软件学报 ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*,2015,26(8):1925−1936 [doi: 10.13328/j.cnki.jos.004671]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

# 测试含有标志变量程序的占优语句(集)选择*

巩敦卫[1], 钟超群[1], 姚香娟[2]

[1](中国矿业大学 信息与电气工程学院,江苏 徐州 221116)

[2](中国矿业大学 理学院,江苏 徐州 221116)

通讯作者: 巩敦卫, E-mail: dwgong@vip.163.com

**摘 要**: 基于占优关系的可测试性转化,是将目标语句覆盖问题转化为位于该语句之前的占优语句(集)覆盖问题,能够对含有标志变量的程序进行测试.但是当占优语句(集)不止一个时,如何从这些语句(集)中选择最容易覆盖的作为新的目标语句(集),至今没有有效的方法,从而限制了可测试性转化的应用范围.研究了占优语句(集)选择问题,提出了基于覆盖难度的占优语句(集)选择方法.首先,提出评价语句覆盖难度的 4 个指标,并给出这些指标的计算方法;然后,基于上述指标,利用 Topsis 方法排序,选择最容易覆盖的占优语句(集);最后,将所提出的方法应用于多个基准与工业程序测试,实验结果表明,覆盖基于该方法选择的占优语句(集)能够显著提高测试数据生成的效率.

**关键词**: 软件测试;语句覆盖;标志变量;可测试性转化;占优语句

**中图法分类号**: TP311

中文引用格式: 巩敦卫,钟超群,姚香娟.测试含有标志变量程序的占优语句(集)选择.软件学报,2015,26(8):1925−1936. http://www. jos.org.cn/1000-9825/4671.htm

英文引用格式: Gong DW, Zhong CQ, Yao XJ. Dominant statement(s) selection in testing programs with flag variables. Ruan Jian Xue Bao/Journal of Software, 2015,26(8):1925−1936 (in Chinese). http://www.jos.org.cn/1000-9825/4671.htm

## Dominant Statement(s) Selection in Testing Programs with Flag Variables

GONG Dun-Wei[1], ZHONG Chao-Qun[1], YAO Xiang-Juan[2]

[1](School of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou 221116, China)

[2](College of Science, China University of Mining and Technology, Xuzhou 221116, China)

**Abstract**: Testability transformation based on dominant relationship, which transforms the problem of covering the target statement into the problem of covering the dominant statement(s) preceding the target statement, can test programs with flag variables. When more than one dominant statement exist, however, there have been no effective methods for selecting a statement subset with best coverage as the new target(s), which limits the scope of applying the testability transformation method. The problem of selecting dominant statement(s) is investigated in this paper, and a method of choosing dominant statement(s) is presented based on the coverage difficulty. First, four indicators for evaluating the coverage difficulty of a statement are presented, and the approaches to calculating them are provided. Then, the dominant statement(s) with best coverage is (are) chosen using Topsis sorting based on the above indicators. Finally, the proposed method is applied to test several benchmarks and industrial programs, and the experimental results show that coverage from the dominant statement(s) selected by the proposed method can greatly improve the efficiency of generating test data.

**Key words**: software testing; statement coverage; flag variable; testability transformation; dominant statement

软件测试是为了发现软件可能存在的缺陷或错误,能够提高软件的可靠性[1].在软件测试中,采用合适的方法自动生成有效的测试数据,对于提高测试效率具有非常重要的作用.

近年来,遗传算法在结构覆盖测试数据生成方面得到国内外学者的高度关注,并取得丰硕的研究成果.在诸

* 基金项目: 国家自然科学基金(61375067, 61203304); 江苏省自然科学基金(BK2012566)
收稿时间: 2014-02-17; 修改时间: 2014-05-09; 定稿时间: 2014-05-29

多结构覆盖准则中,语句覆盖是一种常用的准则.语句覆盖测试数据生成是指采用合适的方法,生成程序的输入以覆盖给定的语句,该给定的语句称为目标语句.可以看出,对于语句覆盖测试数据生成问题,我们的目的是生成能够覆盖目标语句的测试输入,而不关心该测试输入对应的输出.为了采用遗传算法生成测试数据,首先将语句覆盖测试数据生成问题转化为一个约束优化问题[2];然后,采用遗传算法求解上述优化问题时,基于被测程序的插装,计算进化个体的适应值.多次迭代之后,即可生成覆盖目标语句的测试数据.

若标志变量作为程序中某条件语句的谓词表达式(的一部分),且目标语句位于该条件语句的真(或假)分支中,那么生成覆盖该目标语句的测试数据将非常困难,这类问题称为标志变量问题.这里提到的标志变量通常是谓词表达式的布尔变量,具有非真即假的特点[3].即使采用遗传算法,求解标志变量问题的效率依然很低.这是因为该问题建模后,优化问题的目标函数分布特性差,使得基于此设计的适应度函数难以正确引导种群的进化[4].

到目前为止,已有多种求解标志变量问题的有效方法.我们提出了基于占优关系的可测试性转化方法,可用于解决标志变量问题[5].该方法将目标语句覆盖问题转化为该语句占优语句(集)的覆盖问题;转化后,问题的目标函数具有良好的分布特性,提高了基于此设计的适应度函数的导向性,从而缩短了采用遗传算法生成测试数据的时间.

对于复杂程序而言,某目标语句的占优语句(集)往往不止一个,且不同占优语句(集)的覆盖难度可能不同.此时,如果从这些占优语句(集)中选择最容易覆盖的作为新的目标语句,那么将能够进一步提高测试数据生成的效率.考虑到程序插装过程依赖于选择的占优语句(集),并直接决定进化个体适应值的计算,因此,最容易覆盖的占优语句(集)的选择对于采用遗传算法高效生成覆盖目标语句的测试数据是非常关键的.但是,评价占优语句(集)的覆盖难度,并从众多占优语句(集)中选择最容易覆盖的是一个全新的问题,至今还缺乏有效的方法.

鉴于此,本文研究占优语句(集)选择问题,提出基于覆盖难度的占优语句(集)选择方法,主要贡献体现在如下 3 个方面:(1) 提出评价语句覆盖难度的 4 个指标,并给出这些指标的计算方法;(2) 提出最容易覆盖占优语句(集)的选择方法;(3) 通过多个基准和工业程序的测试,验证所提出方法的有效性.

本文第 1 节综述相关工作,提出占优语句(集)选择方法.第 2 节详细加以阐述,包括语句覆盖难度的评价指标、占优语句(集)的选择方法以及示例等.第 3 节通过在基准和工业程序测试的应用,验证所提出方法选择占优语句(集)的有效性.最后,第 4 节总结全文,并指出需要进一步研究的问题.

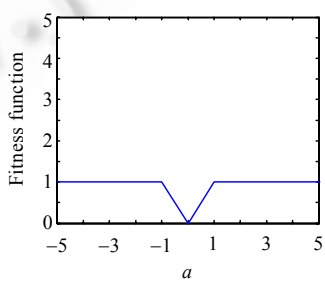
## 1 相关工作

### 1.1 标志变量问题

标志变量广泛存在于实际程序中[6],即使通过代码生成器,如 Matlab,Simulink 以及 Stateflow 生成的代码,也包含大量的标志变量.如图 1(a)所示,该程序的输入变量为 $a$,目标语句为语句 6,那么覆盖该语句的测试数据生成问题就是标志变量问题.

```
   % input a
1    int flag=0;
2    if a==0
3       flag=1;
4    end
5    if flag
6       //target statement
7    end
```

(a)




(b)

Fig.1 Program with a flag and its fitness function curve

图 1 含有标志变量的程序及其适应度函数曲线

采用随机法很难解决标志变量问题,而采用遗传算法求解该问题时,设计的适应度函数曲线如图 1(b)所示.容易看出,除了输入变量为 0、适应度函数取值为最小值 0 之外,对于其他输入变量取值,适应度函数值均为 1.也就是说,适应度函数的分布呈现“平坦”的特性,这使得进化种群无法根据适应度函数的取值搜索期望的测试数据.

### 1.2 基于代码改写的可测试性转化

为了解决上述问题,Harman 等人提出了基于代码改写的可测试性转化方法[3].该方法通过插入一些代码,将原程序转化为另一个程序,使得与转化后程序覆盖问题对应优化问题的目标函数,具有良好的分布特性,从而使所设计的适应度函数能够高效引导种群的进化,产生期望的测试数据.

借鉴上述思想,Wappler 等人考虑函数调用对标志变量赋值的测试数据生成问题,通过将分支补充完整、用双精度类型替代布尔类型的标志变量以及距离插桩等措施,使得转化后程序覆盖问题对应优化问题的目标函数更加光滑[7].Binkley 等人研究循环语句体对标志变量赋值的测试数据生成问题,在转化后程序中采用中间变量记录标志变量重新赋值时循环语句执行的次数并进行距离插桩,使得转化后程序覆盖问题对应优化问题的目标函数更具有导向性[8].Jiang 等人考察含有 return 和 goto 语句的非结构化程序测试数据生成问题,在转化后程序中,利用中间变量记录执行非结构化语句之前循环语句执行的次数,并将其包含在对应优化问题的目标函数中[9].此外,McMinn 等人考虑多层嵌套程序的测试数据生成问题,通过谓词复合,将多层嵌套结构“扁平化”,从而减少嵌套层次,使得转化后程序对应优化问题的目标函数有一定的坡度[10].

不难看出,已有的可测试性转化方法需要在原程序中插装很多代码.程序代码的增加,不仅延长了转化后程序执行需要的时间,而且增大了转化后程序出错的概率;此外,对原程序插装也需要很多额外的工作量.这说明,上述方法解决标志变量问题时仍然具有很大局限性.这限制了该方法在实际软件测试的应用.

### 1.3 基于占优关系的可测试性转化

面向克服基于代码改写的可测试性转化方法存在的不足,我们提出了基于占优关系的可测试性转化方法,用于解决标志变量问题.为了便于理解该方法的思想,首先引入语句占优关系[5]的概念.

考虑被测程序 $P$ 的两条语句,记为$\gamma_i$和$\gamma_j$,对于程序的任一输入,如果$\gamma_i$被执行时$\gamma_j$也一定被执行,那么称$\gamma_i$占优$\gamma_j$,记为$\gamma_i \rightarrow \gamma_j$,称$\gamma_i$为占优语句,$\gamma_j$为被占优语句;对于语句集$\{\gamma_{i1},\gamma_{i2},\ldots\}$和程序的任一输入,如果$\{\gamma_{i1},\gamma_{i2},\ldots\}$被执行时$\gamma_j$也一定被执行,那么称语句集$\{\gamma_{i1},\gamma_{i2},\ldots\}$占优$\gamma_j$,记为$\{\gamma_{i1},\gamma_{i2},\ldots\} \rightarrow \gamma_j$,相应地称$\{\gamma_{i1},\gamma_{i2},\ldots\}$为占优语句集[5].对于程序的一条语句,可能不存在占优语句(集),也可能只存在一个占优语句(集),还可能存在多个占优语句(集).程序的不同语句之间之所以具有占优关系,主要在于与这些语句关联的条件语句之间存在相关性.

我们提出的可测试性转化方法基于条件语句的相关性[5]寻找目标语句的占优语句(集),将覆盖目标语句的测试数据生成问题转化为生成覆盖该语句的占优语句(集)的测试数据问题,而与后者对应优化问题的目标函数具有良好的分布特性,从而提高采用遗传算法生成测试数据的效率.与基于代码改写的可测试性转化方法相比,该方法不对原程序的代码进行任何修改,避免了程序转化导致的诸多不良后果,而仅转化待覆盖的目标语句,使得基于转化后目标语句构造的适应度函数具有一定的坡度,从而正确引导种群的进化.

但是,基于占优关系的可测试性转化方法也产生如下问题:当目标语句的占优语句(集)不止一个时,由于不同的占优语句(集)具有不同的覆盖难度,因此,如果选择覆盖难度大的占优语句(集)作为新的目标语句(集),将难以提高测试数据生成效率.这说明,采用合适的方法寻找目标语句的所有占优语句(集),并选择合适的占优语句(集)作为新的目标语句(集),对于提高测试数据生成的效率是非常有帮助的.

## 2 提出的方法

面向解决基于占优关系的可测试性转化方法存在的问题,以进一步提高语句覆盖测试数据生成的效率,本文提出了基于覆盖难度的占优语句(集)选择方法,其思想是:基于条件语句相关性程序[5],自动获得目标语句的所有占优语句(集)之后,基于评价语句覆盖难度的 4 个指标,利用 Topsis 方法排序,选择最容易覆盖的占优语句

(集)作为新的目标语句.

可以看出,本文方法需要解决的关键技术包括:(1) 确定语句覆盖难度的评价指标,这些指标反映某占优语句(集)作为新的目标语句时,生成测试数据的难易程度;(2) 给出最容易覆盖的占优语句(集)的选择方法,针对每一占优语句(集),计算上述指标值并进行排序,得到最容易覆盖的占优语句(集).

### 2.1 语句覆盖难度的评价指标

为了选择最容易覆盖的占优语句(集),需要给出语句覆盖难度的评价指标.鉴于遗传算法是一种重要的测试数据生成方法,因此,本节基于采用遗传算法生成语句覆盖测试数据的代价[11],反映语句覆盖的难度.不难理解:生成覆盖某语句测试数据的代价越大,该语句就越难覆盖.

总体上讲,采用遗传算法生成测试数据的代价包含如下两个方面:(1) 执行遗传算法的代价;(2) 执行被测程序的代价.其中,当种群的进化策略固定时,执行遗传算法的代价主要取决于进化个体适应值的计算;而执行被测程序的代价,与占优语句(集)所在的路径密切相关.鉴于此,在确定语句覆盖的难度时,主要从进化个体适应值的计算和占优语句(集)所在路径的执行两个方面考虑.本节中,由执行遗传算法的代价定义了关键变量个数和占优语句执行概率两个指标,而由执行被测程序的代价定义了最短路径所在的代码行数和谓词数量两个指标.具体定义如下文所示.

(1) 关键变量个数

容易知道,如果一个占优语句位于某条件语句的真(假)分支中,那么该占优语句被执行的条件是,相应条件语句谓词表达式的真值为真(假).而上述谓词表达式的取值与该表达式包含的程序输入变量有关.当输入变量的取值范围固定时,输入变量个数越多,这些变量形成的决策空间就越大,在该空间中找到期望测试数据的概率也就越小[12].但是,如果不存在程序的输入变量决定该表达式的真值,那么该占优语句往往是不可执行的.

以上分析表明,占优语句的覆盖难度与相应条件语句谓词表达式包含的程序输入变量密切相关.该变量称为关键变量,该条件语句称为关键语句;关键变量个数越多,该占优语句越难覆盖.为便于描述,记占优语句$\gamma_i$的关键变量个数为$Kvar(\gamma_i)$.

(2) 占优语句执行概率

采用遗传算法生成测试数据时,适应度函数的建立十分关键.适应度函数通常由层接近度和分支距离组成.其中,分支距离用来衡量使一个谓词表达式为真(或假)的条件满足程度(取决于目标语句位于真还是假分支).如果条件满足,那么分支距离为 0;否则,分支距离与关键语句的谓词表达式有关[13].研究表明:对于简单谓词,由“==”形成的谓词表达式比“<”,“≤”,“≥”,“>”以及“≠”形成的谓词表达式更难满足;对于复杂谓词,$e_1$ 和 $e_2$ 均为简单谓词表达式,由 $e_1\&\&e_2$ 形成的谓词表达式比谓词表达式 $e_1$ 或 $e_2$ 更难满足,而由 $e_1||e_2$ 形成的谓词表达式却比谓词表达式 $e_1$ 或 $e_2$ 更容易满足[14].

由上述分析可知,占优语句的覆盖难度与关键语句谓词表达式为真(或假)的概率有关.如果占优语句在关键语句的真分支中,那么占优语句执行的概率即为关键语句谓词表达式为真的概率.显然,占优语句执行的概率越大,在程序的输入空间中找到期望测试数据的概率就越大,占优语句也就越容易被执行.

不失一般性,假设占优语句$\gamma_i$在关键语句的真分支中,记$\gamma_i$执行的概率为 $p(\gamma_i)$.如果形成关键语句的谓词为简单谓词,被测程序$P$的输入变量取值范围为$S(P)$,$\gamma_i$的关键语句谓词表达式为真的程序输入变量的取值范围为$S(\gamma_i)$,那么 $p(\gamma_i)$可以表示为

$$p(\gamma_i)=\frac{|S(\gamma_i)|}{|S(P)|} \tag{1}$$

其中,$|S(P)|$和$|S(\gamma_i)|$分别表示 $S(P)$和 $S(\gamma_i)$的测度[15].其中,$|S(P)|$的计算与程序输入变量的个数与这些变量的取值范围有关,$|S(\gamma_i)|$的计算与关键变量的个数与这些变量的取值范围有关.如果 $S(P)$是一个有限集,那么$|S(P)|$等于该集合包含的元素个数;如果 $S(P)$是一个一维区间,那么$|S(P)|$等于该区间的长度;如果 $S(P)$是一个二维区间,那么$|S(P)|$等于该区间的面积;$S(P)$是一个三维区间,那么$|S(P)|$等于该区间的体积;$S(P)$是一个高维区间,那么$|S(P)|$等于该区间的超体积.类似地,可以得到$|S(\gamma_i)|$的值.

同理,若形成关键语句的谓词为复杂谓词,本节仅考虑关键语句包含 2 个简单谓词表达式的情况,记该关键语句为 $e=e_1$ op $e_2$,其中,$e_1$ 和 $e_2$ 均为简单谓词表达式,此时,根据 op 的形式,采用如下方式确定$\gamma_i$执行的概率:

$$p(\gamma_i)=\begin{cases}\dfrac{|S(e_1)\cup S(e_2)|}{|S(P)|}, & \text{op}=\|\\ \dfrac{|S(e_1)\cap S(e_2)|}{|S(P)|}, & \text{op}=\&\&\end{cases} \tag{2}$$

对于关键语句包含 3 个及以上简单谓词表达式的情况,采用类似的方法,能够得到$\gamma_i$执行的概率.

(3) 最短路径包含的代码行数

对于占优语句(集)而言,包含该语句的路径具有的代码行数,是指从程序的起始语句开始,到该占优语句为止的子路径包含的代码行数.在程序测试中,代码行数是度量软件开发规模最直观的指标,常被用于衡量路径的执行难度[16].代码行数大体分为如下 2 种:物理行数和逻辑行数,其中,物理行数是指与计算机存储有关的代码行数,即执行路径中所有语句的个数;逻辑行数反映程序代码的规模,为执行路径中互不相同语句的个数.如果一个程序不包含循环语句,那么该程序的物理行数等于逻辑行数;否则,该程序的物理行数大于逻辑行数.尽管如此,由于物理行数的计算方法简单,因此,本节采用物理行数表示某路径的代码行数.

为了比较不同占优语句(集)所在路径执行的代价,理想的方法是获取所有经过这些占优语句(集)的路径,并计算它们的物理行数之和的平均值.对于一个复杂程序而言,由于难以获得所有经过某占优语句(集)的路径,且统计上述平均值所需的计算量很大,因此,本节从这些路径中选择具有最少代码行的路径,该路径包含的代码行数记为 $Loc(\gamma_i)$,反映$\gamma_i$覆盖的难度.一般来讲,路径包含的代码行数越少,该路径的执行时间就越短.

(4) 最短路径包含的谓词数量

研究表明:对于同一目标语句的两个占优语句$\gamma_1$ 和$\gamma_2$,经过$\gamma_1$ 和$\gamma_2$ 的路径包含的谓词数量分别表示为 $Pred(\gamma_1)$和 $Pred(\gamma_2)$,且 $Pred(\gamma_2)>Pred(\gamma_1)$,那么经过$\gamma_1$ 的路径将比经过$\gamma_2$ 的路径更容易覆盖[17].鉴于此,选择经过占优语句$\gamma_i$的最短路径包含的谓词数量记为 $Pred(\gamma_i)$,反映$\gamma_i$的覆盖难度.

除了采用代码行数和谓词数量作为反映路径执行代价的指标之外,其他学者还采用了 NPATH[18]和 Halstead[19]等.但是,计算这些指标值需要分析和统计大量的程序信息,从而增加了额外的测试代价.因此,本节选用上述容易获取的两个指标.

### 2.2 最容易覆盖的占优语句(集)选择

本节基于上述 4 个指标,采用 Topsis 方法对各占优语句(集)的覆盖难度进行排序,并选择最容易覆盖的占优语句(集)作为新的目标语句(集).Topsis 方法是系统工程中一种常用的有限方案多目标决策方法,该方法对原始数据规范化处理后,消除不同指标量纲的影响;在统计各指标信息时注重数据的相对大小,能够综合反映各目标之间的差距,具有直观和可靠等优点[20].

利用 Topsis 方法,选择最容易覆盖的占优语句(集)的步骤如下:

(1) 计算占优语句(集)的评价向量

假设目标语句有 $n$ 个占优语句(集),对于每一占优语句(集),采用上述 4 个指标反映覆盖的难度,那么能够得到一个评价占优语句(集)覆盖难度的矩阵,记为 $A$,其元素 $a_{ij}$ 表示第 $i$ 个占优语句(集)关于第 $j$ 个评价指标的值,$i=1,2,\ldots,n$;$j=1,2,3,4$.如果决策者对不同指标的重视程度不同,那么可以通过这些指标的权值 $w_j$ 体现.对 $A$ 进行规范化,并记规范化之后的评价矩阵为 $\overline{A}=(\overline{a}_{ij})_{n\times 4}$,那么占优语句(集)$\gamma_i$对于指标 $j$ 的评价值为

$$\left.\begin{aligned} v_{ij}&=\overline{a}_{ij}w_j\\ \overline{a}_{ij}&=\frac{a_{ij}}{\sqrt{\sum_{i=1}^{n}a_{ij}^2}}\end{aligned}\right\} \tag{3}$$

从而,$\gamma_i$的评价向量为$\{v_{i1},v_{i2},v_{i3},v_{i4}\}$.

(2) 计算占优语句(集)的评价值与理想评价值的距离

为了说明占优语句(集)$\gamma_i$在所有占优语句(集)的评价中所处的位置,首先,根据这些占优语句(集)的评价值得到如下两个评价向量,分别是理想评价向量$x^+$和负理想评价向量$x^-$;然后,分别计算$v_i$与$x^+$和$x^-$的距离,记为$s_i^+$和$s_i^-$.这里,$x^+$和$x^-$是各评价指标的最优和最差值构成的向量:

$$\left.\begin{aligned} x^+ &= \{x_1^+, x_2^+, x_3^+, x_4^+\} \\ x^- &= \{x_1^-, x_2^-, x_3^-, x_4^-\} \\ s_i^+ &= \sqrt{\sum_{j=1}^{4}(v_{ij}-x_j^+)^2} \\ s_i^- &= \sqrt{\sum_{j=1}^{4}(v_{ij}-x_j^-)^2} \end{aligned}\right\} \tag{4}$$

其中,

$$x_j^+ = \min_{i\in\{1,2,\ldots,n\}} v_{ij},$$

$$x_j^- = \max_{i\in\{1,2,\ldots,n\}} v_{ij}.$$

(3) 选择最容易覆盖的占优语句(集)

记占优语句(集)$\gamma_i$的评价向量对理想评价向量$x^+$的相对接近度为$c_i$,那么$c_i$可以表示为

$$c_i = \frac{s_i^+}{s_i^+ + s_i^-} \tag{5}$$

容易看出,$c_i\in[0,1]$,且$c_i$越小,$\gamma_i$的覆盖难度就越低;特别地,如果$c_i$=0,那么$\gamma_i$是最容易覆盖的;当$c_i$=1时,$\gamma_i$最难覆盖.这说明,通过$c_1,c_2,\ldots,c_n$的值,能够对各占优语句(集)的覆盖难度进行排序.从上述排序中选择$c_i$最小的占优语句(集)作为新的目标语句.需要说明的是,如果对于所有的占优语句(集)上述指标的取值均相同,则说明这些占优语句(集)的覆盖难度相同,此时,从上述占优语句(集)中任选一个作为新的目标语句.

综上所述,本节给出的 4 个指标能够在一定程度上反映占优语句(集)覆盖的难度.在这些指标中,占优语句执行概率是一个综合指标,能够充分反映采用遗传算法生成测试数据付出的代价,因此是一个非常重要的评价指标,相应地赋予该指标的权值为 0.5.对于其他 3 个指标,考虑到其重要性难以区分,因此其权值大体相等.这样一来,关键变量个数、占优语句执行概率、最短路径包含的代码行数以及最短路径包含的谓词数量等指标的权值,分别为 0.2,0.5,0.15 以及 0.15.容易理解:对于相同的指标值,不同的指标权值选择的最容易覆盖的占优语句(集)也可能不同.本文仅根据专家经验[21]给出一组可能的权值,但是所给的权值并非最优的.事实上,确定各指标的最优权值已经超出了本文研究的范围.

### 2.3 例 子

本节利用图 2 的示例程序,说明所提出的方法选择新的目标语句的过程.

步骤 1. 寻找目标语句的占优语句(集).

在示例程序中,目标语句为语句 13,位于条件语句"if (*initialised*)"的真分支中.

为了覆盖该目标语句,"*initialized*=1"的语句,即语句 3、语句 7、语句 9 或语句 11 必须被执行.由于语句 5 为"*initialized*=0"的赋值语句,因此为了覆盖该目标语句,语句 3 被执行的同时,语句 5 一定不能被执行,记为{语句 3,!语句 5}.这样一来,{语句 3,!语句 5}、{语句 7}、{语句 9}和{语句 11}是目标语句的占优语句(集).鉴于语句 13 的占优语句(集)有 4 个,因此需要从这 4 个占优语句(集)中选择最容易覆盖的作为新的目标语句.

步骤 2. 计算占优语句(集)的评价向量.

以占优语句集$\gamma_1${语句 3,!语句 5}为例,说明其评价向量的计算方法.包含该占优语句集的惟一路径由语句 1、语句 2、语句 3、语句 4 以及语句 6 组成,因此,该路径就是包含该占优语句集的最短路径.该路径有 5 行代码、2 个谓词,语句 2 和语句 4 是关键语句,涉及 2 个输入变量,输入范围都是[0,255]之间的整数,且

$$p(\gamma_1)=\frac{1}{256\times 256}=1.5\times 10^{-5}.$$

同理,可以得到其他占优语句的各指标值,见表 1.

**Table 1** Evaluation indicators of dominant statement(s)

**表 1** 占优语句(集)的评价指标值

| | $Kvar(\gamma_i)$ | $p(\gamma_i)$ | $Loc(\gamma_i)$ | $Pred(\gamma_i)$ |
|---|---|---|---|---|
| {语句 3,!语句 5} | 2 | $1.5\times10^{-5}$ | 5 | 2 |
| {语句 7} | 1 | $3.9\times10^{-3}$ | 6 | 3 |
| {语句 9} | 2 | $3.6\times10^{-3}$ | 7 | 5 |
| {语句 11} | 2 | 0.219 | 8 | 7 |

因此,占优语句集{语句 3,!语句 5}、{语句 7}、{语句 9}和{语句 11}的评价向量分别为{0.1109,0.0000,0.0569,0.0322},{0.0555,0.0089,0.0682,0.0482},{0.1109,0.0082,0.0796,0.0804}和{0.1109,0.4999,0.0910,0.1126}.$x^+$={0.0555,0.4999,0.0569,0.0322},$x^-$={0.1109,0.0000,0.0910,0.1126},$s_1^+=0.5029, s_1^-=0.0873, s_2^+=0.4913, s_2^-=0.0884, s_3^+=0.4976, s_3^-=0.0351, s_4^+=0.1035, s_4^-=0.4998$.

步骤 3. 选择最容易覆盖的占优语句(集).

由于 $c_1$=0.8520,$c_2$=0.8447,$c_3$=0.9341,$c_4$=0.1715,因此第 4 个占优语句是最容易覆盖的占优语句,将其作为新的目标语句.

```
s    void flag(int a,int b)
{
1   int initialised=0;
2   if (a==20)
3      initialised=1;
4   if (a!=b)
5      initialised=0;
6   if (b==50)
7      initialised=1;
8   if (a>20 && b==100)
9      initialised=1;
10 if (a==5||b>200)
11     initialised=1;
12 if (initialised)
   {
13 //target statement
   }
e
}
```

Fig.2 Sample program

图 2 示例程序

## 3 实 验

考虑到本文方法的有效性难以从理论上证明,因此本节通过大量实验评价所提出方法的性能.为此,分别以各占优语句(集)作为目标语句,以生成测试数据需要的评价次数、耗时以及成功率等作为指标,反映不同语句覆盖的难度.显然,生成覆盖某占优语句(集)的测试数据需要的评价次数越少、耗时越短且成功率越高,该占优语句(集)就越容易覆盖.如果基于评价次数、耗时以及成功率等指标选择的最容易覆盖的占优语句(集)与采用本文方法选择的占优语句(集)相同,那么说明本文方法选择的占优语句(集)是有效的.

为了计算生成测试数据的成功率,分别以各占优语句(集)和原语句作为测试目标,独立运行多次,将生成期望测试数据的次数与总的运行次数的比值作为覆盖该语句(集)的成功率.此外,记录该方法多次运行的耗时以及所需要的评价次数.

实验采用的被测程序为 2 个基准程序和 3 个工业程序.

### 3.1 基准程序测试

本节的被测程序 Telephone number 和 Multiple flag2 来自文献[22],这些程序被广泛作为测试含有标志变量的基准程序,其中,程序 Telephone number 的输入变量是整型,其输入域为有限集;程序 Multiple flag2 的输入变量是实数型,其输入域为无限集.因此,本节选择这两个程序作为基准测试程序.

程序 Telephone number 包含 80 行代码、11 个分支,用于确定某电话号码是否是英国的.该程序的主函数实现一系列编码字符 *unicode_char* 的登记功能,仅有 1 个整型输入变量 *unicode_char*,此外还有一些中间变量.鉴于程序输入的范围为[0,65535]的整数,因此使用遗传算法生成测试数据时,种群的搜索空间包含 $2^{16}$ 个数据.由于程序代码较多,本节将给出代码简写,如图 3(a)所示.

程序 Multiple flag2 包含 12 行代码、5 个分支.该程序包含 2 个标志变量,分别是 *initialised* 和 *has_been_fired*,且只有这 2 个标志变量的取值同时为 1 时,目标语句才能被执行.该程序也包含 2 个输入变量 *a* 和 *b*,均为 double 型,且输入范围是[0,10000],精度为 0.01.因此,使用遗传算法生成测试数据时,种群的搜索空间包含 $10^6\times10^6$ 个数据.程序代码如图 3(b)所示.

```
void validate_uk_tel_no(int unicode_char)
{
  …    ;                                    //变量初始化
  if (unicode_char==12)
     {   ...;
     error=1;  }                            //占优语句1
else if (unicode_char>=48&&unicode_char<=57)
     {   ...;
     if (unicode_char==48&&unicode_char!=52)
     error=1;                               //占优语句2
     if (unicode_char==48)
     error=1;                               //占优语句3
     }
  else
     error=0;
  if (error)
     {
        …;                                  //目标语句
     }
}
```

(a)

```
void flag_multiple2(double a,double b)
  {
  int initialised = 0;
  int has_been_fired = 0;

  if (a == 0)
     initialised = 1;          //占优语句1

if (b == 0)
     has_been_fired = 1;   //占优语句2

  if (initialised)
  {
  if (b == 1)
     has_been_fired = 1;   //占优语句3
     if (has_been_fired)
        …;                 //目标语句
  }
}
```

(b)

Fig.3 Code of sample program

图 3 基准程序代码

基于条件语句相关性程序自动寻找 2 个基准程序目标语句的占优语句(集),我们发现,程序 Telephone number 的目标语句有 3 个占优语句,分别是占优语句 1、占优语句 2 以及占优语句 3;程序 Multiple flag2 的目标语句有 2 个占优语句集,分别是$\alpha_1$和$\alpha_2$.$\alpha_1$包含占优语句 1 和占优语句 2,$\alpha_2$包含占优语句 1 和占优语句 3.采用可测试性转化方法生成覆盖目标语句的测试数据时,均存在占优语句(集)的选择问题.

首先分析源程序代码[23],计算并获得的各占优语句(集)所具有的指标值,见表 2.

**Table 2** Indicators of sample programs

**表 2** 基准程序的指标值

| | | $Kvar(\gamma_i)$ | $p(\gamma_i)$ | $Loc(\gamma_i)$ | $Pred(\gamma_i)$ | $c_i$ |
|---|---|---|---|---|---|---|
| Telephone number | 占优语句 1 | 1 | $1.5\times10^{-5}$ | 24 | 7 | 0 |
| | 占优语句 2 | 1 | $1.5\times10^{-5}$ | 28 | 16 | 0.979 3 |
| | 占优语句 3 | 1 | $1.5\times10^{-5}$ | 29 | 14 | 0.779 4 |
| Multiple flag2 | 占优语句集$\alpha_1$ | 2 | $10^{-12}$ | 6 | 2 | 0 |
| | 占优语句集$\alpha_2$ | 2 | $10^{-12}$ | 7 | 3 | 1 |

采用本文的方法,程序 Telephone number 和 Multiple flag2 的目标语句的最容易覆盖占优语句(集),分别是占优语句 1 和占优语句集$\alpha_1$.

其次,分别以各占优语句和原目标语句作为测试目标,采用遗传算法生成测试数据,运行该方法 50 次作为一组,共运行 10 组,记录每组实验生成测试数据的耗时和所需要的评价次数以及成功率,并求取上述指标的平均值.程序 Telephone number 的实验结果见表 3,程序 Multiple flag2 的实验结果见表 4.

**Table 3** Experimental results of Telephone number program

**表 3** 程序 Telephone number 的实验结果

| | $c_i$ | 评价次数 | 耗时(s) | 成功率(%) |
|---|---|---|---|---|
| 占优语句 1 | 0 | 1 047 | 0.156 | 96 |
| 占优语句 2 | 0.979 3 | 5 000 | 1.11 | 0 |
| 占优语句 3 | 0.779 4 | 5 000 | 1.125 | 0 |
| 目标语句 | – | 2 065 | 0.391 | 68 |

**Table 4** Experimental results of Multiple flag2 program

**表 4** 程序 Multiple flag2 的实验结果

| | $c_i$ | 评价次数 | 耗时(s) | 成功率(%) |
|---|---|---|---|---|
| 占优语句集$\alpha_1$ | 0 | 6 250 | 1.953 | 100 |
| 占优语句集$\alpha_2$ | 1 | 7 921 | 2.359 | 100 |
| 目标语句 | – | 50 000 | 22.38 | 0 |

由表 3 可知:(1) 与原目标语句相比,生成覆盖占优语句 1 的测试数据需要的评价次数和耗时均较低,其中,评价次数仅是原目标语句的 51%,耗时仅是原目标语句的 40%,成功率比原目标语句提高了 28%.这说明通过基于占优关系的可测试性转化,有可能提高测试数据生成效率.(2) 与原目标语句相比,生成覆盖占优语句 2 和语句 3 的测试数据需要的评价次数较高,到了最大评价次数也没有找到期望的测试数据,相应地,耗时也多于原目标语句.这说明即使采用基于占优关系的可测试性转化得到新的目标语句,生成测试数据的效率也可能降低,也即从多个占优语句(集)中,选择合适的作为新的目标语句是非常有必要的.(3) 与占优语句 2 和语句 3 相比,生成覆盖语句 1 的测试数据无论是需要的评价次数还是耗时均较低,更重要的是,能够成功地找到覆盖语句 1 的测试数据.这说明采用本文方法选择的占优语句是最容易覆盖的,也即本文提出的基于覆盖难度的占优语句(集)选择方法是有效的.

由表 4 可知:(1) 达到最大评价次数也没有生成覆盖目标语句的测试数据,相应的耗时为 22.38s.这说明如果不采用基于占优关系的可测试性转化,那么生成覆盖目标语句的测试数据的效率将很低.(2) 与目标语句相比,生成覆盖占优语句集$\alpha_1$ 和$\alpha_2$ 的测试数据需要的评价次数和耗时均较低,其中,评价次数仅分别是原目标语句的 13%和 16%,耗时仅分别是原目标语句的 8%和 11%,且均能生成覆盖占优语句集$\alpha_1$ 和$\alpha_2$ 的测试数据.这说明采用基于占优关系的可测试性转化,能够提高测试数据生成效率.(3) 与占优语句集$\alpha_2$ 相比,生成覆盖占优语句集$\alpha_1$ 的测试数据无论是评价次数还是耗时均较低,其中,评价次数仅是占优语句集$\alpha_2$ 的 79%,耗时仅是占优语句集$\alpha_2$ 的 83%.这说明采用本文方法选择的占优语句集,的确是最容易覆盖的.

### 3.2 工业程序测试

从文献[8]给出的工业程序中选择 3 个作为被测程序.这些程序包含的代码行、谓词数量以及标志变量个数见表 5.

**Table 5** Basic information of industrial programs

**表 5** 工业程序基本信息

| | 代码行 | 谓词数量 | 标志变量个数 |
|---|---|---|---|
| acct6.3.2_sa | 1 665 | 88 | 14 |
| gnugo3.8_board | 4 331 | 474 | 46 |
| wdiff-0.5_wdiff | 1 333 | 157 | 18 |

程序 acct6.3.2_sa.c 是 GNU 系统调用、禁止和启用系统以及记录进程信息功能单元的主函数.该函数首先对调入的指令进行解析,根据解析的结果分别赋予不同的标志变量值;然后,根据各标志变量的赋值执行相应的指令.在该函数中,当标志变量 *print_users* 和 *merge_files*,或者 *print_users* 和 *user_summary_flag* 的取值都为 1 时,目标语句才能被执行.通过分析发现,目标语句有 4 个占优语句集.

gnugo3.8_board.c 是 GNU 系统开发的围棋对战程序的子函数,用于识别围棋的局势和更新棋子的位置.本节测试的 *do_play_move* 子函数需要分析棋局成气的形式,为接下来的下棋做准备.标志变量 *neighbor_allies* 有上、下、左、右这 4 个方向的选择,目标语句的占优语句集有 4 个.

wdiff-0.5_wdiff.c 是 GNU 系统比较两个文件的主函数,该函数通过前端的 diff 比较文件的每一个字符.首先创建 2 个临时文件,每一行中间没有空格;然后执行 diff 程序,通过收集 diff 文件的输出结果产生一个显示差异的文档.该函数的 *decode_directive_line* 子函数作为被测程序,有 2 个输入变量 *state* 和 *character*,1 个 *error* 标志变量,其目标语句有 3 个占优语句集.

采用本文方法得到各占优语句(集)的相对接近度 $c_i$.采用遗传算法,生成覆盖目标语句和各占优语句(集)的测试数据,需要的评价指标见表 6.由于 wdiff-0.5_wdiff.c 的目标语句仅有 3 个占优语句集,因此在表 6 的第 4 行没有相应的数据.此外,不计算目标语句的相对接近度 $c_i$ 值,因此相应的数据也为空.

**Table 6** Experimental results of industrial programs

**表 6** 工业程序实验结果

| | acct6.3.2_sa.c | | | | gnugo3.8_board.c | | | | wdiff-0.5_wdiff.c | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | $c_i$ | 评价次数 | 耗时(s) | 成功率(%) | $c_i$ | 评价次数 | 耗时(s) | 成功率(%) | $c_i$ | 评价次数 | 耗时(s) | 成功率(%) |
| 占优语句集 1 | 0 | 1 316 | 12.39 | 100 | 0 | 1 404 | 12.296 | 98 | 0.116 3 | 563 | 7.64 | 100 |
| 占优语句集 2 | 0 | 1 296 | 12.015 | 100 | 0.334 7 | 1 787 | 13.490 | 96 | 0.337 6 | 1 756 | 13.875 | 96 |
| 占优语句集 3 | 0 | 1 305 | 12.281 | 100 | 0.667 0 | 1 898 | 13.785 | 96 | 0.883 0 | 2 236 | 16.312 | 94 |
| 占优语句集 4 | 0 | 1 313 | 12.473 | 100 | 1.00 0 | 1 994 | 14.125 | 96 | – | – | – | – |
| 目标语句 | – | 3 256 | 35.187 | 50 | – | 35 326 | 80.187 | 26 | – | 25 233 | 60.285 | 40 |

由表 6 可知:

(1) 对于程序 acct6.3.2_sa,本文方法得到的各占优语句集的相对接近度 $c_i$ 值均为 0,说明这些占优语句集的覆盖难度相同.从被测程序中可知,占优语句均位于 case 条件分支下.这使得各个占优语句集作为新的目标语句集的代价是一致的,这从遗传算法生成测试数据的评价次数和耗时接近以及成功率相同可以得到反映.此时,可以选择任一占优语句集作为新的目标语句.此外,与目标语句相比,生成覆盖占优语句集 1 的测试数据需要的评价次数仅为原目标语句的 40%,耗时仅为原目标语句的 35.2%,成功率也提高了 50%.这表明采用基于占优关系的可测试性转化方法,能够提高测试数据生成的效率.

(2) 对于程序 gnugo3.8_board,本文方法得到的各占优语句集的相对接近度 $c_i$ 值中,占优语句集 1 最小,其次是占优语句集 2 和占优语句集 3,最大的是占优语句集 4.这说明,这些占优语句集的覆盖难度从易到难依次是占优语句集 1、占优语句集 2、占优语句集 3 以及 4;相应地,生成覆盖这些占优语句集需要的评价次数分别为 1 404,1 787,1 898 以及 1 994,耗时分别为 12.296s,13.490s,13.785s 以及 14.125s.从评价次数和耗时等指标容易看出,采用本文方法选择的占优语句集具有最少的评价次数和耗时,因此最容易覆盖.这说明采用本文方法,选择最容易覆盖的占优语句集是有效的.此外,与目标语句相比,生成覆盖占优语句集 1 的测试数据评价次数和耗时仅分别为原目标语句的 4%和 15.3%,成功率提高了 72%.这进一步验证了采用基于占优关系的可测试性转化方法,能够提高测试数据生成的效率.

(3) 对于程序 wdiff-0.5_wdiff,可以得到与程序 gnugo3.8_board 类似的结果,限于篇幅,在此不再赘述.由这些结果容易看出:采用本文方法得到的占优语句集,的确是最容易覆盖的;且采用基于占优关系的可测试性转化方法,确实能够提高测试数据生成效率.

通过上述基准与工业程序的测试结果与分析,能够得到如下结论:(1) 尽管基于占优关系的可测试性转化方法能够将覆盖原目标语句的问题转化为覆盖占优语句(集)的问题,但是如果选择任一占优语句(集)作为新的

目标语句,那么生成测试数据的效率有可能降低,从而说明采用合适的方法选择占优语句(集)作为新的目标语句是非常有必要的;(2) 采用本文方法能够得到最容易覆盖的占优语句(集),从而提高测试数据生成的效率.

## 4 总 结

标志变量问题在很多程序中普遍存在,尽管采用基于占优关系的可测试性转化方法能够在一定程度上解决标志变量问题,但是当目标语句的占优语句(集)不止一个时,选择的占优语句(集)作为新的目标语句将会影响测试数据生成的效率.因此,研究合适的占优语句(集)选择方法对于提高测试数据生成效率是非常有帮助的.

本文研究占优语句(集)的选择问题,提出一种基于覆盖难度的占优语句(集)选择方法.该方法通过关键变量个数、占优语句执行概率、最短路径包含的代码行数以及最短路径包含的谓词数量等指标,反映占优语句(集)覆盖的难度,并采用 Topsis 方法计算占优语句(集)的相对接近度值,选择该值最小的作为最容易覆盖的占优语句(集).将该方法应用于多个基准与工业程序的测试,实验结果表明:采用遗传算法生成覆盖本文方法选择的占优语句(集)需要的评价次数和耗时少,成功率高.因此,本文方法能够得到最容易覆盖的占优语句(集).

需要说明的是,本文所提出方法的有效性仅通过若干基准和工业程序验证,还没有应用于大量的工业程序,因此,该方法在更多、更复杂程序上的应用是需要进一步考虑的问题.除了本文考虑的指标以外,还有其他指标能够不同程度地反映语句的覆盖难度,采用不同的指标,得到的最容易覆盖的占优语句(集)也会有很大差异.因此,融合其他指标的最容易覆盖占优语句(集)的选择,也是今后研究的课题.此外,基于本文提出的方法开发自动选择最容易覆盖占优语句(集)的系统,也是需要研究的问题.

**References**:


[1] Xu RZ. Software Reliability Engineering. Beijing: Tsinghua University Press, 2007 (in Chinese).

[2] Sagarna R, Yao X. Handling constraints for search based software test data generation. In: Lars F, Mercedes GM, Manuel N, eds. Proc. of the IEEE Int'l Conf. on Software Testing, Verification, and Validation Workshop. Washington: IEEE-Computer Society, 2008. 232−240. [doi: 10.1109/ICSTW.2008.19]

[3] Harman M, Hu L, Hierons R, Wegener J, Sthamer H, Baresel A, Roper M. Testability transformation. IEEE Trans. on Software Engineering, 2004,30(1):3−16. [doi: 10.1109/TSE.2004.1265732]

[4] Harman M, Hu L, Hierons R, Baresel A, Sthamer H. Improving evolutionary testing by flag removal. In: Langdon WB, ed. Proc. of the Genetic and Evolutionary Computation Conf. Amsterdam: Elsevier Science and Technology, 2002. 1359−1366.

[5] Yao XJ, Gong DW. Testability transformation based on dominant relationship of target statements. Acta Electronic Sinica, 2013,41(12):2523−2528 (in Chinese with English abstract). [doi: 10.3969/j.issn.0372-2112.2013.12.033]

[6] Lammermann F, Baresel A, Wegener J. Evaluating evolutionary testability for structure-oriented testing with software measurements. Applied Soft Computing, 2008,8(2):1018−1028. [doi: 10.1016/j.asoc.2006.06.010]

[7] Wappler S, Wegener J, Baresel A. Evolutionary testing of software with function-assigned flags. The Journal of Systems and Software, 2009,82(11):1767−1779. [doi: 10.1016/j.jss.2009.06.037]

[8] Binkley DW, Harman M, Lakhotia K. FlagRemover: A testability transformation for transforming loop assigned flags. ACM Trans. on Software Engineering and Methodology, 2011,20(3):1−33. [doi: 10.1145/2000791.2000796]

[9] Jiang S, Lu Y. Evolutionary testing of unstructured programs using a testability transformation approach. In: Yu ZB, ed. Proc. of the Japan-China Joint Workshop on Frontier of Computer Science and Technology. Washington: IEEE-Computer Society, 2008. 59−66. [doi: 10.1109/FCST.2008.21]

[10] McMinn P, Binkley D, Harman M. Empirical evaluation of a nesting testability transformation for evolutionary testing. ACM Trans. on Software Engineering and Methodology, 2009,18(3):1−27. [doi: 10.1145/1525880.1525884]

[11] Arcuri A. Full theoretical runtime analysis of alternating variable method on the triangle classification problem. In: Massimiliano DP, Simon P, eds. Proc. of the Int'l Symp. on Search Based Software Engineering. Washington: IEEE-Computer Society, 2009. 113−121. [doi: 10.1109/SSBSE.2009.16]

[12] McMinn P, Harman M, Lakhotia K, Hassoun Y, Wegener J. Input domain reduction through irrelevant variable removal and its effect on local, global, and hybrid search based structural test data generation. IEEE Trans. on Software Engineering, 2012,38(2): 453−477. [doi: 10.1109/TSE.2011.18]

[13] Korel B. Automated software test data generation. IEEE Trans. on Software Engineering, 1990,16(8):870−879. [doi: 10.1109/32.57624]

[14] Bottaci L. Predicate expression cost functions to guide evolutionary search for test data. In: Cantú-Paz E, Foster JA, Deb K, *et al*., eds. Proc. of the Genetic and Evolutionary Computation Conf. Heidelberg: Springer-Verlag, 2003. 2455−2464. [doi: 10.1007/3-540-45110-2_149]

[15] Cheng SH. MeasureTheory and Probability Theory. Beijing: Peking University Press, 2007. 23−56 (in Chinese).

[16] Debbarma MK, Tiwari S, Misra AK. Efficient path selection strategy based on static analysis for regression testing. Int'l Journal of Computer Theory and Engineering, 2013,5(2):248−252. [doi: 10.7763/IJCTE.2013.V5.687]

[17] Debbarma MK, Kar N, Saha A. Static and dynamic software metrics complexity analysis in regression testing. In: Sambasivam SH, ed. Proc. of the Int'l Conf. on Computer Communication and Informatics. Washington: IEEE-Computer Society, 2012. 1−6. [doi: 10.1109/ICCCI.2012.6158825]

[18] Nejmeh BA. NPATH: A measure of execution path complexity and its applications. Communications of the ACM, 1988,31(2): 188−210. [doi: 10.1145/42372.42379]

[19] Halstead MH. Elements of Software Science. Amsterdam: Elsevier Science Ltd., 1977.

[20] Deng H, Yeh CH, Willis RJ. Inter-Company comparison using modified TOPSIS with objective weights. Computers and Operations Research, 2000,27(10):963−973. [doi: 10.1016/S0305-0548(99)00069-6]

[21] Liu YL, Feng DG, Wu LH, Lian YF. Performance evaluation of worm attack and defense strategies based on static Bayesian game. Ruan Jian Xue Bao/Journal of Software, 2012,23(3):712−723 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/3997.htm [doi: 10.3724/SP.J.1001.2012.03997]

[22] McMinn P. Evolutionary search for test data in the presence of state behaviour [Ph.D. Thesis]. Sheffield: The University of Sheffield, 2005.

[23] Code visual to flowchart 6.0. 2011. http://code-visual-to-flowchart.soft32.com/

**附中文参考文献**:

[1] 徐仁佐.软件可靠性工程.北京:清华大学出版社,2007.

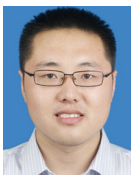
[5] 姚香娟,巩敦卫.基于目标语句占优关系的软件可测试性转化.电子学报,2013,41(12):2523−2528. [doi: 10.3969/j.issn.0372-2112.2013.12.033]

[15] 程士宏.测度论与概率论基础.北京:北京大学出版社,2004.23−56.

[21] 刘玉玲,冯登国,吴丽辉,连一峰.基于静态贝叶斯博弈的蠕虫供方策略政绩评估.软件学报,2012,23(3):712−723. http://www.jos.org.cn/1000-9825/3997.htm [doi: 10.3724/SP.J.1001.2012.03997]

**巩敦卫**(1970－),男,江苏徐州人,博士,教授,博士生导师,CCF 会员,主要研究领域为基于搜索的软件工程,智能优化与控制.

**姚香娟**(1975－),女,博士,副教授,主要研究领域为基于搜索的软件工程.



**钟超群**(1988－),男,硕士,主要研究领域为基于搜索的软件工程.