软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(4):802−813 [doi: 10.13328/j.cnki.jos.004967]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

基于节点概率的路径覆盖测试数据进化生成*

夏春艳, 张 岩, 宋 丽

(牡丹江师范学院 工学院,黑龙江 牡丹江 157011)

通讯作者: 张岩, E-mail: zhangyan@mdjnu.cn

摘 要: 路径覆盖是软件测试领域重要的测试方法之一.为了提高路径覆盖测试效率,在采用遗传算法进化生成路径覆盖的测试数据过程中,利用被测程序条件语句的相关性判定不可达路径,除路径中必经节点外,其他节点在不可达路径中出现的概率越大,穿越该节点的个体就具有越高的穿越度,在进化过程中应得到保护.提出了根据个体的穿越度设计适应度函数方法,从而提高测试数据的生成效率.将所提方法应用于基准程序和工业用例,并与同类方法比较可知,该方法生成路径覆盖的测试数据具有较高的效率.

关键词: 软件测试;路径覆盖;遗传算法;不可达路径;穿越度

中图法分类号: TP311

中文引用格式: 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.软件学报,2016,27(4):802−813. http://www.jos.org.cn/1000-9825/4967.htm

英文引用格式: Xia CY, Zhang Y, Song L. Evolutionary generation of test data for paths coverage based on node probability. Ruan Jian Xue Bao/Journal of Software, 2016,27(4):802−813 (in Chinese). http://www.jos.org.cn/1000-9825/4967.htm

Evolutionary Generation of Test Data for Paths Coverage Based on Node Probability

XIA Chun-Yan, ZHANG Yan, SONG Li

(School of Technoloty, Mudanjiang Normal University, Mudanjiang 157011, China)

Abstract: Path coverage testing is one of the most important software testing methods. This paper presents a process of using genetic algorithms to generate path coverage test data. When an individual traverses the node that might be contained in the unteachable paths (which are determined based on the correlation of conditional statements), the higher the probability the node exists in the unreachable paths, the higher degree of traversing the individual has; and, the individual with higher degree of traversing should be protected. The fitness function of genetic algorithms is designed based on the individual traversing degree, so the efficiency of generating test data is improved. The proposed method is applied to benchmark and industrial programs, and is compared with other methods. The experimental results show that the proposed method is efficient in generating test data for path coverage.

Key words: software testing; path coverage; genetic algorithm; unreachable path; traversing degree

软件工程的主要目标是生产高质量的软件[1].在软件开发中,软件测试是对软件需求分析、设计规格说明和编码的最终复审,是软件质量保证的关键步骤[2].软件测试分为黑盒测试和白盒测试.路径覆盖测试属于白盒测试,要求在测试过程中尽可能地覆盖程序的所有可达到的路径.因此,高效地自动生成覆盖目标路径的测试数据,是该类测试的关键.

* 基金项目: 国家自然科学基金(61573362); 牡丹江师范学院博士科研启动基金(MNUB201414); 牡丹江师范学院科研项目(QN201601, QY2014001)

Foundation item: National Natural Science Foundation of China (61573362); Science-Research Project for the Doctoral Foundation of Mudanjiang Normal University (MNUB201414); Science-Research Project of Mudanjiang Normal University (QN201601, QY2014001)

收稿时间: 2015-08-20; 修改时间: 2015-10-15; 采用时间: 2015-11-20; jos 在线出版时间: 2016-01-13

CNKI 网络优先出版: 2016-01-14 13:16:07, http://www.cnki.net/kcms/detail/11.2560.TP.20160114.1316.006.html

测试数据自动生成是生成一组测试数据,满足给定的测试标准的过程[3].在自动生成测试数据时,为实现路径覆盖而呈现的一个具有挑战性的问题是存在不可达路径,这些路径对于任何输入数据都是不可执行的.

在软件测试中,检测被测程序的不可达路径将有效节省测试成本和提高测试效率.到目前为止,检测不可达路径的方法有两种:动态法和静态法.动态法一般不能直接判定不可达路径,通常在测试后期采用某种手段,比如限定路径覆盖的深度和搜索次数来判定不可达路径[4].静态法主要有符号估值法[5]和检测分支相关性[6]等方法.符号法采用符号作为程序的输入变量,但很难确定哪个符号对结果能起到关键作用.Gupta[7]和Zhuang[8]等人分别提出了检测分支相关性的方法,但这些方法只针对极少的条件谓词种类,使得分支覆盖率偏低.Santone 和 Vaglini[9]、Pourvatan 和 Sirjani[10]、Chen 和 Mitra[11]、Ngo 和 Tan[12]、Delahaye 和 Botella[13]对软件测试中不可达路径的检测也进行了深入的研究.姚香娟[14]在其博士学位论文中提出了基于条件语句相关性的不可达路径自动检测方法.该方法结合上述方法的优点,能够准确地检测程序中全部或大部分的不可达路径.对于被测程序的所有路径,如果删除其中的不可达路径,自动求解覆盖可达路径的测试数据,将大大提高测试效率,缩短测试时间,节约软件开发成本.

遗传算法是自动求解路径覆盖测试数据的有效方法.Ahmed 和 Hermadi[15]、Sofokleous 和 Andreou[16]、Malhotra 和 Garg[17]、Irfan 和 Ranjan[18]、谢晓园等人[19]对遗传算法实现路径覆盖测试数据的自动生成问题进行了研究,并取得了较好的成果.文献[2]在测试数据进化生成中动态扑捉稀有数据,根据目标路径中稀有数据的贡献度设计适应度函数,得到满足路径覆盖的测试数据.但该方法在动态扑捉稀有数据时要求每一次迭代过程都重新统计节点的贡献度,从而增加了计算量.

本文采用条件语句的相关性自动检测不可达路径,在检测不可达路径的同时,统计不可达路径中节点出现的概率,从而确定必经节点(即割点),并且证明:除割点外的节点在不可达路径中出现的概率越大,则其在可达路径中出现的概率越小,该节点就越难以覆盖,即穿越该节点的个体就越少.依据文献[2]指出的稀有数据具有较高的贡献度可知:在可达路径中,穿越该节点的个体具有较高的穿越度.本文根据个体的穿越度设计适应度函数,使得穿越难以覆盖的节点具有较高的适应度,在进化中得以保留,从而提高生成测试数据的效率.

本文第 1 节介绍相关概念以及不可达路径的检测.第 2 节提出个体的穿越度,并且设计适应度函数,分析算法性能,介绍算法步骤.第 3 节是所提方法在基准程序和工业用例中的应用,并与同类方法进行对比及分析实验结果.第 4 节总结全文,并提出需要进一步研究的问题.

1 检测不可达路径确定割点

文献[14]提出采用条件语句的相关性自动检测不可达路径,从而减少目标路径,可以有效地提高测试效率.在此基础上,计算节点在路径中出现的概率,确定割点.为介绍方便起见,首先给出几个基本概念.

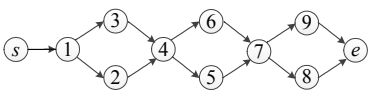
(1) 控制流图

控制流图(control flow graph,简称 CFG)是程序控制结构的图形表示,是一种具有如下结构的有向图 $G=(V,E,s,e)$,其中,V 的元素 v_i 称作 G 的节点,E 的元素 $e_{ij}=\langle v_i,v_j\rangle$ 称为 G 的边,表示从语句 v_i 到语句 v_j 存在控制流.每个程序的控制流图还包含唯一的一个入口节点 s 和一个出口节点 e.

图 1(a)为文献[14]用到的示例程序的源代码,对应的控制流图如图 1(b)所示.

```
s   void Example()
{
①   if (A>0)
②       C=A;
    else
③       C=B;
④   if (A+B−C>0)
⑤       W=A−B;
    else
⑥       W=B−A;
⑦   if (W>0)
⑧       printf ("%s", "result="\OK\"");
    else
⑨       printf ("%s", "result=\"False\"");
e }
```

(a) 程序的源代码



(b) 程序的控制流图

Fig.1 Source code of program and its CFG

图 1 程序的源代码及其控制流图

(2) 路径

路径是指给定有向图 G 的一个节点序列集$\Gamma=\{s,v_1,v_2,\ldots,v_n,e\}$,$v_i\in V(i=1,2,\ldots,n)$,且:

$$\langle s,v_1\rangle\in E,\langle v_i,v_{i+1}\rangle\in E(i=1,2,\ldots,n-1),\langle v_n,e\rangle\in E.$$

如图 1(b)所示的一条路径为$\Gamma=\{s,v_1,v_2,v_4,v_5,v_7,v_8,e\}$.

(3) 割点

给定有向图 $G=(V,E,s,e)$,$\forall v_i\in V(i=1,2,\ldots,n)$,对于任何路径$\Gamma$均有 $v_i\in\Gamma$,则称 v_i 为 G 的割点,即,在 G 中所有路径一定会经过的节点.

割点的确定方法:记被测程序为Φ,$x_1,x_2,\ldots,x_m$ 为Φ的 m 组输入向量,输入域为 D,运行Φ得到 m 条穿越路径 $\Gamma(x_1),\Gamma(x_2),\ldots,\Gamma(x_m)$,根据第 i 个输入向量 x_i 穿越第 j 个节点 v_j 的情况定义随机变量 Z_{ij}.

$$Z_{ij}=\begin{cases}1, & v_j\in\Gamma(x_i)\\ 0, & v_j\notin\Gamma(x_i)\end{cases},$$

得到如下容量为 m 的样本:

$$(z_{11},z_{12},\ldots,z_{1n}),(z_{21},z_{22},\ldots,z_{2n}),\ldots,(z_{m1},z_{m2},\ldots,z_{mn}),$$

则每个节点 v_j 被穿越的概率为

$$p_j=\frac{1}{m}\sum_{i=1}^{m}z_{ij},1\leqslant j\leqslant n \tag{1}$$

其中,n 为Φ中节点的个数,p_j 即为 v_j 在 m 次输入中被穿越的概率.

- 若 $p_j=0$,说明 v_j 在 m 次输入中均未被穿越;
- 若 $p_j=1$,说明 v_j 在 m 次输入中均被穿越,即为割点.如图 1(b)所示,节点 v_1,v_4 和 v_7 均为割点.

(4) 相关性[14]

设 v_i 和 v_j 是两个条件语句:

- 若 v_i 取真分支时 v_j 必然取真分支,则称语句对(v_i,v_j)具有真-真相关性;
- 若 v_i 取真分支时 v_j 必然取假分支,则称语句对(v_i,v_j)具有真-假相关性;
- 若 v_i 取假分支时 v_j 必然取真分支,则称语句对(v_i,v_j)具有假-真相关性;
- 若 v_i 取假分支时 v_j 必然取假分支,则称语句对(v_i,v_j)具有假-假相关性.

如图 1(b)所示,语句组(v_1,v_4)存在假-假相关性,语句组(v_1,v_4,v_7)存在真假-假相关性.

(5) 可达路径与不可达路径

对于路径Γ,如果存在 $x\in D$,使得以 x 为输入运行被测程序Φ,穿越路径$\Gamma(x)=\Gamma$,称Γ为可达路径;反之,若不存在 $x\in D$,使得$\Gamma(x)=\Gamma$,称Γ为不可达路径.本文将可达路径记为Γ,不可达路径记为 $\bar{\Gamma}$.

不可达路径的检测:设 v_i 和 v_j 是两个条件语句.如果(v_i,v_j)具有真-真(或真-假)相关性,则任何包含 v_i 真分支以及 v_j 假分支(或真分支)的路径都是不可达路径;如果(v_i,v_j)具有假-真(或假-假)相关性,则任何包含 v_i 假分支以及 v_j 假分支(或真分支)的路径都是不可达路径[14].

如图 1(b)所示的被测程序共有如下 8 条路径,分别是:

- $\Gamma_1=\{s,v_1,v_2,v_4,v_5,v_7,v_8,e\}$;
- $\Gamma_2=\{s,v_1,v_2,v_4,v_5,v_7,v_9,e\}$;
- $\bar{\Gamma}_3=\{s,v_1,v_2,v_4,v_6,v_7,v_8,e\}$;
- $\Gamma_4=\{s,v_1,v_2,v_4,v_6,v_7,v_9,e\}$;
- $\bar{\Gamma}_5=\{s,v_1,v_3,v_4,v_5,v_7,v_8,e\}$;
- $\bar{\Gamma}_6=\{s,v_1,v_3,v_4,v_5,v_7,v_9,e\}$;
- $\Gamma_7=\{s,v_1,v_3,v_4,v_6,v_7,v_8,e\}$;
- $\Gamma_8=\{s,v_1,v_3,v_4,v_6,v_7,v_9,e\}$.

由于语句组(v_1,v_4)存在假-假相关性,那么任何包含 v_1 的假分支和 v_4 的真分支的路径都是不可达路径,因此,

$\bar{\Gamma}_5$ 和 $\bar{\Gamma}_6$ 是不可达路径.同理,由于语句组(v_1,v_4,v_7)之间存在真假-假相关性,那么任何包含 v_1 的真分支、v_4 的假分支和 v_7 的真分支的路径都是不可达路径,因此,$\bar{\Gamma}_3$ 是不可达路径.其余 5 条路径均为可达路径.

2 基于不可达路径节点出现概率的测试数据进化生成方法

2.1 个体的穿越度

在应用遗传算法生成测试数据的过程中,有大量测试数据运行被测程序,分别穿越被测程序可达路径中的不同节点.这些节点(除割点外)在可达路径中出现的概率不同.节点在可达路径中出现的概率越小,则生成穿越该节点的测试数据(即进化个体)越难,依此定义个体的穿越度.

以个体 x_i 为测试数据运行被测程序 Φ,得到其穿越路径 $\Gamma(x_i)$,该路径包含 n' 个节点,删除割点后,剩余节点数为 n,$n\leqslant n'$.计算该个体的穿越度 $P(x_i)$ 为

$$P(x_i)=\sum_{j=1}^{n}\{[1-P(v_j\mid\Gamma')]\cdot\theta_j\} \tag{2}$$

其中,节点 $v_j\in\Gamma(x_i)$,j=1,2,…,n;Γ' 为 Φ 的所有可达路径集合;$P(v_j|\Gamma')$ 为 v_j 在 Γ' 中出现的概率;θ_j 表示 v_j 是否为目标路径 Γ_0 中的节点,即:

$$\theta_j=\begin{cases}1, & v_j\in\Gamma_0\\ 0, & v_j\notin\Gamma_0\end{cases}.$$

由公式(2)可知,计算个体的穿越度需要统计节点在可达路径集中出现的概率.易知:对于复杂的被测程序,含有较多的不可达路径,在生成路径覆盖测试数据时,为了提高生成效率,总是先检测不可达路径并将其删除.由于程序节点在可达路径集中出现的概率与其在不可达路径集中出现的概率存在一定的对应关系,详见定理 1,因此在检测不可达路径时,统计节点在不可达路径集中出现的概率,利用节点在不可达路径集中出现的概率计算个体的穿越度,将大大减少计算量,提高测试效率.

定理 1. 节点 v_i(除割点外)在不可达路径集 $\bar{\Gamma}'$ 中出现的概率 $P(v_i\mid\bar{\Gamma}')$ 越大,则其在可达路径集 Γ' 中出现的概率 $P(v_i|\Gamma')$ 越小;反之,节点 v_i(除割点外)在不可达路径集 $\bar{\Gamma}'$ 中出现的概率 $P(v_i\mid\bar{\Gamma}')$ 越小,则其在可达路径集 Γ' 中出现的概率 $P(v_i|\Gamma')$ 越大.

证明:对于被测程序 Φ,令 $v_i(i=1,2,\ldots,n)$ 为 Φ 的节点(除割点外),Γ' 为可达路径集,$\bar{\Gamma}'$ 为不可达路径集,$P(v_i\mid\bar{\Gamma}')$ 为 v_i 在 $\bar{\Gamma}'$ 中出现的概率,$P(v_i|\Gamma')$ 为 v_i 在 Γ' 中出现的概率,$P(v_i)$ 为 v_i 在所有路径集中出现的概率,$P(\Gamma')$ 为可达路径集概率,即,可达路径集含有路径条数占所有路径集含有路径条数的比例.

节点 v_i 在不可达路径集 $\bar{\Gamma}'$ 中出现的概率 $P(v_i\mid\bar{\Gamma}')$ 为

$$P(v_i\mid\bar{\Gamma}')=\frac{P(v_i\cdot\bar{\Gamma}')}{P(\bar{\Gamma}')}=\frac{P(\bar{\Gamma}'\mid v_i)\cdot P(v_i)}{P(\bar{\Gamma}')} \tag{3}$$

则有:

$$P(\bar{\Gamma}'\mid v_i)=\frac{P(v_i\mid\bar{\Gamma}')\cdot P(\bar{\Gamma}')}{P(v_i)} \tag{4}$$

节点 v_i 在可达路径集 Γ' 中出现的概率 $P(v_i|\Gamma')$ 为

$$P(v_i\mid\Gamma')=\frac{P(v_i\cdot\Gamma')}{P(\Gamma')}=\frac{P(\Gamma'\mid v_i)\cdot P(v_i)}{P(\Gamma')}=\frac{\{1-P(\bar{\Gamma}'\mid v_i)\}\cdot P(v_i)}{P(\Gamma')} \tag{5}$$

将公式(4)代入公式(5),可得:

$$P(v_i\mid\bar{\Gamma}')=\frac{\left\{1-\dfrac{P(v_i\mid\bar{\Gamma}')\cdot P(\bar{\Gamma}')}{P(v_i)}\right\}\cdot P(v_i)}{P(\Gamma')}=\frac{P(v_i)-P(v_i\mid\bar{\Gamma}')\cdot P(\bar{\Gamma}')}{P(\Gamma')}=\frac{P(v_i)}{P(\Gamma')}-\frac{1-P(\Gamma')}{P(\Gamma')}\cdot P(v_i\mid\bar{\Gamma}') \tag{6}$$

即:

$$P(v_i \mid \bar{\Gamma}') = \alpha - \beta \cdot P(v_i \mid \bar{\Gamma}') \tag{7}$$

其中,$\alpha = \frac{P(v_i)}{P(\Gamma')}, \beta = \frac{1-P(\Gamma')}{P(\Gamma')}$.$P(v_i)$和 $P(\Gamma')$随着被测程序Φ的给定就已经确定,因此,上式中α和β为常数.由此可知:节点在不可达路径集中出现的概率越大,其在可达路径集中出现的概率越小.证毕. □

下面通过实例进一步验证定理 1 的正确性.如图 1(b)中去掉割点 v_1,v_4 和 v_7,计算其余节点 v_i(i=2,3,5,6,8,9)在不可达路径集和可达路径集中出现的概率.

先统计节点在不可达路径集中出现的概率.由第 1 节的示例可知:不可达路径集 $\bar{\Gamma}' = \{\bar{\Gamma}_3, \bar{\Gamma}_5, \bar{\Gamma}_6\}$,即,有 3 条不可达路径.因为 v_2 仅在 $\bar{\Gamma}_3$ 中出现,即 $v_2 \in \bar{\Gamma}_3$, 所以 $P(v_2 \mid \bar{\Gamma}') = \frac{1}{3}$, 因为 v_3 同时在 $\bar{\Gamma}_5$ 和 $\bar{\Gamma}_6$ 中出现,即 $v_3 \in \bar{\Gamma}_5 \cap \bar{\Gamma}_6$, 所以 $P(v_3 \mid \bar{\Gamma}') = \frac{2}{3}$. 同理, $P(v_5 \mid \bar{\Gamma}') = \frac{2}{3}, P(v_6 \mid \bar{\Gamma}') = \frac{1}{3}, P(v_8 \mid \bar{\Gamma}') = \frac{2}{3}, P(v_9 \mid \bar{\Gamma}') = \frac{1}{3}$.

有了上述节点在不可达路径集中出现的概率,可依据公式(7)计算节点在可达路径集中出现的概率.

以节点 v_2 为例, $v_2 \in \Gamma_1 \cap \Gamma_2 \cap \bar{\Gamma}_3 \cap \Gamma_4$, 所以 v_2 在所有路径中出现的概率为 $P(v_2) = \frac{4}{8}$, 可达路径集含有路径条数占所有路径集含有路径条数的比例为 $\frac{5}{8}$, 即可达路径集的概率为 $P(\Gamma') = \frac{5}{8}$. 由上面计算得到 $P(v_2 \mid \bar{\Gamma}') = \frac{1}{3}$, 依据公式(7)可得: $P(v_2 \mid \Gamma') = \alpha - \beta \cdot P(v_2 \mid \bar{\Gamma}') = \frac{P(v_2)}{P(\Gamma')} - \frac{1-P(\Gamma')}{P(\Gamma')} \cdot P(v_2 \mid \bar{\Gamma}') = \frac{4/8}{5/8} - \frac{1-5/8}{5/8} \cdot \frac{1}{3} = \frac{3}{5}$.

同理可计算 $P(v_3 \mid \Gamma') = \frac{2}{5}, P(v_5 \mid \Gamma') = \frac{2}{5}, P(v_6 \mid \Gamma') = \frac{3}{5}, P(v_8 \mid \Gamma') = \frac{2}{5}, P(v_9 \mid \Gamma') = \frac{3}{5}$.

通过手工计算验证公式(7)计算结果的正确性.仍以节点 v_2 为例,在可达路径集Γ=$\{\Gamma_1,\Gamma_2,\Gamma_4,\Gamma_7,\Gamma_8\}$中,$v_2$ 仅出现在 3 条路径Γ_1,Γ_2 和Γ_4 中,即 $v_2 \in \Gamma_1 \cap \Gamma_2 \cap \Gamma_4$,则 $P(v_2 \mid \Gamma') = \frac{3}{5}$, 与公式(7)计算结果相同.同样,节点 v_3,v_5,v_6,v_8 和 v_9 在可达路径集中出现的概率均与公式(7)的计算结果相同.由此说明,依据节点在不可达路径集中出现的概率 $P(v_i \mid \bar{\Gamma}')$ 计算其在可达路径集中出现的概率 $P(v_i|\Gamma')$是正确的.

综上,通过理论证明和实例验证了定理 1 的正确性.

推论 1. 节点 v_i 在不可达路径集 $\bar{\Gamma}'$ 中出现的概率 $P(v_i \mid \bar{\Gamma}')$ 越大,则穿越该节点的个体的穿越度越高.

证明:由定理 1 可知,节点 v_i 在不可达路径集 $\bar{\Gamma}'$ 中出现的概率 $P(v_i \mid \bar{\Gamma}')$ 越大,则其在可达路径集Γ'中出现的概率 $P(v_i|\Gamma')$越小.根据个体的穿越度计算公式(2)可知:在可达路径中,穿越该节点的个体具有较高的穿越度. □

根据定理 1 和推论 1,依据节点 v_j 在不可达路径集 $\bar{\Gamma}'$ 中出现的概率 $P(v_j \mid \bar{\Gamma}')$ 定义新的个体穿越度 $P'(x_i)$.

以个体 x_i 为测试数据运行被测程序Φ,得到其穿越路径$\Gamma(x_i)$,该路径包含 n'个节点,删除割点后,剩余节点数为 n,$n \leqslant n'$.计算该个体的穿越度 $P'(x_i)$为

$$P'(x_i) = \sum_{j=1}^{n} (P(v_j \mid \bar{\Gamma}') \cdot \theta_j) \tag{8}$$

其中,节点 $v_j \in \Gamma(x_i)$,j=1,2,…,n; $\bar{\Gamma}'$ 为Φ的所有不可达路径集合; $P(v_j \mid \bar{\Gamma}')$ 为 v_j 在 $\bar{\Gamma}'$ 中出现的概率;θ_j 表示 v_j 是否为目标路径Γ_0 中的节点,即:

$$\theta_j = \begin{cases} 1, & v_j \in \Gamma_0 \\ 0, & v_j \notin \Gamma_0 \end{cases}.$$

实际上,个体 x_i 的穿越度 $P'(x_i)$为其穿越路径与目标路径相同的节点(除割点外)在不可达路径中出现的概率之和.

如图 1(b)所示中目标路径Γ_0=$\{s,v_1,v_3,v_4,v_6,v_7,v_8,e\}$.若种群规模为 4,进化到某一代时 4 个个体穿越的路径分别为:

- $\Gamma(x_1)$=$\{s,v_1,v_2,v_4,v_5,v_7,v_8,e\}$;

- $\Gamma(x_2)=\{s,v_1,v_3,v_4,v_6,v_7,v_9,e\}$;
- $\Gamma(x_3)=\{s,v_1,v_2,v_4,v_6,v_7,v_9,e\}$;
- $\Gamma(x_4)=\{s,v_1,v_2,v_4,v_5,v_7,v_9,e\}$.

其中,节点 v_1,v_4 和 v_7 为割点.

- 对个体 x_1,除割点外,仅穿越目标路径中的节点 v_8.由前面计算可知, $P(v_8 \mid \overline{\Gamma}')=\frac{2}{3}$, 其穿越度 $P'(x_1)=\frac{2}{3}$;
- 对个体 x_2,除割点外,穿越目标路径中的节点 v_3 和 v_6,由前面计算可知, $P(v_3 \mid \overline{\Gamma}')=\frac{2}{3}, P(v_6 \mid \overline{\Gamma}')=\frac{1}{3}$, 其穿越度 $P'(x_2)=\frac{2}{3}+\frac{1}{3}=1$;
- 同理可得 $P'(x_3)=\frac{1}{3}, P'(x_4)=0$.

由此可见:根据个体的穿越度,可以有效地区分个体的优劣.

由定理 1 及推论 1 可知:在计算个体的穿越度时,可以通过其穿越节点在不可达路径中出现的概率得到,这样要比计算其穿越节点在可达路径中出现的概率简便得多.

2.2 设计适应度函数

采用遗传算法自动生成测试数据,适应度函数体现了自然进化中的优胜劣汰原则.对于优化问题,适应度函数是根据目标函数确定的.因此,适应度函数的设计在遗传算法中发挥着重要作用.到目前为止,针对路径覆盖的测试数据进化生成,适应度函数的设计方法主要有 3 种:分支距离(*branch_distance*)、层接近度(*approach_level*)以及二者相结合的方法.本文采用分支距离与层接近度结合的方法设计适应度函数.

假设个体 x_i 穿越的路径为 $\Gamma(x_i)$,目标路径为 Γ_0.x_i 的层接近度为 $approach\_level(x_i)$,计算方法采用统计个体穿越路径 $\Gamma(x_i)$ 与目标路径 Γ_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 $branch\_distance(x_i)$,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 $1.001^{-branch\_distance(x_i)}$ 进行规范化,其值越大,个体越优[20].

于是,个体 x_i 的适应度 $fit(x_i)$ 可表示为

$$fit(x_i)=approach\_level(x_i)+1.001^{-branch\_distance(x_i)} \tag{9}$$

个体 x_i 的穿越度越大,其适应度应越大.因此,将个体的穿越度作为权重调整原来的适应度,得到新的适应度函数为

$$f(x_i)=fit(x_i)\cdot P'(x_i)=fit(x_i)\cdot\sum_{j=1}^{n}(P(v_j \mid \overline{\Gamma}')\cdot\theta_j) \tag{10}$$

从而得到基于不可达路径节点出现概率的路径覆盖测试数据自动生成方法.

2.3 性能分析

假设种群进化到某一代,有两个个体 x_1 和 x_2 穿越的路径分别为 $\Gamma(x_1)$ 和 $\Gamma(x_2)$.$\Gamma(x_1)$ 和 $\Gamma(x_2)$ 分别与目标路径相比,两条路径穿越目标路径的节点只有第 i 和第 j 两个节点不同,即,个体 x_1 穿越目标路径的第 i 个节点,个体 x_2 穿越目标路径的第 j 个节点,其余节点均相同.按照公式(9)适应度计算方法可得 $fit(x_1)=fit(x_2)$.如果第 i 个节点在不可达路径中出现的概率大于第 j 个节点在不可达路径中出现的概率,即 $P(v_i \mid \overline{\Gamma}')>P(v_j \mid \overline{\Gamma}')$,则由公式(8)可得:个体 x_1 的穿越度大于个体 x_2 的穿越度,即 $P'(x_1)>P'(x_2)$.也就是说,在传统适应度计算相等的情况下,由于两个个体的穿越度不同,导致本文方法对应的适应度调整权重不同.按照本文方法,由公式(10)可得个体 x_1 和 x_2 的适应度分别为:

- $f(x_1)=fit(x_1)\cdot P'(x_1)$;
- $f(x_2)=fit(x_2)\cdot P'(x_2)$.

即:两个个体的适应度不同,$f(x_1)>f(x_2)$.

因此,根据个体的穿越度调整个体的适应度,可以使穿越难以覆盖节点的个体获得较高的适应度,在后续进化过程中得以有效的保留,从而提高了生成覆盖目标路径测试数据的效率.

本文采用遗传算法生成测试数据,算法的时间复杂度主要考虑适应度函数的计算.由公式(10)可知:适应度的计算主要取决于个体穿越度的计算,此值在于统计不可达路径节点出现的概率.节点在不可达路径中出现的概率在检测不可达路径的过程中得到统计计算,其概率值始终保持不变,无需重复计算.因此,本文方法降低了算法的时间复杂度.

2.4 算法步骤

步骤 1. 对算法所需的控制参数赋值,插装被测程序;

步骤 2. 检测不可达路径,确定割点;

步骤 3. 初始化种群;

步骤 4. 对进化个体解码,执行被测程序;

步骤 5. 判断进化个体是否穿越目标路径:若是,保存进化个体,转至步骤 9;

步骤 6. 判断是否达到最大进化代数:若达到,转至步骤 9;

步骤 7. 按公式(10)计算进化个体的适应度函数;

步骤 8. 对种群实施选择、交叉和变异操作,生成子代种群,转至步骤 4;

步骤 9. 停止种群的进化,解码保存的进化个体,输出测试数据.

3 实 验

本节通过实验进一步检验本文所提方法的有效性.首先给出实验需要验证的问题;然后介绍两组实验所用的程序,第 1 组为基准程序,第 2 组为工业用例,并给出实验结果和分析.

3.1 需要验证的问题

本文研究的目的是基于不可达路径节点出现的概率设计适应度函数,进一步提高路径覆盖测试数据的生成效率,需要验证的问题如下:

问题 1. 利用本文方法设计适应度函数,能否提高生成测试用例的效率?

验证方法:通过比较本文方法与同类方法的评价次数和运行时间进行验证.

问题 2. 利用本文方法生成测试用例的有效性如何?

验证方法:主要通过生成测试用例的成功率进行验证.

问题 3. 测试数据的输入范围是否影响本文方法的有效性?

验证方法:在搜索空间爆炸性增长的情况下,进一步验证本文方法性能的优越性.

3.2 实验程序

为了验证本文方法的性能,每组实验结果都与同类方法(即文献[2,21]中方法)的结果进行比较分析.同类方法的选择依据是均使用分支距离和层接近度相结合作为适应度函数,并且均采用遗传算法进化生成测试数据,与其对比,可以更好地验证本文方法生成测试数据的效率.为了降低随机性误差,3 种方法采用相同的实验参数、相同的种群规模和相同的初始种群,比较 3 种方法找到覆盖目标路径测试数据的评价次数、运行时间和成功率.实验条件:Windows XP 操作系统,VC++6.0 仿真环境,计算机主频 2.80GHz,内存 2GB.

3.2.1 两个基准程序

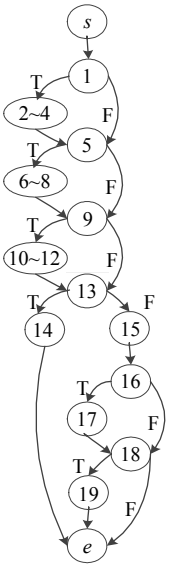
两个基准程序分别选择三角形分类程序和冒泡排序程序.首先,检测两个程序的不可达路径,确定割点;然后,采用遗传算法自动生成覆盖目标路径的测试数据;最后,将本文方法与同类方法对比,以验证本文方法的有效性.

(一) 三角形分类程序

三角形分类程序的源代码与控制流图如图 2 所示.

```
s   void Triangle(int a,int b,int c)
1     { if (a>b)
2         {t=a;
3           a=b;
4           b=t;}
5         if (a>c)
6         {t=a;
7           a=c;
8           c=t;}
9         if (b>c)
10          {t=b;
11          b=c;
12          c=t;}
13    if (a+b≤c)
14          strcpy (Type, “NOT TRIANGLE”);
          else
15      {strcpy (Type, “TRIANGLE”);
16        if (a==b && b==c)
17          strcpy (Type, “EQUILATERAL”);
18        if ((a==b||b==c) && a!=c)
19          strcpy (Type, “ISOSCELES”); }
e     }
```

(a) 三角形分类程序的源代码



(b) 三角形分类程序的控制流

Fig.2 Source code of trangle-classfying program and its CFG

图 2 三角形分类程序源代码及其控制流图

(1) 判定条件语句的相关性.

程序输入变量为{*a*,*b*,*c*},取值范围为 0~300 之间的整数,样本容量为 50,设定{*a*,*b*,*c*}能构成等边或等腰三角形的概率均为 0.1.三角形分类程序中条件语句间的相关性实验数据见表 1[14].

Table 1 Correlations of different conditional statements in trangle-classfying program

表 1 三角形分类程序中条件语句间的相关性

条件语句	相关性类型	条件语句	相关性类型
(16,18)	T→F	(5,13,16)	TF→F
(1,5,9)	TT→F	(9,13,16)	TF→F
(1,13,16)	TF→F	(1,5,13,18)	TTF→F
(5,9,13)	TF→T	(5,9,13,18)	TTF→F,TFF→T

(2) 由条件语句间的相关性检测三角形分类程序中的不可达路径,确定割点,实验数据见表 2.

Table 2 Infeasible paths of trangle-classfying program

表 2 三角形分类程序中的不可达路径

序号	路径
1	$\bar{\Gamma}_1=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
2	$\bar{\Gamma}_2=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
3	$\bar{\Gamma}_3=\{s,v_1,v_2,v_3,v_4,v_5,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
4	$\bar{\Gamma}_4=\{s,v_1,v_2,v_3,v_4,v_5,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
5	$\bar{\Gamma}_5=\{s,v_1,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
6	$\bar{\Gamma}_6=\{s,v_1,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
7	$\bar{\Gamma}_7=\{s,v_1,v_5,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
8	$\bar{\Gamma}_8=\{s,v_1,v_5,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},v_{19},e\}$
9	$\bar{\Gamma}_9=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{13},v_{14},e\}$

Table 2 Infeasible paths of trangle-classfying program (Continued)

表 2 三角形分类程序中的不可达路径(续)

序号	路径
10	$\bar{\Gamma}_{10}=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{18},v_{19},e\}$
11	$\bar{\Gamma}_{11}=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},e\}$
12	$\bar{\Gamma}_{12}=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{18},e\}$
13	$\bar{\Gamma}_{13}=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},e\}$
14	$\bar{\Gamma}_{14}=\{s,v_1,v_2,v_3,v_4,v_5,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},e\}$
15	$\bar{\Gamma}_{15}=\{s,v_1,v_2,v_3,v_4,v_5,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},e\}$
16	$\bar{\Gamma}_{16}=\{s,v_1,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},e\}$
17	$\bar{\Gamma}_{17}=\{s,v_1,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{18},v_{19},e\}$
18	$\bar{\Gamma}_{18}=\{s,v_1,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{18},e\}$
19	$\bar{\Gamma}_{19}=\{s,v_1,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},e\}$
20	$\bar{\Gamma}_{20}=\{s,v_1,v_5,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{17},v_{18},e\}$
21	$\bar{\Gamma}_{21}=\{s,v_1,v_2,v_3,v_4,v_5,v_6,v_7,v_8,v_9,v_{13},v_{15},v_{16},v_{18},v_{19},e\}$
22	$\bar{\Gamma}_{22}=\{s,v_1,v_5,v_6,v_7,v_8,v_9,v_{10},v_{11},v_{12},v_{13},v_{15},v_{16},v_{18},v_{19},e\}$

三角形分类程序中,全部路径 40 条,不可达路径 22 条.其中,割点为 v_1,v_5,v_9 和 v_{13}.

(3) 统计三角形分类程序节点(不考虑割点)在不可达路径集中出现的概率 $P(v_i \mid \bar{\Gamma}')$, 实验数据如下:

$$P(v_2 \mid \bar{\Gamma}')=P(v_3 \mid \bar{\Gamma}')=P(v_4 \mid \bar{\Gamma}')=\frac{12}{22},P(v_6 \mid \bar{\Gamma}')=P(v_7 \mid \bar{\Gamma}')=P(v_8 \mid \bar{\Gamma}')=\frac{15}{22},$$

$$P(v_{10} \mid \bar{\Gamma}')=P(v_{11} \mid \bar{\Gamma}')=P(v_{12} \mid \bar{\Gamma}')=\frac{10}{22},P(v_{19} \mid \bar{\Gamma}')=\frac{12}{22},P(v_{14} \mid \bar{\Gamma}')=\frac{1}{22},$$

$$P(v_{15} \mid \bar{\Gamma}')=P(v_{16} \mid \bar{\Gamma}')=\frac{21}{22},P(v_{17} \mid \bar{\Gamma}')=\frac{15}{22},P(v_{18} \mid \bar{\Gamma}')=\frac{21}{22}.$$

(4) 应用遗传算法生成测试数据.个体采用二进制编码,轮盘赌选择、单点交叉、单点变异,交叉和变异概率分别为 0.9 和 0.3.根据个体的穿越度计算适应度函数,对三角形分类程序中的可达路径生成测试数据.为确保实验条件基本相同,本组实验每种方法均运行 15 次,目标路径选择等边三角形路径Γ_0={s,v_1,v_5,v_9,v_{13},v_{15},v_{16},v_{17},v_{18},e},输入数据范围分 3 种,选择不同的种群规模和最大终止代数,本文方法与文献[2,21]中方法进行比较,实验结果见表 3.

Table 3 Experimental results of trangle-classfying program

表 3 三角形分类程序实验结果

实验设置			本文方法			文献[2]方法			文献[21]方法		
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(%)
[1,256]	50	10 000	10 632.2	0.000 9	100	10 910.7	0.001 2	100	318 240.0	0.033 5	100
[1,512]	100	20 000	29 180.3	0.003 2	100	30 540.7	0.004 1	100	1 225 290.3	0.168 6	86
[1,1024]	200	50 000	89 170.6	0.009 8	100	98 440.3	0.015 5	100	4 727 740.7	0.590 5	80

由表 3 可以看出:

① 从评价次数来看,在 3 种方法中,本文方法以最少的评价次数成功生成了测试数据.如在数据范围为[1,512]时,本文方法的评价次数为 29 180.3,文献[2]中方法的评价次数为 30 540.7,文献[21]中方法的评价次数为 1 225 290.3.本文方法与文献[2]中方法均采用调整适应度函数的方法,所以评价次数相差不大;但文献[21]中方法约是本文方法的 42 倍.另外,两种数据范围得到的结果类似,说明本文方法的性能明显优于同类方法;

② 从运行时间来看,本文方法的运行时间明显少于文献[2,21]中方法.特别是,随着数据范围、种群规模和最大代数的增加,本文方法的优势更加明显.这是因为,本文方法只需计算 1 次节点在不可达路径中出现的概率就可以调整适应度函数,能够较快地生成最优解,说明本文方法生成测试数据的效率明显高于同类方法;

③ 从成功率来看,随着数据范围、种群规模和最大代数的增加,本文方法的评价代数和运行时间也有所增加.但是本文方法仍能有效生成测试数据,而且生成测试数据的成功率为 100%,说明本文方法具有有效性.

为了进一步验证本文方法性能的优越性,在搜索空间呈爆炸性增长的情况下,选择三角形分类程序在更大的输入范围进行实验.在种群规模不变的情况下,仍以等边三角形路径为目标路径,实验设置及结果见表 4.

Table 4 Experimental design and results of trangle-classfying program in large search space

表 4 大搜索空间下三角形分类程序实验设置及结果

实验设置			本文方法			文献[2]方法			文献[21]方法		
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	60 000	172 098.0	0.009 9	100	190 400.0	0.013 6	100	9 892 247.7	1.817 7	60
[1,4096]	200	70 000	236 012.0	0.087 2	100	304 800.0	0.110 4	100	13 604 260.3	3.675 8	26.67
[1,8192]	200	80 000	583 921.2	0.361 7	100	691 840.0	0.507 6	100	15 469 982.0	4.281 2	6.67

由表 4 可以看出:在数据范围变大,导致搜索空间爆炸性增长的情况下,本文方法的性能如下:

① 文献[21]中方法在搜索空间爆炸性增长的情况下,生成测试数据的概率逐渐降低.数据范围在[1,8192]情况下,成功率仅有 6.67%.说明文献[21]中方法在搜索空间爆炸性增长的情况下很难奏效;

② 本文方法和文献[2]中方法的成功率均达到了 100%,但在评价次数和运行时间上,本文方法明显少于同类方法,说明本文方法对测试数据的生成效率仍有所改进.

(二) 冒泡排序程序

冒泡排序也是验证实验的基准程序之一,其特点包含循环结构,功能是对数据进行排序.程序输入变量为 8 个[1,65535]范围内的数据,该程序包含两个相互嵌套的循环语句,其中嵌套一个条件语句.种群规模为 50,最大终止代数为 1 000,3 种方法独立运行 15 次,得到的实验结果见表 5.

Table 5 Experimental results of bubble sort program

表 5 冒泡排序程序实验结果

	评价次数平均值	运行时间平均值(s)	成功率(%)
本文方法	19 386.4	0.089 3	100
文献[2]方法	23 206.7	0.140 9	100
文献[21]方法	50 593.3	0.178 1	86.7

由表 5 可以看出:从评价次数来看,在含有循环结构的基准程序中,本文方法的评价次数更具优势,还不足文献[21]方法的一半,而且对于文献[2]方法来说,本文方法的评价次数也较少;从运行时间来看,由于本文方法计算量的减少,运行时间明显少于同类方法,本文方法的运行时间为 0.089 3,文献[2]中方法的运行时间为 0.140 9,文献[21]中方法的运行时间为 0.178 1;从成功率来看,本文方法和文献[2]中方法均达到了 100%,而文献[21]中方法只有 86.7%,说明本文方法可以有效地生成测试数据.

3.2.2 工业用例

为了进一步验证本文方法的有效性,选择 4 个工业用例进行实验.程序描述及参数设置见表 6.

Table 6 Parameter settings of industrial cases

表 6 工业用例参数设置

被测程序	代码行数	种群规模	最大代数	目标路径节点数
Space (fixsgrid)	115	100	1 500	12
repalce	564	200	30 000	85
sed	8 063	200	50 000	385
flex	11 783	400	50 000	543

本组实验设定每种方法各运行 50 次,分析评价次数、运行时间和生成测试数据的成功率.实验结果见表 7.

由表 7 可以看出:从评价次数来看,在工业用例中,本文方法的评价次数仍少于同类方法;从运行时间来看,本文方法的优势更加明显,对于复杂用例 flex 来说,本文方法的平均运行时间为 62.321 7s,文献[2]中方法的平均

运行时间为 89.741 2s,文献[21]中方法的平均运行时间为 103.436 7s,本文方法比文献[2]中方法少用 27.419 5s,比文献[21]中方法少用 41.115 0s,可见,本文方法在运行时间上的优势非常明显;从成功率来看,本文方法和文献[2]中方法均达到了 100%,而文献[21]中方法对于程序 repalce,sed 和 flex 只有 66%、46%和 36%的成功率,说明本文方法能够很好地生成测试数据.

Table 7 Experimental results of industrial cases

表 7 工业用例实验结果

被测程序	本文方法			文献[2]方法			文献[21]方法		
	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(%)
Space (fixsgrid)	5 106.3	0.027 6	100	5 498.9	0.031 2	100	12 215.7	0.039 9	100
repalce	803 126.4	21.318 1	100	871 308.0	26.511 2	100	1 953 761.6	41.881 3	66
sed	1 398 272.4	48.312 5	100	1 425 717.3	59.812 4	100	4 089 124.5	70.670 9	46
flex	6 132 570.6	62.321 7	100	6 824 130.8	89.741 2	100	13 504 177.0	103.436 7	36

由以上的基准程序和工业用例实验,充分验证了本文方法生成覆盖路径的测试数据的有效性,并且验证了本文方法能够提高生成测试数据的效率.

4 结 论

本文给出了一种依据不可达路径节点出现概率计算个体穿越度,进而调整个体适应度函数的路径覆盖测试数据的生成方法.本文方法在检测不可达路径时,统计节点在不可达路径中出现的概率,从而计算个体的穿越度,设计适应度函数,使得穿越度较高的个体在进化中得到保留,有效提高了生成测试数据的效率.实验结果表明,本文方法与同类方法相比生成测试数据的效率较高.

本文研究的是覆盖单目标路径测试数据进化生成问题,今后将进一步研究覆盖多目标路径测试数据生成问题.

致谢 在此,向对本文工作给予支持和建议的评审专家表示衷心的感谢.

References:

[1] Lenz AR, Pozo A, Vergilio SR. Linking software testing results with a machine learning approach. Engineering Applications of Artificial Intelligence, 2013,26:1631−1640. [doi: 10.1016/j.engappai.2013.01.008]

[2] Zhang Y, Gong DW. Evolutionary generation of test data for paths coverage based on scarce data capturing. Chinese Journal of Computers, 2013,36(12):2429−2440 (in Chinese with English abstract).

[3] Suresh Y, Rath SK. A genetic algorithm based approach for test data generation in basis path testing. The Int'l Journal of Soft Computing and Software Engineering, 2013,3(3):326−332. [doi: 10.7321/jscse.v3.n3.49]

[4] Hedley D, Hennell MA. The causes and effects of infeasible paths in computer programs. In: Proc. of the 8th Int'l Conf. on Software Engineering. Los Alamitos: IEEE Computer Society Press, 1985. 259−266.

[5] Zhang J, Wang XX. A constraint solver and its application to path feasibility analysis. Int'l Journal of Software Engineering and Knowledge Engineering, 2001,11(2):139−156. [doi: 10.1142/S0218194001000487]

[6] Forgacs I, Bertolino A. Feasible test path selection by principal slicing. In: Proc. of the 6th European Software Engineering Conf. on Held Jointly with the 5th ACM Symp. on the Foundations of Software Engineering. 1997. 378−394. [doi: 10.1007/3-540-63531-9_26]

[7] Gupta R, Gopinath P. Correlation analysis techniques for refining execution time estimates of real-time applications. In: Proc. of the 11th IEEE Workshop on Real-Time Operating Systems and Software. 1994. 54−58. [doi: 10.1109/RTOSS.1994.292561]

[8] Zhuang XT, Zhang T, Pande S. Using branch correlation to identify infeasible paths for anomaly detection. In: Proc. of the 39th Annual IEEE/ACM Int'l Symp. on Microarchitecture. 2006. 113−122. [doi: 10.1109/MICRO.2006.48]

[9] Santone A, Vaglini G. Formula-Based abstractions and symbolic execution for model checking programs. Microprocessors and Microsystems, 2004,28:69−76. [doi: 10.1016/S0141-9331(03)00127-3]

[10] Pourvatan B, Sirjani M, Hojjat H, Arbab F. Automated analysis of reo circuits using symbolic execution. Electronic Notes in Theoretical Computer Science, 2009,255:137−158. [doi: 10.1016/j.entcs.2009.10.029]

[11] Chen T, Mitra T, Roychoudhury A, Suhendra V. Exploiting branch constraints without exhaustive path enumeration. In: Proc. of the 5th Int'l Workshop on Worst-Case Execution Time Analysis. 2005. 46−49. https://www.researchgate.net/publication/30815525_Exploiting_Branch_Constraints_without_Exhaustive_Path_Enumeration

[12] Ngo MN, Tan HBK. Heuristics-Based infeasible path detection for dynamic test data generation. Information and Software Technology, 2008,50(7-8):641−655. [doi: 10.1016/j.infsof.2007.06.006]

[13] Delahaye M, Botella B, Gotlieb A. Infeasible path generalization in dynamic symbolic execution. Information and Software Technology, 2015,58:403−418. [doi: 10.1016/j.infsof.2014.07.012]

[14] Yao XJ. Theory of evolutionary generation of test data for complex software and applications [Ph.D. Thesis]. Beijing: China University of Mining and Technology, 2011 (in Chinese with English abstract).

[15] Ahmed MA, Hermadi I. GA-Based multiple paths test data generator. Computers and Operations Research, 2008,35(10): 3107−3124. [doi: 10.1016/j.cor.2007.01.012]

[16] Sofokleous AA, Andreou AS. Automatic, evolutionary test data generation for dynamic software testing. The Journal of System and Software, 2008,81(11):1883−1898. [doi: 10.1016/j.jss.2007.12.809]

[17] Malhotra R, Garg M. An adequacy based test data generation technique using genetic algorithms. Journal of Information Processing Systems, 2011,7(2):363−384. [doi: 10.3745/JIPS.2011.7.2.363]

[18] Irfan S, Ranjan P. A concept of out degree in CFG for optimal test data using genetic algorithm. In: Proc. of the 1st Int'l Conf. on Recent Advances in Information Technology. 2012. 436−441. [doi: 10.1109/RAIT.2012.6194634]

[19] Xie XY, Xu BW, Shi L, Nie CH. Genetic test case generation for path-oriented testing. Ruan Jian Xue Bao/Journal of Software, 2009,20(12):3117−3136 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/580.htm [doi: 10.3724/SP.J.1001.2009.00580]

[20] Zhang Y. Theories and methods of evolutionary generation of test data for path coverage [Ph.D. Thesis]. Beijing: China University of Mining and Technology, 2012 (in Chinese with English abstract).

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

附中文参考文献:

[2] 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.计算机学报,2013,36(12):2429−2440.

[14] 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].北京:中国矿业大学,2011.34−53.

[19] 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. http://www.jos.org.cn/1000-9825/580.htm [doi: 10.3724/SP.J.1001.2009.00580]

[20] 张岩.路径覆盖测试数据进化生成理论与方法[博士学位论文].北京:中国矿业大学,2012.16−20.

夏春艳(1980－),女,黑龙江桦川人,讲师,主要研究领域为数据挖掘与信息处理,基于搜索的软件工程.

宋丽(1975－),女,副教授,主要研究领域为基于搜索的软件工程.

张岩(1972－),女,博士,教授,CCF 会员,主要研究领域为基于搜索的软件工程.