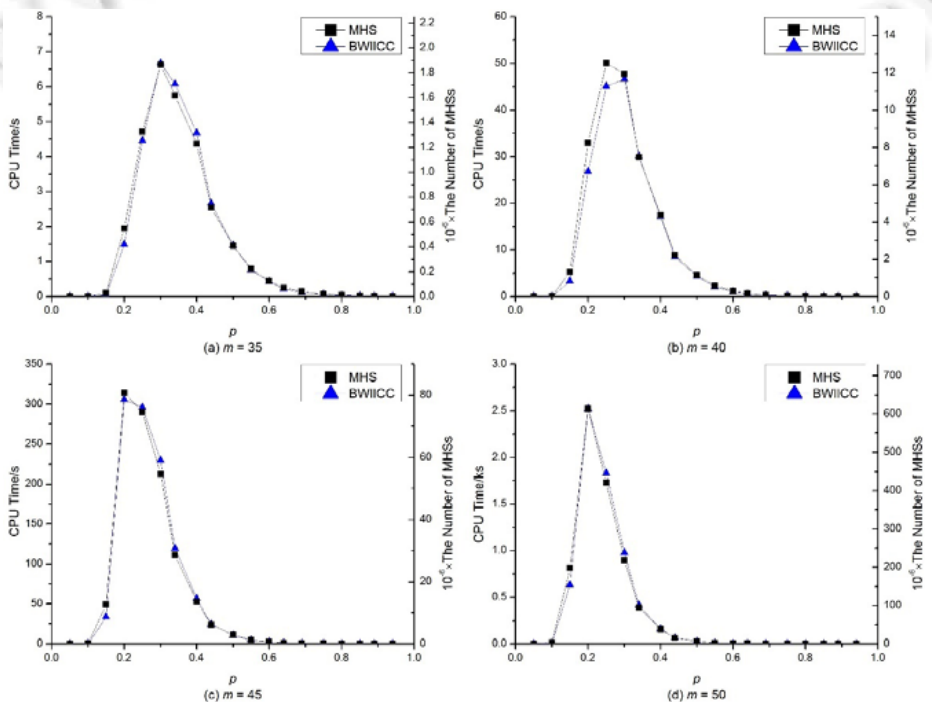


Fig.5 Experimental data of BWIICC for the difficult problems based on diverse number of elements

图 5 BWIICC 方法对于较难问题组实例的实验数据

4 结束语

极小碰集问题在许多重要领域都有着广泛的应用,现有的极小碰集算法主要通过使用子集检测方法来保证最终求得碰集的极小性,但这种方法的效率会受到极小碰集簇大小影响,极小碰集簇中碰集越多,子集检测的耗时也就越高.对此,本文首先提出了基于元素独立覆盖度检测的碰集极小性判定方法 ICC,剥离了碰集极小性判定效率与极小碰集簇大小的相关性.随后,对增量求解过程中非极小碰集的产生原因进行了深入分析,并以此对 ICC 方法进行优化,得到 IICC 方法.该方法可以增量地对候选解的极小性给出判定结果,在提升碰集极小性判定效率的同时,还能够通过尽早发现并丢弃非极小候选解而获得额外的剪枝效果,进一步提升了极小碰集求解算法整体的执行效率.最后,将 IICC 方法与当前求解效率较高的 Boolean 算法相结合,得到 BWIICC 算法,并通过实验对比检测优化效果.实验结果表明:优化后的 BWIICC 算法相较于 Boolean 算法,整体效率提升明显,且优化效果有随问题难度上升而上升的趋势.对于对比组中难度最高的实例,效率提升可达 4 个数量级以上.同时,通过对较难问题组实例的测试,显示出 BWIICC 算法在问题难度较高时,依然可以保持不错的求解效率.此外,对于单一极小碰集的平均求解时间,BWIICC 算法也更为稳定.

致谢 本文作者对所有匿名审稿人的辛勤工作和宝贵意见表示真诚的感谢.

References:

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

[2] Yu Q, Li CQ, Shen YM, Wang J. Method of solving abductive reasoning problem via hitting set. Ruan Jian Xue Bao/Journal of Software, 2015,26(8):1937−1945 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4694.htm [doi: 10.13328/j.cnki.jos.004694]

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

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

[5] Greiner R, Smith BA, Wilkerson RW. A correction to the algorithm in Reiter's theory of diagnosis. Artificial Intelligence, 1989, 41(1):79−88.

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

[7] Jiang YF, Lin L. Computing the minimal hitting set with binary HS-tree. Ruan Jian Xue Bao/Journal of Software, 2002,13(12): 2267−2274 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/13/2267.htm

[8] Jiang YF, Lin L. The computation of hitting sets with Boolean formulas. Chinese Journal of Computers, 2003,26(8):919−924 (in Chinese with English abstract).

[9] Ouyang DT, Ouyang JH, Cheng XC, Liu J. A method of computing hitting set in model-based diagnosis. Chinese Journal of Science Instrument, 2004,25(4):605−608 (in Chinese with English abstract).

[10] Zhao XF, Ouyang DT. A method of combing SE-tree to compute all minimal hitting sets. Progress in Natural Science, 2006,16(2): 169−174.

[11] Chen XM, Meng XF, Qiao RX. Method of computing all minimal hitting set based on BNB-HSSE. Chinese Journal of Science Instrument, 2010,31(1):61−67 (in Chinese with English abstract).

[12] Zhang LM, Ouyang DT, Zeng HL. Computing the minimal hitting set based on dynamic maximum degree. Journal of Computer Reach and Development, 2011,48(2):209−215 (in Chinese with English abstract).

[13] Pill I, Quaritsch T. Optimizations for the Boolean approach to computing minimal hitting sets. In: Proc. of the 20th European Conf. on Artificial Intelligence. Amsterdam: IOS Press, 2012. 648−653.

[14] Pill I, Quaritsch T. RC-Tree: A variant avoiding all the redundancy in Reiter's minimal hitting set algorithm. In: Proc. of the 2015 IEEE Int'l Symp. on Software Reliability Engineering Workshops. 2015. 78−84.

[15] Wang YY, Ouyang DT, Zhang LM, Zhang YG. A method of computing minimal hitting sets using CSP. Journal of Computer Reach and Development, 2015,52(3):588−595 (in Chinese with English abstract).

[16] Lin L, Jiang YF. Computing minimal hitting sets with genetic algorithm. In: Proc. of the 13th Int'l Workshop on Principles of Diagnosis. 2002. 77−80.

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

[18] Huang J, Chen L, Zou P. Computing minimal diagnosis by compounded genetic and simulated annealing algorithm. Ruan Jian Xue Bao/Journal of Software, 2004,15(9):1345−1350 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1345.htm

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

[20] Zhou G, Feng WQ, Jiang BF, *et al*. Computing minimal hitting set based on immune genetic algorithm. Int'l Journal of Modelling, Identification and Control, 2014,21(1):93−100.

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

[22] Zhao XF, Ouyang DT. Deriving all minimal hitting sets based on join relation. IEEE Trans. on Systems, Man, and Cybernetics: Systems, 2015,45(7):1063−1076.

[23] Pill I, Quaritsch T, Wotawa F. From conflicts to diagnoses: An empirical evaluation of minimal hitting set algorithms. In: Proc. of the 22nd Int'l Workshop on the Principles of Diagnosis. 2011. 203−210.

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

附中文参考文献:

[2] 余泉,李承乾,申宇铭,王驹.溯因推理问题的碰集求解方法.软件学报,2015,26(8):1937−1945. http://www.jos.org.cn/1000-9825/4694.htm [doi: 10.13328/j.cnki.jos.004694]

[7] 姜云飞,林笠.用对分 HS-树计算最小碰集.软件学报,2002,13(12):2267−2274. http://www.jos.org.cn/1000-9825/13/2267.htm

[8] 姜云飞,林笠.用布尔代数方法计算最小碰集.计算机学报,2003,26(8):919−924.

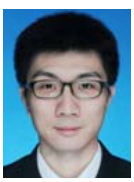
[9] 欧阳丹彤,欧阳继红,程晓春,刘杰.基于模型诊断中计算碰集的方法.仪器仪表学报,2004,25(4):605−608.

[11] 陈晓梅,孟晓风,乔仁晓.基于 BNB-HSSE 计算全体碰集的方法.仪器仪表学报,2010,31(1):61−67.

[12] 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.计算机研究与发展,2011,48(2):209−215.

[15] 王艺源,欧阳丹彤,张立明,张永刚.利用 CSP 求解极小碰集的方法.计算机研究与发展,2015,52(3):588−595.

[18] 黄杰,陈琳,邹鹏.一种求解极小诊断的遗传模拟退火算法.软件学报,2004,15(9):1345−1350. http://www.jos.org.cn/1000-9825/15/1345.htm



刘思光(1988－),男,吉林省吉林市人,硕士,主要研究领域为基于模型诊断,极小碰集求解.

欧阳丹彤(1968－),女,博士,教授,博士生导师,CCF 高级会员,主要研究领域为基于模型诊断.

张立明(1980－),男,博士,高级工程师,CCF 专业会员,主要研究领域为基于模型诊断,SAT.