基于搜索空间自动缩减的路径覆盖测试数据进化生成

张 岩[1,2],巩敦卫[1]
(1.中国矿业大学信息与电气工程学院,江苏徐州 221116;2.牡丹江师范学院计算机科学与技术系,黑龙江牡丹江 157012)

摘 要: 提出一种基于搜索空间自动缩减的路径覆盖测试数据进化生成方法,首先,确定目标路径与输入变量之间的关系,将可分目标路径分离出与部分分量相关的子路径;然后,固定被穿越子路径对应的输入分量,并缩小交叉和变异操作的范围,使种群在不断缩小的空间里寻找测试数据,以提高测试数据生成的效率;最后,将提出的方法用于基准程序的路径覆盖测试数据生成,并与传统方法和随机法比较.结果表明,本文方法在生成测试数据需要的进化代数、运行时间和成功率等指标上均具有优越性.

关键词: 软件测试; 路径覆盖; 测试数据; 遗传算法; 空间缩减
中图分类号: TP301 **文献标识码:** A **文章编号:** 0372-2112 (2012) 05-1011-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.05.024

Evolutionary Generation of Test Data for Path Coverage Based on Automatic Reduction of Search Space

ZHANG Yan[1,2], GONG Dun-wei[1]
(1. *School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu* 221116, *China*;
2. *Department of Computer Science and Technology, Mudanjiang Normal University, Mudanjiang, Heilongjiang* 157012, *China*)

Abstract: A method of evolutionarily generating test data for path coverage based on automatic reduction of search space is presented in this study. First, the relation between the target path and the input variable is determined, and the divisible target path is divided into some sub-paths related to several elements of the input variable. Then, the elements corresponding to the traversed sub-path are fixed, and the range of crossover and mutation operations is reduced, therefore the original population searches for test data in a reduced space so that the efficiency of generating test data is improved. Finally, the proposed method is applied to generate test data for covering paths of a benchmark program, and compared with the traditional and random ones. The experimental results confirm that the proposed method is advantageous in the number of generations, time consumption and success rate.

Key words: software testing; path coverage; test data; genetic algorithms; space reduction

1 引言

软件测试是保证软件质量、提高软件可靠性的重要手段[1].为指定的程序路径自动生成测试数据是软件单元测试中的一个基本问题[2].使用遗传算法生成覆盖目标路径的测试数据,已成为国内外研究的热点,如Ahmed[3]、Bueno[4]以及Lin[5]等利用遗传算法得到满足路径覆盖的测试数据.我们也给出了一种使用遗传算法生成多路径覆盖测试数据的新方法[6].

采用遗传算法自动生成覆盖目标路径的测试数据,需要将测试数据生成问题转化为函数优化问题.由于程序的输入空间与种群的搜索空间有直接关系,而搜索空间的大小又在一定程度上反映了优化解搜索的难易程度.容易理解,搜索空间越大,找到优化解的速度越慢;相反,搜索空间越小,找到优化解的速度就越快.但是,现有方法使用遗传算法生成路径覆盖测试数据时,都是在全空间内搜索满足要求的测试数据,测试数据的生成效率有待提高.

鉴于此,本文提出一种基于搜索空间自动缩减的路径覆盖测试数据进化生成方法.通过分析目标路径与输入变量的关系,将可分目标路径分离出与部分分量相关的子路径,测试数据进化生成过程中,固定被穿越子路径对应的输入分量,并缩小交叉和变异操作的范围不包括已经固定的分量,使搜索空间不断缩减,有效地提高

收稿日期:2011-06-09;修回日期:2011-09-06
基金项目:国家自然科学基金(No.61075061);黑龙江省高校青年学术骨干支持计划项目(No.1252G063);江苏省自然科学基金(No.BK2010187);高等学校博士学科点专项科研基金(博士生导师类)(No.20100095110006);中国矿业大学校优秀创新团队建设专项基金(No.2011ZCX002);中国矿业大学培育学科创新能力提升基金(No.2011XK09)

了测试数据生成的效率.

2 相关工作

Korel 等在改变变量法动态生成测试数据时提出,如果一个变量的变化将影响该数据已经穿越的正确路径片,或者根本不影响路径的走向,则将该变量的值固定下来[7],实际上是实现了搜索空间的缩减,但是该方法并没有通过实验进行验证.

Offutt 等提出一种动态域缩减的测试数据生成方法[8],通过遍历控制流图中的分支节点,使用分支节点的每个谓词逐步地减小变量的域空间,最终找到穿越目标路径的测试数据.

Harman 等利用缩减搜索空间有效提高了分支覆盖测试数据的进化生成效率[9],通过删除与目标分支不相关的变量实现搜索空间缩减.此外,Harman 等还提出了将搜索空间缩减应用于面向方面程序的测试数据生成[10].

Ribeiro 等针对面向对象程序测试数据生成问题,提出了一种输入域缩减的方法[11],缩减策略是根据纯度分析将不能使目标状态发生改变的方法删除,通过实验验证了该方法的有效性.

上述方法得到与测试目标不相关变量,有的是用手工方式[7,8],也有的是使用代码分析工具[9],但是现有的工具中并不能直接分析出子路径与部分分量的关系,而且将搜索空间缩减用于路径覆盖的测试数据进化生成,目前还没有成熟的研究成果.

3 目标路径与输入变量之间的关系

3.1 基本概念

语句块:程序的基本执行单元称为语句块.其在程序运行中要么都执行,要么都不执行;可能是分支语句的前件、一条语句或多条连续语句.如图 1(a)为 C 语言源程序,其中的 $n_1, n_2, \cdots, n_7$ 均为语句块.

控制流图[7]**:**程序的控制流图是一个有向图 $G=(N,E,s,e)$,其中,N 是节点集,程序的每条语句块都对应一个节点;边集 $E=\{<n_1,n_2>\mid n_1,n_2\in N$,且 n_1 执行后,可能立即执行 $n_2\}$;s 和 e 分别为程序的入口和出口.称出度大于 1 的节点为分支节点.如图 1 中(b)为(a)的控制流图,其中 n_1, n_3 和 n_5 为分支节点.

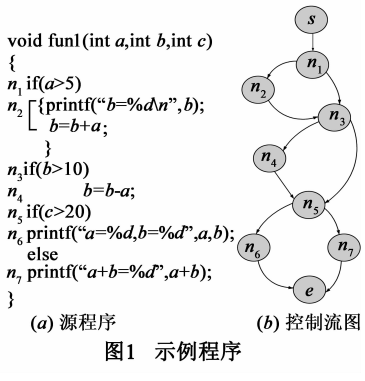


图1 示例程序

路径:程序的路径是一个节点序列"$s, n_1, n_2, \cdots, n_k, e$",其中,$n_i\in N(i=1,2,\cdots,k)$,且 $<s,n_1>\in E$,$<n_i,n_{i+1}>\in E(i=1,2,\cdots,k-1)$,$<n_k,e>\in E$.

被测程序中,一条路径穿越程序的多个节点,输入变量一般有多个分量.为了研究路径与输入变量的关系,首先给出路径节点与输入分量的关系.

3.2 路径节点与输入分量的关系

把节点与输入分量之间的关系分为相关和不相关两种情况.下面分别给出相应的定义.

相关:在路径 P 的节点 n_i 处,分量 x_j 的取值影响 P 在 n_i 或其后的分支节点的走向,称 n_i 与 x_j 相关,记为 $n_i\leftrightarrow x_j$.相关分为直接相关和间接相关两种.

直接相关:如果路径 P 的节点 n_i 与分量 x_j 之间满足如下两种情况之一:

(1) n_i 是分支节点,且有包含 x_j 的条件判断语句;

(2) n_i 中有赋值语句,x_j 出现在赋值语句中,且被赋值的分量出现在 n_i 后续的分支节点中.

则称 n_i 与 x_j 直接相关,记为 $n_i\overset{dc}{\leftrightarrow}x_j$.

如图 1(b)中,选择路径 $P1$:"s, n_1, n_3, n_5, n_6, e",有 $n_1\overset{dc}{\leftrightarrow}a$、$n_3\overset{dc}{\leftrightarrow}b$ 和 $n_5\overset{dc}{\leftrightarrow}c$,都属于第 1 种情况;选择路径 $P2$:"$s, n_1, n_2, n_3, n_4, n_5, n_7, e$",也有 $n_1\overset{dc}{\leftrightarrow}a$、$n_3\overset{dc}{\leftrightarrow}b$ 和 $n_5\overset{dc}{\leftrightarrow}c$,此外,$n_2\overset{dc}{\leftrightarrow}a$ 和 $n_2\overset{dc}{\leftrightarrow}b$ 则属于第 2 种情况.

间接相关:路径 P 的节点 n_i 中虽然没有输入分量 x_j 出现,n_i 与 x_j 不直接相关,但是,与 n_i 相关的其它输入分量与 x_j 有共同的相关节点,或者与 x_j 相关的其它节点与 n_i 有共同的相关分量,称 n_i 与 x_j 间接相关,记作 $n_i\overset{idc}{\leftrightarrow}x_j$.如路径 $P2$ 中由于 $n_1\overset{dc}{\leftrightarrow}a$,且 $n_2\overset{dc}{\leftrightarrow}a$、$n_2\overset{dc}{\leftrightarrow}b$,与 b 直接相关的节点 n_2 与 n_1 有共同的直接相关分量 a,因此,n_1 与 b 间接相关,即 $n_1\overset{idc}{\leftrightarrow}b$.

不相关:如果路径 P 的节点 n_i 与输入分量 x_j 既不直接相关,也不间接相关,称 n_i 与 x_j 不相关,记作 $n_i\not\leftrightarrow x_j$.如路径 $P1$ 和 $P2$ 中都有 $n_5\not\leftrightarrow a$ 且 $n_5\not\leftrightarrow b$.

3.3 路径与输入变量之间的关系

将被测程序的输入变量看成包含 h 个分量的集合,记为 $X=\{x_1,x_2,\cdots x_h\}$;将路径 P 看成 k 个语句块的集合(不包含入口和出口节点),根据集合的性质,如果 P 中有循环执行多次的节点,U 不包含重复的节点,记为 $U=\{n_1,n_2,\cdots,n_k\}$.路径与输入变量的关系 R 可记 $R=\{<n_i,x_j>\mid n_i\in U\wedge x_j\in X\wedge n_i\leftrightarrow x_j\}$.

以路径 $P2$ 为例,在不引起混淆的情况下,仍记它对应的节点集合为 $P2$,则 $P2=\{n_1,n_2,n_3,n_4,n_5,n_7\}$,它与输入分量集合 $X=\{a,b,c\}$的关系记为 $R2$,则有:$R2=\{<n_1,a>,<n_1,b>,<n_2,a>,<n_2,b>,$

$< n_3, a > < n_3, b >, < n_5, c > \}$.

节点与输入分量直接相关容易判定,间接相关需要综合所有直接相关的结果来判定,有一定难度.下面给出相关关系图法确定路径与输入变量的关系.

对于直接相关关系 R,以 U 和 X 的元素为顶点,用"O"表示.如果 $n_i \overset{dc}{\leftrightarrow} x_j$,则在顶点 n_i 和 x_j 之间画一条无向边,称这种方法构造的图为相关关系图.称图中连通 U 的顶点到 X 的顶点的最短路径的边数为步长.上述 $R2$ 的相关关系图如图 2 所示,图中 n_1 到 a 的步长为 1,到 b 的步长为 3.

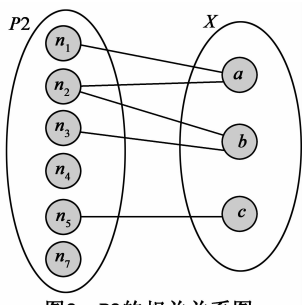


图2　$R2$的相关关系图

定理:如果相关关系图的路径顶点到分量顶点之间有通路,则这两个顶点是相关的;如果通路的步长为 1,则这两个顶点直接相关;如果步长大于 1,则这两个顶点间接相关.

定理的证明可以从直接相关和间接相关的定义得到,不赘述.根据此定理可以方便地判断具有间接相关性的路径节点与输入分量.图 2 的 n_1 与 b 是连通的,且步长为 3,因此 $n_1 \overset{idc}{\leftrightarrow} b$.同理可得 $n_3 \overset{idc}{\leftrightarrow} a$.

可分路径:在 P 的节点集合 U 与输入分量集合 X 的相关关系图中,如果存在一个连通子图,其路径节点集合为 U',输入分量集合为 X',U' 中任何一个节点与 X' 中任何一个分量都相关,与 X' 外的任何一个分量都不相关,且 X' 中任何一个分量与 U' 外的任何一个节点都不相关,记为 $U' \leftrightarrow X'$.若满足 $U' \subset U$ 且 $X' \subset X$,称路径 P 是可分的.

如图 2 中路径 $P2$ 与输入变量 X 的相关关系图中有连通子图,$\{n_1, n_2, n_3\} \leftrightarrow \{a, b\}$ 和 $\{n_5\} \leftrightarrow \{c\}$,因此 $P2$ 是可分路径.

由于很多程序的路径都是可分的,因此,研究覆盖这类路径的测试数据生成问题是非常有意义的.下面,针对该类路径,给出提高测试数据生成效率的方法.

4　基于搜索空间自动缩减的路径覆盖测试数据进化生成

4.1　算法思想

首先,根据目标路径与输入变量的相关关系图,分离出目标路径的子路径,并确定其对应的输入分量;然后,采用遗传算法生成测试数据时,判断是否有个体穿越分离的子路径.如果有,提取该个体与穿越子路径相关的输入分量,用该值修改种群中所有其它个体相应的分量值,且不对该分量所在的基因块实施遗传操作.这样,在进化过程中搜索空间不断缩减,可以有效提高测试数据生成的效率.

4.2　可分目标路径的划分

如果 P 是可分路径,对其相关关系图的一个连通子图 $U' \leftrightarrow X'$,在 P 中找出 U' 的全部节点都包含在内的最小连续子序列,作为 U' 对应的子路径 P'.如果最后一个节点是分支节点,将其下一个节点也包含到 P' 中,得到的子路径 P' 与 X' 的输入分量对应.

如上述路径 $P2$,由 $\{n_1, n_2, n_3\} \leftrightarrow \{a, b\}$,包含 n_1,n_2 和 n_3 的最小连续子序列是"n_1, n_2, n_3",由于 n_3 是分支节点,将其后的 n_4 加到子路径中,分离子路径:"n_1, n_2, n_3, n_4",与分量 a 和 b 对应;又由 $\{n_5\} \leftrightarrow \{c\}$,分离子路径:"$n_5, n_7$",与分量 c 对应.

4.3　个体编码

虽然个体编码方式很多,但是,各种编码方式都容易建立输入分量与基因块之间的对应关系.这里以输入变量是整数并采用二进制编码为例.假定图 1 中 fun1 函数的三个变量 a、b 和 c 的取值范围都为[0,63],那么,个体编码只需要 18 位,其中,前 6 位对应 a,中间 6 位对应 b,最后 6 位对应 c.如果某输入个体编码为 $\underbrace{001111}_{a}\ \underbrace{100001}_{b}\ \underbrace{011001}_{c}$,对应的输入变量为 $a = 15, b = 33, c = 25$,穿越路径"$s, n_1, n_2, n_3, n_4, n_5, n_6, e$".若目标路径为 $P2$:"$s, n_1, n_2, n_3, n_4, n_5, n_7, e$",该个体穿越目标路径的子路径"$n_1, n_2, n_3, n_4$",对应的输入分量为 a 和 b.此时,将其它个体的前 12 位修改为 001111 100001,且交叉和变异操作改为只对后 6 位有效.

4.4　算法描述

步骤 1:对算法的控制参数赋值,确定交叉和变异操作的初始范围,划分目标路径,并插桩被测程序;

步骤 2:初始化种群;

步骤 3:解码进化个体,执行被测程序;

步骤 4:判断是否有与目标路径完全匹配的路径.若有,保存该进化个体,转步骤 8;

步骤 5:判断是否有个体穿越目标路径分离出的子路径.若有,记录被穿越子路径对应的个体基因块取值,修改其它个体相应基因块的取值等于该值,修改交叉和变异操作的范围;

步骤 6:计算进化个体的适应值;

步骤 7:实施选择、交叉和变异操作,生成子代进化种群,转步骤 3;

步骤 8:停止进化,解码进化个体,输出测试数据.

4.5　算法性能分析

本文使用遗传算法生成测试数据,算法的时间复杂度主要考虑适应值的计算次数.假设种群规模为 m,

种群运行 t 代生成测试数据,则遗传算法适应值评估次数为 $m \cdot t$ 次,算法复杂度为 $O(m \cdot t)$.显然,在种群规模 m 不变的情况下,进化代数 t 是影响算法复杂度的重要因素.下面从两个方面分析本文算法通过搜索空间自动缩减,减少算法复杂度的可能性.

首先从搜索空间上分析,某路径的测试数据生成问题可看作如下优化问题:

$$\min \quad f(x_1, x_2, \cdots, x_h),$$
$$x_i \in [a_i, b_i], i = 1, 2, \cdots, h$$

其中 f 为适应值函数,计算方法参见文献[12].定义第 i 个变量的搜索空间测度为 $d_i = b_i - a_i$,则输入变量的搜索空间测度为 $D_1 = \prod_{i=1}^{h} d_i$.若进化到第 t 代,有个体穿越目标路径的子路径,将该子路径对应的分量固定,不失一般性,假设前 w 个分量被固定,搜索空间测度变为 $D_t = \prod_{i=w+1}^{h} d_i$.显然 $D_t \leqslant D_1$,那么,在小的空间测度内搜索,算法的搜索效率将大大提高[13].

其次从交叉变异操作次数上分析.进化过程中若前 w 个分量被固定,所有个体在这 w 个分量上的取值相同,此时交叉点只有落在后 $h - w$ 个未确定分量上才有意义.传统方法中,交叉点选中未确定分量上的概率为 $(h - w)/h$,显然 w 值越大,选择有效交叉点的概率越小,造成许多无效的操作.本文方法修改了交叉范围不包括已固定的分量,交叉点落在未固定分量上的概率为 $(h - w)/(h - w) = 1$,避免了不必要的交叉操作,进化代数会相应减少,算法复杂度得以降低.变异操作与其类似,在此不赘述.

由上述两方面的分析可知,本文算法在搜索效率、进化复杂性方面比使用遗传算法但搜索空间不变的传统方法优越.

5 在基准程序测试中的应用

为了验证本文方法的性能,选择图 3 所示的 sumday 作为基准程序.将本文方法与全空间搜索的传统方法[12]比较,两种方法采用相同的控制参数、适应值计算方法及相同的初始种群,同时也与随机法[14]进行了比较.以找到目标路径的测试数据或种群进化指定代数作为算法的终止条件.为了避免随机因素对算法性能的影响,每组实验使用每种方法各运行 15 次,从找到覆盖目标路径的测试数据需要的进化代数和运行时间进行比较.这两项指标越小,说明算法的性能越好.此外,还比较了 15 次实验中找到测试数据的成功率.

被测程序采用 C 语言编写,在 VC ++ 6.0 环境下运行,机器主频是 2.80GHz,内存为 2GB.采用二进制编码、轮盘赌选择、单点交叉和单点变异,交叉和变异概率分别设为 0.9 和 0.3,种群规模设为 50,设定输入变量的范围均为[0,2047],最大进化代数为 2000 代.

选择 3 条路径作为目标路径,这些路径均是可分的,如表 1 所示.以 $P1$ 为例说明路径的划分方法,首先,构造 $P1$ 包含的节点集合与程序输入分量集合之间的相关关系图,如图 3(c)所示,得到两个连通子图,分别为 $\{n_1\} \leftrightarrow \{year\}$ 和 $\{n_3, n_5\} \leftrightarrow \{month, day\}$,分离子路径 $P1\_1$:"n_1, n_3"和 $P1\_2$:"$n_3, n_5, n_6, n_5, n_6, n_5, n_7$".实验结果如表 2 所示,其中 AVG 为 3 条路径各指标的平均值.

表 1 sumday 的目标路径

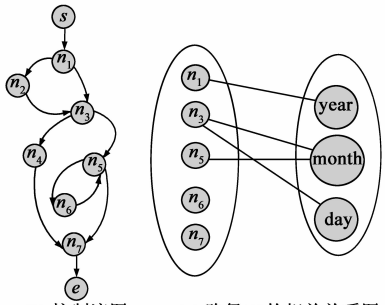
路径	描述	节点序列	分离子路径及其对应的变量
$P1$	合法数据,非闰年,月份为 3 月	"$s, n_1, n_3, n_5, n_6, n_5, n_6, n_5, n_7, e$"	$P1\_1$:"n_1, n_3": $year$ $P1\_2$:"$n_3, n_5, n_6, n_5, n_6, n_5, n_7$": $month, day$
$P2$	合法数据,闰年,月份为 2 月	"$s, n_1, n_2, n_3, n_5, n_6, n_5, n_7, e$"	$P2\_1$:"n_1, n_2": $year$ $P2\_2$:"n_3, n_5, n_6, n_5, n_7": $month, day$
$P3$	合法数据,闰年,月份为 10 月	"$s, n_1, n_2, n_3, n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_6,$ $n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_7, e$"	$P3\_1$:"n_1, n_2": $year$ $P3\_2$:"$n_3, n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_6,$ $n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_6, n_5, n_7$": $month, day$

由表 2(a)可知,对于每条目标路径,本文方法生成测试数据需要的进化代数都是最少的,平均值只有 153.7,传统方法和随机法分别为 961.0 和 1608.3,远大于本文方法;由表 2(b)可知,本文方法生成测试数据需要的运行时间远少于传统方法,与随机法相比优势不明显,这是由于本文方法需要评价个体的时间,而随机法不需要;由表 2(c)可知,本文方法生成测试数据的成功率最高,传统方法次之,随机法最低.随机法生成穿越 $P1$、$P2$ 和 $P3$ 的测试数据成功率分别为 33.3%,6.7% 和 13.3%;传统方法对于 $P2$ 和 $P3$,每次都成功生成了测试数据,而对于 $P1$,成功率只有 86.7%;而本文方法生成穿越 3 个目标路径的测试数据成功率均为 100%.说明本文方法生成测试数据的效率和成功率最高.

```
void sumday(int year, int month, int day)
  {
    int n = 1, month_day_sum = 0, sum = 0;
```

```
        int month_day[13] = {0,31,28,31,30,31,30,31,31,30,31,30,31};
n1      if(year%100!  =0&&year%4 = = 0 ‖ year%400 = = 0)
n2          month_day[2] = 29;
n3      if(day > month_day[month] ‖ day < 1 ‖ month > 12 ‖ month < 1)
n4          printf("error");
        else
        {
n5      while(n < month)
n6      {month_day_sum+=month_day[n];
          n++;
        }
n7      sum=month_day_sum+day;
        printf("%d",sum);
    }
}
```

(*a*)源程序



(*b*) 控制流图　　(*c*) 路径*P*1的相关关系图

图3　被测程序sumday

表 2　不同方法的性能

(*a*)进化代数

路径	*P*1	*P*2	*P*3	*AVG*
本文方法	123.1	220.1	118.0	153.7
传统方法	963.9	1164.7	754.3	961.0
随机法	1706.5	1197.6	1920.9	1608.3

(*b*)运行时间(s)

路径	*P*1	*P*2	*P*3	*AVG*
本文方法	0.012	0.016	0.010	0.013
传统方法	0.072	0.082	0.048	0.067
随机法	0.015	0.016	0.018	0.016

(*c*)成功率(%)

路径	*P*1	*P*2	*P*3	*AVG*
本文方法	100.0	100.0	100.0	100.0
传统方法	86.7	100.0	100.0	95.6
随机法	33.3	6.7	13.3	17.8

为了详细比较本文方法和传统方法在同一目标路径上的测试数据生成效率,体现本文方法搜索空间缩减过程,实验中还记录本文方法每次找到每个子路径的进化代数.以路径 *P*2 为例,结果如表 3 所列.

由表 3 可知,15 次实验中,本文方法每次都在第 1 代就找到穿越子路径 *P*2_1 的个体.固定对应的分量"*year*"后,从第 2 代开始,本文方法就在降低了一维的空间中搜索穿越子路径 *P*2_2 的个体,因此,生成穿越目标路径的测试数据需要的进化代数少.本文方法最多用了 429 代,最少只用了 5 代;而传统方法最多用了 1835 代,最少用了 156 代.本文方法的运行时间也明显少于传统方法,用时最多一次为 0.046 秒;最少用时 0.001 秒,而传统方法最多用时 0.156 秒,最少用时 0.015 秒.从平均进化代数和运行时间上看,本文方法只用了传统方法约 1/5 的进化代数和运行时间就生成了穿越目标路径的测试数据,说明本文采用搜索空间缩减方法能有效提高测试数据生成效率.

表 3　生成路径 *P*2 的测试数据需要的进化代数和运行时间

实验次数	进化代数			运行时间(s)	
	本文方法		传统方法	本文方法	传统方法
	*P*2_1	*P*2_2			
1	1	240	1530	0.021	0.109
2	1	68	810	0.001	0.078
3	1	344	500	0.010	0.016
4	1	429	1293	0.016	0.109
5	1	261	1466	0.031	0.109
6	1	5	156	0.001	0.031
7	1	328	1549	0.015	0.078
8	1	256	1307	0.016	0.093
9	1	34	1835	0.015	0.109
10	1	387	1778	0.046	0.125
11	1	69	267	0.005	0.015
12	1	92	1354	0.006	0.063
13	1	404	746	0.015	0.046
14	1	98	1331	0.015	0.093
15	1	286	1549	0.031	0.156
平均	1	220.1	1164.7	0.016	0.082
方差	0	19879.5	267435.1	0.00014	0.00158

从表 3 的方差数据上看,由于每次实验初始种群差别很大,对生成测试数据的进化代数有一定影响,导致本文方法进化代数方差较大,为 19879.5,但是此值不到传统方法(267435.1)的 1/10;本文方法运行时间的方差(0.00014)也不足传统方法(0.00158)的 1/10,这说明与传统方法相比,本文方法稳定性好.

6　结论

寻求高效的测试数据生成方法,一直是软件测试研究的重要内容之一.本文提出了相关关系图法判定目标路径与输入变量的关系,进一步给出了子路径的划分方法,在测试数据进化生成过程中,当子路径被穿越时,所有个体固定与该子路径对应分量基因块的取值,并缩小交叉和变异操作的作用范围,保证了搜索空间不断缩小,从而提高了测试数据生成效率.

本文方法应用到基准程序测试,并与随机法和全空间搜索的传统方法比较.结果表明,本文方法在进化代数和运行时间方面均具有优越性.

需要说明的是,本文方法的优越性只有对可分路径的测试数据生成才比较明显.此外,目前我们还仅限于人工判定路径是否可分,并分离子路径.如何采用合适的方法自动判定某路径是否可分,并分离子路径,是需要我们进一步研究的问题.

参考文献

[1] 邱晓康,李宣东.一个面向路径的软件测试辅助工具[J].电子学报,2004,32(12A):231 – 234.
Qiu Xiaokang, Li Xuandong. A path-oriented tool suppor-ting for testing[J]. Acta Electronica Sinica, 2004, 32(12A): 231 – 234. (in Chinese)

[2] 单锦辉,王戟,齐治昌.面向路径的测试数据自动生成方法述评[J].电子学报,2004,32(1):109 – 113.
Shan Jinhui, Wang Ji, Qi Zhichang. Survey on path-wise automatic generation of test data[J]. Acta Electronica Sinica, 2004, 32(1): 109 – 113. (in Chinese)

[3] Ahmed M A, Hermadi I. GA-based multiple paths test data generator[J]. Computers and Operations Research, 2008, 35(10): 3107 – 3124.

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

[5] Lin J, Yeh P. Automatic test data generation for path testing using GAs[J]. Information Sciences, 2001, 131(1-4): 47 – 64.

[6] 巩敦卫,张岩.一种新的多路径覆盖测试数据进化生成方法[J].电子学报,2010,38(6):1299 – 1304.
Gong Dunwei, Zhang Yan. Novel evolutionary generation approach to test data for multiple paths coverage[J]. Acta Electronica Sinica, 2010, 38(6): 1299 – 1304. (in Chinese)

[7] Korel B. Automated software test data generation[J]. IEEE Transaction on Software Engineering, 1990, 16(8): 870 – 879.

[8] Offutt J, Jin Z, Pan J. The dynamic domain reduction procedure for test data generation[J]. Software Practice and Experience, 1999, 29(2): 167 – 193.

[9] Harman M, McMinn P, Wegener J. The impact of input domain reduction on search-based test data generation[A]. Proceedings of the ACM SIGSOFT Symposium on the Foundations of Software Engineering[C]. New York: ACM Press, 2007. 155 – 164.

[10] Harman M, Islam F, Xie T, Wappler S. Automated test data generation for aspect-oriented programs[A]. Proceedings of the 8th International Conference on Aspect-Oriented Software Development[C]. New York: ACM Press, 2009. 185 – 196.

[11] Ribeiro J C B, Zenha-Rela M A, Fernández de Vega F. Test case evaluation and input domain reduction strategies for the evolutionary testing of object-oriented software[J]. Information and Software Technology, 2009, 51(11): 1534 – 1548.

[12] McMinn P. Evolutionary search for test data in the presence of state behavior[D]. England: University of Sheffield, 2005. 40 – 56.

[13] 苗金凤,王洪国,邵增珍,赵学臣.基于多级搜索区域的协同进化遗传算法[J].计算机应用研究,2010,27(9):3347 – 3351.
Miao Jinfeng, Wang Hongguo, Shao Zengzhen, Zhao Xuechen. Co-evolutionary genetic algorithm based on multi-level search area[J]. Application Research of Computers, 2010, 27(9): 3347 – 3351. (in Chinese)

[14] Arcuri A, Iqbal M Z, Briand L. Formal analysis of the effectiveness and predictability of random testing[A]. Proceedings of the 19th International Symposium on Software Testing and Analysis[C]. New York: ACM Press, 2010. 219 – 230.

作者简介

张 岩 女,1972 年 5 月出生于黑龙江集贤,副教授,博士研究生,主要研究方向:基于搜索的软件工程.
E-mail: zhangyancumt@126.com

巩敦卫 男,1970 年 3 月出生于江苏铜山,博士,教授,博士生导师,主要研究方向:基于搜索的软件工程、智能优化与控制.
E-mail: dwgong@vip.163.com