基于非参数假设检验的程序缺陷定位方法

叶 钢 李重文 余 丹 马世龙

(北京航空航天大学 计算机学院, 北京 100191)

摘 要: 调查了西门子测试集中的程序失败用例的规模和谓词评估偏差的分布. 结果表明,中心极限定理的前提和参数假设检验的假设条件无法得到普遍满足. 所以,已有的基于参数假设检验的方法存在潜在的问题. 针对这一现象,提出了基于非参数假设检验定位程序缺陷的方法. 实验结果表明,该方法在非正态分布的样本集上仍具有较好的适用性,且在缺陷定位效率上优于已有的基于程序谓词的缺陷定位技术. 同时比较了基于 2 种常用的非参数假设检验模型的缺陷定位方法的效率. 一种方法基于 Kolmogorov-Smirnov 检验,另一种方法基于 Cramér-von-Mises 准则. 比较结果表明,在西门子测试集上,基于 Kolmogorov-Smirnov 检验的方法在缺陷定位效率上优于基于 Cramér-von-Mises 准则的方法.

关 键 词: 软件测试; 缺陷定位; 统计学方法

中图分类号: TP 311.52

文献标识码: A **文 章 编 号**: 1001-5965(2012)08-1056-08

Fault localization based on non-parametric hypothesis testing model

Ye Gang Li Zhongwen Yu Dan Ma Shilong

(School of Computer Science and Technology, Beijing University of Aeronautics and Astronautics, Beijing 100191, China)

Abstract: The size of failed runs and the distributions of the evaluation biases on the Siemens suite were investigated. The empirical results show that the precondition for the central limit theorem and the assumption on feature spectra forming normal distributions are not well-supported by empirical data. Thus, the previous method based on parametric hypothesis testing has a potential problem. New approaches based on non-parametric hypothesis testing models were proposed. The empirical results on the Siemens suite indicate that these approaches can outperform existing predicate-based statistical fault localization techniques, especially on non-normal distributions. The effectiveness comparison between two methods based on two popular non-parametric hypothesis testing models was also investigated. One method was based on the Kolmogorov-Smirnov test and the other was based on Cramér-von-Mises criterion. The comparison results show that the method based on the Kolmogorov-Smirnov test consistently outperforms that based on the Cramér-von-Mises criterion in the task of fault localization.

Key words: software testing; fault localization; statistical methods

程序开发过程中,软件调试是找出程序缺陷并减少其数目的过程. 通过减少缺陷数目,可以使得程序按程序员的预期执行. 但是,随着程序规模的不断增大,以及一个缺陷的修改可能导致另一个缺陷的引入,导致软件调试通常耗费大量时间,并常成为软件开发过程中的一个瓶颈. 显然,在软件开发和维护过程中,高效的程序缺陷定位技术对于保证软件质量和降低软件成本具有重要意义. 近 10 年来,通过比对程序在正确执行与失败执行之间的运行时行为差异而发现缺陷的动态缺

收稿日期: 2011-03-26; **网络出版时间**: 2012-06-15 15:44

网络出版地址: www.cnki.net/kcms/detail/11.2625.V.20120615.1544.036.html

基金项目: 国家自然科学基金资助项目(61003016);软件开发环境国家重点实验室自主研究课题资助项目(SKLSDE-2011ZX-09)

作者简介: 叶 钢(1982 -),男,浙江桐庐人,博士生,yegang@nlsde.buaa.edu.cn.

陷定位技术被广泛地应用到软件测试领域[1-7]. 这类技术的目的旨在协助软件测试人员定位缺陷在软件中的位置. 在所有的动态方法中,基于程序谓词的缺陷定位技术凭借其对程序插桩规模小同时能完整收集程序执行控制信息的特点,受到了许多研究者的关注[8-12]. 基于谓词的缺陷定位产品,如 CBI[10-11]和 SOBER[8-9]等成为这一领域较有影响的缺陷定位工具. 一个程序谓词(简称谓词)是在程序中的某处,关于程序属性的布尔表达式(如分支条件等). 基于谓词的缺陷定位技术通过比较程序谓词在成功用例和失败用例中的统计信息,定位缺陷至相关语句. 在基于谓词的缺陷定位技术中,基于中心极限定理[8-9]的方法被认为是一种高效的缺陷定位手段. 在实践中,这类方法从一些广泛使用的程序中成功地分离出了缺陷. 文献[9]首先定义评估偏差(evaluation bias),记作 $\pi(P)$,度量一个谓词在程序的一次执行中被执行为"true"的可能性. 并根据中心极限定理,认为 $\pi(P)$ 服从正态分布,进而运用参数假设检验的思想,通过比较成功与失败用例中的 $\pi(P)$ 的 2 类分布之间的差异,定位程序缺陷. 然而,其所使用的谓词排序模型存在潜在的问题.

本文通过对西门子测试集(the Siemens suite[13])的研究表明:某些含缺陷的程序,其执行失败的测试用例总数较少,因此失败测试用例的集合可被视为小样本集. 此时,使用中心极限定理是不恰当的. 从小样本的现实出发,本文对 $\pi(P)$ 服从正态分布的假设产生怀疑. 通过进一步的实验证明,西门子测试集中存在大量评估偏差的分布不服从正态分布的谓词,其中甚至包括缺陷所在谓词. 因此,谓词评估偏差分布的这一特性将严重影响基于参数假设检验的方法判别缺陷的能力,最终将使其产生非鲁棒的结果报告.

综上所述,本文提出了基于非参数假设检验的缺陷定位方法. 实验表明,相较先前已存在的基于中心极限定理的参数假设检验的方法,该方法提高了缺陷定位的效率和鲁棒性.

本文的贡献主要有:

1) 首次调查了西门子测试集中失败用例的规模. 结果表明:在许多含缺陷的程序版本中,执行失败的测试用例的总数小于 30,从中采集得到的样本为小样本. 此时,在小样本上运用中心极限定理不符合定理要求.

2) 检验了西门子测试集中谓词评估偏差分布的正态性. 结果表明:西门子测试集中存在大量评估偏差分布不服从正态分布的谓词,其中甚至包括缺陷所在谓词. 因此,谓词评估偏差的实际分布背离了文献[8-9]提出的基于参数假设检验的方法所假设的正态分布,其方法存在潜在的问题.

3) 提出了基于非参数假设检验的缺陷定位方法,并比较了 2 种用于检验分布函数拟合程度的常用非参数假设检验方法在定位程序缺陷中的效率. 实验表明,较先前已存在的基于中心极限定理的参数假设检验方法,基于非参数假设检验的缺陷定位方法提高了缺陷定位的效率和鲁棒性.

1　研究动机

文献[8-9]提出了基于中心极限定理的参数假设检验方法定位程序缺陷. 本节介绍该方法(下文以 SOBER 命名)及其存在的潜在问题.

1.1　SOBER

对于一个程序 $\mathscr{P}$ 中每个谓词 P,SOBER 设定原假设为 $f(X|\mathscr{P}\text{failed})$ 与 $f(X|\mathscr{P}\text{passed})$ 相同,其中 X 为表示谓词 P 的评估偏差的随机变量. 根据中心极限定理,SOBER 认为谓词 P 的评估偏差 $\pi(P)$ 服从正态分布,并使用 $N(\mu_{\text{p}},\sigma_{\text{p}})$ 和 $N(\mu_{\text{f}},\sigma_{\text{f}})$ 分别表示谓词 P 的评估偏差在成功用例与失败用例集上的分布. 令 $X_{\text{s}}=\{x_1,x_2,\cdots,x_m\}$ 表示来自 m 个失败执行用例的谓词 P 的评估偏差的样本集. 令 $X_{\text{f}}=\{x'_1,x'_2,\cdots,x'_n\}$ 是一个来自 n 个失败执行用例的谓词 P 的评估偏差的样本集. 根据正态分布的特性,SOBER 认为统计量 $Z=\dfrac{\frac{1}{m}\sum_{i=1}^{m}x_i-\mu_{\text{p}}}{\sigma_{\text{p}}/\sqrt{m}}(x_i\in X_{\text{f}})$ 服从标准正态分布 $N(0,1)$. 最后,对于 P,SOBER 认为 $f(X|\mathscr{P}\text{failed})$ 与 $f(X|\mathscr{P}\text{passed})$ 差异越大,则 P 是缺陷的可能性越高. 即,若令 $L(P)=\text{Similar}(f(X|\mathscr{P}\text{failed}),f(X|\mathscr{P}\text{passed}))$,$f(X|\mathscr{P}\text{falied})$ 与 $f(X|\mathscr{P}\text{passed})$ 差异越大,则 Similar 函数值越小,P 是缺陷的可能性越高. 因此,SOBER 得出其"缺陷怀疑度"计算函数 $s(P)=-\ln(L(P))=\ln\left(\dfrac{\sigma_{\text{p}}}{\sqrt{m}\varphi(Z)}\right)$,其中,$\varphi(Z)$ 是标准正态分布 $N(0,1)$ 在 Z 处的密度函数值. $s(P)$ 越大,则对应的谓词 P 是缺陷的可能性越高. 使用 $s(P)$,SOBER 对所有谓词排序,并产生一个降序的缺陷相关谓词列表作为程序缺陷报告.

1.2　存在的问题

理论上,中心极限定理要求:如果样本容量足够大,则均值的理论抽样分布近似服从正态分布.

而对于“足够大”不存在统一的严格界定,一般认为样本容量 $n>30$ 是“足够大”.

文献[9]中使用西门子测试集(the Siemens suite)作为实验对象. 因此,本文调查了西门子测试集中每个程序失败用例的规模.

文献[9]将执行成功与执行失败的测试用例分别划分为2类集合,从每类集合中分别得到的谓词 P 的评估偏差 $\pi(P)$ 样本集 X_s 和 X_f. 一般而言,执行失败的测试用例总数小于执行成功的测试用例总数,因此,本文以每个含缺陷程序的所有测试用例作为西门子测试集中程序的输入,并且统计测试输出中失败用例形成的样本集的规模. 图1显示了西门子测试集中,所有130个含缺陷程序的失败执行用例总数的规模. 若将样本容量大于30视为“足够大”,则从图中可以看到,其中36.15%(130中的47个)含缺陷程序,来自失败用例的样本集样本容量过小而不符合使用中心极限定理的要求.

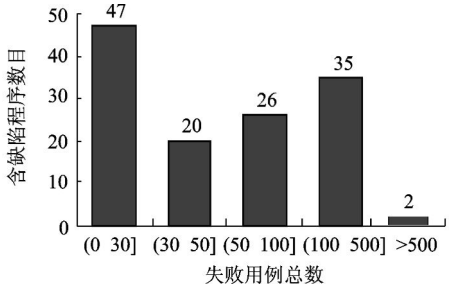


图1 西门子测试集中所有含缺陷程序失败测试用例数图

基于上述分析,本文调查了另一个问题:当样本容量不符合中心极限定理要求时,谓词的评估偏差的分布是否服从正态分布?

为了度量一个谓词的评估偏差是否服从正态分布,本文采用标准正态性检验方法——Jarque-Bera检验(从 http://www.alglib.net/上可以获取Jarque-Bera检验的程序). 针对一个待检验的分布进行Jarque-Bera检验最终将得到一个结果值,记作 p 值. p 值越小,则被检测的分布服从正态分布的可能性越小.

对于西门子测试集中的每个谓词,本文对其在成功和失败执行用例样本集上的评估偏差的分布均进行Jarque-Bera检验,并将得到的2个 p 值中较小的那个作为检验结果[12]. 图2显示了正态性检验的统计结果. 从图中可以看出,如果选择0.05作为正态性检验的显著性水平,则38.3%的分布(所有分布中的2213个)不服从正态分布.

综上所述,实验验证了36.15%的西门子测试集中含缺陷程序,来自失败用例的样本集样本容量过小而不符合使用中心极限定理的要求;同时程序谓词评估偏差的实际分布背离了SOBER所假设的正态分布,因此,SOBER判别缺陷的能力将受到严重影响,并将使其产生非鲁棒的结果报告. 根据统计学原理,对于小样本集上呈现非正态分布的谓词的评估偏差,较之基于中心极限定理的参数假设检验模型,本文认为使用非参数假设检验模型定位缺陷更为合适.

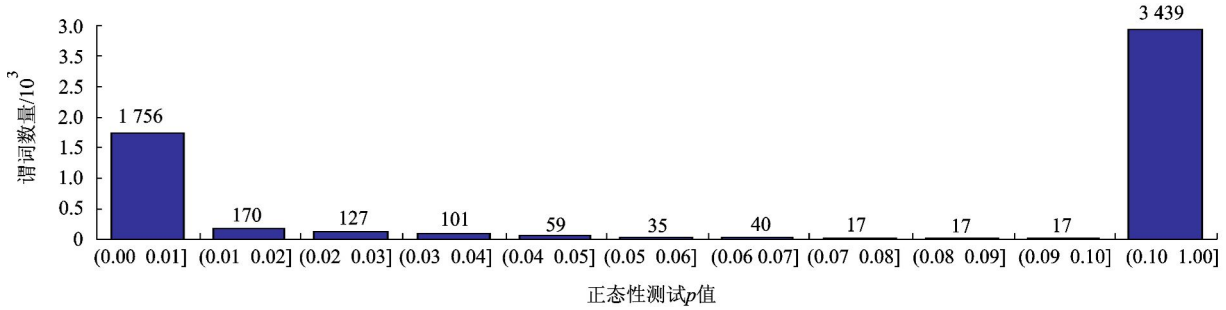


图2 正态性检验的统计结果分布图

2 谓词缺陷相关度排序模型

2.1 预备知识

首先回顾程序谓词和评估偏差的概念.

一般而言,一个程序谓词是一个关于程序任意属性的命题,例如“$p>\mathrm{MAX}$”,“next = = −3”等. 一个谓词在每次被执行时,或者为“true”或者为“false”. 在早期论文中,Liblit等人提出将谓词分为3类[11]:分支(branches),返回(returns)和标量对(scalar-pairs),并定义如下:

1)分支:执行每个条件语句时,存在2类谓词,这2类谓词分别标识该分支被执行为“true”或者为“false”.

2)返回值:在程序中每个函数返回处,存在6类谓词标注返回值与0的关系:<0, $\leqslant 0$, >0, $\geqslant 0$, $=0$,或$\neq 0$.

3)标量对:在每个标量赋值语句 $x=\cdots$ 中,通过检测 x 值与指定变量 y 或常量 C 的6类关系($<$, $\leqslant$, $>$, $\geqslant$, $=$, $\neq$)来发现变量与变量之间或者变量与常量之间的关系. 文献[9]发现不追

踪这些标量对能减少近一半的工作量但并不严重影响缺陷定位的效果.

根据前先的实验,本文插桩源代码以监视 2 类谓词:分支和返回值.

在一个测试用例中,一个谓词可能被执行多次. 在每次执行中,一个谓词或者为“true”或者为“false”. 文献[9]因此给出了评估偏差的概念,度量一个谓词被执行为“true”的可能性:

定义 1　评估偏差[9]. 令 n_t 是某个谓词 P 在程序的一次执行中被判断为“true”的次数,n_f 是谓词被判断为“false”的次数. 定义如下表达式为谓词 P 的评估偏差:$\pi(P) = n_t/(n_t + n_f)$.

如果一个谓词在任何失败测试用例中都没有被执行,则认为该谓词与程序失败无关,并将其从下面的谓词排序模型中除去.

2.2　方法描述

本节描述文中方法的主要思想,其细节将在下节描述.

令 X 是谓词 P 的评估偏差的随机变量. 本文使用 $f(X|\theta_s)$ 和 $f(X|\theta_f)$ 分别表示谓词 P 的评估偏差在 T_s 和 T_f 上的概率模型. 从而得到谓词 P 的缺陷相关性定义如下:

定义 2　缺陷相关性[9]. 一个谓词 P 是程序缺陷相关的,如果其密度函数 $f(X|\theta_f)$ 不同于 $f(X|\theta_s)$. 其中,X 是表示谓词 P 的评估偏差的随机变量.

因此,可以定义以下函数:

$$R(P) = D(f(X|\theta_s), f(X|\theta_f)) \quad (1)$$

来度量 $f(X|\theta_s)$ 和 $f(X|\theta_f)$ 之间的差异. 该差异越大,则 P 与缺陷越相关.

令 $X_s = \{x_1, x_2, \cdots, x_m\}$ 是一个来自 m 个成功执行用例的某个谓词 P 的评估偏差的样本集. 并且,令 $F_s(x)$ 为 X_s 上的实验分布函数. 同样,令 $X_f = \{x'_1, x'_2, \cdots, x'_n\}$ 是一个来自 n 个失败执行用例的谓词 P 的评估偏差的样本集. 并且,令 $F_f(x)$ 为 X_f 上的实验分布函数. 不失一般性,假设 $m \geqslant n$. 根据非参数假设检验,本文通过检验 $F_s(x)$ 与 $F_f(x)$ 是否相等,来判断 $f(X|\theta_s)$ 与 $f(X|\theta_f)$ 是否为同一分布.

算法:基于非参数假设检验模型的程序缺陷定位算法.

输入:测试用例集 $T = \{t_1, t_2, \cdots, t_n\}$(其中 t_i 表示一个测试用例)和一个含缺陷程序 $\mathscr{P}$.

输出:谓词列表. 该列表对每个缺陷相关谓词赋一个“缺陷怀疑度”值 $s(P)$,并根据该值对所有谓词降序排序.

方法:

开始

1) 插桩源程序 $\mathscr{P}$ 以监视 2 类谓词:分支和返回值;

2) 针对每一个测试用例 $t_i = (\text{input}_i, \text{output}_i)$ $(1 \leqslant i \leqslant n)$,执行程序 $\mathscr{P}$,获得输出 $O_i = \mathscr{P}(\text{input}_i)$,从 O_i 中分离原始输出信息 output'_i 和谓词执行相关信息 OP_i;

3) 根据 output'_i 与 output_i 是否相等,将测试用例集 T 分为 2 个不相交的子集,分别记为 T_s 和 T_f;

4) 除去所有缺陷无关的谓词:一个谓词在任何执行失败的测试用例中都没有被执行,则认为该谓词是缺陷无关的;

5) 对于余下的每个谓词,分别计算并收集其在 T_s 和 T_f 上评估偏差 $\pi(P)$ 的样本集,分别记为 X_s 和 X_f;

6) 分别获得谓词评估偏差 $\pi(P)$ 在 X_s 和 X_f 上的实验分布函数 $F_s(x)$ 和 $F_f(x)$;

7) 对每一个缺陷相关谓词 P,基于非参数假设检验模型计算 $F_s(x)$ 与 $F_f(x)$ 间的差异,记作 $s(P)$,作为 P 的“缺陷怀疑度”值;

8) 对每个缺陷相关谓词 P,根据其 $s(P)$ 按降序排序得到谓词列表;

9) 返回结果谓词列表,辅助软件测试人员定位程序缺陷.

结束

2.3　基于非参数假设检验的谓词排序模型

在统计学中,Kolmogorov-Smirnov 检验[14] 和 Cramér-von-Mises 准则[15] 是 2 种常用的检验 2 个样本集的实验分布函数一致性的方法.

2 种方法有相同的原假设 H_0:

$$f(X|\theta_f) = f(X|\theta_s) \quad (2)$$

2.3.1　基于 Kolmogorov-Smirnov 检验的谓词排序模型

对于 X_s,$F_s(x)$ 是一个如下的阶梯函数:

$$F_s(x) = k/m \qquad x_{(i)} \leqslant x < x_{(i+1)} \quad (3)$$

其中,k 表示观测值小于等于 x 的值的个数,而 $\{x_{(i)}\}$ 是 x 的顺序统计量.

同理,可以定义 X_f 上的实验分布函数 $F_f(x)$:

$$F_f(x') = k'/n \qquad x'_{(j)} \leqslant x' < x'_{(j+1)} \quad (4)$$

Kolmogorov-Smirnov 检验计算以下统计量:

$$D = \max |F_s(x) - F_f(x)| \quad (5)$$

根据 Kolmogorov-Smirnov 检验,D 值越大,则拒绝原假设的可能性越高,即 $f(X|\theta_s)$ 与 $f(X|\theta_f)$ 之间的差异越大. 因此,可使用 D 值作为 $R(P)$ 的

一个近似,从而定义出 P 的“缺陷怀疑度”值 $s(P)$:

$$s(P) = D \approx R(P) \tag{6}$$

综合式(1)、式(5)和式(6),得出 Kolmogorov-Smirnov 检验的谓词 P 的缺陷相关度计算函数:

$$s(P) = \max | F_s(x) - F_f(x) | \tag{7}$$

该值越大,则相应谓词 P 是缺陷的可能性越高.

2.3.2 基于 Cramér-von-Mises 准则的排序模型

Cramér-von-Mises 准则首先对 X_s 以递增的顺序排序. 本文使用 X'_s 标记排序后的结果. 令 $X'_s = [x_1, x_2, \cdots, x_m]$ 是对 X_s 以递增的顺序排列后的结果. 其中,$x_1, x_2, \cdots, x_m$ 以升序排序, i 为 x_i 在 X'_s 中的序号. 同样,令 $X'_f = [x'_1, x'_2, \cdots, x'_n]$ 是对 X_f 以递增的顺序排列后的结果,其中 j 为 x_j 在 X'_f 中的序号. 同时,令 $r_1, r_2, \cdots, r_m$ 是联合用例(combined sample)中各 x 的序号,$S_1, S_2, \cdots, S_n$ 是联合用例中各 x' 的序号.

Cramér-von-Mises 准则计算以下统计量:

$$T = U/(nm(n+m)) - (4nm-1)/6(n+m) \tag{8}$$

其中

$$U = m\sum_{i=1}^{n}(r_i - i)^2 + n\sum_{i=1}^{n}(s_j - j)^2 \tag{9}$$

根据 Cramér-von-Mises 准则,T 值越大,则拒绝原假设的可能性越高,即 $f(X|\theta_s)$ 与 $f(X|\theta_f)$ 之间的差异越大. 因此,可以使用 T 值作为 $R(P)$ 的一个近似. 从而定义出 P 的“缺陷怀疑度”值:

$$s(P) = T \approx R(P) \tag{10}$$

综合式(1)、式(8)和式(10),得出基于 Cramér-von-Mises 准则的谓词 P 的缺陷相关度计算函数:

$$s(P) = U/(nm(n+m)) - (4nm-1)/6(n+m) \tag{11}$$

理论上,该值越大,则相应的谓词 P 是缺陷的可能性越高.

根据上述 2 种方法各自定义的“缺陷怀疑度”值 $s(P)$,按递减的顺序对所有缺陷相关谓词排序,得出有序的谓词列表.

3 实 验

3.1 实验对象

本文中使用西门子测试集作为实验对象. 西门子测试集最初由西门子研究院为研究测试充分性标准提出. 如表 1 所示,它包括 7 个程序的 132 个含缺陷版本. 每个程序的含缺陷版本是由 7 个程序中的 1 个加入 1 个缺陷产生. 这些程序可从 Software-artifact Infrastructure Repository (SIR)[16] 网站下载. 表 1 显示了这 7 个程序的以下统计信息,包括:程序名称,含缺陷版本数,代码可执行行数,测试用例数,失败用例中位数和成功用例中位数.

表 1 西门子测试集信息统计表

程序名称	含缺陷版本数	代码可执行行数	测试用例数	失败用例中位数	成功用例中位数
print_tokens	7	539	4130	38	4082
print_tokens2	10	489	4115	223	3892
replace	32	507	5542	83	5459
schedule	9	397	2650	31	2619
schedule2	10	299	2710	32	2678
tcas	41	174	1608	23	1585
tot_info	23	398	1052	71	981

3.2 实验环境的搭建

在 132 个含缺陷程序中,有 2 个没有执行失败的测试用例. 分别是“replace”的“v27”和“schedule2”的“v9”. 这个现象同样在先前的工作中已有指出[9]. 这 2 个版本被排除在外,因为本文中的模型和 SOBER 测试模型均需要成功与失败的测试用例同时出现. 为了评估本文中的模型,以所有的测试集作为本文中的模型和 SOBER 测试模型的输入,并使用分支和返回值(见 2.1 节)作为实验中谓词的位置.

此外,一般而言,一个谓词不会在每一个测试用例中都被执行到. 如果一个谓词在某个用例中没有被执行,$\pi(P)$ 的值等于 0/0. 在这种情况下,SOBER 认为应将 $\pi(P)$ 设为 0.5. 本文遵循此约定,对于在某个测试用例执行时未被执行的谓词,将该执行中此谓词的评估偏差值设为 0.5.

实验在 ThinkPad T61 笔记本上执行,其 CPU 为 Inter Core(TM)2 T7300@1.99 GHz,包含 2 GB 内存. 操作系统为 SunOS 5.10 Solaris,内核版本号为 Generic_141445-09.

3.3 实验评估框架

本节描述了基于实验设计给出的度量方式以评估缺陷定位系统的效率.

本文使用公式(12)作为实验的评估框架. 回顾许多先前的缺陷定位技术,如 CBI[10-11],SOBER[8-9] 等,均产生一个谓词列表. 其中所有缺陷相关的谓词都被赋予一个“怀疑度”值,并以降序排列以辅助测试人员定位程序缺陷. 同时,将程序中离“植入”的缺陷最近的谓词称为缺陷最相关谓词,并使用该谓词在报告的谓词列表中的索引作为缺陷定位技术效率高低的判断依据. 基于以

上特征得出公式(12)所示“得分”作为实验的评估框架.

假设以 L 表示谓词列表,$\tilde{P}$ 表示缺陷最相关谓词,缺陷定位方法的效率度量公式如下:

$$得分 = \tilde{P} 在 L 中的索引 / L 中谓词总数 \times 100\% \quad (12)$$

其中,L 中的第一个元素是以 1 为索引. 得分越小,缺陷最相关谓词在所报告的谓词列表中排位越靠前,则缺陷定位技术效率越高.

此外,一个谓词列表中可能存在 2 个不同谓词恰好具有相同“缺陷怀疑度”值 $s(P)$ 的情况. 此时,本文取具有相同 $s(P)$ 值的最后一个谓词在 L 中的索引作为公式(12)的分子.

3.4　实验结果

3.4.1　各程序实验结果评估

图 3 显示了基于 Cramér-von-Mises 准则的方法(以下简称 CM)、基于 Kolmogorov-Smirnov 检验的方法(以下简称 KS)与 SOBER 在 7 个西门子程序上缺陷定位的效率. 7 个坐标图按 7 个程序的规模依次排列. 本文以程序可执行语句总数定义程序规模,每个程序可执行语句总数列于表 1 的“代码可执行行数”列中. 其中 x 轴是按公式(12)计算得到的得分;y 轴表示定位缺陷的总数.

对于规模相对较小的程序“schedule”和“tcas”(图 3c 和图 3a),SOBER 定位缺陷的效果要优于 CM 和 KS. 对于其余程序,CM 和 KS 效率均高于 SOBER. 实验结果表明,随着程序规模的变大,相较于 SOBER,CM 和 KS 定位缺陷的效率更高. 其主要原因在于:随着程序规模的变大,程序中包含更多的谓词,因此在一次执行中很难执行到所有的谓词. 对于那些没有被执行到的谓词,SOBER 将该次执行中此谓词的评估偏差值设为 0.5. 当存在大量未被执行到的谓词时,样本集会有大量的值为 0.5 的样本. 这将使一个谓词来自成功执行与失败用例的评估偏差的分布进一步偏离正态分布. 因此,SOBER 将失去判断力并产生非鲁棒的结果. 而通过使用非参数假设检验可以不受分布不服从正态分布的影响,从而具有更好的缺陷定位效果.

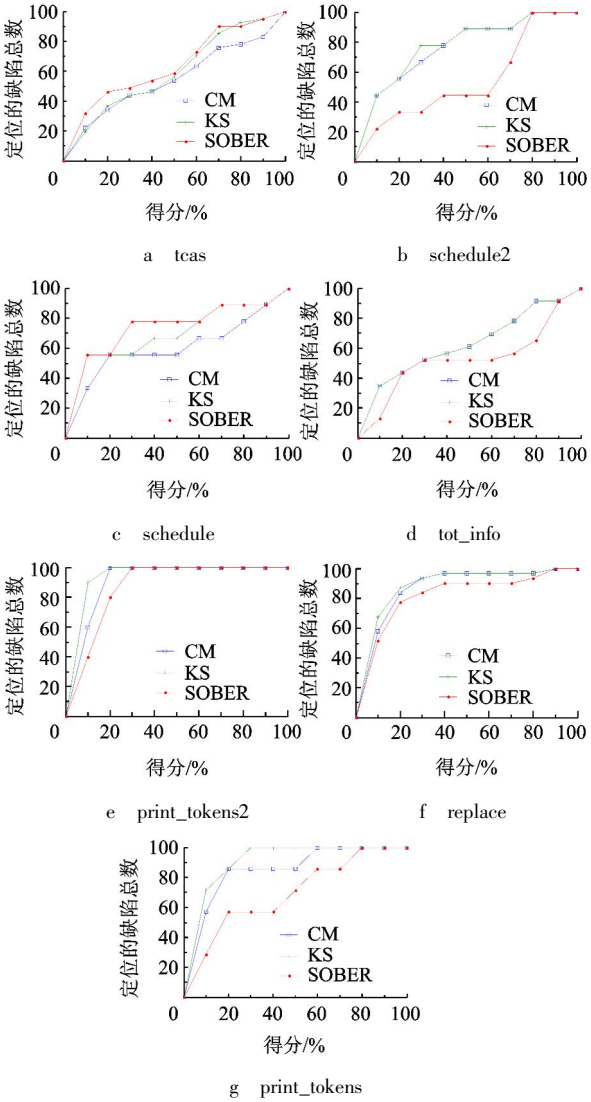


图 3　3 种方法在各程序上定位效率比对图

3.4.2　实验结果整体评估

图 4 描述了通过检测谓词列表中谓词的百分比(如谓词列表中前 10% ~20% 的谓词),程序员可定位的缺陷数目. 从实际使用出发,了解在检查谓词列表中前 $\alpha\%$ 的谓词后多少缺陷可以被定位,可以全面的了解缺陷定位方法的效率. 因此,图 5 给出了方法的累计比对效果. 图 4 与图 5 中

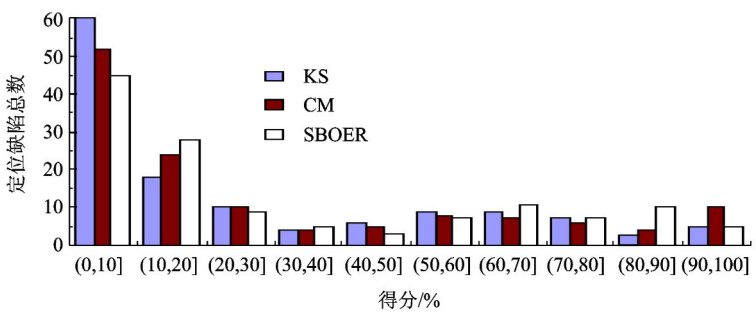


图 4　各得分区间方法定位缺陷数对比关系图

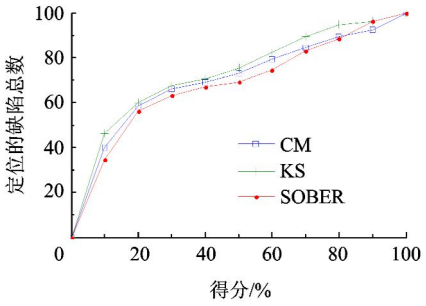


图 5 定位缺陷效率总体对比关系图

的 x 轴与 y 轴的含义与图 3 中一致.

由公式(12)可知,“得分”越小,则在所报告的缺陷列表中,测试人员定位一个程序缺陷所需要查找的谓词数量越少,相应的缺陷定位手段的效率越高. 经验表明,一般测试人员希望通过只检查上述列表中前 10% 左右的谓词就能定位到缺陷. 从图 4 可以看出,在得分区间(0%,10%]中,KS 可以定位 130 个缺陷中的 60 个,定位这 60 个缺陷,KS 至多需要检查谓词列表中前 10% 的谓词;CM 可以定位出 52 个. 同时,在该区间中,KS 和 CM 的效率均优于 SOBER(SOBER 能定位 45 个).

图 5 显示,在西门子测试集的 130 缺陷中,当一个程序员希望检测至多 10% 的相关谓词定位程序缺陷时,SOBER 可以捕获 34.62% 的缺陷(130 个中的 45 个),CM 可以捕获 40.00% 的缺陷(130 个中的 52 个),而 KS 可以定位 46.15% 的缺陷(130 个中的 60 个). 当测试人员可以忍受检查谓词列表中前 20% 的谓词来定位缺陷时,SOBER 可以定位 56.15% 的缺陷,CM 可以捕获 58.46% 的缺陷,而 KS 可以定位 60.00% 的缺陷. 进一步,如果检查谓词列表中前 50% 的谓词,CM 可以定位 73.08%(130 个缺陷中的 95 个)的缺陷,而 KS 可以定位 75.38%(130 个缺陷中的 98 个)的缺陷.

同时,图 5 显示整体上 CM 在得分为[0%,80%]区间中优于 SOBER;而在(80%,100%]区间中,SOBER 优于 CM. 然而,由于得分落在(80%,100%]区间中意味着:为了定位缺陷,一个程序员至少检测报告的谓词列表中前 80% 的谓词. 本文认为这样的优势是微不足道的. 因此,可以认为整体上 CM 在缺陷定位上优于 SOBER.

图 5 同样显示,整体上 KS 在所有得分区间上,均优于 CM. 对比 SOBER,图 5 显示 KS 在得分为[0%,90%]区间中效率占优,在[90%,100%]区间中,两者效率相当. 整体上可以认为 KS 效率高于 SOBER.

综上所述,可以认为基于非参数假设检验的方法在缺陷定位上优于基于中心极限定理的参数假设检验的方法. 并且,对于大规模的程序中,谓词评估偏差不服从正态分布的情况,有更好的鲁棒性和适应性. 在 2 种用于检验 2 个样本集上实验分布函数间差异的常用方法中,Kolmogorov-Smirnov 检验较之 Cramér-von-Mises 准则在缺陷定位上更有效.

3.5 有效性的影响因素

一个影响因素来自本文所选择调查的谓词. 文献[9]发现不追踪标量对能减少近一半的工作量但并不严重影响缺陷定位的效果. 因此,SOBER 不追踪有关标量对的谓词. 与 SOBER 一样,本文使用分支(branches)和返回值(returns)作为实验中的谓词.

外部的影响因素来自所选的西门子测试集. 由于西门子程序中的缺陷由人为手工植入,有可能它们并不能真正代表实际工程中的程序缺陷. 同时,由于西门子测试集包含的主要是小规模的程序,在后续工作中应该进一步验证本文模型在大规模程序中的有效性.

4 结 束 语

缺陷定位通常耗费大量的时间,人力和物力,却是在软件调试研究中一个重要的课题. 许多早先的研究通过比对成功与失败用例的特征定位缺陷至相关的谓词(或者直接定位至含缺陷的语句).

本文提出了基于非参数假设检验模型和程序谓词的自动缺陷定位方法. 同时,在西门子测试集上面通过实验验证了该方法的效率. 对比实验表明,基于非参数假设检验模型的方法在缺陷定位上优于先前提出的基于中心极限定理的参数假设检验的方法. 在同属于非参数假设检验范畴的 Kolmogorov-Smirnov 检验与 Cramér-von-Mises 准则的比较中发现,基于 Kolmogorov-Smirnov 检验的方法在缺陷定位上更有效.

在更多的广泛使用的软件中进一步确认非参数假设检验在定位程序缺陷方面的有效性将是本文作者的下一步工作.

参考文献(References)

[1] Ernst M D, Cockrell J, Griswold W G, et al. Dynamically discovering likely program invariants to support program evolution[J]. IEEE Transactions on Software Engineering, 2001, 27(2): 99 – 123

[2] Hangal S, Lam M S. Tracking down software bugs using automatic anomaly detection[C]//Proceedings of the 24th International Conference on Software Engineering. Orlando: ACM, 2002: 291 – 301

[3] Wong W E, Qi Y, Zhao L, et al. Effective fault localization using code coverage[C]//Proceedings of the 31st Annual International Computer Software and Application Conference. Beijing: IEEE Computer Society, 2007: 449 – 456

[4] Li W, Harrold M J, Görg C. Detecting user-visible failures in AJAX web applications by analyzing users' interaction behaviors [C]//Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering. Antwerp, Belgium: ACM, 2010: 55 – 158

[5] Park S, Vuduc R W, Harrold M J. Falcon: fault localization in concurrent programs[C]//Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering. Cape Town, South Africa: ACM, 2010: 245 – 254

[6] Kim M, Sinha S, Görg C, et al. Automated bug neighborhood analysis for identifying incomplete bug fixes[C]//Proceedings of the 3rd International Conference on Software Testing. Paris: IEEE Computer Society, 2010: 383 – 392

[7] Santelices R A, Harrold M J, Orso A. Precisely detecting runtime change interactions for evolving software[C]//Proceedings of the 3rd International Conference on Software Testing. Paris: IEEE Computer Society, 2010: 429 – 438

[8] Liu C, Han J. Failure proximity: a fault localization-based approach[C]//Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering. Portland, Oregon: ACM, 2006: 46 – 56

[9] Liu C, Fei L, Yan X, et al. Statistical debugging: a hypothesis testing-based approach[J]. IEEE Transactions on Software Engineering, 2006, 32(10): 831 – 848

[10] Liblit B, Aiken A, Zheng A X, et al. Bug isolation via remote program sampling[C]//Proceedings of the 2003 ACM SIGPLAN Conference on Programming Language Design and Implementation. San Diego, California: ACM, 2003: 141 – 154

[11] Liblit B, Naik M, Zheng A X, et al. Scalable statistical bug isolation[C]//Proceedings of the 2005 ACM SIGPLAN Conference on Programming Language Design and Implementation. Chicago: ACM, 2005: 15 – 26

[12] Zhang Z, Chan W K, Tse T H, et al. Is non-parametric hypothesis testing model robust for statistical fault localization? [J]. Information and Software Technology, 2009, 51(11): 1573 – 1585

[13] Hutchins M, Foster H, Goradia T, et al. Experiments of the effectiveness of dataflow-and controlflow-based test adequacy criteria[C]//Proceedings of the 16th International Conference on Software Engineering. Sorrento: IEEE Computer Society/ACM Press, 1994: 191 – 200

[14] Zwillinger D, Kokoska S. Standard probability and statistics tables and formulae[M]. Boca Raton: Chapman and Hall/CRC, 2000: 341 – 343

[15] Anderson T W. On the distribution of the two-sample cramér von mises criterion[J]. The Annals of Mathematical Statistics, 1962, 33(3): 1148 – 1159

[16] Do H, Elbaum S, Rothermel G. Supporting controlled experimentation with testing techniques: an infrastructure and its potential impact[J]. Empirical Software Engineering, 2005, 10(4): 405 – 435

(编　辑:文丽芳)

(上接第 1004 页)

[17] Miao Zhenjiang. Zernike moment-based image shape analysis and its application[J]. Pattern Recognition Letters, 2000, 21(2): 169 – 177

[18] Born M, Wolf E. Principles of optics: electromagnetic theory of propagation, interference and diffraction of light[M]. New York: Cambridge University Press, 1999

[19] Billinge S J L, Thorpe M F. Local structure from diffraction [M]. New York: Springer Us, 1998

[20] Garcia-Sucerquia J, Medina F F, Barrera J F. Minute details detection through Fresnel diffraction domain[J]. Optics Communications, 2005, 253(4 – 6): 250 – 256

[21] Sabatyan A, Taghi Tavassoly M. Determination of refractive indices of liquids by Fresnel diffraction[J]. Optics & Laser Technology, 2009, 41(7): 892 – 896

[22] Goodman J W. Introduction to Fourier optics[M]. Greenwood Village, Colorado, USA: Roberts & Company Publishers, 2005

[23] Liebling M, Blu T, Unser M. Fresnelets: new multiresolution wavelet bases for digital holography[J]. IEEE Transactions on Image Processing, 2003, 12(1): 29 – 43

[24] Liebling M, Blu T, Unser M. Fresnelets: A new wavelet basis for digital holography[C]//Proceedings of the SPIE Conference on Mathematical Imaging: Wavelet Applications in Signal and Image Processing IX. San Diego, CA, USA: Citeseer, 2001, 4478: 347 – 352

[25] 蔡履中,王永钤. 菲涅耳衍射在标度-距离变换下的相似性[J]. 山东大学学报:理学版, 2001, 36(1): 56 – 61
Cai Lvzhong, Wang Yongqian. The Similiarity of Fresnel diffraction under scale-distance transform[J]. Journal of Shandong University: Natural Science Edition, 2001, 36(1): 56 – 61 (in Chinese)

[26] Lazebnik S, Schmid C, Ponce J. A maximum entropy framework for part-based texture and object recognition[C]//Tenth IEEE International Conference on Computer Vision, 2005. 2005, 1: 832 – 838

(编　辑:张　嵘)