基于路径相关性的回归测试数据进化生成

吴 川[1)] 巩敦卫[2)]

[1)](中国矿业大学计算机科学与技术学院 江苏 徐州 221116)

[2)](中国矿业大学信息与电气工程学院 江苏 徐州 221116)

摘 要 尽管回归测试是一种重要的软件测试方法,但是,如何选择测试目标,并充分利用已有的测试数据,目前尚缺乏有效的方法.文中基于路径相关性,研究求解回归测试数据生成问题的新方法,以高效地进化生成可用于回归测试的测试数据集.该方法根据路径与节点的相关矩阵,首先进行目标路径排序,并基于路径相关性,建立新的覆盖影响路径的回归测试数据生成问题的数学模型;其次,结合遗传算法对上述模型求解时,利用穿越已有目标路径的测试数据,编码后取代进化种群的部分个体.将所提方法应用于多个基准和工业程序的测试,并与其他回归测试数据生成方法比较,最后实验结果表明,所提方法能够有效提高生成测试数据的效率.

关键词 回归测试;测试数据生成;路径相关性;路径排序;遗传算法

中图法分类号 TP311 **DOI 号** 10.11897/SP.J.1016.2015.02247

Evolutionary Generation of Test Data for Regression Testing Based on Path Correlation

WU Chuan[1)] GONG Dun-Wei[2)]

[1)](*School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, Jiangsu* 221116)

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

Abstract Regression testing is an important method of software testing. However, there has been no effective method of choosing test objects and making full use of existing test data. This paper, focusing on the problem of generating test data for regression testing, proposes a new effective evolutionary generation approach of test data based on path correlation for regression testing. In our method, firstly, the target paths are sorted according to the correlation matrix between paths and nodes, and a novel mathematical model is built for generating regression test data that cover affected paths based on the matrix. Then, when solving the above model using genetic algorithms, test data that have traversed the existing target paths are coded and employed to replace a part of individuals of the current population. Finally, the proposed method was applied to several benchmarks and industrial programs, and compared with other test data generation methods for regression testing. Our experimental results showed that our method can effectively improve the efficiency of test data generation.

Keywords regression testing; test data generation; path correlation; path sorting; genetic algorithm

---

收稿日期:2014-06-04;最终修改稿收到日期:2015-05-05.本课题得到国家自然科学基金(61375067)、国家"九七三"科技计划项目子课题(2014CB046306-2)、高等学校博士学科点专项科研基金(博士生导师类)(20100095110006)和江苏省自然科学基金(BK2012566)联合资助.

吴 川,男,1980 年生,博士研究生,讲师,中国计算机学会(CCF)会员,主要研究方向为基于搜索的软件工程.E-mail:wuchuan@live.com.

巩敦卫,男,1970 年生,博士,教授,博士生导师,主要研究领域为智能优化与控制、基于搜索的软件工程.

1 引 言

在软件生命周期中,开发人员需要经常对软件进行维护,维护的内容包括:新增代码、删除不必要的代码以及修改原来的代码等.上述工作在满足新需求的同时,也可能引入新的缺陷或错误.因此,有必要对维护后的软件重新测试,以提高软件的可信度.所谓回归测试,是指确认维护的软件修正正确且没有引入新的缺陷而进行的测试[1].已有统计表明,回归测试占软件维护费用的 50%以上[2].这说明,研究合适的方法,提高回归测试效率,对降低软件维护成本是非常必要的.

回归测试流程如下:首先,识别受影响的程序片段;然后,选择已有的部分测试数据,或基于现有测试数据,再生成新的测试数据,使得这些数据的执行路径能够覆盖影响的程序片段;最后,以这些测试数据为输入,执行维护后的程序,以验证其正确性[3].与其他测试方法相比,已有测试数据集包含测试信息的利用,是回归测试的关键.因此,如何有效利用已有测试数据集,以提高测试效率,是回归测试的重要研究内容.

到目前为止,关于利用已有测试数据集的研究工作,主要集中于已有测试数据集的选择和扩展等 2 个方面,其中,在已有测试数据集的选择方面,侧重于确定已有测试数据集中数据的优先级[4-5];在扩展已有的测试数据集方面,主要针对待覆盖的目标,如语句、分支以及路径等,运用搜索的方法,例如遗传算法,再生成新的测试数据[2].遗憾的是,已有方法仅是将测试数据的再生成,看作覆盖受影响程序元素的测试数据生成问题,而没有充分利用已有测试数据集的信息,这造成了大量有用信息的不必要浪费.

近年来,采用遗传算法,生成覆盖路径的测试数据,得到国内外学者的广泛关注,主要做法是:首先,把测试数据生成问题,转化为单目标或多目标优化问题;然后,基于待优化目标,定义用于遗传算法进化种群个体评价的适应度函数;最后,对由测试数据编码形成的种群进化一定代数,即可得到满足测试需求的测试数据.容易看出,遗传算法的进化种群能够携带穿越多个目标路径测试数据的信息,因此,执行一次遗传算法,能够得到覆盖多条目标路径的测试数据[6-10].

本文研究回归测试数据生成问题,在充分利用已有测试数据的基础上,通过执行一次遗传算法,迅速生成穿越所有受影响路径的测试数据.为此,首先,根据修改节点与目标路径的相关矩阵,对目标路径排序,并基于路径相似度,构建覆盖影响路径的测试数据生成问题的数学模型;然后,选用遗传算法求解前述模型时,利用覆盖已有目标路径的现有测试数据,编码后替代进化种群的部分个体.

本文的贡献主要体现在如下 3 个方面:(1)基于路径相似度和差异度,给出路径相关性的度量方法,并根据路径修改情况以及路径复杂度,对应当覆盖的目标路径排序,建立可用于回归测试数据集优化生成问题的数学模型;(2)利用遗传算法进化生成测试数据时,根据覆盖目标之间的路径差异度,将满足条件的已有测试数据编码,替代进化种群的部分个体;(3)将所提方法应用于 6 个规模不等的基准和工业程序测试中,并与多个已有方法比较,以检验本文方法生成回归测试数据的性能.

本文第 2 节是相关工作综述;第 3 节通过路径相关度,对需要覆盖的目标路径排序,并建立回归测试数据生成问题的数学模型;第 4 节阐述采用遗传算法求解上述模型时,已有测试数据的利用;第 5 节给出所提方法在基准和工业程序的应用与对比实验;最后,第 6 节对本文工作总结,并指出下一步工作计划研究的问题.

2 相关工作

我们知道,对一个软件维护后,需要对软件受影响的部分进行快速、全面的回归测试.到目前为止,与回归测试相关的研究工作主要包括测试需求约简、测试数据选择以及测试数据集扩大等.由于在回归测试时,已经存在一个测试数据集,因此,采用合适的方法,有效利用已有测试数据,以提高测试效率,是成为回归测试的关键.

在测试需求约简方面,Tallam 等人[11]提出基于概念格分析的测试需求约简方法,缩减已有测试数据集的规模.Zhang 等人[12]基于测试需求之间关系,约简测试需求,并将约简后的测试需求作为测试数据集生成和约简的基础,实现测试数据集优化.Sun[13]分析错误对应的前提条件之间的蕴含关系,减少需要满足的前提条件,以约简满足前提条件的测试数据.Zhang 等人[14]通过研究程序对应的控制

流图，减少与修改点无关的分支，以覆盖约简分支后构成的路径为目标，生成测试数据. 上述方法通过减少测试需求，降低了需要生成的测试数据数目，使得回归测试数据生成的效率提高. 但是，这些方法没有充分利用约简后测试需求之间的关系，生成覆盖受影响路径的测试数据.

关于回归测试已有测试数据的利用，已有研究工作主要包括已有测试数据集的选择和扩展等 2 个方面.

Nachiyappan 等人[15]利用搜索的方法构造用于再测试的数据集时，将已有测试数据作为进化算法初始种群的一部分，选择测试目标覆盖率高且执行成本较少的已有测试数据. Ye 等人[16]通过分析程序修改前后的活动图，根据程序更新对路径的影响，对路径分类，并基于已有测试数据覆盖分类后路径的情况，选择需要修改的测试数据. Kumar 等人[17]对比修改前后程序的路径，选择新增加和发生改变的路径，作为应该覆盖的目标，基于路径选择方法，达到选择测试数据的目的. Sampath 等人[18]针对目前的测试数据集的优化问题，包括测试数据选择、约简、优先问题给出测试数据优化问题的统一模式，并给出多评价准则下的测试数据的选择方法.

在上述测试数据选择方法中，Nachiyappan 等人选择测试数据的过程繁琐、耗时；Ye 与 Kumar 等人均基于路径选择方法，根据程序更新前后路径的变化，先选择目标路径，然后选择生成测试数据；Sampath 等人提出的方法只从现有测试数据集中，选择满足多个评价准则的测试数据，忽略了生成新的测试数据.

Taneja 等人[19]提出一种受影响路径的识别方法，并仅针对影响输出的路径进行测试. Shin 等人[20]先检验现有测试数据是否失效，然后，利用现有数据，并对输入空间中搜索到的测试数据评价，生成新的满足覆盖目标的测试数据. Santelices 等人[21]研究生成测试数据覆盖更新影响的语句时，分析程序的数据流和控制流，选择与修改点具有依赖关系的路径，生成测试数据覆盖上述路径. Xu 等人[22]提出的测试数据集扩大方法，基于已有测试数据运行修改后的程序，以找出发生改变的语句，对这些语句深度优先排序后，利用已有测试数据，生成新的测试数据.

在上述测试数据集扩大方法中，Taneja 等人通过减少受影响的路径，缩短生成测试数据需要的时间；Shin 等人利用验证后的原有测试数据，通过启发式方法再生成测试数据，但该方法得到的测试数据集具有冗余，还需要进一步约简；Santelices 等人给出的方法，未考虑已有测试数据的利用；而 Xu 等人在测试数据再生成时，没有建立选择已有测试数据的准则. 这意味着，这些方法均没有充分利用已有测试数据提供的有用信息. 另外，Xu 等人[23]运用遗传算法，生成满足覆盖分支准则的测试数据时发现，不同算法、不同的测试目标执行顺序以及是否利用新生成的测试数据，对测试数据生成的效率有很大影响，但是，他们没有利用影响测试数据生成的因素，提高回归测试数据生成效率.

Bueno 和 Maragathavalli 等人[24-25]在单元测试中，采用遗传算法生成覆盖多路径的测试数据时，以穿越的路径相似为准则，选择部分已生成的测试数据，以帮助生成覆盖其他目标路径的测试数据. 但是，如果在回归测试中，采用该方法选择测试数据，需要满足已穿越的路径与待覆盖的目标路径，具有以首节点为起点的多个相同连续节点. 遗憾的是，该方法遗漏了虽然没有满足上述条件，但与待覆盖的目标路径具有较大相关度的已穿越路径对应的测试数据，使得测试数据仅能在较小的输入域里选择；另一方面，在回归测试中，该方法并没有考虑到程序结构的变化，没有利用程序结构变化的信息来选择已有测试数据，帮助生成测试数据.

由上述可以看出，在回归测试中，虽然有不少方法或多或少用到已有测试数据信息，但是，这些方法主要存在以下问题：

(1) 没有利用路径的变化情况以及没有利用已覆盖的路径与未覆盖路径之间的关系，选择已有测试数据.

(2) 没有合理设定需生成测试数据覆盖的目标路径的次序，导致未能充分利用测试数据生成过程中的信息.

(3) 缺乏有效的指标选择已有测试数据，利用已有的未失效数据品质不高.

如果在采用遗传算法生成满足测试需求覆盖的测试数据时，解决以上问题，充分利用已有的测试数据，构成进化种群的部分个体，那么，将无疑会进一步发挥遗传算法生成测试数据的潜能，进而可以高效的进化生成回归测试数据.

3　回归测试数据生成问题的数学模型

本节首先定义一些与程序路径相关的基本概

念;然后,根据路径相关性,确定目标路径的覆盖次序;最后,建立回归测试数据生成问题的数学模型.

3.1 基本概念

定义 1. 节点. 记被测程序为 G,程序 G 的一个节点,记为 d,是该程序的一个语句块,满足在程序 G 的运行中,要么一起执行,要么都不被执行.

节点可以是一个循环结构的循环条件,也可以是一个判断结构的判断条件,还可能是一或多条连续语句. 每个节点有相应的前置约束条件表达式,包含该条件表达式的节点,称为分支节点.

定义 2. 路径. 程序 G 的一条路径,记为 p,为程序 G 在一组输入上运行时,控制流经过的节点序列 $d_1 d_2 \cdots d_n$,其中,n 为路径 p 的长度.

定义 3. 路径相关. 记 p_1 和 p_2 为程序 G 的 2 条路径,对应的节点序列分别为 $d_{11}, d_{12}, \cdots, d_{1|p_1|}$ 和 $d_{21}, d_{22}, \cdots, d_{2|p_2|}$,其中,$|p_1|$ 和 $|p_2|$ 分别为 p_1 和 p_2 包含的节点数. 如果 $p_1 \cap p_2 \neq \varnothing$,那么,称这 2 条路径是相关的;否则,称这 2 条路径不相关. 特别的,当 $p_1 = p_2$ 时,称这 2 条路径完全相关.

为了考察 2 条路径是否相关,需要对程序的路径统一编码. 为此,首先,对程序控制流图的所有分支节点广度优先遍历,记录这些分支节点的序号;然后,对程序插桩,并记录某路径在不同分支节点处,穿越真假分支的情况,穿越真分支记为 1;否则,记为 -1,如果没有穿越该分支结点,记为 0;最后,根据这些真值,对路径进行编码,编码长度等于分支节点数.

如图 1 所示,某程序包含如下 4 条路径 p_1、p_2、p_3 以及 p_4,这些路径包含的节点分别为{1,2,4,5,7},{1,2,4,6,7},{1,3,4,5,7},{1,3,4,6,7}. 采用上述编码方法,这些路径的编码分别为 11、1-1、-11 以及-1-1. 与采用节点序列表示某路径相比,采用编码表示路径,需要的编码长度更短,且所有路径的编码长度均相同,这使得考察路径的相关性更容易.

```
int main(int x, int y)
{
 1:    if (x>=1)
 2:       t=0;
       else
 3:       t=2*x;
 4:    if (y>0)
 5:       t=t+y;
       else
 6:       t=t-y;
 7:    return t;
}
```

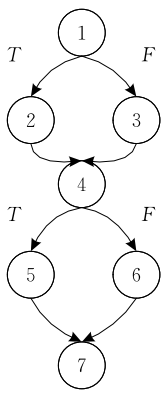


图 1 示例程序及其对应的控制流图

定义 4. 路径相似度. 对于程序 G 的 2 条路径 p_1 和 p_2,从首节点开始,这 2 条路径相同编码的序列长度,与路径编码总长度的比,称为路径 p_1 和 p_2 的相似度,记为 $S(p_1, p_2)$,那么,$S(p_1, p_2)$ 可以表示为

$$S(p_1, p_2) = \frac{1}{n} \sum_{l=1}^{n} \alpha_l(p_1, p_2) \tag{1}$$

其中,

$$\alpha_l(p_1, p_2) = \begin{cases} 1, & \text{前 } l \text{ 个分支节点真值相同} \\ 0, & \text{否则} \end{cases}.$$

由定义 4 可以看出:(1) 当 $0 < S(p_1, p_2) < 1$ 时,路径 p_1 和 p_2 至少存在 1 个从首节点出发的相同节点;(2) $S(p_1, p_2)$ 越大,这 2 条路径的相关性越大;特别的,当 $S(p_1, p_2) = 1$ 时,路径 p_1 和 p_2 完全相关.

定义 5. 路径差异度. 对于程序 G 的 2 条路径 p_1 和 p_2,这 2 条路径真值不同的分支节点数与路径编码总长度的比,称为路径 p_1 和 p_2 的差异度,记为 $D(p_1, p_2)$,那么,$D(p_1, p_2)$ 可以表示为

$$D(p_1, p_2) = \frac{1}{n} \sum_{l=1}^{n} \beta_l(p_1, p_2) \tag{2}$$

其中,

$$\beta_l(p_1, p_2) = \begin{cases} 0, & \text{第 } l \text{ 个分支节点真值相同} \\ 1, & \text{否则} \end{cases}.$$

由定义 5 可以看出:(1) 当 $0 < D(p_1, p_2) < 1$ 时,p_1 和 p_2 至少存在一个相同的分支节点;(2) $D(p_1, p_2)$ 越小,这 2 条路径的相关性越大;特别的,当 $D(p_1, p_2) = 0$ 时,p_1 和 p_2 完全相关.

路径相似度和差异度均能从某一侧面,反映 2 条路径 p_1 和 p_2 的相关度. 对于待覆盖的一条目标路径以及任意一个测试数据,由定义可以得出:

(1) 路径相似度可表明从第一个节点开始,测试数据穿越的路径对目标路径的接近程度. 在程序中,由于前面的节点往往对后面的节点具有占优关系,所以,在生成覆盖目标路径的测试数据过程中,路径相似度可用于评价测试数据从程序入口开始,依次满足目标路径分支节点真值的序列长度.

(2) 与路径相似度相比,路径差异度更能反映 2 条路径所有分支节点的差异. 测试数据穿越的路径与目标路径的差异度,反映了测试数据没有满足

的目标路径上的分支节点真值的个数. 如果不同的测试数据覆盖相同的分支，这说明，这些测试数据的值或相互关系可能具有某些共同的特征. 例如，不同测试数据的值，同时都满足一定的谓词条件，那么，这些测试数据能够覆盖相同的分支. 显然，2 条路径的差异度越小，这些路径之间的相关性就越高，从而穿越这两条路径的测试数据覆盖相同的分支与节点数越多. 如果能从与目标路径具有相关性的路径中提取尽可能多的测试数据信息，这意味着，可以引导测试数据的执行轨迹逐渐向目标路径收敛，进而得到覆盖目标路径的测试数据.

基于以上分析，本文在构建测试数据生成问题的数学模型时，选择路径相似度作为目标函数，而在选择已有测试数据时，基于路径差异度.

3.2　基于相关性的目标路径排序

鉴于存在多条需要覆盖的目标路径，且生成覆盖不同路径的测试数据，需要的计算资源不同，因此，合理的目标路径覆盖顺序，能够有效提高测试数据生成效率. 现有的确定目标路径覆盖顺序的方法有随机法和深度优先搜索法等[23]. 鉴于回归测试中已存在原有的测试数据集，而不同的目标路径与已有测试数据集覆盖的路径具有不同的相关度，如果利用路径相关性，确定被覆盖的目标路径顺序，那么，将会利用已有的测试数据集提供的信息，提高测试数据生成效率.

记程序 G 所有修改的节点构成的集合为 $C=\{c_1,c_2,\cdots,c_m\}$，所有需要覆盖的目标路径构成的集合为 $P=\{p_1,p_2,\cdots,p_s\}$，其中，m 和 s 分别为修改节点和目标路径数，根据修改节点与目标路径的关系，可以构造如下相关矩阵，记为 $\boldsymbol{\gamma}(C,P)$：

$$\boldsymbol{\gamma}(C,P)=\begin{bmatrix}\gamma_{11} & \gamma_{12} & \cdots & \gamma_{1s}\\ \gamma_{21} & \gamma_{22} & \cdots & \gamma_{2s}\\ \vdots & \vdots & \ddots & \vdots\\ \gamma_{m1} & \gamma_{m2} & \cdots & \gamma_{ms}\end{bmatrix} \tag{3}$$

式中，$\gamma_{ij}\in\{0,1\}$，且 $\gamma_{ij}=1$ 表示路径 p_j 经过修改节点 c_i；$\gamma_{ij}=0$ 表示 p_j 不经过 c_i.

根据式(2)，容易知道，如果一个目标路径包含的修改节点越少，那么，该路径与程序修改之前的某一条或几条路径的路径相关性就越大，进而穿越这些路径的测试数据，对生成覆盖该目标路径的测试数据越有利用价值. 鉴于此，基于 $\boldsymbol{\gamma}(C,P)$，提取反映程序变动前后，目标路径 p_j 变动的节点数，记为 $N(p_j,C)$，其表达式为

$$N(p_j,C)=\sum_{i=1}^{m}\gamma_{ij} \tag{4}$$

此外，根据式(1)，在程序 G 中，第 1 个修改节点的位置越靠后，该路径与程序修改之前的某一条或几条路径的相关性就越大，那么，使用已有测试数据求解，较容易生成覆盖该路径的测试数据. 基于 $\boldsymbol{\gamma}(C,P)$，第 1 个修改节点的位置，记为 $F(p_j)$，可以表示为

$$F(p_j)=\min_{i\in\{1,2,\cdots,m\}}\{i\mid\gamma_{ij}\neq 0\} \tag{5}$$

对待覆盖的目标路径排序时，还应考虑目标路径包含的节点数. 一般而言，一条目标路径包含的节点数越多，那么，生成的测试数据应当满足的分支谓词条件越多，从而生成能够覆盖该路径的测试数据越困难. 这表明，该目标路径在排序中应该越靠后，使得在生成覆盖该目标路径的测试数据过程中，可以利用更多已经得到的覆盖其他路径的测试数据，而覆盖程序 G 其他路径的测试数据，包含的满足分支谓词条件的相关信息，可以用来提高覆盖节点数较多的目标路径的测试数据生成效率.

综合上述因素，目标路径 p_j 被覆盖的优先级，记为 $PR(p_j)$，可以表示为

$$PR(p_j)=-w_1N(p_j,C)+w_2F(p_j)-w_3|p_j| \tag{6}$$

式中 w_1、w_2 以及 w_3 分别为 $N(p_j,C)$、$F(p_j)$ 以及 $|p_j|$ 的权值，且 $w_1\gg w_2\gg w_3>0$.

根据式(6)，能够得到所有目标路径的覆盖优先级. 这样一来，对于所有需要覆盖的目标路径 P，按照覆盖优先级从大到小排列[26]，即可得到排序后的目标路径，记为 $P^*=\{p_1^*,p_2^*,\cdots,p_s^*\}$.

3.3　测试数据生成问题的数学模型

测试数据生成问题可以描述为：对于修改后的程序 G，输入向量为 $\boldsymbol{x}$，输入域为 X，在 X 内生成测试数据 $x_1,x_2,\cdots,x_s$，且以 $x_1,x_2,\cdots,x_s$ 为程序 G 的输入，执行 G 之后，穿越的路径分别为 $p_1^*,p_2^*,\cdots,p_s^*$.

对于程序 G 的某一输入向量 $\boldsymbol{x}$，记 $\boldsymbol{x}$ 穿越的路径为 $p(\boldsymbol{x})$. 根据前述路径相关性定义，如果 $p(\boldsymbol{x})$ 与第 j 个目标路径 p_j^* 完全相关，那么，$\boldsymbol{x}$ 即为期望的测试数据. 此时，在输入域中，$S(p(\boldsymbol{x}),p_j^*)$ 达到最大值，$D(p(\boldsymbol{x}),p_j^*)$ 达到最小值. 这样一来，求取覆盖目标路径 p_j^* 的测试数据生成问题，就能够转变成求取 $S(p(\boldsymbol{x}),p_j^*)$ 的最大值或 $D(p(\boldsymbol{x}),p_j^*)$ 的最小值问题. 考虑程序 G 靠近程序入口的节点，对后面的节点具有占优关系，本文以路径相似度作为优化问题的目标函数. 因此，对于所有目标路径而言，生成覆盖目标

路径 $p_1^*,p_2^*,\cdots,p_s^*$ 的测试数据问题，就可以转化为求取 $S(p(\boldsymbol{x}),p_1^*),S(p(\boldsymbol{x}),p_2^*),\cdots,S(p(\boldsymbol{x}),p_s^*)$ 的最大值问题，其数学模型为

$$\begin{aligned}&\max\ (f_1(\boldsymbol{x}),f_2(\boldsymbol{x}),\cdots,f_s(\boldsymbol{x}))\\&\text{s.t.}\ \ \boldsymbol{x}\in X\end{aligned}\tag{7}$$

式中，$f_j(\boldsymbol{x})=S(p(\boldsymbol{x}),p_j^*)$.

由式(7)可以看出，本文的测试数据生成问题建模为一个多目标优化问题，使得生成的测试数据集能够覆盖所有的目标路径.但是，该问题与传统的多目标优化问题又有很大区别，主要体现在：(1)传统的多目标优化问题在决策变量所在的域中，寻找一个 Pareto 解集，使得该解集中任一解对应的所有目标同时达到最优；式(7)表示的优化问题是在测试数据所在的域中，寻找一个测试数据集，使得该集合中任一测试数据仅针对某一目标达到最优；(2)传统多目标优化问题的目标函数求解没有先后顺序；式(7)表示的优化问题的目标函数求解具有先后顺序，具体的讲，只有生成使得 $f_j(\boldsymbol{x})$ 最大的测试数据，才进一步生成使得 $f_{j+1}(\boldsymbol{x})$ 最大的测试数据，直到生成使得 $f_s(\boldsymbol{x})$ 最大的测试数据.这样做的动机是，充分利用已覆盖其他路径测试数据的蕴含信息，提高生成测试数据的效率.

4 基于遗传算法的测试数据生成

本节主要阐述式(7)描述优化问题的求解方法，思想是：基于遗传算法，采用传统的遗传策略进化种群，按照式(7)的目标函数顺序，依次生成使得这些目标函数值最大的测试数据；在种群进化过程中，按照路径差异度准则，利用部分已覆盖路径的测试数据，编码后替换当前进化种群的个体，使得执行一次遗传算法，生成所有覆盖目标路径的测试数据.

4.1 覆盖已有路径的测试数据选择

采用遗传算法求解优化问题时，如果进化种群的性能高，那么，将有利于得到期望的最优解.因此，生成覆盖某一目标路径的测试数据时，采用合适的方法，提高进化种群的性能，是十分必要的.

如第 2 节所述，Bueno 和 Maragathavalli 等人[24-25]采用遗传算法生成覆盖多路径的测试数据时，以穿越的路径相似为准则，选择部分已生成的测试数据，替换进化种群的个体，以生成覆盖其他目标路径的测试数据.但是，该方法并没有充分利用已有测试数据.

如图 1 所示，假设待覆盖的目标路径由语句 1、3、4、6 以及 7 组成，其余路径均有测试数据覆盖.如果以路径相似度作为测试数据选择的准则，那么，只能选择 1 条与待覆盖的目标路径相似度为 0.5 的路径对应的测试数据，该路径由语句 1、3、4、5 以及 7 组成；如果以路径差异度作为选择的准则，那么，将能选择 2 条与待覆盖的目标路径差异度均为 0.5 的路径对应的测试数据，其中，第 1 条路径由语句 1、2、4、6 以及 7 组成，而第 2 条路径由语句 1、3、4、5 以及 7 组成.虽然这 2 个准则都能够利用已有的测试数据，提高进化种群的性能，但是，以路径差异度作为测试数据选择的准则，显然能够利用更多已覆盖路径对应的测试数据.特别是，当程序修改节点的位置在控制流图中比较靠前时，使用路径相似度准则，很难找到满足条件的已有测试数据；而使用路径差异度准则，却容易找到满足条件的已有测试数据.

现在分析路径与输入向量之间的关系.记程序 G 的所有分支节点包含的条件表达式构成的集合为 $\{H_1(\boldsymbol{x}),H_2(\boldsymbol{x}),\cdots,H_n(\boldsymbol{x})\}$，将 $p(\boldsymbol{x})$ 与 p_j^* 取值不相同的分支节点条件表达式前移，能够得到一个新的分支节点条件表达式序列，该序列可以表示为 $H_1^*(\boldsymbol{x}),\cdots,H^*_{nD(p(\boldsymbol{x}),p_j^*)}(\boldsymbol{x}),H^*_{nD(p(\boldsymbol{x}),p_j^*)+1}(\boldsymbol{x}),\cdots,H_n^*(\boldsymbol{x})$，其中，$H_1^*(\boldsymbol{x}),\cdots,H^*_{nD(p(\boldsymbol{x}),p_j^*)}(\boldsymbol{x})$ 是 $p(\boldsymbol{x})$ 与 p_j^* 取值不相同的分支节点条件表达式，而 $H^*_{nD(p(\boldsymbol{x}),p_j^*)+1}(\boldsymbol{x}),\cdots,H_n^*(\boldsymbol{x})$ 是 $p(\boldsymbol{x})$ 与 p_j^* 取值相同的分支节点条件表达式.进一步，统计 $H_1^*(\boldsymbol{x}),\cdots,H^*_{nD(p(\boldsymbol{x}),p_j^*)}(\boldsymbol{x})$ 的输入分量个数，记为 t.容易看出，t 越小，满足目标路径的分支条件表达式的输入分量就越多.在回归测试中，一旦被测程序确定，那么，$H_1^*(\boldsymbol{x}),\cdots,H^*_{nD(p(\boldsymbol{x}),p_j^*)}(\boldsymbol{x})$ 的输入分量个数就是不变的.因此，路径差异度 $D(p(\boldsymbol{x}),p_j^*)$ 越小，那么，t 值越小，此时，通过种群进化求解的输入分量个数就越少.

根据上述分析，选择覆盖程序修改后路径的测试数据方法如下：首先，生成并更新覆盖部分目标路径的已有测试数据构成的集合，记为 T，该集合包含的元素为：(1)覆盖程序修改没有影响的目标路径的测试数据；(2)程序修改之后，在测试数据生成过程中，获取的穿越待覆盖目标路径的测试数据；然后，对于须覆盖的目标路径 p_j^*，计算集合 T 中任一测试数据覆盖的路径与 p_j^* 的差异度，并选择差异度小于或等于某一阈值的测试数据，替换当前进化种群的部分个体.

4.2　进化种群部分个体的替代

当种群进化以生成覆盖目标路径 p_j^* 的测试数据时，记当前种群需要替代的个体数为 e，采用第 4.1 节给出的方法，从集合 T 中选择多个覆盖已有路径的测试数据，替换当前种群的 e 个个体. 记集合 T 中满足已覆盖路径与 p_j^* 的差异度小于指定阈值的测试数据个数为 e'，那么，存在以下 2 种情况：

(1) $e'<e$；

这说明，集合 T 中满足条件的测试数据不够. 此时，选择上述 e' 个覆盖已有路径的测试数据，与随机生成的 $e-e'$ 个测试数据一起，替换当前种群的 e 个个体.

(2) $e'\geq e$；

这说明，集合 T 中满足条件的测试数据比较多. 此时，需要从 e' 个测试数据中选择 e 个，替换当前种群的部分个体. 鉴于新生成的测试数据满足修改节点的分支条件表达式，因此，选择 e 个用于替换个体的测试数据时，优先考虑新生成的测试数据.

针对上述 2 种情况，首先，构造集合 T 的数据结构为带头结点的链表，链表中的数据为能够覆盖修改后程序目标路径的原有测试数据，当生成覆盖目标路径的新的测试数据后，更新集合 T，在头结点指向的位置插入新生成的测试数据. 可以看出，集合 T 在链表中的测试数据穿越路径的顺序与问题模型的目标路径顺序相反；其次，进化种群部分个体的替代策略，如算法 1 所示，从头结点开始，按照第 4.1 节设定的标准，从集合 T 中依次选择满足条件的测试数据，编码后对当前种群 Pop 的前 e 个个体替换，如果集合 T 的元素遍历完毕，且替换的个体数 $e'<e$，那么，随机生成 $e-e'$ 个测试数据，编码后置换余下的个体.

算法 1.　进化种群部分个体替代的伪代码.

```
输入：&Pop, e, &T
输出：替换后的种群
BEGIN
1.    e'=0; i=0;
2.    WHILE(e'<e&&!EOF)
2.1     x=GetElem(T,i)
2.2     IF(D(p(x), p_j^*)<z)
2.2.1     Pop[e']=Code(x);
2.2.2     e'++;
3.      i++;
4.    END WHILE
5.    WHILE(e'<e)
5.1     Pop[e']=Code(Random());
6.    END WHILE
END
```

执行替代算法后的种群由以下个体组成：

(1) e' 个来源于集合 T 中满足差异度阈值的测试数据，该部分个体穿越的路径与 p_j^* 具有一定的相关度，对应的测试数据满足 p_j^* 上的部分分支条件表达式.

(2) 当 $e'<e$ 时，随机生成的 $e-e'$ 个个体.

(3) 替代前，生成覆盖目标路径 p_{j-1}^* 的个体.

由种群构成可以看出，执行替代算法后，既可利用原有测试数据和新生成的测试数据引导种群进化，又可利用生成覆盖目标路径 p_{j-1}^* 的测试数据所在种群的信息，那么，执行替代算法后，种群的性能将明显提升，且具有较好的多样性.

4.3　路径差异度阈值的设定

考虑到不同程序中，目标路径数与集合 T 包含的测试数据数不同，本文的差异度阈值设定如下：

$$z=\min\left(1-\frac{1}{n},\frac{1}{n}+\frac{e}{|P^*|}\right) \tag{8}$$

由式(8)可以看出：

(1) 当 z 等于 $1-1/n$ 时，要求已穿越的路径与目标路径，至少有一个节点相同，此时，这些路径具有相关性. 因此，集合 T 中所有穿越与目标路径具有相关性路径的测试数据，均可以作为选择的测试数据；

(2) 当 z 等于 $\frac{1}{n}+\frac{e}{|P^*|}$ 时，差异度阈值最小为 $1/n$，此时，要求已穿越的路径与目标路径，最多有一个节点不相同，那么，这些路径与目标路径将具有最大的相关性. 因此，选择集合 T 中穿越这些路径的测试数据. 进一步，$|P^*|$ 越大于 e，那么，z 的值越小，此时，要求从集合 T 中选择的测试数据覆盖的路径，与目标路径的相关性越大.

需要说明的是，通过式(8)的差异度阈值，本文提供了一种已有测试数据的利用方法，使得从集合 T 中选择的测试数据覆盖的路径，与当前目标路径具有较大的相关性. 但是，该方法设定的差异度阈值未必是最优的，还可以通过其他的方式，设置更加合理的差异度阈值. 关于这方面的内容，已经超出本文的范围.

4.4　性能分析

对遗传算法每一代种群的进化过程分析，由于对种群中个体计算适应度时需要执行被测程序，因此，算法的时间复杂度主要由适应度函数的求解次

数和复杂度决定.

假设种群规模为 h,目标路径节点数为 n,任一个体计算适应度需调用被测程序,且与被测程序的 n 个节点对比路径相似度,因此,传统方法计算所有个体的适应度的时间复杂度为 $O(h\times n)$.

在种群进化过程中,当使用本文方法生成穿越新的未覆盖目标路径的测试数据时,进化种群需要替换部分个体,此时,需额外计算集合 T 中测试数据所覆盖路径与目标路径的差异度,令路径差异度的计算复杂度为 θ,可以得到,本文方法的最大时间复杂度为 $O(h\times n)+|T|\times O(\theta)$. 而集合 T 中测试数据穿越的路径已知,可以直接根据式(2)计算路径差异度,并不需要运行被测程序,那么,θ 为常量,因此,本文方法的最大时间复杂度为 $O(h\times n)+|T|\times O(\theta)=O(h\times n)$. 和传统方法比较,本文方法并没有增加算法的时间复杂度.

算法每一代进化过程的空间复杂度主要取决于种群中所包含的个体数目,因此,传统遗传算法的空间复杂度可以表示为 $O(h)$. 由图 2 可以看出,使用本文方法,当进化种群进行部分个体替代时,需要存储集合 T 中的测试数据,空间复杂度可以表示为 $O(h+|T|)$. 但是,进化过程中,种群中的个体数目并未增加,对于替换部分个体后种群的进化过程,每一代的空间复杂度仍为 $O(h)$. 极端情况下,当 T 中测试数据能够覆盖所有目标路径时,本文方法每一代的空间复杂度,最坏情况下为 $O(h+|p^*|)$,但很明显的是,此时已得到覆盖所有目标路径的测试数据. 这也说明,$|T|$ 越大,越有利于降低算法的执行代数.

可以看出,和传统遗传算法相比,采用本文方法并没有增加算法的时间复杂度,虽然,在种群替换过程中,需要额外的存储空间,但是,这提升了种群的性能,从而降低了进化代数,提高了遗传算法求解问题的效率.

4.5 算法步骤

本文提出的采用遗传算法生成回归测试数据的步骤如下:

(1)生成覆盖修改后被测程序目标路径的测试数据构成的集合 T,同时,初始化遗传算法的运行参数.

(2)判断目标路径集 $P^*=\{p_1^*,p_2^*,\cdots,p_s^*\}$ 是否为空,如果是,那么,算法结束;否则,选择 P^* 中下标最小的未覆盖的路径,作为当前目标路径,按照第 4.2 节的方法,替代进化种群的部分个体.

(3)以式(7)的目标函数作为评价进化个体性能的适应度函数,如果期望的测试数据已经生成,或进化代数大于设定值,那么,将当前目标路径从 P^* 中删除,转步(2).

(4)运行遗传算法,进行选择、交叉、变异操作,转步(3).

5 实 验

为了评价本文提出的回归测试数据进化生成方法的性能,选择多个基准和工业程序,作为被测程序进行实验. 首先,以三角形分类程序、冒泡排序为例,说明本文方法的步骤与效果;接着,针对选择的被测程序,考察本文方法生成测试数据需要的时间、目标路径覆盖率以及已有测试数据利用率等,并与其他方法比较;然后,对上述实验结果进行分析,并回答如下 4 个问题:

问题 1. 与基于路径相似度相比,基于路径差异度选择测试数据,能否更有效地利用已有的测试数据?

问题 2. 与基于未排序的目标路径建立问题的数学模型相比,本文基于确定覆盖目标路径优先顺序建立的模型,能否提高生成测试数据效率?

问题 3. 与其他方法相比,本文方法能否花费更少的时间达到更高的目标路径覆盖率?

问题 4. 在不同程序的不同修改版本中,本文方法是否具有通用性?

最后,选择三角形分类程序,进一步考察不同差异度阈值以及不同修改版本,对本文方法的影响.

实验的硬件环境为 Intel Due-Core 2.0GHz CPU 和 2GB RAM 内存. 所有方法均采用 VC++ 6.0 实现.

5.1 实例分析

本小节以三角形分类程序、冒泡排序为例,说明本文方法的步骤与效果.

对三角形分类程序进行修改,三角形分类程序的 2 个实现版本,如图 2 所示,一个作为修改前程序,有 4 条可行路径;另一个作为修改后程序,有 19 条可行路径,输入分量的取值范围为[0,2047].

由图 2 容易看出,与修改前程序相比,修改后的程序增加了 11 条路径. 由于这 2 个程序的功能相同,因此,覆盖修改前程序路径的测试数据,仍能覆盖修改后程序的部分路径,这些测试数据对修改后程序路径的覆盖率为 21.1%.

修改前程序	修改后程序	分支节点
Triangle(int *a*, int *b*, int *e*)	Triangle(int *a*, int *b*, int *c*)	
if(c> *a*+*b*||*a*>*b*+*c*||*b*>*c*+*a*)	{if (*a*>*b*)	1
{ printf("NOT A ANGLE")	{swap (*a*,*b*); //交换*a*, *b*值}	
else	if (*a*>*c*)	2
{ if (*a*==*b*||*a*==*c*||*b*==*c*)	{swap (*a*, *c*);}	
if (*a*==*b*&&*b*==*c*)	if (*b*>*c*)	3
printf("EQUILATERAL");	{swap (*b*, *c*); }	
else	if (*a*>*b*+*c*)	4
printf("ISOSCELES");	printf ("NOT A ANGLE")	
else	else	
printf("SCALENE");}}	{ if (*a*==*b*||*b*==*c*)	5
	if (*a*==*b*&&*b*==*c*)	6
	printf("EQUILATERAL")	
	else	
	printf("ISOSCELES");	
	else	
	printf ("SCALENE");}}	

图 2　三角形分类程序的 2 个实现版本

首先，按照第 3.1 节所述，对修改后程序的分支结点，广度优先遍历，将所有路径编码为分支节点序列条件表达式的真值，19 条路径的编码如表 1 所列，表中，粗体显示的路径，为利用修改前程序的测试数据能够覆盖的路径.

表 1　三角形分类实例程序的执行结果

目标路径编码	路径序号	$\|T\|$	测试数据利用数	进化代数	是否找到测试数据
111100	1	4	3	2	True
1-11100	2	5	5	3	True
1-1-1100	3	6	5	4	True
-111100	**4**	**7**	**5**	**4**	**True**
-11-1100	5	7	5	5004	False
-1-11100	6	7	5	5005	True
-1-1-1100	7	8	5	5007	True
111-1-10	8	9	6	5008	True
1-11-1-10	9	10	6	5009	True
1-1-1-1-10	**10**	**10**	**6**	**5009**	**True**
-111-1-10	11	11	7	5013	True
-1-11-1-10	12	12	7	5020	True
-1-1-1-1-10	13	13	7	5021	True
1-11-11-1	14	14	8	10021	False
1-1-1-11-1	**15**	**14**	**8**	**10021**	**True**
1-1-1-11-1	16	14	8	10123	True
-1-11-11-1	17	15	8	10129	True
-1-1-1-11-1	18	16	8	15129	False
-1-1-1-111	**19**	**16**	**8**	**15129**	**True**

接着，采用第 3.2 节的方法，得到这 19 条路径的优先级，并对这些路径按照优先级排序，其序号如表 1 的"路径序号"所列. 基于排序后的路径，按照式(7)建立该程序回归测试数据生成问题的数学模型.

基于该模型，将图2修改前程序的测试数据，保存在集合 T 中. 从表 1 可以看出，对于第 1 条目标路径，集合 T 中包含 4 个测试数据. 随着覆盖目标路径测试数据的不断生成，集合 T 包含的元素个数逐渐增加.

然后，按照第 4.5 节的算法步骤，运行遗传算法，取交叉和变异概率分别为 0.8 和 0.15，用于生成覆盖每个目标路径测试数据的种群最多进化 5000 代，种群规模为 50，需要替代的个体数 $e=20$. 根据 e 的值和目标路径数，由式(8)可以计算出，示例程序的差异度阈值为 5/6. 按照第 4.1 节所述，在运行遗传算法过程中，从集合 T 中选择穿越与目标路径具有相关性路径的测试数据，替换当前种群的部分个体.

最后，统计实验结果，包括已有测试数据利用的个数、时间消耗以及目标路径覆盖率. 由于三角形程序简单，本小节采用进化代数衡量时间消耗. 由表 1 可知，对于三角形分类程序，生成所有测试数据，需要的进化代数为 15 129 代，目标路径覆盖率为 84.1%，已有测试数据利用的个数平均为 6.3. 这说明，本文方法已有测试数据的利用率高，使得需要很少的进化代数，即可生成具有高目标路径覆盖率的测试数据.

以 4 个数冒泡排序为例，在包含循环嵌套结构的程序中，验证本文方法的实用性. 如图 3 所示，冒泡排序的输入分量的取值范围为[0,63]，代码第 5 行语句由 for($j=1$; $j<3-i$; $j++$)修正为 for($j=1$; $j<4-i$; $j++$).

```
1. void bubble(int a[4])
2. {
3.    int i,j;
4.    for(i=0;i<4;i++)
5.       for(j=1;j<4-i;j++)
6.          if(c[j-1]>c[j])//1,2,3,4,5,6
7.          {
8.             int temp=a[j-1];
9.             a[j-1]=a[j];
10.            a[j]=temp;
11.         }
12.}
```

图 3 冒泡程序的修改后版本

在语句 6 之后插桩,记录循环体每次执行,语句 6 的分支谓词取值,由图 3 容易得出,程序修改前共有 3 个分支结点,6 条有效路径;修改后程序共有 6 个分支结点,24 条有效路径.由于这 2 个程序的功能相同,因此,覆盖程序修改前路径的测试数据,仍能覆盖修改后程序的部分路径,这些测试数据对修改后程序路径的覆盖率为 25%.按照三角形分类程序实例分析描述的步骤以及参数,实验结果如表 2 所示.需注意的是,与三角形分类程序实例不同,在 4 个数的冒泡排序实例中,由于目标路径的修改结点位置以及数目都相同,所以,采用第 2 节所述方法,目标路径的优先级相等.因而,冒泡排序实例中的目标路径顺序,是按照本文提出的路径编码方法编码后的默认顺序,但是,在该情况下,目标路径间的差异度值较小,可利用的测试数据增加.

从表 2 可以得出,采用本文方法,已有测试数据利用的个数平均为 12.4,显著高于三角形分类实例中的已有测试数据的平均利用数,同时,目标路径覆盖率达到 95.8%,生成所有测试数据,需要的进化代数为 5532 代.可以看出,在 4 个数的冒泡排序实例中,本文方法充分利用了已有测试数据,体现了较高的效率,这也说明本文方法对于包含循环嵌套结构的程序也是适用和高效的.

表 2 冒泡排序实例程序的执行结果

目标路径编码	路径序号	$\|T\|$	测试数据利用数	进化代数	是否找到测试数据
-1-1-1-1-1-1	1	6	6	23	True
-1-11-1-1-1	**2**	**7**	**7**	**24**	**True**
-1-11-11-1	3	7	7	5024	False
-1-11-111	4	7	6	5078	True
-11-1-1-1-1	**5**	**7**	**7**	**5079**	**True**
-11-11-1-1	6	7	7	5152	True
-111-1-1-1	7	8	8	5189	True
-111-11-1	8	9	9	5250	True
-111-111	**9**	**10**	**10**	**5251**	**True**
-1111-1-1	10	10	10	5275	True
-11111-1	11	11	11	5315	True
-111111	12	12	11	5331	True
1-1-1-1-1-1	**13**	**13**	**12**	**5332**	**True**
1-11-1-1-1	14	13	13	5357	True
1-11-11-1	15	14	14	5361	True
1-11-111	16	15	14	5368	True
11-1-1-1-1	17	16	16	5382	True
11-11-1-1	**18**	**17**	**16**	**5383**	**True**
111-1-1-1	19	17	17	5463	True
111-11-1	20	18	18	5472	True
111-111	**21**	**19**	**19**	**5473**	**True**
1111-1-1	22	20	20	5475	True
11111-1	23	21	20	5479	True
111111	24	22	20	5532	True

5.2 对比实验结果

为了验证本文方法的有效性,除了三角形分类程序之外,还选用了 6 个数的冒泡排序程序、赫夫曼编码,flex 词法分析器以及西门子工业程序集中的 2 个程序,上述程序广泛应用于验证不同测试方法的有效性[27-28],其基本信息如表 3 所列.

表 3 被测程序与实验设置

被测程序	代码行数	函数个数	输入个数	输入范围	编码方式	插桩节点数	目标路径条数
三角形分类	27	1	3	[0,2047]	二进制	6	19
冒泡排序	24	1	6	[0,63]	二进制	15	72
huffcode	153	2	8	[0,10000]	实数	20	360
tcas	138	8	12	[0,10000]	实数	10	96
replace	516	21	可变	[0,128]	字符	30	210
flex	10055	162	可变	[0,128]	字符	64	720

实验中,对于不同的程序,进化个体采用不同的编码方式.对这些被测程序进行修改,将仍能穿越改后程序待覆盖路径的测试数据,加入到集合 T.集合 T 覆盖 6 个被测程序的目标路径数分别为 6、24、120、32、70 以及 100.算法所需参数的取值与第 5.1 节完全相同.对于每一被测程序,实验结果均为 20 次实验的平均值.

首先,为了评价本文采用的测试数据利用准则的性能,与使用路径相似度准则[25-26]选择测试数据比较,并采用了如下指标评价不同准则的性能:

(1)利用数,指在遗传算法生成测试数据过程中,分别使用路径差异度和路径相似度,作为选择测试数据的准则,生成覆盖每条目标路径的测试数据时,利用的已有测试数据的个数.

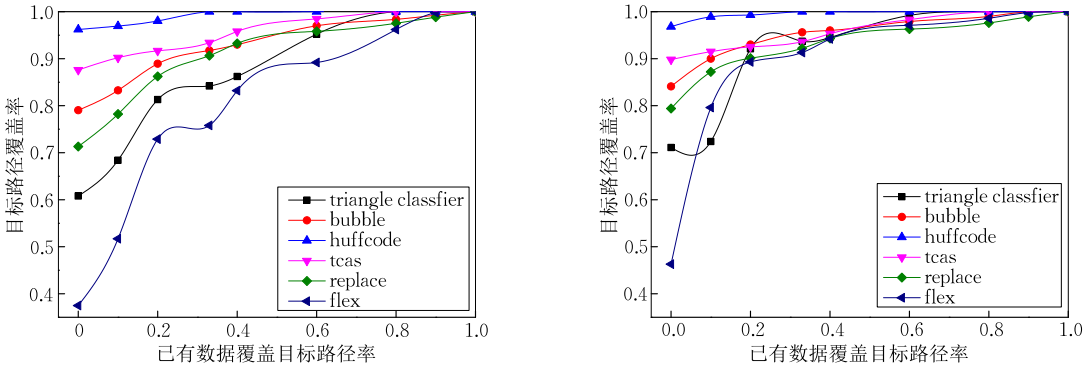
(2)适应值提升率,指替换当前种群的部分个体后,种群适应值变大的目标函数个数,与所有利用已有测试数据的目标函数个数的比值.

(3)命中率,指通过已有测试数据生成测试数据时,能够有测试数据覆盖的目标路径数,与所有目标路径数的比值.

然后,为了评价本文方法,记为 PSEGA 的有效性,将该方法与传统的回归测试数据生成方法(NGA)[25]、优先策略方法(PRGA)以及使用已有测试数据的方法(EGA),应用于基准和工业程序的测试,并从生成测试数据需要的时间、目标路径覆盖率以及通用性等方面进行对比.其中,NGA 直接采用遗传算法,按照随机覆盖目标路径的顺序,生成测试数据;PRGA 先根据本文提出的方法,确定目标路径的覆盖次序,然后,利用遗传算法,依次生成覆盖目标路径的测试数据;EGA 首先随机设定目标路径的覆盖顺序;接着,求出测试数据覆盖的路径与目标路径的差异度,并筛选有用的测试数据,替换当前种群的部分个体;最后,利用更新后的种群进化生成测试数据.

上述方法生成覆盖目标路径需要的时间和目标路径覆盖率,如下文 5.3 节表 5 所列,表中,某方法需要的时间越少,且目标路径的覆盖率越高,那么,该方法的性能就越优.由表 5 可知,本文方法和 EGA 是效率较高的 2 种,这 2 种方法都用了已有测试数据,替换当前种群的部分个体.

最后,针对被测程序的不同修改版本,考察集合 T 中已有测试数据覆盖的目标路径数,与所有目标路径数的比值,对目标路径覆盖率的影响,并与 EGA 方法比较,结果如图 4 所示.



(a) EGA覆盖率变化图　　(b) PSEGA覆盖率变化图

图 4　不同已有数据比例下,EGA 与本文方法生成测试数据的目标路径覆盖率

5.3　实验结果分析

本节对上述实验结果进行分析,并回答前面提出的 4 个问题.

(1)对问题 1 的回答

由表 4 可以得出,与使用路径相似度准则相比,采用本文的方法,可以更有效地利用已有数据.

① 对于每一被测程序,使用路径差异度,作为测试数据选择准则,生成覆盖目标路径的测试数据时,利用的已有测试数据均多于或等于使用路径相似度准则.

② 与使用路径相似度准则相比,本文方法的命中率平均提高了 6.1%.这说明,与使用路径相似度准则选择测试数据相比,采用本文的方法,使用已有测试数据的效率更高.

需要说明的是,采用本文的方法,选择已有的测试数据,生成覆盖目标路径的测试数据,在替换部分个体之后,种群的适应值并没有都提高.这是因为,本文方法以路径相似度,作为进化个体的适应度,而程序中存在与目标路径相关性较高,但路径相似度较低的路径.使用本文的方法,有助于选择穿越这些路径的测试数据.

表 4　基于路径相似度标准和路径差异度标准已有测试数据的利用情况

测试对象	路径相似度标准			路径差异度标准		
	利用数	适应值增长率/%	命中率/%	利用数	适应值增长率/%	命中率/%
三角形分类	5.5	100	85.6	7.20	92.3	94.5
冒泡排序	3.7	100	90.5	12.40	93.1	96.8
huffcode	20.0	100	100.0	20.00	97.2	100.0
tcas	9.8	100	92.6	17.00	92.7	94.9
replace	4.6	100	92.8	16.20	91.5	93.5
flex	7.1	100	71.3	15.37	91.7	89.4

(2)对问题 2 的回答

由表 5 可以得出,与基于未排序的目标路径生成测试数据相比,基于排序后的目标路径生成测试数据,能够减少时间,并提高路径的覆盖率.

①与采用未排序的 NGA 相比,PRGA 的测试数据生成时间明显少于 NGA,平均减少了 18.2% 的时间开销。这是因为,排序后的当前目标路径,与下一路径具有一定的相关性,使得生成覆盖当前目标路径的测试数据之后,当前种群的个体对下一目标也具有较好的适应值.

在实验中,我们也发现,NGA 在生成测试数据时,如果满足当前目标路径约束条件的输入域在整个输入域中占的比值较小,那么,种群将容易丢失多样性,使得生成覆盖后继目标路径的测试数据耗时较长,甚至难以生成覆盖后继目标路径的测试数据.在上述情况下,PRGA 的目标路径覆盖率略高于 NGA.

②由表 5 可以得出,与未排序的 EGA 相比,本文方法生成测试数据需要的时间明显减少,且目标路径覆盖率平均提高了 5.4%,最大提高了 10.2%.这是因为,目标路径排序之后,需要先覆盖的目标路径,与已有测试数据覆盖的路径具有较高的相关度,这有助于利用已有的测试数据.

表 5 不同方法需要的时间和目标路径覆盖率

测试对象	NGA		PRGA		EGA		PSEGA	
	时间/s	覆盖率/%	时间/s	覆盖率/%	时间/s	覆盖率/%	时间/s	覆盖率/%
三角形分类	15.5	59.5	13.60	63.2	5.74	84.2	**3.00**	**93.7**
冒泡排序	328.6	79.0	246.40	80.1	96.70	91.7	**59.40**	**95.6**
huffcode	13.3	96.2	**10.50**	96.8	18.30	**100.0**	16.60	**100.0**
tcas	13.7	87.6	**12.10**	89.7	14.60	93.3	13.80	**93.6**
replace	628.2	71.3	476.90	72.5	419.60	90.6	**312.10**	**92.2**
flex	12815.2	37.3	10917.80	37.7	5619.30	70.9	**3771.00**	**87.8**

(3)对问题 3 的回答

由表 5 容易看出,本文方法能够在较短的时间内,生成具有较高目标路径覆盖率的测试数据.

①与其他方法相比,对于三角形分类、冒泡排序、replace 以及 flex,本文方法均需要最短的时间,并取得最高的目标路径覆盖率.与传统遗传算法相比,本文方法的目标路径覆盖率平均提高了 22%,消耗的时间平均降低了 43%.

②对于程序 huffcode 和 tcas,与耗时最少的 PRGA 相比,本文方法的耗时最高多 5.1 s.这是因为,被测程序的控制结构简单,路径数多,而本文方法和 EGA 均需要对已有测试数据评价.但是,与其他方法相比,本文的方法取得了最高的目标路径覆盖率.在注重软件可靠性的回归测试中,本文方法显然更具有应用价值.这说明,在测试结构简单且路径多的程序时,需要减少已有测试数据的使用率,并优化参与进化的已有测试数据.

③在较大规模的 flex 程序中,本文方法明显优于其他 3 种方法,与 NGA 相比,本文方法减少了 70.6%的时间开销,覆盖率提高 50.5%.这说明,本文方法对于较大规模的程序是适用且高效的.

(4)对问题 4 的回答

由图 4 可以看出,与 EGA 相比,针对被测程序的不同修改版本,本文方法需要较少的已有测试数据,即可达到较高的目标路径覆盖率.对于各被测程序,已有测试数据覆盖目标路径的比例为 50%时,本文方法能够达到 95%的目标路径覆盖率;对于 EGA 而言,上述比例为 70%时,才达到 95%的目标路径覆盖率.这说明,本文方法适用于不同程序的不同修改版本.

当上述比例接近 100%时,这两种方法都覆盖了所有的目标路径.可以看出,对于本文建立的用于回归测试的数据生成问题模型,采用新生成的测试数据,替换当前种群的部分个体,是完全正确的.

5.4 参数敏感性分析

为了进一步考察不同的差异度阈值、程序不同修改版本包含不同的已有测试数据比例,对本文方法的影响,选择三角形分类程序进行实验.

首先,针对图 2 修改的程序,按照第 4.1 节设定的标准,列出不同差异度阈值下,生成测试数据需要的进化代数和目标路径覆盖率,如表 6 所列.其中,第 1 行数据表示遗传算法的进化种群不使用任何已有的测试数据;差异度阈值为 1,表示遗传算法的进化种群使用所有已有的测试数据;其余差异度阈值下的实验结果表示,使用符合第 4.1 节标准的集合 T 的部分数据.由表 6 可以看出:

(1)不使用已有的测试数据,遗传算法的运行代数多,且目标路径的覆盖率低;使用已有的测试数据,目标路径的覆盖率显著增加.这说明,利用已有的测试数据,本文方法能够提高测试数据生成

的效率.

(2)当差异度阈值由 0 变化到 1/6 和 2/6 时,使用已有测试数据替代进化种群的个体数增加,从而降低了进化代数,提高了目标路径覆盖率;当差异度阈值由 2/6 变化至 3/6 和 4/6 时,已有测试数据替代进化种群的个体数没有明显增加,导致进化代数和目标路径覆盖率的差别较小;当差异度阈值由 5/6 变化至 1 时,用来替代进化种群的新个体穿越的路径与目标路径没有相关性,因此,进化代数增多,目标路径覆盖率低.

表 6　不同差异度阈值下,生成测试数据的种群进化代数和目标路径覆盖率

差异度阈值	进化代数		覆盖率/%
	均值	标准差	
—	32592.9	3398.2	60.8
0	23771.9	2217.7	75.0
1/6	18274.9	2443.3	80.8
2/6	15931.5	3432.1	83.4
3/6	16524.3	4891.0	82.6
4/6	18023.1	2990.7	81.1
5/6	7531.3	5955.1	92.1
1	13546.2	5629.6	85.8

这说明,如果集合 T 中可利用的已有测试数据增多,且该测试数据穿越的路径与目标路径有一定的相关性,那么,将能够提高进化种群的性能.

(3)当差异度阈值为 5/6 或 1 时,进化代数的标准差较大,超过了最大进化代数.这说明,当差异度阈值较大时,本文方法的随机性也较大.这就要求,当被测程序包含的目标路径和已有测试数据均较多时,设定的差异度阈值,应保证优先选择穿越与目标路径具有较大相关性路径的测试数据.可以看出,第 4.3 节设定的差异度阈值是合适的,且对于三角形分类程序也是最优的.

在图 2 修改后程序的基础上,进一步对程序修改,集合 T 覆盖目标路径的比例如表 7 所列.由表 7 可以看出,当集合 T 覆盖目标路径的比例较高(79.0%)时,在表 6 目标路径覆盖率较高的 2 种差异度阈值下,所提方法进化 20 代左右,就能够生成覆盖所有目标路径的测试数据.这说明,本文方法在已有测试数据比例较高时是高效的.另外,如果存在可利用的已有测试数据(21.1%～79.0%),本文方法在两种差异度阈值下,均能够获得较高的目标路径覆盖率,其中,差异度阈值为 5/6 时,本文方法的覆盖率最小值为 92.1%,进化代数也较少.这说明,如果存在已有的测试数据,本文方法生成测试数据的效率将很高.

表 7　不同的已有测试数据比例下,生成测试数据需要的进化代数和目标路径覆盖率

已有测试数据比例/%	差异度阈值(2/6)		差异度阈值(5/6)	
	进化代数	覆盖率/%	进化代数	覆盖率/%
0	32801.0	60.5	22775.4	71.1
21.1	15931.5	83.4	7531.3	92.1
42.1	6110.0	93.7	1800.7	98.4
57.9	6271.3	93.4	1106.8	99.0
79.0	20.3	100.0	20.0	100.0

6　总　结

本文研究生成回归测试数据,以覆盖程序维护影响路径时,测试数据进化生成问题的数学模型及其求解方法.为此,首先,通过路径的差异度和相似度,衡量不同路径的相关性;然后,基于修改节点和目标路径集合的相关矩阵,得到目标路径覆盖优先级,并基于覆盖优先级排序后的目标路径,建立覆盖被测程序目标路径的测试数据生成问题的数学模型;最后,采用遗传算法求解上述模型时,以路径相似度作为评价个体性能的适应度函数,并利用穿越与目标路径具有相关性的路径的测试数据,替换进化种群的部分个体,以提高种群的性能,从而提高测试数据生成的效率.

为了验证所提方法的有效性,对多个基准和工业程序测试,并与传统回归测试数据生成方法、利用已有测试数据的测试数据生成方法等比较,最后的实验结果表明,本文方法利用已有的测试数据生成期望的测试数据时,耗时少,且目标路径覆盖率高.

我们发现,不同的被测程序,包含的路径数目、输入个数以及输入类型往往有差异.如果对于不同的被测程序,采用相同的遗传算法参数设置,那么,利用遗传算法生成测试数据的效率将有很大差别.根据被测程序的特征,选择合适的遗传算法控制参数,是需要进一步研究的问题.此外,利用其他的方法选择已有的测试数据,以替换当前种群的部分个体,也将会提高测试数据生成的效率,这也是需要进一步研究的问题.

致　谢　各位审稿专家对本文提出了宝贵评审意见,这些评审意见对提高论文水平具有很大的帮助;编辑也付出了辛勤工作.在此一并致谢!

参　考　文　献

[1]　Rothermel G, Harrold M J. Analyzing regression test selection

techniques. IEEE Transactions on Software Engineering, 1996, 22(8): 529-551

[2] Amitabh S, Jay T. Effectively prioritizing tests in development environment//Proceedings of the 2002 ACM SIGSOFT International Symposium on Software Testing and Analysis. New York, USA, 2002: 97-106

[3] Muccini H, Dias M, Richardson D. Towards software architecture-based regression testing//Proceedings of the 2005 Workshop on Architecting Dependable Systems. New York, USA, 2005: 1-7

[4] Srikanth H, Williams L. On the economics of requirements-based test case prioritization//Proceedings of the 7th International Workshop on Economics-Driven Software Engineering Research. New York, USA, 2005: 1-3

[5] Walcott K R, Soffa M L, Kapfhammer G M, Roos R S. Time aware test suite prioritization//Proceedings of the International Symposium on Software Testing and Analysis. Portland, USA, 2006: 1-12

[6] Xanthakis S, Ellis C, Skourlas C, et al. Application of genetic algorithms to software testing//Proceedings of the 5th International Conference on Software Engineering and Its Applications. Los Alamitos, USA, 1992: 625-636

[7] Watkins A. The automatic generation of test data using genetic algorithms//Proceedings of the 4th Software Quality Conference. Austin Texas, USA, 1995: 300-309

[8] Lakhotia K, Harman M, McMinn P. A multi-objective approach to search-based test data generation//Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation. New York, USA, 2007: 1098-1105

[9] Ahmed M A, Hermadi I. GA-based multiple paths test data generator. Computer & Operations Research, 2008, 35(10): 3107-3127

[10] Zhang Wan-Qiu, Gong Dun-Wei. Evolutionary generation of test data for many paths coverage based on grouping. Journal of Systems and Software, 2011, 84(12): 2222-2233

[11] Tallam S, Gupta N. A concept analysis inspired greedy algorithm for test suite minimization//Proceedings of the 6th ACM SIGPLAN-SIGSOFT Workshop on Program Analysis for Software Tools and Engineering. New York, USA, 2005: 35-42

[12] Zhang Xiao-Fang, Xu Bao-Wen, Nie Chang-Hai, et al. An approach for optimizing test suite based on testing requirement reduction. Journal of Software, 2007, 18(4): 821-831 (in Chinese)
(章晓芳, 徐宝文, 聂长海等. 一种基于测试需求约简的测试用例集优化方法. 软件学报, 2007, 18(4): 821-831)

[13] Sun Chang-Ai. A constraint-based test suite reduction method for conservative regression testing. Journal of Software, 2011, 6(2): 314-321

[14] Zhang Zhi-Hao, Huang Jun, Zhang Bo, et al. Regression test generation approach based on tree-structured analysis//Proceedings of the International Conference on Computational Science and Its Applications. Fukuoka, Japan, 2010: 244-249

[15] Nachiyappan S, Vimaladevi A, SelvaLakshmi C B. An evolutionary algorithm for regression test suite reduction//Proceedings of the International Conference on Communication and Computational Intelligence. Erode, India, 2010: 503-508

[16] Ye Nan, Chen Xin, Jiang Peng, et al. Automatic regression test selection based on activity diagrams//Proceedings of the 5th International Conference on Secure Software Integration & Reliability Improvement Companion. Jeju Island, South Korea, 2011: 166-171

[17] Kumar A, Tiwari S, Mishra K K, et al. Generation of efficient test data using path selection strategy with elitist GA in regression testing//Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology. Chengdu, China, 2010: 389-393

[18] Sampath S, Bryce R, Memon A. A uniform representation of hybrid criteria for regression testing. IEEE Transactions on Software Engineering, 2013, 39(10): 1326-1344

[19] Taneja K, Xie T, Tillmann N, et al. Express: Guided path exploration for efficient regression test generation//Proceedings of the 2011 International Symposium on Software Testing and Analysis. New York, USA, 2011: 1-11

[20] Shin Y, Harman M. Test data regeneration: Generating new test data from existing test data. Software Testing, Verification and Reliability, 2012, 22(3): 171-201

[21] Santelices R, Chittimalli P K. Test-suite augmentation for evolving software//Proceedings of the 23rd IEEE ACM International Conference on Automated Software Engineering. L'Aquila, Italy, 2008: 218-227

[22] Xu Zhi-Hong, Kim Yunho. Directed test suite augmentation: Techniques and tradeoffs//Proceedings of the 18th ACM SIGSOFT International Symposium on Foundations of Software Engineering. New York, USA, 2010: 257-266

[23] Xu Zhi-Hong, Cohen M, Rothermel G. Factors affecting the use of genetic algorithms in test suite augmentation//Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. New York, USA, 2010: 1365-1372

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

[25] Maragathavalli P, Kanmani S, Kirubakar J S, et al. Automatic program instrumentation in generation of test data using genetic algorithm for multiple paths coverage//Proceedings of the Advances in Engineering, Science and Management. Nagapattinam, India, 2012: 349-353

[26] Bentley J L, Sedgewick R. Fast algorithms for sorting and searching strings//Proceedings of the 8th Annual ACM-SIAM Symposium on Discrete Algorithms. Philadelphia, USA, 1997: 360-369

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

[28] Do H, Elbaum S, Rothermel G. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Software Engineering, 2005, 10(4): 405-435

WU Chuan, born in 1980, Ph. D. candidate, lecturer. His main research interest is search-based software engineering.

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

Background

In the life cycle, we need regressively to test the affected part of software rapidly and completely after software updating. Since there has already been a test data set, the key to regression testing lies in making full use of existing test information to improve the efficiency of test data generation.

Up to now, research on the utilization of existing test information focuses mainly on test data selection and augmentation. Among them, test data selection considers how to determine the priority of existing test data, and chooses test data based on it to execute the program under test. However, test data selection is unfit for the modified program when these test data cannot cover the updated elements. Hence, test data selection is difficult in seeking for existing bugs. To overcome the above drawbacks, test data augmentation employs evolutionary optimization to generate new test data for new test targets, such as statements, branches and paths. However, previous test data augmentation methods have not made full use of existing information during the evolution.

In this paper, we focus on the problem of generating test data for regression testing, propose an evolutionary generation approach of test data based on path correlation for regression testing. In this method, firstly, the target paths are sorted according to the correlation matrix between paths and nodes, and the mathematical model is built for generating regression test data that cover affected paths based on the matrix. Then, when solving the above model using genetic algorithms, test data that have traversed the existing target paths are employed to replace a part of individuals of the current population. Finally, we evaluate the proposed approach by performing experimental studies on six benchmarks and industrial programs and comparing it with a commonly used genetic algorithm, and make a conclusive evaluation that the proposed approach can effectively improve the efficiency of test data generation.

This research is jointly sponsored by the National Natural Science Foundation of China (61375067), the National Basic Research Program (973 Program) of China (2014CB046306-2), the Specialized Research Fund for Doctoral Program of Higher Education of China (Doctoral Tutor Class) (20100095110006), and the Natural Science Foundation of Jiangsu Province (BK2012566).