第 36 卷 第 9 期 计 算 机 工 程 2010 年 5 月
Vol.36 No.9 Computer Engineering May 2010
·软件技术与数据库· 文章编号：1000—3428(2010)09—0056—03 文献标识码：A 中图分类号：TP311.56

面向节点-节点的进化测试改进方法

赵庆兰[1]，董春生[2]
(1. 西安邮电学院信息与控制系，西安 710121；2. 陕西师范大学网络信息中心，西安 710061)

摘 要：传统方法不能对进化测试中所有面向节点-节点的测试类型都构造具有良好导向的适值函数。针对该问题，基于面向节点-节点进化测试系统模型，不考虑节点的执行顺序和控制流关系，从节点的独立性出发，提出一种改进的适值函数计算方法。实验结果表明，对离散节点之间没有数据依赖关系的覆盖准则，该方法代价较小、运行稳定。

关键词：进化测试；遗传算法；适值函数

Improved Node-Node-oriented Evolutionary Test Method

ZHAO Qing-lan[1], DONG Chun-sheng[2]
(1. Department of Information and Control, Xi'an Institute of Posts & Telecommunications, Xi'an 710121;
2. Network Information Center, Shaanxi Normal University, Xi'an 710061)

【Abstract】In evolutionary test, design of fitness function is a crucial work. Traditional work can not design a fitness function with favorable guidance to all node-node-oriented tests. A system model of evolutionary test with node-node-oriented method is presented, and an improved method of fitness function is designed. The method doesn't consider the execute order of nodes and control flow, but is based on the independence of nodes. Experimental result indicates that this new method can generate test data with less cost and more steady performance for cover criterion which has no data dependence in discrete nodes.

【Key words】evolutionary test; genetic algorithm; fitness function

1 概述

软件测试数据的生成通常是由软件测试人员手工构造的，这是一项非常繁重且开销很高的劳动。在费用上，通常要消耗一半以上的软件的开发和维护的费用[1]，在时间上也大大延长了软件的开发周期。因此，测试数据生成的自动化一直都被广泛研究，因为它不仅可以减少软件开发的费用，而且更可以大大提高软件的可靠性，缩短软件的开发周期。进化测试(Evolutionary Test)[2-4]把遗传算法等进化算法应用在测试数据的自动生成问题中，把测试数据的生成问题转化成搜索问题，这种思想的出现使测试数据生成的自动化成为可能。

文献[5]将遗传算法应用到自动测试数据生成中，目前研究焦点都在怎样为程序的某个具体分支或路径找到输入数据，但是应用遗传算法的方法却有所不同，主要体现在不同的测试准则下适值函数的设计方法不一样。只有设计出合理的适值函数，才能引导整个搜索过程向着目标测试用例进化，否则会导搜索过程失败或花费额外的搜索开销。本文针对面向节点-节点的进化测试模型设计了一种新的适值函数。

2 面向节点-节点的进化测试模型

在进化测试中，待求解的问题是一项特定的软件测试目标，遗传算法的任务是从测试数据空间中搜索出一条测试数据，使该测试数据可以满足指定的测试目标。比如测试目标可以是覆盖一条指定的语句节点，该语句所在函数的所有合法输入构成解空间，种群中的每个个体就是一条测试数据。现在的测试环境大都是基于控制流图和数据流图的，而根据测试目的的不同，结构化程序测试覆盖准则[4]可分为：面向节点的方法，面向路径的方法，面向节点-路径的方法，面向节点-节点的方法。而在面向对象的程序测试里由于封装特性增加了基于分支的方法[6]。本文主要讨论面向节点-节点方法，这种方法旨在执行覆盖了控制流图上特定节点组合的程序路径，但这一执行是按照预先确定的节点顺序却不需要指定一条具体的路径。大部分面向控制流的方法，如 all-defs 和 all-uses 等。

面向节点-节点的进化测试生成测试的数据的系统模型如图 1 所示，在该模型中，主要由分析插装模块、驱动模块和遗传模块组成。分析插装模块通过对程序进行静态分析，提取与节点-节点相关的参数，并通过插装技术对待测程序进行插装，构建测试环境。对提取的程序合法输入参数进行编码，即种群初始化，对于每一个个体，通过对该个体的编码进行解码得到被测程序的输入参数值，并利用该参数值调用驱动模块运行实际程序，即可得到衡量该个体所表示的参数值与实际测试用例之间偏离距离远近的适值函数值。通过一个个体所提供的适值函数值，即可对当前种群中的个体进行评价，选择优秀个体，通过变异、交叉等遗传算子操作并进化到下一代种群。随着个体的进化，整个种群便越来越受到较优个体的控制，当整个种群收敛到一个合适的目标解或者在一个最大代数中整个种群已经停止进化时，该算法便停止

基金项目：国家“242”信息安全计划基金资助项目(2005C63)；西安邮电学院青年基金资助项目(110-0419)
作者简介：赵庆兰(1981－)，女，助教、硕士，主研方向：软件测试，信息安全；董春生，助理工程师
收稿日期：2009-10-20 **E-mail：**qlz@snnu.edu.cn

运行并返回最优个体。最优个体就是能覆盖目标节点的测试数据。

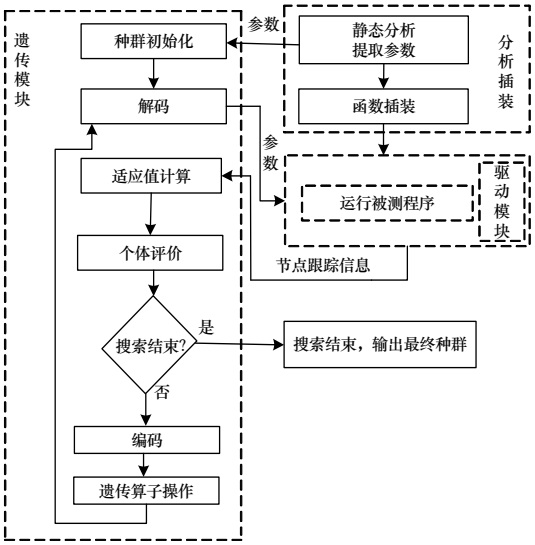

图 1　系统模型

3　适值函数设计

在利用进化测试技术时，最核心的工作是适值函数的设计，只有设计出合理的适值函数，才能引导整个搜索过程向着目标测试用例进化，否则会导致搜索过程失败或花费额外的搜索开销。在不同的测试标准下，适值函数有不同的计算方法。

3.1　关键概念

3.1.1　Branch Distance

Branch Distance 指分支距离。表明当前输入与为了执行另一个分支所需输入之间的靠近程度。表 1 是 Tracey 分支距离的计算方法[7-8]，这是计算分支距离的经典方法，本文的方法也是在这种方法的基础上计算的。

表 1　Tracey 分支距离的计算方法

谓词	分支距离
Boolean	If TRUE then 0 else K
a=b	If abs(a-b)=0 then 0 else abs(a-b)+K
a≠b	If abs(a-b)≠0 then 0 else K
a<b	If a-b<0 then 0 else (a-b)+K
a≤b	If a-b≤0 then 0 else (a-b)+K
a>b	If b-a<0 then 0 else (b-a)+K
a≥b	If b-a≤0 then 0 else (b-a)+K
¬a	Negation is moved inwards and propagated over a

3.1.2　Critical Branch

Critical Branch 指导致测试目标被偏离的分支。如果执行了该类分支，那么程序的此次运行将不会达到目标点。例如在图 2 中，假设目标节点是 6，则所有偏离测试目标的分支都被称为 Critical Branch，如图 2 中的虚线所示。

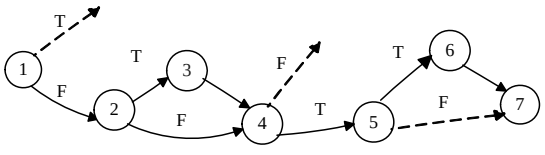

图 2　带有关键分支的控制流图

3.1.3　Approximation Level

Approximation Level 也称 Approach Level，是从程序控制结构的角度衡量一个个体的执行路径与测试目标距离的远近。计算时，考虑只有那些包含了 Critical Branch 的分支才可能错过目标点。例如，一种计算方法，根据当前节点到测试目标之间 Critical Branch 的个数与 1 之间的差值，可以计算出不同 Critical Branch 的 Approximation Level 的大小，例如图 2 中，节点 5 中 F 分支的 Approximation Level 的大小为 1-1=0，则节点 4 中 F 分支的 Approximation Level 的大小为 1，节点 1 中 T 分支的 Approximation Level 为 2。

3.2　一种新的适值函数

文献[4]提出了面向节点-节点方法的适值函数计算方法，该方法根据节点的个数以及节点之间的先后关系给每个节点的 Critical Branch 都赋予了一个 Approximation Level 值，并且对在控制流上处于前边位置的节点，其相应的 Approximation Level 值在其后边最靠近该节点的后继节点的 Approximation Level 值的基础上进行叠加。适值函数的计算为当第一个节点未覆盖到时，则使用第一个节点的适值函数值作为整体的适值函数值，如果第一个节点已被覆盖到，则使用第 2 节点的适值函数值作为整体的适值函数值。因此，这种方法实际上是把面向节点-节点的覆盖变成了逐步到达，逐个节点覆盖的方法。

抛开节点与节点之间严格的先后顺序，可以将每一个节点看成一个独立的待覆盖节点，假设某个属于面向节点-节点类别的覆盖准则需要寻找一个可以覆盖 n 个节点的测试用例，则这 n 个节点都可以被看成一个单独的目标节点，并且对每一个单独节点可采用面向节点的适值函数计算方法进行计算，如式(1)所示，从而整体的适值函数可为这 n 个适值函数值的叠加，如式(2)所示：

$$f = approximation\_level + branch\_distance \tag{1}$$

$$F = \sum_{i=1}^{n} f_i \tag{2}$$

其中，*branch_distance* 被规划到[0, 1]区间，n 为节点个数，而 f_i 则表示第 i 个节点的适值函数值，其计算使用面向节点方法计算。由该计算式可以看出，使用该计算式无需关注不同节点之间的先后顺序，也无需知道不同节点之间的控制流层次关系。为了讨论方便，称这个计算方法为本文方法，而文献[4]中提出的则为传统方法。

在遗传方法中，笔者最关心的就是所设计的适值函数导向性的好坏，本文方法的计算虽然比传统方法简单，但其适用范围究竟与传统方法有什么区别，或在哪些情况下，才是传统方法不适合而本文方法适合的，是较为重要的问题。假设共有 m 个节点，分别为 $N1, N2, \cdots, Nm$，有 2 个遗传个体，分别为

$f_1=f_{11}, f_{12}, \cdots, f_{1k}$

$f_2=f_{21}, f_{22}, \cdots, f_{2k}$

其中，f_1 能够覆盖 m 个节点中前边的 i 个节点，而在节点 i+1 处开始偏离；而 f_2 则刚好相反，它能够覆盖 m 个节点中后边的 j 个节点，在节点 j-1 之前一直与目标节点偏离。按照传统方法的计算，f_1 的适值函数值明显优于 f_2，则 f_2 在选择过程中将被淘汰，于是，虽然 f_2 也能够覆盖部分节点，但其有用信息却在进化过程中被丢弃了，当 f_1 胜出后，它必须继续进化使其可以覆盖后边的节点。而使用本文方法，则 f_1 和 f_2 应该具有同等层次的适值函数值，因此，这 2 个个体都能够胜出并进行交叉，可能会得到这样的个体：

$f_3=f_{11}, f_{12}, \cdots, f_{2k}$

在这种情况下，f_3 不仅保留了 f_1 的有用信息，而且也保留了 f_2 的有用信息，因此，可以避免了传统方法中的再次搜索问题，从而加快了算法的收敛速度。

4 实验分析

在实验中，笔者对例 1 和例 2 分别应用传统方法和本文方法对其建立适值函数，并利用 2 种适值函数设计方法进行测试用例搜索。

例 1

```
void f(int x, int y, int z){
  int t=0;
  if(x==y){
  //node 1
  t=6;
  …}
   if(z==t){
   //node 2
     …}
  if(x>5){
   //node 3
  …}}
```

例 2

```
void f(int x,int y,int z){
  int t=0;
  if(x==y){
  //node 1
  t=6;
   …}
   if(z==2){
   //node 2
     …}
   if(x>z){
   //node3
  …}
  if((x+z)>20){
     //node4
    …}}
```

其中，例 1 有 3 个目标节点，并存在数据依赖。例 2 有 4 个目标节点不存在数据依赖。设定个体的总个数为 50，正交交叉概率为 0.6，变异概率为 0.04，并设定只有在找到测试用例时才允许算法停止退出(在进化测试中，可指定当适值函数的值为 0 时，便找到了测试用例)，对每个例子分别采用传统方法和本文方法各运行 100 次。实验结果如表 2 所示。

表 2　例 1 和例 2 使用不同方法的实验结果

使用方法	例 1 的被测程序		例 2 的被测程序	
	平均执行次数	平均执行次数标准差	平均执行次数	平均执行次数标准差
传统方法	801	301.4	2 089	994.9
本文方法	973	1 130.7	1 478	558.3

例 1 和例 2 虽然简单，但实际上测试用例搜索的困难程度与代码的复杂程度没有关系，而取决于适值函数的导向，所以，代码的简单与否并不影响实验的代表性。由表 2 中可以看出，正如笔者所预测的，如果后边节点中条件分支的控制变量数据依赖于前边节点的变量定义语句，则利用本文方法进行适值函数构造的执行效果差于传统方法，并且运行没有传统方法稳定，标准差远大于传统方法；如果各个节点相互之间不存在数据依赖，则本文方法明显优于从传统方法，而且运行也比传统方法稳定。

在实际的应用中，可根据不同节点之间的数据依赖关系选择适当的方法进行适值函数的构建，如果各个节点分布在不同函数或方法中，而且各个函数或者方法之间没有严格的控制流关系，则这时就无法采用传统方法进行适值函数计算，而必须采用本文方法进行适值函数的构建。这样即可加快搜索进程，提高搜索稳定性，减少搜索代价。

5 结束语

进化测试是一种有效的软件测试用例自动生成技术，为了应用进化测试，其关键的一步就是在不同的覆盖准则下如何为测试目标构建一个合理的、导向性好的适值函数。对于面向节点-节点方法，传统工作虽然也对其提出了一种适值函数的设计思路，然而，该设计思路并不是对所有属于面向节点-节点类别的覆盖准则都能够为其建立一个具有很好导向的适值函数，特别是当各个节点分布在不同函数中，而且各个函数或方法之间没有严格的控制流关系时，传统方法就不再适用了。针对这个问题，本文提出了一种新的适值函数计算方法，并对 2 种方法各自不同的适用领域进行了研究。实验结果证明了对于类似 all-defs 的数据流覆盖准则，不同节点相互之间没有数据依赖关系时，本文方法能够以更小的代价、更稳定的运行效果为测试目标产生高质量的测试用例。

参考文献

[1] Beizer B. Software Testing Techniques[M]. New York, USA: Van Nostrand Reinhold Co., 1990.

[2] Harman M, Jones B. Search-based Software Engineering[J]. Information and Software Technology, 2001, 43(14): 833-839.

[3] Clark J, Dolado J J, Harman M, et al. Reformulating Software Engineering as a Search Problem[J]. IEE Proceedings-Software, 2003, 150(3): 161-175.

[4] Wegener J, Baresel A, Sthamer H. Evolutionary Test Environment for Automatic Structural Testing[J]. Information and Software Technology, 2001, 43(14): 841-854.

[5] Xanthakis S, Ellis C, Skourlas C, et al. Application of Genetic Algorithms to Software Testing[C]//Proc. of the 5th International Conference on Software Engineering and Its Applications. Toulouse, France: [s. n.], 1992.

[6] 艾丽蓉, 赵庆兰, 刘西洋, 等. 面向 Java 语言的进化测试中分支依赖图的构建[J]. 计算机科学, 2006, 33(7): 249-252.

[7] Tracey N, Clark J, Mander K, et al. An Automated Framework for Structural Test-data Generation[C]//Proc. of International Conference on Automated Software Engineering. Hawaii, USA: IEEE Computer Society, 1998.

[8] Tracey N, Clark J, Mander K. The Way Forward for Unifying Dynamic Test-case Generation: The Optimization-based Approach[C]//Proc. of International Workshop on Dependable Computing and Its Applications. Johannesburg, South Africa: [s. n.], 1998.

编辑　金胡考