软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2016,27(4):828−838 [doi: 10.13328/j.cnki.jos.004973]

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

融入神经网络的路径覆盖测试数据进化生成*

姚香娟[1,3], 巩敦卫[2], 李 彬[3]

[1](软件工程国家重点实验室(武汉大学),湖北 武汉 430072)

[2](中国矿业大学 信息与电气工程学院,江苏 徐州 221116)

[3](中国矿业大学 理学院,江苏 徐州 221116)

通讯作者: 姚香娟, E-mail: yxjcumt@126.com, http://cos.cumt.edu.cn/List.aspx?id=3883

摘 要: 利用遗传算法生成复杂软件的测试数据,是软件测试领域一个全新的研究方向.传统的基于遗传算法的测试数据生成技术,需要以每个测试数据作为输入运行被测程序,以获得个体的适应值,因此,需要消耗大量的运行时间.为了降低运行程序带来的时间消耗,提出一种基于神经网络的路径覆盖测试数据进化生成方法,主要思想是:首先,利用一定样本训练神经网络,以模拟个体的适应值;在利用遗传算法生成测试数据时,先利用训练好的神经网络粗略计算个体适应值;对适应值较好的优秀个体,再通过运行程序,获得精确的适应值.最后的实验结果表明,该方法可以有效降低运行程序产生的时间消耗,从而提高测试数据生成的效率.

关键词: 软件测试;测试数据生成;进化优化;神经网络;路径覆盖

中图法分类号: TP311

中文引用格式: 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

英文引用格式: 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). http://www.jos.org.cn/1000-9825/4973.htm

Evolutional Test Data Generation for Path Coverage by Integrating Neural Network

YAO Xiang-Juan[1,3], GONG Dun-Wei[2], LI Bin[3]

[1](State Key Laboratory of Software Engineering (Wuhan University), Wuhan 430072, China)

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

[3](College of Science, China University of Mining and Technology, Xuzhou 221116, China)

Abstract: It is a novel research direction in the field of software testing to generate test data using genetic algorithms for complex software. Traditional techniques of test data generation based on genetic algorithms need to run a program using each test datum as an input, so as to obtain its fitness value and as a result, they consume a large amount of executing time. In order to reduce the time consumption of running a program, this paper proposes a method of test data generation for path coverage based on neural networks. First, a neural network is trained using a certain amount of samples to simulate an individual's fitness value. Then, when generating test data by the genetic algorithm, an individual's fitness value is roughly estimated using the trained neural network. Finally, for individuals with good estimated fitness values, their precise fitness value are calculated by running the program. The experimental results show that this method can effectively reduce the time consumption of running a program, therefore improve the efficiency of test data generation.

Key words: software testing; test data generation; evolutionary optimization; neural network; path coverage

* 基金项目: 国家自然科学基金(61375067, 61573362, 61203304); 软件工程国家重点实验室开放基金(SKLSE20100819); 江苏省自然科学基金(BK2012566)

Foundation item: National Natural Science Foundation of China (61375067, 61573362, 61203304); Fund of State Key Laboratory of Software Engineering (SKLSE20100819); Natural Science Foundation of Jiangsu Province (BK2012566)

收稿时间: 2015-09-03; 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 13:15:56, http://www.cnki.net/kcms/detail/11.2560.TP.20160114.1315.002.html

软件产业是我国大力发展的战略性新兴产业之一,对促进国民经济和社会发展具有非常重要的作用.因此,软件质量越来越受到人们的重视与关注.提高软件质量的重要途径之一,是在软件投入使用之前进行大量测试,以发现软件中存在的缺陷甚至错误.而进行软件测试的首要任务,是采用有针对性的理论和方法生成有效的测试数据,以满足既定的测试充分性准则.

按照测试过程所需要的信息,软件测试分为黑盒测试、白盒测试以及二者融合的测试.其中,白盒测试需要已知被测软件的代码,更容易发现软件的缺陷或错误[1].在白盒测试的诸多充分性准则中,路径覆盖是最重要也是研究最多的一种.单锦辉等人[2]认为,许多软件测试问题都可以归结为路径覆盖测试数据生成问题.该问题描述为:给定程序的一条目标路径,在程序的输入空间寻找测试数据,使得以该测试数据为输入所穿越的路径为目标路径.

生成满足要求的测试数据往往需要耗费大量的时间,自动求解上述问题,将有效地减轻测试人员的劳动强度、提高软件测试的效率和软件质量.其中,利用遗传算法生成复杂软件的测试数据,是近年来兴起的一个全新的研究方向.

作为一种受自然界生物进化和遗传变异机制启发产生的全局概率搜索方法,遗传算法近年来在软件测试中的应用得到了国内外学者的广泛关注,并取得了丰硕的研究成果[3].但是,在利用遗传算法生成测试数据时,需要使用每个测试数据运行插装后的程序,以评价测试数据的性能.这样,运行程序的时间花费是非常巨大的.

鉴于此,本文提出一种利用神经网络模拟个体评价函数的方法,从而有效地降低了运行程序的代价.首先,利用一定样本训练神经网络,以模拟个体的适应值;在利用遗传算法生成测试数据时,先利用训练好的神经网络粗略计算个体适应值;对适应值较好的优秀个体,再通过运行程序获得精确的适应值.实验结果表明:该方法可以有效地降低运行程序产生的时间消耗,从而提高测试数据生成的效率.

本文第 1 节介绍相关工作.第 2 节是需要用到的一些预备知识.本文提出的基于神经网路的路径覆盖测试数据进化生成方法在第 3 节给出.第 4 节是实验结果及分析.最后,第 5 节给出结论.

1 相关工作

1.1 基于遗传算法的测试数据生成

借助搜索算法生成复杂软件的测试数据,是近年来软件测试领域一个非常活跃的研究方向,其基本思想是:通过定义合适的适应值函数,将测试数据生成问题转化为函数优化问题;然后,利用搜索算法对建立的优化问题进行求解.典型的算法包括进化算法、爬山法、模拟退火和禁忌搜索等.其中,应用最为广泛的是遗传算法.

1992 年,Xanthakis 等人[4]首次提出采用遗传算法生成路径覆盖测试数据的思想,从而开创了一个全新的研究方向.1995 年以后,遗传算法开始频繁地应用于软件测试数据自动生成,并涌现出大量的研究成果.

1996 年,Sthamer[5]完成了第一篇采用遗传算法生成测试数据的博士学位论文,主要采用遗传算法解决分支覆盖测试、循环测试等问题.

Wegener 等人[6]建立了基于遗传算法的测试数据生成环境,并应用于许多工业程序的测试.与随机法的比较结果表明,基于遗传算法生成的测试数据具有更高的覆盖率.

到目前为止,遗传算法不但用于解决传统的分支覆盖测试[7]、条件覆盖测试[8]、路径覆盖测试[9]等,还用于解决变异测试问题[10]、蜕变测试问题[11]以及标记变量问题[12]等;应用的范围有面向对象的软件测试[13]、嵌入式软件的测试[14]、并行软件的测试[15]等;采用的方法除了单纯的遗传算法以外,还有遗传算法与其他智能算法,如模拟退火算法[16]及局部搜索算法[17]等等的混合算法.

这些成果极大地丰富了基于遗传算法的软件测试数据生成理论,有效提高了软件测试数据生成的效率,从而有力地促进了软件质量的提高.但是,利用遗传算法解决测试数据生成问题时,需要以每个测试数据作为输入运行程序,以获得个体的适应值.这样,使得运行程序的代价非常巨大.

1.2 神经网络在软件测试的应用

Aggarwal等人[18]研究了如何使用神经网络解决Oracle问题.首先,根据被测程序的规格说明书,获得被测程序的输入输出对,并作为神经网络的训练样本;然后,使用训练样本对多层前馈式神经网络进行训练,得到的神经网络模型即可作为应用模型;对给定输入数据,通过神经网络预测其期望输出.

Anderson 等人[19]研究了如何使用神经网络进行错误预测,主要工作包括训练和预测阶段.在神经网络训练阶段,使用训练样本对神经网络进行训练.在他们建立的神经网络模型中,输入是测试数据,输出是错误类型.在预测阶段,对给定的输入数据,通过神经网络预测其可能发现的错误.

目前,神经网络在软件测试中的应用主要体现在两个方面:一是将神经网络作为分类器,预测测试数据的揭错能力,据此选用精简测试数据;二是神经网络用作系统逼近器,代替真实的被测软件实施测试.但是,将神经网络用于测试数据生成的还很少.

2 预备知识

2.1 BP神经网络基本原理

BP 神经网络是目前应用最为广泛的神经网络之一[20],它是在 1986 年由 Rumelhart 和 McClelland 提出的,是一种包含多层网络的"逆推"学习算法[21].BP 神经网络的学习过程由信号的正向传播与误差的反向传播两个过程组成:正向传播时,输入样本从输入层传入,经隐含层逐层处理后传向输出层,若输出层的实际输出与期望输出不符,则转向误差的反向传播阶段;误差的反向传播是将输出误差以某种形式通过隐含层向输入层逐层反传,并将误差分摊给各层的所有单元,从而获得各层单元的误差信号,该误差信号即作为修正各单元权值的依据.这种信号正向传播与误差反向传播的各层权值调整过程周而复始地进行,权值不断调整的过程,也就是网络的学习训练过程.该过程一直进行,直到网络输出的误差减少到可以接受的程度,或进行到预先设定的学习次数为止.

2.2 路径覆盖测试数据生成问题的优化模型

要使用遗传算法解决路径覆盖问题,需要把该问题建模为一个函数优化问题,其中,适应值函数的建立是关键.常用的适应值函数通常包括层接近度和分支距离[22].层接近度用于衡量测试数据穿越的路径与目标路径的偏离程度,分支距离是指使一个谓词为真(或假)的条件满足程度.

设 P 是一条目标路径,$\boldsymbol{X}=(x_1,x_2,\ldots,x_n)^T$ 为某个测试数据,$\boldsymbol{X}$ 穿越的路径为 $P(\boldsymbol{X})$.把 $\boldsymbol{X}$ 穿越路径 $P(\boldsymbol{X})$偏离目标路径 P 的分支到 P 的终点之间的分支语句个数(不包含分岔点本身)称为 $\boldsymbol{X}$ 对目标路径 P 的层接近度,记为 $Approach\_level(\boldsymbol{X})$.

分支距离函数的概念最初由 Korel 提出[21],用于描述输入数据接近各条件语句分支条件的程度.例如,设路径中某条件语句为"if $a\geqslant 20$",我们的目标是让该语句的真分支得到执行.设当程序以 $\boldsymbol{X}$ 为输入数据执行到该语句时,a 的值为 $a(\boldsymbol{X})$,那么 $\boldsymbol{X}$ 对应该语句真分支的分支距离函数为

$$Branch\_dist(\boldsymbol{X})=\begin{cases}0, & a(\boldsymbol{X})\geqslant 12\\ 12-a(\boldsymbol{X}), & \text{otherwise}\end{cases} \tag{1}$$

对应不同分支条件的分支距离函数见表 1.

Table 1 Branch distances for different conditions

表 1 不同分支条件对应的分支距离

分支条件		$a\geqslant b$	$a>b$	$a=b$	$a\neq b$
分支距离	取真	0	0	0	0
	取假	$\lvert a-b\rvert$	$\lvert a-b\rvert+0.1$	$\lvert a-b\rvert$	1

一般情况下,个体 $\boldsymbol{X}$ 的适应值记为 $fitness(\boldsymbol{X})$,为层接近度和标准化的分支距离之和,即:

$$fitness(\boldsymbol{X})=Approach\_level(\boldsymbol{X})+Narmal(Branch\_dist(\boldsymbol{X})) \tag{2}$$

其中,

$$Narmal(Branch\_dist)=1-1.001^{-Branch\_dist} \tag{3}$$

$fitness(\boldsymbol{X})=0$ 的充要条件是 $\boldsymbol{X}$ 正好覆盖测试目标.$fitness(\boldsymbol{X})$的值越小,$\boldsymbol{X}$越接近于测试目标.因此,覆盖测试目标的测试数据生成问题,就可以转化为函数 $fitness(\boldsymbol{X})$的最小化问题.

3 基于 BP 神经网络的测试数据进化生成

3.1 神经网络的构建

本文主要利用 BP 神经网络模拟公式(2)给出的个体适应值函数,该函数包含 n 个输入 $\boldsymbol{X}=(x_1,x_2,\ldots,x_n)^T$、1 个输出 $y=fitness(\boldsymbol{X})$.因此,我们构建的 BP 神经网络包含 n 个输入变量、1 个输出变量.假设隐含层的神经元个数为 l,则本文采用的 BP 神经网络结构如图 1 所示,其中,

- 输入向量为 $\boldsymbol{X}=(x_1,x_2,\ldots,x_n)^T$,对应被测程序的输入向量;
- 隐含层输出向量为 $\boldsymbol{O}=(o_1,o_2,\ldots,o_l)^T$;
- 输出层的输出变量为 y,对应输入的适应值函数;
- 输入层到隐含层之间的权值用矩阵 $\boldsymbol{V}=[v_{ij}]$表示,其中,v_{ij}为输入层第 i 个神经元到隐含层第 j 个神经元的权重.隐含层到输出层的权值用向量 $\boldsymbol{W}=(w_1,\ldots,w_j,\ldots,w_l)^T$ 表示,其中,w_j 为隐含层第 j 个神经元到输出层的权重.

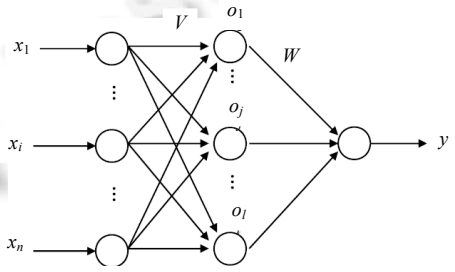


Fig.1 Structure of BP neural network

图 1 BP 神经网络的结构

隐含层节点的数目直接关系到 BP 神经网络的学习效率和泛化能力:隐含层节点数目太少,则不能把问题描述清楚,甚至无法完成 BP 神经网络的生成;隐含层节点数目太多,BP 神经网络虽能准确建立样本的函数,但是对于新的测试数据的预测效果可能会很差.所以隐含层节点数目的选择也非常重要,本文确定隐含层节点数目的方法为

$$l=\sqrt{n+m}+r \tag{4}$$

其中,n 表示输入层节点数,m 表示输出层节点数,r 为一个 1~10 的随机整数.

3.2 样本设计

我们需要使用一定数量的样本对构建的神经网络进行训练.这里,一个样本就是一个包含输入和预期输出的二元组$(\boldsymbol{X},d)$,其中,输入 $\boldsymbol{X}=(x_1,x_2,\ldots,x_n)^T$ 为被测程序的输入向量,预期输出 d 为其适应值.

首先生成一定数量的测试数据,设为 $\boldsymbol{X}^1,\boldsymbol{X}^2,\ldots,\boldsymbol{X}^\alpha$,$\alpha$为测试数据的个数;然后,以这些测试数据作为输入运行插装后的程序,得到相应的适应值 $d^1,d^2,\ldots,d^\alpha$(期望输出).这样,就可以得到一个容量为α的样本$\{(\boldsymbol{X}^1,d^1),(\boldsymbol{X}^2,d^2),\ldots,(\boldsymbol{X}^\alpha,d^\alpha)\}$.需要说明的是:为了使样本更具代表性,我们在选取测试数据时,会尽量使其均匀分布在被测程序的输入域.

另外,训练样本数越多,训练结果就越能正确反映其内在非线性规律;但样本越多,消耗的计算资源也越多,并且当样本数量达到一定程度时,BP 神经网络的精度也很难大幅度提高,而是收敛于一个固定值.因此,我们使

用的样本数设定在总的网络连接权重的 10 倍左右.

因为输入变量的取值范围有大有小,导致不同输入对神经网络输出的影响各不相同.因此,需要对测试数据作归一化处理,使每个变量都具有同等重要的地位.对测试数据 $\boldsymbol{X}=(x_1,x_2,\ldots,x_n)^T$,设 x_j 的输入域为 $[x_j^{\min},x_j^{\max}]$,则归一化变换公式如下:

$$x_j=\frac{x_j-x_j^{\min}}{x_j^{\max}-x_j^{\min}} \tag{5}$$

3.3 各层神经元的输出

对图 1 所示的神经网络,隐含层的输入为

$$net_j=\sum_{i=1}^{n}v_{ij}x_i \tag{6}$$

隐含层的输出为

$$o_j=f(net_j)=f\left(\sum_{i=1}^{n}v_{ij}x_i\right) \tag{7}$$

其中,x_i 为输入层第 i 个神经元的输入,v_{ij} 为输入层第 i 个神经元到隐含层第 j 个神经元的权重,$f(\cdot)$为处理神经元的激活函数.输出层的输入为

$$r=\sum_{j=1}^{l}w_jo_j \tag{8}$$

输出层的输出为

$$y=f(r)=f\left(\sum_{j=1}^{l}w_jo_j\right) \tag{9}$$

其中,w_j 为隐含层第 j 个神经元到输出层的权重.本文采用 S 形函数作为激活函数,即: $f(u)=\dfrac{1}{1+\mathrm{e}^{-u}}$.

3.4 各层权值的修正

对某个样本$(\boldsymbol{X},d)$,其中,$\boldsymbol{X}=(x_1,x_2,\ldots,x_n)^T$ 为输入,d 为期望输出.当网络实际输出 y 与期望输出 d 不相等时,存在输出误差 E,其中,

$$E=\frac{1}{2}(y-d)^2=\frac{1}{2}\left[f\left(\sum_{j=1}^{l}w_jo_j\right)-d\right]^2=\frac{1}{2}\left\{f\left[\sum_{j=1}^{l}w_jf\left(\sum_{i=1}^{n}v_{ij}x_i\right)\right]-d\right\}^2 \tag{10}$$

对于样本$\{(\boldsymbol{X}^1,d^1),(\boldsymbol{X}^2,d^2),\ldots,(\boldsymbol{X}^\alpha,d^\alpha)\}$,其综合误差为

$$E_T=\sum_{k=1}^{\alpha}E^k=\frac{1}{2}\sum_{k=1}^{\alpha}(y^k-d^k)^2=\frac{1}{2}\sum_{k=1}^{\alpha}\left\{f\left[\sum_{j=1}^{l}w_jf\left(\sum_{i=1}^{n}v_{ij}x_i^k\right)\right]-d^k\right\}^2 \tag{11}$$

其中,

$$\begin{cases}y^k=f(r^k)=f\left(\sum_{j=1}^{l}w_jo_j^k\right)\\ o_j^k=f(net_j^k)=f\left(\sum_{i=1}^{n}v_{ij}x_i^k\right)\end{cases} \tag{12}$$

隐含层和输出层以及输入层和隐含层的权值修正依据如下梯度下降原则:

$$\Delta w_j(t+1)=-\eta_2\frac{\partial E_T(t)}{\partial w_j(t)} \tag{13}$$

$$\Delta v_{ij}(t+1)=-\eta_1\frac{\partial E_T(t)}{\partial v_{ij}(t)} \tag{14}$$

3.5 基于神经网络的路径覆盖测试数据进化生成

对路径覆盖测试数据生成问题,已有很多研究成果.本文主要研究使用神经网络模拟个体适应值,以减少运行程序的时间.因此在生成测试数据时,只采用比较基本的方法.被测程序可能包含很多目标路径,我们每次只针对一条目标路径运行算法,以生成覆盖该路径的测试数据.有多少条目标路径,就运行多少次算法.

3.5.1 算法描述

首先生成一定数量的测试数据,通过运行插装后的程序得到真实的适应值(期望输出),从而得到所需的样本;接着,利用样本训练神经网络;然后,使用遗传算法生成覆盖目标路径的测试数据.在遗传算法的进化过程中,先使用训练好的神经网络粗略计算个体适应值;对那些适应值较好的优秀个体,再通过运行原程序得到真实的适应值.另外,由于本文只是使用神经网络估计个体的适应值,所以对同一条目标路径,只在算法开始时训练神经网络,算法运行过程中不再更新神经网络.

图 2 给出了基于神经网络生成测试数据的算法框图.

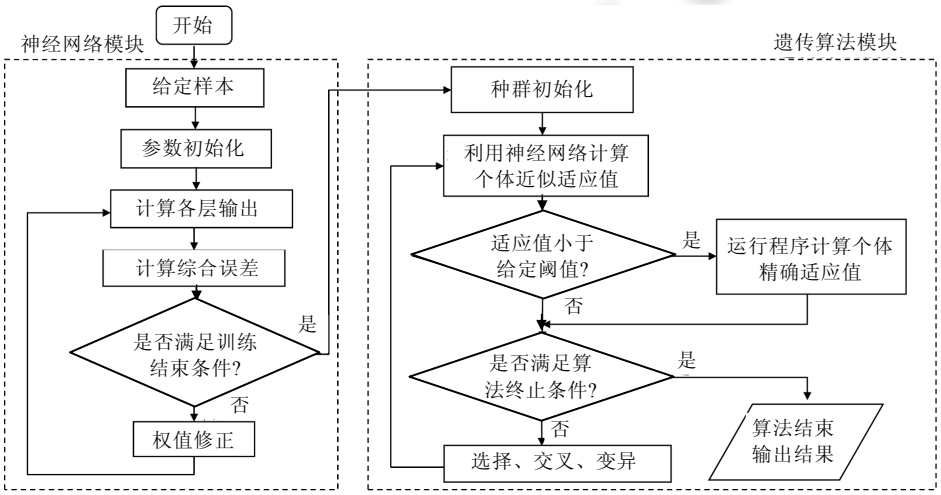


Fig.2 Algorithm diagram

图 2 算法框图

由图 2 可以看出,该算法主要包含两大模块:一个是神经网络模块,另一个是遗传算法模块.下面分别对这两个模块的详细步骤进行介绍.

3.5.2 神经网络模块

本文采用标准反向传播算法对神经网络进行训练,其主要步骤如下:

步骤 1. 获得一定数量的样本;

步骤 2. 采用随机法初始化权值等参数;

步骤 3. 对每个训练样本,依据公式(7)和公式(9)计算各层的输出;

步骤 4. 依据公式(11)计算综合误差;

步骤 5. 若综合误差小于给定阈值,或者训练次数达到规定上限,则终止训练过程;否则,转步骤 6;

步骤 6. 按照公式(13)和公式(14)对权值进行修正,转步骤 3.

3.5.3 遗传算法模块

本文采用遗传算法生成满足路径覆盖要求的测试数据,其主要步骤如下:

步骤 1. 种群初始化;

步骤 2. 对每个个体,利用训练好的神经网络计算其近似适应值;

步骤 3. 对优良个体,运行插装后的程序得到其精确适应值;

步骤 4. 如果找到最优个体,或者算法迭代次数达到规定的上限,终止算法的迭代过程;否则,转步骤 5;

步骤 5. 对个体进行选择、交叉和变异操作,转步骤 2.

4 实 验

为了验证本文方法的性能,我们通过大量实验进行分析.首先提出研究的问题;接着对采用的被测程序进行描述;然后给出实验设计方法;最后是实验结果和分析.

4.1 研究的问题

本文提出了一种利用神经网络模拟个体适应值的测试数据进化生成方法.神经网络能不能很好地模拟个体的适应值,是决定该方法是否有效的关键.因此,我们首先要研究的问题是:

RQ1:神经网络模拟个体适应值的精度如何?

另外,利用神经网络模拟个体适应值,是想减少运行程序所要花费的时间,从而提高测试数据生成的效率.因此,我们想要研究的第 2 个问题是:

RQ2:利用神经网络计算个体适应值是否可以节省时间?

综合以上两个问题,我们最后要研究的问题是:

RQ3:利用神经网络生成测试数据的效率如何?

4.2 被测程序

我们共选用 8 个不同大小和难度的实际程序进行实验,表 2 列出了每个程序的名称、代码行数、包含的子函数个数和简单的功能描述.其中,被测程序依据代码行数进行排序.这些程序均被广泛用于各种软件的测试和分析实验,具有一定的代表性.

Table 2 Basic information of programs under test

表 2 被测程序基本信息

序号	被测程序	代码行	子函数个数	简单功能描述
1	Hashmap	455	12	信息管理
2	Replace	564	21	模式匹配
3	Space	9 564	136	数组语言解释器
4	Flex	10 459	162	Unix 应用程序
5	Cadp	11 068	480	协议工程工具箱
6	Prepro	14 328	530	输入数据处理工具
7	Go	28 547	2 982	一种抽象策略板游戏
8	Spice	149 050	7 254	与虚拟桌面设备交互

4.3 实验设计

针对第 1 个要研究的问题 RQ1,对每个被测程序,随机抽取若干测试数据;然后,分别使用插装后的程序和神经网络(分别称为插装法和神经网络法)得到这些数据的适应值;最后,通过比较两种适应值之间的差异,对神经网络的精度进行评价.

针对第 2 个要研究的问题 RQ2,对每个被测程序,随机抽取若干测试数据,对特定目标路径,分别使用插装法和神经网络法计算个体适应值,比较两种方法所用时间,以此比较各自的效率.

针对第 3 个要研究的问题 RQ3,对每个被测程序,选择部分路径作为目标路径,使用遗传算法生成覆盖所有目标路径的测试数据.在算法的进化过程中,分别使用本文方法和原始方法对个体进行评价.所谓原始方法是指只通过插装法计算个体适应值.最后,通过比较两种方法生成测试数据的质量和时间评价各自的优劣.对本文方法,生成测试数据的时间包括对神经网络进行训练的时间.另外,两种方法都需要对原程序进行插装,从而得到个体的精确适应值.因此,对程序进行插装所需要的花费是一样的,可以忽略.两种方法的设置完全相同:输入变量采用二进制编码,种群规模为 20;采用轮盘赌选择,单点交叉和单点变异,交叉和变异概率分别为 0.75 和 0.05;

算法终止条件是找到覆盖目标语句的测试数据,或已经达到最大进化代数,本实验设置为 10 000.

每个程序包含的目标路径数目见表 3.

Table 3 Number of target paths of programs under test

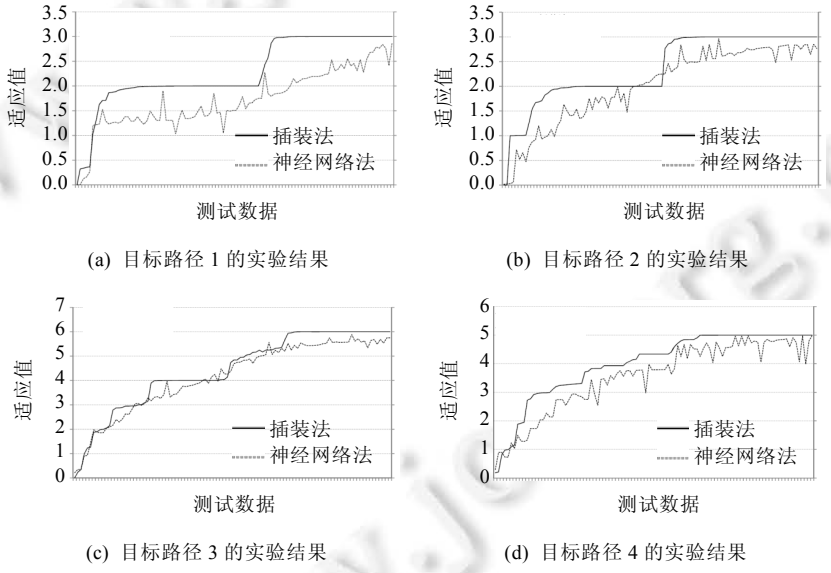
表 3 被测程序包含的目标语句数量

被测程序	目标路径数量
Hashmap	20
Replace	20
Space	30
Flex	30
Cadp	50
Prepro	50
Go	100
Spice	100

需要说明的是:有些路径很容易被覆盖,对这些路径,使用随机法很容易生成相应的测试数据,没有必要使用遗传算法.因此,我们在选择目标路径时尽量选择难覆盖的路径作为目标路径,以评价算法的性能.首先,使用随机法生成一定数量的测试数据;然后,以这些测试数据为输入运行程序,记录这些数据的路径覆盖情况;最后,选择未被覆盖的路径作为目标路径.另外,有些路径是不可达路径.不可达路径同样不会对算法的性能做出正确评价.目前,已有多种不可行路径检测方法[23],因此,我们可以保证所选择的都是可行路径.

4.4 实验结果及分析

针对第 1 个实验目的,我们随机抽取了被测程序的 4 条目标路径;针对每条目标路径,随机生成 100 个测试数据,然后,分别采用运行插装后的程序和神经网络的方法(分别称为插装法和神经网络法)计算这些测试数据对应的适应值,对比结果如图 3 所示(按照插装法得到适应值的大小排序).



(a) 目标路径 1 的实验结果 (b) 目标路径 2 的实验结果

(c) 目标路径 3 的实验结果 (d) 目标路径 4 的实验结果

Fig.3 Contrast experiment results for fitness value

图 3 适应值对比实验结果

从图 3 我们可以看出:由神经网络法得到的个体适应值和插装法稍微有所差别,但是基本能够反映个体适应值的高低,二者变化的整体趋势也是相同的.另外,对较优的个体,会使用插装法重新计算个体适应值,不会造成误差.因此,使用神经网络法模拟个体适应值的策略是完全可行的.

针对第 2 个实验目的,对每个程序,随机生成一定数目的测试数据,分别使用插装法和神经网络法计算每个

测试数据对每个目标路径的适应值,记录每种方法所用的时间.最后的实验结果见表 4.

Table 4 Contrast experiment results for efficiency of fitness calculation

表 4 适应值计算效率对比实验结果

程序	测试数据数目	评价次数	插装法所用时间(s)	神经网络法所用时间(s)
Hashmap	50 000	1 000 000	1.22	1.03
Replace	50 000	1 000 000	1.35	1.14
Space	100 000	3 000 000	34.68	5.02
Flex	100 000	3 000 000	35.26	5.34
Cadp	100 000	5 000 000	82.43	8.93
Prepro	100 000	5 000 000	68.74	8.67
Go	100 000	10 000 000	158.46	17.55
Spice	100 000	10 000 000	257.58	17.93

从表 4 可以看出:对所有被测程序,使用神经网络法所用的时间都少于插装法;特别是随着程序规模的不断扩大,神经网络法比插装法可以节省更多的时间开销.这就充分说明:利用神经网络法计算个体适应值,可以极大地减少运行程序的时间.

针对第 3 个实验目的,我们分别使用原始方法和本文方法生成覆盖所有目标路径的测试数据,记录生成测试数据所需的时间以及对目标路径的覆盖情况.所谓原始方法是指只采用插装法获得个体适应值.最后的实验结果见表 5,其中,评价次数是指算法进化过程中生成的个体总数,时间是指运行算法所花费的总时间,覆盖率是指已经覆盖的路径占目标路径总数的百分比,节约时间比是指本文方法节约的时间占原始方法所用时间的百分比.

Table 5 Contrast experiment results for test data generation

表 5 测试数据生成对比实验结果

程序	原始方法			本文方法			节约时间比(%)
	评价次数	时间(s)	覆盖率(%)	评价次数	时间(s)	覆盖率(%)	
Hashmap	109267×20	4.94	90.0	100939×20	4.35	95.0	11.94
Replace	157409×20	12.57	90.0	168781×20	10.04	90.0	20.13
Space	260187×20	116.7	83.3	257812×20	98.42	86.7	15.66
Flex	236916×20	180.81	88.0	222416×20	135.39	84.0	25.12
Cadp	329533×20	272.25	90.0	336933×20	204.68	92.0	24.82
Prepro	398425×20	293.42	97.0	382174×20	214.53	96.0	26.89
Go	579267×20	441.6	89.0	594543×20	287.53	85.0	34.89
Spice	635455×20	635.23	92.0	646354×20	448.71	93.0	29.36

从表 5 可以看出:对所有被测程序,本文方法需要的时间总是最少的.另外,随着程序规模以及目标路径个数的增加,本文方法所节约的时间越来越多.其中,对程序 Go,本文方法可以节约 34.89%的时间消耗.

从表 5 还可以看出:本文方法所需总的评价次数和达到的路径覆盖率与原始方法相比并没有明显差别,对 8 个被测程序,本文方法评价次数多于原始方法的有 4 个(Replace,Cadp,Go 和 Spice),少于原始方法的也有 4 个(Hashmap,Space,Flex 和 Prepro);本文方法的路径覆盖率低于原始方法的有 3 个(Flex,Prepro 和 Go),高于原始方法的有 4 个(Hashmap,Space,Cadp 和 Spice),等于原始方法的有 1 个(Replace).

为了更加科学地对两种方法进行对比,对评价次数、时间和覆盖率均采用 T 检验方法进行分析.其中,对评价次数和时间进行分析时,为了保证各个程序的结果都在同一数量级,对其进行了归一化处理.具体方法是:对每个被测程序,两种方法的结果均除以两者的最大值.评价次数对比的结果是,统计量 T_1=0.399;时间对比的结果是,统计量 T_2=8.988;覆盖率对比的结果是,统计量 T_3=−0.263.查表得 $t_{0.1}$=1.356.检验结果表明:在显著性检验条件下,本文方法所需评价次数以及获得的覆盖率与原始方法无明显差别;而本文方法所需算法运行时间却明显低于原始方法.

上述结果充分说明:本文提出的基于神经网络的测试数据进化生成方法可以在不影响算法性能的前提下,有效降低运行程序需要的时间消耗.特别是对大规模程序,由此节省的时间是非常可观的.

5 结 论

采用遗传算法生成复杂软件的测试数据,是近年来软件测试领域非常有潜力的研究方向之一.该方法首先需要定义合适的适应值函数,从而将测试数据生成问题转化为函数优化问题;在算法的进化过程中,需要以每个测试数据为输入运行插装后的程序,以得到个体的适应值,从而产生巨大的程序运行代价.

鉴于此,本文提出一种基于神经网络的测试数据进化生成方法,采用神经网络模拟个体的适应值,有效降低运行程序的代价:首先,利用一定样本训练神经网络,以模拟个体的适应值;在利用遗传算法生成测试数据时,先利用训练好的神经网络粗略计算个体适应值;对适应值较好的优秀个体,再通过运行程序,获得精确的适应值.

本文方法用神经网络代替被测程序来获得个体的适应值,从而减少了运行程序所需的代价.程序的规模越大,结构越复杂,使用本文方法的优势就越明显.另外,本文方法不仅可以用于路径覆盖测试,还可用于其他类型的覆盖测试.

神经网络的训练精度主要取决于训练样本数据的质量.所以,本文方法的优劣会受到训练数据的影响.如何获得更高质量的训练数据,是下一步需要认真研究的课题.另外,本文方法只是在算法开始时对神经网络进行训练,如果能够随着测试数据的增加对神经网络进行再训练,则其精度会更高.那么,如何在算法运行过程中不断地对神经网络进行重新训练,也是值得进一步研究的问题.

References:

[1] Beizer B. Software Testing Techniques. 2nd., New York: John Wiley & Sons, Inc., 1990.

[2] 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).

[3] Harman M, Afshin Mansouri S, Zhang Y. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. 2009. https://www.researchgate.net/publication/228671024_Search_Based_Software_Engineering_A_Comprehensive_Analysis_and_Review_of_Trends_Techniques_and_Applications

[4] Xanthakis S, Ellis C, Skourlas C, Gal AL, Katsikas S, Karapoulios K. Application of genetic algorithms to software testing. In: Perry D, Jeffery R, Notkin D, eds. Proc. of the Int'l Conf. on Software Engineering and Its Applications. Los Alamitos: IEEE, 1992. 625−636.

[5] Sthamer H. The automatic generation of software test data using genetic algorithms [Ph.D. Thesis]. Pontypridd: University of Glamorgan, 1996.

[6] Wegener J, Baresel A, Sthamer H. Evolutionary test environment for automatic structural testing. Information and Software Technology, 2001,43(14):841−854. [doi: 10.1016/S0950-5849(01)00190-2]

[7] Miller J, Reformat M, Zhang H. Automatic test data generation using genetic algorithm and program dependence graphs. Information and Software Technology, 2006,48:586−605. [doi: 10.1016/j.infsof.2005.06.006]

[8] Michael C, McGraw G, Schatz M. Generating software test data by evolution. IEEE Trans. on Software Engineering, 2001,27(12): 1085−1110. [doi: 10.1109/32.988709]

[9] Hermadi I, Lokan C, Sarker R. Dynamic stopping criteria for search-based test data generation for path testing. Information and Software Technology, 2014,56(4):395−407. [doi: 10.1016/j.infsof.2014.01.001]

[10] Shan JH, Gao YF, Liu MH, Liu HJ, Zhang L, Sun JS. A new approach to automated test data generation in mutation testing. Chinese Journal of Computers, 2008,31(6):1025−1034 (in Chinese with English abstract).

[11] Dong GW, Nie CH, Xu BW. Effectively metamorphic testing based on program path analysis. Chinese Journal of Computers, 2009, 32(5):1002−1013 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2009.01002]

[12] Gong DW, Yao XJ. Testability transformation based on equivalence of target statements. Neural Computing & Applications, 2012, 21(8):1871−1882. [doi: 10.1007/s00521-011-0568-8]

[13] Arcuri A, Yao X. Search based software testing of object-oriented containers. Information Sciences, 2008,178(15):3075−3095. [doi: 10.1016/j.ins.2007.11.024]

[14] Vudatha CP, Jammalamadaka SKR, Nalliboena S, Duvvuri BKK, Reddy LSS. Automated generation of test cases from output domain of an embedded system using Genetic algorithms. In: Proc. of the National Conf. on Emerging Trends and Applications in Computer Science. Kanyakumari: IEEE, 2011. 1−6. [doi: 10.1109/NCETACS.2011.5751411]

[15] Tian T, Gong DW. Model of test data generation for path coverage of message-passing parallel programs and its evolution-based solution. Chinese Journal of Computers, 2013,36(11):441−450 (in Chinese with English abstract).

[16] Yoo S, Harman M. Using hybrid algorithm for Pareto efficient multi-objective test suite minimization. The Journal of Systems and Software, 2010,83:689−701. [doi: 10.1016/j.jss.2009.11.706]

[17] Harman M, McMinn P. A theoretical and empirical study of search based testing: Local, global and hybrid search. IEEE Trans. on Software Engineering, 2010,36(2):226−247. [doi: 10.1109/TSE.2009.71]

[18] Aggarwal KK, Singh Y, Kaur A, Sangwan OP. A neural net based approach to test oracle. ACM Software Engineering Notes, 2004, 29(3):1−6. [doi: 10.1145/986710.986725]

[19] Anderson C, Mayrhauser AV, Mraz RT. On the use of neural networks to guide software testing activities. In: Proc. of the IEEE Int'l Test Conf. on IEEE Computer Society Test Technology Technical Committee and IEEE Philadelphia Section. Washington, 1995. 720−729. [doi: 10.1109/TEST.1995.529902]

[20] Zhang L, Wu FC, Zhang B, Han M. An learning and synthesis algorithm of multilayered feedforward neural networks. Ruan Jian Xue Bao/Journal of Software, 1995,16(7):440−448 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/440.htm

[21] Rumelhart DE, McClelland JL. Parallel Distributed Processing. Cambridge: MIT Press, 1986.

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

[23] Gong DW, Yao XJ. Automatic detection of infeasible paths in software testing. IET Software, 2010,4(5):361−370. [doi: 10.1049/iet-sen.2009.0092]

附中文参考文献:

[2] 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.

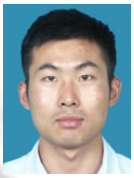
[10] 单锦辉,高友峰,刘明浩,刘江红,张路,孙家骕.一种新的变异测试数据自动生成方法.计算机学报,2008,31(6):1025−1034.

[11] 董国伟,聂长海,徐宝文.基于程序路径分析的有效蜕变测试.计算机学报,2009,32(5):1002−1013. [doi: 10.3724/SP.J.1016.2009.01002]

[15] 田甜,巩敦卫.消息传递并行程序路径覆盖测试数据生成问题的模型及其进化求解方法.计算机学报,2013,36(11):441−450.

[20] 张铃,吴福朝,张钹,韩玫.多层前馈神经网络的学习和综合算法.软件学报,1995,16(7):440−448. http://www.jos.org.cn/1000-9825/16/440.htm

姚香娟(1975－),女,河北赵县人,博士,教授,CCF 会员,主要研究领域为基于搜索的软件工程.



李彬(1989－),男,硕士生,主要研究领域为基于搜索的软件工程.

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