计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210200096

基于深度卷积网络的多错误定位方法

张 慧
江苏科技大学计算机学院 江苏 镇江 212003

摘 要 目前的错误定位方法大多数解决的是单错误定位。然而,错误之间是相互关联的,如何找到这些语句与测试结果之间的关联关系和错误之间的关联关系,并减轻偶然性正确的测试用例和相似测试用例对语句可疑度的影响,对提高多错误定位的效率至关重要。为了解决以上问题,提出了基于深度卷积网络的多错误定位方法,通过一种特殊结构的深度卷积网络得到一组准确度比较高的语句可疑度,再将其应用于前向切片和后向切片中,寻找到错误与错误之间的关联定位多错误。实验表明,所提方法的多错误定位效率高于目前存在的经典的错误定位方法的错误定位效率。

关键词:错误定位;深度卷积网络;切片
中图法分类号 TP311.5

Multiple Fault Localization Method Based on Deep Convolutional Network

ZHANG Hui
School of Computer Science,Jiangsu University of Science and Technology,Zhenjiang,Jiangsu 212003,China

Abstract Most of the current fault localization methods solve single fault localization,but the faults are related to each other. How to find the relationship between these faults and the test results and the relationship between the faults,and reduce the impact of coincidental correct test cases and similar test cases on the suspiciousness of sentences is very important to improve the efficiency of multiple fault localization. In order to solve the above problems,this paper proposes a multiple fault localization method based on deep convolutional network. A set of suspiciousness with high accuracy is obtained through a deep convolutional network with a special structure,and then applied to forward slicing and backward slicing,the correlation between faults and faults is found to locate multiple faults. Experiments show that the multiple fault localization efficiency of the method in this paper is stronger than that of the existing classic fault localization methods.

Keywords Fault localization,Deep convolutional network,Slicing

1 前言

在软件工程中,开发好的程序根据测试用例进行测试,得到测试结果,然后开发人员根据测试用例和缺陷报告定位错误的位置并且修改(即软件调试),测试人员根据修改好的程序和回归测试用例进行回归测试发现未解决的错误,开发人员再根据回归测试用例和缺陷报告继续定位和修改错误,一直到解决所有的错误。软件测试是一个迭代的过程,所以如何通过回归测试用例定位到错误是软件工程中的难中之难。

目前经典的错误定位方法主要有基于程序切片方法[1-5]、基于依赖方法[6-7]、基于状态变更方法[8-10]和基于覆盖分析方法[11-16]等。其中,基于程序切片、基于依赖方法和基于状态变更方法主要考虑的是控制依赖和数据依赖,这些方法虽然考虑了程序之间的依赖关系,但是计算复杂度高,影响了错误定位的效率。基于覆盖分析方法主要通过特定的公式计算成功的测试用例和失败的测试用例数目,组合得到语句可疑度,进而定位错误,虽然计算复杂度低,但是由于没有考虑程序之间的依赖关系和偶然性正确的测试用例(即执行了错误的语句但是测试结果为成功的测试用例)的存在等,错误定位的效率受到影响,并且错误越多错误定位效率越低。

为了解决以上问题,本文提出了基于深度卷积网络的多错误定位方法。该方法利用一种特殊结构的深度卷积网络得到语句与测试结果之间的关系,减轻了偶然性正确的测试用例和相似测试用例对错误定位效率的影响,进而得到一组准确度比较高的语句可疑度,再将其用于前向切片和后向切片进行多错误定位。

本文第2节介绍相关工作;第3节介绍基于深度卷积网络的多错误定位方法;第4节为实验描述和结果分析;最后总结全文。

2 相关工作

在多错误定位方法方面,Wen等[18]通过建立与错误相关的条件执行切片,然后依次计算条件执行切片中的语句可疑度,进行多错误定位。Cao等[19]通过一个失败的测试用例与所有成功的测试用例合并计算可疑度,从中选取高可疑度语句为特征元素,再利用特征元素对失败的测试用例进行约简,通过聚类分析将失败的测试用例进行分簇,每簇包含一个错误,再将失败的测试用例簇与所有成功的测试用例进行合并,根据合并后的簇生成的可疑度序列定位多错误。Wang等[20]通过模糊C均值聚类分析每个失败的测试用例在不同的错

通信作者:张慧(794014804@qq.com)

误中的隶属程度,根据这种隶属程度定义不同测试用例的权重,降低不相关失败的测试用例的干扰,计算语句可疑度进行程序调试。Wang 等[21]将多错误定位变为搜索问题,以二进制染色体表示多错误的候选分布,改进基于覆盖分析技术的 Ochiai 算法作为适应值函数,采用遗传算法得到最优解,从而定位多错误的位置。

从以上的多错误定位方法中我们可以得出两点:首先,已有基本都是从错误本身进行分析;其次,它们都用到了基于覆盖分析技术中的可疑度进行多错误定位。但是,它们都忽略了回归测试用例的重要性。在利用可疑度进行多错误定位时我们会遇到两个问题:一个是相似测试用例中的大众语句,此语句本身没有错误,但却是程序必经路径,所有失败的测试用例和成功的测试用例都可能遍历过,所以语句可疑度的计算有可能会提高这种类型语句的可疑度,进而干扰我们找到真正错误的效率;另一个是偶然性正确的测试用例(执行了错误语句但是结果还是正确的测试用例)的存在会降低错误语句的可疑度。

3　基于深度卷积网络的多错误定位方法

深度学习中的神经网络已经被用于错误定位技术中[22-30],其中的神经元有多个输入[31],每个输入对应一个权重,即传入神经元的刺激,权重类似于神经元的敏感度,所以权重越大,神经元对这个刺激越敏感。最后,将每个输入与其对应的权重相乘,再加上偏置,经过非线性激活函数的转换,得到神经元的最终输出。根据最终输出和预期输出的误差,我们可以再进行反向传播,推导修改每个输入的权重来训练神经网络。经过反复训练,确定与最小误差相对应的权重,训练结束。在错误定位中,通过神经元中每个输入的权重与最终输出之间的关系模拟语句覆盖与执行结果(成功或者失败)之间的关系(即输入的测试用例的路径矩阵),通过神经网络的输出结果调整权值,当达到设定的误差时,输入测试语句的单位矩阵,得到语句的可疑度。由于基于神经网络的错误定位技术完全可以在 Anaconda 等有着丰富的第三方库的集成开发环境下实现科学计算,所以计算复杂度并不高,又由于考虑语句与结果之间的直接关系定位的错误准确度比较高。因此这种技术被越来越多的研究者探索。

神经网络的公式如式(1)所示:

$$\begin{cases} h^{(1)}=\varphi^{(1)}\left(\sum_{i=1}^{m}x_i w_i^{(1)}+b^{(1)}\right) \\ y=\varphi^{(2)}\left(\sum_{j=1}^{n}h_j^{(1)}w_j^{(2)}+b^{(2)}\right) \end{cases} \tag{1}$$

其中,输入 $x\in R^m$,隐含层输出 $h\in R^n$,输出 $y\in R^k$,$w^{(1)}\in R^{m\times n}$ 与 $b^{(1)}\in R^n$ 分别为输入到隐藏层的权值连接矩阵和偏置,$w^{(2)}\in R^{n\times k}$ 与 $b^{(2)}\in R^k$ 分别为隐藏层到输出层的权值连接矩阵和偏置,$\varphi^{(1)}$ 和 $\varphi^{(2)}$ 为相应的激活函数。而反向传播通过总误差反向求导,得到修正后的权值和偏置。

在神经网络中,为了提高准确率,往往采取增加神经网络层数的方式,然而全连接使得层数的增加会导致神经元个数也越来越多,这就造成了计算复杂度高、可扩展性差的问题。为了解决这个问题,研究者们提出了深度卷积网络[31],其也是一种特殊的神经网络。深度卷积网络的主要特点是局部连接和权重共享,局部连接是相对于神经网络的全连接,每个神经元只对局部区域进行感知,而不是对全局图像进行感知。权重共享减少了大部分参数,使得与图像局部连接的所有神经元使用同一组参数。局部连接和权重共享在深度卷积网络中主要通过卷积层和池化层实现,卷积层的作用是运用卷积核提取特征,池化层是将语义上相似的特征合并起来(比如取对应区域的最大值、平均值等)。深度卷积网络根据输出信息(即最终的图像特征)会有不同的网络结构,一般是卷积层、池化层和全连接交叉使用以达到最优的效果。

从以上相关工作中的多错误定位技术的研究可以看出它们都是以错误分析为出发点,并且由于 80% 的软件错误集中在 20% 的程序模块中说明错误与错误之间具有关联性,所以我们也考虑了在这方面表现较好的程序切片[32]。其通过构造程序依赖图 PDG(主要是控制依赖和数据依赖),制定程序切片规则 C,依据切片规则 C,在 PDG 上应用图可达性算法,获得与切片相对应的依赖图,将依赖图切片映射到目标程序中,与依赖图对应的语句集合形成目标程序基于切片准则 C 的程序切片。在程序切片中又分为前向切片和后向切片,在程序分析和测试过程中,前向切片和后向切片的作用均不可忽略。例如,如果对一个已经测试完成的程序进行了简单的修改,我们一定要知道这点修改可能会带来哪些副作用,也就是这点修改可能会影响后面哪些语句和控制谓词。显然这是一个只要计算前向切片的问题,所以前向切片是指构造一个集合 $affect(v/n)$,使得这个集合由受到 n 点的变量 v 的值的影响的所有语句和谓词构成。而另一方面,如果我们在程序测试过程中发现某个错误,我们一定要知道是谁制造了这个错误,也就是前面的那条语句或哪个谓词表达式产生了这个错误,并如何传播到我们发现它的地方。这显然是一个计算后向切片的问题,所以后向切片是指构造一个集合 $affect(v/n)$,使得这个集合由所有影响变量 v 在 n 点的语句和谓词组成。但是,在错误定位中,程序切片技术只用失败的测试用例,这种测试用例用于深度学习会造成欠拟合,需要成功的测试用例进行平和。

本文基于深度卷积网络的多错误定位方法的框架如图 1 所示。

我们首先得到回归测试用例的覆盖路径矩阵,然后开始卷积操作,即让测试用例的覆盖路径矩阵的每一行与[0,0,0,0,0,0,0,0,0,0,1]$^{\mathrm{T}}$相乘,将结果为 1 的特征的测试用例覆盖路径矩阵即失败的测试用例挑选出来,结果为 0 的特征的测试用例覆盖矩阵归为一类(即成功的测试用例)。然后分别对失败的测试用例的卷积层和成功的测试用例的卷积层进行全连接操作(进行全连接操作是分别找到错误语句与失败的测试结果之间的关系以及正确语句与正确的测试结果之间的关系),在进行反向传播推导修改参数时我们将误差设定为 0.5(即实际输出与预期输出的差的绝对值大于 0.5 时要进行反向传播推导修改参数),当达到设定的误差时,输入测试语句的单位矩阵得到两组语句的可疑度,即失败的测试用例的语句可疑度和成功的测试用例的语句可疑度。根据前面得到的结果,我们进入池化层。池化层是将语义上相似的特征合并起来,我们也要找到各种可能性并进行分类。

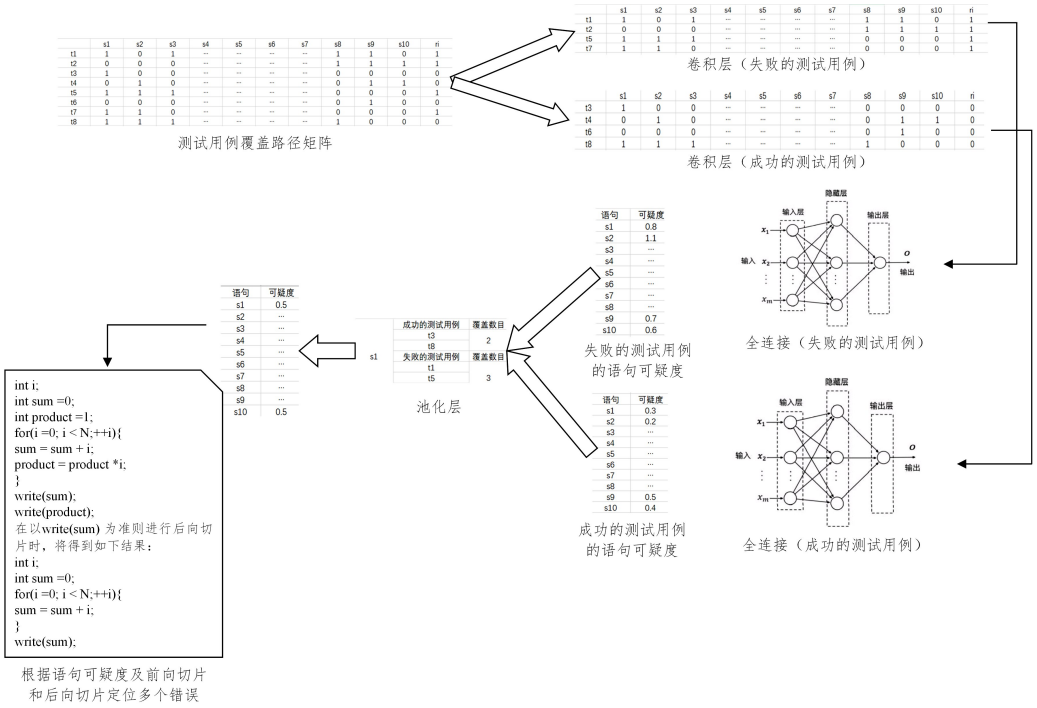

图 1 基于深度卷积的多错误定位方法

Fig. 1 Multiple fault localization method based on deep convolution

(1)相似测试用例中的大众语句：正确的但是成功的测试用例和失败的测试用例可能都要遍历的语句。

这类语句的存在有可能会由于可疑度比较高进而影响错误定位的效率，所以需要给这样的语句一个合适的可疑度。这类语句的特征是在失败的测试用例的语句可疑度中可疑度高，在成功的测试用例的语句可疑度中可疑度低，并且这类语句在失败的测试用例中的数目与在成功的测试用例中的数目差不多。所以当在失败的测试用例的语句可疑度与其在成功的测试用例的语句可疑度之差大于 0.5，在失败的测试用例中的数目与在成功的测试用例中的数目相差 3 以内时，此类语句可疑度一概定义为 0.5。

(2)偶然性正确的测试用例执行的错误语句：对于隐藏性过高的语句，我们还可以通过后期前向切片和后向切片寻找遗漏的偶然性正确的测试用例执行的错误语句。

这类语句的存在会降低错误语句的可疑度，所以需要减轻这类测试用例对可疑度的影响。这类语句的特征是在失败的测试用例的语句可疑度中可疑度高，在成功的测试用例的语句可疑度中可疑度不算低，并且这类语句在失败的测试用例中的数目比较多。所以当在失败的测试用例的语句可疑度与其在成功的测试用例的语句可疑度之差小于 0.5，在失败的测试用例中的数目与在成功的测试用例中的数目之差在 3 以上时，这类语句被认定为偶然性正确的测试用例执行的错误语句，其语句可疑度定义为原在失败的测试用例的语句可疑度加上在失败的测试用例的语句可疑度与其在成功的测试用例的语句可疑度之差的和。

(3)只有失败的测试用例执行的语句：可疑度高(保留其在失败的测试用例的语句可疑度中的可疑度)。

(4)只有成功的测试用例执行的语句：可疑度低(保留其在成功的测试用例的语句可疑度中的可疑度)。

(5)其余语句，保留其在失败的测试用例的语句可疑度中的可疑度。

通过池化层我们得到新的语句可疑度，根据可疑度排序我们选出前 3 名可疑度最高的语句，这 3 条语句分别放入程序中，画出程序依赖图(PDG)，无论这 3 条语句是否错误语句都分别找到这 3 条语句的前向切片和后向切片。例如选择其中一条语句得到其前向切片和后向切片，在两种切片中的每条语句放入前面得到的语句可疑度，此语句的前向切片中可疑度排行前 3 名的语句进行错误定位，此语句的后向切片中可疑度排行前 3 名的语句进行错误定位，由于错误与错误之间的关联，我们可以用此方法发现前期遗漏的偶然性的测试用例里隐藏性很高的错误语句，进而提升多错误定位的效率。

4 实验

(1)实验设计

本文采用 Siemens Suite 和 Space 来验证以上方法的效果，具体实验对象如表 1 所列。Siemens Suite 含有多组不同功能的 C 程序，其模拟了实际项目中可能存在的错误。Space 由欧洲航天局开发，拥有更多的测试用例和代码，它包含 6218 行代码和 38 个错误版本，这些错误都是实际开发过程中产生的。

表 1 实验对象描述

Table 1 Description of experimental subjects

Program	No.	Line	Num	Description
print_tokens	7	472	4130	Lexical analyzer
print_tokens2	10	399	4115	Lexical analyzer
replace	32	512	5542	Pattern replacement
schedule	9	292	2650	Priority scheduler
Schedule2	10	301	2710	Priorityscheduler
tcas	41	141	1608	Altitude separation
tot_info	23	440	1052	Information measure
space	38	6218	13585	Array definition interpreter

本文在 Visual Studio 实验平台上，采用 C 语言修复实验

对象各个代码中的错误,再进行回归测试,进而进行错误定位。使用 Visual Unit[1),运行测试用例,收集测试用例覆盖信息,进而得到测试用例的执行结果。利用 Anaconda 集成开发环境训练本文中的深度卷积网络。结合上述信息,利用本文提出的基于深度卷积网络的多错误定位方法提高错误定位效率。

我们采用代码检查率来验证错误定位效率,即从最高语句可疑度开始检查,一直检查到错误位置,期间所检查的代码占所有代码的比值,比值越低说明错误定位效率越高。

(2)结果分析

实验对比了经典的基于覆盖分析的错误定位方法 Ochiai[33-34]、基于依赖的错误定位方法 PPDG[6]、基于反向传播神经网络的错误定位方法 BP[22]和本文的错误定位方法,结果如图 2 所示。

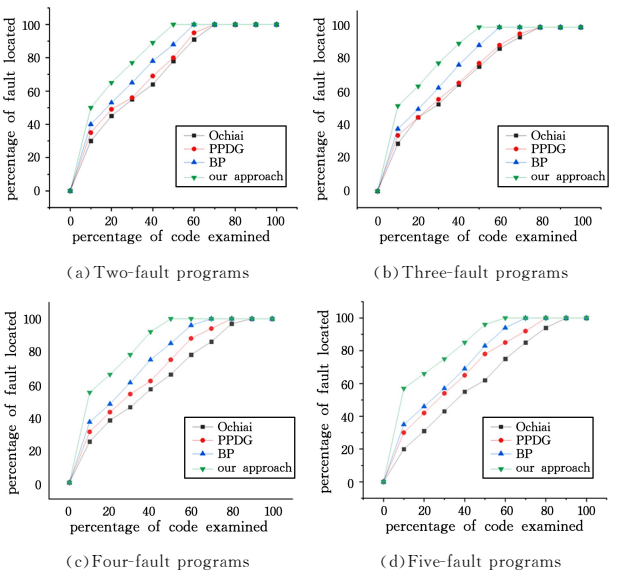

图 2　本文方法和几种经典错误定位方法的多错误的错误定位效率比较

Fig. 2　Comparison of fault localization effectiveness of multiple faults between this method and several classicfault localization methods

从图 2 我们可以看出,Ochiai 方法虽然计算复杂度低,但是没有考虑语句之间的联系,也没有考虑语句与测试结果之间的联系,同时测试用例中存在相似测试用例和偶然性正确的测试用例,使得其多错误的错误定位效率是最低的。其次,PPDG 方法首先通过程序依赖图得到子节点与父节点之间的条件概率,然后在此基础上通过执行失败测试用例的轨迹得到语句可疑度,其虽然考虑了语句之间的关系,但是并没有更多地考虑语句与测试结果之间的关系,同样没有考虑提高测试用例的质量(即相似测试用例和偶然性正确的测试用例的存在),所以其多错误定位的效率高于 Ochiai 方法,但是相对于 BP 方法和本文方法还是比较弱。BP 方法很好地考虑了语句与测试结果之间的关系,这对于错误定位至关重要,但是相似测试用例和偶然性正确的测试用例的存在使得其错误定位效率受到影响,并且最终得到的语句可疑度没有通过程序依赖图进行多错误定位也影响了错误定位的效率。所以随着程序中的错误越来越多,PPDG 方法的多错误定位效率与 BP 方法的多错误定位效率越来越接近。本文方法由于首先考虑了错误语句与失败的测试结果以及正确语句与正确的测试结果之间的关系,同时减轻了相似测试用例和偶然性正确的测试用例对错误定位的影响,最后通过基于程序依赖图的前向切片和后向切片定位多错误,所以一般检查 60%的代码就可以找到多个错误。

(3)有效性威胁

本文没有考虑历史信息对多错误定位的有效价值,回归测试与软件调试是一直迭代直至找到所有代码中的错误为止。根据错误与错误之间的关联性以及错误的隐藏性,如何有效利用上一轮错误定位得到的错误信息(例如错误语句的具体位置,上一轮得到的语句可疑度),以期更好地提高多错误定位的效率,值得去探索。

本实验选择的对象是广泛应用于错误定位研究的经典程序,但是目前软件系统越来越复杂,特别是智能化系统越来越多,在未来的工作中,我们将关注更多大型的现实工程中的程序,模拟尽可能多的实际错误,进行更有实际应用价值的实验。

结束语　本文方法使用一种特殊的深度卷积网络解决目前多错误定位面临的问题,其中卷积层的作用是运用卷积核提取特征,池化层是将语义上相似的特征合并起来。我们首先将卷积层的特征应用于错误定位中,结合全连接分别得到失败的测试用例的语句可疑度和成功的测试用例的语句可疑度(即错误语句与失败的测试结果之间的关系以及正确语句与成功的测试结果之间的关系),再利用池化层的特征找到相似测试用例中的大众语句、偶然性正确的测试用例等,最后根据得到的更新后的语句可疑度通过前向切片和后向切片进行多错误定位。实验证明,本文方法可以提升多错误定位的效率。

未来,我们准备将前期得到的错误语句可疑度和最新回归测试用例的测试结果都进行应用,并且通过多特征向量、定向搜索和多条件概率等将过去得到的结果与现在已有的信息很好地结合,更好地提升多错误定位效率。

参 考 文 献

[1] WEISER M. Program Slicing[J]. IEEE Transactions on Software Engineering,1984,SE-10(4):352-357.

[2] JU X,JIANG S,XIANG C,et al. HSFal:Effective fault localization using hybrid spectrum of full slices and execution slices[J]. Journal of Systems & Software,2014,90(APR.):3-17.

[3] KOREL B. PELAS-program error-locating assistant system[J]. IEEE Transactions on Software Engineering,1988,14(9):1253-1260.

[4] LIAN L,KUSUMOTO S,KIKUNO T,et al. A new fault localizing method for the program debugging process[J]. Information & Software Technology,1997,39(4):271-284.

[5] MAO X,YAN L,DAI Z,et al. Slice-based statistical fault localization[J]. Journal of Systems & Software,2014,89(MAR.):51-62.

[6] BAAH G K,PODGURSKI A,HARROLD M J. The Probabilistic Program Dependence Graph and Its Application to Fault Diagnosis[J]. IEEE Transactions on Software Engineering,2009,36(4):528-545.

1) http://www.kailesoft.com

[7] ZHANG Z, CHARS W K, TSETH, et al. Capturingpropagation of infected program states[C]// Proceedings of the7th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering(ESEC/FSE'09). New York: ACM, 2009: 43-52.

[8] ZELLER A, HILDEBRANDT R. Simplifying and Isolating Failure-Inducing Input[J]. IEEE Transactions on Software Engineering, 2002, 28(2): 183-200.

[9] CHENG H, LO D, ZHOU Y, et al. Identifying bug signatures using discriminative graph mining[C]// Proceedings of the Eighteenth International Symposium on Software Testing and Analysis. Chicago: ACM, 2009: 141-152.

[10] NIU X T, NIE C H, LEI Y, et al. Identifying failure-inducing combinations using tuple relationship[C]// 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. Luxembourg: IEEE, 2013: 271-280.

[11] JONESJ A, HARROLDM J. Empirical evaluation of the tarantula automatic fault-localization technique[C]// Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering. Long Beach CA: ACM, 2005: 273-282.

[12] CLEVE H, ZELLER A. Locating causes of program failures [C]// 27th International Conference on Software Engineering. St. Louis, MO: IEEE, 2005: 342-351.

[13] HARROLD M J, ROTHERMEL G, SAYRE K, et al. An empirical investigation of the relationship between spectra differences and regression faults[J]. Software Testing, Verification and Reliability, 2000, 10(3): 171-194.

[14] RENIERES M, REISS S P. Fault localization with nearest neighbor queries[C]// 18th IEEE International Conference on Automated Software Engineering. Montreal, QC: IEEE, 2003: 30-39.

[15] WONG W E, DEBROY V, CHOI B. A family of code coverage-based heuristics for effective fault localization[J]. Journal of Systems & Software, 2010, 83(2): 188-208.

[16] WONG W E, DEBROY V, LI Y, et al. Software fault localization using dstar (d*)[C]// 2012 IEEE Sixth International Conference on Software Security and Reliability. Gaithersburg, MD: IEEE, 2012: 21-30.

[17] WONG W E, DEBROY V, GAO R, et al. The DStar Method for Effective Software Fault Localization[J]. IEEE Transactions on Reliability, 2014, 63(1): 290-308.

[18] WEN W Z, LI B X, SUN X B, et al. Multiple error location based on conditional execution slice spectrum[J]. Computer Research and Development, 2013, 50(5): 1030-1043.

[19] CAO H L, JIANG S J. Multiple error location method based on Chameleon cluster analysis[J]. Acta Electronica Sinica, 2017, 45(2): 394-400.

[20] WANG X Y, JIANG S J, GAO P F, et al. A software multiple defect location method based on fuzzy C-means clustering[J]. Chinese Journal of Computers, 2020, 43(2): 206-232.

[21] WANG Z, FAN X Y, ZOU Y G, et al. A Multiple Defect Location Method Based on Genetic Algorithm[J]. Journal of Software, 2016, 27(4): 879-900.

[22] WONG W E, QI Y. BP Neural Network-Based Effective ault Localization[J]. International Journal of Software Engineering and Knowledge Engineering, 2011, 19(4): 573-597.

[23] WONG W E, SHI Y, QI Y, et al. Using an RBF neural network to locate program bugs[C]// 2008 19th International Symposium on Software Reliability Engineering (ISSRE). Seattle, WA: IEEE, 2008: 27-36.

[24] DEBROYV, WONG W, XU X F, et al. A grouping-based strategy to improve the effectiveness of fault localization techniques [C]// 2010 10th International Conference on Quality Software. IEEE, 2010: 13-22.

[25] LAM A N, NGUYEN A T, NGUYEN H A, et al. Bug localization with combination of deep learning and information retrieval [C]// 2017 IEEE/ACM 25th International Conference on Program Comprehension (ICPC). Buenos Aires: IEEE, 2017: 218-229.

[26] ZHANG Z, LEI Y, TAN Q, et al. Deep Learning-Based Fault Localization with Contextual Information[J]. IEICE Transactions on Information and Systems, 2017, E100. D(12): 3027-3031.

[27] ZHENG W, HU D, WANG J. Fault Localization Analysis Based on Deep Neural Network[J]. Mathematical Problems in Engineering, 2016, 2016(pt. 4): 1-11.

[28] ENISER H F, GERASIMOU S, SEN A. Deepfault: Fault localization for deep neural networks[C]// International Conference on Fundamental Approaches to Software Engineering. Prague: Springer, 2019: 171-191.

[29] MARU A, DUTTA A, KUMAR K V, et al. Effective Software Fault Localization Using a Back Propagation Neural Network [C]// Computational Intelligence in Data Mining. Singapore: Springer, 2020: 513-526.

[30] HERIS S R, KEYVANPOUR M R. Effectiveness of Weighted Neural Network on Accuracy of Software Fault Localization [C]// 2019 5th International Conference on Web Research (ICWR). Tehran: IEEE, 2019: 100-104.

[31] LIU P. Artificial Intelligence[M]. Beijing: China Water Power Press, 2021: 92-98.

[32] LI B X. Technology and Application of Program Slicing[M]. Bei-jing: Science Press, 2006: 7.

[33] ABREU R, ZOETEWEIJ P, VAN GEMUND A J C. On the accuracy of spectrum-based fault localization[C]// Testing: Academic and Industrial Conference Practice and Research Techniques-MUTATION (TAICPART-MUTATION 2007). Windsor: IEEE, 2007: 89-98.

[34] ABREU R, ZOETEWEIJ P, VAN GEMUND A J C. An evaluation of similarity coefficients for software fault localization[C]// 2006 12th Pacific Rim International Symposium on Dependable Computing (PRDC'06). Riverside, CA: IEEE, 2006: 39-46.

ZHANG Hui, born in 1982, Ph.D, lecturer. Her main research interests include fault localization and software testing.