面向路径的测试数据自动生成方法述评

单锦辉, 王 戟, 齐治昌
(国防科学技术大学计算机学院, 湖南长沙 410073)

摘 要: 为指定的程序路径自动生成测试数据是软件单元测试中一个基本问题. 求解该问题的实质在于约束系统的建立和求解, 其主要困难之一在于非线性约束求解是一个理论上困难的问题. 文中将面向路径的测试数据自动生成方法分为四类))) 随机法、静态法、动态法和试探法, 分析和比较了每一类中有代表性的方法, 并探讨了研究方向.

关键词: 测试数据自动生成; 程序路径; 约束求解; 数值优化
中图分类号: TP311 文献标识码: A 文章编号: 0372-2112 (2004) 01-0109-05

Survey on Path-Wise Automatic Generation of Test Data

SHAN Jin-Hui, WANG Ji, QI Zhi-Chang
(School of Computer Science, National University of Defence Technology, Changsha, Hunan 410073, China)

Abstract: Automatic generation of test data for a given path in a program is one of the elementary problems in software testing, the essence of which lies in the deriving and solving of system of constraints. The issue of how to solving nonlinear constraint is a hard problem in theory. In this paper, we classify the approaches of generation of test data for a given path into four categories: random, static, dynamic, and heuristic. The representative methods in each category are analyzed and compared. The direction of research is also explored.

Key words: automatic generation of test data; program path; constraints solving; numerical optimization

1 引言

计算机已经应用到国民经济和社会生活各个方面. 作为计算机的灵魂, 软件在其中起着举足轻重的作用. 软件的失效有可能造成巨大的经济损失, 甚至危及人的生命安全. 软件质量问题现已成为制约计算机应用领域的主要因素之一. 许多计算机科学家在展望 21 世纪计算机科学发展方向和策略时, 把提高软件质量放在优先于提高软件功能和性能的地位[1]. 虽然形式化方法和程序正确性证明技术开始走向实用[2], 但软件测试在今后较长时间内仍将是保障软件质量的重要手段. 软件测试在整个软件开发周期中所占比重很大, 但是效率很低[2]. 对程序所完成的功能特性的动态测试可以分为两个范畴: 结构性测试和功能性测试. 结构性测试即为白盒测试, 它用软件的代码确定测试数据. 功能性测试是黑盒测试, 它根据软件所需的功能或/和所实现的功能选择测试数据[1, 3, 4].

白盒测试中的许多诸如控制流测试和数据流测试问题以及黑盒测试中的一些问题都可以归结为面向路径的测试数据生成问题(本文简称为问题 Q), 其非形式化描述为: 给定一个程序 P 和 P 中一条路径 W, 设 P 的输入空间为 D, 求 x∈D, 使得 P 以 x 为输入运行, 所经过的路径为 W.

自动求解问题 Q 将有效地提高单元测试的效率. 求解问题 Q 的实质在于约束系统的建立和求解. 求解约束系统的主要困难之一在于其中所存在的非线性约束. Davis[5] 于 1973 年证明不存在能求解任意约束系统的有效数值方法* . 1979 年, Weyuker 进一步证明不存在有效的算法, 对于任意的 P 和 W, 能生成使 W 被经过的输入数据[6].

尽管这些理论结果令人沮丧, 但是实际应用的需要迫使人们进行研究, 并提出各种方法求解问题 Q. 过去, 人们将各种求解方法分为随机法、静态法和动态法[7]. 静态法对程序进行静态分析和转换, 不涉及程序的实际运行, 它包括符号执行法[8~11]、区间算术法[12] 和文献[13] 提出的方法等. 动态方法则基于程序的实际运行, 其生成测试数据的过程是确定性的, 它包括直线式程序法[14] 和文献[15]、[16] 和[17] 分别提出的方法等. 近年来, 人们将遗传算法[18, 19, 7, 20] 和模拟退火算法[4] 用于求解问题 Q, 这些方法虽然也需要实际运行程序, 但是其生成测试数据的过程不完全是确定性的, 而采用了概率论的思想, 故本文将这些方法归为试探法.

收稿日期: 2001-08-14; 修回日期: 2003-10-28
基金项目: 国家自然科学基金(No. 60233020, 60373003, 90104007); 国家 863 项目(No. 2001AA113202); 霍英东高等院校青年教师基金(No. 71064)

* : 不定方程求解是著名的 Hilbert 第十难题, 由 Hilbert 1900 年提出, 1970 年 Matiyasevič 给出否定的答案, 1973 年 Davis 在文献[5] 给出完整的证明过程. 有兴趣的读者亦可参阅莫绍揆所著《递归论》(北京: 科学出版社, 1987 年) 中相关内容.

对于一种求解问题 Q 的方法,如果对于任意的 P 和 W,其中 W 上的所有分支谓词所对应的谓词函数均为输入变量的线性函数,若 W 是可行的,则该方法能在有限的时间内生成使 W 被经过的输入数据,否则能在有限的时间内判定 W 不可行,那么称该方法对于线性路径约束是完备的,否则对于线性路径约束是不完备的.当 W 上的所有分支谓词所对应的谓词函数中含有输入变量的非线性函数时,即使是可行的,任何求解问题 Q 的方法都不能保证在有限的时间内生成使 W 被经过的输入数据,即任何求解问题 Q 的方法对于非线性路径约束都是不完备的.下面简要介绍和分析面向路径的测试数据生成中有代表性的一些方法.

2 随机法

随机法的基本思想是对输入数据空间 D 进行随机取样,其主要优点是生成单个测试数据的开销较小,简便易行.它可以不受 D 的结构的限制,迅速生成大量的测试数据.但是该方法需要按照产生的大量的测试数据运行程序.故该方法对于求解问题 Q 实质上是困难的,当输入空间无穷时,该方法通常不可行.随机法对于线性路径约束是不完备的.

3 静态法

静态法采用符号执行等方法将路径上的所有(判断)语句转换成约束系统,然后进行求解.静态法所建立的约束系统包括了问题的全部解,它主要包括符号执行[8~11]、区间算术法[12]和文献[13]提出的方法等.

3.1 符号执行

符号执行方法的基本思想是允许程序的输入不仅是具体的数值数据,而且包括符号值.它建立约束系统的方式主要有两种:从前向后替换和从后向前替换[8].

从前向后替换建立约束系统的主要思想是:把符号值作为程序输入,执行路径 W 上的语句,将 W 上各分支谓词用逻辑乘符号联结起来,即得到所需的约束系统.

在从后向前分析路径的过程中[11],每遇到分支谓词就记录下来;每遇到赋值语句,就把已记录下来的所有分支谓词更新.具体的更新过程是:若在布尔表达式中出现了被赋值的变量,则直接替换为赋值语句右端的值;如果被赋值的是数组成员,还必须考虑它的下标和布尔表达式中同名数组的下标是否相等,是则替换,否则不替换.当数组多层嵌套时,必须为数组标上层数,再从里往外更新.与从前向后替换方法不同的是,从后向前替换方法无须保存各变量的符号值,但是两者得到的约束系统相同[8].

文献[8]对输入变量排序,采用解方程、随机法、回溯技术相结合的方法进行求解;文献[9]采用域削减、选择域最小的变量随机赋值和回代相结合的方法进行求解.它们的问题在于,当所建立的约束系统有解时,不保证总能找到解.文献[10]将各有关变量之和定义为目标函数,然后用线性规划求解线性约束系统.文献[11]将从后向前替换方法用于线性路径约束系统,建立约束系统后,用线性规划方法进行求解.由于能采用线性规划方法求解线性约束系统,故符号执行方法对于线性路径约束是完备的.对于非线性路径约束,因为采用近似求解,所以该方法对于非线性路径约束是不完备的.

符号执行方法通常要进行复杂的代数运算,并且难于处理依赖于输入变量的循环条件、数组元素下标和模块调用[10],阻碍了它更广泛的应用.

3.2 区间算术法[12]

区间算术法通过静态分析,将路径 W 上的所有语句转换成变量表和正则约束式表.变量表保存了 W 上各语句定义或引用的所有变量,包括常量和临时变量.正则约束式表示变量之间的约束关系,其形式为:$r = d_1\ \mathrm{option}\ d_2$,其中 option 是+ , - , * , / 等算术运算符或= , X , > , \ , < , [等关系运算符或 and, or, not 等逻辑运算符,d_1, d_2 是整型变量或常量.正则约束式表中用指向变量表的索引表示变量 r, d_1, d_2.

这种方法通过穷举布尔变量值和对参与乘除运算的变量分区间,确定各变量在所有可能的情况下的取值区间,然后根据正则约束式,用区间削减和区间对分穷举相结合的方法逐渐缩小各变量的取值范围,直至找到问题 Q 的解,或者发现 W 不可行.由于正则约束式的引入,该方法能够处理复杂的逻辑表达式,并能处理包含二次函数等的非线性约束.经改进后,该方法能够处理包含三角函数等的表达式.当操作符为普通算术、逻辑、关系等类型运算符的正则约束式的左部量削减成功后,借助逆运算,该方法能对该正则约束式的右部量进行削减.

但是在通常情况下,该方法所生成的约束系统可能包括与求解问题 Q 无关的变量.由于需要对区间进行对分穷举,故当变量的取值区间为无穷区间时,该方法对于线性路径约束是不完备的.另外,该方法需要保存大量的中间结果,故空间消耗很大.

3.3 文献[13]提出的方法

文献[13]提出的方法先将整个程序静态地转换成一个静态单一赋值(Static Single Assignment,简称 SSA)形式,将程序切片与路径条件2约束求解相结合,对于系统依赖图中的路径 W,确定 W 执行的必要条件,简化该路径条件,然后用约束求解器求解,试图获得相应输入变量的值.该文献证明控制依赖图中的圈可以忽略,从而使得任何路径都是有穷的.该方法用二元决策图(Binary Decision Diagram)共享路径条件中的子表达式,以避免路径条件的组合爆炸.该方法要求路径条件中所有变量都是存在量词,以便采用基于量词消解的约束求解器.由于可判定性问题,量词消解只限于各种算术公式,需要用其它的约束求解技术能求解另一些公式.由于文献[13]提出的方法都需要将被测试的程序(路径上的语句)转换成 SSA,故所建立的约束系统都会很大,可能包括与求解问题无关的变量.并且对于线性路径约束是不完备的.

4 动态法

动态法从 D 中选一组数据 x 作为问题 Q 的一个解 y(如果 y 存在)的近似,运行程序 P,根据运行情况,修改 x 的值,使之收敛到 y.由于在程序运行时各输入变量值已经确定,故动态法可以克服静态法中的一些缺点.但是,当问题 Q 有解

时, 动态法不保证总能找到. 动态法可分为直线式程序法[14]和文献[15]、[16]和[17]分别提出的方法等几种.

4.1.1　直线式程序法[14]

直线式程序法通过将 W 上的判断语句替换成形如 $c_i = 0$, $c_i > 0$, $c_i \backslash 0$ 的布尔型赋值语句, 将 W 上的语句转换成等价的直线式程序, 其中 c_i 反映了第 i 个分支谓词接近满足的程度. 选择目标函数 f, 使得若所有路径约束为真, 则 f 的值为正数(或 0), 否则为负数(或 0). 问题 Q 转化为数值最大化问题: 若 f 的值为负数(或 0), 则用数值优化方法最大化 f, 直至其为正数(或 0). 该方法对于线性路径约束是完备的.

直线式程序法能够处理含有三角、指数等类型函数的非线性约束, 并且对被测程序的预处理工作很少. 但该方法要求用户先提供问题 Q 的部分解: 确定所有整数类型的变量值.

对于非线性路径约束, 由于数值优化过程有可能陷于局部极值, 故该方法对于非线性路径约束是不完备的.

4.1.2　文献[15]提出的方法

该方法从 D 中任选一组数据, 按步进的方式运行 P, 即一次前进一个分支谓词. 若当前分支谓词的值不符合要求, 就建立一个分支函数 f. 然后用函数最小化方法调整输入变量值, 使得在仍能经过已成功经过的子路径的前提下, f 值为负数(或 0). 文献[15]先用探测性搜索确定前进方向, 再用模式性搜索使分支函数值尽快达到最小, 并采用动态数据流分析技术确定影响分支谓词的变量, 减小搜索的盲目性. 虽然该方法对于线性路径约束是完备的, 但是模式性搜索的步长与谓词函数的系数相关, 从而导致求解问题 Q 的时间与谓词函数的系数相关. 由于采用了步进方式, 故这种方法可以尽早发现不可行路径. 然而, 由于该方法一次只考虑一个分支谓词和一个输入变量, 并且使用回溯技术, 所以即使 W 上所有谓词函数都是输入变量的线性函数, 它也要进行大量的迭代. 如果 W 上几个分支谓词依赖于公共的输入变量, 那么在回溯时会造成大量的浪费. 另外, 对于非线性路径约束, 该方法只能找到局部极小值, 当谓词函数有多个局部极小值时难以找到问题 Q 的解. 所以该方法对于非线性路径约束是不完备的.

1996 年, 这种方法被扩充为面向目标的链方法[21], 后者被应用于面向断言的测试数据自动生成[22]和回归测试数据自动生成[23].

4.1.3　文献[16]提出的方法

该方法用插装程序强制程序以 D 中任选一组数据为输入, 执行路径 W, 并用插装语句向测试数据生成器返回各变量和分支谓词的状态. 对 P 中每个判断语句进行插装, 使插装语句恰好插在判断语句之前. 对于 if 语句, 插装语句恰好位于该 if 语句之前. 对于 while 循环则需要插入两条插装语句: 一条恰好在该 while 语句之前, 另一条位于该 while 语句的循环体尾部.

这种方法在执行 W 时, 给 W 上每个分支谓词施加一个指数形式的罚函数, 将各罚函数之和定义为目标函数 f, 从而将带约束的数值优化问题简化为不带约束的数值优化问题, 然后用带 BFGS 校正的拟牛顿优化技术[24]最小化 f.

这种方法能生成整型、实型、离散类型的测试数据, 并支持子程序和异常处理. 文献[16]用该方法为 60000 行 Ada 大程序生成测试数据. 由于采用了罚函数, 故该方法可以有效地处理含有逻辑运算符的分支谓词. 为尽早发现不可行路径, 该方法一次只考虑一个分支谓词和一个输入变量, 并且使用回溯技术, 所以会造成大量的资源浪费.

因为罚函数呈指数形式, 所以当初始程序输入距离问题的解太远时, 即使对于线性路径约束, 该方法也有可能无法找到解, 因此它对于线性路径约束是不完备的.

4.1.4　文献[17]提出的方法(简称迭代松弛法)

迭代松弛法采用程序切片[25,26]思想, 从 D 中任选一组输入考察 W 上各分支谓词, 通过静态、动态数据流分析确定各谓词函数对输入变量的依赖关系, 构造谓词片和动态切片, 建立这些分支谓词所对应的谓词函数关于输入变量的线性关系, 进而建立输入变量的增量的线性方程系统, 求解后得到各输入变量的增量, 从而获得一组新的输入.

该方法最大的优点在于所建立的是路径 W 上各谓词函数关于输入变量的线性约束系统. 该方法在通常情况下能够避免文献[15]和[16]提出的方法回溯带来的资源浪费. 当 W 上的谓词函数都是输入变量的线性函数时, 该方法迭代一次或者找到问题 Q 的解, 或者保证 W 不可行, 故该方法对于线性路径约束是完备的. 当谓词函数中含有非线性函数时, 因为该方法是用线性函数来近似非线性函数, 所以可能需要迭代多次才有可能找到问题 Q 的解, 因此该方法对于非线性路径约束是不完备的. 我们对迭代松弛法进行改进, 省略分析数据依赖关系和构造程序片的过程, 按照一组程序输入运行 W 上的语句, 计算 W 上各谓词函数的线性算术表示, 然后直接为输入变量建立线性方程系统, 求解后直接获得一组新的输入. 我们已经证明, 改进后的方法与原方法生成的约束系统相同[27]. 在通常情况下改进后的方法效率更高. 由于改进后的方法在建立约束系统的过程中不必分析路径上语句之间的依赖关系, 故当路径上的某些语句仅为目标代码或可执行代码而无源代码时, 改进后的方法仍然能够工作.

Gupta 等起初采用高斯消去法求解线性方程系统[17], 若自由变量取值不适当, 则可能导致线性方程系统不相容, 需要试探新的取值. 在文献[28]中 Gupta 等提出用线性方程系统的最小二乘解逐步向可行解逼近. 这种解法同样适用于改进后的方法. 目前, 该方法已被用于分支覆盖测试数据的自动生成[29]. 我们以改进后的方法为核心算法, 开发出为指定程序路径自动生成测试数据的原型工具. 初步的实验结果表明改进后的方法是比较有效的, 它也可以用于面向断言的测试数据自动生成[22]和回归测试数据的自动生成[23]. 有关实验结果请参看文献[30].

5　试探法

试探法的基本思想是从 D 中选择输入数据, 运行 P, 然后根据运行结果, 结合概率论的思想产生新的输入数据继续试探. 其优点是不受搜索空间限制性条件(如可微、连续、单峰)的约束, 并且不需要其它辅助性信息(如导数), 所以对于一些传统方法难于处理的大空间、多峰、非线性、全局优化等

高复杂度问题,试探法具有独特的优势和高效性. 然而,试探法对于线性路径约束是不完备的,不保证总能找到解. 试探法主要包括遗传算法和模拟退火算法两种.

511 遗传算法[7, 18, 19, 20]

遗传算法采用编码技术将 D 映射到基因空间 G,并通过选择、杂交、变异等遗传操作和优胜劣汰的自然选择确定搜索方向. 由于它采用种群方式组织搜索,故可以同时搜索 G 内多个区域. 杂交和变异操作为种群引入新的信息,从而更有利于找到全局最优解.

对于遗传算法,CPU 运算时间随输入变量取值范围的增大呈亚线性增长(幂约为 0.5),随机法则为超线性增长(幂大于 1.5). 故遗传算法比随机法更适合于大型程序[18].

例如,文献[19]将遗传算法应用于基于路径覆盖的 Ada 软件结构测试数据的自动生成,并就遗传算法、爬山法和随机法生成测试数据的效率进行对比实验.

尽管遗传算法生成的输入数据数比随机法有明显的改进,但是仍然相当可观. 遗传算法本身很复杂,其理论研究目前相当有限,结果也不太深入,作为遗传算法的理论基石之一的隐性并行性的证明还存在严重缺陷[31]. 另外,文献[7]在计算评价函数时,仅考虑到发生分支违反的分支谓词为止的部分分支谓词,在计算评价函数时考虑路径上所有分支谓词可以更加准确地反映当前输入数据的适应程度.

512 模拟退火算法[4]

模拟退火算法模拟高温材料的冷却(该物理过程称为退火). 其工作原理是在给定的候选解的邻域选择候选解. 关于目标函数更好的候选解总被接受. 但是按一种受控制的方式接受恶化解,其目的是使搜索过程从局部极值中摆脱出来. 用一个控制参数(称为温度)控制是否接受恶化解. 起初温度很高,以允许在搜索空间几乎无限制地自由移动. 在搜索过程中温度逐步降低,限制对恶化解的接受. 最终进入冻结状态,不再接受恶化解,搜索过程简化为简单的爬山法.

模拟退火算法有着完善的全局收敛理论,但是它的收敛速度很缓慢,与遗传算法恰好构成优势互补的关系. 但是至今尚无人在求解问题 Q 时采用退火遗传算法,这是一个有意义的研究方向.

6 结束语

下表对上述各种方法进行对比.

表 1 面向路径的测试数据自动生成方法对比

性质 \ 方法		随机性	静态法				动态法				试探法	
			符号执行		区间算术法	文献[13]方法	直线式程序	文献[15]方法	文献[16]方法	迭代松弛法	遗传算法	模拟退火
			从前向后	从后向前								
线性路径约束	完备	NO	YES	YES	NO	NO	YES	YES	NO	YES	NO	NO
	回溯	NO	NO	YES	YES	YES	NO	YES	YES	NO	NO	NO
	约束求解	N/A	线性规划	线性规划	区间削减,区间对分穷举	量词消解	数值优化方法	函数最小化	带 BFGS 校正的拟牛顿优化	最小二乘求解	N/A	N/A
	工具	YES	YES	YES	YES	YES	NO	YES	YES	YES	YES	YES
非线性路径约束	完备	NO	NO	)	NO	NO	NO	NO	NO	NO	NO	NO
	回溯	NO	YES	)	YES	YES	NO	YES	YES	NO	NO	NO
	约束求解	N/A	近似求解	)	区间削减、区间对分穷举	量词消解	数值优化方法	函数最小化	带 BFGS 校正的拟牛顿优化	最小二乘求解、迭代松弛	N/A	N/A
	工具	YES	YES	)	YES	YES	NO	YES	YES	YES	YES	YES

问题 Q 的研究主要取决于相辅相成的两个方面:建立约束系统和求解约束系统. 约束系统的求解占整个测试数据生成过程中很大一部分开销. 文献[16]对一组线性约束系统的实验表明,测试数据生成的时间随输入变量个数呈近似线性增长,随 w 上分支谓词个数按指数律增长. 当前基于约束求解的各种方法都把所建立的约束系统作为一个整体对待. 然而,在通常情况下,一些输入变量只与部分谓词函数相关. 可以建立输入变量与谓词函数的关系矩阵,通过矩阵的行交换和列交换,将该关系矩阵变换成准对角矩阵,从而按分而治之的思想将原来的约束系统转换成若干个规模更小的约束系统分别进行求解.

问题 Q 的研究近年来虽然取得了可喜的进展,但是离真正实用还有较大的差距. 我们相信,随着研究的不断深入,真正实用的面向路径的测试数据自动生成工具必将出现.

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] King S, Hammond J, Chapman R, Pryor A. Is proof more cost-effective than testing? [J]. IEEE Trans on Software Engineering, 2000, 26(8): 675-686.

[3] Chen H Y, Tse T H, Chan F T, Chen T Y. In black and white: an integrated approach to class-level testing of object-oriented programs [J]. ACM Trans on Software Engineering and Methodology, 1998, 7(3): 250

- 295.
[4] Tracey N, Clark J, Mander K. Automated program flaw finding using simulated annealing [A]. Proc of the ACM SIGSOFT Int. Symp. on Software Testing and Analysis [C]. Clearwater Beach, Florida, USA, 1998. 73- 81.
[5] Davis M. Hilbert s tenth problem is unsolvable[J]. American Mathematics Monthly, 1973, 80: 233- 269.
[6] Weyuker E J. The applicability of program schema results to programs [J]. Int. J. Computer Information Sciences, 1979, 8: 387- 403.
[7] Tracey N, Clark J, Mander K, McDermid J. Automated test2data generation for exception conditions [J]. Software Practice and Experience, 2000, 30: 61- 79.
[8] Ramanmoorthy C V, Ho S2B F, Chen W T. On the automated generation of program test data [J]. IEEE Trans on Software Engineering, 1976, SE22 (4): 293- 300.
[9] DeMillo R A, Offutt A J. Constraint2based automatic test data generation [J]. IEEE Trans on Software Engineering, 1991, 17(9): 900- 910.
[10] Coward P D. Symbolic execution and testing. Information and Software Technology [J]. 1991, 33(1): 53- 64.
[11] Zhang J, Wang X. A constraint solver and its application to path feasibility analysis [J]. International Journal of Software Engineering & Knowledge Engineering, 2001, 11(2): 139- 156.
[12] 王志言, 刘椿年. 区间算术在软件测试中的应用[J]. 软件学报, 1998, 9(6): 438- 443.
[13] Robschink T, Snelting G. Efficient path conditions in dependence graphs [A]. Proc of the 24th Int Conf on Software Engineering [C]. Orlando, Florida, USA, 2002. 478- 488.
[14] Miller W, Spooner D L. Automatic generation of floating2point test data [J]. IEEE Trans on Software Engineering, 1976, SE22 (3): 223- 226.
[15] Korel B. Automated software test data generation [J]. IEEE Trans on Software Engineering, 1990, 16(8): 870- 879.
[16] Gallagher M, Narasimhan V L. ADTEST: A test data generation suite for Ada software systems [J]. IEEE Trans on Software Engineering, 1997, 23(8): 473- 484.
[17] Gupta N, Mathur A P, Soffa M L. Automated test data generation using an iterative relaxation method [A]. Proc. of the ACM SIGSOFT Sixth Int. Symp. on the Foundations of Software Engineering [C]. Orlando, Florida, USA, 1998. 231- 244.
[18] Jones B F, Eyres D E, Sthamer H2H. A strategy for using genetic algorithms to automate branch and fault2based testing [J]. The Computer Journal, 1998, 41(2): 98- 107.
[19] 荚　伟, 谢军　, 奚红宇, 高仲仪. 遗传算法在软件测试数据生成中的应用 [J]. 北京航空航天大学学报, 1998, 24(4): 434-437.
[20] Michael C C, McGraw G, Schatz M A. Generating software test data by evolution [J]. IEEE Trans on Software Engineering, 2001, 27(12): 1085-1110.
[21] Ferguson R, Korel B. The chaining approach for software test data generation [J]. ACM Trans on Software Engineering and Methodology, 1996, 5(1): 63-86.
[22] Korel B, and Al2Yami A M. Assertion2oriented automated test data generation [A]. Proc. of the 18th Int Conf on Software Engineering [C]. Berlin, Germany, 1996. 71-80.
[23] Korel B, and Al2Yami A M. Automated regression test generation [A]. Proc of the ACM SIGSOFT Int. Symp. on Software Testing and Analysis [C]. Clearwater Beach, Florida, USA, 1998. 143-152.
[24] 袁亚湘, 孙文瑜. 最优化理论与方法 [M]. 北京: 科学出版社, 1997.
[25] Weiser M. Program slicing [J]. IEEE Trans on Software Engineering, 1984, SE210(4): 352- 357.
[26] Korel B. Computation of dynamic program slices for unstructured programs [J]. IEEE Trans on Software Engineering, 1997, 23(1): 17- 33.
[27] Shan J H, Wang J, Qi Z C, Wu J P. Improved method to generate path2wise test data [J]. Journal of Computer Science and Technology, 2003, 18(2): 235- 240.
[28] Gupta N, Mathur A P, Soffa M L. UNA based iterative test data generation and its evaluation [A]. Proc. of the 14th IEEE Int. Conf. on Automated Software Engineering [C]. Cocoa Beach, Florida, USA, 1999. 224- 232.
[29] Gupta N, Mathur A P, Soffa M L. Generating test data for branch coverage [A]. Proc. of the 15th IEEE Int Conf on Automated Software Engineering [C]. Macau SAR, China, 2000. 219- 227.
[30] Shan J H, Wang J, Qi Z C. On path2wise automatic generation of test data for both white2box and black2box testing [A]. Proc of the 8th Asia2Pacific Software Engineering Conf [C]. Grenoble, France, 2001. 237 - 240.
[31] 张文修, 梁　怡. 遗传算法的数学基础 [M]. 西安: 西安交通大学出版社, 2000.

作者简介:

单锦辉　男, 1970 年生于江西宜春, 博士, 现为北京大学信息科学技术学院软件研究所博士后, 主要从事软件测试、构件技术、形式化方法等方面的研究. Email: shanjh@163.com.

王　戟　男, 1969 年生于上海, 博士, 教授, 博士生导师, 主要从事高可信软件技术、软件工程、语义 Web 等方面的研究. Email: jiwang@nudt.edu.cn.

齐治昌　男, 1942 年生于北京, 教授, 博士生导师, 主要从事软件工程、软件方法学、软件测试等方面的研究.