一种基于改进遗传算法的路径测试用例生成方法

包晓安[1] 熊子健[1] 张 唯[1] 吴 彪[2] 张 娜[1]
(浙江理工大学信息学院 杭州 310018)[1] (山口大学东亚研究所研究生院 山口 753-8513)[2]

摘 要 采用遗传算法求解路径覆盖的测试用例生成问题是软件测试自动化的研究热点。针对传统标准遗传方法搜索测试用例易产生早熟收敛和收敛速度较慢的不足,设计了自适应的交叉算子和变异算子,提高了算法的全局寻优能力。基于动态生成算法框架,通过程序静态分析,考虑了分支嵌套深度的影响,结合层接近度和分支距离法,提出一种新的适应度函数。实验结果表明,该算法在面向路径的测试用例生成上优于传统方法,提高了测试效率。

关键词 软件测试,测试用例生成,遗传算法,适应度函数
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.08.031

Approach for Path-oriented Test Cases Generation Based on Improved Genetic Algorithm

BAO Xiao-an[1] XIONG Zi-jian[1] ZHANG Wei[1] WU Biao[2] ZHANG Na[1]
(School of Information Science and Technology, Zhejiang Sci-tech University, Hangzhou 310018, China)[1]
(The Graduate School of East Asian Studies, Yamaguchi University, Yamaguchi 753-8513, Japan)[2]

Abstract Using genetic algorithms to solve the problem of generating test cases for path coverage is a hot topic in software testing automation. In view of the problems in traditional standard genetic methods, such as premature convergence and slow search efficiency, this paper designed adaptive crossover operator and mutation operator, thus enhancing the global optimal capability of genetic algorithm. Meanwhile, a new fitness function was introduced to evaluate individuals based on dynamic generation algorithm framework, which combines approach level and branch distance and takes the nesting degree of branches into consideration to compute the fitness values of test data. The experimental results confirm that the proposed improved method is more efficient in generating test cases for path coverage compared with the traditional method.

Keywords Software testing, Test cases generation, Genetic algorithm, Fitness function

1 引言

软件测试是提高软件质量的重要保证。相对于功能测试而言,软件结构测试能更有效地发现软件缺陷,而大多数软件结构测试问题都可以转化为求解路径覆盖的测试用例生成问题[1]。该问题可以描述为:将待测程序给定的任意一条可执行路径作为目标路径,在程序数据输入的范围内,寻找至少一个能覆盖目标路径的数据作为测试用例。

运用启发式搜索算法进行软件结构测试用例生成已被证实为一种有效的方法,一些基于搜索的演化测试工具得以开发与应用,如 EvoSuite[2] 和 AgitarOne[3],其中采用遗传算法求解路径覆盖的测试用例生成问题是近年来许多学者的研究热点。薛云志等[4]提出了基于混乱遗传算法的测试用例生成方法,该方法将测试覆盖率作为测试输入集的评价函数来指导测试生成。Awedikan 等[5]提出了一种扩展分支距离的计算方法,利用遗传算法框架和变量间的依赖关系指导测试用例生成。Mahajan 等[6]和 Girgis 等[7]研究了基于结构导向技术的测试数据的自动生成,该方法使用遗传算法对数据流进行结构性分析,得到满足路径覆盖的测试用例。巩敦卫等[8]将被测程序表示成一颗二叉树,对目标路径进行赫夫曼编码表示,根据个体的进化过程中的穿越路径与目标路径的相似度来计算适应值,从而优化路径测试。但是,遗传算法作为一种基于自然选择的随机搜索算法,仍然存在早熟收敛、后期收敛速度慢等固有缺陷,而传统的标准遗传算法采用固定交叉率和变异率的进化算子,在进化后期,个体的趋同化程度越来越高,种群多样性急剧降低,不利于搜索全局最优的测试用例[19]。更重要的是,利用遗传算法生成测试用例时需要设计适应度函数来评估其优劣,而传统方法在构造以路径覆盖为目标的适应度函数时忽略了分支嵌套深度对个体评价的影响,阻碍了测试用例生成效率的提高。

到稿日期:2017-06-23 返修日期:2017-09-06 本文受国家自然科学基金(61502430,61379036,61562015),浙江理工大学 521 人才培养计划资助。

包晓安(1973-),男,硕士,教授,主要研究方向为自适应软件、软件测试与智能信息处理;**熊子健**(1993-),男,硕士,主要研究方向为软件工程、软件测试;**张 唯**(1994-),女,硕士,主要研究方向为软件测试、软件形式化方法;**吴 彪**(1989-),男,博士,主要研究方向为软件工程、软件测试;**张 娜**(1977-),女,硕士,副教授,主要研究方向为软件工程、软件测试,E-mail:zhangna@zstu.edu.cn(通信作者)。

本文针对以上不足，在传统遗传算法的基础上，利用海明距离计算个体相似度，以此来量化计算种群多样性的指标，进而设计自适应的交叉和变异算子，增强算法的搜索能力；在适应度函数构造方面，针对路径覆盖测试，借鉴前人的研究成果，结合层接近度和分支距离法，通过程序静态分析，考虑了分支嵌套深度对分支谓词的影响，动态调整个体的评价值，从而提高了搜索效率。

2　基于改进遗传算法的测试用例生成方法

2.1　测试用例生成模型

测试用例自动生成技术是实际软件开发中亟需解决的难题，学者们提出了很多解决的方法[9-11]，其中基于遗传算法来实现测试用例自动生成技术是较为常用且简单高效的一种方法。基于遗传算法的测试用例自动生成模型如图 1 所示。该模型由测试环境、遗传算法搜索、测试环境运行 3 个部分组成。在路径覆盖的软件结构测试中，测试环境部分根据测试需求（待覆盖的目标路径）对被测程序进行静态分析，这里以各分支谓词作为标志进行分支路径分析，从而实现程序插装；遗传算法搜索部分作为测试用例生成模型中的核心部分，在种群初始化后，将位串解码后的输入参数传递给测试驱动模块，测试运行部分通过当前参数值驱动程序，由插桩的被测程序评价参数值，并在个体评价后，经选择、交叉、变异遗传算子操作实现种群的不断进化，直到搜索到满足要求的最优解。

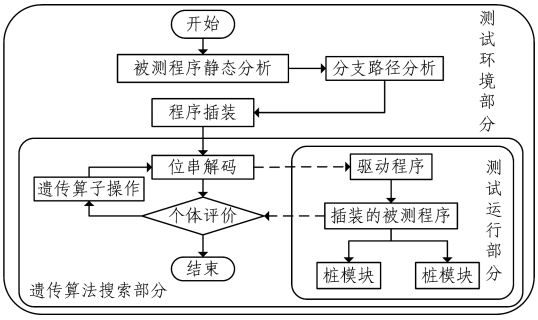

图 1　基于遗传算法的测试用例生成模型

Fig. 1　Generation model of test case based on genetic algorithm

2.2　适应度函数的设计

适应度函数的构造是遗传算法中最为关键的部分[12-13]，也即设计一个标准来判定一个可行解的优劣。许多学者针对路径覆盖的测试用例自动生成问题给出了不同的适应度函数构造方法[14-17]，其大致可以分为两类：层接近度法（approach_level）和分支距离法（branch_distance）。

层接近度法计算变量在测试执行中穿越路径与目标路径的接近程度，并将其作为适应度函数来引导种群个体进化[17]。假设第 i 个个体在某一次迭代进化的测试执行中穿越的路径为 $P(x_i)$，目标路径为 $P(target)$，则利用层接近度法计算个体的适应度值的公式为：

$$approach\_level(x_i)=\frac{\xi(x_i)}{|P(target)|} \tag{1}$$

其中，$\xi(x_i)$ 表示个体 x_i 穿越路径 $P(x_i)$ 所遍历的节点中相对于目标路径 $P(target)$ 未穿越节点的个数，$|P(target)|$ 表示目标路径遍历的节点个数。实际上，层接近度法本质上借鉴了距离计算的思想，特别地，当个体穿越的路径与目标路径重合，即当 $P(x_i)=P(target)$ 时，个体穿越的路径覆盖目标路径的所有节点，相对目标路径的未穿越节点的个数 $\xi(x_i)$ 为 0，层接近度 $approach\_level(x_i)$ 的计算值为 0，表征个体穿越路径与目标路径无限接近，距离为 0。

然而，利用层接近度构造适应度函数仍然存在不足，即其未能考虑分支谓词对测试路径的影响，随着分支取真值与取假值的不同，某些路径的覆盖将随着迭代次数的增加而减少，这不利于对全局最优解的寻找，生成的大部分测试数据很可能会陷入局部最优解。

分支距离法通过判断个体驱动程序的执行轨迹与目标路径各分支的偏离程度来计算个体适应度。本文在考虑分支条件对适应度函数的影响时，结合 Tracey 等[14]的工作，采用分支距离函数插装的方法在测试用例驱动程序执行时获得必要的覆盖路径信息和分支条件执行信息。该方法在测试路径所经过的第 i 个分支谓词 P_i 前插入分支距离函数（branch-distance function），即 $f(P_i)$。当分支谓词为真时，$f(P_i)=0$；当分支谓词为假时，分支距离函数 $f(P_i)$ 的构造方法如表 1 所列。其中，K 表示一个正常数，目的是使目标函数总能够返回一个非负值，保证计算有意义。

表 1　几种典型的分支距离函数

Table 1　Branch distance functions for several kinds of branch predicates

No.	Branch predicate	Branch distance function $f(P_i)$
1	$a=b$	If $\|a-b\|=0$ then 0 else $\|a-b\|+K$
2	$a!=b$	If $\|a-b\|\neq 0$ then 0 else K
3	$a<b$	If $a-b<0$ then 0 else $(a-b)+K$
4	$a\leqslant b$	If $a-b\leqslant 0$ then 0 else $(a-b)+K$
5	$a>b$	If $b-a<0$ then 0 else $(b-a)+K$
6	$a\geqslant b$	If $b-a\leqslant 0$ then 0 else $(b-a)+K$
7	$a\cap b$	$f(a)+f(b)$
8	$a\cup b$	$\min(f(a),f(b))$
9	$!a$	negation is propagated over a
10	boolean	If true then 0 else K

基于上述分支距离表示，结合层接近度，对于给定目标路径 T 的结构测试，评价个体的适应度函数可以构造为：

$$Fitness(\boldsymbol{x},t)=\frac{1}{\varepsilon+approach\_level(x)+\sum_{i=1}^{l}f(P_i)} \tag{2}$$

其中，$\boldsymbol{x}=(x_1,x_2,\cdots,x_{len})$ 是针对被测程序由一串输入变量级联后的向量，len 为构成一次完整输入的变量个数，l 为被测程序针对给定目标路径 t 的分支判定数目，$f(P_i)$ 为第 i 个分支的分支距离，ε 是一个极小常量以保证分母不为 0。在本文的实验中，设 ε 为 0.01。

当然，利用式（2）直接评价个体适应度仍然存在两点不足：1）分支距离的大小与输入数据密切相关，其计算值可能远大于层接近度的计算值；2）式（2）未能考虑嵌套深度（Nesting Degree，ND）对分支谓词的影响，将目标路径所有分支权重的影响值都简化为 1。因此针对以上不足，本文在式（2）的基础上加以改进，一方面对分支距离进行规范化处理，另一方面结合分支嵌套深度来调整不同分支的权重。改进后的适应度函数为：

$$Fitness(\boldsymbol{x},t)=\frac{1}{\varepsilon+approach\_level(x)+\sum_{i=1}^{l}w_i\cdot normalize(f(P_i))} \tag{3}$$

其中，$normalize(d)=1-1.001^{-d}$，目的在于将分支距离规范到[0,1)内。而分支权重 w_i 可按式(4)计算：

$$w_i=\frac{nd_i}{\sum_{i=1}^{l}nd_i} \tag{4}$$

分支权重的意义在于，对于分支 $P_i(1\leqslant i\leqslant s)$，通过程序静态分析获知其嵌套深度 nd_i。分支 P_i 的嵌套深度越高，生成测试用例覆盖该分支的可达性就越低，其分支的嵌套权重相对越大，即该分支对实现目标路径完全覆盖的重要度越高，对计算个体评价值的影响越大，因此对适应度调整的影响就得到相应提升。图 2 为某一程序的控制流图，假设以经过分支判断节点的所有真分支为目标路径，即 $P(target)=$“$s\rightarrow 1\rightarrow 2\rightarrow 3\rightarrow 4\rightarrow 8\rightarrow e$”。下面举例计算各分支节点插装分支距离函数后的权重。

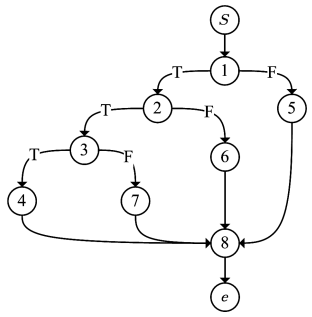

图 2 分支节点权重计算示例的程序控制流图

Fig. 2 Control flowchart of calculating example for branch node weight

通过程序静态分析，目标路径分支节点 node1，node2，node3 的分支嵌套深度分别为 1，2，3，目标路径分支判定数目 $l=3$，最大嵌套深度 $nd_{\max}=3$。根据式(4)，可以计算出各分支节点权重为：

$$\text{node1}:w_1=\frac{1}{1+2+3}=\frac{1}{6}$$

$$\text{node2}:w_2=\frac{2}{1+2+3}=\frac{2}{6}=\frac{1}{3}$$

$$\text{node3}:w_3=\frac{3}{1+2+3}=\frac{3}{6}=\frac{1}{2}$$

2.3 遗传算子的设计

将遗传算法应用于测试生成的情境中，需要解决的一个关键问题就是避免早熟收敛而陷入局部最优解。已有相关研究表明：种群的进化过程建立在一定的种群多样性的基础上，早熟收敛与个体趋同性逐代增加、后期种群多样性降低有关。本文借助海明距离的概念，提出一种基于个体相似度的种群多样性度量函数，进而设计自适应的交叉和变异算子，以增强算法的搜索能力。

1) 种群多样性的描述

定义 1(种群个体相似度(Similarity Among Individuals，SAI)) 种群个体相似度用于定量描述种群个体基因间的相似性，本文利用海明距离来刻画该相似值。

定义两个个体间的海明距离为：

$$d_{ij}=\|x_i-x_j\|=\sqrt{\sum_{k=1}^{n}(x_{ik}-x_{jk})^2} \tag{5}$$

其中，$i,j=1,2,\cdots,N$ 且 $i\neq j$，x_i 表示第 i 个输入变量，x_{ik} 表示第 i 个输入变量的二进制串中的第 k 个符号(“0”或者“1”)，n 表示二进制串的长度。

此时，种群的海明距离可以表示为：

$$D_{SAI}=\sum_{k=1}^{n}\sum_{j=1}^{n}d_{ij} \tag{6}$$

定义 2(种群多样性(Population Diversity，PD)) 种群多样性用于描述一个种群中个体基因的多样性和广泛性，通过个体之间的海明距离和适应值的方差来度量。

种群中个体的平均适应度为：

$$f_{\text{avg}}=\frac{1}{M}\sum_{i=1}^{M}f_i \tag{7}$$

此时，种群的适应值多样性可以表示为：

$$F=\frac{1}{M}\sum_{i=1}^{M}(f_i-f_{\text{avg}})^2 \tag{8}$$

其中，f_i 为种群中第 i 个个体的适应值，M 为该种群的个体总数。

最后，种群的多样性可计算如下：

$$V_{PD}=\rho\cdot D_{SAI}+(1-\rho)\cdot F \tag{9}$$

其中，$\rho(0\leqslant\rho\leqslant 1)$ 为权值参数。本文实验采用均衡策略将 ρ 设为 0.5。

2) 自适应交叉算子和变异算子

①交叉策略

交叉率 P_c 的设置是种群多样性的保证，本文采用单点交叉策略，自适应地选取交叉概率 P_c，使得适应度低于种群适应度平均值的较差个体以较大的恒定概率 P_{c0} 交叉；而适应值大于或等于平均值的较优个体通过种群多样性函数 V_{PD} 动态调整交叉率，当种群多样性较低时增大交叉率，当种群多样性较高时减小交叉率。假设经过选择算子操作后的双亲个体为 x_i 和 x_j，则它们发生交叉的自适应概率为：

$$P_c=\begin{cases}P_{c0}(1-normalize(V_{PD})), & f'\geqslant f_{\text{avg}}\\ P_{c0}, & f'<f_{\text{avg}}\end{cases} \tag{10}$$

其中，$normalize$ 为规范化函数，即 $normalize(d)=1-1.001^{-d}$；$f_{\text{avg}}$ 是当前种群中的平均适应度值；f' 是考虑交叉的两个适应度较大的个体的适应度值。

②变异策略

变异操作是遗传算法中另外的一个重要算子。本文采用二进制基本位变异，并设计自适应变异概率 P_m，使适应度值低于平均值的个体以较大的恒定概率 P_{m0} 变异；适应度值大于平均值的个体根据种群多样性动态调整变异概率，即当种群多样性值 V_{PD} 较小时，增大变异率，以保证种群多样性，当种群多样性 V_{PD} 较大时，相应减小变异率，以保证个体的优秀基因不被破坏。

$$P_m=\begin{cases}P_{m0}(1-normalize(V_{PD})), & f_i\geqslant f_{\text{avg}}\\ P_{m0}, & f_i<f_{\text{avg}}\end{cases} \tag{11}$$

其中，$normalize$ 为规范化函数，即 $normalize(d)=1-1.001^{-d}$；$f_{\text{avg}}$ 是当前种群中的平均适应度值；f_i 是考虑交叉的两个适应度较大的个体的适应度值。

2.4 TIGA 算法步骤

算法 1 给出了本文基于改进遗传算法的测试用例进化生

成(TIGA)的伪代码。

算法 1　基于改进遗传算法的测试用例进化生成算法

输入：插装后的被测程序(PUT)，被测程序的控制流图(CDG)，目标路径 P(target)，种群大小 M_k，终止准则 SC，计算种群多样性的权重参数 ρ，交叉算子$\tilde{c}$的交叉率 p_c，变异算子$\tilde{m}$的变异率 p_m

输出：测试用例解集 s_t

```
1. s0←GenerateRandomSolutions(Mk)//种群初始化
2. t=0
3. while !SC do
4.    Evaluate(f,st) //根据式(3)计算第 t 代种群的个体适应度值
5.    parentst= SelectParents(s̃,st)
6.    Compute VPD(ρ) //根据式(9)计算种群多样性
7.    Update pC//根据式(10)动态调整交叉率
8.    chlidrent=Recombination(c̃,pc)
9.    Construct new population st′ fro mparentst and offspring
10.   Update pm//根据式(11)动态调整变异率
11.   Mutate st′(m̃,pm)
12.      st←st′
13.      t=t+1
14. end while
15. return st
```

种群将根据遗传算法策略不断进化，直到出现最优解。算法 1 的迭代终止准则 SC 采用两个终止条件：

1)记录生成的测试用例执行 PUT 所覆盖的路径轨迹，若目标路径的所有节点被覆盖，*approach_level* 的计算值为 0，说明找到最优解，算法结束；

2) 由于目标路径可能存在难以覆盖的节点，故设定搜索的最大进化代数，当运行到预先设定的最大迭代次数时，算法终止。

3　实验分析

为了验证本文方法(TIGA)在路径结构测试生成测试用例的有效性，选取了一组基准被测程序用于实验，并从算法的进化代数、搜索时间等方面同文献[16]的方法(该方法采用简单遗传算法，并使用层接近度和分支距离结合的适应度函数，但未考虑嵌套深度的权重影响，记作 TSGA 方法)进行了实验比较。

参数设置如下：每种方法在各组实验中采用相同的初始化种群，个体均采用二进制编码，采用轮盘赌选择、单点交叉和单点变异。由于遗传算法中没有恒定的最佳参数取值，一般取较大的交叉率以便产生新的个体，取较小的变异率防止破坏优良个体，这里借鉴 Schaffer 等[20]的工作将交叉率设置为 0.9，变异率设置为 0.2(由于本文方法采用自适应的交叉率和变异率，因此 p_{c0} 和 p_{m0} 分别对应传统方法的交叉率和变异率)。算法终止准则设定为成功生成穿越目标路径的测试用例或者达到预先设定的最大迭代次数。为了避免随机性带来的不利影响，每组实验的不同方法运行 50 次。

3.1　三角形分类程序实验

三角形分类程序由于具有清晰但又比较复杂的逻辑结构，常被用来作为软件测试的基准程序进行实验研究。它被描述为将输入程序的 3 个数作为三角形的 3 条边进行分类。本文首先选取实现覆盖等边三角形的路径作为目标路径进行测试用例的生成。实验记录了算法在不同数据范围内搜索目标路径的进化代数，以 50 次进化代数的平均值作为性能分析指标。另外，由于实验每次生成测试数据的搜索时间只有几毫秒，因此实验记录的运行时间为每组实验的总搜索时间。实验结果如表 2 和表 3 所列。

表 2　三角形分类程序平均进化代数的实验结果

Table 2　Experimental results of average evloutionary generation of triangle classification program

数据范围 $[0,N]^3$	种群规模	平均进化代数		
		TIGA	TSGA	代数比/%
$[0,128]^3$	30	21.2	63.8	33.2
$[0,256]^3$	50	33.5	71.4	46.9
$[0,512]^3$	80	44.3	211.2	20.9
$[0,1024]^3$	100	68.5	428.7	15.9

表 3　三角形分类程序总进化时间的实验结果

Table 3　Experimental results of total evolutionary time of triangle classification program

数据范围 $[0,N]^3$	种群规模	总搜索时间		
		TIGA/s	TSGA/s	时间比/%
$[0,128]^3$	30	0.079	0.164	48.1
$[0,256]^3$	50	0.086	0.255	33.7
$[0,512]^3$	80	0.335	1.606	20.8
$[0,1024]^3$	100	0.634	3.607	17.5

从表 2 和表 3 可以看出：随着数据输入范围的倍数化增长，两种方法搜索目标路径覆盖的成本不断提高，但 TIGA 的进化代数和运行时间均少于 TSGA；且在较大的数据范围内，TIGA 的优势更加明显，如在$[0,512]^3$ 和$[0,1024]^3$ 的数据输入范围内，TIGA 较 TSGA 的时间比分别为 20.8 和 17.5，在$[0,1024]^3$ 的范围内，两种方法的平均进化代数分别为 68.5 和 428.7，不到 TSGA 的 1/6。

另外，本文针对三角形分类程序，以输入数据范围$[0,1024]^3$ 为例，记录实现覆盖 PATH1(非三角形)，PATH2(普通三角形)，PATH3(等腰但不等边的三角形)，PATH4(等边三角形)4 条不同目标路径所需的进化代数。每组实验运行 50 次取平均值，结果如表 4 所列。

表 4　覆盖不同目标路径的平均进化代数

Table 4　Results of average evolutionary generation for covering different target paths

	PATH1	PATH2	PATH3	PATH4
TIGA	1	1	4.5	21.2
TSGA	1	1	6.3	63.87

从表 4 可以看出：1)TIGA 和 TSGA 在覆盖 PATH1(非三角形)和 PATH2(普通三角形)的测试用例生成时，平均进化代数相同，即都是平均进化 1 代就可以实现穿越目标路径；2)针对 PATH3(等腰但非等边的三角形)的目标路径覆盖，两种方法的平均进化代数相当，且 TIGA 较 TSGA 有轻微优势；3)针对较难覆盖的 PATH4(等边三角形)，TIGA 在 50 次实验中平均以 21.2 代实现穿越目标路径的测试用例生成，而 TSGA 则需要 63.8 代，约为 TIGA 的 3 倍，这说明对于搜寻难以覆盖的目标路径的测试用例，TIGA 在进化代数上较 TSGA 具有明显的优势。

3.2 工业用例实验

为了进一步验证 TIGA 在路径测试用例生成上的有效性，选取了 4 个工业用例程序进行实验，这些程序被广泛应用于验证软件结构测试的测试生成[18]。本文针对每个程序随机选择一条可行路径作为目标路径，被测程序的相关描述信息如表 5 所列。

表 5 4 个工业用例被测程序的相关描述信息

Table 5 Related description information of four industrial case tested programs

被测程序	代码行数	目标路径的节点个数	变量个数	输入范围
sed	25	6	3	$[0,1024]^3$
interserve	52	10	4	$[0,512]^4$
totinfo	142	30	6	$[0,32]^6$
spice	215	64	9	$[0,128]^9$

本组实验中，设定每种方法分别独立运行 50 次。对于程序 sed 和 interserve，设种群大小为 30，最大运行代数为 500；对于程序 totinfo 和 spice，设种群大小为 50，最大运行代数为 2000。实验统计进化代数的平均值及测试用例生成的总进化时间，结果如表 6 所列。

表 6 工业用例程序进化代数及进化时间的实验结果

Table 6 Experimental results of evolutionary generation and evolutionary time of industrial case programs

被测程序	平均进化代数		总搜索时间/s	
	TIGA	TSGA	TIGA	TSGA
sed	34.2	216.3	0.883	1.061
interserve	69.9	412.7	0.147	0.053
totinfo	47.5	1318.6	0.785	2.294
spice	116.3	1912.2	0.981	4.722

由表 6 可以看出：1)从平均进化代数来看，TIGA 生成覆盖目标路径的测试用例所用的进化代数明显少于 TSGA。2)从程序运行的总进化时间来看，对于选取的 4 个工业用例程序，TIGA 生成覆盖目标路径的测试用例所需的总进化时间分别为 0.883 s，0.147 s，0.785 s 和 0.981 s，分别小于 TSGA 的 1.061 s，0.053 s，2.294 s 和 4.722 s，这说明相对于 TSGA，TIGA 能够以较少的耗时生成覆盖目标路径的测试用例。

综合上述分析，对于路径覆盖的测试用例生成问题，本文提出的改进算法(TIGA)相对于传统遗传算法(TSGA)在进化代数和算法的执行时间上具有一定优势。

结束语 采用合适的方法求解面向路径的测试用例生成问题，是软件结构测试的重要内容。本文基于遗传算法的测试用例生成框架，综合层接近度和分支距离法，考虑分支嵌套深度对分支权重的影响，构造了一种新的适应度函数。另外，在遗传算子方面，通过引入种群多样性函数，设计自适应交叉和变异算子来进一步优化算法。实验通过三角形分类基准程序和 4 个工业用例验证了本文方法较传统方法在路径覆盖的测试用例生成上的优越性。值得说明的是，本文采用控制流覆盖准则进行单一路径的测试生成研究，今后将结合数据流覆盖准则研究多目标路径覆盖的测试用例生成，以增强测试的充分性，提高测试的效率。

参 考 文 献

[1] SAGARNA R, LOZANO J A. Scatter search in software testing, comparison and collaboration with estimation of distribution algorithms[J]. European Journal of Operational Research, 2006, 169(2): 392-412.

[2] FRASER G, ARCURI A. EvoSuite at the SBST 2015 Tool Competition[C]// IEEE/ACM, International Workshop on Search-Based Software Testing. IEEE, 2015: 25-27.

[3] GALLER S J, AICHERNIG B K. Survey on test data generation tools[J]. International Journal on Software Tools for Technology Transfer, 2014, 16(6): 727-751.

[4] XUE Y Z, CHEN W, WANG Y J, et al. An automated approach for structural test data generation based on Messy GA[J]. Journal of Software, 2006, 17(8): 1688-1697. (in Chinese)
薛云志，陈伟，王永吉，等. 一种基于 Messy GA 的结构测试数据自动生成方法[J]. 软件学报，2006，17(8)：1688-1697.

[5] AWEDIKIAN Z, AYARI K, ANTONIOL G. MC/DC automatic test input data generation[C]// Genetic and Evolutionary Computation Conference(GECCO 2009). 2009: 1657-1664.

[6] MAHAJAN M, PORWAL R. Applying genetic algorithm to increase the efficiency of a data flow-based test data generation approach[J]. Acm Sigsoft Software Engineering Notes, 2012, 37(5): 1-5.

[7] GIRGIS M R, GHIDUK A S, ABDELKAWY E H. Automatic Generation of Data Flow Test Paths using a Genetic Algorithm [J]. International Journal of Computer Applications, 2014, 89(12): 29-36.

[8] GONG D W, ZHANG Y. Novel evolutionary generation approach to test data for multiple paths coverage[J]. Acta Electronica Sinica, 2010, 38(6): 1299-1304. (in Chinese)
巩敦卫，张岩. 一种新的多路径覆盖测试数据进化生成方法[J]. 电子学报，2010，38(6)：1299-1304.

[9] XIE X Y, XU B W, SHI L, et al. Genetic test case generation for path-oriented testing[J]. Journal of Software, 2009, 20(12): 3117-3136. (in Chinese)
谢晓园，徐宝文，史亮，等. 面向路径覆盖的演化测试用例生成技术[J]. 软件学报，2009，20(12)：3117-3136.

[10] RAJKUMARI M R, GEETHA B G. Automatic test data generation using genetic algorithm and program dependence graph [J]. Journal of Computer Applications, 2012, 48(7): 586-605.

[11] ALSHRAIDEH M A, MAHAFZAH B A, SALMAN H S E, et al. Using Genetic Algorithm as Test Data Generator for Stored PL/SQL Program Units[J]. Journal of Software Engineering & Applications, 2015, 6(2): 65-73.

[12] RAUF A, JAFFAR A, SHAHID A A. Fully automated gui testing and coverage analysis using genetic algorithms[J]. International Journal of Innovative Computing Information & Control Ijicic, 2011, 7(6): 3281-3294.

[13] SHI J J, JIANG S J. Automatic test data generation tool of dynamic variable parameters based on genetic algorithm[J]. Computer Science, 2012, 39(5): 124-127. (in Chinese)
史娇娇，姜淑娟. 基于遗传算法的动态可变参数的测试数据自动生成工具[J]. 计算机科学，2012，39(5)：124-127.

[14] TRACEY N, CLARK J, MANDER K, et al. An Automated Framework for Structural Test-Data Generation[C]// IEEE International Conference on Automated Software Engineering, 1998. IEEE, 1998: 285-288.

(下转第 190 页)

在图 2 中，保持类内阈值与传统单一阈值相等，逐渐减小类间阈值。可以发现，4 个数据集的邻域近似质量单调增加，而邻域条件熵单调减小，即知识的不确定性下降。另外，随着属性数量的逐渐增多，4 个数据集的邻域近似质量和邻域条件熵并没有呈现出单调性变化，这是由于在属性变化下采用了多次归一化的实验方法。通过观察图 2 发现，监督邻域粗糙集在粒度变化情况下可以通过保持类内阈值不变而调整类间阈值来实现对邻域粒化效果和信息不确定性的有效控制。

由图 3 可知，在不同的类内阈值下，逐渐减小类间阈值。可以发现，随着类间阈值的减小，4 个数据集的邻域近似质量呈现单调增加的趋势，而邻域条件熵呈现单调减小的趋势，知识不确定性也有所下降。另外，随着类内阈值的增大，4 个数据集的邻域近似质量单调减小，而邻域条件熵单调增加。通过图 3 可以发现，在数据集不变的情况下，通过保持类内阈值不变而调整类间阈值也能实现对邻域粒化效果和信息不确定性的有效控制。

结束语 传统的邻域粗糙集使用单一邻域阈值，大大限制了对论域粒化以及降低信息不确定性的能力。本文借助监督学习中对象已有或预测的标记信息，引入类内阈值和类间阈值的概念，提出了一种监督邻域粗糙集。该模型是传统邻域粗糙集的推广形式，可以通过保持类内邻域不变和减小类间邻域来优化邻域粒子群。实验结果显示，该模型在粗糙近似质量和条件信息熵两种度量下均能降低信息的不确定性。下一步工作将在监督邻域系统下对属性约简开展深入探讨和研究。

参 考 文 献

[1] PAWLAK Z. Rough sets[J]. International Journal of Computer & Information Sciences, 1982, 11(5): 341-356.

[2] LIN T Y. Granular computing: practices, theories, and future directions [M]. New York: Springer, 2009: 4339-4355.

[3] LIN T Y. Granular computing on binary relation I: Date mining and neighborhood systems [J]. Rough Sets in Knowledge Discovery, 1998(2): 165-166.

[4] YAO Y Y. Relational interpretation of neighborhood operators and rough set approximation operators [J]. Information Sciences, 1998, 111(198): 239-259.

[5] WU W Z, ZHANG W X. Neighborhood operator systems and approximations [J]. Information Sciences, 2002, 144(1-4): 201-217.

[6] HU Q H, YU D R, XIE Z X. Neighborhood classifiers[J]. Expert Systems with Applications, 2008, 34(2): 866-876.

[7] HU Q H, PEDRYCZ W, YU D R, et al. Selecting discrete and continuous featuresbased on neighborhood decision error minimization [J]. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2010, 40(1): 137-150.

[8] DUAN J, HU Q H, ZHANG L J, et al. Feature selection for multi-label classification based on neighborhood rough sets[J]. Journal of Computer Research and Development, 2015, 52(1): 56-65.

[9] CHEN D G, LI W L, ZHANG X, et al. Evidence-theory-based numerical algorithms of attribute reduction with neighborhood-covering rough sets[J]. International Journal of Approximate Reasoning, 2014, 55(3): 908-923.

[10] YANG X B, ZHANG M, DOU H L, et al. Neighborhood Systems-Based Rough Sets in Incomplete Information System[J]. Knowledge-Based Systems, 2011, 24(6): 858-867.

[11] YANG X B, CHEN Z H, DOU H L, et al. Neighborhood system based rough set: models and attribute reductions[J]. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 2012, 20(3): 399-419.

[12] ZHANG J B, LI T R, RUAN D, et al. Neighborhood rough sets for dynamic data mining [J]. International Journal of Intelligent Systems, 2012, 27(4): 317-342.

[13] CHEN H M, LI T R, CAI Y, et al. Parallel attribute reduction in dominance-based neighborhood rough set[J]. Information Sciences, 2016, 373: 351-368.

[14] LIANG J Y, SHI Z, LI D, et al. Information entropy, rough entropy and knowledge granulation in incomplete information systems [J]. International Journal of General Systems, 2006, 35(6): 641-654.

[15] HU Q H, GUO M Z, YU D R, et al. Information entropy for ordinal classification [J]. Science China Information Sciences, 2010, 53(6): 1188-1200.

(上接第 178 页)

[15] NIRPAL P B, KALE K V. Using Genetic Algorithm for Automated Efficient Software Test Case Generation for Path Testing [J]. International Journal of Advanced Networking & Applications, 2011, 2(6): 911-915.

[16] MCMINN P. Evolutionary Search for Test Data in the Presence of State Behaviour[J]. University of Sheffield, 2005, 16(12): 41-46.

[17] PACHAURI A, SRIVASTAVA G. Automated test data generation for branch testing using genetic algorithm: An improved approach using branch ordering, memory and elitism[J]. Journal of Systems & Software, 2013, 86(5): 1191-1208.

[18] 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.

[19] ALETI A, GRUNSKE L. Test data generation with a Kalman filter-based adaptive genetic algorithm[J]. Journal of Systems & Software, 2015, 103(C): 343-352.

[20] SCHAFFER J D, CARUANA R A, ESHELMAN L J, et al. A study of control parameters affecting online performance of genetic algorithms for function optimization[C] // International Conference on Genetic Algorithms, George Mason University, Fairfax, Virginia, USA. DBLP, 1989: 51-60.