基于冗余覆盖信息约简的软件缺陷定位方法

王浩仁[1],崔展齐[1*],岳 雷[1],陈 翔[2],郑丽伟[1]
(1. 北京信息科技大学计算机学院,北京 100101;2. 南通大学计算机科学与技术学院,江苏南通 226019)

摘 要: 软件规模和复杂程度的不断提高,为软件质量保障带来了严峻的挑战. 软件缺陷定位是一种重要的软件质量保障技术,其中基于频谱的缺陷定位(Spectrum-based Fault Localization,SFL)是应用最为广泛的软件缺陷定位技术,其通过分析语句覆盖信息矩阵计算代码语句的可疑度值,并根据可疑度值定位缺陷所在语句. 然而,语句覆盖信息矩阵中存在着严重的数据冗余问题,冗余的数据极大地影响了SFL的缺陷定位性能. 以Defects4J数据集中395个程序的语句覆盖信息矩阵为例,在超过一半的语句覆盖信息矩阵中有90%的语句存在与其具有相同覆盖信息的语句. 特征选择是常用的数据预处理技术,通过去除冗余和不相关特征来获取原始特征集中有价值的特征子集. 因此,我们将语句覆盖信息矩阵作为原始特征集,将冗余覆盖信息约简建模为特征选择问题,提出了一种基于冗余覆盖信息约简的软件缺陷定位方法(Fault Localization based on Redundant coverage information Reduction,FLRR). 首先,使用特征选择技术对语句覆盖信息和测试用例执行结果组成的语句覆盖信息矩阵进行约简,得到语句覆盖信息矩阵子集;然后,使用SFL计算语句覆盖信息矩阵子集中语句的可疑度值,并根据可疑度值对语句进行降序排列,以定位缺陷语句. 本文使用六种常用的特征选择技术对语句覆盖信息矩阵进行特征选择和约简,以得到语句覆盖信息矩阵子集,并使用四种典型的SFL技术对语句覆盖信息矩阵子集中的语句进行缺陷定位. 为评估FLRR的缺陷定位性能,本文使用$E_{inspect}@n$和MRR(Mean Reciprocal Rank)评价指标在基于Defects4J的数据集上与四种典型的SFL技术进行了对比实验. 实验结果表明,FLRR能够有效提升SFL的缺陷定位性能. 对于$E_{inspect}@n$指标,当n=1时,FLRR相比DStar、Ochiai、Barinel和OP2分别多定位到23条、26条、14条和13条缺陷语句,分别增加了69.70%、76.47%、45.16%和38.24%;对于MRR指标,FLRR相比DStar、Ochiai、Barinel和OP2分别提升了20.08%、24.94%、17.45%和19.15%.

关键词: 缺陷定位;特征选择;软件调试;可疑度;语句覆盖信息;测试用例

基金项目: 江苏省前沿引领技术基础研究专项(No.BK20202001);国家自然科学基金(No.61702041);北京信息科技大学"勤信人才"培育计划(No.QXTCPC201906)

中图分类号: TP311.5 **文献标识码:** A **文章编号:** 0372-2112(2024)01-0324-14

电子学报URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20220820

Software Fault Localization Based on Redundant Coverage Information Reduction

WANG Hao-ren[1], CUI Zhan-qi[1*], YUE Lei[1], CHEN Xiang[2], ZHENG Li-wei[1]
(1. *School of Computer Science, Beijing Information Science and Technology University, Beijing* 100101, *China*;
2. *School of Computer Science and Technology, Nantong University, Nantong, Jiangsu* 226019, *China*)

Abstract: As the scale and complexity of software increase, it becomes more difficult to ensure its quality and reliability. Some of the most important software quality and reliability assurance methods are software fault localization techniques, of which spectrum-based fault localization (SFL) is the most commonly used. SFL calculates the suspicious values of code statements by analyzing the statement coverage matrix, locating the faulty statements according to the suspicious values. However, the statement coverage matrix suffers from a serious redundancy problem, which severely impairs the fault localization performance of SFL. For instance, in more than half of the statement coverage matrices of 395 programs in the Defects4J dataset, there are other statements with the same coverage information for 90% of the statements. Feature

收稿日期:2022-07-14;修回日期:2023-02-24;责任编辑:梅志强
*通讯作者:崔展齐

selection, a data preprocessing technique, is often used to obtain valuable feature subsets by removing redundant and irrelevant features. We propose a software fault localization approach, based on redundant coverage information reduction (FLRR), by taking the statement coverage matrix as the original feature set and modeling the reduction of redundant coverage information as a feature selection problem. First, feature selection techniques are applied to reduce the statement coverage matrix, which includes both statement coverage information and test case execution results, to obtain a subset of the matrix. Second, SFL is used to calculate the suspicious values of statements in the statement coverage matrix subset, and the statements are sorted in descending order according to their suspicious values. The method presented in this paper uses six common feature selection techniques to perform feature selection and reduction on the statement coverage matrix, to obtain the subset of the matrix, and then uses four typical SFL techniques to localize faulty statements in the subset. To evaluate the fault localization performance of FLRR, comparative experiments were conducted with four typical SFL techniques on the Defects4J dataset, using $E_{\text{inspect}}@n$ and MRR (Mean Reciprocal Rank) as evaluation metrics. Experimental results show that FLRR can improve the fault localization performance of SFL. When compared with DStar, Ochiai, Barinel, and OP2, FLRR located 23, 26, 14, and 13 more faulty statements, improved $E_{\text{inspect}}@n\,(n=1)$ by 69.70%, 76.47%, 45.16%, and 38.24%, and improved MRR by 20.08%, 24.94%, 17.45%, and 19.15%, respectively.

Key words: fault localization; feature selection; software debugging; suspicious value; statement coverage information; test case

Foundation Item(s): Jiangsu Provincial Frontier Leading Technology Fundamental Research Project (No. BK20202001); National Natural Science Foundation of China (No.61702041); Beijing Information Science and Technology University "Qin-Xin Talent" Cultivation Project (No.QXTCPC201906)

1 引言

随着软件规模的扩大和复杂度不断提高,如何保障软件质量成为了软件工程研究人员的关注焦点. 软件调试是检测软件缺陷、保障软件质量的重要手段,其中缺陷定位是软件调试过程中最费时费力的一个环节. 为此,已有研究提出了一系列软件缺陷定位技术,如:基于频谱的缺陷定位技术(Spectrum-based Fault Localization, SFL)[1,2]、基于变异的缺陷定位技术(Mutation-based Fault Localization, MFL)[3,4]及其他缺陷定位技术[5,6]等. 其中,SFL是一类应用最为广泛的软件缺陷定位技术,主要通过分析测试用例执行结果和语句覆盖信息构成的语句覆盖信息矩阵来识别程序中的可疑语句[2]. SFL技术已取得了较好的缺陷定位性能,例如,使用DStar[7]对Defects4J[8]数据集中的程序进行缺陷定位时,通过检查排名前10的可疑语句可以定位到约39%的缺陷语句. 研究表明,开发人员在检查给定的可疑语句列表时,更关注排名靠前特别是排名第1的语句[9]. 然而,在检查排名第1的可疑语句时,DStar仅能定位到8.33%的缺陷语句. 因此,SFL的精度还有待提高.

通过对语句覆盖信息矩阵进行分析后,我们发现其存在严重的数据冗余现象,会影响SFL定位缺陷的精度. 这里的数据冗余是指语句覆盖信息矩阵中存在冗余特征和无关特征. 其中,冗余特征大量或完全重复了其他特征中含有的信息,而无关特征是指不能为机器学习模型提供有用信息的特征. 在本文中冗余特征即具有相同覆盖信息的语句,而无关特征表示与测试用例执行结果相关度极低的语句. 数据冗余会增加模型的构建时间,导致模型准确度降低. 语句覆盖信息矩阵中也同样存在数据冗余情况,会对软件缺陷定位技术带来干扰,降低缺陷定位精度. 例如,使用DStar对Lang-63版本(Lang项目的第63个版本)进行缺陷定位时,共有37条语句与缺陷语句的可疑度值相同,导致需要依次检查33行可疑语句才能检查到缺陷语句,这些冗余语句极大地影响了缺陷定位精度. 分析发现,在Defects4J数据集395个程序的语句覆盖信息矩阵中,超过半数矩阵中90%的语句是冗余的. 例如:Closure-107版本的语句覆盖信息矩阵中共包含34 974条语句,其中仅有322条语句具有与其他语句均不相同的覆盖信息,即语句覆盖信息矩阵中99.08%的语句存在冗余情况. 此外,该版本中甚至有16 910条语句的覆盖信息完全相同,占该程序总语句的48.35%. 这表明语句覆盖信息矩阵中存在严重冗余问题,会对缺陷定位结果产生影响.

为此,我们将SFL中的冗余覆盖信息约简建模为特征选择(feature selection)问题,将特征选择技术引入SFL来提升缺陷定位的性能. 特征选择是常用的数据预处理技术,通过去除冗余和不相关特征来获取原始特征集中最有价值的特征子集,以达到减少计算开销、提高机器学习模型性能的目的[10,11],特征选择通过降低数据冗余度提升模型性能,被广泛用于网络安全和计算机视觉等领域. 例如,CFS-BA(Correlation-based Feature Selection-Bat Algorithm)[12]通过在RF(Random

Forest)[13]等模型的训练过程中进行特征选择,提升检测网络入侵的准确率;CBIR(Content-Based Image Retrieval)[14]通过使用在线特征选择技术去除冗余特征,以提高图像检索精度.受这类研究启发,本文对语句覆盖信息矩阵进行特征选择,通过降低语句覆盖信息矩阵的冗余度来提升缺陷定位性能.

基于上述分析,本文提出了基于冗余覆盖信息约简的软件缺陷定位方法(Fault Localization based on Redundant coverage information Reduction,FLRR).首先,对语句覆盖信息和测试用例结果组成的语句覆盖信息矩阵进行约简,得到语句覆盖信息矩阵子集;然后,计算语句覆盖信息矩阵子集中剩余语句的可疑度值以进行缺陷定位.现有基于频谱的软件缺陷定位工作未关注语句覆盖信息矩阵中的数据冗余问题,缺陷定位效果会受冗余数据的影响.FLRR方法通过对SFL的输入数据,即语句覆盖信息矩阵进行特征选择,降低语句覆盖信息矩阵中数据的冗余度,可提高缺陷定位性能.本文在数据集Defects4J上进行的实验结果表明,FLRR能够提升缺陷定位性能,对于$E_{\text{inspect}}@n$指标,当n=1时,FLRR相比DStar、Ochiai[15]、Barinel[16]和OP2[17]分别多定位到23条、26条、14条和13条缺陷语句,分别增加了69.70%、76.47%、45.16%和38.24%;对于MRR(Mean Reciprocal Rank)指标,FLRR相比DStar、Ochiai、Barinel和OP2分别提升了20.08%、24.94%、17.45%和19.15%.

本文的主要贡献如下:

(1)首次分析了SFL中使用覆盖信息存在的严重冗余现象,将其建模为特征选择问题,并提出一种基于冗余覆盖信息约简的软件缺陷定位方法来提升SFL的缺陷定位效果.

(2)使用Pearson相关系数、互信息系数等三类共六种特征选择技术对Defects4J数据集中语句覆盖信息矩阵的数据冗余情况进行了分析.

(3)基于所提出的方法实现了软件缺陷定位工具,在Defects4J数据集上进行了实验,并与SFL进行比较,以验证所提出方法的有效性.

2 基于冗余覆盖信息约简的软件缺陷定位方法

为解决语句信息覆盖矩阵中严重的数据冗余问题,提升软件缺陷定位精度,本文提出了基于冗余覆盖信息约简的软件缺陷定位方法,流程如图1所示.首先,对语句覆盖信息和测试用例执行结果组成的语句覆盖信息矩阵进行约简,以得到语句覆盖信息矩阵子集;然后,计算语句覆盖信息矩阵子集中语句的可疑度值,得到可疑语句列表以进行缺陷定位.特征选择技术通常使用重要性、相关性和一致性等作为评价指标,以对数据集进行特征选择.本文中使用了多种特征选择技术来约简冗余覆盖信息,这些技术的评估指标不尽相同,为便于描述,我们将重要性等评价指标统称为相关性.

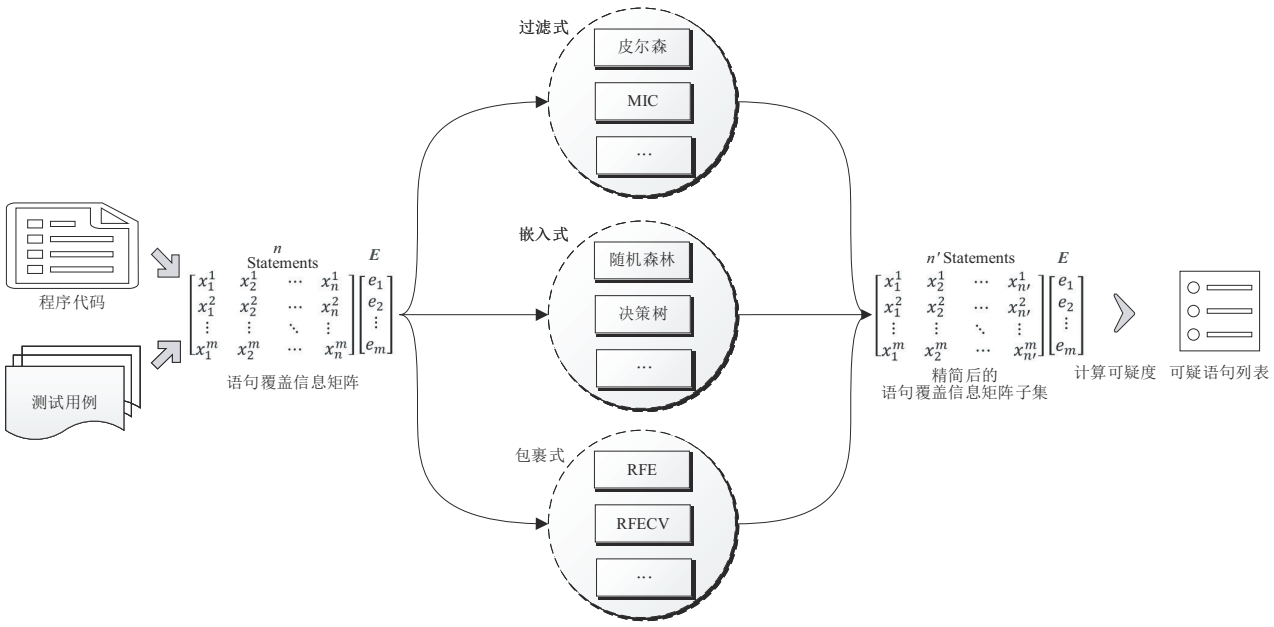


图1 基于冗余覆盖信息约简的软件缺陷定位方法流程图

2.1 使用特征选择技术对冗余覆盖信息进行约简

本小节将介绍如何使用特征选择技术对冗余覆盖信息进行约简.特征选择技术是机器学习模型训练中关键的数据预处理步骤,通过去除冗余和不相关的特征,可获取原始特征集中最有价值的子集.常见的特征选择技术可分为三类:过滤式(filter)、嵌入式(embed-

ding)和包裹式(wrapper)特征选择技术[10,11]本文将使用上述三类技术对语句覆盖信息矩阵进行特征选择,以解决SFL中所使用语句覆盖信息矩阵的数据冗余问题.

首先根据程序语句覆盖信息和测试用例执行结果得到语句覆盖信息矩阵,图1中的语句覆盖信息矩阵包括语句覆盖信息和测试用例执行结果.对于由n条语句组成的程序$P=\{s_1, s_2, \cdots, s_i, \cdots, s_n\}$,使用包含$m$个测试用例的测试集$T=\{t_1, t_2, \cdots, t_j, \cdots, t_m\}$进行测试.矩阵左侧的$x_i^j$表示语句覆盖信息,若语句$s_i$被测试用例$t_j$执行覆盖,则$x_i^j$为1,否则为0.测试集$T$对语句$s_i$的覆盖情况为向量$\boldsymbol{X}_i=[x_i^1, x_i^2, \cdots, x_i^m]^{\mathrm{T}}$,对所有语句的覆盖情况为矩阵$[\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_i, \cdots, \boldsymbol{X}_n]$.矩阵右侧的$\boldsymbol{E}=[e_1, e_2, \cdots, e_j, \cdots, e_m]^{\mathrm{T}}$为测试集$T$的执行结果,若测试用例$t_j$执行通过,则$e_j$为0,若执行未通过则$e_j$为1.通过这些信息,可以计算出每条语句的特征相关性,即语句覆盖信息对测试用例执行结果的影响程度,并通过特征相关性大小筛选出排名前k的语句,以得到语句覆盖信息矩阵子集.

我们在过滤式、嵌入式和包裹式三类特征选择技术中分别选择两种常用技术对语句覆盖信息矩阵进行特征选择.其中,过滤式特征选择技术选择了皮尔森相关系数[18]和最大信息系数[19];嵌入式特征选择技术选择了随机森林和决策树[20];包裹式特征选择技术选择了递归特征消除和交叉验证递归特征消除[11].

2.1.1　过滤式特征选择

2.1.1.1　皮尔森相关系数

皮尔森相关系数反映了两个变量线性相关性的强弱程度,是过滤式特征选择技术中最常用的一种算法.在本文中,通过计算语句s_i的覆盖信息$\boldsymbol{X}_i$与测试用例结果间的协方差和两者标准差乘积的比值,得到语句与测试用例执行结果间的皮尔森相关系数r_i,计算方法如式(1)所示.

$$r_i=\frac{\sum_{j=1}^{m}(x_i^j-\overline{x_i})(e_j-\overline{e})}{\sqrt{\sum_{j=1}^{m}(x_i^j-\overline{x_i})}\times\sqrt{\sum_{j=1}^{m}(e_j-\overline{e})}} \tag{1}$$

其中,$\overline{x_i}$为测试用例集T覆盖语句s_i情况的平均值,$\overline{x_i}=\sum_{j=1}^{m}x_i^j/m$;$\overline{e}$为测试用例集$T$执行结果的平均值,$\overline{e}=\sum_{j=1}^{m}e_j/m$.

相关系数r_i的值介于-1到1之间,绝对值越大则相关性越强,即语句与测试用例执行结果的相关性越强.当$r_i>0$时,表明语句覆盖信息$\boldsymbol{X}_i$与测试用例执行结果$\boldsymbol{E}$呈线性正相关;当$r_i<0$时,表明语句覆盖信息$\boldsymbol{X}_i$同测试用例执行结果$\boldsymbol{E}$呈线性负相关;当$r_i=0$时,表明语句覆盖信息$\boldsymbol{X}_i$同测试用例执行结果$\boldsymbol{E}$不存在线性相关.

2.1.1.2　互信息系数

在信息论中,互信息常用于度量两个随机变量间的相互依赖性.可通过计算语句s_i的覆盖信息$\boldsymbol{X}_i$和测试用例执行结果$\boldsymbol{E}$之间的互信息系数MI(Mutual Information),来度量语句与测试用例执行结果间的相关性.MI的计算方法如式(2)所示.

$$\mathrm{MI}(\boldsymbol{X}_i,\boldsymbol{E})=\sum_{j=1}^{m}p(x_i^j,e_j)\log\frac{p(x_i^j,e_j)}{p(x_i^j)p(e_j)} \tag{2}$$

$\boldsymbol{X}_i$与的$\boldsymbol{E}$互信息系数是指二者的联合分布概率与边缘分布概率的相对熵.其中,$p(x_i^j,e_j)$为x_i^j与e_j的联合分布概率,表示在$[\boldsymbol{X}_i;\boldsymbol{E}]$中与$(x_i^j,e_j)$相同的元素个数与总元素个数$m$的比值;$p(x_i^j)$和$p(e_j)$分别为$x_i^j$和$e_j$的边缘分布概率,其中,$p(x_i^j)$表示在$\boldsymbol{X}_i$中与$x_i^j$值相同的元素个数与总元素个数$m$的比值,$p(e_j)$表示在$\boldsymbol{E}$中与$e_j$值相同的元素个数与总元素个数$m$的比值.互信息系数的取值范围在[0, 1]之间,$\mathrm{MI}(\boldsymbol{X}_i;\boldsymbol{E})$的值越大,则$\boldsymbol{X}_i$和$\boldsymbol{E}$之间依赖性越高.

2.1.2　嵌入式特征选择

2.1.2.1　决策树

决策树(Decision Tree, DT)是一种常用的分类方法.决策树模型呈树形结构,也可以看作一种if-then规则的集合.在分类问题中,决策树表示基于特征对实例进行分类的过程.在决策树训练过程中需要从数据集中选择特征作为节点,并根据不同的分裂标准计算各个特征与测试用例执行结果之间的相关性.

在对语句覆盖信息矩阵进行特征选择时,可将特征选择嵌入到决策树模型的训练过程中,通过度量各个语句对模型精度影响程度的大小来进行特征选择.具体来讲,可使用式(3)计算各个语句与测试用例执行结果之间的相关性.相关性的值越大,特征对模型预测精度的影响程度越大,即特征对应语句对测试用例执行结果影响越大.

$$\mathrm{imp}(s_i)=\frac{\mathrm{num}(s_i)\times\mathrm{mea}(s_i)-\sum_{k=0}^{1}\mathrm{num}(s_k^i)\times\mathrm{mea}(s_k^i)}{m} \tag{3}$$

其中,s_i表示当前数据集中用来分类的特征(即语句)节点,$\mathrm{num}(s_i)$表示覆盖语句s_i的测试用例数,即$\boldsymbol{X}_i$中等于1的元素个数;$\mathrm{num}(s_k^i)$表示被分到s_k^i的样本个数,s_1^i、s_0^i为s_i的两个子节点,其中$\mathrm{num}(s_1^i)$为覆盖s_i的未通过测试用例数,$\mathrm{num}(s_0^i)$为覆盖s_i的通过测试用例数;$\mathrm{mea}(s_i)$、$\mathrm{mea}(s_k^i)$分别表示使用某种分裂标准对节点s_i和s_k^i进行度量的结果;分母m为测试用例的总数.分裂

标准可以采用信息增益、基尼系数等,我们使用基尼系数作为分裂标准,其计算如式(4)所示.

$$\mathrm{Gini}(s_i)=\sum_{k=0}^{1}p_k(1-p_k) \tag{4}$$

其中,p_1为覆盖当前语句测试用例未通过的概率,即$\mathrm{num}\left(s_1^i\right)/\left(\mathrm{num}\left(s_0^i\right)+\mathrm{num}\left(s_1^i\right)\right)$,$p_0$表示覆盖当前语句测试用例通过的概率,即$\mathrm{num}\left(s_0^i\right)/\left(\mathrm{num}\left(s_0^i\right)+\mathrm{num}\left(s_1^i\right)\right)$.基尼系数可反映数据集的不确定程度,即语句覆盖信息矩阵以s_i划分时的不确定程度.

2.1.2.2 随机森林

随机森林(Random Forest,RF)是基于随机决策树的机器学习模型,具有准确率高、鲁棒性好、易于使用等优点,这使得它成为了目前最流行的机器学习模型之一[10].在拟合数据后,RF会对特征进行相关性度量,因此也可以将嵌入式特征选择技术应用在模型的训练过程中.

RF中同样可使用式(3)计算语句覆盖信息矩阵中所有特征的相关性.与决策树相同,特征相关性值越大,则该特征对模型预测的准确度影响就越大,即对测试用例的执行结果影响越大.

2.1.3 包裹式特征选择

2.1.3.1 递归特征消除

递归特征消除RFE(Recursive Feature Elimination)是一个基于序列后向选择的算法,它与嵌入式特征选择技术相似,也将特征选择和模型训练过程融合,不同之处是RFE会对每个特征的相关性得分进行排序,以删除相关性最低的特征,并使用剩余特征再次训练模型,以进行下一次特征选择,最后选出符合终止条件的特征子集.

RFE需要进行多次训练,其相关性计算方式与嵌入式相同.首先,通过把语句的覆盖信息$[\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_i,\boldsymbol{X}_n]$($\boldsymbol{X}_i=[x_i{}^1,x_i{}^2,\cdots,x_i,\cdots,x_i{}^m]^{\mathrm{T}}$,其中,$x_i{}^j$的取值为0或1)和测试用例执行结果$\boldsymbol{E}=[e_1,e_2,\cdots,e_j,\cdots,e_m]^{\mathrm{T}}$输入模型训练,即将嵌入式特征选择技术应用在模型的训练过程中,并计算出每条语句与模型准确度的相关性,每次训练均会根据相关性大小对语句覆盖信息矩阵进行约简,最终得到符合终止条件的语句覆盖信息矩阵子集.

2.1.3.2 交叉验证的递归特征消除

交叉验证的递归特征消除RFECV(Recursive Feature Elimination Cross Validation)在RFE基础上增加了交叉验证.首先,根据RFE确定的特征相关性,多次进行特征集合约简,每次删除相关性最小的特征.然后,对选定的特征集进行交叉验证,将验证过程中模型预测准确度最高的特征数量作将要选择的特征数量,并按照特征相关性大小进行特征选择.

RFECV也需要将语句覆盖信息$[\boldsymbol{X}_1,\boldsymbol{X}_2,\cdots,\boldsymbol{X}_n]$和测试用例执行结果$\boldsymbol{E}=[e_1,e_2,\cdots,e_j,\cdots,e_m]^{\mathrm{T}}$输入模型进行多次训练.在RFECV的交叉验证中,通过将不同特征(语句)列组合,形成训练集,并计算出相同特征数量训练得到模型准确度的均值,以得到最优特征数量.当减少语句数量会造成模型准确度损失时,将不再继续约简特征,并将剩余的特征列组成语句覆盖信息矩阵子集.

2.2 可疑度值计算

在本小节中将介绍如何使用可疑度公式得到语句覆盖信息矩阵中各语句的可疑度值,以得到根据可疑度值降序排列的可疑语句列表.

语句覆盖信息矩阵是描述软件行为特征的信息矩阵[2],通常会记录实体(例如语句,类等)执行测试套件的信息,如实体是否被测试用例覆盖或实体被测试用例覆盖的次数等.语句执行测试用例的二进制覆盖状态信息是SFL技术中最为常用的频谱信息[21].

常用的可疑度公式大多涉及到4个参数a_{np}、a_{nf}、a_{ep}、a_{ef}.其中,a_{np}表示未覆盖当前语句的通过测试用例数;a_{nf}表示未覆盖当前语句的未通过测试用例数;a_{ep}表示覆盖当前语句的通过测试用例数;a_{ef}表示覆盖当前语句的未通过测试用例数.通常,语句的可疑度值与a_{ep}和a_{nf}成反比,与a_{ef}成正比,即当一个语句具有相对较高的a_{ef}值及较低的a_{ep}、a_{nf}值时,则该语句具有较高的可疑度值[1].

在FLRR计算可疑度时,首先根据语句信息覆盖矩阵子集获取语句的a_{np}、a_{nf}、a_{ep}、a_{ef}等参数,以使用可疑度公式计算每条语句的可疑度值,然后根据可疑度值对语句进行排名,定位缺陷语句的所在位置.

相关实证研究[20,21]和综述[1]表明,DStar、Ochiai、Barinel和OP2是最有效的SFL技术.因此,本文在实现FLRR过程中采用上述四种SFL技术计算语句信息覆盖矩阵子集中各语句的可疑度值.它们的可疑度计算方法分别如式(5)~(8)所示.

$$\mathrm{DStar}=\frac{a_{\mathrm{ef}}^{*}}{a_{\mathrm{nf}}+a_{\mathrm{ep}}},*=2,3,5,\cdots \tag{5}$$

$$\mathrm{Ochiai}=\frac{a_{\mathrm{ef}}}{\sqrt{(a_{\mathrm{ef}}+a_{\mathrm{nf}})\times(a_{\mathrm{ef}}+a_{\mathrm{ep}})}} \tag{6}$$

$$\mathrm{Barinel}=1-\frac{a_{\mathrm{ep}}}{a_{\mathrm{ef}}+a_{\mathrm{ep}}} \tag{7}$$

$$\mathrm{OP2}=a_{\mathrm{ef}}-\frac{a_{\mathrm{ep}}}{1+a_{\mathrm{ep}}+a_{\mathrm{np}}} \tag{8}$$

2.3 实例分析

本小节将通过实例来介绍基于冗余覆盖信息约简的软件缺陷定位方法工作原理.图2为Defects4J中Clo-

sure-65版本中CodeGenerator.java文件的程序片段,表1为图2中程序片段的语句覆盖信息矩阵及各语句可疑度值、相关性及排名. 表1中s_1~s_6为图2中行号为1 012~1 017的可执行语句,其中,s_6(即1 017行)为存在缺陷的语句,t_1~t_{10}为覆盖这些语句的测试用例. 表中第2~11行的2~6列为语句被测试用例覆盖的情况,“1”表示语句被测试用例覆盖,“0”表示未被覆盖. ***E***列为测试用例t_1~t_{10}的执行结果,其中“0”时表示测试用例通过,“1”表示测试用例未通过.

表1中SFL-value、FS-value分别为使用DStar技术计算的语句可疑度值和使用Pearson相关系数计算的各语句与测试用例执行结果的相关性值. SFL-Rank、FS-Rank分别为根据SFL-value和FS-value对语句的降序排名. FLRR-Rank(k=1、k=2)分别表示通过特征选择选取FS-Rank排名前1和前2的语句覆盖信息矩阵子集,并使用SFL进行缺陷定位得到的可疑度值排名. 根据SFL-Rank可以得到使用DStar进行缺陷定位时的可疑语句列表为[s_4, s_5, s_1, s_2, s_3, s_6],此时缺陷语句s_6排名第六.

```
1 012    for (int i = 0; i<s.length(); i++) {
1 013        char c = s.charAt(i);
1 014        switch(c){
1 015            case 't': sb.append("\\t"); break;
1 016            case 'n':sb.append("\\n"); break;
1 017            case '()': sb.append("\\()"); break;
1 018        }
1 019    }
```

图2　CodeGenerator.java文件代码片段

若使用FLRR进行缺陷定位,首先根据FS-Rank得到各语句与测试用例执行结果的相关性排名为[s_6, s_3, s_5, s_2, s_4, s_1],并根据该排名对语句覆盖信息矩阵约简,选择相关性排名前k的语句覆盖信息矩阵子集,然后使用SFL对矩阵子集进行缺陷定位. 当k=1和k=2时,进行特征选择分别按相关性降序选择语句覆盖信息矩阵中排名前1(即s_6)和前2(即s_6、s_3)的语句,以得到语句覆盖信息矩阵子集. 在完成特征选择进行缺陷定位时,原可疑度值排名为第六的缺陷语句s_6将分别排名第1(k=1)和第2(k=2). 从上述示例可以看出,使用基于冗余覆盖信息约简的缺陷定位方法可提升缺陷定位的性能.

表1　语句覆盖信息矩阵及各语句可疑度值、相关性及排名

测试用例	s_1	s_2	s_3	s_4	s_5	s_6	***E***
t_1	0	0	0	0	0	0	0
t_2	0	0	0	0	0	0	0
t_3	1	1	1	1	1	1	1
t_4	0	0	0	0	0	0	0
t_5	1	1	1	1	1	0	0
t_6	0	0	0	0	0	0	0
t_7	0	0	0	0	0	0	0
t_8	0	0	0	0	0	0	0
t_9	1	1	1	1	1	0	0
t_{10}	1	1	1	1	1	0	0
SFL-value	0.001 93	0.001 93	0.001 93	0.002 03	0.002 03	0	—
SFL-Rank	3	4	5	1	2	6	—
FS-value	0	0.002 26	0.013 85	0.000 95	0.008 07	0.130 73	—
FS-Rank	6	4	2	5	3	1	—
FLRR-Rank(k=1)	—	—	—	—	—	1	—
FLRR-Rank(k=2)	—	—	1	—	—	2	—

3　实验及评估

实验运行硬件环境为:Intel(R) Xeon(R) E5-2678 v3 @ 2.50 Hz、64 GB物理内存;软件环境为:Windows10、Ubuntu20.04、JDK1.8.0、Python3.7.0.

3.1　实例设计

3.1.1　研究问题

为了评估FLRR方法的有效性,本文提出2个研究问题,针对这2个研究问题在Defects4J数据集上进行实验,并使用$E_{\text{inspect}}@n$和MRR评价指标评估FLRR方法的缺陷定位性能.

RQ1:语句覆盖信息矩阵中是否存在冗余? 冗余情况如何?

通过计算语句覆盖信息与测试用例执行结果的矩阵的Pearson相关系数等两种评价指标,评估Defects4J数据集中语句覆盖信息矩阵是否存在冗余,并分析其

冗余情况.

RQ2:相比于基准技术DStar、Ochiai、Barinel和OP2,FLRR定位软件缺陷的性能如何?

FLRR使用特征选择技术约简语句覆盖信息矩阵后进行缺陷定位,相比基准技术DStar、Ochiai、Barinel和OP2定位软件缺陷的性能如何?

3.1.2 实验对象

实验对象采用被广泛应用于软件缺陷定位研究的数据集Defects4J. 基本信息如表2所示. Defects4J包含来自6个开源项目的395个含有缺陷的版本,分别为Chart(26个版本)、Closure(133个版本)、Lang(65个版本)、Math(106个版本)、Mockito(38个版本)和Time(27个版本). 在表2中,第1列是项目名称,第二列是项目的版本数量,每个版本中包含至少一个缺陷,第3列是每个项目的平均代码行数,程序的每个版本包含一组由开发人员设计的测试用例,第4列是每个项目中所有版本的测试用例总数.

表2 Defects4J数据集(V1.4.0)

项目名称	版本数量	代码行数	测试用例数量
Chart	26	104 k	5 k
Closure	133	2 200 k	446 k
Lang	65	52 k	6 k
Math	106	241 k	18 k
Mockito	38	67 k	25 k
Time	27	139 k	68 k
Sum	395	2 804 k	568 k

3.1.3 评价指标

为评价FLRR的有效性,我们在研究中采用$E_{\text{inspect}}@n$和MRR(Mean Reciprocal Rank)2个被广泛使用的评价指标.

$E_{\text{inspect}}@n$指可疑度值排在前n的语句中存在缺陷的语句数量,n越小且$E_{\text{inspect}}@n$的值越大,缺陷定位方法的性能就越好. Kochhar等人[9]发现,开发人员在逐一检查给定列表中少量语句后,若未发现缺陷便会失去耐心,开始跳跃着检查,甚至放弃检查. 这表明,人们更加关注缺陷定位方法是否能将缺陷语句排在可疑列表中靠前位置. 因此,在本实验中采用了$E_{\text{inspect}}@n(n=1)$作为评价指标.

MRR也是一种通用的对缺陷定位进行评估的指标,它通过查找可疑列表中第一条缺陷语句,并计算其排名的倒数值来评估缺陷定位的性能. 例如,排名第1的可疑语句为缺陷语句则MRR值为1,排名第2的可疑语句为缺陷则MRR值为0.5. MRR值越大,表示缺陷定位技术的性能越好.

3.2 实验结果分析

3.2.1 RQ1结果

为分析Defects4J数据集中语句覆盖信息矩阵的冗余情况,实验使用了Pearson相关系数和相同覆盖信息语句数量两种评价指标来进行度量.

为分析语句覆盖情况与测试用例执行结果相关性,表3统计了各语句与测试用例执行结果的Pearson相关系数取值分布情况. 通常认为,Pearson相关系数的值在0.8~1.0为极强相关,0.6~0.8为强相关,0.4~0.6为中等程度相关,0.2~0.4为弱相关,0.0~0.2为极弱相关或无相关. Pearson相关系数为两组数据之间的协方差和标准差乘积的比值. 在计算Pearson相关系数时,若一组数据标准差为0,即一组数据相同时,会使Pearson相关系数公式的分母为0,导致计算出错. 例如,在执行测试用例时,全部测试用例均未覆盖或均覆盖某一条语句,则该语句的覆盖信息全部为0或1,在这种情况下,将无法计算该语句与测试用例执行结果间的Pearson相关系数. 实验中使用开源的Python算法库Scipy计算Pearson相关系数,对于上述无法计算Pearson相关系数的情况,将返回错误值NaN(Not a Number).

表3 Defects4J数据集语句覆盖信息与测试用例执行结果的Pearson相关系数分布统计

项目	Pearson相关系数				
	0.0~0.2	0.2~0.4	0.4~0.6	0.6~0.8	0.8~1.0
Chart	44 860	4 796	1 172	1 375	3 254
Time	1 510 362	27 728	6 663	2 936	2 842
Closure	26 159	1 140	947	765	485
Math	61 640	9 312	2 349	1 635	1 362
Lang	49 429	1 472	443	376	360
Mockito	101 488	1 135	387	541	50
Sum	1 793 938	45 583	11 961	7 628	8 353

表3中列出了各个项目中所有语句与测试用例执行结果间Pearson相关系数的分布情况,其中未统计Pearson相关系数为NaN的语句. 从表3中可以看出,与测试用例执行结果极弱相关或无相关(区间为0.0~0.2)的语句数为1 793 938,弱相关(区间为0.2~0.4)的语句数为45 583,中等相关(区间为0.4~0.6)的语句数为11 961,强相关(区间为0.6~0.8)的语句数为7 628,极强相关(区间为0.8~1.0)的语句数仅为8 353. 其中极弱相关或无相关的语句占总语句的96.06%,这表明语句覆盖信息矩阵中绝大多数语句与测试用例执行结果相关性很弱.

为了进一步分析Defects4J数据集中语句覆盖信息矩阵的冗余情况,对其中具有相同覆盖信息的语句数量进行分析. 当不同语句被测试用例集中完全一致的测试用

例执行时,则认为这些语句具有相同的覆盖信息. 表4中列出了各项目中具有相同覆盖信息的语句数量,其中第二列为具有与其他语句覆盖信息均不相同的语句数量;第三列为与2~5条语句具有相同覆盖信息的语句数量;第四、五列为与6~10条和超过10条语句具有相同覆盖信息的语句数量;第六列为一个项目中具有相同覆盖信息最多的语句数量. 例如,Closure项目中具有与其他语句不相同覆盖信息的语句数为159 214条,与2~5条语句具有相同覆盖信息的语句数为349 472条,与6~10条语句具有相同覆盖信息的语句数为224 485条,与超过10条语句具有相同覆盖信息的语句数为1 466 460条. 其中,Closure-107版本中有16 910条语句的覆盖信息相同,占该程序总语句数的48.35%. 此外,统计还发现在Defects4J数据集的所有语句覆盖信息矩阵中,共有188 693条语句与其他语句的覆盖信息均不相同,这些语句仅占总语句数的6.73%,这表明程序中大多数语句都存在与其覆盖信息相同的其他语句.

对RQ1的回答 统计结果表明语句覆盖信息矩阵中存在严重的数据冗余现象. 语句覆盖信息与测试用例执行结果的Pearson相关系数分布情况显示,超过96%的语句与测试用例执行结果呈无关或极弱相关. 此外,超过90%的语句与其他语句具有相同的覆盖信息.

表4 Defects4J数据集中语句覆盖信息相同的语句数量统计

项目	具有相同覆盖信息的语句数量				具有相同覆盖信息的最大语句数量
	1	2~5	6~10	大于10	
Chart	1 143	5 680	4 888	93 027	12 289
Time	14 390	37 802	52 286	34 578	3 794
Closure	159 214	349 472	224 485	1 466 460	16 910
Math	3 327	9 075	7 368	221 653	5 998
Lang	2 205	7 079	7 255	35 750	1 690
Mockito	8 414	18 525	8 542	32 072	2 701
Sum	188 693	427 633	304 824	1 883 540	16 910

3.2.2 RQ2结果

为验证FLRR的缺陷定位性能,并分析不同特征选择技术对缺陷定位性能的影响,选用了六种特征选择技术和四种缺陷定位技术进行实验. 实验先通过特征选择,分别选取Defects4J中6个项目的语句覆盖信息矩阵中相关性排名前1至前200的语句,构成语句覆盖信息矩阵子集,再用于缺陷定位. 然后与直接使用基准技术进行缺陷定位的效果就$E_{\text{inspect}}@n(n=1)$和MRR指标进行对比.

图3和图4中,我们用"特征选择技术名-缺陷定位技术名"来表示将不同特征选择技术应用于不同SFL技术的FLRR方法,例如,P-DStar表示使用Pearson进行特征选择,然后使用DStar进行缺陷定位的实验结果RF-Barinel表示使用RF进行特征选择,然后使用Barinel进行缺陷定位的实验结果. 而DStar、Ochiai、Barinel和OP2则表示直接使用四种基准技术进行缺陷定位的实验结果,由于k值不会对上述基线技术产生影响,因此它们的$E_{\text{inspect}}@n(n=1)$和MRR指标值在图3和图4中均为一条水平横线. 此外,在图3和图4中将FLRR方法定位效果最好时k的取值和对应$E_{\text{inspect}}@n(n=1)$、MRR指标的数值进行了标注. 例如:图3($a$)中的(12,56)表示RF-DStar的$E_{\text{inspect}}@n(n=1)$值在$k$=12时最高,为56,图4($a$)中的(113,0.227 3)表示,MI-DStar的MRR值在k=113时最高为0.227 3.

图3为FLRR与直接使用DStar、Ochiai、Barinel和OP2进行缺陷定位时,$E_{\text{inspect}}@n(n=1)$指标的对比情况. 其中横坐标表示使用特征选择技术得到的语句覆盖信息矩阵子集中语句的数量,k值表示选择相关性排名为前k的语句,并由k条语句组成语句覆盖信息矩阵子集. 纵坐标表示检查可疑度值排名第1语句时,可成功定位到的缺陷数量. 从图3中可以看出,在使用$E_{\text{inspect}}@n(n=1)$指标时,使用RF、MI和DT进行特征选择均可以提高DStar、Ochiai、Barinel和OP2的缺陷定位性能,而使用Pearson、REF、REFCV进行特征选择则不能提高以上基准技术的缺陷定位性能.

直接使用DStar进行缺陷定位时,检查可疑列表排名第1的语句可以定位到33条缺陷语句. RF-DStar和MI-DStar在k值为1至200范围内所能定位到缺陷数量均高于DStar. 其中,RF-DStar在k=12时定位效果最好,检查可疑列表排名第1的语句可定位到56条缺陷语句,MI-DStar在k=137时定位效果最好,检查可疑列表排名第1的语句可以定位到47条缺陷语句. DT-DStar在k值为30至200范围内所能定位到缺陷数量均高于DStar,且在k=133时定位效果最好,此时检查可疑列表排名第1的语句可以定位到37条缺陷语句.

直接使用Ochiai进行缺陷定位时,检查可疑列表排名第1的语句可以定位到34条缺陷语句. 使用RF-Ochiai时,在k值为1至200范围内所能定位到缺陷的数量均高于Ochiai,且在k=147时定位效果最好,检查可疑列表排名第1的语句可以定位到60条缺陷语句.

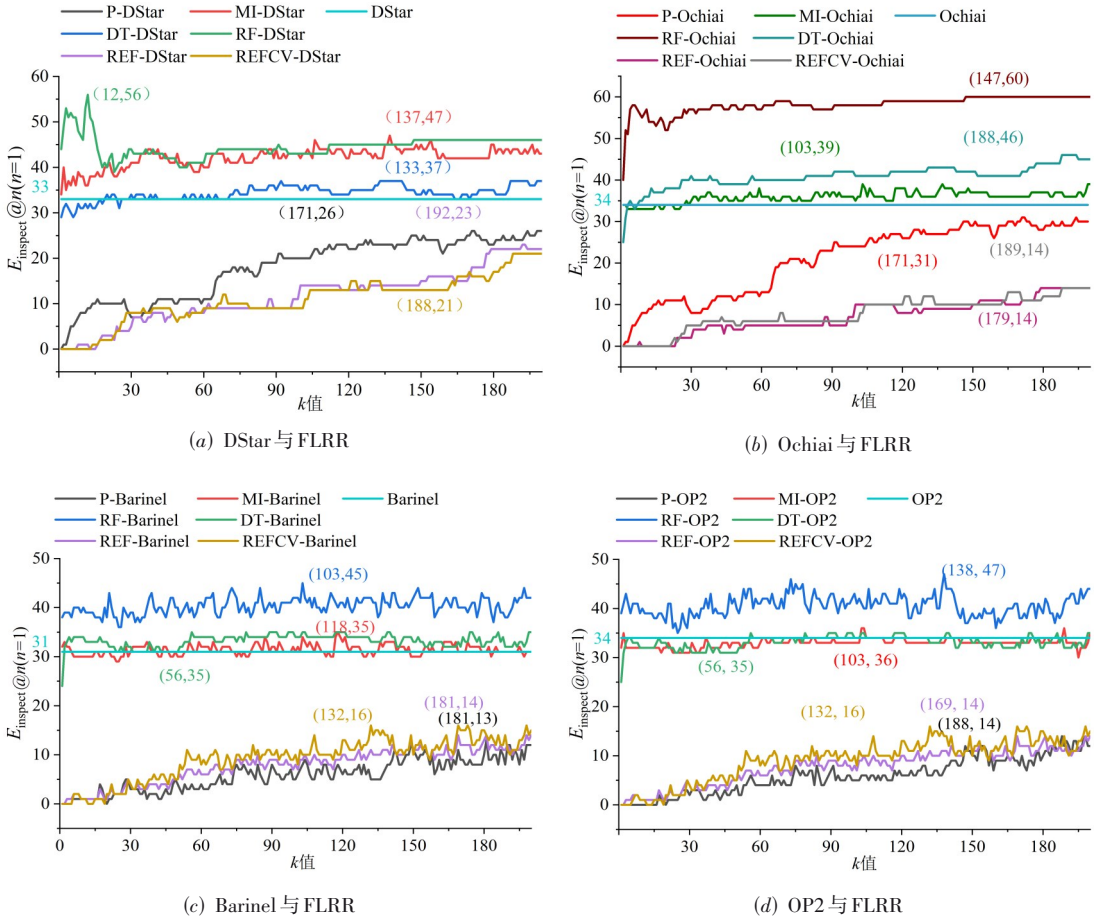


图3 FLRR与DStar、Ochiai、Barinel和OP2缺陷定位性能对比情况($E_{inspect}@n(n=1)$)

MI-Ochiai在k值为26至200范围内所能定位到缺陷的数量均高于Ochiai,且在k=103时定位效果最好,检查可疑列表排名第1的语句可以定位到39条缺陷语句. DT-Ochiai在k值为7至200范围内所能定位到缺陷的数量均高于Ochiai,且在k=188时定位效果最好,通过检查可疑列表排名第1的语句可以定位到46条缺陷语句.

直接使用Barinel进行缺陷定位时,检查可疑列表排名第1的语句可以定位到31条缺陷语句. RF-Barinel在k值为1至200范围内所能定位到缺陷的数量均高于Barinel,且在k=103时定位效果最好,检查可疑列表排名第1的语句可以定位到45条缺陷语句. MI-Barinel在k=118时定位效果最好,检查可疑列表排名第1的语句可以定位到35条缺陷语句. DT-Ochiai在k值为2至200范围内所能定位到缺陷的数量均高于Barinel,且在k=56时定位效果最好,通过检查可疑列表排名第1的语句可以定位到35条缺陷语句.

直接使用OP2进行缺陷定位时,检查可疑列表排名第1的语句可以定位到34条缺陷语句. 使用RF-OP2时,在k值为1至200范围内所能定位到缺陷的数量均高于OP2,且在k=138时定位效果最好,检查可疑列表排名第1的语句可以定位到47条缺陷语句. MI-OP2在k=103时定位效果最好,检查可疑列表排名第1的语句可以定位到36条缺陷语句. DT-OP2在k=56时定位效果最好,通过检查可疑列表排名第1的语句可以定位到35条缺陷语句.

从以上分析可以看出,使用RF进行特征选择效果最好,对于$E_{inspect}@n(n=1)$指标,RF-DStar、RF-Ochiai、RF-Barinel和RF-OP2分别比DStar、Ochiai、Barinel和OP2最多可多定位到23、26、14和13条缺陷语句,增加了69.70%、76.47%、45.16%和38.24%.

图4为FLRR与直接使用DStar、Ochiai、Barinel和OP2进行缺陷定位时的MRR指标对比情况. 其中横坐标与图3的含义相同,纵坐标表示不同缺陷定位技术的MRR值,即在语句可疑度列表中,第1条缺陷语句排名的倒数. 图4与图3中的结果相似,使用RF、MI和DT进行特征选择均可以提高DStar、Ochiai、Barinel和OP2的缺陷定位性能,而使用Pearson、REF、REFCV进行特征选择则不能提高以上基准技术的缺陷定位性能.

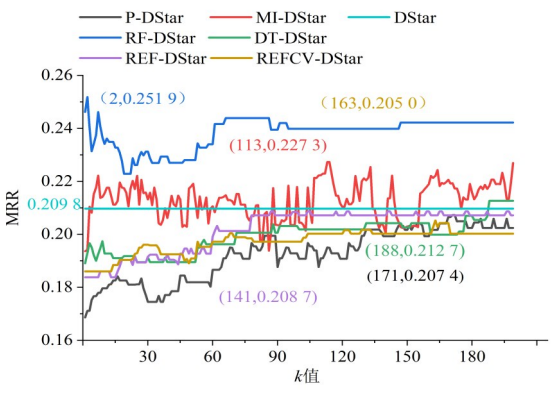
直接使用DStar时,MRR值为0.209 8. RF-DStar在k

值为 1 至 200 时,MRR 值均高于 DStar,其中,RF-DStar 在 k=2 时定位效果最好,MRR 值为 0.251 9. MI-DStar 在 k 值为 1 至 200 范围内 140 个语句覆盖信息矩阵子集定位结果的 MRR 值高于 DStar,且在 k=113 时定位效果最好,MRR 值为 0.227 3. DT-DStar 在 k 值为 188 至 200 范围内的 MRR 值均高于 DStar,且在 k=188 时定位效果最好,MRR 值为 0.212 7.

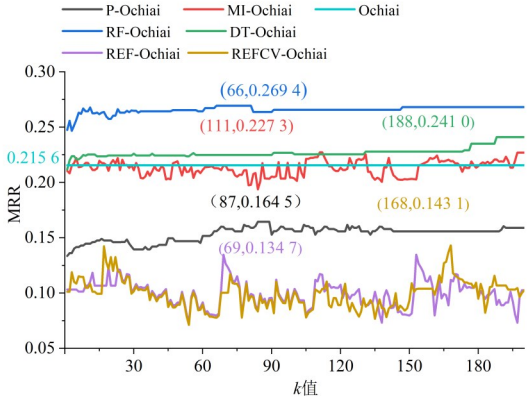
直接使用 Ochiai 时,MRR 值为 0.215 63. RF-Ochiai 在 k 值为 1 至 200 范围内 MRR 值均高于 Ochiai,并且在 k=66 时定位效果最好,MRR 值为 0.269 4. MI-Ochiai 在 k 值为 1 至 200 范围内 77 个语句覆盖信息矩阵子集定位结果的 MRR 值高于 Ochiai,且在 k=111 时定位效果最好,MRR 值为 0.227 3. DT-Ochiai 在 k 值为 2 至 200 范围内 MRR 的值均高于 Ochiai,且在 k=188 时定位效果最好,MRR 值为 0.241 0.

直接使用 Barinel 时,MRR 值为 0.205 9. RF-Barinel 在 k 值为 1 至 200 范围内 MRR 值均高于 Barinel,并且在 k=103 时定位效果最好,MRR 值为 0.241 9. MI-Barinel 在 k 值为 1 至 200 范围内 161 个语句覆盖信息矩阵子集定位结果的 MRR 值高于 Barinel,且在 k=118 时定位效果最好,MRR 值为 0.234 3. DT-Barinel 在 k 值为 3 至 200 范围内 MRR 的值均高于 Barinel,且在 k=56 时定位效果最好,MRR 值为 0.228 2.

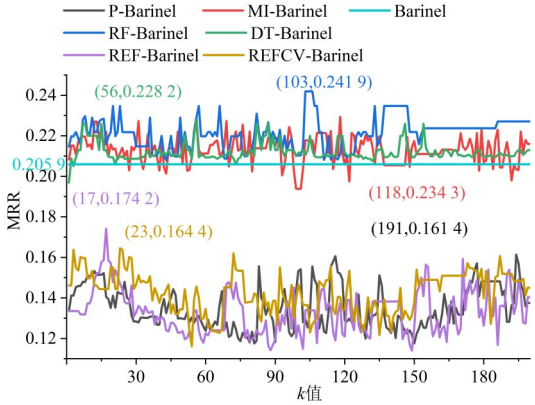
当直接使用 OP2 时,MRR 值为 0.202 3. RF-OP2 在 k 值为 1 至 200 范围内的 199 个 k 值的 MRR 均高于 OP2,并且在 k=64 时定位效果最好,MRR 值为 0.241 1. MI-OP2 在 k 值为 1 至 200 范围内 111 个语句覆盖信息矩阵子集定位结果的 MRR 值高于 OP2,且在 k=69 时定位效果最好,MRR 值为 0.224 7. DT-OP2 在 k 值为 1 至 200 范围内 75 个语句覆盖信息矩阵子集定位结果的 MRR 值高于 OP2,且在 k=77 时定位效果最好,MRR 值为 0.214 9.



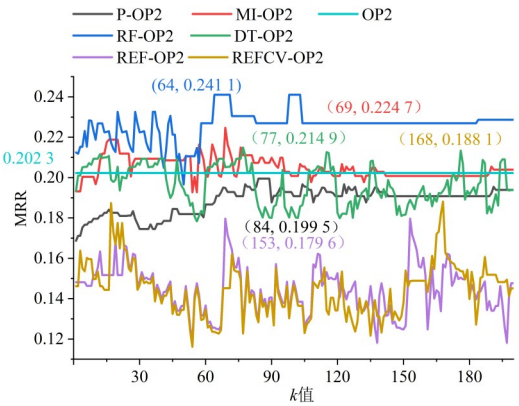
(a) DStar 与 FLRR



(b) Ochiai 与 FLRR



(c) Barinel 与 FLRR



(d) OP2 与 FLRR

图 4　FLRR 与 DStar、Ochiai、Barinel 和 OP2 缺陷定位性能对比情况(MRR)

从以上分析可以看出,使用 RF 进行特征选择效果最好,对于 MRR 指标,RF-DStar、RF-Ochiai、RF-Barinel 和 RF-OP2 分别比 DStar、Ochiai、Barinel 和 OP2 最多提升 20.08%、24.94%、17.45% 和 19.15%.

对RQ2的回答 实验结果表明,两种嵌入式特征选择技术(RF和DT)和一种过滤式特征选择技术(MI)提升了所有基线SFL技术的性能,而另外两种包裹式(REF和REFCV)特征选择技术和一种过滤式(Pearson)特征选择技术则没有提升基线SFL技术的性能. 其中,使用RF进行特征选择效果最好,对于$E_{\text{inspect}}@n(n=1)$指标,RF-DStar、RF-Ochiai、RF-Barinel和RF-OP2分别比DStar、Ochiai、Barinel和OP2最多可多定位到23、26、14和13条缺陷语句;对于MRR指标,RF-DStar、RF-Ochiai、RF-Barinel和RF-OP2分别比DStar、Ochiai、Barinel和OP2最多提升了20.08%、24.94%、17.45%和19.15%.

3.3 有效性分析

文本节从内部有效性、外部有效性和构造有效性3个方面进行有效性分析.

(1)内部有效性

内部有效性主要体现在方法实施和实验过程的正确性上. FLRR中主要包括特征选择和基于频谱的缺陷定位两个步骤. 为减少内部有效性威胁,实验采用Scikit-learn和Scipy库中的特征选择算法进行特征选择,实现了常用的缺陷定位技术DStar和Ochiai,并对所编写的代码进行了多次内部审查. 此外,还将实验中DStar和Ochiai的定位结果与Pearson等人[22]的相关实验结果进行了对比验证,以确保定位结果的正确性.

(2)外部有效性

外部有效性主要与实验对象的代表性有关. 本文使用Defecst4J作为实验对象,它包括来自6个开源项目的395个真实缺陷. 该数据集被广泛用于评估缺陷定位方法的性能[23,24]. Defects4J数据集虽然具有一定的代表性,但无法保证FLRR在其他数据集上能得到相同的结果. 此外,我们仅在实验中使用了DStar和Ochiai等四种定位效果最好的SFL技术和Pearson、MI、RF等六种广泛使用的特征选择技术,上述技术虽然具有一定的代表性,但不能保证使用其他SFL和特征选择技术进行软件缺陷定位时具有相同效果.

(3)构造有效性

构造有效性主要与实验中使用的评价指标有关. 本文使用$E_{\text{inspect}}@n$和MRR评估FLRR的性能,这是两种常用的软件缺陷定位性能评价指标[22~24]. 其中,$E_{\text{inspect}}@n$用于统计缺陷定位的绝对排名,因为开发人员在检查给定的可疑语句列表时,更关注排名靠前,特别是排名第1的语句,所以实验中重点关注了$E_{\text{inspect}}@n(n=1)$;而MRR则用于统计缺陷定位的相对排名.

4 相关工作

本节主要介绍基于频谱的缺陷定位技术和特征选择技术的相关应用情况.

4.1 基于频谱的软件缺陷定位技术

国内外研究人员针对软件缺陷定位已经进行了大量研究,常用技术包括基于频谱的缺陷定位技术[1,2]、基于变异的缺陷定位技术[3,4]及其他缺陷定位技术[5,6]等. 本文将重点关注基于频谱的缺陷定位技术.

Jones等人[25]最早提出了基于频谱的缺陷定位技术Tarantula,使用覆盖率和测试用例执行结果(通过或未通过)来度量每条语句的可疑度值. 随后,通过改进可疑度公式,研究者相继提出了一些效果更佳的SFL技术. 例如Abreu等人[15]受聚类分析中相似度系数的启发,提出了Ochiai,提高了软件缺陷定位性能;Wong等人[7]提出了DStar,通过改变可疑度公式中参数的权重来提高软件缺陷定位性能.

还有一些工作尝试将更多的信息与程序频谱进行结合,以提升软件缺陷定位的性能. 例如,姜淑娟等人[26]提出了一种基于路径分析和信息熵的缺陷定位方法FLPI,通过对程序中所有执行路径的数据依赖关系进行分析,得到程序执行路径的上下文信息,同时利用信息熵理论将测试事件信息引入可疑度公式中,实验表明FLPI相比其他缺陷定位方法只需检查更少的代码就可以定位到更多缺陷. Wen等人[24]提出了一种基于历史谱的缺陷定位技术HSFL,该技术先从版本历史记录中识别导致缺陷的提交,再基于导致缺陷的提交构建历史谱(Histrum),并通过结合历史谱和常规频谱进行缺陷语句定位. 实验结果表明,基于历史谱的缺陷定位技术明显优于Tarantula、Ochiai和DStar等5种SFL技术.

语句覆盖信息矩阵的二进制状态信息仅表明该语句是否被执行,无法体现该语句在程序中的重要程度,这可能会导致缺陷定位的有效性降低. 为此,张卓等人[27]提出了基于词频-逆文件频率的缺陷定位技术,采用词频-逆文件频率(Term Frequency - Inverse Document Frequency,TF-IDF)量化每个测试用例中各语句的影响程度,以构建出信息表达能力更强的TF-IDF矩阵,并根据构建的TF-IDF矩阵使用DStar、Ochiai等SFL技术来计算语句的可疑度值. 实验结果表明,基于TF-IDF的技术通过量化矩阵中每个测试用例及各语句的权重,大幅提升了DStar、Ochiai等SFL技术的性能.

针对现有的SFL技术仅将原始数据作为输入,很少考虑数据的类不平衡问题而导致性能较差的问题,Xie等人[28]提出了一种通用的数据增强方法Aeneas,首先应用修正的主成分分析(Principal Component Analysis,PCA)来缩减特征空间,以更简洁地表示原始覆盖矩阵,然后通过条件变分自动编码器(Conditional Variational AutoEncoder,CVAE)从缩减后的特征空间中合成失败测试用例来处理数据的类不平衡问题,以提升缺陷定

位效果.

上述工作取得了较好的软件缺陷定位结果,但未考虑解决语句覆盖信息矩阵中的数据冗余问题,会明显影响缺陷定位的精度和性能. 为此,本文提出的FLRR方法通过对SFL的语句覆盖信息矩阵进行特征选择,降低矩阵中数据的冗余度,以提高缺陷定位性能.

4.2 特征选择技术及应用

特征选择是常用的数据预处理技术,通过去除冗余和不相关特征来获取原始特征集中最有价值的特征子集,以达到减少计算开销,防止过拟合的效果. 特征选择技术在自然语言处理、图像检索和入侵检测等领域中取得了大量成功应用[29~32].

在自然语言处理领域,何径舟等人[29]使用特征选择技术分析了汉语词义消岐中特征模板对消歧结果的影响,并提出了一套基于最大熵分类模型的自动特征选择技术. 实验结果表明,特征选择技术不仅简化了特征模板,而且提高了汉语词义消歧的性能. 在图像检索领域,Jiang等人[12]提出了CBIR(Content-Based Image Retrieval)技术,通过使用在线特征选择算法去除冗余特征,选取具有代表性的特征子集来训练模型,以提高图像检索的精度. 大量实验表明,与MDA(Multiple Discriminant Analysis)等典型的图像检索技术相比,CBIR有效提高了图像检索精度并节省了处理时间. 在入侵检测领域,陈友等人[30]尝试使用信息增益等多种特征选择技术,通过消除冗余特征和噪音,提高了入侵检测系统的检测速度和效果. 唐成华等人[31]提出了基于特征选择的聚类异常入侵检测模型,利用信息增益算法对网络攻击中的特征进行排序,以得到低维特征集合,并构建异常入侵检测模型,所构建的模型对绝大多数类型的网络攻击均具有较好的检测能力. FLRR与上述工作相同之处在于均使用了特征选择技术来减小特征空间,降低数据冗余度. 不同在于应用领域不同,FLRR将特征选择技术应用于提高软件缺陷定位的性能,通过降低语句覆盖信息的数据冗余来提高软件缺陷定位性能.

在软件缺陷定位领域,Lei等人[32]提出了Feature-FL,该方法首先根据测试用例的执行轨迹计算语句执行概率,来替代频谱中的二进制覆盖信息,以更加精确地描述每条语句在程序执行过程中的行为;然后使用特征选择技术来量化每条语句与测试用例执行结果之间的相关性,把相关性作为语句存在缺陷的可疑度,以进行缺陷定位. 与DStar、Ochiai等效果最好的SFL技术进行对比的实验结果表明,Feature-FL具有更好的缺陷定位性能. 与Feature-FL相比,本文首次分析了软件缺陷定位方法中使用的覆盖信息存在的严重冗余现象,将其建模为特征选择问题,提出一种基于冗余覆盖信息约简的软件缺陷定位方法,通过降低覆盖信息的冗余度来提升缺陷定位效果,并较为完整地分析了过滤式、嵌入式、包裹式三类不同的特征选择技术对SFL性能的影响. 而Feature-FL则是通过语句执行概率来更加精确地描述语句覆盖信息,并不会降低覆盖信息的冗余度. 此外,可将本文方法对Feature-FL中由语句执行概率构成的覆盖信息进行约简,减少冗余信息,降低无缺陷语句带来的干扰,从而进一步提升Feature-FL的缺陷定位性能.

5 结论

本文分析了软件缺陷定位方法中使用的覆盖信息存在严重冗余的现象,并提出了一种基于冗余覆盖信息约简的软件缺陷定位方法. 该方法将SFL中的覆盖信息约简建模为特征选择问题,通过特征选择技术识别出语句覆盖信息矩阵中各语句对测试用例执行结果影响程度的大小,得到语句覆盖信息矩阵子集,再计算其中剩余语句的可疑度值,以得到可疑语句列表. 在Defects4J数据集上进行的实验结果表明,特征选择技术通过降低语句覆盖信息矩阵的冗余度可有效提升缺陷定位技术的精度. 对于$E_{\text{inspect}}@n(n=1)$和MRR指标,FLRR比直接使用DStar和Ochiai均有较大提升.

在下一步工作中,我们计划尝试将FLRR应用于MFL等其他类型的软件缺陷定位技术,以提升其缺陷定位性能,并将深入分析FLRR用于软件多缺陷定位的效果. 此外,我们还将尝试在更多大规模的数据集上使用更多特征选择技术和缺陷定位技术来进一步验证FLRR的有效性.

参考文献

[1] WONG W E, GAO R Z, LI Y H, et al. A survey on software fault localization[J]. IEEE Transactions on Software Engineering, 2016, 42(8): 707-740.

[2] 陈翔, 鞠小林, 万志, 等. 基于程序频谱的动态缺陷定位方法研究[J]. 软件学报, 2015, 26(2): 390-412.
CHEN X, JU X L, WAN Z, et al. Review of dynamic fault localization approaches based on program spectrum[J]. Journal of Software, 2015, 26(2): 390-412. (in Chinese)

[3] PAPADAKIS M, LE TRAON Y. Metallaxis-FL: Mutation-based fault localization[J]. Software Testing, Verification and Reliability, 2015, 25(5/6/7): 605-628.

[4] HE T, WANG X M, ZHOU X C, et al. A software fault localization technique based on program mutations[J]. Chinese Journal of Computers, 2014, 36(11): 2236-2244.

[5] 曹鹤玲, 姜淑娟. 基于Chameleon聚类分析的多错误定位

方法[J]. 电子学报, 2017, 45(2): 394-400.
CAO H L, JIANG S J. Multiple-fault localization based on chameleon clustering[J]. Acta Electronica Sinica, 2017, 45(2): 394-400. (in Chinese)

[6] 王建峰, 魏长安, 盛云龙, 等. 基于错误交互集的组合测试软件故障定位方法[J]. 电子学报, 2014, 42(6): 1173-1178.
WANG J F, WEI C A, SHENG Y L, et al. Locating errors in combinatorial testing using set of possible faulty interactions[J]. Acta Electronica Sinica, 2014, 42(6): 1173-1178. (in Chinese)

[7] WONG W E, DEBROY V, GAO R Z, et al. The DStar method for effective software fault localization[J]. IEEE Transactions on Reliability, 2014, 63(1): 290-308.

[8] JUST R, JALALI D, ERNST M D. Defects4J: A database of existing faults to enable controlled testing studies for Java programs[C]//Proceedings of the 2014 International Symposium on Software Testing and Analysis. New York: ACM, 2014: 437-440.

[9] KOCHHAR P S, XIA X, LO D, et al. Practitioners' expectations on automated fault localization[C]//Proceedings of the 25th International Symposium on Software Testing and Analysis. New York: ACM, 2016: 165-176.

[10] JAIN A, ZONGKER D. Feature selection: Evaluation, application, and small sample performance[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 1997, 19(2): 153-158.

[11] CHANDRASHEKAR G, SAHIN F. A survey on feature selection methods[J]. Computers & Electrical Engineering, 2014, 40(1): 16-28.

[12] ZHOU Y Y, CHENG G, JIANG S Q, et al. Building an efficient intrusion detection system based on feature selection and ensemble classifier[J]. Computer Networks, 2020, 174: 107247.

[13] BREIMAN L. Random forests[J]. Machine Learning, 2001, 45(1): 5-32.

[14] JIANG W, ER G H, DAI Q H, et al. Similarity-based online feature selection in content-based image retrieval[J]. IEEE Transactions on Image Processing, 2006, 15(3): 702-712.

[15] ABREU R, ZOETEWEIJ P, GOLSTEIJN R, et al. A practical evaluation of spectrum-based fault localization[J]. Journal of Systems and Software, 2009, 82(11): 1780-1792.

[16] ABREU R, ZOETEWEIJ P, VAN GEMUND A J C. Spectrum-based multiple fault localization[C]//2009 IEEE/ACM International Conference on Automated Software Engineering. Piscataway: IEEE, 2009: 88-99.

[17] NAISH L, LEE H J, RAMAMOHANARAO K. A model for spectra-based software diagnosis[J]. ACM Transactions on Software Engineering and Methodology, 2011, 20(3): 1-32.

[18] BENESTY J, CHEN J D, HUANG Y T, et al. Pearson correlation coefficient[M]//Noise Reduction in Speech Processing. Berlin: Springer, 2009: 1-4.

[19] FRASER A M, SWINNEY H L. Independent coordinates for strange attractors from mutual information[J]. Physical Review A, 1986, 33(2): 1134-1140.

[20] LOH W Y. Classification and regression trees[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2011, 1(1): 14-23.

[21] ROTHERMEL G, HARROLD M J, OSTRIN J, et al. An empirical study of the effects of minimization on the fault detection capabilities of test suites[C]//Proceedings of the International Conference on Software Maintenance. Bethesda: IEEE, 1998: 34-43.

[22] PEARSON S, CAMPOS J, JUST R, et al. Evaluating and improving fault localization[C]//2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE). Piscataway: IEEE, 2017: 609-620.

[23] ZOU D M, LIANG J J, XIONG Y F, et al. An empirical study of fault localization families and their combinations [J]. IEEE Transactions on Software Engineering, 2021, 47(2): 332-347.

[24] WEN M, CHEN J J, TIAN Y Q, et al. Historical spectrum based fault localization[J]. IEEE Transactions on Software Engineering, 2021, 47(11): 2348-2368.

[25] JONES J A, HARROLD M J, STASKO J. Visualization of test information to assist fault localization[C]//Proceedings of the 24th International Conference on Software Engineering. New York: ACM, 2002: 467-477.

[26] 姜淑娟, 张旭, 王荣存, 等. 基于路径分析和信息熵的错误定位方法[J]. 软件学报, 2021, 32(7): 2166-2182.
JIANG S J, ZHANG X, WANG R C, et al. Fault localization approach based on path analysis and information entropy[J]. Journal of Software, 2021, 32(7): 2166-2182. (in Chinese)

[27] 张卓, 雷晏, 毛晓光, 等. 基于词频-逆文件频率的错误定位方法[J]. 软件学报, 2020, 31(11): 3448-3460.
ZHANG Z, LEI Y, MAO X G, et al. Fault localization approach using term frequency and inverse document frequency[J]. Journal of Software, 2020, 31(11): 3448-3460.

(in Chinese)

[28] XIE H, LEI Y, YAN M, et al. A universal data augmentation approach for fault localization[C]//Proceedings of the 44th International Conference on Software Engineering. New York: ACM, 2022: 48-60.

[29] 何径舟, 王厚峰. 基于特征选择和最大熵模型的汉语词义消歧[J]. 软件学报, 2010, 21(6): 1287-1295.
HE J Z, WANG H F. Chinese word sense disambiguation based on maximum entropy model with feature selection [J]. Journal of Software, 2010, 21(6): 1287-1295. (in Chinese)

[30] 陈友, 程学旗, 李洋, 等. 基于特征选择的轻量级入侵检测系统[J]. 软件学报, 2007, 18(7): 1639-1651.
CHEN Y, CHENG X Q, LI Y, et al. Lightweight intrusion detection system based on feature selection[J]. Journal of Software, 2007, 18(7): 1639-1651. (in Chinese).

[31] 唐成华, 刘鹏程, 汤申生, 等. 基于特征选择的模糊聚类异常入侵行为检测[J]. 计算机研究与发展, 2015, 52(3): 718-728.
TANG C H, LIU P C, TANG S S, et al. Anomaly intrusion behavior detection based on fuzzy clustering and features selection[J]. Journal of Computer Research and Development, 2015, 52(3): 718-728. (in Chinese)

[32] LEI Y, XIE H, ZHANG T, et al. Feature-FL: Feature-based fault localization[J]. IEEE Transactions on Reliability, 2022, 71(1): 264-283. .

作者简介

王浩仁　男,1997年7月出生,河北石家庄人. 现为北京信息科技大学硕士研究生. 主要研究方向为软件缺陷定位.
E-mail: wanghaoren@bistu.edu.cn

崔展齐　男,1984年2月出生,贵州金沙人. 现为北京信息科技大学教授,硕士生导师,博士. 主要研究方向为软件分析与软件测试技术.
E-mail: czq@bistu.edu.cn

岳　雷　男,1998年10月出生,甘肃兰州人. 现为北京信息科技大学硕士研究生. 主要研究方向为软件缺陷定位.
E-mail: ruoxi41@bistu.edu.cn

陈　翔　男,1980年3月出生,江苏南通人. 现为南通大学副教授,硕士生导师,博士. 主要研究领域为软件缺陷预测、软件缺陷定位、回归测试和组合测试.
E-mail: xchencs@ntu.edu.cn

郑丽伟　男,1979年7月出生,山西忻州人. 现为北京信息科技大学副教授,硕士生导师,博士. 主要研究领域为需求软件工程、社交网络和数据质量增强.
E-mail: zlw@bistu.edu.cn