软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, 2022,33(2):434−454 [doi: 10.13328/j.cnki.jos.006149]

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

结合关键点概率与路径相似度的多路径覆盖策略*

钱忠胜, 祝 洁, 朱懿敏, 俞情媛, 李端明, 宋 佳

(江西财经大学 信息管理学院, 江西 南昌 330013)

通信作者: 钱忠胜, E-mail: changesme@163.com

摘 要: 利用多种群遗传算法解决多路径覆盖问题, 是测试数据自动生成领域一个重要的研究方向. 为了提高多路径覆盖测试数据自动生成的效率, 提出一种将关键点概率和路径相似度相结合的多路径覆盖策略. 首先, 将理论路径划分成易覆盖、难覆盖及不可达路径; 然后, 通过易覆盖路径统计关键点概率, 依此概率计算个体对生成测试数据的贡献度, 并利用贡献度改进适应度函数, 同时根据关键点概率对目标路径进行排序; 最后, 使用多种群遗传算法生成覆盖目标路径的测试数据, 在进化过程中, 子种群覆盖当前目标路径后, 继续尝试覆盖该目标路径的相似路径. 实验结果表明, 该方法能够有效地提高多路径覆盖测试数据生成的效率.

关键词: 多种群遗传算法; 多路径覆盖; 关键点概率; 路径相似度; 贡献度
中图法分类号: TP311

中文引用格式: 钱忠胜, 祝洁, 朱懿敏, 俞情媛, 李端明, 宋佳. 结合关键点概率与路径相似度的多路径覆盖策略. 软件学报, 2022, 33(2): 434–454. http://www.jos.org.cn/1000-9825/6149.htm

英文引用格式: Qian ZS, Zhu J, Zhu YM, Yu QY, Li DM, Song J. Multi-path Coverage Strategy Combining Key Point Probability and Path Similarity. Ruan Jian Xue Bao/Journal of Software, 2022, 33(2): 434–454 (in Chinese). http://www.jos.org.cn/1000-9825/6149.htm

Multi-path Coverage Strategy Combining Key Point Probability and Path Similarity

QIAN Zhong-Sheng, ZHU Jie, ZHU Yi-Min, YU Qing-Yuan, LI Duan-Ming, SONG Jia

(School of Information Management, Jiangxi University of Finance & Economics, Nanchang 330013, China)

Abstract: Using multi-population genetic algorithm to solve the problem of multi-path coverage is an important research direction in the field of automatic generation of test data. In order to improve the efficiency of multi-path coverage test data automatic generation, a multi-path coverage strategy combining key point probability and path similarity is proposed. Firstly, the theoretical path is divided into easily-covered, difficultly-covered, and unreachable paths. Then, the key point probability is counted through the easily-covered paths, the contribution of the individual to the generated test data is calculated by using this probability, and the contribution isusedto improve the fitness function, at the same time, the target path is sorted according to the key point probability. Finally, the test data covering the target path is generated by using multi-population genetic algorithm. After the sub-population covers the current target path during the evolution process, it continues to try to cover similar paths of the target path. The experimental results show that the proposed method can improve the efficiency of multi-path coverage to generate test data.

Key words: multi-population genetic algorithm; multi-path coverage; key point probability; path similarity; contribution

在软件开发过程中, 软件测试可看成是对软件需求分析、设计规格说明和编码等的复审, 是保证软件质量的重要阶段[1]. 可见, 软件测试在软件开发的整个生命周期中都起着关键性作用[2]. 而在软件测试过程中, 被测程序能够进行测试的先决条件就是要有合适的测试数据, 因此, 测试数据生成是软件测试中一项必不可少的工作.

* 基金项目: 国家自然科学基金(61762041); 江西省自然科学基金(20181BAB202009); 江西省教育厅科技重点项目(GJJ180250)
收稿时间: 2020-05-30; 修改时间: 2020-07-14; 采用时间: 2020-09-12

在实际的软件测试中, 要生成满足被测程序要求的测试数据, 需要专业的测试人员花费大量时间进行实验, 这无疑会耗费庞大的时间和人力资源. 而利用计算机自动求解生成测试数据, 即生成一组测试数据, 满足给定的测试标准[3], 将大幅度地提高软件测试的效率, 使有限的资源得到更充分、合理的利用.

路径覆盖是对已知代码的被测程序, 自动生成其测试数据的常用方法[4]. 因此, 在测试数据自动生成领域中, 许多研究者对路径覆盖进行了探讨. 张岩等人[1]提出了基于稀有数据扑捉的路径覆盖测试数据进化生成方法, 通过保护稀有数据, 提高路径覆盖测试数据生成效率. 姚香娟等人[5]提出了融入神经网络的路径覆盖测试数据进化生成方法, 通过改进遗传算法, 求解路径覆盖问题.

可见, 大多数软件测试数据自动生成问题可以转化成目标路径的覆盖问题[6]. 单锦辉等人[7]将路径覆盖问题描述为: 给定程序的一条、多条或全部目标路径, 在程序的输入空间寻找测试数据, 对于任一目标路径, 测试数据集中至少存在一个测试数据, 使得以该测试数据为输入所穿越的路径为该目标路径.

路径覆盖可分为单路径覆盖及多路径覆盖[8]: 多路径覆盖测试数据自动生成的复杂程度远超过单路径覆盖, 但却更加贴近实际应用, 测试数据的生成也更加高效[9]. 在多路径覆盖测试数据自动生成的研究历史上, 诸多学者采用经典的遗传算法进行求解, 如夏春艳等人[10]提出了基于否定选择遗传算法的路径覆盖测试数据生成, 将否定选择策略融入遗传算法, 避免遗传算法过早收敛. 已有研究虽然都取得了卓越的成果, 但也仍存在部分待解决的问题. 例如: 如何保护测试数据进化过程中的优秀个体, 进而使优秀基因遗传给下一代, 以加快遗传算法生成测试数据的速度.

为此, 提出了一种结合关键点概率与路径相似度的多路径覆盖策略, 具体过程如下.

1) 基于关键点路径获取被测程序的理论路径. 随机生成测试数据集, 运行被测程序, 得到该数据集对应的覆盖路径, 即为易覆盖路径. 通过不可达路径自动检测模型探测到不可达路径, 将剩余理论路径分为难覆盖路径及不可达路径. 因为由随机法就可以实现覆盖的易覆盖路径作为目标路径没有实际意义, 因此选择部分难覆盖路径作为目标路径. 若能高效地生成这些目标路径的测试数据, 则可表明该策略有效;
2) 统计易覆盖路径中的关键点被覆盖情况, 计算关键点概率以及个体对生成覆盖目标路径测试数据的贡献度. 将个体贡献度作为适应度函数权重, 调整多种群遗传算法的适应度函数. 根据关键点概率对目标路径进行排序, 优先执行相对容易覆盖的目标路径对应的子种群. 因为越早移除已覆盖的目标路径及对应的子种群, 就可以避免过多的子种群占用计算机资源, 从而拖慢覆盖进度;
3) 利用多种群遗传算法生成覆盖目标路径的测试数据. 采用个体信息共享策略, 实现各子种群之间的信息交互, 但各子种群的个体不参与其他子种群的进化. 为使种群中的个体信息得到充分利用, 子种群覆盖其目标路径后, 继续尝试覆盖与该目标路径相似度高的其他目标路径.

本文第 1 节介绍相关工作. 第 2 节给出关键点路径、关键点概率、个体贡献度、路径相似度以及个体信息共享策略等相关定义. 第 3 节给出结合关键点概率与路径相似度的多路径覆盖策略的整体模型, 并对多种群遗传算法测试数据自动生成过程进行详细阐述. 第 4 节在基准程序及工业程序上进行实验, 将实验结果与其他方法作对比分析, 以验证该策略的有效性. 第 5 节总结全文, 并给出下一步的工作.

1 相关工作

将搜索算法运用于测试数据生成领域, 在近些年来得到广泛关注. 相对于模拟退火、禁忌搜索、粒子群等搜索算法, 遗传算法因其良好的全局搜索能力, 得到更多的应用.

针对路径覆盖测试数据生成问题, 遗传算法可以在搜索空间中快速找出覆盖目标路径的测试数据, 诸多学者对遗传算法求解路径覆盖问题进行了尝试性研究.

Praveen 等人[11]于 2009 年提出了一种通过识别程序中最关键的路径集来优化软件测试效率的方法, 根据路径的关键度对路径集进行加权, 再通过遗传算法来优化和选择路径集.

姚香娟[9]在其博士论文中提出了建立包含多个目标函数的模型, 然后采用多种群遗传算法进化求解. 模

型中的目标函数相互独立, 算法中不需要个体迁移, 而是使用个体信息共享策略求解模型. 此后, Yao 等人[12]建立了一个受约束的多目标测试数据生成模型, 该模型的两个目标函数分别是测试数据的空间分散性和路径视差, 约束条件是生成的测试数据满足语句覆盖准则, 并提出一种基于集合进化的遗传算法来求解该模型.

我们曾设计了一种改进个体信息共享策略的多路径覆盖策略[13], 该方法针对多种群遗传算法进化生成测试数据问题, 提出在子种群个体信息共享过程中, 当子种群覆盖对应目标路径后, 继续尝试覆盖其他目标路径.

Maragathavalli 等人[14]提出了一种基于路径重用的多路径覆盖方法, 如果某测试用例满足覆盖路径 p, 则先保存该用例, 再确定其他路径与路径 p 共有的节点, 并从路径 p 中获取这些节点的测试数据. 若存在未求解的节点, 则采用遗传算法求解.

张岩等人[1]提出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度值, 保护进化过程中的稀有个体.

夏春艳等人[15]提出在遗传算法进化过程中, 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, 根据个体的穿越度来设计适应度函数.

丁蕊等人[16]提出了基于关键点的路径表示及其计算方法, 得到待测程序的理论路径数, 再使用不可行路径检测模型确定不可行路径, 并且在遗传算法的优化过程中, 根据已覆盖路径及其测试数据提供的信息, 利用遗传算法生成难覆盖路径的测试数据.

Gong 等人[17]提出了一种基于分组的多路径覆盖测试数据进化生成方法, 根据目标路径的相似性将目标路径分成若干组, 每一组形成一个子优化问题, 然后在遗传算法求解这些问题时, 设计了一个基于域的适应度.

Zhu 等人[18]为平衡每个计算资源的负载, 通过改进的分组策略实现不平均的目标路径集划分, 再利用符号执行工具收集同一组中目标路径的共同约束, 将初始种群及其后代限制在给定区域的范围内, 以减少测试数据的搜索空间.

Tian 等人[19]根据并行程序的非确定性行为, 将多路径覆盖测试数据生成问题归结为多目标优化问题, 并采用遗传算法求解上述问题.

Ahmed 等人[20]设计了一个基于遗传算法的路径覆盖测试数据生成器, 它使用跨站点脚本 XSS 攻击模式的数据库来生成表示潜在攻击的输入值, 自动生成测试数据以测试 XSS 类型的漏洞问题.

以上研究中, 国内外学者从多路径测试数据生成模型、个体优劣判断方法、目标路径分组策略、算法改进等角度着手, 已经利用遗传算法在路径覆盖测试数据生成领域取得了一定成果, 提高了路径覆盖测试数据生成的效率. 但同时, 这些方法也存在一些不足. 例如: 只能实现单路径覆盖, 要生成覆盖多条目标路径的测试数据, 需要多次执行遗传算法; 在测试数据进化生成时动态扑捉稀有数据, 要求每一次迭代过程都重新统计节点的贡献度, 增加了计算量; 子种群进行个体信息共享, 未能充分应用进化过程中的个体信息.

在前述工作的基础上, 为了继续提高多路径覆盖测试数据生成效率, 本文提出一种结合关键点概率与路径相似度的多路径覆盖策略, 根据关键点概率设计适应度函数, 保护优秀个体, 同时对目标路径进行排序及相似度计算, 并进一步完善个体信息共享策略, 既合理利用种群进化过程中的个体资源, 又避免了在子种群进化过程中浪费过多的时间, 这无疑大幅度提高了测试数据进化生成的效率.

2 相关定义

为便于理解, 首先定义本文用到的相关概念, 包括关键点路径、关键点概率、个体贡献度、路径相似度、个体共享策略等, 并以典型的三角形分类程序为例进行说明, 其程序伪代码如图 1 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”); }
15 else {
16    strcpy (Type, “TRIANGLE”);
17    if (a==b && b==c) {
18       strcpy (Type, “EQUILATERAL”); }
19    if ((a==b||b==c) && a!=c) {
20       strcpy (Type, “ISOSCELES”); }}
e }
```

图 1　三角形分类程序伪代码示例

2.1 关键点路径及其表示

将程序转化成更加直观的图形结构，再用形式化语言对被测程序的路径进行描述，其详细定义如下.

定义 1(控制流图[21]). 可描述为 $G=(V,L,s,e)$，其中，V 是节点集合，节点对应程序中的语句；L 是节点的边集，边对应程序语句的流向；s 和 e 分别是控制流图的起点和终点.

定义 2(关键点图[16]). 由控制流图转化而来，其转化规则为:

- 控制流图中的分支节点，对应在关键点图中，分支之后一定归结于某一节点;
- 若分支节点的真假分支都有可执行语句，则这两个分支中的节点是关键点；若真假分支中只有一个分支有可执行语句，则在没有可执行语句的分支中插入一个新的关键点;
- 循环节点分解为若干分支关键点.

三角形分类程序的控制流图如图 2 所示，其对应的关键点图如图 3 所示.

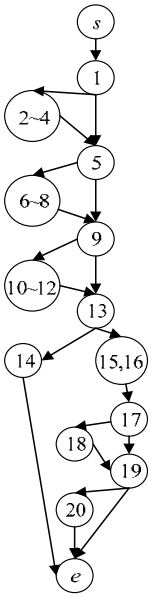


图 2　三角形分类程序控制流图

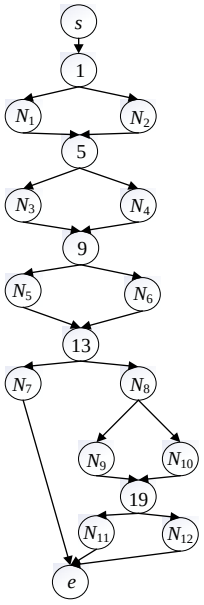


图 3　三角形分类程序关键点图

定义 3(关键点[16]). 关键点图中的节点包括分支关键点(对应控制流图中有两个直接后继节点的节点)、分支子关键点(分支关键点的两个直接后继节点)、普通关键点(既是分支关键点，也是其他分支关键点的分支子关键点，是一种特殊的分支子关键点)、起始关键点 s 及终止关键点 e.

例 1: 在图 3 中，关键点 1、5、9、13、17、19 等为分支关键点；关键点 N_1 和 N_2 为分支子关键点，它们互为兄弟；关键点 N_8 既是分支关键点，有 N_9 和 N_{10} 两个直接后继关键点，也是关键点 13 的分支子关键点，故为普通关键点.

定义 4(关键点路径[16]). 为约简路径, 关键点路径仅采用被测程序的分支子关键点进行描述: 关键点路径 $P=\{s,N,e\}$, 其中, $N=\{N_1,N_2,\ldots,N_n\}$为分支子关键点集合.

定义 5(关键点路径表达式[16]). 将关键点图中所有关键点用数学运算符连接起来的表达式. 其中, 兄弟关键点之间表示为相加的“或”关系, 普通关键点与其分支子关键点之间表示为相乘的“与”关系.

例 2: 在图 3 中, 兄弟关键点 N_1 和 N_2 的表达式应表示为 N_1+N_2, 普通关键点 N_8 与其分支子关键点 N_9 和 N_{10} 的表达式表示为 $N_8(N_9+N_{10})$, 关键点 N_7 与其兄弟关键点 N_8 的表达式表示为 $N_7+N_8(N_9+N_{10})(N_{11}+N_{12})$, 所有关键点以数值 1 代入计算, 可得到程序的理论路径数, 如三角形分类程序的理论路径数为

$$P_{tri}=(N_1+N_2)(N_3+N_4)(N_5+N_6)[N_7+N_8(N_9+N_{10})(N_{11}+N_{12})=40.$$

由于分支子关键点成对出现, 关键点路径表示方法有利于快速判断多条路径是否为相似路径, 便于求解路径相似度. 且对于分支中执行语句特别多的被测程序, 关键点路径比基于控制流图描述的路径要简洁得多, 有利于加快测试数据生成速度.

2.2 关键点概率

对于路径中的关键点, 为判断其被覆盖的难易程度, 引入关键点概率的概念, 关键点概率详细定义如下.

定义 6(关键点概率). 对被测程序输入 m 组测试数据 $in=(l_1,l_2,\ldots,l_m)$, 其中, $l_i(1\leqslant i\leqslant m)$为一组输入向量, 即个体, 可以得到 m 条覆盖路径集 $P_{cover}=\{P_1,P_2,\ldots,P_m\}$, 程序中每个分支子关键点 $N_j(1\leqslant j\leqslant n)$被路径覆盖的概率, 记为 $G(N_j)$.

利用随机生成的测试数据集及其对应的易覆盖路径, 统计易覆盖路径集中的关键点被路径覆盖的情况, 如公式(1)所示:

$$C_{ij}=\begin{cases}1, & P_i\text{穿过节点}N_j\\ 0, & P_i\text{未穿过节点}N_j\end{cases} \tag{1}$$

其中, $P_i\in P_{cover}$, 再得到覆盖矩阵, 记为 $Cover$, 如公式(2)所示:

$$Cover=\begin{matrix}N_1 & \ldots & N_n\\ \downarrow & & \downarrow\end{matrix}\begin{bmatrix}C_{11} & \ldots & C_{1n}\\ \vdots & \ddots & \vdots\\ C_{m1} & \ldots & C_{mn}\end{bmatrix}\begin{matrix}\leftarrow P_1\\ \vdots\\ \leftarrow P_m\end{matrix} \tag{2}$$

其中, 覆盖矩阵的行表示执行被测程序得到的 m 条覆盖路径 $P_1,P_2,\ldots,P_m$, 列为路径中的 n 个关键点 $N_1,N_2,\ldots,N_n$.

根据覆盖矩阵 $Cover$, 得到覆盖关键点 N_j 的路径数记为 S_j, 如公式(3)所示:

$$S_j=\sum_{i=1}^{m}C_{ij} \tag{3}$$

被测程序的关键点概率 $G(N_j)$可以表示成如公式(4)所示:

$$G(N_j)=\frac{S_j}{m} \tag{4}$$

由公式(4)可知: 关键点被易覆盖路径覆盖的次数越多, 关键点概率越高, 说明该关键点越容易被覆盖. 我们参考文献[1]中的做法, 首先排除起始关键点、终止关键点, 仅计算关键点路径中的其他关键点被覆盖的概率, 这可大幅度减少关键点概率的计算量.

例 3: 随机生成 3 组测试数据, 运行插桩后的三角形分类程序, 若得到其对应的易覆盖路径分别为 $P_1=\{s,N_1,N_4,N_6,N_8,N_{10},N_{12},e\}$, $P_2=\{s,N_1,N_3,N_5,N_7,e\}$, $P_3=\{s,N_1,N_3,N_5,N_8,N_{10},N_{12},e\}$, 则相应的覆盖矩阵:

$$Cover=\begin{bmatrix}1&0&0&1&0&1&0&1&0&1&0&1\\ 1&0&1&0&1&0&1&0&0&0&0&0\\ 1&0&1&0&1&0&0&1&0&1&0&1\end{bmatrix}.$$

可知: 覆盖关键点 N_1 的路径数 S_1=3, 其关键点概率 $G(N_1)$=1, 覆盖关键点 N_3 的路径数 S_3=2, 其关键点概率 $G(N_3)$=2/3.

2.3 个体贡献度

为了对个体的优劣程度进行判定, 考虑在算法进化过程中是否需要保留其基因, 故需计算个体的贡献度, 其详细定义如下.

定义 7(个体贡献度). 个体数为 m 的种群中的个体 $l_i(1\leqslant i\leqslant m)$, 对于进化生成覆盖目标路径集 $P_{tar}=\{P_1,P_2,P_3,\ldots,P_k\}$的目标路径 $P_k(1\leqslant k\leqslant n)$的测试数据所做出的贡献, 记为 $Con(l_i,P_k)$.

根据关键点概率的定义, 本文将个体的贡献度 $Con(l_i,P_k)$表达成如公式(5)所示:

$$Con(l_i,P_k)=\sum e^{-G(N_j)} \tag{5}$$

其中, e 为自然底数, $N_j\in(P(l_i)\cap P_k)$,$G(N_j)$为 N_j 的关键点概率(见公式(4)).

在多种群遗传算法进化过程中, 某关键点概率越高, 该关键点越容易被易覆盖路径覆盖. 按照常规逻辑理解, 若某个体 l_i 对应的覆盖路径 $P(l_i)$与目标路径 P_k 包含的相同关键点都是容易被易覆盖路径覆盖(关键点概率高)的关键点, 那么该个体 l_i 对于生成该难覆盖路径 P_k 能做出的贡献就越小; 若某个体 l_i 对应的覆盖路径 $P(l_i)$与目标路径 P_k 包含的相同关键点都是很难被易覆盖路径覆盖(关键点概率低)的关键点, 则可以合理地认为该个体包含了稀有的数据, 对生成该目标路径 P_k 有较大的贡献, 应当在种群进化过程中保护该个体. 因此, 认为个体贡献度与个体对应覆盖路径(与目标路径相同的关键点)的关键点概率成反比关系.

例 4: 若三角形分类程序的关键点概率分别为 $G(N_1)$=2/5, $G(N_2)$=3/5, $G(N_3)$=3/5, $G(N_4)$=2/5, $G(N_5)$=1/5, $G(N_6)$=4/5, $G(N_7)$=3/5, $G(N_8)$=2/5, $G(N_9)$=0, $G(N_{10})$=2/5, $G(N_{11})$=0, $G(N_{12})$=2/5, 个体 l_1 对应的覆盖路径为 $P_1=\{s,N_1,N_4,N_6,N_8,N_{10},N_{12},e\}$, 目标路径 $P_2=\{s,N_2,N_4,N_6,N_7,e\}$, 则个体 l_1 对于目标路径 P_2 的贡献度为

$$Con(l_1,P_2)=e^{-G(N_4)}+e^{-G(N_6)}=e^{-2/5}+e^{-4/5}.$$

2.4 个体信息共享

在多种群遗传算法进化过程中, 子种群进行交互的方式, 常见的情况有子种群迁移[22,23]、子种群信息共享[9,13,24]等. 本策略选择个体信息共享作为子种群交互的方式, 其详细定义如下.

定义 8(个体信息共享[9]). 在多种群遗传算法中, 每经历一次操作, 不仅判断该种群中个体是否为当前种群中最优的, 还继续判断它是否是其他种群问题的解.

对于种群集 $pop=\{pop_1,pop_2,\ldots,pop_n\}$, 第 $i(1\leqslant i\leqslant n)$个子种群 $pop_i=\{l_{i1},l_{i2},\ldots,l_{im}\}$中的个体 $l_{ij}(1\leqslant j\leqslant m)$, 除了需要判定是否是对应适应度函数 $\max(F_i)$的最优解, 还需要判断个体是否是其他子种群对应适应度函数 $\max(F_k)(1\leqslant k\leqslant n$ 且 $k\neq i)$的最优解, 但是判断个体 l_{ij} 是否是 $\max(F_k)$的最优解, 不需要计算 $F_k(l_{ij})$, 只需判断个体 l_{ij} 穿越的路径 $P(l_{ij})$是否为目标路径 P_k. 实际上, 个体 l_{ij} 不参与子种群 pop_k 的进化过程, 个体不在多个子种群间进行迁移, 只进行信息共享, 详细示意图如图 4 所示.

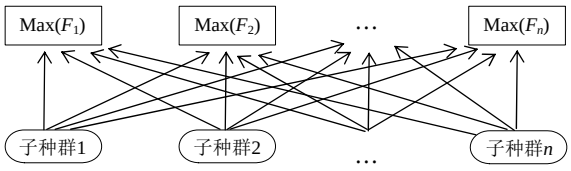


图 4 个体信息共享

2.5 路径相似度

对个体共享策略进行改进, 当种群 pop_i 覆盖其对应目标路径后, 继续尝试覆盖其目标路径的其他相似路径. 为此, 引入路径相似度的概念, 用以判断两条路径是否相似, 其详细定义如下.

定义 9(路径相似度). 目标路径集 $P_{tar}=\{P_1,P_2,P_3,\ldots,P_n\}$中的目标路径 $P_j(1\leqslant j\leqslant n)$与目标路径 $P_k(1\leqslant k\leqslant n$ 且

$k\neq j$)相同的关键点个数与路径 P_j、P_k 的最大关键点个数之比，记为 $Pro(P_j,P_k)$.

统计目标路径 P_j 与目标路径 P_k 关键点的异同情况，如公式(6)所示:

$$l(N_{ji},N_{ki})=\begin{cases}0, & N_{ji}\text{与}N_{ki}\text{不相同}\\ 1, & N_{ji}\text{与}N_{ki}\text{相同}\end{cases} \tag{6}$$

其中，$1\leqslant i\leqslant \min(len(P_j),len(P_k))$，而 $len(P_j)$表示路径 P_j 的关键点个数，$len(P_k)$表示路径 P_k 的关键点个数，$\min(len(P_j),len(P_k))$表示路径 P_j 及路径 P_k 较小关键点个数，N_{ji} 为路径 P_j 的第 i 个位置上的关键点，N_{ki} 为路径 P_k 的第 i 个位置上的关键点(特别注意的是，i 表示关键点在路径中的位置索引).

根据路径相同序列长度，可以得到路径相似度 $Pro(P_j,P_k)$，如公式(7)所示:

$$Pro(P_j,P_k)=\frac{1}{\max(len(P_j),len(P_k))}\sum_{i=1}^{\min(len(P_j),len(P_k))}l(N_{ji},N_{ki}) \tag{7}$$

其中，$\max(len(P_j),len(P_k))$表示路径 P_j 及路径 P_k 较大关键点个数.

例 5: 在三角形分类程序中，假定目标路径 $P_1=\{s,N_2,N_4,N_6,N_8,N_{10},N_{11},e\}$，目标路径 $P_2=\{s,N_1,N_4,N_5,N_8,N_{10},N_{11},e\}$，目标路径 $P_3=\{s,N_2,N_4,N_6,N_7,e\}$，则路径 P_1 与路径 P_2 的路径相似度 $Pro(P_1,P_2)=3/4$，路径 P_1 与路径 P_3 的路径相似度 $Pro(P_1,P_3)=5/8$，路径 P_2 与路径 P_3 的路径相似度 $Pro(P_2,P_3)=3/8$.

3 关键点概率与路径相似度相结合的多路径覆盖

这里给出一种结合关键点概率与路径相似度的多路径覆盖策略. 首先，对理论路径划分，确定目标路径；其次，根据个体贡献度调整适应度函数，确定多路径覆盖问题的数学模型；然后，根据关键点概率对目标路径进行排序，并求解目标路径之间的相似度；最后，改进个体信息共享策略，利用多种群遗传算法生成测试数据. 多路径覆盖策略总体框架如图 5 所示.

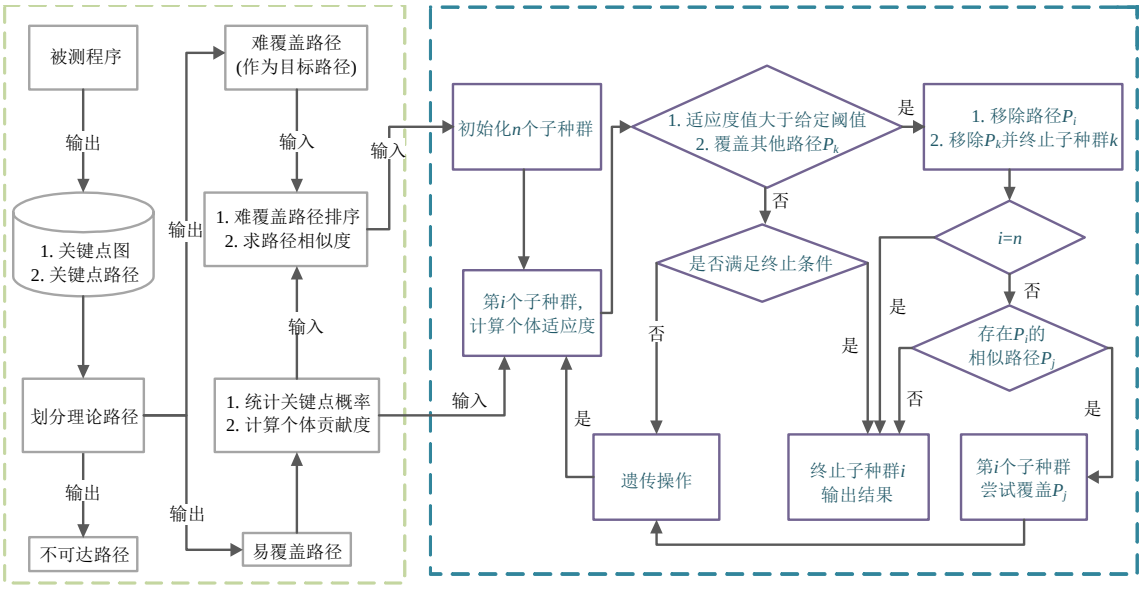


图 5 多路径覆盖策略总体框架

3.1 理论路径划分

为提高多路径覆盖测试数据的生成效率，首先要对被测程序的理论路径进行划分，划分为不可达路径、易覆盖路径、难覆盖路径，再对不同类型的路径进行不同的处理. 检测出被测程序的不可达路径，避免种群进化过程中因尝试覆盖不可达路径而浪费不必要的资源；对于使用随机法生成测试数据就能覆盖的易覆盖路

径, 没有必要使用遗传算法进化生成, 因此多种群遗传算法的目标路径应选择难覆盖路径.

随机生成测试数据集, 结合理论路径, 得到测试数据对路径的覆盖情况, 被覆盖的路径定义为易覆盖路径. 对目标路径进行划分, 主要目的是为了得到部分易覆盖路径及需要作为目标路径的难覆盖路径, 实际上并不需要产生所有的不可达路径, 即, 不需要给出产生不可达路径的方法, 仅对未被覆盖的理论路径使用不可达路径自动检测模型[9], 检测出不可达路径, 剩余路径即可定义为难覆盖路径. 具体过程见算法 1.

算法 1. 理论路径划分算法.

输入: 路径集Γ_{paths};

输出: 易覆盖路径Γ_{easy}, 不可达路径Γ_{un}, 难覆盖路径Γ_{hard}.

Begin

1 Generate some test data randomly and run the instrumentation program;
//随机生成部分测试数据, 运行插桩程序

2 Output Γ_{easy}; //易覆盖路径

3 $\Gamma_{paths}\leftarrow$*Delete*(Γ_{easy}); //调用子函数 *Delete*(·)删除易覆盖路径

4 Use unreachable paths detection model to find Γ_{un}; //用不可达路径检测模型, 找出不可达路径

5 Output Γ_{un};

6 $\Gamma_{paths}\leftarrow$*Delete*(Γ_{un}); //调用子函数 *Delete*(·)删除不可达路径

7 $\Gamma_{hard}\leftarrow\Gamma_{paths}$; //剩余为难覆盖路径

8 Output Γ_{hard}; //难覆盖路径

//子函数 *Delete*(·), 删除路径集Γ_{paths}中的路径 *solutions*

9 **Function** *Delete*(*solutions*)

10 **For** i=0$\rightarrow$*solutions*. *length* **Do**

11 **For** $j\in\Gamma_{paths}$ **Do**

12 **If** $\Gamma_{paths}[j]\equiv$*solutions*[i] **Then**

13 Delete *solutions*[i] in Γ_{paths}; //删除Γ_{paths}中的路径 *solutions*[i]

14 **End If**

15 **End For**

16 **End For**

17 **Return** Γ_{paths}

18 **End Function**

End

例如, 对于三角形分类程序, 假设随机生成 10 组测试数据, 其覆盖路径 $P_1,P_2,\ldots,P_{10}$ 为易覆盖路径, 删除理论路径集中的这些易覆盖路径; 接着, 利用不可达路径自动检测模型[9]检测出 22 条不可达路径, 同样删除; 剩余路径为需要进行测试数据生成的目标路径, 即难覆盖路径.

根据生成的易覆盖路径, 计算路径中关键点的关键点概率(见第 2.2 节关键点概率定义), 将难覆盖路径作为目标路径, 生成覆盖难覆盖路径的测试数据.

3.2 适应度函数设计

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)[25]、层接近度(approach_level)[26]以及两者相结合[27]的方法. 本文采用将分支距离与层接近度相结合的方法设计适应度函数.

定义 10(适应度函数). 个体数为 m 的种群中的个体 $l_i(1\leqslant i\leqslant m)$的适应度函数由其层接近度、分支距离及个体贡献度组成, 记为 $F(l_i)$.

个体 l_i 的层接近度为: l_i 对应覆盖路径 $P(l_i)$与目标路径集 $P_{tar}=\{P_1,P_2,P_3,\ldots,P_n\}$中的目标路径 $P_j(1\leqslant j\leqslant n)$相

同的关键点个数除以路径 P_j 的关键点数，记为 $approach\_level(l_i,P_j)$. 个体 l_i 的分支距离参考 Tracey 等人[28]提出的分支谓词的分支距离计算函数及复合谓词的计算方法，记为 $branch\_distance(l_i,P_j)$. 为了权衡分支距离与层接近度的大小，并统一为最大化运算，将分支距离规范化表示为 $1.001^{-branch\_distance(l_i,P_j)}$. 个体 l_i 对于目标路径 P_j 的贡献度 $Con(l_i,P_j)$(见公式(5))作为适应度函数的权重. 可将适应度函数 $F(l_i)$表达成如公式(8)所示:

$$F(l_i)=[approach\_level(l_i,P_j)+1.001^{-branch\_distance(l_i,P_j)}]\times Con(l_i,P_j) \tag{8}$$

对于目标路径集 $P_{tar}=\{P_1,P_2,\ldots,P_k\}$中的每条路径 $P_j(1\leqslant j\leqslant k)$, 对被测程序输入一组测试数据 $l_j=(s_{j1},s_{j2},\ldots,s_{jm})$能够覆盖路径 P_j 时，目标函数 $f_j=F(l_j)$取得最大值. 多路径覆盖问题要求寻找至少 k 个测试数据，使其能够分别覆盖这 k 个目标路径，则问题转化成求解 $f_1,f_2,\ldots,f_k$ 最大值的优化问题，即如公式(9)所示:

$$f_{(P_j)}=\max(F(l_j)) \tag{9}$$

其中, $P_j\in P_{tar}$.

借鉴文献[9]中的建模方法，各个目标函数都对应一条目标路径，目标函数之间相互独立，最终要求对于每个目标函数能够找到其对应的一组测试数据. 因此，多路径覆盖问题的最终数学模型可表达成如公式(10):

$$\begin{cases}\max(F_1(l_1))\\ \max(F_2(l_2))\\ \vdots\\ \max(F_k(l_k))\end{cases} \tag{10}$$

在公式(10)中，最终模型由 k 个函数构成，每个函数对应一个优化问题，每个优化问题对应一个覆盖目标路径的测试数据. 因为每个子函数都相互独立，所以最终需要求解的问题是找到对应每个子函数的解，最后形成一个包含多个解的解集合.

3.3 目标路径排序及相似度计算

在利用多种群遗传算法生成测试数据之前，先根据关键点概率对目标路径进行排序及相似度的计算.

多种群遗传算法中的种群集 *pop* 中的种群数量，随着目标路径集中的路径被覆盖逐步减少. 如: 当第 i 个子种群 pop_i 覆盖第 k 条目标路径 P_k 时，就将路径 P_k 对应的子种群 pop_k 从种群集 *pop* 中移除. 对于多种群遗传算法，越早减少子种群及目标路径的数量，使被占用的资源越早得到释放，算法的效率就越高. 因此，我们先对目标路径进行排序，让可使目标路径更易覆盖的子种群优先执行.

目标路径的优先级为路径中所有关键点概率(计算方法见公式(4))的均值，优先级越大，说明该目标路径相对容易被覆盖，应当优先被执行. 目标路径优先级判定的详细过程见算法 2.

算法 2. 目标路径优先级判定算法.

输入: 目标路径集Γ;

输出: 数组 *pri*, 表示路径的优先级.

Begin

1 *node*←0; //路径中关键点数
2 **For** $i\in\Gamma$ **Do** //所有路径
3 **For** $j\in i$ **Do** //路径 i 的关键点
4 **If** $j\equiv 1$ **Then** //路径 i 的关键点 j 为 1
5 *node*←*node*++; //路径关键点总数加 1
6 Get the keypoint probability *exponent* of *j*; //获得 *j* 的关键点概率 *exponent*
7 *totalProb*[*i*]←*totalProb*[*i*]+*exponent*; //路径中关键点概率总和
8 **End If**
9 **End For**
10 *pri*[*i*]←*totalProb*[*i*]/*node*; //路径 *i* 的优先级为平均分支子关键点概率

11 **End For**

12 **Return** *pri*

End

例如, 以第 2.3 节个体贡献度定义中例 4 的关键点概率为基础, 判断在第 2.5 节路径相似度定义中例 5 的目标路径优先级, 路径 P_1 的优先级 pri_1=13/30, 路径 P_2 的优先级 pri_2=3/10, 路径 P_3 的优先级 pri_3=3/5. 因此, 应当按 P_3、P_1、P_2 的顺序执行目标路径对应的子种群.

并且, 为了能够充分利用子种群进化过程中的个体资源, 同时节约进化过程所用的时间, 对多种群遗传算法进行了改进. 在种群遗传算法进化过程中, 当子种群 pop_i 覆盖当前目标路径 P_i 后, 并不是立刻终止子种群 pop_i, 而是让子种群 pop_i 继续尝试覆盖目标路径 P_i 的相似路径.

所以在对目标路径进行排序后, 还要继续计算目标路径之间的相似度(计算方法见公式(7)), 当路径相似度 $Pro(P_j, P_k)$大于给定阈值 T(确定三角形分类程序的相似路径和不相似路径, 分别计算路径之间的相似度, 发现相似的路径, 路径相似度都不低于 5/8, 反之都低于 5/8, 故本文将阈值设为 5/8)时, 定义目标路径 P_j 与 P_k 相似. 因为在基于关键点表达的路径中, 分支子关键点都是成对存在的, 便于计算路径的相似度, 以确定相似路径.

例如, 在第 2.5 节路径相似度定义中例 5 得到的路径相似度, 与给定阈值(T=5/8)进行比较, 可得出, 路径 P_1 与 P_2 相似, 路径 P_1 与 P_3 相似, 路径 P_2 与 P_3 不相似.

3.4 多路径覆盖测试数据进化生成

经过第 3.1 节–第 3.3 节的处理, 最后采用改进的多种群遗传算法, 对第 3.1 节选出的多条目标路径, 求解覆盖这些路径的测试数据.

首先, 对被测程序进行插桩处理, 初始化参数, 包括子种群数(目标路径集中目标路径数)n, 子种群中个体数 m, 终止代数, 种群进化需要的选择、交叉和变异概率值等, 并采用二进制格式编码个体.

其次, 多种群进化过程的具体步骤又包括: (1) 对任意属于排序后的目标路径集 P_{tar}={P_1,P_2,P_3,…,P_n}的目标路径 P_i, 随机生成个体数为 m 的子种群 pop_i, 对第 i 个种群 pop_i, 计算该种群中个体覆盖第 i 条路径的适应度值的最大值 $\max(F_i(in_i))$. 如果存在个体的适应度值达到最大值, 说明该个体覆盖目标路径 P_i, 将 P_i 从目标路径集中移除; 若不是, 则对该种群执行选择、交叉、变异等遗传操作; (2) pop_i 中个体除了判定是否是 $y_i=\max(F_i(in_i))$的最优解, 还需要判定是否是 $y_k(k\neq i)$的最优解, 如果 pop_i 中个体能够覆盖第 k 条目标路径, 则 pop_k 终止; (3) 当 $i\neq n$ 时, pop_i 需要继续对与该子种群对应的目标路径 P_i 的相似路径尝试覆盖, 如果找到覆盖第 $j(j\neq i \ \& \ j>i)$条路径的个体, 则将 pop_j 及路径 P_j 移除, 直到完成尝试对所有相似目标路径覆盖后, 终止 pop_i 执行.

最后, 若目标路径 P 被全部覆盖(表明算法完成任务), 或种群进化代数超出阈值, 则终止程序执行. 具体过程如算法 3 所述.

算法 3. 测试数据进化生成算法.

输入: 目标路径集Γ, 个体数 m;

输出: 测试数据 *results*.

Begin

1 Set parameters; //设置交叉概率 *crossRate*、变异概率 *mutationRate* 等参数

2 Here for each $P_i\in\Gamma$, *Pop*[*i*] with *m* individuals is randomly generated;
//对任意属于Γ的目标路径 P_i, 随机生成个体数为 m 的子种群 *Pop*[*i*]

3 **For** *j*=0→*Pop.length* **Do** //种群集中所有子种群

4 *Newpop*←*Pop*[*j*];

5 **If** *Newpop*≠∅ **Then** //*Newpop* 不空

6 *solLen*←Γ.*length*; //目标路径数

7 **For** $k\in\Gamma$ **Do** //所有目标路径

```
        If Pop≡∅ Then    //Pop 为空，说明前一种群进化覆盖了所有目标路径
          Break;
        End If
        If P_k∈L_delete Then    //P_k 存在删除列表中
          If proximity≠∅ Then    //P_k 有相似路径
            Pop[j] continues to try to cover similar paths;    //Pop[j]继续尝试覆盖相似路径
          Else
            Delete Pop[j];    //删除种群
          End If
        End If
        Get the fitness Max of P_k;    //获取 P_k 适应度值 Max
        If Newpop. popNo≠k+1 Then
          Determine whether Ind in Newpop covers P_k;    //Newpop 中个体 Ind 能否覆盖其他目标路径
          If some Ind can cover P_k Then    //存在个体 Ind 覆盖其他路径
            Delete Pop[k] and P_k;    //删除子种群 Pop[k]和目标路径 P_k
          End If
        Else
          Get the best Ind in Newpop;    //获取种群中最优个体 Ind
          Calculate the fitnesss S of Ind;    //计算 Ind 适应度值 S
        End If
        If S<Max Then    //适应度值 S 小于目标适应度值
          If Pop[j].evolveNum≥Evolve Then    //进化代数超过阈值 Evolve
            Add P_k to L_delete;    //将 P_k 加入删除列表
          Else
            Genetic operations are performed on Newpop;    //遗传操作
          End If
        Else
          Output results    //输出测试数据
        End If
      End For
    End If
End For
End
```

例如，假设使用算法 3 生成经过排序后的目标路径集Γ={P_1,P_2,P_3}的测试数据，其中，P_1和 P_2相似. 首先，分别为路径集Γ中的路径初始化子种群 Pop_1、Pop_2、Pop_3. 接着，计算子种群 Pop_1中个体的适应度值，判断种群中是否存在个体覆盖该种群的目标路径 P_1，同时判断个体是否覆盖其他目标路径 P_2或 P_3. 若存在个体覆盖目标路径 P_1，则将 P_1加入删除列表，输出测试数据. 若存在个体覆盖其他目标路径，则删除该目标路径及其所对应的子种群. 若没有覆盖目标路径 P_1，则该种群 Pop_1进行遗传操作. 需要注意的是：在 Pop_1覆盖目标路径 P_1后，不立即删除子种群 Pop_1，而是还要判断 P_1的相似路径 P_2的覆盖情况. 若 P_2还未被覆盖，则种群 Pop_1还要继续尝试覆盖 P_2. 直到Γ中所有路径被覆盖，或者超出进化代数的阈值，算法才终止.

以上对结合关键点概率与路径相似度的多路径覆盖策略进行了详细说明，下一节将对算法的有效性进行实验对比分析.

4 实验及其结果分析

为评价本文提出的基于关键点概率与路径相似度的多路径覆盖策略的性能, 需要进行实验, 对如下两个问题进行探讨.

问题 1: 相对于其他同类方法, 本文策略有什么优势, 在哪些方面得到改进?

为了对问题 1 做出解答, 本文基于相同的环境及参数进行实验, 具体的实验过程为: 首先, 选择三角形分类程序、冒泡排序等典型的基准程序进行实验说明, 并以工业程序 Vector、Flex、Sed、Space、Replace(来源于 software-artifact infrastructure repository)[29]进一步验证其优越性; 再与文献[9,13]提出的多路径覆盖测试数据生成方法进行对比, 考察本文提出的多路径覆盖策略的 3 个评价指标(生成测试数据的时间、进化代数、目标路径覆盖率)是否优于其他方法.

问题 2: 实验过程可能存在一定误差, 实验结果是否准确, 本文策略是否真实有效?

实验进行比较分析时, 按照前面 3 个评价结果的不同得出的结论可能还会出现一些偏差(比如系统误差、随机误差). 故针对该问题, 我们再展开进行统计学分析, 对实验结果进行显著性检验, 以鉴别结果差异是由实验误差引起的, 还是由特定的实验方法引起的. 因此, 本文取不同方法的 50 组平均生成时间(种群个体数为 100)进行显著性检验.

而常见的检验方法有 t 检验、U 检验、方差分析等. t 检验比较适用于进行一组或两组单因素小样本检验, 本文需要进行 3 种方法的实验结果对比分析及评估, 使用 t 检验不太合适; 并且不同方法的实验均采取相同参数独立运行, 得到实验结果进行对比, 也不考虑样本的非参数 U 检验; 方差分析可以比较多组样本的均值, 检验样本均数的差异是否有统计学意义, 因此我们选择方差分析对本文结果进行显著性检验.

4.1 实验环境及参数设置

本文对不可达路径的检测, 使用文献[9]中提出的基于条件语句相关性判定的不可达路径检测模型. 所需实验环境的相关参数包括: 操作系统、编程语言、JDK 版本、编程工具等, 详细信息见表 1. 为了与文献[9,13]进行对比, 本文的多种群遗传算法参数包括交叉概率、变异概率、选择策略等, 详细设置见表 2.

表 1 系统环境参数

参数	值
操作系统	Win10
编程语言	Java
内存	8 GB
系统类型	X64
主频	3.30 GHz
JDK 版本	1.8
编程工具	IntelliJ IDEA

表 2 算法参数

参数	值
交叉概率	0.9
变异概率	0.1
选择策略	轮盘赌法
最大迭代次数	10 000
种群大小	{20,50,100,150,200}
值域	(0,255]

选用的被测程序包括三角形分类程序、冒泡排序以及工业程序等, 其路径详细信息包括理论路径数、可达路径数、不可达路径数, 具体情况见表 3.

表 3 被测程序基本信息

被测程序	理论路径数	可达路径数	不可达路径数
三角形分类	40	18	22
冒泡排序	105	24	81
Vector	138	66	72
Flex	397	79	318
Sed	821	190	631
Space	735	210	525
Replace	976	257	719

4.2 实验过程及结果

本文实验的详细过程为: 首先, 随机生成测试数据集, 作为输入被测程序, 得到对应的易覆盖路径, 确定目标路径; 然后, 采用多种群遗传算法自动生成覆盖目标路径的测试数据; 最后, 将本文方法与其他方法进行对比, 检验本文方法的性能. 下面从基准程序和工业程序两个方面展开实验.

4.2.1 基准程序

三角形分类程序是测试领域最典型的基准程序, 为便于理解, 下面主要选择三角形分类程序进行实验说明. 根据图 3 三角形分类程序的关键点图, 对被测三角形分类程序进行插桩, 再随机生成 15 组测试数据, 运行插桩后的程序, 生成对应的易覆盖路径集 P_{cover}, 实验数据见表 4.

表 4 三角形分类程序中的易覆盖路径

P_1={s,N_1,N_4,N_6,N_8,N_{10},N_{12},e}	P_2={s,N_1,N_3,N_5,N_7,e}	P_3={s,N_1,N_3,N_5,N_8,N_{10},N_{12},e}
P_4={s,N_1,N_4,N_5,N_7,e}	P_5={s,N_2,N_4,N_5,N_8,N_{10},N_{12},e}	P_6={s,N_1,N_3,N_5,N_7,e}
P_7={s,N_2,N_4,N_5,N_7,e}	P_8={s,N_2,N_4,N_6,N_8,N_{10},N_{12},e}	P_9={s,N_2,N_3,N_5,N_7,e}
P_{10}={s,N_1,N_4,N_6,N_7,e}	P_{11}={s,N_1,N_4,N_6,N_8,N_{10},N_{12},e}	P_{12}={s,N_1,N_4,N_5,N_8,N_{10},N_{12},e}
P_{13}={s,N_2,N_4,N_5,N_7,e}	P_{14}={s,N_1,N_4,N_5,N_8,N_{10},N_{12},e}	P_{15}={s,N_1,N_4,N_6,N_7,e}

根据表 4 中三角形分类程序关键点被路径覆盖的情况, 得到覆盖矩阵 *Cover*(矩阵的行代表 15 条易覆盖路径, 列代表程序中 12 个分支子关键点), 实验数据如下面的 *Cover* 矩阵:

$$Cover=\begin{bmatrix}1&0&0&1&0&1&0&1&0&1&0&1\\1&0&1&0&1&0&1&0&0&0&0&0\\1&0&1&0&1&0&0&1&0&1&0&1\\1&0&0&1&1&0&1&0&0&0&0&0\\0&1&0&1&1&0&0&1&0&1&0&1\\1&0&1&0&1&0&1&0&0&0&0&0\\0&1&0&1&1&0&1&0&0&0&0&0\\0&1&0&1&0&1&0&1&0&1&0&1\\0&1&1&0&1&0&1&0&0&0&0&0\\1&0&0&1&0&1&1&0&0&0&0&0\\1&0&0&1&0&1&0&1&0&1&0&1\\1&0&0&1&1&0&0&1&0&1&0&1\\0&1&0&1&1&0&1&0&0&0&0&0\\1&0&0&1&1&0&0&1&0&1&0&1\\1&0&0&1&0&1&1&0&0&0&0&0\end{bmatrix}.$$

统计关键点概率, 具体数据为: $G(N_1)$=2/3, $G(N_2)$=1/3, $G(N_3)$=4/15, $G(N_4)$=11/15, $G(N_5)$=2/3, $G(N_6)$=1/3, $G(N_7)$= 8/15, $G(N_8)$=7/15, $G(N_9)$=0, $G(N_{10})$=7/15, $G(N_{11})$=0, $G(N_{12})$=7/15.

采用不可达路径自动检测模型检测不可达路径后, 选择以下难覆盖路径作为目标路径: P_1={s,N_2,N_4,N_6,N_7,e}, P_2={s,N_2,N_4,N_6,N_8,N_{10},N_{11},e}, P_3={s,N_1,N_4,N_5,N_8,N_{10},N_{11},e}, P_4={s,N_2,N_4,N_5,N_8,N_{10},N_{11},e}, P_5={s,N_2,N_4,N_6,N_8,N_9,N_{12},e}.

在利用多种群遗传算法生成测试数据之前, 结合关键点概率对目标路径进行排序, 路径优先级越大, 越容易覆盖. 例如, 目标 P_2 的优先级为(1/3+11/15+1/3+7/15+7/15+0)/8=7/24. 同理, 求其他目标路径的优先级, 排序后的目标路径集 P_{tar}={P_1,P_2,P_4,P_3,P_5}.

并求目标路径两两之间的相似度, 得到 P_1 的相似路径为 P_2 和 P_5; P_2 与所有路径相似; P_3 的相似路径为 P_2 和 P_4; P_4 的相似路径为 P_2、P_3、P_5; P_5 的相似路径为 P_1、P_2、P_4.

对选择的目标路径应用本文方法进行种群个体数为 20、50、100、150、200 的 20 次实验, 每次实验分别重复 5 次多目标路径覆盖. 给出个体数为 20 的一次实验结果, 5 组目标路径的测试用例具体见表 5.

表 5　目标路径覆盖的测试情况

	P_1	P_2	P_3	P_4	P_5
1	42, 69, 209	40, 209, 209	226, 201, 201	190, 231, 190	209, 209, 209
2	2, 162, 254	43, 229, 229	225, 139, 139	212, 238, 212	220, 220, 220
3	7, 68, 222	224, 224, 226	248, 241, 241	171, 253, 171	169, 169, 169
4	47, 143, 227	4, 102, 102	59, 42, 42	130, 202, 130	147, 147, 147
5	23, 161, 252	102, 248, 248	237, 201, 201	95, 176, 95	197, 197, 197

从表 5 可知: 生成的测试用例都能完全覆盖到目标路径, 即可以正确地找到每条目标路径的测试数据. 另外, 统计不同种群个体数的 100 组实验数据, 得到相应的性能指标值, 包括测试数据平均进化时间、种群平均进化代数、目标路径覆盖率, 详情见表 6.

表 6　测试结果详情

个体数	平均进化时间	平均进化代数	目标路径覆盖率(%)
20	0.557	901	100
50	1.138	1 291	100
100	1.957	1 955	100
150	3.504	2 267	100
200	5.568	2 678	100

由表 6 可知, 本文策略可快速、准确地生成目标路径的测试数据. 同时, 在个体数为 20 的条件下, 得到分别覆盖目标路径 P_1–P_5 的种群号及进化代数, 见表 7. 我们以此来观察使用本文策略实现的个体信息共享在路径覆盖中所起到的作用.

表 7　覆盖目标路径的种群详情

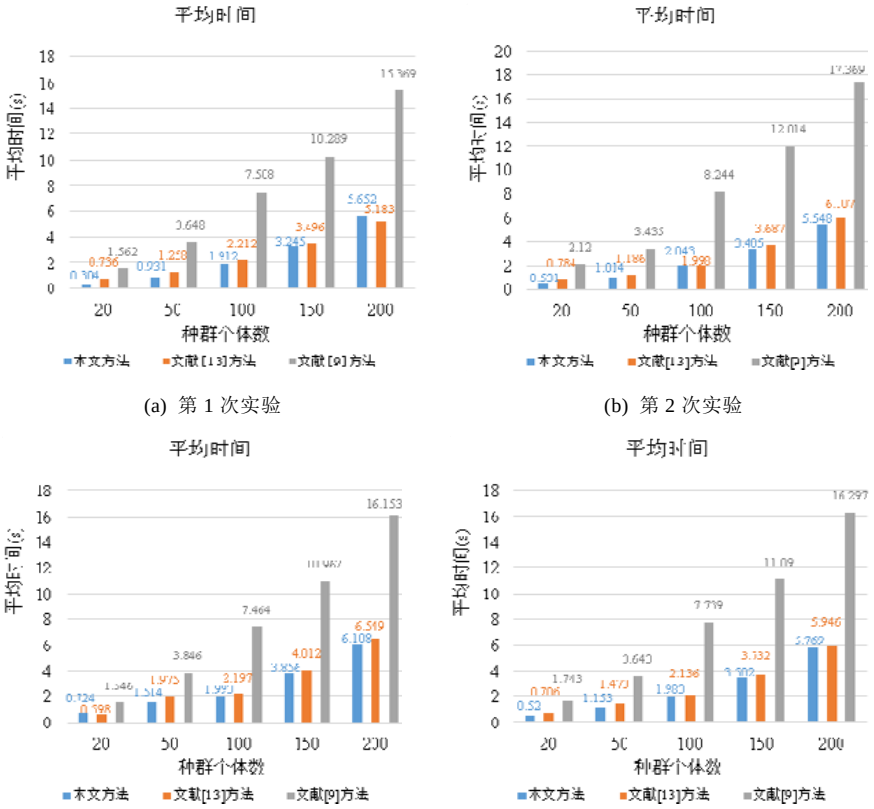
	种群号	进化代数
1	1, 2, 3, 3, 2	0, 3, 4, 57, 125 2
2	1, 4, 3, 3, 1	0, 3, 57, 9, 652
3	2, 4, 4, 4, 4	1, 5, 13, 8, 138 1
4	2, 5, 5, 4, 4	1, 2, 3, 38, 113 2
5	1, 1, 3, 4, 4	0, 19, 0, 41, 969

从表 7 可知: 在第 1 次实验中, 种群 1 未经进化就生成了目标路径 P_1 的测试数据; 种群 2 和种群 3 分别在进化了 3 代、4 代之后覆盖相应的目标路径 P_2 和 P_3; 然而, 种群并没有终止而是继续尝试覆盖相似路径, 两个种群之后, 又成功分别覆盖了相似路径 P_5 和 P_4; 随后的 4 次实验, 也都存在各种群在生成对应的目标路径之后, 种群继续尝试覆盖相似路径的现象. 这表明, 本文策略所提出的在种群实现目标路径的覆盖后, 继续尝试覆盖相似路径, 确实有利于多目标路径的测试数据进化生成.

并且在第 2 次实验中, 种群 4 进化了 3 代, 生成了目标路径 P_2 的测试数据, 即终止种群 2; 之后, 种群 4 继续生成路径 P_4 的测试数据, 但是种群 3 在进化了 9 代后覆盖了 P_4, 继而种群 4 被终止; 种群 3 在覆盖了其对应的目标路径 P_3 后, 不存在相似路径, 因此种群 3 被终止. 这些过程表明, 在多种群遗传算法进化过程中, 种群之间进行了个体信息共享.

此外, 从表 7 的进化代数的统计数据中还可以发现: 在完成所有目标路径覆盖所需的时间中, 覆盖路径 P_5 占用了最多的时间, 种群需要经过千次左右的进化才能成功获得其测试数据. 这是因为路径 P_5 的关键点概率很小, 路径中的关键点比较难覆盖, 其对应一个等边三角形.

为检验本文多路径覆盖策略的效率，针对三角形分类程序，进行 3 次本文方法与文献[9,13]中方法的多路径测试数据平均生成时间(秒)对比实验．每次实验基于不同种群个体数分别重复 100 次，得到生成时间的平均值，结果如图 6 所示．



(a) 第 1 次实验　(b) 第 2 次实验

(c) 第 3 次实验　(d) 3 次实验平均时间对比

图 6　测试数据生成平均时间对比

并且，在对比实验中可以发现：在规定进化代数内，本文策略与文献[13]中方法的目标路径覆盖率都能达到 100%；然而文献[9]中方法却不能实现对目标路径的完全覆盖，结果如图 7 所示．

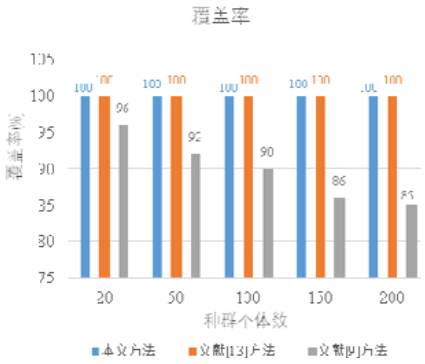


图 7　目标路径覆盖率对比

从图 6 和图 7 可以看出：本文策略较文献[9]中的方法具有比较明显的优势，生成覆盖目标路径的测试数据所用时间远远小于文献[9]中的方法，覆盖率也明显更高．但与我们之前在文献[13]中研究的方法相比，覆

盖率都可以达到 100%, 且测试数据生成平均时间非常接近, 本文策略大部分情况下优于文献[13], 但不是稳定的碾压式超越.

因此, 接下来我们将测试数据生成的平均进化代数作为评价指标, 对本文策略与文献[13]中的方法进行比较, 结果如图 8 所示.

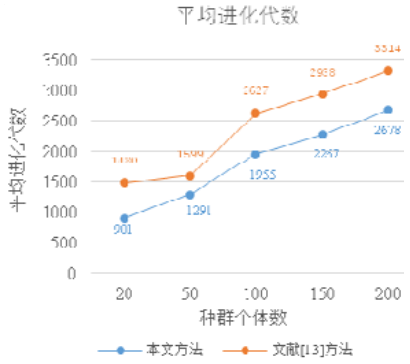


图 8 测试数据生成平均进化代数对比

由图 8 可以发现: 本文策略能够在更少的进化代数内覆盖目标路径, 完成目标路径的测试数据生成. 另外, 对冒泡排序程序进行了多路径测试数据平均生成时间的对比实验, 程序包括两个循环嵌套, 内层嵌套有一个条件语句, 路径包括 18 个关键点, 其实验结果如图 9 所示.

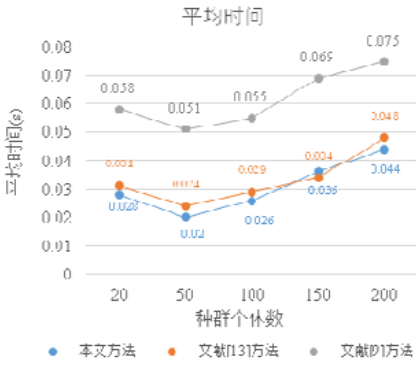


图 9 冒泡排序的测试数据生成平均时间对比

由以上基准实验结果发现: 本文策略具有较高的准确性, 能够提高多路径覆盖测试数据生成的效率. 相对于文献[13]中的方法, 时间未能得到显著缩短, 其原因可能是: 基于基准程序, 仅选取数量较少的目标路径, 在多种群遗传算法进化过程中, 一个种群在覆盖其目标路径后, 继续覆盖其他目标路径以及继续覆盖相似路径, 所用时间差距并不明显. 因此, 接下来进行了程序规模更大的工业程序对比实验, 对以上猜测进行验证.

4.2.2 工业程序

选择 5 个工业程序进行实验, 验证本文策略的有效性. 对每个被测的工业程序, 选取部分子函数, 被测工业程序实际的详细信息见表 8.

对工业程序 Vector 选择 1 个子函数, 随机生成 40 组测试数据, 得到 40 条易覆盖路径, 用以计算关键点概率, 选择 16 条难覆盖路径作为目标路径进行实验; Flex 选择 5 个子函数, 随机生成 50 组测试数据, 选择 20 条难覆盖路径作为目标路径进行实验; Sed 程序也选择 5 个子函数, 随机生成测试数据得到 50 条覆盖路径, 再选择 30 条难覆盖路径作为目标路径; Space 程序选择 3 个子函数, 随机生成测试数据得到 100 条覆盖路径, 再选择 50 条难覆盖路径作为目标路径; Replace 程序选择 3 个子函数, 随机生成测试数据得到 150 条覆盖路径, 再

选择 50 条难覆盖路径作为目标路径.

表 8 被测工业程序

程序	函数个数	函数代码数	目标路径数
Vector	1	254	16
Flex	5	427	20
Sed	5	585	30
Space	3	272	50
Replace	3	162	50

实验步骤同基准程序, 对不同方法分别进行种群个体数为 100 的 50 次实验, 得到相应的性能指标, 见表 9. 由表 9 可以看出: 就覆盖率来看, 在规定代数内, 本文策略与文献[13]中的方法都可以实现目标路径的完全覆盖, 而文献[9]中的方法对程序 Vector、Flex、Sed、Space、Replace 只能分别达到 85%、66%、58%、83%、67%的覆盖率; 从测试数据平均生成时间和种群平均进化代数来看, 本文策略的平均生成时间约等于文献[9]中方法的一半, 比文献[13]中的方法也少得多, 进化代数远小于另外两种方法. 这表明, 本文策略能够在更少的进化代数中更快地准确覆盖目标路径.

表 9 工业程序实验结果

程序	本文方法			文献[13]方法			文献[9]方法		
	平均生成时间	平均进化代数	覆盖率(%)	平均生成时间	平均进化代数	覆盖率(%)	平均生成时间	平均进化代数	覆盖率(%)
Vector	23.605	3 468	100	36.487	5 005	100	52.476	6 997	85
Flex	115.292	5 947	100	152.204	7 869	100	317.447	8 927	66
Sed	126.745	6 292	100	193.658	8 635	100	389.361	9 814	58
Space	52.565	4 184	100	84.671	5 312	100	120.948	7 713	83
Replace	370.945	5 491	100	456.794	6 975	100	691.042	8 549	67

由以上基准程序和工业程序的实验结果可以看出: 本文提出的结合关键点概率与路径相似度的多路径覆盖策略更有优势, 也验证了本文策略能够提高测试数据生成的效率.

4.3 实验结果的方差分析

为阐明实验结果的准确性, 本文采用方差分析(analysis of variance, ANOVA)方法对实验结果进行显著性分析, 使用 spss 软件对实验结果进行 F 检验(显著性水平: α=0.05).

以三角形分类程序为例, 取不同方法的种群个体数为 100 的 50 组平均生成时间, 进行实验结果方差分析, 其说明如下.

- 首先, 建立原假设“H_0: 各组平均数相等, 没有显著性差异”;
- 然后, 组间自由度为 2, 计算得到组间均方差为 585.78; 组内自由度为 147, 组内均方差为 0.58;
- 最后, 再构造统计量“F=组间方差/组内方差=1006.03”; F 满足第 1 自由度为 2、第 2 自由度为 147 的 F 分布, F 界值为 3.058; 比较 F 值与 F 界值可以发现, 本文方法与文献[13]、文献[9]中方法的多路径覆盖测试数据生成时间作为 3 组数据所得 F 值远远大于 F 界值, 则拒绝原假设.

以上情况说明, 不同方法的平均数存在显著性差异. 这意味着本文方法有效地改变了目标路径测试数据生成的时间, 排除了随机因素对实验造成的干扰. 同时得到了方差分析的具体描述, 见表 10.

表 10 方差分析描述

	个案数	平均值	标准偏差	标准错误	平均值的 95%置信区间		最小值	最大值
					下限	上限		
本文方法	50	1.955 40	0.528 255	0.074 707	1.805 27	2.105 53	0.401	2.952
文献[13]方法	50	2.307 26	0.542 466	0.076 716	2.153 09	2.461 43	1.094	3.578
文献[9]方法	50	8.051 9	1.083 272	0.153 198	7.744 10	8.359 82	5.978	10.875
总计	150	4.104 87	2.904 695	0.237 167	3.636 23	4.573 52	0.401	10.875

可以看出: 在相同条件下, 本文方法的平均值、标准偏差、标准错误、平均值的 95%置信区间的上下限等指标都要小于其他方法. 说明本文方法生成测试数据更加快速, 准确性高, 优于其他同类方法.

根据实验结果, 对基准程序及工业程序进行方差分析, 检验结果见表 11. 可以看出: 所有被测程序的 F 值都远大于 F 界值, 拒绝原假设, 3 种方法之间具有显著性差异, 验证了本文方法的有效性.

表 11 检验结果

被测程序	F 值	检验结果
三角形分类	1 006.03	拒绝 H_0
冒泡排序	485.91	拒绝 H_0
Vector	3 179.48	拒绝 H_0
Flex	7 744.72	拒绝 H_0
Sed	6 514.21	拒绝 H_0
Space	4 541.57	拒绝 H_0
Replace	8 473.73	拒绝 H_0

根据基准程序、工业程序的实验结果及其方差分析, 可以对本节开始部分提出的两个问题给出如下回答.

回答问题 1: 相对于其他的同类方法, 本文策略有什么优势, 在哪些方面得到改进?

从与文献[9,13]中方法的对比结果来看, 本文策略在生成测试数据的时间、进化代数、目标路径覆盖率这 3 个方面都得到了提升. 在基准程序中, 本文的数据生成时间是文献[9]方中法的 50%, 且实现了目标路径的全覆盖, 同时, 虽然数据生成时间、覆盖率与献[13]中方法比较接近, 但本文策略能够在更少的进化代数内找到目标路径的测试数据; 同样, 在工业程序中, 本文策略的数据生成时间不到文献[9]方法的一半, 并且远低于文献[13]中方法的时间, 本文策略的进化代数比文献[9,13]中方法分别平均减少约 1 000 代、3 000 代, 在覆盖率方面也具有明显优势. 因此可以看出: 本文策略能够在更短的进化代数内快速地找到目标路径的测试数据, 并且具有较高的实际应用价值, 有利于大型工业程序测试数据的生成.

回答问题 2: 实验过程可能存在一定误差, 实验结果是否准确, 本文策略是否真实有效?

针对不同方法的测试数据平均生成时间的方差分析, 证明了本文的实验结果具有统计学意义, 基本能够排除误差对实验结果造成的影响, 验证了本文策略的确具有较高的效率. 可以发现: 无论是在基准程序还是工业程序的结果上进行方差分析, 得到的 F 值都远超过 F 界值. 另外, 观察方差分析的描述(见表 10), 本文策略的平均生成时间明显少于其他方法, 偏差与错误值更低. 这就表明, 本文的结合关键点概率与路径相似度的多路径覆盖策略是真实有效的.

5 总结与下一步工作

本文提出一种通过种群中个体对生成测试数据的贡献度判断其优劣程度, 并设计了相应的适应度函数, 利用多种群遗传算法来实现多路径覆盖测试数据生成的策略, 以提升多路径覆盖测试数据生成的效率. 实验结果表明, 本文所提出的策略是有效的, 其主要贡献包括以下几点.

(1) 划分理论路径, 起到对目标路径预处理的作用.

将路径划分为不可达路径、易覆盖路径及难覆盖路径. 检测出被测程序的不可达路径, 避免种群进化过程中因尝试覆盖不可达路径而浪费不必要的资源, 使资源得到充分利用; 对于使用随机法就能轻易覆盖的易覆盖路径, 没有必要使用遗传算法进化生成, 只有将筛选出的难覆盖路径作为多种群遗传算法需要覆盖的目标路径才有价值.

(2) 改进适应度函数, 提高测试数据生成效率.

使用随机法生成部分测试数据集, 将该测试数据集作为输入, 运行被测程序, 得到易覆盖路径, 统计易覆盖路径中的关键点概率, 根据这一概率计算个体贡献, 将贡献度作为适应度函数的权重, 改进适应度函数. 这样可以使得贡献度较高的优秀个体具有较好的适应度, 并且能够减少适应度值的计算量, 提高了遗传算法生成测试数据的效率. 使用易覆盖路径求解关键点概率, 针对不可达路径及难覆盖路径数量非常多或者非常少的程序都是非常有利的, 既可以避免关键点概率的计算资源消耗过大, 也可以解决统计关键点概率可用的有效数据太少这样的问题, 例如实验中的工业程序 Sed 和 Replace, 不可达路径数非常庞大, 若根据不可达路径

计算关键点概率，则过程非常耗时.

(3) 优化多种群遗传算法，用以快速生成测试数据.

将多种群遗传算法进行优化，用于求解路径覆盖测试数据生成问题. 通过对目标路径进行排序，优先尝试执行能够覆盖相对易覆盖目标路径的子种群，使进化过程中产生的个体信息启发之后执行的子种群，并且更早地减少子种群数，释放其占用的资源，可使算法变得更高效；同时，改进多种群遗传算法的个体信息共享策略，子种群在覆盖该种群目标路径之后不立即终止，让其继续尝试覆盖相似目标路径. 若存在相似路径，则已有的个体信息有很大的概率能够覆盖该相似路径，使得个体信息得到更充分的利用；当没有相似路径时，已有个体信息能覆盖不相似路径的概率较小，继续进行尝试覆盖可能会拖慢整个进程，终止该子种群能减少不必要的尝试造成的时间浪费.

本文提出的结合关键点概率与路径相似度的多路径覆盖策略与传统的多路径覆盖方法相比，在覆盖率以及生成时间等方面效果更佳. 但当目标路径集中的路径条数比较庞大时，利用单纯的多目标路径覆盖算法来提高生成测试用例的效率实现的效果不太明显. 种群经过多代进化，基因趋于相似，实验会出现过拟合现象，需要花费更多的时间生成目标路径的测试数据. 另外，为了避免在前期预处理工作中花费过多的时间而增加整个测试过程的时间耗费，本文只考虑了路径覆盖难易程度与其包含节点的难覆盖程度的关系，对于影响个体贡献度的其他因素未进行探讨.

因此，接下来拟探索将更优的机器学习等算法与搜索算法相结合，在提高基因丰富性的同时，通过机器学习算法模拟个体近似的适应度值，减少适应度函数的计算量，进一步提升测试生成的效率.

References:

[1] 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). [doi: 10.3724/SP.J.1016.2013.02429]

[2] Zeng MF, Chen SY, Zhang WQ, Nie CH. Generating covering arrays using ant colony optimization: exploration and mining. Ruan Jian Xue Bao/Journal of Software, 2016, 27(4): 855−878 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4974.htm [doi: 10.13328/j.cnki.jos.004974]

[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] Ghiduk AS. Automatic generation of basis test paths using variable length genetic algorithm. Information Processing Letters, 2014, 114(6): 304−316. [doi: 10.1016/j.ipl.2014.01.009]

[5] Yao XJ, Gong DW, Li B. Evolutional test data generation for path coverage by integrating neural network. Ruan Jian Xue Bao/Journal of Software, 2016, 27(4): 828−838 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4973.htm [doi: 10.13328/j.cnki.jos.004973]

[6] Lv XW, Huang S, Hui ZW, Ji HJ. Test cases generation for multiple paths based on metamorphic relation. In: Proc. of the Int'l Conf. on Dependable Systems & Their Applications. IEEE, 2017. 143−143. [doi: 10.1109/DSA.2017.31]

[7] Shan JH, Jiang Y, Sun P. Research progress in software testing. Acta Scientiarum Naturalium Universitatis Pekinensis, 2005, 41(1): 134−145 (in Chinese with English abstract). [doi: 10.3321/j.issn:0479-8023.2005.01.020]

[8] Liao WZ, Xia XY, Jia XJ. Test data generation for multiple paths coverage based on ant colony algorithm. Acta Electronica Sinica, 2020, 48(7): 1330−1342 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2020.07.011]

[9] Yao XJ. Theory of evolutionary generation of test data for complex software and applications [Ph.D. Thesis]. Xuzhou: China University of Mining and Technology, 2011 (in Chinese with English abstract). [doi: CNKI:CDMD:1.1011.281090]

[10] Xia CY, Zhang Y, Wan L, Song Y, Xiao N, Guo B. Test data generation of path coverage based on negative selection genetic algorithm. Acta Electronica Sinica, 2019, 47(12): 2630−2638 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2019.12.024]

[11] Praveen RS, Taihoon K. Application of genetic algorithm in software testing. Int'l Journal of Recent Technology & Engineering, 2009, 3(5): 1−4. [doi: 10.1145/1980422.1980445]

[12] Yao XJ, Gong DW, Zhang GJ. Constrained multi-objective test data generation based on set evolution. IET Software, 2015, 9(4): 103−108. [doi: 10.1049/iet-sen.2014.0058]

[13] Qian ZS, Hong DF, Zhao C, Zhu J, Zhu ZG. A strategy for multi-target paths coverage by improving individual information sharing. KSII Trans. on Internet and Information Systems, 2019, 13(11): 5464−5488. [doi: 10.3837/tiis.2019.11.011]

[14] Maragathavalli P, Kanmani S, Kirubakar JS, Sriraghavendrar P, Prasad AS. Automatic program instrumentation in generation of test data using genetic algorithm for multiple paths coverage. In: Proc. of the IEEE Int'l Conf. on Advances in Engineering, Science and Management. India: IEEE, 2012. 349−353.

[15] 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 with English abstract). http://www.jos.org.cn/1000-9825/4967.htm [doi: 10.13328/j.cnki.jos.004967]

[16] Ding R, Dong HB, Zhang Y, Feng XB. Fast automatic generation method for software testing data based on key-point path. Ruan Jian Xue Bao/Journal of Software, 2016, 27(4): 814−827 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4971.htm [doi: 10.13328/j.cnki.jos.004971]

[17] Gong DW, Zhang WQ, Yao XJ. Evolutionary generation of test data for many paths coverage based on grouping. The Journal of Systems & Software, 2011, 84(12): 2222−2233. [doi: 10.1016/j.jss.2011.06.028]

[18] Zhu Z, Xu X, Jiao L. Improved evolutionary generation of test data for multiple paths in search-based software testing. In: Proc. of the 2017 IEEE Congress on Evolutionary Computation. Spain: IEEE, 2017. 612−620. [doi: 10.1109/CEC.2017.7969367]

[19] Tian T, Gong DW. Evolutionary generation approach of test data for multiple paths coverage of message-passing parallel programs. Chinese Journal of Electronics, 2014, 2(23): 291−296. [doi: 10.1016/j.image.2014.01.009]

[20] Ahmed MA, Ali F. Multiple-path testing for cross site scripting using genetic algorithms. Journal of Systems Architecture, 2016, 64: 50−62. [doi: 10.1016/j.sysarc.2015.11.001]

[21] Paul CJ, Wrote; Han K, Du XT, Trans. Software Testing. 2nd ed., Beijing: Machine Press, 2003. 62−83 (in Chinese).

[22] Qi RZ, Wang ZJ, Huang YH, Li SY. Spark-based parallel combination test case generation method. Chinese Journal of Computers, 2018, 41(6): 1064−1079 (in Chinese with English abstract). [doi: 10.11897/SP.J.1016.2018.01284]

[23] Deng XL, Wei B, Zeng H, Gui L, Xia XW. A multi-population based self-adaptive migration PSO. Acta Electronica Sinica, 2018, 46(8): 1858−1865 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2018.08.009]

[24] Feng YH, Yang J, He YC, Wang GG. Monarch butterfly optimization algorithm with differential evolution for the discounted {0-1} knapsack problem. Acta Electronica Sinica, 2018, 46(6): 1343−1350 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2018.06.010]

[25] Korel B. Automated software test data generation. IEEE Trans. on Software Engineering, 1990, 16(8): 870−879. [doi: 10.1109/32.57624]

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

[27] Ahmed MA, Hermadi L. GA-based multiple paths test data generator. Computer & Operations Research, 2008, 35(10): 3107−3127. [doi: 10.1016/j.cor.2007.01.012]

[28] Tracey N, Clark J, Mander K, McDermid J. An automated framework for structural test-data generation. In: Proc. of the Int'l Conf. on Automated Software Engineering. IEEE Computer Society, 1998. 285−288. [doi: 10.1109/ASE.1998.732680]

[29] SIR. A repository of software-related artifacts meant to support rigorous controlled experimentation. 2020. http//sir.unl.edu/portal/index.php

附中文参考文献:

[1] 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. [doi: 10.3724/SP.J.1016.2013.02429]

[2] 曾梦凡, 陈思洋, 张文茜, 聂长海. 利用蚁群算法生成覆盖表: 探索与挖掘. 软件学报, 2016, 27(4): 855−878. http://www.jos.org.cn/1000-9825/4974.htm [doi: 10.13328/j.cnki.jos.004974]

[5] 姚香娟, 巩敦卫, 李彬. 融入神经网络的路径覆盖测试数据进化生成. 软件学报, 2016, 27(4): 828−838. http://www.jos.org.cn/1000-9825/4973.htm [doi: 10.13328/j.cnki.jos.004973]

[7] 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. [doi: 10.3321/j.issn: 0479-8023.2005.01.020]

[8] 廖伟志, 夏小云, 贾小军. 基于蚁群算法的多路径覆盖测试数据生成. 电子学报, 2020, 48(7): 1330−1342. [doi: 10.3969/j.issn.0372-2112.2020.07.011]

[9] 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]. 徐州: 中国矿业大学, 2011. [doi: CNKI:CDMD:1.1011.281090]

[10] 夏春艳, 张岩, 万里, 宋妍, 肖楠, 郭冰. 基于否定选择遗传算法的路径覆盖测试数据生成. 电子学报, 2019, 47(12): 2630−2638. [doi: 10.3969/j.issn.0372-2112.2019.12.024]

[15] 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 [doi: 10.13328/j.cnki.jos.004967]

[16] 丁蕊, 董红斌, 张岩, 冯宪彬. 基于关键点路径的快速测试用例自动生成方法. 软件学报, 2016, 27(4): 814−827. http://www.jos.org.cn/1000-9825/4971.htm [doi: 10.13328/j.cnki.jos.004971]

[21] Paul CJ 著; 韩柯, 杜旭涛, 译. 软件测试. 第 2 版, 北京: 机械工业出版社, 2003. 62−83.

[22] 戚荣志, 王志坚, 黄宜华, 李水艳. 基于 Spark 的并行化组合测试用例集生成方法. 计算机学报, 2018, 41(6): 1064−1079. [doi: 10.11897/SP.J.1016.2018.01284]

[23] 邓先礼, 魏波, 曾辉, 桂凌, 夏学文. 基于多种群的自适应迁移 PSO 算法. 电子学报, 2018, 46(8): 1858−1865. [doi: 10.3969/j.issn.0372-2112.2018.08.009]

[24] 冯艳红, 杨娟, 贺毅朝, 王改革. 差分进化帝王蝶优化算法求解折扣{0-1}背包问题. 电子学报, 2018, 46(6): 1343−1350. [doi: 10.3969/j.issn.0372-2112.2018.06.010]

钱忠胜(1977－), 男, 博士, 教授, 博士生导师, CCF 专业会员, 主要研究领域为智能化软件, 软件自动化, 智能推荐算法, 人工智能, 大数据, 数据挖掘.

祝洁(1996－), 女, 硕士, 主要研究领域为软件工程, 软件测试, 测试数据自动生成.

朱懿敏(1984－), 男, 博士生, CCF 学生会员, 主要研究领域为人工智能, 机器学习.

俞情媛(1997－), 女, 硕士生, 主要研究领域为软件测试.

李端明(1995－), 男, 硕士生, 主要研究领域为软件测试.

宋佳(1997－), 女, 硕士生, 主要研究领域为软件测试.