基于目标语句占优关系的软件可测试性转化

姚香娟[1,3],巩敦卫[2,3]
(1.中国矿业大学理学院,江苏徐州 221116;2.中国矿业大学信息与电气工程学院,江苏徐州 221116;
3.软件工程国家重点实验室,湖北武汉,430072)

摘 要: 标记变量问题是基于搜索的软件测试数据生成的关键问题之一.本文提出一种基于目标语句占优关系的软件可测试性转化理论与方法,思想是:对于涉及标记变量问题的目标语句,如果存在另一目标语句(集),使得该目标语句(集)占优原有目标语句,则用新的目标语句(集)代替原有目标语句生成测试数据,从而消除标记变量的不利影响.将本文方法应用于典型被测程序,实验结果表明,该方法可以有效解决标记变量问题,从而提高测试数据的生成效率.

关键词: 测试数据生成; 标记变量; 可测试性转化; 遗传算法
中图分类号: TP311.5 **文献标识码:** A **文章编号:** 0372-2112 (2013) 12-2523-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2013.12.033

Testability Transformation Based on Dominant Relation of Target Statements

YAO Xiang-juan[1,3], GONG Dun-wei[2,3]
(1. *College of Science, China University of Mining and Technology, Xuzhou, Jiangsu* 221116, *China*;
2. *School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu* 221116, *China*;
3. *State Key Laboratory of Software Engineering, Wuhan, Hubei* 430072, *China*)

Abstract: Flag problem is a key problem in search based software test data generation. This study proposed a testability transformation method based on the dominant relationship of target statements. The basic idea is that: for a target statement involving flag variables, if another target statement (or target statement set) dominates the original one, then the original target statement is substituted with the new one to generate test data. Experimental results showed that the proposed method can effectively solve the flag problem, therefore improve the efficiency of generating test data.

Key words: test data generation; flag variable; testability transformation; genetic algorithm

1 引言

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[1,2].要把遗传算法应用到测试数据生成问题,首先需要定义一个合适的目标函数.但是,如果条件语句存在标记变量,构造目标函数时将会出现信息缺失,从而导致相应的目标函数分布特性差,难以引导搜索过程找到问题的最优解[3,4].

标记变量又称布尔变量,是指有两种逻辑状态的变量,包含两个值:真和假.标记变量广泛存在于实际的程序中[5],即使由代码生成器创建的程序,例如 Matlab/Simulink/Stateflow,也包含大量标记变量[6].因此,标记变量问题是基于搜索的软件测试数据生成的关键问题之一.

近年来,人们提出各种可测试性转化方法解决标记变量问题[7].该类方法通过在原始的源程序中插入一些代码,将其转化为另一个源程序,使得转化后的源程序对应的优化问题的目标函数具有期望的分布特性,从而高效地引导遗传算法搜索到覆盖目标语句的测试数据.

但是,现有可测试性转化方法需要在源程序中插装大量代码,需要耗费很多工作量;另外,为了消除标记变量的不利影响,往往要使用多条语句,代替源程序中某一条语句,导致插装后的程序代码行成倍增加,不但增

收稿日期:2013-05-07;修回日期:2013-08-05;责任编辑:诸叶梅
基金项目:国家自然科学基金(No.61203304,No.61075061);高等学校博士学科点专项科研基金(博士生导师类)(No.20100095110006);江苏省自然科学基金(No.BK2012566,No.BK2010187);中央高校基本科研业务专项资金(No.212QNA41)

加了转化后程序的测试复杂度,而且增大了程序出错的概率;此外,每种插装方法能够解决的问题类型很有限.上述问题大大限制了已有转化方法在实际测试中的应用.

为有效解决标记变量问题,本文提出一种基于目标语句占优关系的可测试性转化方法.首先,给出目标语句占优关系的定义,再给出基于语句占优关系的测试目标转化方法,从而消除标记变量的影响.该方法不需要对源程序进行转化,而是对目标语句进行转化,克服了已有方法的缺陷和不足.

2 可测试性转化

尽管遗传算法已经成功解决了很多测试数据生成问题,但是,对于含有标记变量的被测程序,用遗传算法生成测试数据覆盖目标语句的效率却很低,基本上退化为随机法.其根源在于,与之对应的优化问题的目标函数分布特性差,难以高效地引导进化种群的搜索.

为解决标记变量问题,人们提出各种可测试性转化方法,基本思想是:通过在源程序插入一些代码,以增加足够的辅助信息,从而使得构造的目标函数具有更好的分布特性.另外,要保证满足转化后程序充分性准则的测试数据也符合源程序的要求.

Wappler 等人通过将分支补充完整,用双精度类型替代布尔类型的标记变量,以及距离插装等措施,使转化后的程序具有更加光滑的适应值函数[8].

Binkley 等人考虑循环语句体内对标记变量赋值的测试数据生成问题.他们通过增加中间变量记录标记变量的赋值情况,以及循环语句的执行次数,并通过距离插装,使得转化后的源程序对应的优化问题的目标函数具有更好的导向性[9].Jiang 等人对含有 return 和 goto 等非结构化语句的程序进行了转化,在转化后的源程序中,通过中间变量记录循环语句的执行次数,并将其包含在新的目标函数中,以改善目标函数的分布特性[10].

McMinn 等人考虑多层嵌套程序的测试数据生成问题,通过合并谓词将多层嵌套结构"扁平化",减少嵌套的层次,使得转化后的源程序对应的优化问题的目标函数有一定的坡度[11].

Bottaci 提出一种目标函数替换方法解决标记变量问题:找到标记变量的赋值语句,用赋值语句对应的分支距离代替原来的分支距离[12].而 Baresel 和 Sthamer 则把标记变量赋值语句对应的分支距离增加到适应值函数的计算中[13].

在 Liu 等人的工作中,标记变量被分为有需求和无需求两类.当一个标记变为有需求时,就使用额外增加的变量的适应值函数来对个体进行评价[14].

上述研究成果提供了有效解决标记变量问题的多种途径.然而,现有可测试性转化方法需要对程序代码作大量转化,所用的转化方法往往非常繁琐,转化后的程序也常常变得非常冗长、复杂,甚至破坏了程序的原有结构,降低了其可读性.因此,研究新的可测试性转化理论与方法非常必要.

3 基于目标语句占优关系的可测试性转化

本节给出一种新的转化方法来解决标志变量问题.该方法需要转化的不是源程序,而是测试目标.

3.1 基本概念

设 γ_i 和 γ_j 为两个目标语句,如果 γ_i 被执行,γ_j 一定会被执行,则称 γ_i 占优 γ_j,记为 $\gamma_i \rightarrow \gamma_j$.

另外,设 γ 是一条目标语句,Γ 是一目标语句集,如果当 Γ 中所有目标语句都被执行时,γ 一定会被执行,则称 Γ 占优 γ,记为 $\Gamma \rightarrow \gamma$.

设目标语句 γ 包含于条件语句 δ 的真分支或假分支,则称条件语句 δ 支配目标语句 γ.

目标语句之间之所以存在占优关系,主要在于支配这些目标语句的条件语句之间存在相关性[15].

设 δ_i 和 δ_j 为两个条件语句,如果当 δ_i 的输出为真时,δ_j 的输出必然为真,称(δ_i,δ_j)具有真-真相关性;如果当 δ_i 的输出为真时,δ_j 的输出必然为假,称(δ_i,δ_j)具真-假相关性.类似地,可以定义条件语句之间的假-真和假-假相关性.

设当条件语句 $\delta_1,\cdots,\delta_t$ 取 $\triangleleft_1,\cdots,\triangleleft_t$($\triangleleft_i \in$ {真,假})时,条件语句 δ 的取值一定为 $\triangleleft$($\triangleleft \in$ {真,假}),则称$(\delta_1,\cdots,\delta_t,\delta)$具有 $\triangleleft_1\cdots\triangleleft_t \rightarrow \triangleleft$ 相关性.

语句之间相关性的判定方法可以参考文献[15].可以根据条件语句之间的相关性来判定它们所控制的目标语句之间的占优关系.

定理 1 设目标语句 γ_i 属于条件语句 δ_i 的真分支,且目标语句 γ_j 属于条件语句 δ_j 的真分支,那么 $\gamma_i \rightarrow \gamma_j$ 当且仅当(δ_i,δ_j)具有真-真相关性.

证明 必要性:当 δ_i 的输出为真时,因为 γ_i 属于 δ_i 的真分支,则 γ_i 被执行;由于 $\gamma_i \rightarrow \gamma_j$,则 γ_j 必然被执行,而 γ_j 属于 δ_j 的真分支,这样 δ_j 的输出必然为真,因此(δ_i,δ_j)具有真 – 真相关性.

充分性:当则 γ_i 被执行时,δ_i 的输出必然为真.因为(δ_i,δ_j)具有真-真相关性,δ_j 的输出必然也为真.又因为 γ_j 属于条件语句 δ_j 的真分支,所以 γ_i 被执行.因此 $\gamma_i \rightarrow \gamma_j$.

对于其他几种情况,比如 γ_i 属于条件语句 δ_i 的真分支,且 γ_j 属于条件语句 δ_j 的假分支,可以使用类似的方法进行证明,这里就不一一给出.

对于目标语句集占优某一目标语句的情形,可以

给出以下判定方法.

定理 2　设 $\Gamma=\{\gamma_1,\gamma_2,\cdots,\gamma_t\}$是一个包含 t 个目标语句的语句集,目标语句 γ_i 属于条件语句δ_i 的$\triangleleft_i(\triangleleft_i\in\{$真,假$\})$分支,且目标语句 γ 属于条件语句δ 的$\triangleleft(\triangleleft\in\{$真,假$\})$分支,那么 $\Gamma\rightarrow\gamma$ 当且仅当$(\delta_1,\cdots,\delta_t,\delta)$具有 $\triangleleft_1\cdots\triangleleft_t\rightarrow\triangleleft$ 相关性.

(证明略)

3.2　基于目标语句占优关系的可测试性转化

本文提出的可测试性转化方法思想如下:设 γ 是涉及标记变量问题的目标语句,则寻找另外一条目标语句(或目标语句集),使得该目标语句(集)占优原有目标语句.如果新的目标语句(集)不涉及标记变量问题,则用新的目标语句(或语句集)代替原有目标语句.其主要步骤如下:

步骤 1　考察分支相关性

设 γ 是一条涉及标记变量问题的目标语句,控制 γ 的条件语句为 δ,δ 包含一个标记变量,记为 *flag*.经验表明,对变量 *flag* 进行赋值的语句很可能与目标语句存在占优关系.那么,应用文献[15]所提出的方法考察控制 *flag* 赋值语句的条件语句,和控制原有目标语句 γ 的条件语句δ 之间的相关性.

步骤 2　确定语句之间的占优关系

如果步骤 1 确定相应条件语句之间存在相关性,则根据定理 1 和定理 2 判定目标语句之间的占优关系.

步骤 3　用新的目标语句(集)代替原有目标语句

如果存在目标语句 γ'(或语句集 Γ)占优原有目标语句 γ,则将其作为新的测试目标来生成测试数据.如果有多个这样的语句,则选择位于程序最前面的语句作为新的测试目标.因为目标语句越靠前,执行程序的代码行也就越少,就越简单.

步骤 4　构造针对新的目标语句(集)的适应度函数

为了利用遗传算法生成测试数据,需要把测试数据生成问题转变为一个优化问题,其中目标函数的结构非常关键,其构造方法见文献[16].

步骤 5　优化问题的求解

可以使用多种方法来对步骤 4 建立的优化问题进行求解.本文采用遗传算法作为求解该问题的工具.遗传算法的基本流程如图 1 所示.

3.3　本文方法的优越性

和已有的可测试性转化方法相比,本文方法具有以下优点:

(1)在本文方法中,所转化的不是程序的源代码,而是目标语句.

已有可测试性转化方法,需要在源程序中插装大量代码.本文提出的可测试性转化方法,不对源程序进行转化,而是根据目标语句之间的占优关系转化需要覆盖的目标,从而消除标记变量的影响.

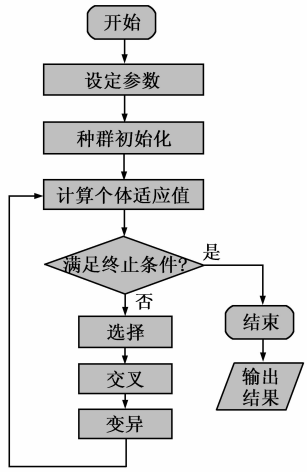

图1　遗传算法流程图

(2)利用本文方法对测试目标进行转化后,目标函数值的计算复杂度大大减少.

一般情况下,新的目标语句总是位于原有目标语句的前面.这样,为了得到目标函数值而需要执行的代码行数会大大减少;另外,决策变量的数目也可能会降低,从而使搜索空间得以缩减.

(3)转化后的测试目标对应的目标函数具有较好的光滑性和分布特性.

通过目标语句之间的转化,新的测试目标将不再涉及标记变量问题,从而可以使相应的目标函数具有很好的分布特性,以引导遗传算法找到问题的最优解.

(4)本文方法可用于解决各种类型的标记变量问题.

不论是简单的标记变量问题,带有嵌套结构的标记变量问题,或者是函数赋值的标记变变量问题,都可以通过考察目标语句之间的占优关系进行合理转化.因此,本文的转化方法具有更广泛的适用性.

4　实例分析

本节将通过一个具体的小程序来详细说明本文方法的主要步骤.如图 2 所示,该程序的输入变量为 m 和 n.设定其输入域均为$[-50,50]$.原有目标语句是语句 12.其对应的分支距离形式如下:

$$\text{branch\_dist}=\begin{cases}0, & \text{object\_fun}()=\text{true}\\ 1, & \text{object\_fun}()=\text{false}\end{cases}$$

另一方面,

$$\text{object\_fun}()=\begin{cases}\text{true}, & m=n \text{ 或 } m=n-10\\ \text{false}, & \text{否则}\end{cases}$$

因此

$$\text{branch\_dist} = \begin{cases} 0, m = n \text{ 或 } m = n - 10 \\ 1, \text{否则} \end{cases}$$

```
1   bool object_fum(int m,int n)
    {
2       bool result;
3       if(m==n)
4           result=true;
5       else if(m==n-10)
6           result=true;
7       else
8           result=false;
9       return result;
    }
10  main(int m,int n)
    {
11      if(object_fun(m,n))
12          printf("Yes!\n")
        ……
```

图2 示例程序代码

如图 3 所示. 该函数在直线 $m = n$ 和 $m = n - 10$ 上的值为 0,其他各点处的函数值都是 1. 这样的目标函数不能对进化过程起到引导作用. 下面介绍如何利用本文的方法进行可测试性转化.

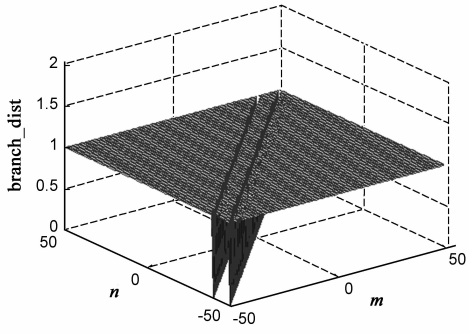

图3 转化前的分支距离

步骤 1 考察分支相关性

目标语句 12 位于条件语句 11 的真分支,而后者的谓词是输出为标记变量的函数 Decision_fun(). 函数 Decision_fun()返回的是标记变量 result 的值. 而 result 分别在条件语句 3 和条件语句 5 的真分支或假分支内进行赋值,因此需要考察语句(3,11) 和(5,11)的相关性.

应用文献[15]所提出的确定分支相关性的方法,确定语句(3,11)和(5,11)具有真 - 真相关性.

步骤 2 确定语句之间的占优关系

由步骤 1 和定理 1,语句 4 和语句 6 都占优语句 12,即语句 4→语句 12,语句 6→语句 12.

步骤 3 用新的目标语句(集)代替原有目标语句

由步骤 2,语句 4 和语句 6 都占优语句 14. 在这种情况下,把位于程序前面的语句 4 作为新的目标语句来代替原来的目标语句. 因为在同等条件下,目标语句越靠前,相应的测试数据生成问题就越简单.

步骤 4 构造针对新的目标语句(集)的适应度函数

对新的目标语句 4,其适应度函数定义如下:

$$\text{branch\_dist}^* = |m - n|$$

branch_dist^* 的图形如图 4 所示. 可以看出该函数可以提供足够的引导来找到最优解. 覆盖语句 4 的测试数据生成问题可以转化为以下优化问题:

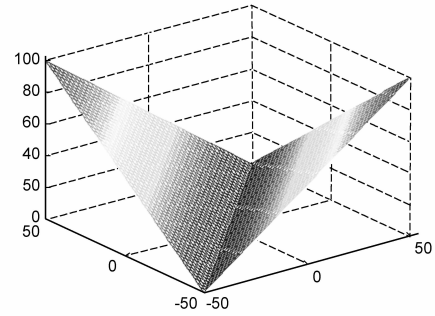

图4 转化后的分支距离

$$\begin{aligned} &\min \quad \text{branch\_dist}^* \\ &\text{s.t.} \quad (m,n) \in [-50,50]^2 \end{aligned} \tag{1}$$

步骤 5 优化问题的求解

对于式(1)所描述的优化问题,利用遗传算法很容易得到其最优解.

5 实验

5.1 被测程序

我们选用 7 个不同大小和难度的实际程序进行实验,基本信息如表 1 所列. 这些程序均被广泛用于各种软件测试和分析实验,具有一定代表性.

表 1 被测程序基本信息

被测程序	代码行	子程序个数	条件语句数
Postcode	170	5	25
Sliding Window	137	10	72
Sortcode	95	4	13
Cadp	11068	480	174
Go	28547	2982	489
Li	6916	364	94
Prepro	14328	530	374
Spice	149050	7254	2368

由于本文只需要对含有标记变量问题的目标语句进行转化,所以我们只选择一部分需要进行可测试转化的语句进行研究,目标语句的数量见表 2.

5.2 实验设计

进行实验的目的主要有两个:一是对本方法的适用性进行验证;二是对本文方法的效率进行验证.

表 2　被测程序包含的目标语句数量

被测程序	目标语句数量
Postcode	10
Sliding Window	10
Cadp	100
Go	100
Li	100
Prepro	150
Spice	150

针对第一个实验目的,对每个目标语句,判定是否可以利用本文方法进行转化.针对第二个实验目的,把本文方法和另外两种方法进行了对比,即随机法和 Binkley 方法.随机法指不对程序做任何转化,利用随机方法生成测试数据;Binkley 方法是指文献[9]提出的可测试性转化方法.进行对比的性能指标是生成覆盖所有目标语句的评价次数和覆盖率.

在进行第二组实验时,针对每个被测程序和每种方法,在相同条件下进行 30 次独立实验,记录每次实验生成所有测试数据需要的个体评价次数,以及所生成测试数据集的覆盖率.这里,覆盖率指被覆盖的目标语句个数与所有目标语句个数的比值.

输入变量采用实数编码,种群规模为 100,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 0.8 和 0.3.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,本实验为 1000.

5.3　实验结果

针对第一个实验目的,我们记录了每个目标语句的可转化情况,结果如表 3 所示.其中,转化率是指可以转化的目标语句个数和目标语句总个数之比.

表 3 目标语句的转化率

被测程序	目标语句个数	可转化个数	转化率(%)
Postcode	10	8	80
Sliding Window	10	10	100
Cadp	100	97	97
Go	100	98	98
Li	100	90	90
Prepro	150	141	94
Spice	150	134	89.3

由表 3 可以看出,除了程序 Postcode 和 Spice 之外,本文方法的转化率都达到 90% 以上.这就说明,本文提出的可测试转化方法,可以解决绝大多数标记变量问题,同时,不需要对程序本身进行转化.

第二组实验中,本文方法无法转化的目标语句,用原始方法生成测试数据.实验结果如表 4 所列,其中,评价次数和覆盖率是 30 次实验结果的平均值.

从表 4 可以看出,除了程序 Li 之外,本文方法需要的评价次数总是最少的.本文方法所需评价次数最少的是 Sliding Window,为 9273.对同一程序,随机法和 Binkley 方法需要的评价次数分别为 9516 和 15590.本文方法所需评价次数最多的是 Spice,为 28153.对同一程序,随机法和 Binkley 方法需要的评价次数分别为 82841 和 39745,明显多于本文方法.对程序 Li,本文方法需要的评价次数为 23452,随机法和 Binkley 方法分别为 38543 和 22719.

表 4　测试数据生成的效率和覆盖率

程序	随机法		Binkley 方法		本文方法	
	评价次数	覆盖率(%)	评价次数	覆盖率(%)	评价次数	覆盖率(%)
Postcode	29453	64.9	36335	92.1	14636	93.4
Sliding Window	9516	75.3	15590	96.6	9273	95.7
Cadp	41354	84.6	17623	98.2	13426	98.4
Go	43137	81.3	35364	99.7	31723	99.5
Li	38543	61.7	22719	96.2	23452	96.3
Prepro	58524	65.3	42179	93.5	35781	95.2
Spice	82841	69.9	39745	95.7	28153	96.3

从表 4 还可以看出,对所有程序而言,除了程序 Sliding Window 和 Go 外,本文方法得到的测试数据的覆盖率总是最高的.对 Sliding Window,本文方法的覆盖率为 95.7%,而随机法和 Binkley 方法分别为 75.3% 和 96.6%.对 Go,本文方法的覆盖率为 99.5%,而随机法和 Binkley 方法分别为 81.3% 和 99.7%.虽然本文方法的结果稍微低于 Binkley 方法,但都远远高于随机法.

为了更加科学的对结果进行分析,对评价次数采用 T 检验方法进行对比,显著性水平 $\alpha = 0.1$.本文方法和随机法对比的结果,统计量 $T_1 = 2.2159$;本文方法和 Binkley 方法对比的结果,统计量 $T_2 = 1.4211$.查表得 $t_{0.1} = 1.356$.检验结果表明,本文方法显著优于随机法,也优于 Binkley 方法.

上述结果充分说明,本文提出的可测试性转化方法,可以有效解决大部分标记变量问题.另外,由于不需要对程序进行插装,因此,相对传统的可测试性转化方法,大大提高了测试的效率.

6　结论

本文提出一种新的可测试性转化方法解决标记变量问题.和已有方法的区别在于,本文方法转化的不是源程序,而是目标语句.对一个涉及标记变量问题的目

标语句,只需找到另一个对其存在占优关系的目标语句,来代替原有目标语句.和传统方法相比,本文方法实现起来要容易得多;另外,新的目标语句总位于原来的目标语句的前面,这也有利于提高测试数据生成的效率.实验结果充分验证了所提方法的优越性.

参考文献

[1] 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 Jin-hui, Wang Ji, Qi Zhi-chang. Survey on path-wise automatic generation of test data[J]. Acta Electronica Sinica, 2004,32(1):109 – 113.(in Chinese)

[2] 姚香娟,巩敦卫.基于路径比较的变异测试方法[J].电子学报,2012,40(1):103 – 107.
Yao Xiang-juan, Gong Dun-wei. Mutation testing based on comparison of paths[J]. Acta Electronica Sinica, 2012, 40(1): 103 – 107.(in Chinese)

[3] P McMinn. Evolutionary Search for Test Data in the Presence of State Behaviour[D]. Sheffield, England: the University of Sheffield, 2005.

[4] M Harman, L Hu, et al. Improving evolutionary testing by flag removal[A]. Proceedings of the Genetic and Evolutionary Computation Conference[C]. New York, USA: Morgan Kaufmann, 2002. 1359 – 1366.

[5] F Lammermann, A Baresel, J Wegener. Evaluating evolutionary testability for structure-oriented testing with software measurements[J]. Applied Soft Computing, 2008, 8(2): 1018 – 1028.

[6] M Halstead. Elements of Software Science[M]. New York, England: Elsevier Science Inc, 1977.

[7] M Harman, L Hu, et al. Testability transformation[J]. IEEE Transactions on Software Engineering, 2004, 30(1): 3 – 16.

[8] S Wappler, J Wegener, A Baresel. Evolutionary testing of software with function-assigned flags[J]. The Journal of Systems and Software, 2009, 82(11): 1767 – 1779.

[9] D W Binkley, M Harman, K Lakhotia. Flag remover: A testability transformation for transforming loop assigned flags[J]. ACM Transactions on Software Engineering and Methodology, 2009, 2(3): 110 – 146.

[10] S Jiang, Y Lu. Evolutionary testing of unstructured programs using a testability transformation approach[A]. Proceedings of Japan-China Joint Workshop on Frontier of Computer Science and Technology[C]. Nagasaki, Japan: IEEE, 2008. 59 – 66.

[11] P McMinn, D Binkley, M Harman. Empirical evaluation of a nesting testability transformation for evolutionary testing[J]. ACM Transactions on Software Engineering and Methodology, 2009, 18(3): 1 – 26.

[12] L Bottaci. Instrumenting programs with flag variables for test data search by genetic algorithms[A]. Proceedings of the Genetic and Evolutionary Computation Conference[C]. New York, USA: Morgan Kaufmann, 2002. 1337 – 1342.

[13] A Baresel, H Sthamer. Evolutionary testing of flag conditions[A]. Proceedings of the Genetic and Evolutionary Computation Conference[C]. Chicage, USA: Springer, 2003. 2442 – 2454.

[14] X Liu, H Liu, B Wang, et al. A unified fitness function calculation rule for flag conditions to improve evolutionary testing[A]. Proceedings of the International Conference on Automated Software Engineering[C]. Long Beach, USA: ACM, 2005. 337 – 341.

[15] D W Gong, X J Yao, M Xia. Automatic determination of branch correlations in software test[A]. Proceedings of the World Congress on Software Engineering[C]. Xiamen, China: Newswood Limited, 2009. 211 – 215.

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

作者简介

姚香娟 女,1975 年出生于河北赵县,博士,副教授,硕士生导师,主要研究方向为基于搜索的软件工程.
E-mail: yxjcumt@126.com

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