



## 基于天牛须搜索的软件测试数据扩增方法

王曙燕,胡乾花,孙家泽

(西安邮电大学 计算机学院,西安 710121)

**摘 要:**为使原测试用例集满足软件演化后新版本程序的测试需求,提出一种基于天牛须搜索算法的软件测试数据扩增方法。静态分析新旧版本程序,获取调用图和程序执行信息并得到所需测试的目标方法集,通过计算目标方法包含错误的影响度获得有序目标方法集。根据原测试用例集的覆盖信息选取部分测试用例作为初始的进化种群,基于分支距离和分支嵌套深度设计适应度函数,采用改进的天牛须搜索算法对有序目标方法集实现测试数据扩增。实验结果表明,与基于遗传算法和粒子群优化算法的测试数据扩增方法相比,该方法的测试数据扩增效率约平均提升49.91%和24.76%,且有效降低了回归测试成本。

**关键词:**回归测试;测试用例扩增;目标方法集;天牛须搜索算法;Metropolis准则

开放科学(资源服务)标志码(OSID):



**中文引用格式:**王曙燕,胡乾花,孙家泽.基于天牛须搜索的软件测试数据扩增方法[J].计算机工程,2021,47(9):191-196.

**英文引用格式:**WANG S Y, HU Q H, SUN J Z. Software test data augmentation method based on beetle antennae search[J]. Computer Engineering, 2021, 47(9): 191-196.

## Software Test Data Augmentation Method Based on Beetle Antennae Search

WANG Shuyan, HU Qianhua, SUN Jiaze

(School of Computer Science and Technology, Xi'an University of Posts and Telecommunications, Xi'an 710121, China)

**[Abstract]** To bring the original test case set to the test requirements of an evolved program, a data augmentation method based on Beetle Antennae Search (BAS) algorithm is proposed for software test. Through the static analysis of the old and new versions of the program, the call graph and program execution information are extracted, and on this basis the to-be-tested target method set is obtained. By calculating the influence of the method that contains faulty data, an ordered target method set is acquired. According to the coverage information of the original test case set, some test cases are selected as the initial evolutionary population, and the fitness function is designed according to the branch distance and branch nesting depth. On this basis, the improved BAS algorithm is used to augment the test data of the ordered target method set. Experimental data show that the method improves the amplification efficiency by 49.91% compared with the Genetic Algorithm (GA)-based method, and 24.76% compared with the Particle Swarm Optimization (PSO) algorithm-based method. In addition, it reduces the cost of regression test.

**[Key words]** regression test; test case augmentation; target method set; Beetle Antennae Search (BAS) algorithm; Metropolis criterion

**DOI:** 10.19678/j.issn.1000-3428.0058936

### 0 概述

回归测试是指测试人员对演化后的新版本程序重新进行测试,确认修改部分没有影响未修改部分的功能<sup>[1]</sup>。测试用例集扩增强调利用原测试数据信

息和程序演化信息辅助生成新的测试数据,以期提高生成效率并获得更具针对性的测试用例<sup>[2]</sup>。

测试数据扩增概念由 HARDER 等<sup>[3]</sup>提出,之后国内外研究人员对其进行大量研究并取得一定的研究成果。吴川等<sup>[4]</sup>提出基于分析路径相关性来建立

**基金项目:**陕西省重点研发计划项目“多线程程序并发故障智能测试关键技术研究”(2020GY-010);西安市科技计划项目“基于群体智能的多目标软件测试优化关键技术研究”(2019218114GXRC017CG018-GXYD17.10)。

**作者简介:**王曙燕(1964—),女,教授、博士,主研方向为软件测试、数据挖掘、智能信息处理;胡乾花(通信作者),硕士研究生;孙家泽,教授、博士。

**收稿日期:**2020-07-14 **修回日期:**2020-08-18 **E-mail:** flower\_hqh@163.com

回归测试数据生成模型,虽然目标路径的覆盖率较高,但对于不同的被测程序,需要根据程序的特征设置不同的控制参数。YOO等<sup>[5]</sup>将测试数据分为失效和未失效两类,且仅修改程序中的简单运算符。XU等<sup>[6]</sup>研究表明使用已有测试数据可以提高测试数据集扩增效率。QI等<sup>[7]</sup>利用符号执行技术和状态传递树CEPT<sup>[8]</sup>确保测试数据可以覆盖修改后程序的变化节点,JIANG等<sup>[9]</sup>通过分析软件信息图扩增测试数据,但QI和JIANG等的研究均未利用原测试数据集。巩敦卫等<sup>[10]</sup>根据与目标路径的相似度选取初始种群,采用遗传算法生成测试数据,虽然合理地利用了原测试用例,但是并未将程序的演化信息与交叉算子相结合。殷鹏川等<sup>[11]</sup>利用原测试用例跳过重叠初始子路径,对后续目标子路径进行concolic测试,但该方法受限于约数求解器。王曙燕等<sup>[12]</sup>利用路径相似度识别新增路径并选取初始种群,采用自适应粒子群算法生成测试用例,但该方法不适用于大规模软件测试。吴川等<sup>[13]</sup>基于路径与测试数据之间的关系,决定需要扩增的测试数据,但该方法未利用原测试用例。XU等<sup>[14]</sup>研究表明已有测试用例的使用方式对回归测试的效率有较大的影响。王曙燕等<sup>[15]</sup>从方法级别和语句级别提取覆盖目标,将原测试用例和正交种群作为初始种群,虽然利用了原测试数据,但并没有对其进行选择,数据冗余性较大。对于上述研究存在的问题,需要一种与程序演化信息密切结合、能充分利用原测试用例且适用于较多程序的测试用例扩增算法,以在降低回归测试成本的同时提高扩增效率。

基于符号执行的软件测试方法受限于本身的高复杂度,难以适用于大规模软件测试,同时由于影响回归测试效果的主要因素是测试用例生成算法,而元启发式算法能有效提高测试用例扩增效率<sup>[16-17]</sup>。天牛须搜索(Beetle Antennae Search, BAS)算法是一种新型的元启发式算法<sup>[18-19]</sup>,具有参数少且鲁棒性强等优点,自提出以来受到了学术界的广泛应用并取得了一系列的成果,但在测试用例扩增应用中鲜有公开的文献。本文在文献[15]研究的基础上,提出基于天牛须搜索算法的软件测试数据扩增方法MBAS。结合软件演化信息,基于原测试用例集的覆盖信息选取部分测试用例作为初始的进化种群,根据分支距离和分支嵌套深度设计适应度函数,采用改进的天牛须搜索算法对有序目标方法集进行测试数据的扩增,以期提高原测试用例的利用率和程序扩增的效率。

## 1 有序目标方法集获取

利用已有的测试数据执行新旧程序,根据程序覆盖与执行信息得到需测试的目标方法集合。获取

程序的方法调用图并将其用邻接矩阵存储,计算方法执行概率和权值,得到方法包含错误的影响度,获取优先覆盖的目标方法。

### 1.1 程序覆盖与执行信息获取

一个程序包括 $m$ 个方法 $M=\{f_1, f_2, \dots, f_m\}$ ,对程序执行测试用例后,得到的方法覆盖信息与执行信息用矩阵 $A$ 表示如下:

$$A = \begin{bmatrix} t_{11}u_{11} & \cdots & t_{1n}u_{1n} \\ \vdots & & \vdots \\ t_{m1}u_{m1} & \cdots & t_{mn}u_{mn} \end{bmatrix}$$

在矩阵 $A$ 中,测试数据为 $t_1, t_2, \dots, t_n, t_{ij} (1 \leq i \leq m, 1 \leq j \leq n)$ 表示测试用例 $t_j$ 对于 $f_i$ 方法的覆盖情况,如果 $t_j$ 覆盖了 $f_i$ ,则 $t_{ij}=1$ ,否则 $t_{ij}=0$ 。 $t_j$ 的执行结果为 $u=[u_{1j}, u_{2j}, \dots, u_{ij}, \dots, u_{mj}]^{-1}$ , $u_{ij}$ 表示 $t_j$ 对 $f_i$ 的执行结果,如果成功,则 $u_{ij}=1$ ,否则 $u_{ij}=0$ 。

测试用例在新旧版本程序中的方法覆盖与执行情况表示为 $A$ 和 $A'$ ,通过对同一测试用例的覆盖信息和执行信息做异或运算,得到目标方法集 $C=\{f_i \in M | (t_{ij} \oplus t'_{ij}=1) \cup ((t_{ij} \oplus t'_{ij}=0) \wedge (u_{ij} \oplus u'_{ij}=1))\}$ 。

### 1.2 目标方法集排序

获取程序的方法调用图,方法调用图是一个有向图,其中每个节点都附加一个权值。建立方法调用图的邻接矩阵,若存在调用关系,则矩阵元素为1,否则为0。对矩阵进行遍历,统计每个方法可调用的方法个数。

#### 1.2.1 目标方法包含错误的概率

通过目标方法在新旧版本程序上的执行概率得到其包含错误的概率。假定方法集相互独立,初始方法 $f_i$ 的执行概率 $P(f_i)=1$ ,若 $f_i$ 调用 $f_a \cdots f_b$ 等 $w_i$ 个方法,则 $f_a$ 的执行概率 $P(f_a)$ 表示如下:

$$P(f_a) = \begin{cases} \frac{P(f_i)}{w_i}, & a \neq 1 \text{ 且 } P(f_a)=0 \\ 1.00, & a=1 \\ P(f_a) + \frac{P(f_i)}{w_i}, & a \neq 1 \text{ 且 } P(f_a) \neq 0 \end{cases}$$

若被调用方法的执行概率不为0时,则叠加概率值,直至计算至最后一个节点。计算新旧程序中标方法 $c_k (1 \leq k \leq m)$ 的执行概率 $P(c_k)$ 和 $P'(c_k)$ ,目标方法 $c_k$ 包含错误的概率 $P''(c_k)$ 表示如下:

$$P''(c_k) = |P(c_k) - P'(c_k)|$$

#### 1.2.2 目标方法包含错误的影响度

将方法调用图中所有非叶子节点的权值 $N(f_i)$ 初始化为0,所有叶子节点的权值为1。若 $f_i$ 调用 $f_a \cdots f_b$ 等 $w_i$ 个方法,则 $f_i$ 的方法权值 $N(f_i)$ 表示如下:

$$N(f_i) = \begin{cases} 1, f_i \text{ 为叶子节点} \\ \sum_{i=a}^b N(f_i), \text{ 其他} \end{cases}$$

计算新程序中目标方法  $c_k$  的权值  $N(c_k)$ , 将权值和包含错误的概率进行乘积得到目标方法包含错误的影响度  $S(c_k)$ :

$$S(c_k) = P''(c_k) \cdot N(c_k)$$

将  $S(c_k)$  值按递减的顺序进行排列, 得到有序目标方法集  $C'$ , 并对其依次生成测试数据。

## 2 测试数据集扩增

基于天牛须搜索算法的软件测试数据扩增方法可分为预处理和测试用例扩增两个模块, 如图1所示。

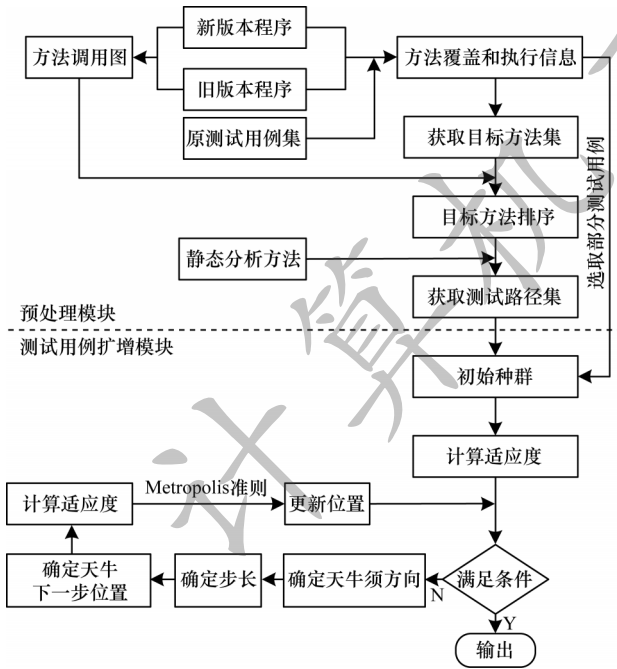


图1 MBAS模型

Fig.1 MBAS model

预处理模块主要包括程序静态分析和获取目标方法集, 抽取方法调用图, 使用原测试用例执行新旧程序, 得到程序执行信息, 获得目标方法集并对目标方法进行排序。测试用例扩增模块首先根据方法覆盖信息从原测试用例集中选取部分用例作为初始种群, 再通过改进的天牛须搜索算法的位置更新公式引导天牛向最优解进化, 最终生成覆盖目标路径的测试数据。

### 2.1 初始种群选择

将原测试用例集在新程序中执行, 得到程序执行信息矩阵, 对于目标方法  $c_k$ , 将所有覆盖该方法的测试用例作为扩增的初始测试数据, 若无测试数据覆盖该方法, 则将在旧程序中覆盖该方法的用例添加至初始种群。

### 2.2 适应度函数

使用代码分析工具 Soot 得到  $C'$  中每个方法的控制流程图, 根据路径的分支节点数获得需要覆盖的路径集, 在目标路径的每个分支节点前插入分支距离函数  $H_p$ , 表示第  $p$  个分支的分支距离。分支距离是一种常见的启发式方法, 用于指导输入数据进行搜索, 以解决分支的逻辑谓词中的约束问题<sup>[20]</sup>。通过分支嵌套深度调整不同分支节点权重:

$$G_p = \frac{D_p}{\sum_{p=1}^q D_p}$$

其中:  $D_p$  为通过程序流程图静态分析得到分支节点  $p$  的嵌套深度;  $q$  为被测程序针对给定目标路径的分支判定数目。通过上式计算出  $G_p$  为分支节点  $p$  的权重。

根据分支距离函数和分支嵌套深度设计适应度函数:

$$F(x) = \frac{1}{\varepsilon + \sum_{p=1}^q G_p \cdot (1 - 1.001)^{-H_p}}$$

其中:  $\varepsilon$  是一个极小的常量以保证分母不为 0, 在此处设置  $\varepsilon = 0.01$ 。若  $F(x)$  的值越小, 则表示越接近目标, 因此目标路径覆盖问题可转换为适应度函数最小化问题。

### 2.3 改进的天牛须搜索算法

在  $D$  维解空间中, 天牛位置  $x = (x_1, x_2, \dots, x_D)$ , 其左右两触须的位置为:

$$\begin{cases} x_l = x + l \cdot d \\ x_r = x - l \cdot d \end{cases}$$

$$d = \left[ r \times \frac{2}{\text{norm}(r)} \right]$$

其中:  $x$  为当前位置;  $l$  为天牛质心与触须间的距离;  $x_l$  为左须的位置;  $x_r$  为右须的位置;  $d$  为天牛的随机朝向;  $r = \text{rand}(n, 5)$  是一个随机向量。在本文中, 天牛的朝向为右须指向左须。天牛通过不断对比  $x_r$  与  $x_l$  附近位置的适应度, 向适应度低的方向进行移动。

当前位置天牛根据规则移动到下一个位置:

$$x' = x + s \cdot d \cdot \text{Sign}(F(x_l) - F(x_r))$$

其中:  $s$  为可变步长;  $F(x_l)$  为左须的气味强度, 气味强度即适应度函数;  $F(x_r)$  为右须的气味强度, 若  $F(x_l) > F(x_r)$ ,  $\text{Sign}(\cdot)$  取 1, 天牛在  $d$  的方向上以步长  $s$  移动, 反之, 则向  $d$  的反方向移动。

在本文中, 使用 Metropolis 准则更新天牛的下一步位置, 若新位置  $x'$  的适应值小于  $x$  的适应值, 则接受  $x'$  为当前位置, 反之, 则以随机概率接受  $x'$ 。步长遵循:

$$s = \begin{cases} 1 + s - \left\lfloor s \times \frac{F_{\text{ave}} - F}{\frac{F_{\text{max}} - F_{\text{min}}}{2}} \right\rfloor, & F \leq F_{\text{ave}} \\ s, & F > F_{\text{ave}} \end{cases}$$



其中: $F$ 为当前个体的适应值; $F_{ave}$ 为去掉最大最小适应值后剩余个体适应值的平均数,称为假平均适应值; $F_{max}$ 为最大适应值; $F_{min}$ 为最小适应值。若 $F \leq F_{ave}$ ,则此时个体的性能良好,减小步长,反之,增大步长。

若达到最大迭代次数或测试数据覆盖目标路径,则输出测试数据,选取 $C'$ 中下一个目标方法,将已测试目标方法成功的测试用例和根据程序执行信息选取的测试用例作为初始测试数据,继续进行扩增,直至所有目标方法都被测试。

## 2.4 算法步骤

基于天牛须搜索的测试数据扩增算法的具体步骤如下:

**输入** 旧版本程序 $P$ 的测试用例集 $T$ ,新版本程序 $P'$

**输出** 覆盖 $P'$ 的测试用例集 $T'$

**步骤1** 抽取方法调用图,使用原测试用例集运行新旧程序,获得程序的执行信息,得到需覆盖的目标方法集。

**步骤2** 对目标方法集进行排序,逐个选取目标路径,进行分支函数的插桩。

**步骤3** 根据程序执行信息选取部分测试用例。

**步骤4** 判断是否满足终止条件(测试用例覆盖目标路径或达到最大迭代次数),若满足,则转步骤9。

**步骤5** 计算测试用例的适应值,确定全局最大适应值、最小适应值和假平均适应值。

**步骤6** 利用位置更新方程预测天牛的位置。

**步骤7** 根据Metropolis准则更新天牛下一步位置。

**步骤8** 更新步长,转步骤4。

**步骤9** 算法结束,输出测试数据集。

## 3 实验与结果分析

### 3.1 实验设置

为验证MBAS方法的数据扩增效果,选用西门子工业程序集中的Schedule和Tcas、三角形判定程序Triangle以及基准程序NextDay作为实验中的测试程序,这些程序被广泛应用于验证不同测试方法的有效性<sup>[21]</sup>。被测程序的基本信息如表1所示。

表1 被测程序的基本信息

被测程序	行数	方法个数	输入实参个数	输入范围
Schedule	412	18	可变	[0,2 047]
Tcas	138	9	12	[0,10 000]
Triangle	50	8	3	[0,100]
NextDay	60	2	3	[0,10 000]

在实验中应用Eclipse IDE for Eclipse Committers (Mars.2)编程实现天牛须算法和测试程序。针对不同程序运行算法30次,取实验结果的均值进行比较。实验对比方法为基于遗传算法(Genetic Algorithm,GA)的测试数据扩增方法(下文简称为GA方法)和基于粒子群优化(Particle Swarm Optimization,PSO)算法的测试数据扩增方法(下文简称为PSO方法),具体参数设置如表2所示。

表2 对比方法参数设置

对比方法	参数名称	参数取值
GA	变异概率	0.9
	交叉概率	0.2
PSO	学习因子	$c_1=c_2=2$
	惯性权重	$W_{max}=1.8, W_{min}=0.2$
	种群编码	二进制编码

### 3.2 实验结果

实验选择GA和PSO方法与本文MBAS方法进行比较,在相同的环境下记录每种方法的迭代次数和运行时间,并计算实验数据的均值和标准差,实验结果如表3和表4所示。

表3 3种方法扩增测试数据的迭代次数

被测程序	指标	迭代次数		
		GA方法	PSO方法	MBAS方法
Schedule	均值	165.1	109.9	94.4
	标准差	27.5	15.7	7.4
Tcas	均值	81.6	53.2	46.1
	标准差	35.3	7.2	5.3
Triangle	均值	7.8	5.1	3.2
	标准差	8.4	3.3	1.0
NextDay	均值	39.0	27.1	17.8
	标准差	31.3	5.2	2.9

表4 3种方法扩增测试数据的运行时间

被测程序	指标	运行时间/s		
		GA方法	PSO方法	MBAS方法
Schedule	均值	6.698	5.695	4.126
	标准差	0.950	0.461	0.401
Tcas	均值	3.603	2.566	2.029
	标准差	0.473	0.200	0.169
Triangle	均值	0.210	0.164	0.073
	标准差	0.214	0.061	0.030
NextDay	均值	1.033	0.481	0.261
	标准差	0.409	0.069	0.052

以程序NextDay为例,在表3中,GA方法需迭代39.0次,PSO方法需迭代27.1次,MBAS方法仅需迭代17.8次即可生成覆盖目标路径的测试数据。在表4中,GA方法的运行时间为1.033 s,PSO方法的运行时间为0.481 s,而MBAS方法的运行时间仅为0.261 s。在迭代次数和运行时间上MBAS方法明显优于其他方法。

对实验结果中的迭代次数进行计算:

$$v(A|B) = \frac{e(B) - e(A)}{e(B)}$$

其中: $v(A|B)$ 为方法A相对于方法B提高的效率; $e$ 表示方法的迭代次数。MBAS方法相对GA和PSO方法的扩增效率提升结果如图2所示。

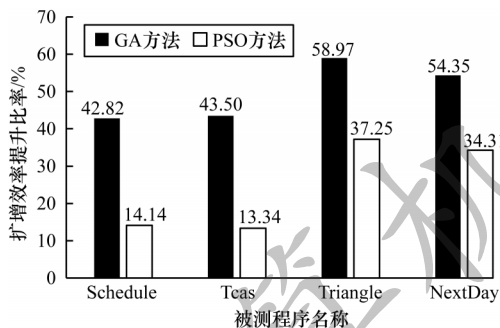


图2 MBAS方法相对GA和PSO方法的扩增效率提升结果  
Fig.2 Improvement results of augmentation efficiency for MBAS method compared to GA and PSO method

在图2中,MBAS方法的扩增效率均高于GA和PSO方法。对于输入实参个数少的被测程序Triangle和NextDay,提升的平均扩增效率分别为48.11%和44.33%,而对于Schedule和Tcas,提升的平均扩增效率仅为28.48%和28.42%,即测试数据处于低维空间时扩增效率提升的更明显。对于测试数据扩增效率,MBAS方法比GA方法约平均提升了49.91%,比PSO方法约平均提升了24.76%。

根据表3和表4中的标准差,对于被测程序,MBAS方法迭代次数的平均标准差为4.15,均小于GA方法的平均标准差25.62和PSO方法的平均标准差7.85;MBAS方法运行时间的平均标准差为0.163,均小于GA方法的平均标准差0.511和PSO方法的平均标准差0.197,即MBAS方法具有更好的稳定性。

通过测试数据的迭代次数和目标路径覆盖率来评价不同回归测试数据扩增方法的能力,以程序Triangle为例,3种方法的目标路径覆盖率如图3所示。

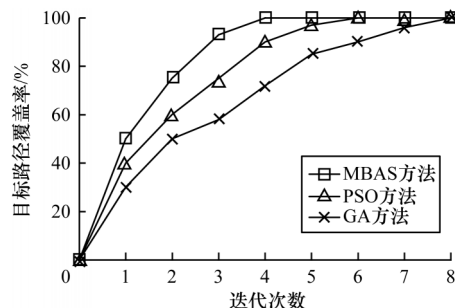


图3 3种方法的目标路径覆盖率对比结果

Fig.3 Comparison results of target path coverage of three methods

由图3可知,MBAS方法的折线位置位于GA和PSO方法的上方,当迭代次数相同时,MBAS方法对目标路径的覆盖率最大;当目标路径覆盖率相同时,以100%为例,MBAS方法所需的迭代次数最少,优先完成路径覆盖,即MBAS方法在迭代次数和目标路径覆盖率方面均具有一定的性能优势。

通过上述分析可知,对于被测程序,MBAS方法可以有效利用原测试集,快速演化生成覆盖目标路径的测试集。

#### 4 结束语

本文提出一种基于天牛须搜索的软件测试数据扩增方法MBAS,采用目标方法覆盖信息分析需要测试的方法,通过计算目标方法包含错误的影响度,对包含错误影响度大的目标方法进行优先分析生成测试数据集,并利用原测试数据集的目标方法覆盖信息来衡量测试数据的优劣,最终使用天牛须搜索算法演化测试用例使其覆盖目标路径。通过对多个程序进行测试,并将该方法与基于GA和PSO的测试数据扩增方法进行比较,实验结果表明,MBAS方法在测试数据的扩增效率和稳定性上均具有明显的性能优势。但由于天牛须搜索算法未能较好解决高维空间中解的收敛速度较慢的问题,因此后续将对此做进一步改进,提升MBAS方法在高维空间中的扩增效率。

#### 参考文献

- [1] YOO S, HARMAN M. Regression testing minimization, selection and prioritization: a survey[J]. Software Testing, Verification and Reliability, 2012, 22(2): 67-120.
- [2] 张智轶,陈振宇,徐宝文,等. 测试用例演化研究进展[J]. 软件学报, 2013, 24(4): 663-674.  
ZHANG Z Y, CHEN Z Y, XU B W, et al. Research progress on test case evolution[J]. Journal of Software, 2013, 24(4): 663-674. (in Chinese)
- [3] HARDER M, MELLE J, ERNST M D. Improving test suites via operational abstraction[C]//Proceedings of the 25th International Conference on Software Engineering. Washington D. C., USA: IEEE Press, 2003: 60-71.

- [4] 吴川, 巩敦卫. 基于路径相关性的回归测试数据进化生成[J]. 计算机学报, 2015, 38(11): 2247-2261.  
WU C, GONG D W. Evolutionary generation of test data for regression testing based on path correlation[J]. Chinese Journal of Computers, 2015, 38(11): 2247-2261. (in Chinese)
- [5] YOO S, HARMAN M. Test data regeneration: generating new test data from existing test data[J]. Software Testing, Verification and Reliability, 2012, 22(3): 171-201.
- [6] XU Z H, COHEN M B, ROTHERMEL G. Factors affecting the use of genetic algorithms in test suite augmentation[C]// Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. New York, USA: ACM Press, 2010: 1365-1372.
- [7] QI D W, ROYCHOUDHURY A, LIANG Z K. Test generation to expose changes in evolving programs[C]// Proceedings of IEEE/ACM International Conference on Automated Software Engineering. New York: ACM Press, 2010: 397-406.
- [8] PEARL J. Probabilistic reasoning in intelligent systems: networks of plausible inference[J]. Computer Science Artificial Intelligence, 1988, 70(2): 1022-1027.
- [9] JIANG B, TSE T H, GRIESKAMP W, et al. Regression testing process improvement for specification evolution of real-world protocol software[C]// Proceedings of the 10th International Conference on Quality Software. Washington D. C., USA: IEEE Press, 2010: 62-71.
- [10] 巩敦卫, 任丽娜. 回归测试数据进化生成[J]. 计算机学报, 2014, 37(3): 489-499.  
GONG D W, REN L N. Evolutionary generation of regression test data[J]. Chinese Journal of Computers, 2014, 37(3): 489-499. (in Chinese)
- [11] 殷鹏川, 袁可荣. 基于路径引导的回归测试用例集扩增方法[J]. 计算机工程与科学, 2014, 36(11): 2159-2163.  
YIN P C, BEN K R. Path-directed regression test suite augmentation[J]. Computer Engineering and Science, 2014, 36(11): 2159-2163. (in Chinese)
- [12] 王曙燕, 温春琰, 孙家泽. 基于自适应粒子群优化算法的测试数据扩增方法[J]. 计算机应用, 2016, 36(9): 2492-2496.  
WANG S Y, WEN C Y, SUN J Z. Test data augmentation method based on adaptive particle swarm optimization algorithm[J]. Journal of Computer Applications, 2016, 36(9): 2492-2496. (in Chinese)
- [13] 吴川, 巩敦卫, 姚香娟. 基于分支覆盖的回归测试路径选择[J]. 软件学报, 2016, 27(4): 839-854.  
WU C, GONG D W, YAO X J. Selection of paths for regression testing based on branch coverage[J]. Journal of Software, 2016, 27(4): 839-854. (in Chinese)
- [14] XU Z, KIM Y, KIM M, et al. Directed test suite augmentation: an empirical investigation[J]. Software Testing, Verification and Reliability, 2015, 25(2): 77-114.
- [15] 王曙燕, 高露, 孙家泽. 基于搜索的分层回归测试数据集扩增方法[J]. 计算机应用研究, 2019, 36(7): 2075-2080.  
WANG S Y, GAO L, SUN J Z. Search-based hierarchical regression test suite augmentation method[J]. Application Research of Computers, 2019, 36(7): 2075-2080. (in Chinese)
- [16] XU Z H, KIM Y H, KIM M Z. Directed test suite augmentation: techniques and tradeoffs[C]// Proceedings of the 18th ACM SIGSOFT International Symposium on the Foundations of Software Engineering. New York, USA: ACM Press, 2010: 257-266.
- [17] PHIL M. Search-based software test data generation: a survey[J]. Software Testing, Verification and Reliability, 2004, 14(2): 105-156.
- [18] JIANG X Y, LI S. Beetle Antennae Search Without Parameter Tuning (BAS-WPT) for multi-objective optimization[J]. Filomat, 2020, 34(15): 5113-5119.
- [19] WANG J, CHEN H. BSAS: beetle swarm antennae search algorithm for optimization problems[EB/OL]. [2020-06-14]. <https://arxiv.org/abs/1807.10470>.
- [20] KOREL B. Automated software test data generation[J]. IEEE Transactions on Software Engineering, 1990, 16(8): 870-879.
- [21] 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 陆燕菲