面向有效错误定位的偶然正确性识别方法

曹鹤玲[1,2,3],姜淑娟[1],王兴亚[1],薛 猛[1],钱俊彦[3]
(1. 中国矿业大学计算机科学与技术学院,江苏徐州 221116; 2. 河南工业大学信息科学与工程学院,河南郑州 450001;
3. 桂林电子科技大学广西可信软件重点实验室,广西桂林 541004)

摘 要: 错误定位是软件调试中耗时费力的活动之一. 针对偶然正确性影响错误定位效率的问题,提出面向错误定位的偶然正确性识别方法. 该方法首先识别偶然正确性元素;然后,挑选"偶然正确性特征元素",使用该特征元素约简程序执行轨迹;在此基础上,建立基于模糊 c 均值聚类的偶然正确性识别模型,将其结果应用于错误定位. 为验证该方法的有效性,基于 3 组测试程序开展偶然正确性识别,并将其结果应用于 Tarantula 等 4 种错误定位方法. 实验结果表明,与基于 k-means 聚类的偶然正确性识别方法相比,该方法在偶然正确性识别方面具有较低的误报率和漏报率,并且更能提高错误定位的效率.

关键词: 软件调试; 错误定位; 偶然正确性; 聚类分析
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2016)12-3026-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2016.12.030

Identifying Coincidental Correctness for Effective Fault Localization

CAO He-ling[1,2,3], JIANG Shu-juan[1], WANG Xing-ya[1], XUE Meng[1], QIAN Jun-yan[3]
(1. *School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangshu* 221116, *China*;
2. *College of Information Science and Engineering, Henan University of Technology, Zhengzhou, Henan* 450001, *China*;
3. *Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*)

Abstract: Fault localization is one of the most time-consuming activities in software debugging. An identifying coincidental correctness approach for effective fault localization is proposed to decrease the effect of coincidental correctness on the effectiveness of fault localization. First, the elements of coincidental correctness are computed. Second, the higher suspicious coincidental correctness elements are selected as feature elements of coincidental correctness, and then program execution traces are reduced in terms of feature elements. Finally, fuzzy c-means based coincidental correctness identification model is created based on the reduced execution traces to locate faults. It was applied to analyze three groups of programs, and test cases removing coincidental correctness were used as input for four popular fault localization approaches, such as Tarantula. The experimental results show that our approach had low false positives and false negatives, and performed well in terms of the effectiveness.

Key words: software debugging; fault localization; coincidental correctness; clustering

1 引言

基于覆盖信息的错误定位方法深受工业界和学术界广泛关注. 偶然正确性指软件中存在缺陷但是仍然通过测试的现象,它通常对基于覆盖信息的错误定位方法的效率产生负面的影响. 并且,Hierons 和 Masri 等人[1,2]研究了偶然正确性在软件系统中存在的普遍性. Masri 等人[3]研究表明程序大部分程序依赖不能传递任何可度量的信息,这意味着很多感染的程序状态不能传播到输出,从而导致偶然正确性的发生.

针对偶然性正确问题,Wang 等人[4]假设调试人员能预先知道程序缺陷类型,对错误触发前后的控制流和数据流信息与特定模式匹配,提纯代码覆盖信息,从而移除偶然正确性对错误定位效率的不利影响. Bandyopadhyay[5]根据成功测试用例和失败测试用例的相似度来识别偶然正确性,并赋予偶然正确性测试用例

收稿日期:2015-03-19;修回日期:2015-08-19;责任编辑:覃怀银
基金项目:国家自然科学基金(No. 61202006, No. 61340037, No. 61502497, No. 61562015, No. 61602154);广西可信软件重点实验室研究课题资助(No. kx201616, No. kx201532);河南省高等学校重点科研项目计划资助(No. 16A520005)

较低的权重,进而根据测试用例的权重而不是测试用例数量来计算程序语句的怀疑度,减少偶然正确性对错误定位效率的影响. 陈振宇等人[6]对程序覆盖信息进行聚类分析,将和失败测试用例在同一簇中的成功测试用例视为可能的偶然正确性测试用例,通过移除偶然正确性测试用例来减少其影响. 上述研究可以显著提高错误定位的效率,这促使我们去研究更有效率的偶然正确性识别方法.

文献[7]采用 k-means 聚类分析来识别偶然正确性测试用例,而 k-means 聚类属于硬分类方法,每个程序执行轨迹信息只能属于所有类别中的某一类,非此即彼. 这样带来的问题是:所有程序执行轨迹信息对计算聚类中心贡献度相同. 而模糊理论允许数据对象属于任何一类(簇),不同的是它们属于某类(簇)的可能性大小不同. 为此,我们引入模糊 c 均值聚类(Fuzzy C-Means,FCM)来识别偶然正确性测试用例. 将去除偶然正确性测试用例后的测试用例集应用于错误定位,则可以提高错误定位方法的效率.

本文贡献在于:

(1)提出了偶然正确性特征元素挑选策略,使用该特征元素对程序执行轨迹信息进行维度约简.

(2)提出面向错误定位的偶然正确性识别方法,该方法使用基于 Fuzzy c-means 聚类来识别偶然正确性测试用例,从而提高错误定位效率.

2　预备知识

2.1　偶然正确性

定义 1　偶然正确性元素 cc_e[7]. 给定一个程序元素 e,$P(T_F,e)$ 为失败测试用例 T_F 执行 e 的概率,$P(T_P,e)$ 为成功测试用例 T_P 执行 e 的概率,e 出现在所有失败测试中,即 $P(T_F,e)=1$,而出现在成功测试中的概率为 $0<P(T_P,e)<1$,则 e 为偶然正确性元素 cc_e,定义如下:

$$\mathrm{cc}_e=\{e\mid P(T_F,e)=1\wedge 0<P(T_P,e)<1\} \tag{1}$$

定义 2　偶然正确性特征元素 fcc_e. 使用某怀疑度计算式计算偶然正确性元素的怀疑度,并对其按怀疑度从大到小排序得到序列 $\mathrm{rank}(\mathrm{cc}_e)$,取序列的前 $\theta(0<\theta<1)$ 部分为偶然正确性特征元素.

$$\mathrm{fcc}_e=\{e\mid e\in \mathrm{cc}_e\wedge \theta*\mathrm{rank}(\mathrm{cc}_e)\} \tag{2}$$

定义 3　偶然正确性测试用例 T_{CC}[7]. 一个测试用例 T 执行了偶然正确性元素 cc_e,而没有导致程序执行结果失败,则该测试用例为偶然正确性测试用例 T_{CC}.

定义 4　偶然正确性(Coincidental Correctness,CC)[7]. 指的是程序执行了缺陷语句,而缺陷语句的错误状态并没有传播到程序输出结果,而使程序执行成功或没有抛出异常.

下面以函数 foo( ) 为例说明偶然正确性存在. 函数 foo( ) 中的错误在语句 s_5,正确语句为:ret = x + y. 我们设计了以下 6 个测试用例:$t_1(6,3,9)$,$t_2(2,4,6)$,$t_3(4,0,3)$,$t_4(5,2,0)$,$t_5(5,4,2)$ 和 $t_6(5,0,2)$ 来测试程序. 获取 6 个测试用例执行的覆盖信息及执行结果如表 1 所示,其中,"·"和"∘"分别表示语句被覆盖和未被覆盖,T 和 F 分别表示程序执行成功和失败. 由定义 4 可知,测试用例 t_3 和 t_6 执行了错误语句 s_5 而程序没有报错,则说明在程序的执行过程中存在偶然正确性. 根据定义 3 可知,测试用例 t_3、t_6 为偶然正确性测试用例.

表 1　偶然正确性示例

foo( ){		覆盖信息					
		t_1	t_2	t_3	t_4	t_5	t_6
s_1	int x,y,z;	·	·	·	·	·	·
s_2	int ret = 10;	·	·	·	·	·	·
s_3	read (x,y,z);	·	·	·	·	·	·
s_4	if (x > = y){	·	·	·	·	·	·
s_5	ret = x - y;	·	∘	·	·	·	·
s_6	if (y > z)	·	∘	·	·	·	·
s_7	ret = ret + 10;}	∘	∘	∘	·	·	∘
s_8	return ret;}	·	·	·	·	·	·
测试结果		F	T	T	F	F	T

2.2　模糊 c 均值聚类

软件错误定位场景中的聚类分析从数学模型角度刻画如下:$\boldsymbol{X}=(x_1,x_2,\cdots,x_j,\cdots,x_n)$ 表示聚类分析的数据对象(程序执行轨迹信息),即测试用例 $(t_1,t_2,\cdots,t_j,\cdots,t_n)$ 执行程序的轨迹信息,其中 x_j 表示第 t_j 个测试用例的执行轨迹信息. 对给定数据对象 $\boldsymbol{X}$ 进行聚类分析就是将其划分成 c 簇,对于识别偶然性正确测试用例来说就是将成功测试用例集 T_P 划分成偶然正确性测试用例集和真正成功测试用例集. 我们采用欧几里德距离 d_{ij} 如式(3)所示,来度量两个程序执行轨迹的相似度.

$$d_{ij}=\sqrt{\sum_{k=1}^{m}(x_{ki}-x_{kj})^2} \tag{3}$$

FCM 聚类就是通过找到使用的最佳组对 $(\boldsymbol{U},V)$ 来求解目标函数 $J(\boldsymbol{U},V)$ 的过程,从而将聚类问题转换成一个带约束的非线性规划问题,通过不断迭代求解得到数据集的最优划分,其中,$\boldsymbol{U}$ 表示隶属度矩阵,V 表示聚类中心点. FCM 聚类将数据对象 $\boldsymbol{X}$ 分为 c 个模糊簇,并求每簇的聚类中心点 v_i,$i=1,2,\cdots,c$,使得目标函数 $J(\boldsymbol{U},V)$ 达到最小值. 目标函数定义如下:

$$\begin{aligned}J(\boldsymbol{U},V)&=J(\boldsymbol{U},v_1,v_2,\cdots,v_c;\boldsymbol{X})\\&=\sum_{i=1}^{c}J_i=\sum_{i=1}^{c}\sum_{j=1}^{n}u_{ij}^m d_{ij}^2\end{aligned} \tag{4}$$

且满足下列等式:

$$\sum_{i=1}^{c}u_{ij}=1,\ \forall j=1,\cdots,n \tag{5}$$

其中,$d_{ij}=\|v_i-x_j\|$ 表示样本 x_j 与第 i 个聚类中心 v_i 的

欧几里德距离,$m \in [1,\infty)$是加权指数,随 m 值增大,聚类的模糊性增大,本文取 $m=3$. 另外,隶属度 $u_{ij} \in \{0,1\}$(其中,$i=1,2,\cdots,c$;$j=1,2,\cdots,n$)表示第 j 个数据对象属于第 i 个聚类中心的隶属度;每个数据对象与相应的聚类中心的隶属度构成一个 $c \times n$ 大小的隶属矩阵 $\boldsymbol{U}=[u_{ij}]_{c\times n}$.

对于所有输入的参数求导数,使得 $J(\boldsymbol{U},V)$ 达到最小值且满足等式(5)的必要条件为式(6)和(7). 其中,式(6)是最佳聚类中心点,式(7)是最佳隶属度函数.

$$v_i = \frac{\sum_{j=1}^{n} u_{ij}^m x_j}{\sum_{j=1}^{n} u_{ij}^m} \tag{6}$$

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left[\frac{d_{ij}}{d_{kj}}\right]^{\frac{2}{m-1}}} \tag{7}$$

3 本文方法

3.1 方法概述

针对偶然正确性影响错误定位效率的问题,提出面向有效错误定位的偶然正确性识别方法,并建立基于模糊 c 均值聚类的偶然正确性测试用例识别模型 CC-FCM (cleansing Coincidental Correctness test cases based on Fuzzy C-Means) 如图 1 所示. 该模型的目标是识别测试用例集中的偶然正确性测试用例,移除后应用于错误定位,能提高其效率.

该模型主要步骤如下:

(1)获取执行轨迹信息. 通过插桩程序获取程序执行轨迹信息,根据测试谕言将其分为成功和失败测试用例的执行轨迹信息.

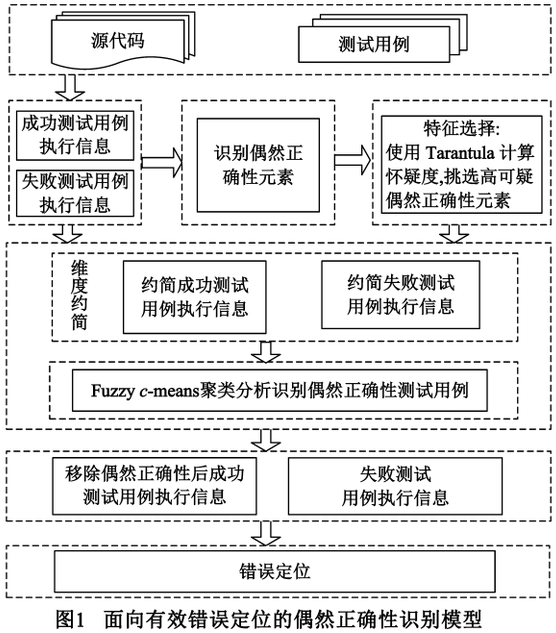

图1 面向有效错误定位的偶然正确性识别模型

(2)根据定义 1 识别偶然正确性元素 cc_e.

(3)特征元素挑选. 采用 Tarantula 方法对 cc_e 进行怀疑度计算,将怀疑度的大小作为偶然正确性元素的权重,对偶然正确性元素进行排序,设置参数 θ 作为选择偶然正确性元素的比例,挑选出特征元素.

(4)维度约简. 根据挑选的特性元素对步骤(1)中获得的程序执行轨迹信息进行维度约简.

(5)识别偶然正确性. 对降维后的程序执行轨迹信息采用 Fuzzy c-means 聚类算法将其分为两簇,一般认为偶然正确性测试用例的执行轨迹和失败测试用例的执行轨迹最相似[7],因此,将包含失败测试用例较多的那个簇中的成功测试用例视为偶然正确性测试用例,另外一个簇中的成功测试用例为真正成功测试用例.

(6)将移除偶然正确性测试用例后的测试用例集应用于错误定位,从而提高其效率.

3.2 算法

算法 1 描述了基于 Fuzzy c-means 聚类的偶然正确性测试用例识别算法,主要实现 3.1 节偶然正确性识别模型中的步骤 2 ~ 5. 第 1 ~ 5 行识别偶然正确性元素 cc_e,其中第 1 行初始化偶然正确性元素为 ϕ. 第 2、3 行根据计算偶然正确性元素,其中 $L(t)$ 表示程序执行轨迹中所有元素. 第 4 行计算偶然正确性元素集合. 第 6、7 行,特征元素选择,使用 Tarantula 计算怀疑度,对偶然正确性元素 cc_e 按怀疑度从大到小排序,设置挑选高可疑偶然正确性元素比例 θ,来挑选特征元素. 第 8 行执行轨迹维度约简,按挑选出的特征元素对所有程序执行轨迹进行维度约简,约简后程序执行轨迹信息作为 Fuzzy c-means 聚类的输入. 第 9 ~ 15 行为 Fuzzy c-means 聚类过程,第 9 行为初始化隶属度矩阵 $\boldsymbol{U}_0$ 且矩阵满足式(5). 第 10 ~ 12 行为聚类迭代求解过程,根据式(6)和式(7)迭代求解最优$(\boldsymbol{U}_f,\boldsymbol{V}_f)$,其中 $\boldsymbol{U}_f$ 表示最优隶属度矩阵,$\boldsymbol{V}_f$ 表示最优聚类中心点,第 14、15 行根据最优$(\boldsymbol{U}_f,\boldsymbol{V}_f)$生成最终簇 Cluster1,Cluster2,其中一个簇中包含偶然正确性测试用例.

算法 1 基于 Fuzzy c-means 聚类的偶然正确性测试用例识别算法

输入: k; //聚类的簇数 $k=2$
T; //所有程序执行轨迹信息
m; //聚类迭代次数最大值
θ; //偶然正确性元素挑选阈值
ε; //聚类分析终止阈值,$\varepsilon>0$
输出:Cluster1,Cluster2
begin
1. $\mathrm{cc}_e \leftarrow \phi$;
2. for each e in $\cup t \in \mathrm{Trace}(L(t))$
3. if $(\mathrm{P}(T_F,e)=1 \wedge 0<\mathrm{P}(\mathrm{T_P},e)<1)$

4. $cc_e \leftarrow cc_e \cup \{e\}$;
5. end for
6. calculate the suspiciousness of cc_e with Tarantula;
7. pick the most suspicious cc_e with suspiciousness sorted by Tarantula setting parameterθ;
8. reduce trace T using cc_e being picked;
9. randomly generate membership matric U_0 and make u_{ij} satisfy formula(5);
10. while ($\|U_t - U_{t-1}\| < \varepsilon \| \text{iteration} \leqslant m$)
11. calcalate the center of cluster V_t with U_t and formula(6);
12. calcalate membership matric U_t with V_{t-1} and formula(7);
13. end while
14. produce optimal $(U_f, V_f) = (U_t, V_t)$;
15. generate Cluster1 and Cluster2 according to (U_f, V_f);

end

4　实验评估

为评估偶然正确性对错误定位效率的影响,选取 Tarantula[8]、Ochiai[9]、Naish2[10] 和 Russel&Rao[10] 共 4 种错误定位方法进行对比实验. 运行环境:Ubuntu 64 位操作系统,版本 12.04,2 核 CPU 3.07GHz Intel(R),内存 16GB.

4.1　实验对象

实验采用西门子套件 C 版本中的 3 个程序 print_tokens2,schedule2 和 tcas 作为代码较小规模的程序代表;西门子套件目前使用广泛,但程序规模较小,为保障实验结论更具一般性,选取了代码中等规模的 Nanoxml 程序中版本 $v1$、$v2$ 和 $v5$;另外选取了 Unix 工具程序(gzip、grep、sed、flex). 实验对象详细信息见表 2,从 SIR(Subject Infrastructure Repository)(SIR 网址:http://sir.unl.edu/portal/index.html)下载.

表 2　实验对象

程序	描述	代码行数	错误版本数	测试用例数
print_tokens2	词法解析器	446	10	4115
schedule2	优先级调度器	245	9	2710
tcas	避免碰撞程序	131	20	1608
NanoXMLv1	XML 解析器	3497	7	207
NanoXMLv2	XML 解析器	4009	6	207
NanoXMLv5	XML 解析器	4782	8	206
gzip	数据压缩/解压	5365	20	211
grep	搜索文本模式	9205	14	806
sed	流文本编辑器	6763	22	360
flex	词法分析生成器	9766	20	567

4.2　评测指标

(1)漏报率(False Negatives,FN). 主要用于评估某种方法未准确识别偶然正确性测试用例的比率[7],见式(8),其中,true CC NO. 表示的是真实的偶然正确测试用例数,CC identifed NO. 表示的是通过聚类分析技术判别出来的偶然正确性测试用例数.

$$\mathrm{FN} = \frac{|\text{true CC NO.} - \text{CC identifed NO.}|}{|\text{true CC NO.}|} \tag{8}$$

(2) 误报率(False Positives,FP). 主要用于评估某种方法将非偶然正确性测试用例判别为偶然正确性测试用例的比率,见式(9),其中,Tp 表示成功测试用例数.

$$\mathrm{FP} = \frac{|(\text{Tp-true CC NO.}) \cap \text{CC identifed NO.}|}{|\text{Tp-true CC NO.}|} \tag{9}$$

(3)错误定位代价(cost). 为找到缺陷时需要检查的语句数与程序的总语句数的比率[11],见式(10).

$$\mathrm{cost} = \frac{\text{rank of faults}}{\text{program size}} \tag{10}$$

(4)检查得分(EXAM)[11]. 为错误检出率(% of faults located)与代码检查率(% of code examined)的比率.

$$\mathrm{EXAM} = \frac{\%\ \text{of faults located}}{\%\ \text{of code examined}} \tag{11}$$

4.3　实验评估

为评估方法有效性,进行以下 3 个方面的实验.

(1)偶然正确性存在的普遍性

通过程序插桩来检测成功测试用例的执行过程中缺陷语句是否被执行到,若被执行到,则为偶然正确性测试用例;否则为真正成功测试用例. 图 2 展示了测试的 136 个程序错误版本(见表 1)中普遍存在偶然正确性现象. 其中,横坐标为成功测试用例中包含偶然正确性百分比;纵坐标为包含一定比例偶然正确性测试用例的错误版本数比率.

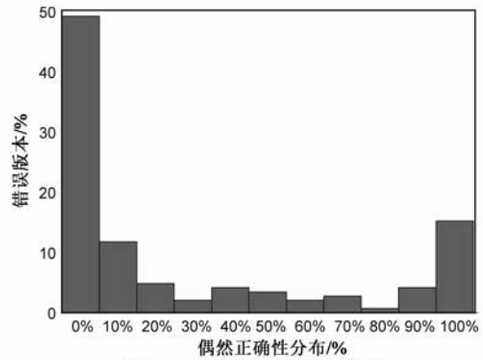

图2　偶然正确性测试用例分布

(2)识别偶然正确性的误报率(FN)和漏报率(FP)

由表 3 可见,CC-FCM 模型的误报率和漏报率都稍低于对比方法 Tech-I. 由于真正成功和偶然正确性测试用例较难区分,且聚类分析对数据对象的识别存在一定的偏差,导致这两种方法都有一定程度的误报和漏报.

表 3　平均误报率和漏报率

	Tech-I	CC-FCM
误报率	11.44%	9.46%
漏报率	27.32%	18.51%

(3)去除偶然正确性前后错误定位效率的对比

本文主要采用 4 种错误定位方法 Tarantula, Ochiai, Naish2 和 Russel&Rao 进行对比实验. 在下面 3 种情况下比较其错误定位的效率:①不考虑偶然正确性,在原始测试用例集上进行错误定位,记为 Orig. ②在文献[7]中 Tech-I 方法去除偶然正确性的测试用例集上进行错误定位,记为 Tech-I. ③在 CC-FCM 模型识别并移除偶然正确性的测试用例集上进行错误定位,记为 CC-FCM.

对于某方法来说,错误定位代价越小说明其效率越高. 实验结果如表 4 中所示,在 CC-FCM 和 Tech-I 移除偶然正确性的基础上,4 种方法的错误定位代价均低于在原始测试用例集上(Orig.)的错误定位代价. 大多数情况下,在 CC-FCM 识别偶然正确性的基础上的错误定位代价低于在 Tech-I 识别偶然正确性基础上的错误定位代价;但在个别情况下,在 CC-FCM 基础上的错误定位代价稍高于或等于在 Tech-I 基础上的错误定位代价,如表 4 中阴影所示. 例如,对于 schedule2 程序,Russel&Rao 方法在 CC-FCM 基础上的错误定位代价 18.32% 稍高于在 Tech-I 基础上的错误定位代价 18.23%. 其原因在于:这些测试程序不尽相同,程序特征及错误分布可能不同,这使得本文方法的效果可能受到一定程度的影响,在个别情况下,效率提升不明显.

接下来比较在 CC-FCM 与 Tech-I、Orig. 基础上错误

表 4 去除偶然正确性前后各错误定位方法的错误定位代价

程序	Tech.	Tarantula	Ochiai	Naish2	Russel&Rao
print_tokens2	Orig.	22.74%	23.61%	3.89%	12.76%
	Tech-I	11.21%	15.70%	3.44%	8.59%
	CC-FCM	10.01%	8.63%	3.25%	7.06%
schedule2	Orig.	29.00%	28.13%	22.47%	19.32%
	Tech-I	26.08%	24.02%	19.05%	18.23%
	CC-FCM	23.81%	22.02%	18.10%	18.32%
tcas	Orig.	21.06%	20.04%	19.12%	18.15%
	Tech-I	20.06%	17.29%	17.42%	15.61%
	CC-FCM	18.67%	16.54%	17.35%	15.96%
Nanoxmlv1	Orig.	10.21%	8.12%	7.50%	7.93%
	Tech-I	8.14%	7.38%	6.86%	6.53%
	CC-FCM	6.42%	7.41%	6.39%	6.53%
Nanoxmlv3	Orig.	5.15%	4.27%	3.95%	4.21%
	Tech-I	3.00%	2.59%	2.99%	2.99%
	CC-FCM	2.20%	1.63%	2.49%	2.00%
Nanoxmlv5	Orig.	7.34%	4.76%	4.45%	4.75%
	Tech-I	4.95%	2.86%	4.15%	4.53%
	CC-FCM	3.41%	2.11%	4.04%	4.31%
gzip	Orig.	9.45%	9.38%	9.42%	8.24%
	Tech-I	6.86%	7.28%	8.70%	7.77%
	CC-FCM	5.13%	5.78%	8.13%	7.38%
grep	Orig.	2.33%	0.85%	0.41%	3.04%
	Tech-I	1.90%	0.61%	0.35%	2.57%
	CC-FCM	1.25%	0.52%	0.33%	2.41%
sed	Orig.	1.02%	0.32%	0.28%	3.74%
	Tech-I	0.78%	0.30%	0.25%	3.33%
	CC-FCM	0.78%	0.26%	0.22%	3.04%
flex	Orig.	10.61%	5.45%	4.87%	5.18%
	Tech-I	8.81%	5.12%	4.71%	4.54%
	CC-FCM	7.74%	4.83%	4.52%	4.18%

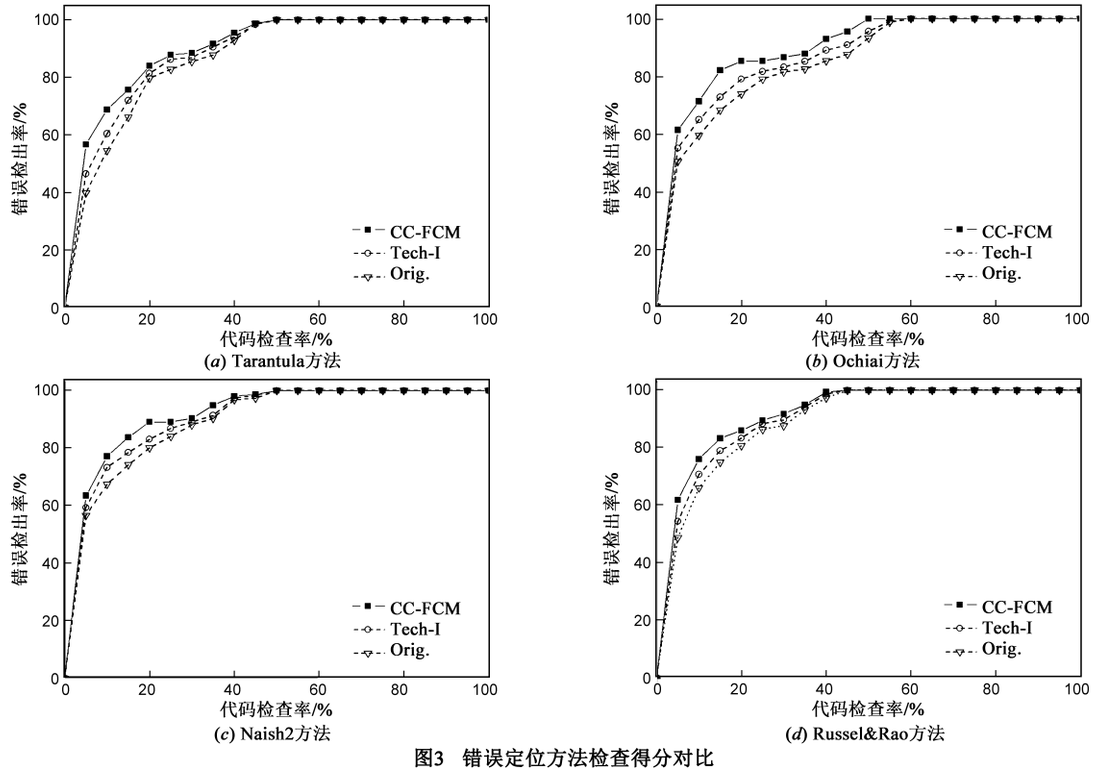

图3 错误定位方法检查得分对比

定位的检查得分. 图 3 分别展现了在 3 种情况下 4 种错误定位方法的检查得分. 其中,横坐标代表代码检查率,纵坐标代表在一定代码检查率下的错误检出率. 相同代码检查率下,某方法定位出的错误比例越高则该方法越有效. 从图 3 可以看出,4 种错误定位方法在 CC-FCM 识别偶然正确性情况下检查得分高于在 Tech-I 识别偶然正确性情况下的检查得分;并且 CC-FCM 和 Tech-I 识别并移除偶然正确性后,错误定位方法的检查得分都高于不考虑偶然正确性测试用例 Orig. 情况下的检查得分.

本文方法与 Tech-I 方法相比的优势在于:①引入 Fuzzy c-means 聚类分析,该聚类允许数据对象属于任何一簇,只是它们属于某簇的可能性大小不同;而 Tech-I 方法使用的 k-means 聚类属于硬分类,使得每个程序执行轨迹只能属于所有类别中的某一类,对聚类中心点的贡献度相同. ②设计了偶然正确性特征元素挑选策略,这些特征元素更能反映程序执行轨迹信息的偶然正确性特征.

5　结论

针对偶然正确性问题,提出一种面向有效错误定位的偶然正确性识别方法. 首先,识别出偶然正确性元素;其次,对偶然正确性元素排序,挑选出高可疑偶然正确性元素作为特征元素,利用其对程序执行轨迹进行维度约简;最后,使用模糊 c 均值聚类对维度约简后程序执行轨迹进行聚类,识别出偶然正确性测试用例,将其移除偶然正确性后的结果应用于错误定位. 实验表明:与 Tech-I 方法相比,CC-FCM 识别偶然正确性的误报率和漏报率有所下降;移除偶然正确性测试用例后,错误定位方法的定位效率有一定程度的提升.

参考文献

[1] Hierons R M. Avoiding coincidental correctness in boundary value analysis[J]. ACM Transactions on Software Engineering and Methodology,2006,15(3):227 – 241.

[2] Masri W,Assi R A. Cleansing test suites from coincidental correctness to enhance fault-localization[A]. Proceedings of the 3rd International Conference on Software Testing, Verification and Validation[C]. Paris,France:IEEE,2010. 165 – 174.

[3] Masri W,Podgurski A. Measuring the strength of information flows in programs[J]. ACM Transactions on Software Engineering and Methodology,2009,19(2):5 – 37.

[4] Wang X,Cheung S C,et al. Taming coincidental correctness:coverage refinement with context patterns to improve fault localization[A]. Proceedings of the 31th International Conference on Software Engineering[C]. Los Alamitos, CA:IEEE Computer Society,2009. 45 – 55.

[5] Bandyopadhyay A. Mitigating the effect of coincidental correctness in spectrum based fault localization[A]. Proceedings of the 5th International Conference on Software Testing,Verification and Validation[C]. Montreal,QC:IEEE, 2012. 479 – 482.

[6] Miao Y,Chen Z,Li S,et al. A Clustering-based strategy to identify coincidental correctness in fault localization[J]. International Journal of Software Engineering and Knowledge Engineering,2013,23(05):721 – 741.

[7] Masri W,Assi R A. Prevalence of coincidental correctness and mitigation of its impact on fault localization[J]. ACM Transactions on Software Engineering and Methodology, 2014,23(1):1 – 28.

[8] Jones J A,Harrold M J,Stasko J. Visualization of test information to assist fault localization[A]. Proceedings of the 24th International Conference on Software Engineering [C]. Los Alamitos,CA:IEEE Computer Society,2002. 467 – 477.

[9] Rui A,Peter Z,et al. An evaluation of similarity coefficients for software fault localization[A]. Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing[C]. Riverside,CA:IEEE,2006. 39 – 46.

[10] Naish L,Lee H J,et al. A model for spectra-based software diagnosis[J]. ACM Transactions on Software Engineering and Methodology,2011,20(3):11 – 43.

[11] Renieris M,Reiss S P. Fault localization with nearest neighbor queries[A]. Proceedings of the 18th IEEE International Conference on Automated Software Engineering[C]. New York,USA:ACM,2003. 30 – 39.

作者简介

曹鹤玲　女,1980 年 5 月出生于河南南阳. 中国矿业大学博士生,CCF 会员. 主要研究领域为软件分析与测试、数据挖掘.
E-mail:caohl410@ cumt. edu. cn

姜淑娟(通信作者)　女,1966 年 12 月出生于山东莱阳. 现为中国矿业大学计算机科学与技术学院教授、博士生导师,CCF 会员. 主要研究领域为编译技术、软件工程等.
E-mail:shjjiang@ cumt. edu. cn