基于均衡优化理论的路径覆盖测试数据进化生成

范书平[1],张 岩[1],马宝英[2],万 里[3],姚念民[4],宋 妍[1]
(1. 牡丹江师范学院计算机与信息技术学院,黑龙江牡丹江 157011; 2. 牡丹江医学院卫生管理学院,黑龙江牡丹江 157011;
3. 天津大学智能与计算学部,天津 300350; 4. 大连理工大学计算机科学与技术学院,辽宁大连 116024)

摘 要: 为了快速生成覆盖目标路径的测试数据,提出在测试数据进化生成中,利用种群中个体穿越程序各分支的均衡程度调整进化过程. 首先,在个体运行被测程序后,统计个体穿越各分支节点真假分支的情况;然后,设计并计算个体穿越程序的均衡度;最后,计算个体对程序均衡度的影响,使对程序均衡度影响大的个体具有较高的适应值,有更多机会参与到后续进化中,有效地提高了测试数据的生成效率. 基准程序和工业用例的实验结果表明,与同类方法比较,所提出的方法在生成测试数据的运行时间与成功率方面具有优越性.

关键词: 软件测试; 路径覆盖; 遗传算法; 测试数据; 程序均衡度
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2020)07-1303-08
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2020.07.008

Evolutionary Generation of Test Data for Paths Coverage Based on Balance Optimization Theory

FAN Shu-ping[1], ZHANG Yan[1], MA Bao-ying[2], WAN li[3], YAO Nian-min[4], SONG Yan[1]
(1. *School of Computer and Information Technology, Mudanjiang Normal University, Mudanjiang, Heilongjiang* 157011, *China*;
2. *School of Health Management, Mudanjiang Medical University, Mudanjiang, Heilongjiang* 157011, *China*;
3. *Department of Intelligence and Computing, Tianjin University, Tianjin* 300350, *China*;
4. *School of Computer Science and Technology, Dalian University of Technology, Dalian, Liaoning* 116024, *China*)

Abstract: In order to speed up the generation of test data that covers the target path, the paper makes good use of the balance of individual traversing program to adjust the evolutionary process of generating test data. First, after the individuals run the program, the number of individuals crossing the true and false branches of each branch node is counted. Then, the program balance is designed and calculated. Finally, the influence on the program balance of each individual is calculated. The individual with high influence has a bigger fitness value to have greater chance to participate in subsequent evolution. The proposed method effectively improves the efficiency of test data generation. The experiment results of benchmark programs and industrial cases show that our methods have superiority in running time and success rate of test data generation when compared with similar methods.

Key words: software testing; path coverage; genetic algorithm; test data; program balance

1 引言

软件测试是保证软件产品质量的重要手段,自动生成测试数据则是软件测试的重点与难点[1]. 近年来,众多学者研究应用进化理论来生成满足目标要求的测试数据,如 Jiang 等[2]、Jia 等[3]和薛猛等[4]应用粒子群优化算法、Dahiya 等[5]和 Adisrikanth 等[6]用人工蜂群算法以及 Mao 等[7]用蚁群算法来生成测试数据,Shahbazi

收稿日期:2019-05-08;修回日期:2019-12-06;责任编辑:孙瑶
基金项目:黑龙江省省属高等学校基本科研业务费(No. 1353ZD003, No. 2018-KYYWFMY-0104);黑龙江省自然科学基金(No. F2016039);牡丹江市科学技术计划项目(No. Z2018g023);牡丹江师范学院科学技术研究项目(No. YB2019003);大连市科技创新项目(No. 2018J12GX045)

and Miller[8]、Mohi and Mohamad 等[9]、Mao and Lin[10]、Mei and Wang 等[11]、Tian and Gong 等[12]则对遗传算法自动生成测试数据的问题进行了深入研究.

众所周知,适应值函数的设计是遗传算法的关键.截止到目前,众多学者提出不同方法来构造适应值函数,主要包括三种:分支距离、层接近度以及二者相结合的方法[13].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,层接近度则表示输入变量所穿越路径与目标路径的接近程度.近年来已提出多种将分支距离与层接近度之和作为个体适应值的方法.如 McMinn[14]及 Harman 等[15]将分支距离规范化到[0,1)内,给出基于规范化后的分支距离与层接近度的适应值函数,并在进化过程中将适应值函数最小化.Deepak 等[16]结合遗传算法与爬山算法进化生成测试数据,方法中将层接近度乘以一个常数来增加其在适应值函数计算中的比例.张岩等[17]提出扑捉稀有数据,对穿越难以覆盖节点的稀有数据进行保护,并计算个体对目标路径测试数据生成中的贡献度,将个体贡献度应用于适应值计算中.夏春艳等[13]在层接近度与分支距离基础上,利用被测程序条件语句的相关性判定不可达路径,根据节点在不可达路径中出现的概率,提出了基于个体穿越度的适应值函数设计方法,该方法提高了测试数据的生成效率,但在适应值计算上还有待提高.

现有适应度函数设计方法中往往只考虑单一个体满足目标路径的程度,并没有有效利用种群中个体穿越程序各分支的均衡情况,如果在测试数据进化生成时,考虑均衡优化的思想[18],使得测试数据能均衡穿越程序的多个分支,将会加快生成覆盖目标路径的测试数据.为此,本文在测试数据生成中,通过统计个体穿越程序分支节点的真假分支情况,设计并计算分支均衡度、程序均衡度,评价个体对程序均衡性的影响,使有利于改善程序均衡性的个体获得较高的适应值,从而在进化中得到保留,以提高路径覆盖测试数据的生成效率.

2 基本概念

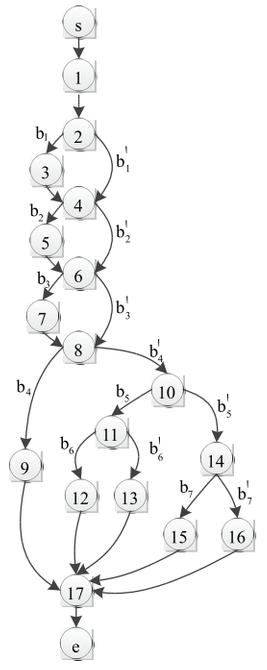
为了介绍方便,先给出几个与被测程序相关的基本概念.

(1)控制流图

控制流图[19]是程序控制结构的图形表示,是一个有向图 $G(N,E,s,e)$,其中 N 称作图 G 的节点集合,与程序的某一条语句对应;E 称作图 G 的边集合,$(node_i, node_j)$称为 G 的边,表示从语句 $node_i$ 到语句 $node_j$ 存在控制流.每个程序的控制流图还包含一个唯一的入口节点 s 和出口节点 e,如图 1(b)是图 1(a)中三角形分类源程序[17]所对应的控制流图.

```
void triangle(int a,int b,int c)
 {
1  int t;
2  if(a>=b)
3      { t=a;a=b;b=t;}
4  if(a>=c)
5     { t=a;a=c;c=t ;}
6  if(b>=c)
7     {t=b;b=c;c=t;}
8  if(a+b<=c)
9     printf("Not Triangle");
   else
   {
10      if(a==b)
        {
11            if(b==c)
12               printf("Equilateral");
              else
13               printf("Isosceles");
        }
        else
        {
14            if(b==c)
15               printf("Isosceles");
              else
16               printf("Triangle");
        }
   }
17 return 0;
 }
```

(a)程序源代码



(b)控制流图

图1 三角形分类程序

控制流图中,出度大于 1 的节点,称为分支节点,以分支节点为起点的两条边为分支节点的真假分支.需要说明的是,根据 Z 路径覆盖[20],循环结构按照执行循环体次数可以化成双分支选择结构,而 switch 语句本身可以表示成双分支选择结构,因此,本文研究每个分支节点有两个分支的情况.

(2)分支均衡度

所有个体对应的数据运行程序后,穿越某分支节点真假分支的个体数之差,即为该分支节点的分支均衡度.

(3)程序均衡度

程序目标路径上所有分支节点的分支均衡度之和,即为程序均衡度.该值反映了所有个体穿越被测程序全部或部分分支的整体均衡情况.

3 程序均衡度的计算

从图 1(b)的控制流图中可知,穿越顺序节点及其直接后继节点的用例数目一定相等,因此,为了降低计算代价,程序均衡度计算中不考虑顺序节点.为了计算程序均衡度,首先给出个体穿越分支矩阵的建立过程.

3.1 建立个体穿越分支矩阵

记遗传算法种群规模为 m,被测程序包含 n 个分支节点,对于第 t 代种群,通过统计目标路径上分支节点 $node_j(j=1,2,\cdots,n)$真分支被个体 $x_i(i=1,2,\cdots,m)$穿越的情况,得到个体穿越真分支矩阵,记 $\boldsymbol{cross}(t)$,表示

如下:

$$\boldsymbol{cross}(t)=\begin{array}{c}\begin{matrix}node_1 & node_2 & \cdots & node_n\\ \downarrow & \downarrow & \cdots & \downarrow\end{matrix}\\ \begin{bmatrix}a_{11}(t) & a_{12}(t) & \cdots & a_{1n}(t)\\ a_{21}(t) & a_{22}(t) & \cdots & a_{2n}(t)\\ \vdots & \vdots & \vdots & \vdots\\ a_{m1}(t) & a_{m2}(t) & \cdots & a_{mn}(t)\end{bmatrix}\begin{matrix}\leftarrow x_1\\ \leftarrow x_2\\ \vdots\ \ \vdots\\ \leftarrow x_m\end{matrix}\end{array}$$

其中:$a_{ij}(t)=\begin{cases}0, & x_i \text{ 未穿越 } node_j \text{ 的真分支}\\ 1, & x_i \text{ 穿越 } node_j \text{ 的真分支}\end{cases}$

矩阵中的行代表第 t 代种群中个体 x_i 是否穿越目标路径上不同分支节点,矩阵的列代表不同个体是否穿越目标路径上的分支节点 $node_j$.

按照同样的方法,通过统计分支节点 $node_j$ 假分支被个体 x_i 穿越的情况,可以得到个体穿越假分支矩阵,记为 $\boldsymbol{cross}'(t)$,表示如下:

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

其中:$a'_{ij}(t)=\begin{cases}0, & x_i \text{ 未穿越 } node_j \text{ 的假分支}\\ 1, & x_i \text{ 穿越 } node_j \text{ 的假分支}\end{cases}$

以图 1 中的三角形分类程序为被测程序. 假设种群规模 $m=4$(即种群中包含 4 个个体),选择目标路径为“s,1,2,3,4,5,6,7,8,10,11,12,17,e”,进化到第 6 代时,统计图 1(b)中目标路径上分支节点 $node_2$、$node_4$、$node_6$、$node_8$、$node_{10}$、$node_{11}$的 6 个真分支 $b_1\sim b_6$ 被个体穿越的情况,得到个体穿越真分支矩阵 $\boldsymbol{cross}(6)$:

$$\boldsymbol{cross}(6)=\begin{array}{c}\begin{matrix}2 & 4 & 6 & 8 & 10 & 11\\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow\end{matrix}\\ \begin{bmatrix}0 & 1 & 1 & 1 & 0 & 0\\ 1 & 1 & 1 & 0 & 1 & 0\\ 1 & 1 & 1 & 0 & 0 & 0\\ 1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}\begin{matrix}\leftarrow x_1\\ \leftarrow x_2\\ \leftarrow x_3\\ \leftarrow x_4\end{matrix}\end{array}$$

同样,通过统计 6 个分支节点的假分支 $b'_1\sim b'_6$ 被个体穿越情况,得到个体穿越假分支矩阵 $\boldsymbol{cross}'(6)$:

$$\boldsymbol{cross}'(6)=\begin{array}{c}\begin{matrix}2 & 4 & 6 & 8 & 10 & 11\\ \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow\end{matrix}\\ \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 1 & 0\\ 0 & 1 & 1 & 1 & 1 & 0\end{bmatrix}\begin{matrix}\leftarrow x_1\\ \leftarrow x_2\\ \leftarrow x_3\\ \leftarrow x_4\end{matrix}\end{array}$$

3.2　计算程序均衡度

当所有个体对应数据运行程序后,根据个体穿越分支矩阵,首先计算出各分支节点的分支均衡度,进而得到程序均衡度.

3.2.1　计算分支均衡度

分支均衡度用以表示测试数据穿越某分支节点真假分支的均衡程度. 根据 3.1 节中建立的个体穿越真分支矩阵 $\boldsymbol{cross}(t)$,计算第 t 代种群中穿越第 k($k=1,2,\cdots,n$)个分支节点真分支的个体数目,记为 $Num_{kT}(t)$,容易得到:

$$Num_{kT}(t)=\sum_{i=1}^{m}a_{ik}(t) \tag{1}$$

同样,根据 3.1 节中个体穿越假分支矩阵 $\boldsymbol{cross}'(t)$,得到穿越第 k 个分支节点假分支的个体数目,记为 $Num_{kF}(t)$,则

$$Num_{kF}(t)=\sum_{i=1}^{m}a'_{ik}(t) \tag{2}$$

根据式(1)和式(2),以及分支均衡度的定义,可以将第 k 个分支节点的分支均衡度表示为 $bb_k(t)$:

$$bb_k(t)=\begin{cases}0, & Num_{kT}(t)=0 \text{ 且 } Num_{kF}(t)=0\\ \dfrac{|Num_{kT}(t)-Num_{kF}(t)|}{|Num_{kT}(t)+Num_{kF}(t)|}, & \text{否则}\end{cases} \tag{3}$$

由式(3)可知,所定义的分支均衡度实际上反应了穿越第 k 个分支节点的所有个体穿越其真假分支的均衡程度. 若穿越第 k 个分支节点真假分支的个体数目差异较小,此时,$bb_k(t)$的值较小,说明个体更均衡的穿越了该节点的真假分支,不难得出,分支均衡度 $bb_k(t)$的值越小越好.

3.2.2　计算程序均衡度

根据式(3),计算程序目标路径上所有分支节点的分支均衡度,并取所有分支节点的分支均衡度之和,作为第 t 代种群中个体穿越程序的程序均衡度,记为 $pb(t)$,如式(4)所示. 根据分支均衡度的定义可知,程序均衡度越小越好,其值越小反映出所有个体对应数据运行被测程序后,被测程序的均衡性越好.

$$pb(t)=\sum_{k=1}^{n}bb_k(t) \tag{4}$$

仍以图 1 中的三角形分类程序为例. 根据第 6 代个体穿越真分支矩阵 $\boldsymbol{cross}(6)$,以及穿越分支节点真分支的个体数目计算公式(详见式(1)),将矩阵 $\boldsymbol{cross}(6)$第一列作和,可以得到穿越第一个分支节点 $node_2$ 真分支的个体数目 $Num_{1T}(6)=\sum_{i=1}^{m}a_{i1}(t)=3$,同理,根据第 6 代个体穿越假分支矩阵 $\boldsymbol{cross}'(6)$,以及穿越分支节点假分支的个体数目计算公式(详见式(2)),得到穿越 $node_2$ 假分支的个体数目 $Num_{1F}(6)=\sum_{i=1}^{m}a'_{i1}(6)=1$,

依此类推,可以得到穿越三角形分类程序分支节点 $node_4$、$node_6$、$node_8$、$node_{10}$、$node_{11}$ 真假分支的个体数目,见表1,进一步根据式(3)计算出第1个分支节点 $node_2$ 的分支均衡度:

$$bb_1(6)=\frac{|Num_{1T}(6)-Num_{1F}(6)|}{|Num_{1T}(6)+Num_{1F}(6)|}=\frac{|3-1|}{|3+1|}=0.5$$

按照相同方法,计算出其它分支节点的分支均衡度,如表1所示.

表1 分支均衡度的计算

分支节点	穿越真分支个体数	穿越假分支个体数	分支均衡度
$node_2$	3	1	0.50
$node_4$	3	1	0.50
$node_6$	3	1	0.50
$node_8$	1	3	0.50
$node_{10}$	1	2	0.33
$node_{11}$	0	1	1.00

最后,根据程序均衡度的计算方法(详见式(4)),得到程序均衡度:

$$pb(6)=\sum_{k=1}^{6}bb_k(6)\approx 0.5*4+0.33+1\approx 3.33$$

4 基于均衡优化理论的测试数据进化生成

为了考查某个体是否能改善程序的均衡性,本文根据删除个体前后程序均衡度的变化情况计算该个体对程序均衡性的影响.

4.1 删除个体后程序均衡度的计算

本小节给出删除某个体后程序均衡度的计算方法. 根据3.1节中的个体穿越真分支矩阵 ***cross***(t) 与3.2节中穿越分支节点真分支的个体数目计算公式(详见式(1)),将删除个体 $x_w(1\leqslant w\leqslant m)$ 后穿越第 k 个分支节点真分支的个体数目 $Num'_{kT}(x_w,t)$ 表示为

$$Num'_{kT}(x_w,t)=\sum_{i=1}^{m}a_{ik}(t)-a_{wk}(t) \tag{5}$$

由式(5)可知,如果删除矩阵 ***cross***(t) 的第 w 行,再重新计算穿越第 k 个分支节点真分支的个体数目,即得到 $Num'_{kT}(x_w,t)$.

同理,根据矩阵 ***cross'***(t) 与式(2),删除个体 x_w 后,穿越第 k 个分支节点的假分支个体数目 $Num'_{kF}(x_w,t)$ 表示为

$$Num'_{kF}(x_w,t)=\sum_{i=1}^{m}a'_{ik}(t)-a'_{wk}(t) \tag{6}$$

进一步根据式(5)与式(6),将删除个体 x_w 后第 k 个分支节点的分支均衡度 $bb'_k(x_w,t)$ 表示为

$$bb'_k(x_w,t)=\begin{cases}0, & Num'_{kT}(x_w,t)=0 \text{ 且 } Num'_{kF}(x_w,t)=0\\ \dfrac{|Num'_{kT}(x_w,t)-Num'_{kF}(x_w,t)|}{|Num'_{kT}(x_w,t)+Num'_{kF}(x_w,t)|}, & \text{否则}\end{cases} \tag{7}$$

从而计算出删除个体 x_w 后第 k 个分支节点的程序均衡度,记为 $pb'(x_w,t)$,则 $pb'(x_w,t)$ 可以表示为

$$pb'(x_w,t)=\sum_{k=1}^{n}bb'_k(x_w,t) \tag{8}$$

4.2 个体适应值的计算

个体适应值计算过程中,根据第 t 代种群中所有个体对应数据运行程序后得到的程序均衡度与删除个体 x_w 后得到的程序均衡度间的大小关系,得到个体对程序均衡性的影响,即为个体 x_w 的适应值 $f(x_w,t)$,可以表示为

$$f(x_w,t)=\begin{cases}pb'(x_w,t)-pb(t), & pb'(x_w,t)>pb(t)\\ 0, & pb'(x_w,t)\leqslant pb(t)\end{cases} \tag{9}$$

由式(9)可知,如果个体 x_w 的删除使程序均衡度增加,即 $pb'(x_w,t)>pb(t)$,说明个体 x_w 的存在能有效改善程序均衡性,则在进化过程中提高个体 x_w 的适应值,优先保留这样的个体,若同时有多个个体能改善程序均衡性,式(9)保证了个体对程序均衡度改善程度越大,个体的适应值就越高. 反之,如果个体 x_w 的删除使程序均衡度不变或者减小,即 $pb'(x_w,t)\leqslant pb(t)$,说明个体 x_w 不能改善程序均衡性,此时要将这样的个体淘汰,降低个体 x_w 的适应值,将其设为0.

仍以三角形分类程序为例进行说明. 根据3.1节中的个体穿越真假分支矩阵 ***cross***(t)、***cross'***(t) 与3.2节中穿越分支节点真假分支的个体数目计算公式(详见式(1)、(2)),运行到第6代,当删除个体 x_1 时,除矩阵 ***cross***(6)中第1行的值,将 ***cross***(6)中第一列剩余元素作和,得到删除个体 x_1 后穿越分支节点 $node_2$ 真分支的个体数目 $Num'_{1T}(x_1,6)=\sum_{i=1}^{m}a_{i1}(t)-a_{11}(t)=3$,同理,得到穿越其它分支节点真分支的个体数目;按照同样的方式,根据 ***cross'***(6),得到,$Num'_{1F}(x_1,6)=\sum_{i=1}^{m}a'_{i1}(t)-a'_{11}(t)=0$,同理计算出穿越其它分支节点真假分支的个体数目. 从而计算删除个体 x_1 后 $node_2$ 的分支均衡度:

$$bb'_1(x_1,6)=\frac{|Num'_{1T}(x_1,6)-Num'_{1F}(x_1,6)|}{|Num'_{1T}(x_1,6)+Num'_{1F}(x_1,6)|}=\frac{|3-0|}{|3+0|}=1$$

同理计算出其它节点的分支均衡度,穿越分支节点真假分支个体数目及分支均衡度如表2所示. 根据表2,进一步计算出删除个体 x_1 后的程序均衡度 $pb'(x_1,6)=\sum_{k=1}^{n}bb'_k(x_1,6)\approx 3.99$. 按照同样的方法,可以计算出删除其它个体的程序均衡度 $pb'(x_2,6)\approx 2.33$,$pb'(x_3,6)\approx 2.33$,$pb'(x_4,6)\approx 3.67$.

表 2　删除个体 x_1 后分支均衡度的计算

分支节点	穿越真分支个体数	穿越假分支个体数	分支均衡度
$node_2$	3	0	1.00
$node_4$	2	1	0.33
$node_6$	2	1	0.33
$node_8$	0	3	1.00
$node_{10}$	1	2	0.33
$node_{11}$	0	1	1.00

根据第三节中未删除个体前计算的程序均衡度 $pb(6)\approx3.33$,以及删除个体 x_1 后程序均衡度 $pb'(x_1,6)\approx3.99$,由式(9)计算出个体 x_1 的适应值 $f(x_1,6)=pb'(x_1,6)-pb(6)\approx0.66$,同理,得到 $f(x_i,6)\approx0(i=2,3)$,$f(x_4,6)\approx0.32$. 根据 $f(x_1,6)>f(x_i,6)(i=2,3,4)$,说明个体 x_1 能最大程度改善原有的程序均衡性;而 $f(x_1,6)$ 与 $f(x_4,6)$ 均大于 0,说明个体 x_1 与 x_4 均能有效改善程序被覆盖的均衡程度,在进化中将有较大几率被保留;个体 x_2 与 x_3 的适应值为 0,进化过程中将会被淘汰. 可以看出,本文提出的方法能将不同个体对程序均衡性的影响得到有效的区分.

4.3　测试数据进化生成算法

本文提出一种基于均衡优化理论的测试数据进化生成方法,算法描述见算法 1.

算法 1　基于均衡优化理论的路径覆盖测试数据进化生成算法

```
输入:算法的参数值
//目标路径 p,最大迭代次数 G
输出:覆盖目标路径 p 的测试数据
BEGIN
    Setprameters( );//设置各参数值
    Initialize(x_w);//初始化种群
    generation ← 0;//进化代数
    Do WHILE(generation≤G ‖ path(x_w)≠p)
        //如果大于最大迭代次数或生成覆盖目标路径的测试数据,
          算法终止
        generation ← generation + 1
        Fitness(x_w);//计算个体的适应值
        Select(x_w);//选择操作
        Cross(x_w);//交叉操作
        Mulate(x_w);//变异操作
        IF(path(x_w) = = p)) THEN
          DataSet( ) ← x_w;
          //如果个体覆盖未找到的目标路径 p,则将进化个体加入
            数据集
        END IF
    END WHILE
END
```

4.4　算法分析

(1)算法实现简单

分支距离的计算往往针对节点中每一个简单谓词,若节点均为复合谓词,则计算量会大大增加,且当程序中存在 flag 现象,会导致算法变成随机搜索[21]. 而层接近度计算过程中,需要比较个体执行路径与目标路径中相同节点个数,比较的次数越多,计算越复杂. 本文方法无需计算分支距离与层接近度,方法实现上更为容易.

(2)计算复杂度小

在遗传算法的各代种群进化过程中,算法的计算时间复杂度主要考虑适应值函数. 当初始种群为 m,目标路径上分支节点数目为 n,总节点数为 $n'(n'>n)$,现有方法[13,14]适应值函数计算中,若每个个体需要与 n' 个节点对比层接近度和分支距离,则这类方法所有个体适应值的时间复杂度为 $o(m*n')$. 而本文适应值函数计算中,时间主要花费在建立个体穿越分支矩阵上,这部分时间复杂度为 $o(m*n)$. 可知,与这些方法相比,本文方法并未增加算法的计算复杂度.

(3)实现数据对程序分支的均衡覆盖

本文方法在种群进化中,根据个体对程序均衡度的影响程度计算适应值,以区分不同个体,保证有效改善程序均衡性的个体在进化中有更大几率被保留,从而保证数据对程序分支的均衡分布,快速生成覆盖目标路径的测试数据.

由此可知,从理论上分析,本文方法优于其它方法,下面通过实验进行验证.

5　实验

为了验证本文方法的有效性,被测程序选择了 1 个基准程序和 5 个工业用例,均为 C 语言程序,仿真环境为 VC ++6.0. 计算机处理器为 Pentium(R) Dual-Core CPU E5200 2.5GHz,内存为 2GB,32 位操作系统.

5.1　评价指标与对比方法

为了更好地验证本文算法生成测试数据的有效性,时间有效性和算法运行代价,仿真实验采用成功率、运行时间和评价次数三个指标来评价[17],成功率表示成功生成测试数据的实验次数与实验总数的比值,评价次数表示每次实验所有个体进化代数之和,评价次数越少、运行时间越短,说明算法性能越好. 实验中遗传算法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,实验结果为多次实验的平均值.

选择的对比方法包括:文献[14]所述方法,采用分支距离与层接近度相结合作为适应值函数;文献[13]所述方法,适应值计算中考虑了分支距离、层接近度和个体穿越度. 两种方法均通过设置适应值函数,采用遗传算法进化生成测试数据,与其对比,以验证本文方法生成测试数据的效率.

5.2 基准程序实验

实验中基准程序选择三角形分类程序,目标路径选择等边三角形路径,“s,1,2,3,4,5,6,7,8,10,11,12,17,e”.实验中除特别说明的参数取值外,三种方法采用其它实验参数以及遗传算法的控制参数均相同,考查在遗传算法最大进化代数较大的情况下,通过设置不同数据范围、不同种群规模、不同进化代数的六种实验条件,来验证三种方法的性能,实验结果如表3所示.

表3 三角形分类程序实验结果

实验设置			本文方法			文献[13]方法			文献[14]方法		
数据范围	种群规模	最大代数	评价次数	运行时间/s	成功率/%	评价次数	运行时间/s	成功率/%	评价次数	运行时间/s	成功率/%
[1,128]	50	5000	**1993.3**	**0.0033**	**100**	2156.2	0.0037	**100**	54443.3	0.0760	**100**
[1,256]	50	10000	**5786.7**	**0.0087**	**100**	8972.6	0.0098	**100**	249164.7	0.4424	87
[1,512]	100	20000	**9153.3**	**0.0167**	**100**	21542.7	0.0265	**100**	438676.0	2.0077	53
[1,1024]	200	50000	**11746.7**	**0.0281**	**100**	45865.8	0.0458	**100**	1897397.3	12.8397	40
[1,2048]	200	60000	**21240.0**	**0.0579**	**100**	76541.5	0.0954	**100**	1367669.3	16.9981	27
[1,4096]	200	70000	**29720.0**	**0.0787**	**100**	98725.4	0.2154	**100**	1554032.0	21.2714	20

由表3可知:

(1)从成功率上看,不难得出,随着数据范围的逐渐增大,用文献[14]所提出的方法生成覆盖目标路径测试数据的成功率越来越小.而在不同的数据范围、种群规模与最大进化代数情况下,本文方法与文献[13]中方法的成功率均能达到100%,说明在这些情况下,应用这两种方法均能有效生成覆盖目标路径的测试数据.

(2)从运行时间上看,对于不同的数据范围,本文方法所需要的运行时间都比其它两种方法少,且随着数据范围的增大,这种趋势更加明显,这充分说明了本文方法的时间有效性更好.这是因为文献[14]在计算适应值时,考虑层接近度与分支距离,文献[13]在文献[14]的基础上,考虑了个体穿越度,本文方法则仅需计算个体对程序均衡性的影响.

(3)从评价次数上看,针对六种实验情况,本文方法评价次数均明显少于其它两种方法.这说明与同类方法相比,本文能用更少的评价次数成功地生成测试数据,且随着数据范围、种群规模与最大进化代数的增加,本文的优势更加明显.

5.3 工业用例实验

为了验证本文提出的方法在工业用例上的有效性,选择五个工业用例进行实验,每个程序随机选择一条可行路径作为目标路径,程序描述及其它参数设置如表4[17]所列,其中节点数表示选择目标路径的节点数,fixgramp与fixsgrid是Space程序的两个函数,实验结果见表5.

表4 工业用例参数设置

被测程序	代码行数	节点目	种群规模	最大代数
fixgramp	90	18	100	2000
fixsgrid	115	12	100	1500
Replace	564	85	200	30000
Sed	8063	382	200	50000
Flex	13225	543	400	50000

表5 工业用例实验结果

被测程序	本文方法			文献[13]方法			文献[14]方法		
	评价次数	运行时间/s	成功率/%	评价次数	运行时间/s	成功率/%	评价次数	运行时间/s	成功率/%
Space(fixgramp)	**3591.4**	**0.0248**	**100**	6405.3	0.0252	**100**	14185.7	0.0423	**100**
Space (fixsgrid)	**3512.2**	0.0295	**100**	5076.8	**0.0274**	**100**	12313.2	0.0513	**100**
Replace	**62348.7**	**12.3746**	**100**	810556.4	19.6435	**100**	1977228.1	55.6490	63
Sed	**389654.1**	**79.4356**	**100**	1045647.2	96.8479	**100**	4158838.1	231.3150	50
Flex	**1965700.4**	**97.3572**	**100**	5475700.4	125.1578	**100**	13747252.2	354.8913	38

由表5可知:从运行时间的平均值来看,对于选择的所有工业用例,除了Space(fixsgrid),本文方法运行时间略高于文献[13],对于其它用例本文方法所用的时间明显少于文献[13]、文献[14]所提出的方法;从成功率来看,本文方法更具有优势,对于所有用例,应用本文方法均能生成满足目标路径的测试数据.对于工业用例Replace、Sed、Flex,本文方法与文献[13]中方法均能生成覆盖目标路径的测试数据,成功率均为100%,而文献[14]中方法的成功率分别为63%、50%与38%.从评价次数来看,在工业用例实验中,本文方法的平均评价次数比同类方法次数要少,这是因为本文方法在测试数据生成时考虑优先选择能改善程序均衡性的个体,使测试数据均衡穿越目标路径上各个分支,因此,应用本文方法需要更少的运行代数进化生成

目标测试数据. 实验结果充分验证了本文方法生成目标路径测试数据的有效性,并且验证了本文方法能够提高测试数据的生成效率.

6 结论

本文通过统计穿越程序中各分支的测试数据数目,并据此设计种群中个体对程序均衡性影响程度相关的适应值函数,保证改善程序均衡性大的个体具有更好的适应值,通过与同类方法比较,结果表明本文方法在评价次数、成功率和进化时间方面具有优越性,有效提高了测试数据的生成效率.

然而,我们的工作有局限性. 虽然基本遗传算法运算简单,并能有效地解决问题,但存在早熟现象且局部寻优能力差,因此,研究优化的遗传算法并应用到本文所提出的方法中,以进一步提高测试数据的生成效率,这是我们下一步的重点工作.

参考文献

[1] 靳蓉,姜淑娟,张红昌. 一种新的数据流覆盖测试数据进化生成方法[J]. 小型微型计算机系统,2012,33(4):722-726.
JIN Rong, JIANG Shu-juan, ZHANG Hong-chang. Novel evolutionary generation approach to test data for data-flow coverage[J]. Journal of Chinese Computer Systems, 2012, 33(4):722-726. (in Chinese)

[2] JIANG S J, SHIN J J, ZHANG Y M, et al. Automatic test data generation based on reduced adaptive particle swarm optimization algorithm [J]. Neurocomputing, 2015, 158: 109-116.

[3] JIA Ya-hui, CHEN Wei-neng, ZHANG Jun, et al. Generating software test data by particle swarm optimization[A]. Proceedings of 10th International Conference on Simulated Evolution and Learning[C]. Springer International Publishing Switzerland, 2014. 37-47.

[4] 薛猛,姜淑娟,张争光,等. 一种基于 Kalman 滤波和粒子群优化的测试数据生成方法[J]. 电子学报,2017,45(10):2473-2483.
XUE Meng, JIANG Shu-juan, ZHANG Zheng-guang, et al. A test data generation method based on Kalman filter and particle swarm optimization algorithm[J]. Acta Electronica Sinica, 2017, 45(10):2473-2483. (in Chinese)

[5] DAHIYA S S, CHHABRA J K, Kumar S. Application of artificial bee colony algorithm to software testing[A]. 21st Australian Software Engineering Conference [C]. Auckland, New Zealand: ASWEC, 2010. 149-154.

[6] ADISRIKANTH KULKARNI N J, NAVEEN K V, et al. Test case optimization using artificial bee colony algorithm [J]. Communications in Computer & Information Science, 2011, 192: 570-579.

[7] MAO C, XIAO L, YU X, et al. Adapting ant colony optimization to generate test data for software structural testing [J]. Swarm & Evolutionary Computation, 2015, 20: 23-36.

[8] SHAHBAZI A, MILLER J. Black-box string test case generation through a multi-objective optimization [J]. IEEE Transactions on Software Engineering, 2016, 42(4): 361-378.

[9] MOHI-ALDEEN S M, MOHAMAD R, DERIS S. Application of negative selection algorithm (NSA) for test data generation of path testing [J]. Applied Soft Computing, 2016, 49: 1118-1128.

[10] MAO Y H, LIN R Q. Application of dynamic program slicing technique in test data generation[J]. Procedia Computer Science, 2017, 111: 355-360.

[11] MEI J, WANG S Y. An improved genetic algorithm for test cases generation oriented paths[J]. Chinese Journal of Electronics, 2014, 23(3): 494-498.

[12] TIAN T, GONG D W. Evolutionary generation approach of test data for multiple paths coverage of message-passing parallel programs[J]. Chinese Journal of Electronics, 2014, 23(2): 291-296.

[13] 夏春艳,张岩,宋丽. 基于节点概率的路径覆盖测试数据进化生成[J]. 软件学报,2016,27(4):802-813.
XIA Chun-yan, ZHANG Yan, SONG Li. Evolutionary generation of test data for paths coverage based on node probability[J]. Journal of Software, 2016, 27(4): 802-813. (in Chinese)

[14] McMinn P. Evolutionary search for test data in the presence of state behavior[D]. University of Sheffield, England, 2005.

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

[16] DEEPAK G, PALLVI G. Basis path testing using SGC&HGA with ExLB fitness function[A]. International Conference on Eco-friendly Computing and Communication Systems[C]. Procedia Computer Science, 2015. 593-602.

[17] 张岩,巩敦卫. 基于稀有数据扑捉的路径覆盖测试数据进化生成方法[J]. 计算机学报,2013,36(12):2429-2439.
ZHANG Yan, GONG Dun-wei. Evolutionary generation of test data for paths coverage based on scarce data capturing [J]. Chinese Journal of Computers, 2013, 36(12): 2429-2439. (in Chinese)

[18] 王亚鑫,杨大成. 一种基于 C-RAN 的 BBU 负载均衡优化算法[J]. 北京邮电大学学报,2018,41(5):115

–119.
WANG Ya-xin, YANG Da-cheng. An optimization algorithm for BBU load balancing in C-RAN[J]. Journal of Beijing University of Posts and Telecommunications, 2018,41(5):115 – 119. (in Chinese)
[19] KAEWYOTHA J, SONGPAN W. Finding the critical path with loop structure for a basis path testing using genetic algorithm[J]. Advances in Intelligent Systems and Computing. 2015,361:41 – 52.
[20] 夏辉,宋昕,王理. 基于 Z 路径覆盖的测试用例自动生成技术研究[J]. 现代电子技术,2006,(6):92 – 94.
XIA Hui, SONG Xin, WANG Li. Research of test case auto generating based on Z path coverage[J]. Modem Electronics Technique,2006,(6):92 – 94. (in Chinese)
[21] 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.
GONG Dun-wei, ZHANG Yan. Novel evolutionary generation approach to test data for multiple paths coverage [J]. Acta Electronica Sinica,2010,38(6):1299 – 1304. (in Chinese)

作者简介

范书平 男,1977 年出生,黑龙江牡丹江人. 副教授,CCF 会员,主要从事复杂软件的测试数据生成和进化计算研究.
E-mail:f 8259@163. com

张 岩 女,1972 年出生,辽宁本溪人. 教授,CCF 会员,主要研究方向为基于搜索的软件工程、进化计算.
E-mail:zhangyancumt@126. com

马宝英(通信作者) 女,1985 年出生,黑龙江鸡西人. 讲师,主要从事复杂软件的测试数据生成研究.

万 里 男,1994 年出生,湖北荆州人. 天津大学在读研究生,主要研究软件测试、语音信号处理.

姚念民 男,1974 年出生,黑龙江大庆人. 教授,主要从事基于搜索的软件工程的研究.

宋 妍 女,1977 年出生,黑龙江牡丹江人. 副教授,主要研究方向为复杂软件的测试数据生成.