基于稀有数据扑捉的路径覆盖测试数据进化生成方法

张 岩[1),2)] 巩敦卫[1)]

[1)](中国矿业大学信息与电气工程学院 江苏 徐州 221116)

[2)](牡丹江师范学院工学院 黑龙江 牡丹江 157012)

摘 要 采用遗传算法自动生成路径覆盖的测试数据是软件测试自动化研究的热点.现有方法设计适应值函数时,对穿越难以覆盖节点的稀有数据保护不够理想,因而影响测试数据生成效率的提高.文中在测试数据进化生成时动态扑捉稀有数据,通过统计每代种群中目标路径各节点被穿越的个体数量,得到个体对生成穿越目标路径测试数据的贡献,以此作为权重调整个体的适应值,使得稀有数据的适应值增加,以便在后续进化中得到保留,从而提高测试数据生成的效率.基准程序和工业用例的测试结果表明,与传统方法及随机法比较,文中方法生成覆盖路径的测试数据效率较高.

关键词 软件测试;路径覆盖;遗传算法;稀有数据;适应值调整

中图法分类号 TP301 **DOI号** 10.3724/SP.J.1016.2013.02429

Evolutionary Generation of Test Data for Paths Coverage Based on Scarce Data Capturing

ZHANG Yan[1),2)] GONG Dun-Wei[1)]

[1)](*School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu* 221116)

[2)](*School of Technology, Mudanjiang Normal University, Mudanjiang, Heilongjiang* 157012)

Abstract Using genetic algorithms to generate test data for path coverage is a hot topic in software testing automation. The established fitness functions of previous methods cannot provide adequate protection to a scarce datum which covers a node difficult to be covered, so the efficiency of generating test data needs to be improved. In this study, scarce data are dynamically captured during the evolutionary generation of test data. We obtain the contribution of an individual by counting up the number of individuals which traverse each node of the target path, and regard this contribution as a weight to adjust the fitness of the individual. In this way, the fitness of a scarce datum can be increased and the scarce datum can be kept in the subsequent evolution, so the efficiency of generating test data is improved. The proposed method is applied to generate test data for covering paths of two benchmark and six industrial programs, and is compared with traditional and random methods. The experimental results confirm that the proposed method is efficient in generating test data for path coverage.

Keywords software testing; path coverage; genetic algorithms; scarce data; fitness adjustment

收稿日期:2011-11-25;最终修改稿收到日期:2013-09-29.本课题得到国家自然科学基金(61075061)、黑龙江省高校青年学术骨干支持计划项目(1252G063)、江苏省自然科学基金(BK2012566,BK2010187)、高等学校博士学科点专项科研基金(20100095110006)、中国矿业大学校优秀创新团队建设专项基金(2011ZCX002)、牡丹江市科学技术计划项目(Z2013s043)以及牡丹江师范学院重点创新预研项目(SY201216)资助.**张 岩**,女,1972年生,博士研究生,副教授,中国计算机学会(CCF)会员,主要研究方向为复杂软件的测试数据生成.E-mail:zhangyancumt@126.com.**巩敦卫**,男,1970年生,博士,教授,博士生导师,主要研究领域为基于搜索的软件工程、进化计算及智能控制.

1 引 言

软件测试能有效保障软件的质量，按照测试过程的不同，分为静态测试和动态测试[1]，都是为了发现程序代码可能存在的缺陷，但前者不执行程序代码，而后者则基于测试数据运行程序代码，测试数据的自动生成是该类测试的关键. 按照生成测试数据依据的信息来源，动态测试又被分为黑盒测试、白盒测试以及二者融合的测试.

路径覆盖测试属于白盒测试，它要求在测试过程中尽可能覆盖程序所有能够达到的路径. 单锦辉等人认为，许多软件测试问题都可以归结为路径覆盖测试数据生成问题[2]. 该问题描述为：对于被测程序的任意一条目标路径，在程序的输入数据范围中寻找一个测试数据，保证以该数据作为输入，所经过的路径为目标路径. 自动求解上述问题将有效缩短软件测试的时间，提高测试效率，节约软件开发的成本.

遗传算法是求解路径覆盖测试数据自动生成问题的有效方法，近年来国内外的相关研究成果较多. Ahmed 和 Hermadi[3]、Bueno 和 Jino[4]、Lin 和 Yeh[5]、Watkins 和 Hufnagel[6]、Malhotra 和 Garg[7] 以及 Irfan 和 Ranjan[8] 等都利用遗传算法得到满足路径覆盖的测试数据；Wegener 等人[9] 给出多种路径覆盖准则，谢晓园等人[10] 提出了基于相似性度量的适应值构造方法.

众所周知，遗传算法通过适应值评价个体的优劣，进而引导算法搜索最优解，适应值函数的设计直接影响算法找到最优解的速度. 现有的适应值函数设计方法都只考虑单一个体满足目标路径的程度，没有有效利用进化种群所反映出的综合信息，即没有考虑个体穿越的节点是否为目标路径难以覆盖的节点，因此没有给予穿越难以覆盖节点的个体较高的适应值，从而使穿越难以覆盖节点的稀有数据没有得到有效保护；尤其对于大型复杂程序，目标路径的节点个数较多，或是在搜索空间爆炸性增长时，这种问题尤为突出. 如果采用适当的方法对穿越难以覆盖节点的稀有数据进行正确判断，并通过调整适应值对其进行保护，将能有效提高测试数据的生成效率.

目标路径节点被覆盖的难易程度可通过静态分析得到，但是会花费较多的分析时间，而且容易出现错报和漏报等现象. 使用遗传算法生成测试数据时，多个个体同时并行搜索最优解，进化中的每一代都会以这些个体对应的测试数据运行被测程序，目标路径节点被穿越的个体数量直接反映了该节点被覆盖的难易程度，即难以覆盖的节点仅有较少个体穿越；相反，容易覆盖的节点却有较多个体穿越. 因此，通过统计这些个体穿越目标路径节点的情况，即可得到节点被覆盖的难易程度.

本文通过统计测试数据进化生成中个体穿越目标路径的节点情况，评价个体对生成穿越目标路径测试数据的贡献(简称个体贡献度)，并用一个精确值表示，以此作为权重调整个体的适应值，使得穿越难以覆盖节点的个体获得较高的适应值，从而在进化中得到保留，以提高测试数据生成的效率. 将本文的方法用于基准程序和复杂的工业用例的测试，结果表明，本文方法生成测试数据的效率较高.

2 常见适应值函数设计方法

为了介绍方便，先给出几个基本概念. 记被测程序为 F，与 F 有关的概念如下：

(1) 节点. F 的一个基本执行单元称为节点，其在 F 的任何一次运行中，要么都执行，要么都不执行. 节点可能是分支语句的前件、一条语句或多条连续语句.

如图 1(a)为文献[11]用到的三角形分类程序，语句左边的标号为节点编号. 节点 1、3、5、7 等为分支语句的前件，节点 8、11、12 等只包含 1 条语句，节点 2、4、6 等由多条连续的语句构成.

(2) 控制流图[1]. F 的控制流图是一有向图 $G=(N,E,s,e)$，其中，N 是节点集，边集 $E=\{\langle node_i, node_j\rangle \mid node_i, node_j \in N$，且 $node_i$ 执行后可能立即执行 $node_j\}$，s 和 e 分别为程序的入口和出口节点.

如图 1(b)为图 1(a)所示的三角形分类程序的控制流图.

(3) 路径[12]. F 的路径 p 是控制流图的节点序列“$s, node_1, node_2, \cdots, node_n, e$”，其中，$node_i \in N$ $(i=1,2,\cdots,n)$，且 $\langle s, node_1\rangle \in E$，$\langle node_i, node_{i+1}\rangle \in E(i=1,\cdots,n-1)$，$\langle node_n, e\rangle \in E$.

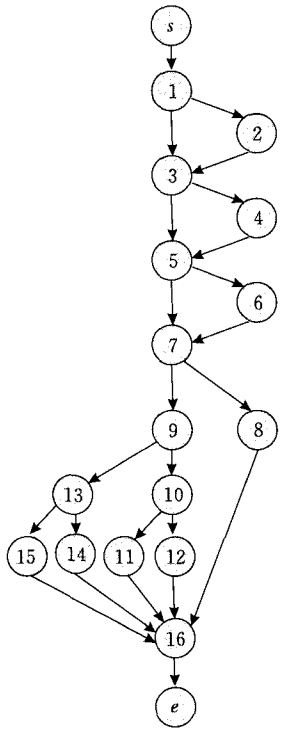
图 1(b)的“$s,1,2,3,4,5,6,7,9,13,15,16,e$”为三角形分类程序的一条路径.

用遗传算法生成覆盖目标路径的测试数据，需要把该问题转化为一个优化问题，适应值函数的设计是求解该问题的关键，因此众多学者针对适应值函数的构造给出多种方法[13]，大体分为分支距离

(branch_distance)、层接近度(approach_level)以及二者结合的方法.

```
void trangle(int a,int b,int c)
  {
    int type, t;
1. if(a>=b)
   {
2          t=a; a=b; b=t;
   }
3  if(a>=c)
   {
4          t=a; a=c; c=t;
   }
5  if(b>=c)
   {
6          t=b; b=c; c=t;
   }
7  if(a+b<=c)
8          type=4;
    else
   {
9          if(a==b)
           {
10            if(b==c)
11                   type=1;
              else
12                   type=2;
           }
           else
           {
13            if(b==c)
14                   type=2;
              else
15                   type=3;
           }
   }
16 return type;
}
```

(a) 源程序



(b) 控制流图

图 1　三角形分类程序及其控制流图[11]

2.1　分支距离计算方法

分支距离是当个体执行路径偏离目标分支时反映偏离该分支的大小. Korel 采用改变变量方法生成测试数据时，给出一种分支距离的计算方法，详见文献[12]. 如图 1(a)的程序中，假定其第 1 个分支节点的谓词($a>=b$)的真分支为目标分支，该方法的分支距离函数为 $b-a$，对于第 t 代的第 i 个个体 x_i($a=10, b=20, c=30$)的分支距离为 $20-10=10$，该值越小说明个体 x_i 越接近该目标分支；当有多个目标分支没有达到时，将这些分支的分支距离之和作为个体 x_i 的相对于目标路径的分支距离.

Tracey 进一步给出各分支谓词的分支距离计算函数及针对含有复合谓词情况的计算方法，其方法与 Korel 提出的方法相似，只是在谓词计算结果为假值时多使用一个正常数 K，使目标函数总是返回一个大于或等于 0 的值，如果违背期望目标分支，则目标函数值大于 0，而当执行了目标分支时，该值等于 0，详见文献[14].

2.2　层接近度计算方法

层接近度表示输入变量所穿越路径与目标路径的接近程度. 假设第 t 代的第 i 个个体 x_i 穿越的路径为 $p(x_i)$，目标路径为 p_0，其节点个数表示为 $|p_0|$，比较 $p(x_i)$ 与 p_0 的各个节点，将层接近度记为 $approach\_level(t,i)$，常见的层接近度计算方法是：统计 $p(x_i)$ 未穿越 p_0 中节点的个数，记为 $\alpha(x_i)$，用其除以目标路径 p_0 的节点个数，即

$$approach\_level(x_i,t)=\frac{\alpha(x_i)}{|p_0|}.$$

若 $p(x_i)$="1,2,4,5"，p_0="1,2,3,5"，则按照上述方法计算得 $approach\_level(x_i,t)=\frac{1}{4}=0.25$.

谢晓园等人[10]针对路径覆盖测试数据进化生成问题，给出测试数据与目标路径距离的如下计算方法：

$$distance(test,target)=length(target)-similarity(track,target),$$

其中，$length(target)$ 是目标路径的长度，$similarity(track,target)$ 是用测试数据穿越路径与目标路径相同节点个数表示的相似度，并进一步给出了 3 种不同的计算相似度的方法.

我们曾提出一种新的适应值函数计算方法[15]，该方法针对多目标路径覆盖测试数据生成问题，将所有目标路径用赫夫曼编码表示，根据个体穿越路径与目标路径编码的匹配程度设计适应值函数，实际上也给出了一种表示层接近度的新方法. 后来，我们又给出覆盖大规模路径的测试数据生成方法，该方法的适应值函数仅考虑个体穿越路径与目标路径从前至后连续相同节点的个数[16].

2.3　分支距离与层接近度结合的计算方法

Wegener 等人[9]及 Ahmed 等人[3]在多路径覆盖测试数据生成中，均采用分支距离与层接近度之和作为个体适应值. 即第 t 代种群中第 i 个个体的适应值 $fit(x_i,t)$ 计算方法如下：

$$fit(x_i,t)=approach\_level(x_i,t)+branch\_distance(x_i,t).$$

一般来说，分支距离的大小与输入数据有关，其值可能会远大于层接近度的值，因此分支距离常被执行规范化处理. McMinn[17]及 Harman 等人[18]给出的适应值函数是将规范化后的分支距离与层接近度结合，即

$$fit(x_i,t)=approach\_level(x_i,t)+normalize(branch\_distance(x_i,t)),$$

其中，$normalize(d)=1-1.001^{-d}$，将分支距离规范化到[0,1)内，进化过程中将适应值函数最小化.

Arcuri[19]分析不同的分支距离规范化方法，提出

一种新的规范化方法，该方法将原有的分支距离除以该值再加上一个大于 0 的常数，即 $normalize(d)=\frac{d}{d+\beta}$（$\beta$ 为大于 0 的常数），并验证了新方法的优越性.

上述方法有效解决了路径覆盖测试数据自动生成问题，但均只考虑单一个体满足目标路径的程度，没有考虑个体穿越的节点是否为目标路径难以覆盖的节点，因此不能保证对穿越难以覆盖节点的个体给予较高的适应值，从而穿越难以覆盖节点的稀有数据没有得到保护.

鉴于此，本文根据个体穿越节点的难易程度调整不同个体的适应值，使稀有数据的适应值得到增加，可以有效保护这些稀有数据，从而提高测试数据生成的效率. 调整适应值的方法是通过统计种群中个体穿越目标路径节点的信息，计算个体对目标路径的贡献程度，以其作为权重调整个体适应值. 下面首先介绍个体贡献度的定义及求解方法.

3 个体贡献度计算

应用遗传算法生成测试数据时，有大量数据运行被测程序. 穿越路径节点的测试数据个数与该节点被穿越的概率有关，概率越大，穿越该节点的数据越多，即个体数越多，从而测试数据越容易生成；概率越小，穿越该节点的测试数据越少，从而测试数据越难以生成. 本文将难以穿越的节点称为小概率节点，穿越小概率节点的测试数据为稀有数据. 使用遗传算法生成测试数据时，穿越小概率节点的个体对生成穿越目标路径的测试数据的贡献大. 因此，本文基于每代种群的个体穿越目标路径节点信息计算个体的贡献度.

在目标路径的节点中，有些节点被任何个体穿越的路径包含，为程序运行的必经节点. 如果考虑这些节点，将无助于区分个体的优劣，还会增加计算量. 因此，在计算个体的贡献度时，先对目标路径的节点约简.

3.1 路径节点约简

(1) 必经节点. 从程序的起点到终点必须经过的节点，称为必经节点. 以任何数据运行被测程序都穿越必经节点，如图 1 中的节点 1、3、5、7、16 均为必经节点.

(2) 选经节点. 程序的节点中，除必经节点之外的节点，称为选经节点. 不是所有的测试数据都能穿越选经节点，如图 1 中的节点 2、4、6、8、9、10、11、12、13、14、15 均为选经节点.

只有选经节点能反映一条路径的走向，如图 1 中选择路径“s,1,2,3,4,5,6,7,9,13,15,16,e”，那么，2、4、6、9、13、15 为反映路径走向的选经节点.

必经节点不能区分测试数据对穿越目标路径的贡献，因此没有必要统计这些节点被测试数据穿越的情况，仅考虑选经节点即可. 于是，在计算个体贡献度时先对目标路径进行约简.

(3) 约简方法. 去除路径中的必经节点、起始节点和终止节点，仅用选经节点表示路径.

如图 1 所示，若目标路径为“s,1,2,3,4,5,6,7,9,13,15,16,e”，不考虑起始和终止节点，有 11 个节点，该路径可约简为“2,4,6,9,13,15”，仅包含 6 个节点；若目标路径为“s,1,2,3,5,7,9,10,11,16,e”，约简后为“2,9,10,11”，仅包含 4 个节点. 对路径的约简将使程序插桩和个体贡献度计算量大大减少.

被测程序可能含有循环结构，这样目标路径中的某个节点可能在其中出现一次，也可能出现多次. 不同个体执行该节点的次数可能不同. 下面先给出目标路径中不包含循环执行节点的个体贡献度计算方法；再进一步给出包含循环执行节点的个体贡献度的计算.

3.2 个体贡献度计算

记种群规模为 m，目标路径 p 包含 n 个节点，有 n' 个选经节点，均不在循环结构中. 对于第 t 代种群的个体 x_i，按照如下方法，计算该个体对生成穿越目标路径 p 的测试数据的贡献度.

通过统计目标路径的节点被个体穿越的情况，得到个体穿越矩阵，记为 $\boldsymbol{trav}(t)$，表示如下：

$$\boldsymbol{trav}(t)=\begin{array}{c} \begin{matrix} node_1 & node_2 & \cdots & node_{n'} \\ \downarrow & \downarrow & \cdots & \downarrow \end{matrix} \\ \begin{bmatrix} c_{11}(t) & c_{12}(t) & \cdots & c_{1n'}(t) \\ c_{21}(t) & c_{22}(t) & \cdots & c_{2n'}(t) \\ \vdots & \vdots & \vdots & \vdots \\ c_{m1}(t) & c_{m2}(t) & \cdots & c_{mn'}(t) \end{bmatrix} \end{array} \begin{matrix} \leftarrow x_1 \\ \leftarrow x_2 \\ \vdots \\ \leftarrow x_m \end{matrix}$$

其中，$c_{ij}(t)=\begin{cases}1, & x_i\text{穿越}n_j \\ 0, & x_i\text{没穿越}n_j\end{cases}$.

根据 $\boldsymbol{trav}(t)$，计算第 t 代种群中穿越节点 $node_j(j=1,2,\cdots,n')$ 的个体数目，记为 $s_j(t)$，容易得到 $s_j(t)=\sum_{i=1}^{m}c_{ij}(t)$. 那么，个体 x_i 对目标路径第 j 个节点的贡献度 $Q_j(x_i,t)$ 可以表示为

$$Q_j(x_i,t)=\begin{cases}0, & c_{ij}(t)=0 \\ \dfrac{1}{s_j(t)}, & c_{ij}(t)=1\end{cases} \tag{1}$$

由式(1)可知，穿越小概率节点的个体比较少，从而这些个体具有较大的贡献度，最大值为 1，对应仅有一个个体穿越的情况；穿越容易覆盖节点的个体相对较多，从而这些个体具有较小的贡献度，最小值为 0，对应没有穿越任何选经节点的个体. 这些值可以构成个体对目标路径节点的贡献度矩阵，记为 ***contr***(t)，并表示如下：

$$\boldsymbol{contr}(t)=\begin{array}{c} \begin{matrix} node_1 & node_2 & \cdots & node_{n'} \\ \downarrow & \downarrow & \cdots & \downarrow \end{matrix} \\ \begin{bmatrix} Q_1(x_1,t) & Q_2(x_1,t) & \cdots & Q_{n'}(x_1,t) \\ Q_1(x_2,t) & Q_2(x_2,t) & \cdots & Q_{n'}(x_2,t) \\ \vdots & \vdots & \vdots & \vdots \\ Q_1(x_m,t) & Q_2(x_m,t) & \cdots & Q_{n'}(x_m,t) \end{bmatrix} \end{array} \begin{matrix} \leftarrow x_1 \\ \leftarrow x_2 \\ \vdots \\ \leftarrow x_m \end{matrix}$$

矩阵的行代表第 t 代种群的某个体对目标路径不同节点的贡献度，矩阵的列代表不同个体对目标路径某节点的贡献度. 容易理解，如果某个体对目标路径的多个节点的贡献度都比较大，那么该个体对目标路径的贡献度也比较大. 因此，个体对目标路径的贡献度可以通过该个体对目标路径的所有节点的贡献度求和得到，记为 $Q(x_i,t)$，那么，$Q(x_i,t)$ 可表示为

$$Q(x_i,t)=\sum_{j=1}^{n'}Q_j(x_i,t) \tag{2}$$

由式(2)可知，如果某个个体对应的测试数据是稀有数据，那么该个体的贡献度就会较大.

例如，对于图 1 的目标路径“s,1,2,3,4,5,6,7,9,10,11,16,e”，约简后，该路径可以表示为“2,4,6,9,10,11”，若种群规模为 5，进化到第 3 代时 5 个个体 $x_1,x_2,\cdots,x_5$ 穿越的路径分别为

$p(x_1)$=“s,1,2,3,4,5,6,7,9,13,15,16,e”，

$p(x_2)$=“s,1,3,5,6,7,8,16,e”，

$p(x_3)$=“s,1,2,3,5,6,7,9,13,15,16,e”，

$p(x_4)$=“s,1,3,4,5,6,7,9,10,12,16,e”，

$p(x_5)$=“s,1,2,3,5,6,7,8,16,e”，

则个体穿越矩阵和贡献度矩阵分别为

$$\boldsymbol{trav}(3)=\begin{array}{c} \begin{matrix} 2 & 4 & 6 & 9 & 10 & 11 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} \leftarrow x_1 \\ \leftarrow x_2 \\ \leftarrow x_3 \\ \leftarrow x_4 \\ \leftarrow x_5 \end{matrix}$$

和

$$\boldsymbol{contr}(3)=\begin{array}{c} \begin{matrix} 2 & 4 & 6 & 9 & 10 & 11 \\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow \end{matrix} \\ \begin{bmatrix} \frac{1}{3} & \frac{1}{2} & \frac{1}{5} & \frac{1}{3} & 0 & 0 \\ 0 & 0 & \frac{1}{5} & 0 & 0 & 0 \\ \frac{1}{3} & 0 & \frac{1}{5} & \frac{1}{3} & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{5} & \frac{1}{3} & 1 & 0 \\ \frac{1}{3} & 0 & \frac{1}{5} & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{matrix} \leftarrow x_1 \\ \leftarrow x_2 \\ \leftarrow x_3 \\ \leftarrow x_4 \\ \leftarrow x_5 \end{matrix}$$

从而，这些个体的贡献度分别为

$$Q(x_1,3)=\frac{1}{3}+\frac{1}{2}+\frac{1}{5}+\frac{1}{3}\approx 1.37,$$

$$Q(x_2,3)=\frac{1}{5}=0.2,$$

$$Q(x_3,3)=\frac{1}{3}+\frac{1}{5}+\frac{1}{3}\approx 0.87,$$

$$Q(x_4,3)=\frac{1}{2}+\frac{1}{5}+\frac{1}{3}+1\approx 2.03,$$

$$Q(x_5,3)=\frac{1}{3}+\frac{1}{5}\approx 0.53.$$

可以看出，不同个体对目标路径的贡献度得到有效区分.

3.3　对循环结构的处理

在循环体中的节点可能在目标路径中出现多次. 如路径 p=“$s,node_1,node_2,\cdots,node_n,e$”，约简后选经节点可能会有重复的情况. 不失一般性，这里假设前 n' 个节点不重复，$n'<n$，并记节点 $node_j$ 在 p 中出现的次数为 $r(node_j)(j=1,2,\cdots,n')$.

对于目标路径的每个节点，统计穿越该节点的个体数目，记 x_i 穿越目标路径的前 n' 个节点的次数分别为 $r_i(node_j)(j=1,2,\cdots,n')$，那么，设定 x_i 对目标路径节点 n_j 的贡献度修正系数为

$$\vartheta_j(x_i)=\frac{1}{|r_i(node_j)-r(node_j)|+1} \tag{3}$$

例如，节点 $node_1$ 在目标路径 p 中出现 3 次，则 $r(node_1)=3$，如果个体 x_1 穿越的路径执行 1 次 $node_1$，则 $r_1(node_1)=1$，由式(3)可得

$$\vartheta_1(x_1)=\frac{1}{|r_1(node_1)-r(node_1)|+1}=\frac{1}{|1-3|+1}$$
$$\approx 0.333.$$

如果个体 x_2 穿越的路径执行 3 次 $node_1$，则 $r_2(node_1)=3$，由式(3)可得

$$\vartheta_1(x_2)=\frac{1}{|r_2(node_1)-r(node_1)|+1}=\frac{1}{|3-3|+1}=1.$$

如果个体 x_3 穿越的路径执行 7 次 $node_1$，则 $\vartheta_1(x_3)=0.2$. 这样，根据个体穿越目标路径节点的次数与目标路径本身执行该节点次数的接近程度，修正个体对目标路径节点的贡献度，保证次数越接近目标路径的要求，贡献度修正系数越大；与目标路径次数相同时，贡献度修正系数达到最大值 1.

对于包含循环体结构在内的所有程序，x_i 对目标路径节点 n_j 的贡献度可以表示为

$$Q'_j(x_i,t)=Q_j(x_i,t)\cdot\vartheta_j(x_i) \tag{4}$$

此时，个体 x_i 对目标路径的贡献度为

$$Q'(x_i,t)=\sum_{j=1}^{n'}Q'_j(x_i,t)=\sum_{j=1}^{n'}(Q_j(x_i,t)\cdot\vartheta_j(x_i)) \tag{5}$$

使用此贡献值作为权重调整个体适应值，得到基于稀有数据有效扑捉的路径覆盖测试数据进化生成方法.

4 基于稀有数据有效扑捉的路径覆盖测试数据进化生成

使用遗传算法生成路径覆盖的测试数据，需将路径覆盖测试数据生成问题转化为函数优化问题. 具体过程为：(1) 在程序的输入空间中随机生成一定数量的测试数据，对其进行编码成为进化个体作为初始种群；(2) 循环执行以下操作：将解码后的进化个体作为程序的输入，执行插桩后的被测试程序；通过适应值函数评价进化个体的优劣；采用遗传算子生成新的进化种群. 如此反复，解码后的优化解就可能是穿越目标路径的测试数据. 其中，个体编码方法、解码方法以及遗传算子中的交叉和变异方式，均可以根据数据特点选择不同的方法，此不赘述. 适应值设计是影响算法效率的关键，本文基于个体的贡献度调整个体的适应值，得到基于稀有数据扑捉的路径覆盖测试数据进化生成方法.

4.1 适应值函数调整

记第 t 代个体 x_i 穿越的路径为 $p(x_i)$，目标路径为 p_0，x_i 的层接近度为 $approach\_level(x_i,t)$，计算方法采用统计个体穿越路径 $p(x_i)$ 与目标 p_0 相同节点的个数，用其除以目标路径的节点数来计算，该值越大个体越优；分支距离为 $branch\_distance(x_i,t)$，计算方法与 Tracey 方法相同，为了权衡其与层接近度的大小，并统一为最大化运算，将其采用 $1.001^{-branch\_distance(x_i,t)}$ 进行规范化，其值越大个体越优，于是 x_i 的适应值 $fit(x_i,t)$ 可表示为

$$fit(x_i,t)=approach\_level(x_i,t)+1.001^{-branch\_distance(x_i,t)} \tag{6}$$

将式(5)作为式(6)的权重调整原来的适应值，得到调整后的个体的适应值记为 $f(x_i,t)$，表示如下：

$$f(x_i,t)=fit(x_i,t)\cdot Q'(x_i,t)=(approach\_level(x_i,t)+1.001^{-branch\_distance(x_i,t)})\cdot\sum_{j=1}^{n'}(Q_j(x_i,t)\cdot\vartheta_j(x_i)) \tag{7}$$

在传统适应值函数的基础上，乘以体现个体贡献度的权重，对穿越小概率节点的个体，赋以较大的权重，保证其有较高的适应值，从而使稀有数据得到有效保护.

4.2 性能分析

为了便于分析，假设目标路径不包含循环执行的选经节点，并在种群进化到第 t 代，有 2 个体 x_1 和 x_2，穿越的路径分别为 $p(x_1)$ 和 $p(x_2)$. 与目标路径对照，除了分别穿越目标路径在第 j 和第 k 个节点上不同外，其余都相同. 按照传统的适应值计算方法，它们对应的层接近度将相等. 这里假设它们对应的分支距离也相等，则按照式(6)计算得到 $fit(x_1)=fit(x_2)$. 如果穿越目标路径的第 j 个节点的个体数少于穿越目标路径的第 k 个节点的个体数，即 $s_j(t)<s_k(t)$，则由式(1)得 $Q_j(x_1,t)>Q_k(x_2,t)$. 由于两个体穿越目标路径其它节点都相同，即 $Q_y(x_1,t)=Q_y(x_2,t)$，$y\neq j$ 或 k，于是由式(2)得 $Q(x_1,t)>Q(x_2,t)$，假设第 j 个和第 k 个节点均不是循环体中的节点，从而由式(5)得 $Q'(x_1,t)>Q'(x_2,t)$. 也就是说，按照传统的适应值计算相等的情况下，由于两个体穿越目标路径节点的难易程度不同，导致本文方法对应的适应值调整权重不同. 按照本文方法，由式(7)得到 x_1 和 x_2 的适应值分别为

$$f(x_1,t)=Q'(x_1,t)\cdot fit(x_1),$$
$$f(x_2,t)=Q'(x_2,t)\cdot fit(x_2),$$

即两个体的适应值不同，$f(x_1,t)>f(x_2,t)$.

这样，根据进化过程中个体的贡献度调整个体的适应值，可以使得穿越难以覆盖节点的稀有数据获得较高的适应值，有利于在后续进化中保留下来.

上述分析表明，通过本文提出的调整策略，对于覆盖目标路径的难以覆盖的选经节点的进化个体，其适应值能够得到增加，这样增大了这些进化个体

参与后续遗传操作的机会，从而提高了生成覆盖目标路径的测试数据的效率.

4.3 算法步骤[20]

1. 确定目标路径，插桩被测程序，并对算法的控制参数赋值；

2. 将种群初始化；

3. 分别将每个进化个体解码，并以其为输入运行被测程序；

4. 判断是否有个体穿越路径与目标路径完全相同. 若有，保存该个体，并转至步骤 8；

5. 判断是否达到事先设定的最大进化代数，若达到，则停止种群的进化；

6. 依据式(7)计算进化个体的适应值；

7. 对种群依次实施选择操作、交叉操作以及变异操作，生成子代种群，并转至步骤 3；

8. 停止种群的进化过程，解码保存的进化个体，输出穿越目标路径的测试数据.

5 实　验

为了验证本文方法的有效性，被测程序选择了 2 个基准程序和 6 个工业用例，均为 C 语言程序，仿真环境为 VC++6.0，实验机器主频和内存分别为 2.80GHz 和 2GB.

5.1 对比方法与参数设置

选择如下两种对比方法：

一是文献[17]方法，这里称其为传统方法. 该方法使用分支距离和层接近度结合作为适应值，与本文方法均采用遗传算法进化生成测试数据. 本文方法是在其基础上增加了对稀有数据的扑捉技术. 与其对比，是为了验证本文方法生成测试数据的效率是否有所提高.

另一种是随机法[21]，该方法在程序的输入域随机生成均匀分布的测试数据，通过运行被测程序，判断是否找到穿越目标路径的测试数据. 该方法为常见的测试数据生成对比方法.

为保证抽样个体间的差异尽可能的小，3 种方法采用相同的种群规模及初始种群. 本文方法和传统方法使用的遗传算法控制参数也相同，个体均采用二进制编码，采用轮盘赌选择、单点交叉、单点变异，交叉和变异概率分别设定为 0.9 和 0.3.

每种算法运行过程中，如果生成了穿越目标路径的测试数据或者是达到了预先设定的最大进化代数则终止. 由于不同作者的算法，其算法机制不尽相同，一般来说，在相同的仿真环境下，使用找到覆盖目标路径的测试数据需要的平均评价次数来比较不同算法的性能，是相对比较公平和简便的方法，评价次数越少，说明算法的性能越好；同时，由于算法的运行时间一方面与评价次数有关，另一方面也体现了算法每次评价个体耗时的多少，为了体现不同算法的时间消耗，实验过程中也记录了各种算法多次运行的总时间，求得每次运行的平均时间进行比较，该时间越少，说明算法效率越高；此外，还比较了生成测试数据的成功率，该值越大，说明算法越有效.

5.2 基准程序实验

为避免随机性对各算法性能的影响，本组实验设定每种方法均运行 15 次.

5.2.1 三角形分类程序

首先，对图 1 所示的三角形分类程序进行实验. 选择一条难度较大的路径，即等边三角形“s,1,2,3,4,5,6,7,9,10,11,16,e”作为目标路径，输入数据的范围分 4 种情况，不同情况选择不同的种群规模和最大终止代数，实验设置及结果如表 1 所列，其中，成功率指在最大运行代数之内成功生成测试数据的实验次数与实验总数(即 15 次)的比值.

表 1　三角形分类程序实验设置及实验结果

实验设置			本文方法			传统方法			随机法		
数据范围	种群规模	最大代数	评价次数平均值	运行时间平均值/s	成功率/%	评价次数平均值	运行时间平均值/s	成功率/%	评价次数平均值	运行时间平均值/s	成功率/%
[1,128]	50	5000	7125.0	0.0008	100	120315.3	0.0121	100.00	128055.3	0.0008	100.00
[1,256]	50	10000	10910.7	0.0012	100	318240.0	0.0335	100.00	535965.0	0.0037	100.00
[1,512]	100	20000	30540.7	0.0041	100	1225290.3	0.1686	86.67	1673340.0	0.0124	80.00
[1,1024]	200	50000	98440.3	0.0155	100	4727740.7	0.5905	80.00	5832380.7	0.0419	66.67

由表 1 可以看出：

(1) 对于每种数据范围，本文方法都以最少的评价次数成功生成了测试数据. 如数据范围为[1,256]时，本文方法的评价次数为 10910.7；传统方法为 318240.0，约是本文方法的 29.2 倍；随机法为 535965.0 次，约是本文方法的 49.1 倍. 其它 3 种数据范围得到的结果也类似，说明本文方法的优势十分明显.

(2) 从运行时间平均值上看，只有数据范围在[1,128]时，本文方法为 0.0008s，与随机法相同，其

它数据范围本文方法都明显优于传统方法和随机方法;可以看出,随机法的时间消耗相对于评价次数来说,要少得多,这是由于随机法不涉及对每个个体的适应值计算时间,而且每一代个体均由随机生成,使得时间消耗较少;传统方法评价次数虽然少于随机法,但是由于每个个体适应值计算以及交叉变异操作产生新个体都需要消耗时间,总体运行时间较多;本文方法虽然个体生成过程与传统方法相同,但是通过对适应值的调整,对生成最优解十分有利,使得评价次数大大减少,因此评价时间达到最少,这说明本文方法生成测试数据的效率明显高于其它两种方法.

(3)在输入数据的范围小时,每种方法都能成功生成测试数据.随着输入数据范围的增加,生成测试数据的难度也不断增加,传统方法和随机法都有生成测试数据失败的情况.如输入数据的范围是[1,512]和[1,1024]时,传统方法生成测试数据的成功率分别为 86.67% 和 80.00%;随机法分别为 80.00%和 66.67%.这说明,随着问题难度的加大,本文方法的优势更加明显.

为了进一步验证本文方法在搜索空间爆炸性增长情况下的算法性能,选择三角形分类程序更大的输入范围进行实验.实际上,当输入的 3 个数据范围均为[1,2048]时,搜索空间为$[1,2048]^3$,此空间中共有数据2048^3个,而能够穿越等边三角形路径的测试数据只有 2048 个,因此,生成需要的测试数据的概率为$2048/2048^3=2^{-22}$,显然生成等边三角形的概率很小,继续加大被测程序的输入数据范围时会导致搜索空间爆炸性增长,在种群规模不变的情况下,种群中的稀有数据变得更加稀少,此时保护穿越难以覆盖节点的稀有数据更加重要.仍以等边三角形路径为目标路径.实验设置及结果如表 2 所列.

表 2 大搜索空间下三角形分类程序实验设置及实验结果

实验设置			本文方法			传统方法			随机法		
数据范围	种群规模	最大代数	评价次数平均值	运行时间平均值/s	成功率/%	评价次数平均值	运行时间平均值/s	成功率/%	评价次数平均值	运行时间平均值/s	成功率/%
[1,2048]	200	60000	190400.0	0.0136	100	9892247.7	1.8177	60.00	11315746.7	0.0743	33.33
[1,4096]	200	70000	304800.0	0.1104	100	13604260.3	3.6758	26.67	14000000.0	0.1192	0.00
[1,8192]	200	80000	691840.0	0.5076	100	15469982.0	4.2812	6.67	16000000.0	0.1781	0.00
[1,16384]	200	90000	1299760.0	1.8752	100	18000000.0	5.9809	0.00	18000000.0	0.3412	0.00
[1,32768]	200	100000	2364224.0	4.1250	100	20000000.0	7.5611	0.00	20000000.0	0.3897	0.00

从表 2 可以看到,当数据范围变大,导致搜索空间爆炸性增长的情况下,

(1)随机法对于小概率的情况,很难生成测试数据,在本组实验的 5 种数据范围情况下,只有范围在[1,2048]时有 5 次成功生成了测试数据,因此成功率为 5/15≈33.33%,其它实验均没有生成需要的测试数据,成功率均为 0.

(2)传统方法的评价次数比随机法少,但是远多于本文方法,导致平均运行时间也比本文方法多很多;生成测试数据的成功率虽高于随机法,但是远不如本文方法,该方法在数据范围是[1,2048],[1,4096]和[1,8192]时,生成测试数据的成功率分别为 60.00%,26.67%和 6.67%,尤其是在范围扩大到[1,16384]和[1,32768]时,成功率为 0,即15 次实验中每次达到最大运行代数,都没能生成需要的测试数据;这说明传统方法在搜索空间爆炸性增长的情况下很难奏效.

(3)本文方法随着数据范围的加大,虽然评价次数逐渐增多,运行时间也随之变长,但是仍能有效生成测试数据,而且测试数据生成率均为 100%,这说明本文方法在搜索空间爆炸情况下对测试数据生成效率仍有所改进.

5.2.2 冒泡排序程序

为了验证本文方法在包含循环结构的基准程序上的有效性,选择按照升序排列数据的冒泡排序程序作为被测程序,对 8 个[1,65535]范围内的数据排序.种群规模为 100,最大终止代数为 1000.选择一条 8 个数逆序排列的测试数据穿越的路径作为目标路径,3 种方法各独立运行 15 次,得到的实验结果如表 3 所列,由于实验过程中统计的是 15 次运行的总时间,所以没有给出算法每次运行时间,而是用总时间求得平均运行时间列在表 3 的最后一列.

表 3 冒泡排序程序实验结果

	平均值评价次数	成功次数(成功率)	运行时间平均值/s
本文方法	23206.7	15(100%)	0.1409
传统方法	50953.3	13(86.7%)	0.1781
随机法	58273.3	12(80%)	0.1599

由表 3 可以看出:

(1)本文方法能在设定的进化代数内成功生成测试数据,成功率为 100%;传统方法成功次数为

13，成功率为 86.7%；随机法成功次数为 12，从而生成测试数据的成功率为 80.0%.

(2) 对于每次实验，本文方法总以最少的评价次数生成穿越目标路径的测试数据，平均评价次数为 23206.7，不足传统方法 50953.3 次的一半，更少于随机法的 58273.3 次.

(3) 本文方法的平均运行时间为 0.1409 s，小于传统方法的 0.1781 s，但是大于传统方法运行时间的一半，这说明，相对于评价次数上与传统方法的差别，本文方法在运行时间上与传统方法的差别变小，这是由于本文方法在个体贡献值计算上需要耗费一定的时间所致.

总体看来，对于两个基准测试程序，与传统方法和随机法相比，本文方法在生成测试数据的成功率、评价次数以及运行时间上均有明显的优势.

5.3　工业用例实验

为进一步验证本文方法在工业用例上的有效性，选择 6 个工业用例[22]进行实验，每个程序随机选择一条可行路径作为目标路径. 程序描述及参数设置如表 4 所列.

表 4　工业用例的目标路径及参数设置

被测程序	代码行数	目标路径节点数	种群规模	最大运行代数
space(fixgramp)	90	18	100	2000
space(fixsgrid)	115	12	100	1500
tot_info	406	53	100	20000
replace	564	85	200	30000
sed	8063	382	200	50000
flex	11783	543	400	50000

本组实验，设定每种方法分别独立运行 50 次，统计评价次数的平均值及标准差、生成测试数据的成功率、运行时间平均值如表 5 所列.

表 5　工业用例实验结果

被测程序	本文方法				传统方法				随机法			
	评价次数平均值	评价次数标准差	运行时间平均值/s	成功率/%	评价次数平均值	评价次数标准差	运行时间平均值/s	成功率/%	评价次数平均值	评价次数标准差	运行时间平均值/s	成功率/%
space(fixgramp)	8934.1	3798.5	0.0299	100	14213.8	8739.2	0.0337	100	19273.3	12319.4	0.0367	100
space(fixsgrid)	5498.9	2911.1	0.0312	100	12215.7	5643.8	0.0399	100	16296.8	8451.9	0.0407	100
tot_info	690465.7	22094.9	8.8430	100	1046532.9	32442.5	8.0167	72	1590486.7	54076.5	5.6197	40
replace	871308.0	28076.7	26.5112	100	1953761.6	44287.9	41.8813	66	2539860.7	72127.8	8.9021	26
sed	1425717.3	491482.5	59.8124	100	4089124.5	997058.8	70.6709	46	8809324.2	2178923.1	12.5034	16
flex	6824130.8	985390.8	89.7412	100	13504177.0	1565907.1	103.4367	36	18215701.9	9120437.3	21.0053	18

由表 5 可以看出：

(1) 从评价次数均值来看，本文方法所用的评价次数仍然明显少于传统方法和随机法.

(2) 从评价次数标准差来看，本文方法的标准差最小，这说明本文方法比另外两种方法的稳定性好.

(3) 从运行时间平均值来看，和传统方法比，只有“tot_info”程序本文方法平均运行时间为 8.8430s，略高于传统方法的 8.0167 s，其它被测程序本文方法的运行时间均少于传统方法；和随机法比较，对于成功率相同的 space(fixgramp)和 space(fixsgrid)程序，本文方法平均运行时间少于随机法.

(4) 从成功率来看，由于 space 程序的两个函数相对简单，3 种方法都成功生成了测试数据，即成功率都是 100%. 但是，对于比较复杂且目标路径节点个数较多的 tot_info 和 replace 程序，测试数据生成的难度较高，传统方法的成功率分别为 72.0%和 66.0%；随机法分别为 40.0%和 26.0%；但本文方法的成功率仍均为 100%，每次都成功生成了测试数据，特别是对于更加复杂的 sed 和 flex 程序，传统方法和随机法的成功率都很低，而本文方法生成测试数据的成功率仍为 100%.

以上实验充分验证了本文方法的优越性，但在比较本文方法与传统方法或是与随机法的测试数据生成效率是否有显著性区别时，实际上是需要对这些算法性能进行总体的推断. 而对两个总体进行比较时，常见的方法是通过多次实验采集到这两个总体的样本，根据这些样本的不同来推断总体的不同. 而根据样本推断总体特性的一种有效的数理统计方法是假设检验.

假设检验是统计推断的一个重要分支[23]，根据样本分布规律和概率原理，由样本结果去推断总体特征. 本文采用假设检验方法通过实验结果对算法的实际性能进行预测.

设本文方法、传统方法和随机法每次运行的评价次数分别为 X_1、X_2 和 X_3，为方便表述，每种方法对不同被测程序使用相同的符号，则 X_1、X_2 和 X_3 都是随机变量. 另外，因为 X_1、X_2 和 X_3 的值都受很多随机因素的影响，在多次实验时它们服从正态分布. 设 $X_i \sim N(\mu_i, \sigma_i^2)$，$i=1,2,3$. 比较每种方法随机变

量的平均值 $\mu_i(i=1,2,3)$的大小.其值越小,则该方法生成测试数据需要的评价次数期望值越低,表明该方法生成测试数据的效率越高.

以工业用例 space(fixgramp)为例,给出利用假设检验对 μ_1 和 μ_2 进行比较的具体过程.考虑到样本方差是总体方差的无偏估计,因此,用样本方差的值作为对总体方差的估计值,即用样本标准差的值作为对总体标准差的估计值[24],得到 $\sigma_1=3798.5$ 和 $\sigma_2=8739.2$.样本容量 $n_1=n_2=50$,$\overline{X}_1=8934.1$,$\overline{X}_2=14213.8$,显著性水平 α 的值为 0.01.

第 1 步. 建立原假设 $H_0:\mu_1\geq\mu_2$ 和对立假设 $H_1:\mu_1<\mu_2$;

第 2 步. 构造统计量 $U_1=\dfrac{\overline{X}_1-\overline{X}_2}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_2^2}{n_2}}}$;

第 3 步. 给出拒绝域 $U_1=\dfrac{\overline{X}_1-\overline{X}_2}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_2^2}{n_2}}}\leq-Z_\alpha$;

第 4 步. 计算统计量的值:

$$U_1=\frac{\overline{X}_1-\overline{X}_2}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_2^2}{n_2}}}=\frac{8934.1-14213.8}{\sqrt{\dfrac{3798.5^2}{50}+\dfrac{8739.2^2}{50}}}\approx-3.92;$$

第 5 步. 给出结论:

因为 $U_1=-3.92\leq-Z_\alpha=-2.325$,落在拒绝域内,因此,拒绝 H_0,接受 H_1,即认为本文方法评价次数的期望值比传统方法明显小[24].这说明,与传统方法相比,本文方法生成测试数据需要的评价次数明显少.

在相同的显著性水平 $\alpha=0.01$ 下,$\sigma_1=3798.5$,$\sigma_3=12319.4$,样本容量 $n_1=n_3=50$,建立假设和对立假设分别为 $H_0:\mu_1\geq\mu_3$; $H_1:\mu_1<\mu_3$,构造统计量并计算

$$U_2=\frac{\overline{X}_1-\overline{X}_3}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_3^2}{n_3}}}=\frac{8934.1-19273.3}{\sqrt{\dfrac{3798.5^2}{50}+\dfrac{12319.4^2}{50}}}$$

$$\approx-5.67\leq-Z_\alpha=-2.325.$$

这说明,本文方法生成测试数据需要的评价次数明显少于随机法.

其它 5 个程序比较结果如表 6 所列.由表 6 可以看出,对所有工业用例,本文方法需要的评价次数都明显少于其它两种方法.

总体看来,对于工业用例,本文方法不仅具有最高的成功率;需要的评价次数也明显少于其它两种方法;在生成测试数据成功率相同的情况下,本文方法的用时最少.这充分说明,本文方法生成覆盖目标路径的测试数据效率高.

表 6 假设检验结果

被测程序	对比方法	计算统计量的值	给出结论
space (fixsgrid)	传统方法	$U_1\approx-7.48<-2.325$	接受 $H_1:\mu_1<\mu_2$
	随机法	$U_2\approx-8.54<-2.325$	接受 $H_1:\mu_1<\mu_3$
tot_info	传统方法	$U_1\approx-64.14<-2.325$	接受 $H_1:\mu_1<\mu_2$
	随机法	$U_2\approx-108.94<-2.325$	接受 $H_1:\mu_1<\mu_3$
replace	传统方法	$U_1\approx-145.97<-2.325$	接受 $H_1:\mu_1<\mu_2$
	随机法	$U_2\approx-152.44<-2.325$	接受 $H_1:\mu_1<\mu_3$
sed	传统方法	$U_1\approx-16.94<-2.325$	接受 $H_1:\mu_1<\mu_2$
	随机法	$U_2\approx-23.37<-2.325$	接受 $H_1:\mu_1<\mu_3$
flex	传统方法	$U_1\approx-25.53<-2.325$	接受 $H_1:\mu_1<\mu_2$
	随机法	$U_2\approx-8.78<-2.325$	接受 $H_1:\mu_1<\mu_3$

6 总 结

本文给出一种个体对生成穿越目标路径的测试数据贡献度的计算方法,并基于个体的贡献度调整个体的适应值,使穿越难以覆盖节点的个体具有较高的适应值,从而在后续进化中得到保留,有效提高了测试数据生成的效率.算法分析及在基准程序与工业用例的实验结果表明,与传统方法和随机法相比,本文方法生成测试数据的效率较高.

需要注意的是,本文考虑的是单路径覆盖测试数据进化生成问题.今后将进一步研究一次运行遗传算法,同时生成多个目标路径覆盖测试数据时,个体贡献度的计算方法,提高多路径覆盖测试数据的生成效率.

参 考 文 献

[1] Alshraideh M, Mahafzah B A, Al-Sharaeh S. A multiple-population genetic algorithm for branch coverage test data generation. Software Quality Journal, 2011, 19(3): 489-513

[2] Shan Jin-Hui, Jiang Ying, Sun Ping. Research progress in software testing. Acta Scientiarum Naturalium Universitatis Pekinensis, 2005, 41(1): 134-145(in Chinese)
(单锦辉, 姜瑛, 孙萍. 软件测试研究进展. 北京大学学报(自然科学版), 2005, 41(1): 134-145)

[3] Ahmed M A, Hermadi I. GA-based multiple paths test data generator. Computers and Operations Research, 2008, 35(10): 3107-3124

[4] Bueno P M S, Jino M. Automatic test data generation for program paths using genetic algorithms. International Journal of Software Engineering and Knowledge Engineering, 2002, 12(6): 691-709

[5] Lin J, Yeh P. Automatic test data generation for path testing using gas. Information Sciences, 2001, 131(1-4): 47-64

[6] Watkins A, Hufnagel E M. Evolutionary test data generation: A comparison of fitness functions. Software Practice and

Experience, 2006, 36(1): 95-116

[7] Malhotra R, Garg M. An adequacy based test data generation technique using genetic algorithms. Journal of Information Processing Systems, 2011, 7(2): 363-384

[8] Irfan S, Ranjan P. A concept of out degree in CFG for optimal test data using genetic algorithm//Proceedings of the 1st International Conference on Recent Advances in Information Technology. Dhanbad, India, 2012: 436-441

[9] Wegener J, Baresel A, Sthamer H. Evolutionary test environment for automatic structural testing. Information and Software Technology, 2001, 43(14): 841-854

[10] Xie Xiao-Yuan, Xu Bao-Wen, Shi Liang, Nie Chang-Hai. Genetic test case generation for path-oriented testing. Journal of Software, 2009, 20(12): 3117-3136(in Chinese)
(谢晓园，徐宝文，史亮，聂长海. 面向路径覆盖的演化测试用例生成技术. 软件学报, 2009, 20(12): 3117-3136)

[11] Cao Y, Hu C H, Li L M. An approach to generate software test data for a specific path automatically with genetic algorithm//Proceedings of the 8th International Conference on Reliability, Maintainability and Safety. Chengdu, China, 2009: 888-892

[12] Korel B. Automated software test data generation. IEEE Transactions on Software Engineering, 1990, 16(8): 870-879

[13] McMinn P. Search-based software testing: Past, present and future (keynote paper)//Proceedings of the 4th International Workshop on Search-Based Software Testing. Berlin, Germany, 2011: 153-163

[14] Tracey N, Clark J, Mander K, McDermid J. An automated framework for structural test data generation//Proceedings of the International Conference on Automated Software Engineering. Honolulu, Hawaii, USA. 1998: 285-288

[15] Gong Dun-Wei, Zhang Yan. Novel evolutionary generation approach to test data for multiple path coverage. 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. 电子学报, 2010, 38(6): 1299-1304)

[16] Gong D W, Zhang W Q, Zhang Y. Evolutionary generation of test data for multiple paths coverage. Chinese Journal of Electronics, 2011, 19(2): 233-237

[17] McMinn P. Evolutionary search for test data in the presence of state behaviour [Ph. D. dissertation]. University of Sheffield, England, 2005

[18] Harman M, McMinn P. A theoretical and empirical study of search-based testing: local, global, and hybrid search. IEEE Transactions on Software Engineering, 2010, 36(2): 226-247

[19] Arcuri A. It does matter how you normalise the branch distance in search based software testing//Proceedings of the 3rd International Conference on Software Testing, Verification and Validation. Paris, France, 2010: 205-214

[20] Zhang Yan, Gong Dun-Wei. Evolutionary generation of test data for path coverage based on automatic reduction of search space. Acta Electronica Sinica, 2012, 40(5): 1011-1016 (in Chinese)
(张岩，巩敦卫. 基于搜索空间自动缩减的路径覆盖测试数据进化生成. 电子学报, 2012, 40(5): 1011-1016)

[21] Arcuri A, Iqbal M Z, Briand L. Formal analysis of the effectiveness and predictability of random testing//Proceedings of the 19th International Symposium on Software Testing and Analysis. New York, NY, USA, 2010: 219-230

[22] Hyunsook D, Sebastian E, Gregg R. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Software Engineering: An International Journal, 2005, 10(4): 405-435

[23] Gou Peng-Cheng, Zhao Yang, Yi Hong-Gang, Bai Jian-Ling, Yu Hao, Chen Feng. The application of Permutation Test in the hypothesis test. Application of Statistics and Management, 2006, 25(5): 616-622(in Chinese)
(荀鹏程，赵杨，易洪刚，柏建岭，于浩，陈峰. Permutation Test在假设检验中的应用. 数理统计与管理, 2006, 25(5): 616-622)

[24] Yao Xiang-Juan. Theory of evolutionary generation of test data for complex software and applications [Ph. D. dissertation]. China University of Mining and Technology, Xuzhou, 2011(in Chinese)
(姚香娟. 复杂软件测试数据进化生成理论及应用[博士学位论文]. 中国矿业大学，徐州，2011)

ZHANG Yan, born in 1972, Ph. D. candidate, associate professor. Her research interest is the generation of test data of complex software.

GONG Dun-Wei, born in 1970, Ph. D., professor, Ph. D. supervisor. His main research interests include search-based software engineering, evolutionary computation and intelligent control.

Background

Test data generation for path coverage is an important issue in software testing. It can be described as follows: for a target path of a program under test, search for a test datum in the input domain of the program so that the traversed path

of the test datum is just the target one. Automatically solving this problem will reduce the labor intensity of testers, and improve the efficiency of software testing.

Recently, heuristic search algorithms are used to solve the above problem, among which genetic algorithms are the most widely used. It needs to convert the problem of generating test data to an optimization one. First, a number of input data are randomly generated in the input space of the program under test, and encoded as individuals of the initial population. Then the following processes are repeatedly performed: individuals are decoded as inputs to run the instrumented program, so that their merits can be evaluated. Genetic operators are then applied to these individuals to generate a new population until the termination criterion is reached. The decoded optimal solutions will be the test data traversing the target paths.

It is clearly that a well-designed fitness function is essential to efficiently search of the population. A fitness function evaluates each candidate solution with a fitness value so that different test data can be compared. Various fitness functions have been designed and applied to path testing. However, previous work cannot provide adequate protection to a scarce datum which covers a node difficult to be covered. So the efficiency of generating test data needs to be improved.

In this study, scarce data are dynamically captured during the evolutionary generation of test data. We obtain the contribution of an individual by counting up the number of individuals which traverse each node of the target path, and regard this contribution as a weight to adjust the fitness of the individual. In this way, the fitness of a scarce datum can be increased and the scarce datum can be kept in the subsequent evolution. The efficiency of this method is evaluated in the experiments carried out on two benchmark and six industrial programs. The experimental results confirm that the proposed method is efficient in generating test data for path coverage compared with traditional and random methods.

This work was jointly supported by the National Natural Science Foundation of China under Grant No. 61075061, the Excellent Young Scholars of Higher University of Heilongjiang Province under Grant No. 1252G063, the Natural Science Foundation of Jiangsu Province under Grant No. BK2012566 and No. BK2010187, the Specialized Research Fund for the Doctoral Program of Higher Education under Grant No. 20100095110006, Project of Science and Technology Plan of Mudanjiang City under Grant No. Z2013s043, and the Key Innovative Foundation of Mudanjiang Normal University under Grant No. SY201216.