第48卷第1期 中国科学技术大学学报 Vol.48,No.1
2018年1月 JOURNAL OF UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA Jan. 2018

文章编号:0253-2778(2018)01-0028-07

基于故障森林的组合测试故障定位研究

王 勇[1,2],黄志球[1],韦良芬[3],卢桂馥[2]

(1.南京航空航天大学计算机科学技术学院,江苏南京 210000;
2.安徽工程大学计算机与信息学院,芜湖 241000;3.安徽三联学院计算机工程系,安徽合肥 230601)

摘要:组合测试作为一种对参数组合空间抽样的系统方法,适用于待测系统中存在由特定参数组合所引发的软件失效.依据组合测试结果,定位出最小失效诱因模式(minimal failure-causing schema,MFS)有助于程序员进行故障源检测与修复.然而,组合测试可能存在mask effect,使得测试用例中即使包含MFS也未必一定触发软件失效.因此,在存在mask effect的系统中精确定位最小失效诱因模式尤为困难.为此提出了一种基于故障森林的组合测试故障定位方法.给定一个t-路组合测试集($t \geqslant 2$)及其附加测试集,该方法首先学习由多个深度为t的基本故障分类树所组成的故障森林,然后从故障森林中提取基本故障组合模式,最后将可疑MFS进行排序,并提交给程序员进行进一步诊断.仿真实验结果表明,该方法能有效定位系统中存在的组合故障模式.特别地,对于存在mask effect的待测系统,故障定位结果健壮.

关键词:组合测试;故障定位;故障森林;最小失效诱因模式

中图分类号:TP311 **文献标识码**:A doi:10.3969/j.issn.0253-2778.2018.01.004

引用格式:王勇,黄志球,韦良芬,等.基于故障森林的组合测试故障定位研究[J].中国科学技术大学学报,2018,48(1):28-34.

WANG Yong, HUANG Zhiqiu, WEI Liangfen, et al. Locating failure-inducing combinations based on fault forest[J]. Journal of University of Science and Technology of China, 2018,48(1):28-34.

Locating failure-inducing combinations based on fault forest

WANG Yong[1,2], HUANG Zhiqiu[1], WEI Liangfen[3], LU Guifu[2]

(1. *College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics Nanjing* 210000, *China*;
2. *College of Computer and Information, Anhui Polytechnic University Wuhu* 241000, *China*;
3. *Department of Computer Engineering, Anhui SanLian University Hefei* 230601, *China*)

Abstract: Combinatorial testing, a method for sampling parameter combination in the parameter space of a system, is suitable for systems in which failure is caused by a specific parameter combination. Based on the results of combination testing, locating the minimal failure causing schema (MFS) can help programmers to localize faults and repair them. However, combination testing might be affected by the mask effect, and even test cases containing MFSs may not necessarily trigger a failure. Therefore, it is extremely difficult to pinpoint MFSs in systems affected by the mask effect. A fault location method based on fault forest is proposed. Given a set of t-way combination test ($t \geqslant 2$) and their augment test set, this method first learns some basic fault trees which generate a fault forest, then extracts the basic suspicious MFS from the

收稿日期:2017-05-16;**修回日期**:2017-06-22

基金项目:国家高技术研究发展计划(863)(2015AA105303);国家自然科学基金(61272083,61572033);软件新技术与产业化协同创新中心;安徽高校优秀青年人才支持计划重点项目(gxyqZD2016124);安徽省高校自然科学基金重点项目(KJ2016A252);安徽省自然科学基金(1608085MF147)资助.

作者简介:王勇(通讯作者)男,1979生,博士/副教授,研究方向:软件测试,故障定位,机器学习等.Email:yongwang@ahpu.edu.cn

forest, and finally orders those suspicious MFSs by their suspiciousness which will help programmers perform further diagnosis. The simulation results show that the presented method can effectively identify MFS. In particular, for the systems affected by mask effect, result is robust.

Key words: combinatorial testing; fault localization; fault forest; MFS

0　引言

组合测试旨在对软件的输入参数组合空间进行抽样,用以触发参数组合所引发的软件失效[1-3].该方法研究的关键是构造符合某种组合测试覆盖准则的最小测试用例集.大多数组合测试研究主要集中于构造有效的测试用例生成算法以及评估其触发软件失效的有效性[4].近年来,如何利用组合测试的结果进行软件失效原因的诊断受到广泛关注[5-7].Niu 等提出最小失效诱因模式(minimal failure-causing schema),定位最小失效诱因模式有助于软件调试中的故障定位与理解[5].对于一个软件包含 m 个取值的 n 输入参数,其参数组合为 m^n,因此,如何在巨大的输入参数组合空间中定位最小失效诱因模式是利用软件组合测试结果集实施故障定位的关键.

为了定位最小的 MFS,Nie 等提出逐个替换法,该方法采用启发式方法将触发软件失效的测试用例中的值模式进行替换,以减少故障模式的可能范围[8-9].Zhang 等提出 FIC 定位方法,该方法利用触发软件失效的测试用例作为种子进行初始化,然后通过自适应修改种子测试用例进行附加测试,反复直至找到最小失效诱因模式[10].Niu 等构建了一个特定元组关系树(tuple relationship tree,TRT)来描述所有特定值模式的关系,利用 TRT 可以减少生成附加测试用例的数目,并提供故障模式的精确定位[11].

以上方法都存在潜在的假设:运行包含 MFS 的测试用例中一定触发软件失效.Dumlu 等强调在现实系统中,由于屏蔽效应(masking effects)使得运行部分包括最小失效模式的测试未必一定触发软件失效[12].在现实系统中,即使测试用例包含 MFS,软件失效也呈间歇性,因此以上精确故障定位方法在不满足潜在假设的情况下很难取得满意的定位效果.一种可行的方法是计算各可疑 MFS 的概率,并将可疑排名列表提交给程序员进行进一步诊断[5,13].Yilmaz 等首次使用故障分类树分析给定的组合测试集,用以探测软件复杂配置空间中的潜在故障组合模式[14-15].Shakya 等采用 OPOT 准则生成尽可能多的失败测试用例,以降低故障分类树由于类的不平衡所带来的结果偏置[16].由于故障分类树所构建的可疑失效诱因模式强依赖于根节点的选择,因此若组合覆盖表中存在多个故障模式,故障分类树很难得到理想的分类结果.

为了解决以上问题,本文提出了一种基于故障森林的组合测试故障定位方法.给定一个 t-路组合测试集($t \geqslant 2$)及其附加测试集,该方法首先学习由多个深度为 t 的基本故障分类树所组成的故障森林,然后从故障森林中提取基本失效分类规则,最后对基本失效分类规则进行可疑度排序,并提交给程序员进行进一步诊断.

1　相关定义

假定待测软件(SUT)存在 n 个输入或配置参数 v_i,v_i 取值来自一个离散集合 V_i.SUT 可能由某一个或多个参数相互作用而导致软件系统产生失效.一个待测软件的配置如表 1 所示,该软件的配置参数为 H,O,B 和 U,表 1 中给出了每一个参数的取值集合,如 H 的取值集合为{Pc,Mac}.为了方便讨论,我们给出如下相关定义.

定义 1.1　测试用例集.组合测试用例集可表示为集合 $\{(v_1,v_2,\cdots,v_n) \mid v_1 \in V_1, v_2 \in V_2, \cdots, v_n \in V_n\}$,其中,$V_i$ 是对参数 P_i 进行划分后所形成的离散集合,v_i 是 V_i 中的一个具体的值.如(Pc,WinXP,Explorer,NewUser).

表 1　待测系统的配置表实例

Tab.1　Configuration table of a system under test

硬件(H)	操作系统(O)	浏览器(B)	用户(U)
Pc	Win2000	Explorer	NewUser
Mac	WinXP	Netscape	OldUser
	OS9	Firefox	
	OS10	Mozilla	

定义 1.2　k-值模式.对于某个组合测试用例集,一个 n 元组 $[-,\cdots,vl_1,-,\cdots,vl_k,\cdots]$ 被称作一个 k-值模式($n \geqslant k > 0$).在该元组中,vl_i 取 V_i 中的某一个固定值,"－"表示对应对应参数可以取

任何值.当 $k=n$ 时,该 k-值模式即为一个测试用例.例如,[—, WinXp, —, NewUser]即为一个 2-值模式.

定义 1.3 k-值失效诱因模式.一个 k-值失效诱因指包含此 k-值模式测试用例一定使得软件失效.与此相反,若包含某 k-值模式的所有测试用例均能运行成功,则其被称为 k-值正确模式.可疑 k-值故障模式指该 k-值模式与故障模式高度相关.

定义 1.4 最小 k-值失效诱因模式.如果一个 k-值故障模式是最小故障组合,则其所有的子模式都为正确模式.最小 k-值失效诱因模式是触发软件失效根源.

由于存在 mask effect,使得即使测试用例中包含最小的 MFS,软件的运行也未必触发软件失效.

定义 1.5 混合取值覆盖表.一个混合取值覆盖表可以表示为 $\mathrm{MCA}=[M,N,s_1^{k1}s_2^{k2}\cdots s_p^{kp},t,\lambda)$,其中,$M$ 表示测试用例的数目,N 为参数的个数,s_i 表示参数取值集合. si^{ki} 表示取值个数为 s_i 的参数有 k_i 个, $N=\sum_{i=1}^{p}k_i$,t 表示需覆盖组合的参数数目,λ 为覆盖的强度.

表 1 所示的输入空间构建的混合覆盖表 $\mathrm{MCA}(9,4,2^2 4^2,2,1)$,其包含 9 个测试用例,参数个数为 4,有 2 个参数的取值为 2 个,2 个参数的取值为 4 个,形成的组合覆盖强度为 2,在任何 9×2 的子矩阵的 2 元组在覆盖表中刚好出现 1 次($\lambda=1$).

在软件测试阶段,组合测试能够探测出 t-路组合故障.为了降低测试代价,其构建的测试用例集通常较小.如使用组合测试工具 AETG 对 TCAS 软件构建 2-路组合测试只需要 112 个测试用例.在故障模式定位中,需要大量的失败测试和成功测试,因此,需要充分利用已有的失败测试用例来生成尽可能多的失败测试用例.

定义 1.6 基本选择覆盖.一个基本选择参数值是针对每一个参数选择一个固定的参数值,且一个基本测试是通过对每一个特征使用基本选择形成的.新的测试用例通过如下方法生成:每次在某一个参数上选择非基本参数值,其他参数保持基本选择.

将失败测试作为一个基本选择.在这个基本选择的基础上以基本选择覆盖准则生成新的测试用例.因为每个生成的测试用例中只会有一个参数的值与失败测试用例相同,因此,生成的新测试更可能触发软件失效.

我们利用组合测试集及其测试结果,通过故障分类树对可疑的故障模式进行建模.

定义 1.7 故障分类树.故障分类树是一种描述对软件失效进行分类的树形结构,分类树由节点(node)和有向边(edge)组成.节点存在两种类型,内部节点和外部节点.内部节点表示一个参数,叶节点表示软件是否失效,边表示参数的某一个取值.图 1 所示为一个故障分类树实例,该树给出了三个分类规则,(1) $c=0,a=3$;(2) $c=1$;(3) $c=2$.规则 1 即为可疑故障模式,需要我们下一步重点关注.

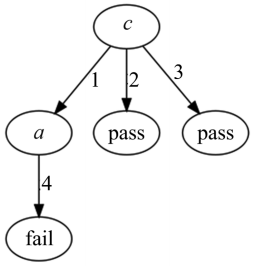

图 1 故障分类树实例

Fig.1 Example of fault classification tree

故障分类树算法主要有 ID3, C4.5 和 CART[17].它们主要在分裂属性选择使用的信息度量方式上存在区别.考虑到数据集为离散数据以及各个参数取值数目存在差异,本文选择采用 C4.5 算法.

2 故障定位模型

2.1 方法框架

如算法 2.1 所示,故障定位模型主要包括 3 步.①增强测试.利用基本选择覆盖准则以每一个失败测试用例作为种子,生成新的测试用例集,并进行增强测试.利用增强测试生成尽可能多的失败测试用例,以降低样本不平衡性问题带来的潜在风险.(第 2~7 行);②学习故障森林.利用组合测试用例集和测试结果向量,学习表示软件失效分类的故障森林(第 9 行);③故障分类规则提取与排名.先利用故障森林生成失效分类规则,再提取可疑 MFS,最后计算每一个可疑 MFS 并进行可疑度排名(第 10~13 行).

在故障定位框架中,最关键的是构建故障森林模型以及最终的定位可疑 MFS.下面我们主要介绍故障森林模型,以及可疑 MFS 提取.

算法 2.1 Identify inducing combinations

input: SUT, Testsuits, ErrorVector

Output: a set R of suspicious combinations ranking

```
1. //Step 1. Augment testing
2. for each failedTestcase do
3 newTestcases←generateNewtestcase(failedTestcase)
4. resultVector←execution(SUT,newTestcases)
5. Testsuits←Testsuits∪newTestcases
6. ErrorVector←ErrorVector∪resultVector
7. end for
8. //Step 2. learning fault classification forest
9. DF←learnFaultForest(testSuites,errorVector)
10. //Step 3. generateclassification rules
11. πf←generateFaultRules(DF)
12. πfm=generateMFS(πf)
13. R←ranking(πfm)
14. return R
```

2.2　故障森林模型

已有工作显示,故障分类树在组合测试故障定位具有很好效果[15-16].由于故障分类树学习可能受类不平衡的影响,且故障分类树所形成的规则总是强依赖于根节点,因此进一步利用故障分类树定位最小失效诱因模式需要解决两个问题:类的不平衡性以及提高多故障组合的定位效果.

我们首先对每一个失败测试用例使用基本选择覆盖生成尽可能多的新失败测试用例集,以减少测试结果的不平衡性.然后,我们构建故障森林以减少失效分类规则对单棵故障分类树的依赖.

定义 2.1　故障森林.故障森林是对基本故障分类树进行线性组合,故障森林可以表示为

$$fM(x)=\sum_{m=1}^{M}\alpha_i T(x;\Theta_m) \tag{1}$$

式中,α_i 是第 i 棵故障分类树的权重,$T(x;\Theta_m)$ 表示故障分类树,Θ_m 为故障分类树的参数,M 为故障分类树的个数.

本文所构建的故障森林模型接受一个组合测试覆盖集 TestSuits 及其测试结果 ErrorVector.覆盖表的覆盖强度为 t,根据文献[1]可知,对于一个强度为 t 的覆盖表可以命中小于等于 k 个参数交互的故障,以不少于 $\dfrac{1}{|v_k+1|\times|v_k+2|\times\cdots|v_m|}$ 的概率命中 m 个参数之间的交互故障.也就是说,当$k>t$时,k 个参数组合故障被命中可能性较小,此中情况下,使用故障分类树的分类结果会存在较大偏置,因此,基本故障分类树的深度设置为覆盖强度 t.我们输入中指定构建基本故障分类树的数目 M.

故障森林模型的基本思想是通过学习多个深度为 t(考虑组合测试代价,t 通常不会超过 4)的基本故障分类树,然后组合这些基本故障分类树构成分类准确度更高的故障森林.故障森林学习主要通过改变各测试用例的权重分布来实现基本故障分类树学习.因此,故障森林学习需要解决两个问题:①在每轮学习基本故障分类树时,如何改变每个测试用例的权重分布;②如何将基本故障分类树组合成故障森林.

对于问题①,我们首先将每个测试用例的权重设置为 $1/d$,构造基本故障分类树,然后针对被上一轮故障分类树错误分类的测试用例的权重 M_{w_i} 进行调整,调整策略如下:

$$M_{w_i}\leftarrow\frac{1-\mathrm{error(mDtreei)}}{\mathrm{error(mDtreei)}}\times M_{w_i} \tag{2}$$

式中,error(mDtreei)为上一轮故障分类树的误分率.

对于问题②,我们加大错误率小的基本故障分类树的权重,使其在决策中起较大作用,减少分类误差大的弱分类器的权重,使其在决策中起较小作用.具体算法如算法 2.2 所示.

算法 2.2　Generating fault forest

```
input:TestSuits: a martrix d * n
      ErrorVector:
      K:the number of basic classification tree in DF
      t:the depth of basic classification tree
output:DF:fault forest
1. // initialize weights for every test case
2. ∀ i≤d,Mwi←1/d
3. DF←∅
4. for i=1 toK do
5. // learn basic fault classification tree
6. Mtreei←
creatDTree(CTestSuits,ErrorVector,Mw,t)
7. //compute error(Mtreei)
8. error(Mtreei)=Σ_{j=1}^{d} Mwi×err(Xj)
9. if error(Mtreei)< 0.5 then
10. jump line 4
11. end if
12. DF←DF∪mDtreei
13. for each testcase failed to predict correctly do
14. // update Mwi
Mwi←(1−error(mDtreei))/error(mDtreei)×Mwi
16. end for
```

17. //normalize Mw_i

18. Mw←norm(Mw)

19. end for

20. return DF

2.3 MFS提取与排名

对基本故障分类器进行深度优先遍历，容易生成故障分类规则集合.我们依据MFS的定义对分类规则集合进行归并，得到可疑的MFS集合.

依据故障森林的定义 $fM(x)=\sum_{m=1}^{M}\alpha_i T(x;\Theta_m)$，$\alpha_i$ 为每棵故障分类树的投票系数.显然，一个故障分类器的分类错误率越低，其准确度就越高.因此，其表决权重就越高.α_i 可定义为

$$\alpha_i=\log\frac{1-\mathrm{error}(\mathrm{mDtree}_i)}{\mathrm{error}(\mathrm{mDtree}_i)} \tag{3}$$

对可疑MFS集合所在的各个基本故障树的表决权重进行累加，可以得到每一个MFS的可疑度.由此，产生可疑MFS的可疑列表，并提交程序员进行进一步诊断.

3 仿真实验

我们用仿真实验来验证基于故障森林的组合测试故障定位方法的有效性，主要回答以下问题：

(Ⅰ)该方法是否能有效定位单个 t 值的MFS；

(Ⅱ)该方法是否能有效定位多个 t 值的MFS；

(Ⅲ)该方法是否能有效定位存在mask effect情况下的MFS.

3.1 实验设置

仿真实验中，假设待测系统包含8个参数，每个参数有3个参数值.fault forest中的基本故障分类树的最大深度设置为 t，基本故障分类树的最大个数设置为5.

利用组合测试工具ACTS推荐的IPOG算法生成覆盖矩阵[18].例如，生成的2-way的覆盖矩阵包含15个测试用例，3-way覆盖矩阵包含60个测试用例，4-way覆盖矩阵包含191个测试用例.然后，向测试用例集注入最小失效诱因模式，将包含最小诱因模式的测试用例设置为失败的测试用例.将失败测试用例作为种子，利用基本选择覆盖，生成增强测试用例.对增强测试用例中包含最小诱因模式的测试用例设置成失败的测试用例.

3.2 有效性度量指标

为了评价本文故障定位方法的有效性，我们给出两个有效性度量指标：组合故障定位相似度(S_{ab})，故障定位命中率(hitRate).

对于故障定位相似度，我们对文献[11]的有效性度量方法进行修订.给定 MFS_a，和实际系统的 MFS_b，组合故障定位相似度定义为

$$S_{ab}=\frac{\mathrm{NumSame}_{ab}-\mathrm{NumDiff}_{ab}}{\mathrm{NumMax}_{ab}} \tag{4}$$

式中，$\mathrm{NumSame}_{ab}$ 是两个值模式固定参数值相同的个数，而 $\mathrm{NumDiff}_{ab}$ 是值模式固定参数值不同的个数，NumMax_{ab} 为两个值模式的固定参数个数最多的值模式的固定参数个数.例如，对于值模式 $a=[-,1,2,-]$，$b=[2,1,1,-]$，则 $\mathrm{NumSame}_{ab}=1$，$\mathrm{NumDiff}_{ab}=1$，$\mathrm{NumMax}_{ab}=3$.因此，$S_{ab}=(1-1)/3=0/3$.理想状态下，$S_{ab}=1$ 表示我们精确定位了最小失效诱因模式.假设存在多个MFS，我们取多个 S_{ab} 的最大值.

hitRate表示多个可疑MFS包含的值覆盖真实MFS的值集合的百分比.我们取可疑排名列表的top$k$ (k 可以取1,3,5)中的可疑最小值模式的值集合计算hitRate.例如，我们的定位结果为{[—,1,2,—],[—,1,3—]}，真实的最小诱因模式为{[—,1,2,—],[—,1,4]}.则我们定位结果覆盖了真实最小诱因模式中的3个值中的2个，因此，hitRate=2/3.

3.3 结果及分析

我们用3个仿真实验来验证组合故障模式定位方法的有效性.针对以上研究问题，主要考虑如下3种情形：

第1种情形.假设系统中仅仅包含单个 t 值的MFS(t 取2,3,4)，并且不存在mask effect影响.

第2种情形.假设系统既包含有大于 t 值故障模型也包含有小于 t 值的MFS，不存在mask effect影响.

第3种情形.假设系统中包含多个 t 值的MFS，并且存在mask effect影响.

3.3.1 情形1

对于情形1，我们通过ACTS工具生成2-way，3-way和4-way的覆盖矩阵.模拟不存在mask effect影响情况下，注入包含单个 t 值的MFS，并得到相应的成功测试用例集和失败测试用例集合.将失败测试用例集作为种子，采用基本选择覆盖准则，生成附加测试用例集，并得到附加测试中的成功测试用例集和失败测试用例集.如表2所示，MFS表示注入的最小失效诱因模式，N_{bt}，N_{bft}，N_{et}，

N_{eft}，N_t，N_{ft} 分别表示基本测试用例的数目，基本测试的失败用例数目，附加测试用例数目，附加测试的失败用例数目，测试用例的总数目以及失败用例总数目.

表 2　情形 1 的仿真实验结果

Tab.2　The results on question 1

	$t=2$	$t=3$	$t=4$
MFS	[-,2,-,2,-,-,-,-]	[-,1,-,3,1,-,-,-]	[-,1,2,3,1,-,-,-]
N_{bt} / N_{bft}	15/1	60/4	191/2
N_{et} / N_{eft}	8/6	32/20	16/8
N_t / N_{ft}	23/7	92/24	207/10
top1 S_{ab} /hitRate	1.0/1.0	0.67/0.67	0.5/0.5
top3 S_{ab} /hitRate	1.0/1.0	0.67/1.0	0.75/0.75

由表 2 可知，对于单个 t 值的最小失效诱因模式，在 $t=2$ 时，本文方法将真实的最小失效模式排在可疑列表的 top1，$S_{ab}=1$，hitRate=1.0，但是，随着 t 的增大，S_{ab} 与 hitRate 均呈下降趋势.我们分析，组合测试覆盖表的构成，不难发现：t 越大，包含最小失效诱因模式的测试用例数目越少，故障森林的效果越差.我们发现处于 fault report 的 top3 的 hitRate 一直维持高位，因此对于高维最小失效诱因模式，我们可以利用 fault report 中的 top$k$ 所组成的值集合进行进一步定位.

3.3.2　情形 2

对于情形 2，我们通过 ACTS 工具生成 2-way，3-way 和 4-way 的覆盖矩阵.模拟不存在 mask effect 影响情况下，注入包含小于多个小于 t 值，大于 t 值的最小失效诱因模式.考虑测试代价，在实际的 t-way 构造组合测试覆盖的构造中，通常使用从低维覆盖表向高维覆盖表扩展.也就是说，只有低维覆盖表没有触发故障时，我们才考虑开展到高维覆盖表进一步测试.表 3 中，$t=3$ 的情况下，失效模式[-,-,2,-,1,1,-,2]在基本覆盖表中并未触发软件失效；在附加测试中也只是触发了一次软件失效，因此基于 t-way 的组合测试结果，本文方法定位大于 t 的最小失效诱因模式存在挑战.

由表 3 可知，对于 2 个 t 值的最小失效诱因模式 S_{ab}，hitRate 指数较好.随着 t 的增大，我们很难进行精确定位，但基于 top$k$ 的参数值集合，可以为程序员进行进一步排查提高依据.

表 3　情形 2 的仿真实验结果

Tab.3　The results on question 2

	$t=2$	$t=3$	$t=4$
MFS	[-,1,2,-,-,-,-,-] [-,-,-,3,2,-,-,-]	[-,1,-,3,1,-,-,-] [-,-,2,-,1,1,-,2]	[2,-,-,-,1,2,3,1] [1,1,3,-,-,-,-,-]
N_{bt} / N_{bft}	15/4	60/4	191/7
N_{et} / N_{eft}	32/24	32/20	40/35
N_t / N_{ft}	47/28	92/24	247/42
top1 S_{ab} / hitRate	1.0/0.5	0.67/0.29	0.67/0.25
top3 S_{ab} / hitRate	1.0/1.0	0.67/0.57	0.67/0.75

3.3.3　情形 3

对于情形 3，我们通过 ACTS 工具生成 2-way，3-way 和 4-way 的覆盖矩阵.模拟存在 mask effect 影响情况下，注入包含单个 t 值的最小失效诱因模式.我们设置 mask effect 的影响为 33%，即存在 33%的概率，使得包含 MFS 的测试用例未发生软件失效.

由表 4 可知，对于存在 2 个 t 值的最小失效诱因模式，故障森林能成功地将 MFS 排在最前面，在 top3 中，成功定位 MFS.然而，随着 t 的增大，精确定位效果下降，但是我们可利用 top$k$ 中的参数值集合进行进一步定位.结果表明，存在 mask effect 影响的情况下，基于故障森林的故障定位方法健壮性较好.

表 4　情形 3 的仿真实验结果

Tab.4　The results on question 3

	$t=2$	$t=3$	$t=4$
MFS	[-,1,2,-,-,-,-,-] [-,-,-,3,2,-,-,-]	[-,1,-,3,1,-,-,-]	[2,-,-,-,-,2,3,1]
N_{bt} / N_{bft}	15/3	60/4	191/7
N_{et} / N_{eft}	32/16	32/20	40/35
N_t / N_{ft}	47/19	92/24	247/42
top1 S_{ab} /hitRate	1.0/0.5	0.67/0.67	0.5/0.5
top3 S_{ab} /hitRate	1.0/1.0	0.67/1.0	0.5/0.75

4　结论

本文提出了一种基于故障森林的软件组合测试定位方法.该方法首先以每个失败测试用例为种子进行增强测试，通过增加失败测试用例的数目，降低

故障分类树由于类的平衡性带来的影响.基于组合测试用例集和软件测试的结果,本文方法通过学习多个深度为 t 的基本故障分类树的故障森林,并通过故障森林提取失效分类规则.最后生成可疑的最小失效诱因模式列表提交给程序员进一步诊断.仿真实验结果表明,该方法能有效定位 t-way 最小失效诱因模式.对存在 mask effect 的数据集具有较强的健壮性.

考虑我们仅使用仿真实验验证方法的有效性,我们的未来工作包括开展更多的实证研究,考虑在实际的程序集中进一步验证该方法的有效性.另外,该方法可能受到故障森林相应参数的影响,如基本故障分类树的深度,基本故障分类树的数目等参数.我们需要开展更多的实证研究,为该方法的实际使用推荐最佳使用参数.

参考文献(References)

[1] NIE C, LEUNG H. A survey of combinatorial testing [J]. ACM Computing Surveys (CSUR), 2011, 43(2): 11(1-29).

[2] 严俊, 张健. 组合测试: 原理与方法[J]. 软件学报, 2009, 20(6): 1393-1405.
YAN J, ZHANG J. Combinatorial testing: Principles and methods[J]. Journal of Software, 2009, 6: 004.

[3] GHANDEHARI L S, BORAZJANY M N, LEI Y, et al. Applying combinatorial testing to the Siemens suite [C]// Proceeding of IEEE International Conference on Software Testing Verification and Validation Workshops. Luxembourg: IEEE Press, 2013: 362-371.

[4] GARGANTINI A, PETKE J, RADAVELLI M, et al. Validation of constraints among configuration parameters using search-based combinatorial interaction testing[C]// 8th International Symposium on Search Based Software Engineering. Springer, 2016: 49-63.

[5] GHANDEHARI L S G, LEI Y, XIE T, et al. Identifying failure-inducing combinations in a combinatorial test set [C]// IEEE 5th International Conference on Software Testing, Verification and Validation (ICST). IEEE, 2012: 370-379.

[6] GARGANTINI A, PETKE J, RADAVELLI M. Combinatorial Interaction Testing for Automated Constraint Repair [C]// Proceeding of IEEE International Conference on Software Testing Verification and Validation Workshops. Tokyo: IEEE Press, 2017: 239-248.

[7] ARCAINI P, GARGANTINI A, VAVASSORI P. Automatic detection and removal of conformance faults in feature models [C]// Proceeding of IEEE 9th International Conference on Software Testing Verification and Validation . Chicago: IEEE Press, 2013: 102-112.

[8] 徐宝文,聂长海,史亮,等. 一种基于组合测试的软件故障调试方法[J]. 计算机学报,2006,(01):132-138.
XU B W, NIE C H, SHI L, et al. A software failure debugging method based on combinatorial design approach for testing [J]. Chinese Journal of Computers, 2006, 29(1): 132.

[9] NIE C, LEUNG H. The minimal failure-causing schema of combinatorial testing[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 2011, 20(4): 15(1-38).

[10] ZHANG Z, ZHANG J. Characterizing failure-causing parameter interactions by adaptive testing [C]// Proceedings of the 2011 International Symposium on Software Testing and Analysis. Portland: ACM Press, 2011: 331-341.

[11] NIU X, NIE C, LEI Y, et al. Identifying failure-inducing combinations using tuple relationship [C]// IEEE 6th International Conference on Software Testing, Verification and Validation Workshops. Chicago: IEEE Press, 2013: 271-280.

[12] DUMLU E, YILMAZ C, COHEN M B, et al. Feedback driven adaptive combinatorial testing[C]// Proceedings of the 2011 International Symposium on Software Testing and Analysis. Portland: ACM Press, 2011: 243-253.

[13] GHANDEHARI L S, CHANDRASEKARAN J, LEI Y, et al. BEN: A combinatorial testing-based fault localization tool[C]// Proceeding of 2015 IEEE 8th International Conference on Software Testing Verification and Validation Workshops. Chicago: ACM Press, 2015: 1-4.

[14] YILMAZ C, FOUCHE S, COHEN M B, et al. Moving forward with combinatorial interaction testing [J]. Computer, 2014, 47(2): 37-45.

[15] YILMAZ C, COHEN M B, PORTER A A. Covering arrays for efficient fault characterization in complex configuration spaces [J]. IEEE Transactions on Software Engineering, 2006, 32(1): 20-34.

[16] SHAKYA K, XIE T, LI N, et al. Isolating failure-inducing combinations in combinatorial testing using test augmentation and classification [C]// IEEE 5th International Conference on Software Testing, Verification and Validation. Washington,:IEEE Press, 2012: 620-623.

[17] HAN J, PEI J, KAMBER M. Data Mining: Concepts and Techniques[M]. Elsevier, 2011.

[18] YU L, LEI Y, KACKER R N, et al. ACTS: A combinatorial test generation tool[C]// Proceeding of IEEE International Conference on Software Testing Verification and Validation. Luxembourg: IEEE Press, 2013: 370-375.