一种变异测试中冗余变异体的寻找方法

钱莨南[1,2,3],王雅文[1,2,3],宫云战[1,2,3],孟凡荣[4]
(1.北京邮电大学网络与交换技术国家重点实验室,北京 100876;2.桂林电子科技大学广西云计算与大数据协同创新中心,广西桂林 541004;3.桂林电子科技大学广西高校云计算与复杂系统重点实验室,广西桂林 541004;4.长春汽车工业高等专科学校,吉林长春 130013)

摘 要: 变异测试是一种有效的基于故障的测试方法,但大量冗余变异体所带来的昂贵的测试成本问题,阻碍了它在实际工程开发中的应用.为解决该问题,本文针对程序中的顺序语句所产生的变异体,基于故障的可达-感染-传播模型,提出了使用区间抽象域来表示程序状态,通过区间运算判断变异体之间冗余关系的算法;针对程序中的条件语句,基于谓词故障层级,分别给出了面向简单谓词和复合谓词的冗余变异体选择算法.并对这两种算法对冗余变异体的判定效果进行了分析,最后给出了在分层抽样背景下,非冗余变异体生成的约束边界条件.对Siemens和开源项目等共8个工程进行了实验,并与随机选择法进行了对比.结果表明,本文所提方法在减少变异测试时间成本的同时,可以保持较高的变异得分.

关键词: 变异测试;变异算子;冗余变异体;变异成本;变异体约减
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2017)08-1970-06
电子学报URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.08.023

A Method for Finding Redundant Mutants in Mutation Testing

QIAN Gen-nan[1,2,3], WANG Ya-wen[1,2,3], GONG Yun-zhan[1,2,3], MENG Fan-rong[4]
(1. *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing* 100876, *China*; 2. *Guangxi Cooperative Innovation Center of Cloud Computing and Big Data, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*; 3. *Guangxi Colleges and Universities Key Laboratory of Cloud Computing and Complex Systems, Guilin University of Electronic Technology, Guilin, Guangxi* 541004, *China*; 4. *Changchun Automobile Industry Institute, Changchun, Jilin* 130013, *China*)

Abstract: Mutation testing is an effective fault-based testing method. However, the application of mutation testing in engineering development has been restricted by the high testing costs caused by a large number of redundant mutants. Regarding the mutants arising from the sequential statements in a program, an algorithm based on propagation-infection-execution (PIE) model was proposed, which employs the interval abstract domain to represent program state and the interval algorithm to evaluate the redundancy relation between the mutants. Meanwhile, regarding the conditional statements in a program, the redundant mutant selection algorithms based on the predicate fault hierarchy are also presented. The algorithms are designed for simple predicate and compound predicate respectively. By analyzing the effects of these algorithms, the constrained boundary condition for the development of non-redundant mutants under the condition of stratified sampling is concluded. Siemens Test Suite and other three open source projects are used to conduct experiments to compare the proposed method with random selection method. Experimental results show that the proposed method can reduce the mutant testing time cost while maintaining a high mutation score.

Key words: mutation testing; mutation operators; redundant mutants; mutation cost; mutant reduction

1 引言

变异测试[1]是一种基于故障的软件测试技术,满足变异测试准则的用例集具有更高的检错能力.但数目庞大的变异体所带来的测试成本问题,严重限制了其在实际工程中的应用[2].

收稿日期:2016-10-26;修回日期:2017-01-16;责任编辑:马兰英
基金项目:国家自然科学基金(No.91318301,No.61202080);广西云计算与大数据协同创新中心、广西高校云计算与复杂系统重点实验室资助(No.YD16508)

针对该问题,国内外学者提出了变异样本、变异选择等方法,但精度有限或者严重依赖测试环境,适用度低. 近来在研究中发现,变异体之间并不是平等关系,而是存在包含关系,并且冗余变异体是广泛存在的. 因此将降低变异测试成本问题,转化为在尽量保持变异得分不变的前提下,在全体变异体集合中寻找一个非冗余变异体子集的问题. 新近提出了基于程序分析的冗余变异体寻找方法. 具体有 Just 的占优分析法[3]、Namin 的基于程序特征法[4]、Kurtz 的感染分析法[5]等方法. 为解决变异测试昂贵的测试成本问题,本文基于程序分析法,提出了一种冗余变异体识别技术. 从变异体全集中发现并剔除冗余的变异体,保留数目较少的变异体进行变异测试,以此降低整体的测试成本. 并对所提方法对冗余变异体的判定效果进行了分析,最后给出了在分层抽样背景下,非冗余变异体生成的约束边界条件. 和文献[3,4]不同,本文提出的方法并不需要提前执行所有的变异体;和文献[5]等挑选粒度为变异算子级别的方法相比,本文是更细致的变异体级别,可更好的在实际工程开发中应用.

2　问题描述

2.1　问题描述

定义 1　变异体的包含关系

对被测程序 P,输入 i 是一个 n 元组,$i=\{i_1,i_2,\cdots i_n\}$. 所有可能的 i 构成了 P 的输入空间 I. m_1 和 m_2 是 P 生成的两个变异体. P 在输入 i 时的输出状态为 $O(P,i)$,变异体 m 在输入 i 时的输出状态为 $O(m,i)$. 称变异体 m_1 包含 m_2,当且仅当:

$$(1)\ \exists i\in I, O(P,i)\neq O(m_1,i)\wedge O(P,i)\neq O(m_2,i)$$

$$(2)\ I_\Delta(m_1)\subseteq I_\Delta(m_2),\quad I_\Delta(m)=\{i\in I\mid O(P,i)\neq O(m,i)\}\tag{1}$$

记为 $m_1\mapsto m_2$.

定义 2　冗余变异体

当存在包含关系 $m_1\mapsto m_2$ 时,称 m_2 为冗余变异体. 对两个由变异体组成的集合 M_1 和 M_2,如果对 $\forall m_k\in M_2$,$\exists m_t\in M_1$,使得 $m_t\mapsto m_k$,则称 M_2 为冗余变异体集合,M_1 为非冗余变异体集合. 当 M_2 为变异体全集时,M_1 称为充分非冗余变异体集合. 我们的目标就是寻找非冗余变异体集合.

3　冗余变异体寻找算法

3.1　基于区间运算的冗余变异体寻找算法

针对顺序执行语句中所产生的变异体,利用数值区间作为程序状态的一种抽象,使用约束求解获得具体区间范围,再比较不同变异体感染区间的范围,来判断是否存在冗余变异体.

根据 PIE 理论[6],如果一个变异体被检测出来,必须满足三个必要条件:用例执行到变异语句(Execution);变异语句状态发生变化(Infection)以及这种状态可以传播到程序出口(Propagation). 当用例执行完变异语句之后,程序状态改变的程度越大,这种感染状态越有可能传播到程序出口,也就使得程序的最终输出和源程序不一致,即该变异体有更大可能性被检测出来.

定义 3　区间运算

区间运算最早由 Moore 提出[7],本文主要是指在程序分析中使用区间运算. 如果程序中的变量 x 的取值范围满足 $\underline{x}\leqslant x\leqslant\bar{x}$,$\underline{x}$、$\bar{x}$ 为实数,则变量 x 的区间记为 $x=[\underline{x},\bar{x}]$. 对两个数值型程序变量 X,Y,定义其基本的区间运算为:

$$X+Y=[\underline{x}+\underline{y},\bar{x}+\bar{y}];$$

$$X-Y=[\underline{x}-\bar{y},\bar{x}-\underline{y}];$$

$$X*Y=[\min(\underline{xy},\underline{x}\bar{y},\bar{x}\,\underline{y},\bar{xy}),\max(\underline{xy},\underline{x}\bar{y},\bar{x}\,\underline{y},\bar{xy})];$$

$$X/Y=[\underline{x},\bar{x}]*[1/\bar{y},1/\underline{y}],\text{若}\ 0\notin[\underline{y},\bar{y}]\tag{2}$$

更详细的区间运算规则,可见[8]

定义 4　控制流图及路径

控制流图是有唯一入口节点和唯一出口结点的有向图,是程序控制结构的一种表示方法. 通常用四元组的方式表示:$G=\{N,E,i,o\}$,其中 N 是结点的集合,E 是边的集合,i 与 o 分别表示入口和出口结点. 路径(Path)是指在控制流图上的一个节点序列,$p=\{n_1,n_2,\cdots n_q\}$,对所有的 r,$1\leqslant r<q$,都有 $(n_r,n_{r+1})\in E$.

定义 5　可达区间(Reachability Interval)

存在一条从控制流图入口节点到变异语句所在节点 n_m 的路径 p,其上包含的约束集合为 Con,满足这个约束集合的程序变量取值区间称为可达区间.

定义 6　感染区间(Infection Interval)

指程序变量在变异体所在语句的控制流图节点 n_m 上的一个取值区间,该区间满足相关变量在其中取任意值,均可使原程序状态和变异后的程序状态不相同. 即 $\forall x_i\in D_{\inf}, P(n,x_i)\neq P(n_m,x_i)$,其中 $D_{\inf}$ 是感染区间,$P(n,x_i)$ 指在控制流图上 n 点变量以 x_i 执行后的程序状态. 在同一语句位置生成的两个变异体的感染区间,可使用文氏图表示,如图 1 所示. 最外方框代表该语句位置的可达区间,D_{m1} 和 D_{m2} 分别代表两个变异体的感染区间.

情况 1　$D_{m2}\subseteq D_{m1}$ 此时可以检测 m_2 的用例也可以检测 m_1. m_1 相对 m_2 是冗余的.

情况 2　$D_{m2}\not\subset D_{m1}\wedge D_{m2}\cap D_{m1}\neq\varnothing$ 此时两个变异体的感染区间存在交集,但不存在包含关系. 此时选择感染区间相对小的变异体.

情况 3　$D_{m2}\cap D_{m1}=\varnothing$ 此时两个变异体的感染区间不存在交集,即相互不存在包含关系,同时选择这两

个变异体.

具体算法步骤如下:

(1)对源程序 P 进行预处理,生成控制流图和路径集,完成变量符号化和区间初始化;

(2)当路径集不为空时,选择一条路径,逐节点遍历路径收集可达约束;

(3)对当前节点进行可达约束求解,计算可达区间 D 和各变异体的感染区间;

(4)按上述的三种情况,判断冗余算子;

(5)当所有路径上的节点遍历完毕后,给出冗余算子集,否则转入步骤(2).

下面对所提方法的识别效果进行分析.

如果将程序变量在某一语句处的取值空间看作全集,各个变异体的感染区间则是它的子集,寻找非冗余变异体集合可以看作求最小集合覆盖的过程.

下面进行识别效果分析:设变量 x 在语句 l 处的取值空间 C 由 n 个数值元素组成,即 $C=\{c_1,\cdots,c_n\}$,每一个变异体的感染区间 D 是 C 的一个子集,则非冗余变异集合 $R=\{D_1,\cdots,D_m\}$,当寻找到有最小基数的 R 时(即 R 是 C 的最小覆盖集),本方法达到最佳效果的上限.

将最小非冗余变异体集合 $R_{\min}$ 的基数设为 k,即 $|R_{\min}|=k$;令 C_t 是经过 i 次的选择后,C 中仍未被覆盖的元素集合,可见有 $C_0=C$. 并且会在 $i+1$ 次时,选择最大的 C_t. 此时这个 C_t 必须覆盖至少 $|C_t|/k$ 个元素. 本算法完成最后一步的选择,即 $|C_t|<1$ 时,有

$$\left(1-\frac{1}{k}\right)^t<\frac{1}{n}\Rightarrow n<\left(\frac{k}{k-1}\right)^t$$
$$\Rightarrow \ln n\leqslant t\ln\left(1+\frac{1}{k-1}\right)^t\Rightarrow \frac{t}{k}\leqslant\ln n \tag{3}$$

可见,本方法选出的非冗余变异体的个数 t,其与最优解的比不会超过 $\ln n$,n 为全集的元素个数.

3.2 面向程序谓词表达式的冗余变异体寻找方法

在结构化的程序语言中,谓词表达式是指决定程序分支走向的判断语句. 下面给出相关定义:

定义 7 简单谓词表达式

简单谓词表达式是由布尔变量和关系运算符组成的表达式,布尔变量是值为真或假的程序变量. 形式为:$E_{spe}=a\oplus b$,其中 a,b 是布尔变量,$\oplus$ 是关系运算符集合,$\oplus=\{>,\geqslant,=,\neq,<,\leqslant\}$.

定义 8 复合谓词表达式

复合谓词表达式由 2 个及以上的简单谓词表达式或布尔变量通过逻辑运算符及括号连接而成. 形式为:$E_{cpe}=a\otimes b\otimes c\cdots$,其中 a,b,c 是布尔变量或简单谓词表达式,并可以使用括号将其隔开,$\otimes$ 是逻辑运算符集合,$\otimes=\{\&\&,\|,!\}$. 如 $a\geqslant b,a!=b$ 是简单型,$(a>c)\&\&(b<d)$ 是复合型.

3.2.1 简单谓词表达式冗余变异体寻找算法

一个测试用例执行通过一个简单谓词表达式后,接下来这个用例会沿程序的真分支或假分支继续执行. 但对一些简单谓词表达式所生成的变异体,测试用例执行完毕后,程序走向必然会发生变化,这样的变异体即为冗余变异体.

下面给出简单谓词表达式中冗余变异体[9]. 测试用例 t 在变异语句处的取值情况共有三种情况,用区间表示为 $I_a<I_b$ 等. 表 1 是 > 运算符产生的变异体及其后继真假分支走向情况.

表 1 关系运算符 > 相关变异体在用例执行后的程序走向

用例区间关系	源程序	变异体及真假分支情况						
	if($a>b$)	$a\geqslant b$	$a!=b$	$a<b$	$a\leqslant b$	$a==b$	True	False
$I_a<I_b$	*F*	*F*	[*T*]	[*T*]	[*T*]	*F*	[*T*]	*F*
$I_a=I_b$	*F*	[*T*]	*F*	*F*	[*T*]	[*T*]	[*T*]	*F*
$I_a>I_b$	*T*	*T*	*T*	[*F*]	[*F*]	[*F*]	*T*	[*F*]

针对简单谓词表达式,其非冗余变异体集合如下式所示:

$$O(>=)\Rightarrow M(>,==,true);O(>)\Rightarrow M(>=,!=,false)$$
$$O(==)\Rightarrow M(<=,>=,true);O(!=)\Rightarrow M(<,>,true)$$
$$O(<=)\Rightarrow M(<,==,true);O(<)\Rightarrow M(<=,!=,false) \tag{4}$$

3.2.2 复合谓词表达式冗余变异体寻找算法

本文结合[10]和[11],图 2 给出了面向通用类型的复合谓词表达式的故障层级.

其中的 Level 3 → Level 4 的证明见[10],其余的证明可见[11].

复合谓词表达式冗余变异体寻找算法具体步骤如下:

(1)对源程序 P 进行预处理,生成抽象语法树(AST,Abstract Syntax Tree);

(2)对 AST 进行遍历,如果当前节点类型为 LogicalExpression,则提取以其为根节点的复合谓词表达式;否则进入下一节点;

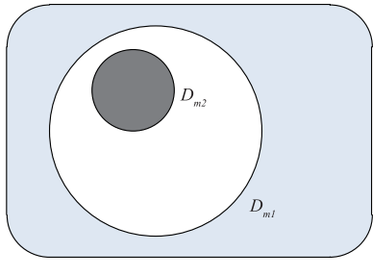
(3)遍历复合谓词的 AST 子树. 如果满足故障层级 Level 1 的生成条件,生成 Level 1 故障,否则进入 Level 2,生成该层故障,直至 Level 5,退出;

(4)所有 AST 节点遍历完毕后,给出生成的变异体集合;否则进入(2).

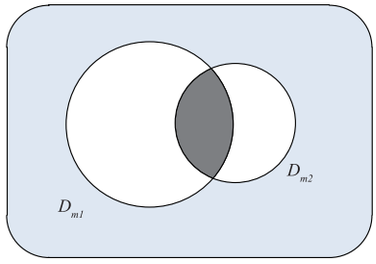
下面对面向程序谓词表达式的冗余变异体识别效果进行分析.

设谓词表达式具有形式:$E=n_1\otimes n_2\otimes n_3\cdots n_m$,即共有 m 个子表达式构成. E 中共包含 n 个唯一谓词变量,每个子表达式 n_i 中含有 a_i 个谓词变量. 则每种故障可

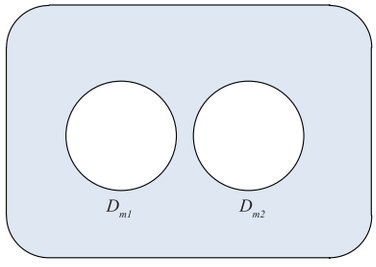
能产生的变异体个数如图 2 所示. 需要注意 ASF 型故障和实际程序的具体情况有关,这里设为变量 Δ.



(*a*)



(*b*)



(*c*)

图1　变异体感染区间的关系

缩写	描述	可产生变异体个数	缩写	描述	可产生变异体个数
CCF	添加并操作	$\sum_{i=1}^{m}(n-a_i)$	VRF	将变量替换为同表达式其余变量	$\sum_{i=1}^{m}a_i(n-a_i)$
CDF	添加或操作	$\sum_{i=1}^{m}(n-a_i)$	ORF+	或操作替换为并操作	$m-1$
ASF	使用括号改变运算优先级	Δ	ORF.	并操作替换为或操作	$\sum_{i=1}^{m}a_i-m$
SA0	将变量用False替换	$\sum_{i=1}^{m}a_i$	MVF	变量丢失	$\sum_{i=1}^{m}a_i$
SA1	将变量用True替换	$\sum_{i=1}^{m}a_i$	VNF	变量值取反	$\sum_{i=1}^{m}a_i$
故障层级			ENF	表达式取反	1

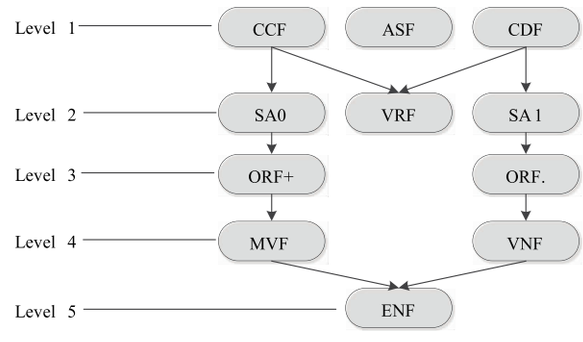


图2　复合谓词表达式的故障层级

下面给出面向程序谓词表达式的冗余变异体寻找方法的检测效果. 假设程序可以生成所有层级的故障,检测效果 P 定义为具有高检测能力的 Level 1 的变异体个数占 Level 1 ~ 5 所有变异体总数的比例,数值越小则约简效果越好.

$$P = \frac{N_{\text{Level1}}}{\sum_{i=1}^{5} N_{\text{Level}i}} \tag{5}$$

由图 2 可得每个 Level 的具体数据,带入上式,有:

$$P = \frac{2mn - 2\sum_{i=1}^{m} a_i + \Delta}{2mn + \Delta + \sum_{i=1}^{m} a_i(n - a_i) + 3\sum_{i=1}^{m} a_i}$$

为便于讨论,设 $a_i = \frac{1}{2}n$,代入上式,可得在唯一谓词均匀分布在每个子谓词表达式中的情况下,本方法的约简效率为:

$$P \approx \frac{1}{1 + 5mn} \tag{6}$$

可以看出,本文所提方法的效率在谓词表达式复杂的情况下,即 mn 总数量较多的情况下,会有更佳的识别效率.

3.3　算法约束边界条件分析

对于冗余变异体生成的约束边界条件,使用分层抽样调查法进行分析. 由于不同算子生成的变异体相互独立,因此寻找非冗余变异体集合可以看作是从种群整体中进行分层抽样. 这里假设变异体在每一层中是均匀分布.

设变异体全体为 M,选出的非冗余变异体最优集合为 M_{opt},记为理想解. 本文所选的变异体集合为 M_{str},函数 $D(T,M)$ 表示用例集 T 可以检测出 M 中变异体的个数,则本文方法与理想解的比较可表示为:

$$Q_{str} = \left| \frac{D(T_{opt},M)}{D(T_{str},M)} \right| - 1 \tag{7}$$

设共有 k 个层,每层抽样 n 个变异体,则抽样总数是 $s = n * k$,理想解可以抽取出所有的 k 个非冗余变异体. 设随机变量 X_i,

$$X_i = \begin{cases} 1 & ,\text{当第 } i \text{ 层非冗余变异体被选中} \\ 0 & ,\text{否则} \end{cases}$$

令 X 表示本文方法所选出的变异体个数,则 $X=\sum_{1}^{k}X_i$,则其效果可用 X 的数学期望来表示. 因为假设变异体是均匀分布的,只需要获得 $X_1=1$ 的概率即可. 有:

$$P(X_i=1)=P(X_1=1)=1-\left(\frac{k-1}{k}\right)^{nk},$$

则有:

$$Q_{str}=\frac{k-(k-k(\frac{k-1}{k})^{nk})}{k-k(\frac{k-1}{k})^{nk}}=\frac{1}{(\frac{k}{k-1})^{nk}-1}=\frac{1}{e^n-1} \quad (8)$$

综上,本文所提方法在变异体均匀分布的情况下,冗余变异体生成的约束边界条件不低于$\frac{1}{e^n-1}$,其中 n 为每层抽样的变异体.

4 实验与结果分析

4.1 实验设置

本文的实验对象使用了广泛应用于变异测试实验的 Siemens Suite 和三个开源项目 aa200c、qlib 和 de118i-2(moshier.net).

实验的环境配置为:Ubuntu 12.04、JDK 1.6、gcc 4.6;所使用硬件配置为 Intel Pentium 2.90GHz,4GB 内存. 对于 aa200c、qlib 和 de118i-2 三个开源工程,分别选取其中的 dms.c、qcmplx.c 和 jplmp.c 文件,用例集是使用 CTS[8] 自动生成及随机生成组成,满足语句充分覆盖准则. 变异工具是 Proteum 和 CTS.

4.2 结果分析

图 3 显示了可检测非冗余算子的用例集对变异体全集的检测效果,使用变异得分作为评价指标. 可以看出对于所有被测工程,变异得分均超过了 85%. 但此时仅使用了变异体全集中较少一部分的变异体:据表 2 所示,变异体分别只使用了 19.79%,32.29%,26.44%,16.28%,18.16%,28.07%,34.2% 和 31.79%. 即使用本文方法,在较大幅度减少变异体执行次数的情况下,仍能保持较高的变异得分.

图 4 给出了本文方法减少变异测试总体时间成本的效果. 在变异测试过程中,变异成本主要由变异体执行时间和变异体预处理时间(包含变异体生成、编译及 IO 操作等)两部分组成. 时间度量分别以分钟和秒为单位. 从结果可以看出,由于整体变异体数量的减少,使得测试的执行时间和预处理时间都有较大幅度的减少,因而降低了变异测试整体的时间成本. 可以看出尽管对于小工程 aa200c 和 qlib 效果并不十分突出,但对规模较大的工程 replace 等,时间成本的降低十分明显.

表 2 实验对象

项目名	行数	有效变异体数	非冗余变异体数	约简率	用例数	功能描述
tcas	137	3799	752	19.79%	1608	aircraft collision avoidance system
totinfo	281	6827	2205	32.29%	1052	statistics data computes
schedule	296	3210	849	26.44%	2650	priority schedulers
printtokens	343	7865	1281	16.28%	4130	lexical analyzer
replace	513	11985	2117	18.16%	5542	pattern matching and substitution
aa200c	68	317	89	28.07%	129	astronomical ephemerides
qlib	79	226	77	34.2%	98	extended precision library
de118i-2	996	7133	2268	31.79%	59	asteroids ephemeris

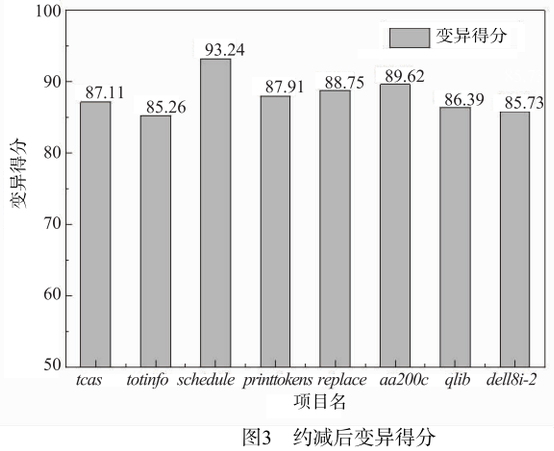


图3 约减后变异得分

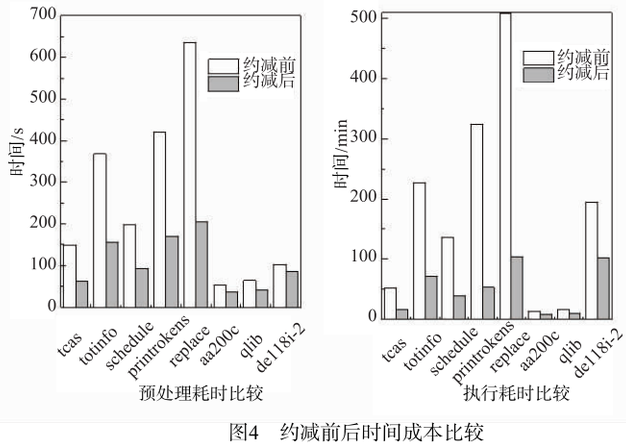


图4 约减前后时间成本比较

5 结束语

针对变异测试成本昂贵的问题,本文基于 PIE 模型和区间运算及谓词故障层级,分别对程序中的顺序语句和条件语句所产生的变异体,提出了冗余变异体寻找算法,并进行了检测效果分析,最后在分层抽样调查背景下,对冗余变异体生成的约束边界条件进行了分析. 结果表明,该方法可以在保持一定变异得分的前提下,减少变异测试的成本,有助于变异测试的实际应用. 下一步研究工作包括:将对不同变异体的感染区间赋予不同的权重,依据权值进行优选;扩充复杂谓词的故障层级,以支持更多的谓词类故障.

参考文献

[1] Jia Y, Harman M. An analysis and survey of the development of mutation testing[J]. IEEE transactions on software engineering, 2011, 37(5): 649 – 678.

[2] Offutt A J, Untch R H. Mutation 2000: Uniting the Orthogonal[M]. Springer US, 2001. 34 – 44.

[3] Just R, Ernst M D. Efficient mutation analysis by propagating and partitioning infected execution states[A]. Proceedings of the 2014 International Symposium on Software Testing and Analysis[C]. San Jose, USA: ACM, 2014. 315 – 326.

[4] Siami Namin A, Andrews J H. Sufficient mutation operators for measuring test effectiveness[A]. Proceedings of the 30th international conference on Software engineering [C]. Leipzig, Germany: ACM, 2008. 351 – 360.

[5] Kurtz B, Ammann P. Static analysis of mutant subsumption [A]. Software Testing Verification and Validation Workshops (ICSTW)[C]. Graz, Austria: IEEE, 2015. 1 – 10.

[6] DeMilli R A, Offutt A J. Constraint-based automatic test data generation[J]. IEEE Transactions on Software Engineering, 1991, 17(9): 900 – 910.

[7] Alefeld G, Mayer G. Interval analysis: theory and applications[J]. Journal of computational and applied mathematics, 2000, 121(1): 421 – 464.

[8] 王雅文,宫云战,肖庆,杨朝红. 基于抽象解释的变量值范围分析及应用[J]. 电子学报,2011,39(2):296 – 303. WANG Ya-wen, et al. A Method of Variable Range Analysis Based on Abstract Interpretation and Its Applications [J]. Acta Electronica Sinica, 2011, 39(2): 296 – 303. (in Chinese)

[9] Kaminski G, Ammann P. Improving logic-based testing [J]. Journal of Systems and Software, 2013, 86(8): 2002 – 2012.

[10] Kaminski G, et al. A logic mutation approach to selective mutation for programs and queries[J]. Information and Software Technology, 2011, 53(10): 1137 – 1152.

[11] Chen Z, Chen T Y, Xu B. A revisit of fault class hierarchies in general Boolean specifications[J]. ACM Transactions on Software Engineering and Methodology, 2011, 20(3): 13 – 42.

作者简介

钱莨南(通信作者)　男,1982 年生于黑龙江,2009 年获得山东大学硕士学位,现为北京邮电大学网研院博士研究生. 主要研究方向为软件测试,程序分析.
E-mail:634531068@qq.com

王雅文　女,1983 年生于陕西. 北京邮电大学副教授、硕士生导师. 主要研究方向为软件测试、程序分析.
E-mail:wangyawen@bupt.edu.cn