第16卷 第9期
2006年9月

计算机技术与发展
COMPUTER TECHNOLOGY AND DEVELOPMENT

Vol. 16 No. 9
Sep. 2006

测试数据自动生成方法

邢 恺,伦立军
(哈尔滨师范大学 计算机科学系,黑龙江 哈尔滨 150080)

摘 要:软件测试是提高软件可靠性、保证软件质量的重要手段,可分为静态分析、路径选择、测试数据生成和动态分析四个阶段,而软件测试过程中的一个重要任务是生成测试数据。文中首先给出了遗传算法的形式化描述,然后提出了遗传算法和函数极小化相结合的方法自动生成测试数据,并通过具体实例表明其有效性。

关键词:软件测试;测试数据;遗传算法;函数极小化
中图分类号:TP311.5 **文献标识码**:A **文章编号**:1673-629X(2006)09-0053-03

Automation Generation Methods for Test Data

XING Kai, LUN Li-jun
(Dept. of Computer Sci., Harbin Normal Univ., Harbin 150080, China)

Abstract: Software test is an important way for improving the software reliability and ensuring software quality. There are four phases in software testing: static analysis, path selection, test data generation and dynamic analysis. One of the important tasks during software testing is generation of test data. Discusses the formal description of genetic algorithms at first. Then presents the use of genetic algorithms and function minimization for automatic software test data generation. An example shows its effectiveness.

Key words: software testing; test data; genetic algorithms; function minimization

0 引 言

测试数据自动生成通常可分为功能测试数据生成和结构测试[1]数据生成。根据软件规格需求说明书的功能、性能指标设计测试数据称为功能测试;根据程序的逻辑结构设计测试数据称为结构测试。在结构测试过程中,测试者对程序的语句、分支和逻辑路径进行各种覆盖[2]测试,可以在不同点检查程序的状态,以确定实际状态与预期状态是否一致。但是,不论选择哪种测试策略,测试数据的自动生成[3]都是测试阶段最关键的技术问题。目前,设计测试数据大多采用经验和手工方法,测试成本高、测试效率低、软件质量难以保证。因此改进软件测试方法,提高软件测试的自动化程度,具有十分重要的现实意义。

遗传算法作为一种高效的搜索寻优算法,在解决大空间、多峰值、非线性、全局化等高复杂度问题时显示了独特的优势和高效性。因此将遗传算法引入到传统的程序直接执行技术中,为解决测试数据自动生成问题开阔了视野,提高了其实用化程度。

Pargas, Harrold 和 Peck 提出了利用控制依赖图[4],计算遗传算法的适应度函数;Korel 提出利用函数极小化方法[5,6]生成测试数据;Christoph, Gary 和 Michael 提出遗传算法中适应度函数的构造与函数极小化之间的关系[7],用以生成测试数据;Sthamer 使用遗传算法以生成复杂数据结构[8]的测试数据。

文中提出了基于遗传算法和函数极小化相结合的方法生成测试数据,并将函数极小化用于构造适应度函数,且通过实例验证其有效性。

1 测试数据生成理论

设被测试程序的输入变量为 $x_1, x_2, \cdots, x_n$,输出变量为 $y_1, y_2, \cdots, y_n$,实现功能 M:

$$(x_1, x_2, \cdots, x_n \rightarrow y_1, y_2, \cdots, y_n)$$

$\forall x_i (i = 1, \cdots, n)$,均有一个对应的取值范围 Dx_i,所有输入变量取值的笛卡尔乘积,构成程序的定义域 D,即:

$$D = Dx_1 \times \cdots \times Dx_n = \{(x_1, \cdots, x_n) \mid \forall x_1 \in Dx_1 \wedge \cdots \wedge \forall x_n \in Dx_n\}$$

其中每个 $x \in D$,作为程序的一个输入。

程序结构可用控制流图 $G = (V, E, s, e)$ 表示,其中 V 是结点集,结点表示语句;E 是边集,边表示语句间可能的控制流向;s 是唯一的源结点,对应程序的开始语句;e 是唯一的汇结点,对应程序的终止语句。

由于输入参数数据类型不同,因此不同类型参数的编码映射是不同的。另外,当输入参数有多个时,应首先将每个输入参数单独编码成二进制位串,然后将所有位串连接起来,形成一个个体,称之为多参数二进制级联编码。

收稿日期:2005-12-21
基金项目:黑龙江省教育厅科技资助项目(10541098)
作者简介:邢 恺(1971-),男,天津人,讲师,研究方向为软件工程。

适值函数的构造方法是检测条件语句的真假值关系，若能满足给定的真假值，则适值函数值为 0；否则进行如表 1 所示的运算。

表 1 适值函数值计算方法

表达式	适值函数值
布尔表达式	0;(表达式为 TURE) k;(表达式为 FALSE)
$a = b$	0;(abs(a - b) = 0) abs(a - b) + k;(其它情况)
$a! = b$	0;(abs(a - b)! = 0) k;(其它情况)
$a < b$	0;(a - b < 0) (a - b) + k;(其它情况)
$a \leqslant b$	0;(a - b) ⩽ 0 (a - b) + k;(其它情况)
$a > b$	0;(b - a) < 0 (b - a) + k;(其它情况)
$a \geqslant b$	0;(b - a) ⩾ 0 (b - a) + k;(其它情况)
$a \vee b$	min(fit(a),fit(b))
$a \wedge b$	fit(a) + fit(b)

文中在“分支函数”基础上，引入“分支函数叠加法”。分支函数是一个分支谓词到实际值的映射，可以量化地反映在测试数据的驱动下，被测试程序的实际执行路径对选定路径的覆盖程度。具体做法是(假定选定路径上有 m 个分支点，参与编码的“参数”个数为 n 个)：扫描被测试程序，在决定路径转移的语句自动插入一段代码，同时返回一个适值函数值。若一条路径所经过的条件判断语句不止一个，则将每个适值函数值进行累加，最后的和为此个体的适值函数值。

$\varphi_1 = f_1(x_1,x_2,\cdots,x_n), \varphi_2 = f_2(x_1,x_2,\cdots,x_n), \cdots, \varphi_m = f_m(x_1,x_2,\cdots,x_n)$

最后适值函数值为：

$$F = \Psi(\varphi_1) + \Psi(\varphi_2) + \cdots + \Psi(\varphi_m)$$

其中 $\Psi(x) = \begin{cases} 0 & x \leqslant 0 \\ x & x > 0 \end{cases}$

2 测试数据生成方法

采用遗传算法生成测试数据的步骤如下：

(1)初始化个体：扫描给定的路径，找出需要生成测试数据的变量，并为每个变量赋随机的 0,1 串，按照多参数级联编码原则，组成个体。

(2)计算适值函数值：根据构造的适值函数，分别计算每组变量的适值函数值。若满足终止条件，则转(4)步。

(3)改进个体：若没有满足条件的个体，则进行以下运算：

①按适值函数值选择下一代个体。

②解的交叉：从产生的个体中随机选择两个个体进行单点交叉，得到新个体，重复该步骤，直到所有个体均被选中为止。

③解的变异：在交叉后的个体中随机加入一些变异，产生新个体。

④转到第(2)步。

(4)拆分满足条件的个体：把每个变量对应的 0,1 串转换成十进制数，这些数据即为生成的测试数据。

(5)算法结束。

图 1 是一个被测试程序，下面给出测试数据生成的具体过程，其中编码方式采用有效编码与多参数二进制级联编码相结合原则，适值函数采用分支函数正相叠加法，而叠加前采用函数极小化策略确定各条件语句的适值函数值，种群大小为 10，每个变量的编码长度为 5。

```
  main()
    {
1>   int i,j,k,l;
       scanf("%d,%d,%k",&i,&j,&k);
2>   if(i  >j)
3>        l=i;
           else
4>        l=j;
5>   if(l⩽k)
6>        l=k;
     printf("MAX=%d\n",l);
7> }
```

图 1 一个简单程序

此程序共有四条路径，下面以 $P = 1-2-4-5-6-7$ 为例进行说明。经过扫描可知，参与该路径的变量有 i，j，l，k，然后对它们初始化，如表 2 所示，其中的数字是随机的。

表 2 个体初始化

个体编号	i	j	l	k	i,j,l,k 所对应的十进制数
1	01001	10110	10101	00011	9,22,21,3
2	01110	10110	00101	10010	14,22,5,18
3	10100	01110	11010	00010	20,14,26,2
4	10010	01001	00101	11000	18,9,5,24
5	01101	11010	01100	00111	13,26,12,7
6	10110	11101	10001	01101	22,29,17,13
7	10011	10011	00110	01111	19,19,6,15
8	10101	00110	11001	11001	21,6,25,25
9	01101	01110	11010	10101	13,14,26,21
10	10010	00111	10010	01110	18,7,18,14

根据多参数二进制级联编码原则可知，第 0 代每个个体如表 3 所示。

表 3 第 0 代每个个体

个体编号	个体编码	适值函数值
1	01001101101010100011	0 + 19
2	01110101100010110010	0 + 4
3	10100011101101000010	6 + 18
4	10010010010010111000	9 + 0
5	01101110100110000111	0 + 19
6	10110111011000101101	0 + 16
7	10011100110011001111	0 + 4
8	10101001101100111001	15 + 0
9	01101011101101010101	0 + 7
10	10010001111001001110	11 + 4

然后对每个个体计算其适值函数值，适值函数的主要

作用是判断此组数据能否遍历指定路径,即若某个体的适值函数值为0,则它能遍历此路径,此时的数据即为生成的测试数据。

由于每个个体的适值函数值不为0,而且1,3,5号个体的适值函数值较大,2,7,9号适值函数值较小,所以淘汰适值函数大的个体,小的个体自身复制。

因此,第1代个体如表4所示。

表4　第1代个体

个体编号	父个体	子个体(新个体)	适值函数值
1	0111 0101100010110010	0111 0010010010111000	5+0
2	0111010 1100010110010	0111010 0110011001111	0+4
3	1001 0010010010111000	1001 0101100010110010	0+4
4	1011011011000101 101	1011011011000101 111	0+14
5	1001110 0110011001111	1001110 1100010110010	0+4
6	1001110011001101 111	1001110011001101 101	0+0
7	1 0101001101100111001	1 1101011101101010101	15+8
8	0110101110110 1010101	0110101110110 1001110	0+0
9	0 1101011101101010101	0 1001011101101010101	0+0
10	1001000111100 1001110	1001000111100 1010101	9+0

由于个体6,8,9的适值函数值为0,即得测试数据 $i=19,j=19,l=6,k=29$;$i=13,j=14,l=26,k=14$;$i=9,j=14,l=26,k=21$。

由表4可以看出,有3个个体的适值函数值为0,说明有3组数据可以遍历给定的路径。将个体按照初始化时的方式进行拆分:$i=10011,j=10011,l=00110,k=11101$;$i=01101,j=01110,l=11010,k=01110$;$i=01001,j=01110,l=11010,k=10101$。再将每个变量的0,1串转换成十进制数。$i=19,j=19,l=6,k=29$;$i=13,j=14,l=26,k=14$;$i=9,j=14,l=26,k=21$,即是生成的测试数据,它们能遍历指定的路径 $P=1-2-4-5-6-7$。

3　总　结

测试数据的自动生成是测试阶段最关键的技术问题,改进软件测试方法,对提高软件测试的自动化程度,具有十分重要的现实意义。文中应用遗传算法和函数极小化策略解决软件结构测试数据生成问题,克服了传统的以测试数据为核心的测试方法的不足和缺陷,对软件测试中的测试数据自动生成具有很强的使用价值。

参考文献:

[1] 朱　鸿,金凌紫.软件质量保障与测试[M].北京:科学出版社,1997.

[2] 伦立军,丁雪梅,李英梅.路径覆盖自动生成技术研究[J].计算机工程与应用,2003,39(16):123-125.

[3] 伦立军,丁雪梅,李英梅.基于遗传算法的测试数据生成研究[J].计算机工程,2005,31(23):82-84.

[4] Pargas R, Harrold M, Peck R. Test-Data Generation Using Genetic Algorithms[J]. Journal of Software Testing, Verification & Reliability, 1999, 9(4): 263-282.

[5] Korel B. Automated Software Test Data Generation[J]. IEEE Trans on Software Eng, 1990, 16(8): 870-879.

[6] Korel B. Dynamic Method of Software Test Data Generation [J]. Journal of Software Testing, Verification & Reliability, 1992, 2(4): 203-213.

[7] Christoph C, Gary M, Michael S. Generating Software Test Data by Evolution[J]. IEEE Trans on Software Eng, 2001, 27(12): 1085-1110.

[8] Sthamer H. The Automatic Generation of Software Test Data Using Genetic Algorithms [D]. Great Britain: University of Glamorgan, 1996.

(上接第52页)

至此已经给出了静态验证和动态修改并验证的算法,根据以上的算法,可以在过程的设计阶段和执行阶段随时检查时间约束一致性,保证业务过程能够被有效地执行。

5　结　论

将时间参数引入工作流模型中并对模型进行时间维分析是工作流技术的重要研究内容。文中采用工作流时间约束Petri网模型,对时间约束进行分类,在此基础上提出了动态修改时间约束并判断一致性的算法,该算法便于动态执行时对时间冲突进行违反处理,也可以用于动态运行时检查时间一致性。

参考文献:

[1] Chen Jinjun, Yang Yun, Chen T Y. Dynamic Verification of Temporal Constraints on-the-fly for Workflow Systems [A]. 11th Asia-Pasific Software Engineering Conference, (APSEC'2004)[C]. Busan, Korea: [s. n.], 2004. 30-37.

[2] 范玉顺.工作流管理技术基础—实现企业业务重组[A].过程管理与过程自动化的核心技术[C].北京:清华大学出版社,2001.

[3] Eder J, Panagos E. Time management in workflow systems [A]. BIS'99 3rd International Conference on Business Information Systems [C]. [s. l.]: Springer Verlag, 1999. 265-280.

[4] van der Aalst W M P. The application of Petri nets to workflow management [J]. The Journal of Circuits Systems and Computers, 1998, 8(1): 21-66.

[5] Bowden F D J. A brief Survey and Synthesis of the Roles of Time in Petri nets[J]. Mathematical and Computer Modeling, 2000, 31: 55-68.

[6] 孙瑞志,史美林.一个支持动态变化的工作流元模型[J].电子学报,2002(12A):2052-2056.

[7] 孙瑞志,史美林.支持动态变化的工作流过程元模型[J].软件学报,2003,14:62-67.

[8] Eder J, Panagos E. Time Constraints in Workflow Systems [A]. 11th Conference on Advanced Information Systems Engineering (CAISE99) [C]. Heidelberg, Germany: [s. n.], 1999. 1-14.