











### 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.

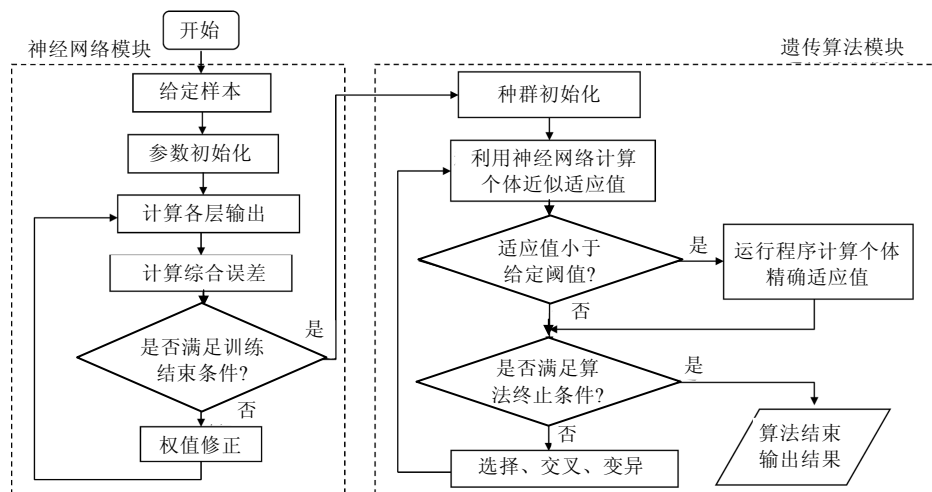


Fig.2 Algorithm diagram

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

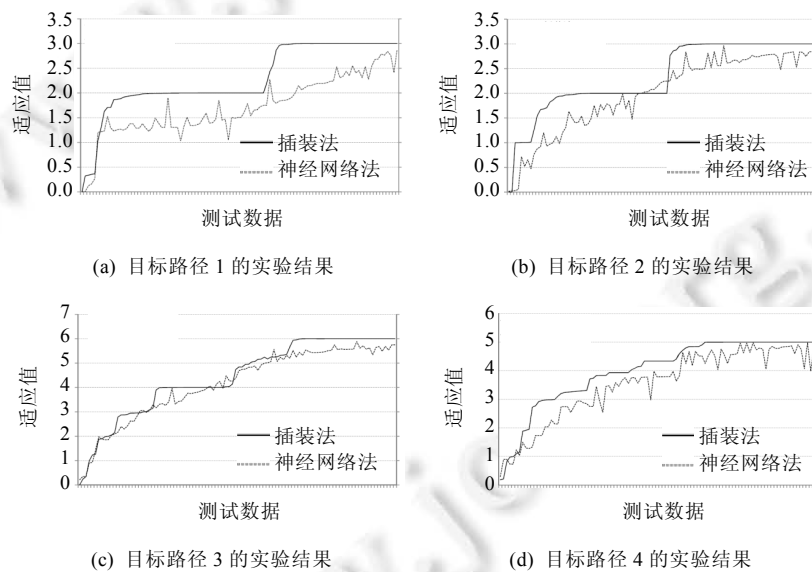
**表 3** 被测程序包含的目标语句数量

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<sup>[23]</sup>,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).



**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](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, Karapoulos 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]

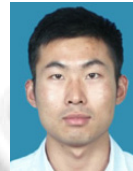
- [14] Vudatha CP, Jammalamadaka SKR, Nalliboena S, Duvvuri BKK, Reddy LSS. Automated generation of test cases from output domain of an embedded system using Genetic algorithms. In: Proc. of the National Conf. on Emerging Trends and Applications in Computer Science. Kanyakumari: IEEE, 2011. 1–6. [doi: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