基于语句重要度的变异测试对象选择方法

巩敦卫[1,2],秦 备[1],田 甜[3]
(1. 中国矿业大学信息与电气工程学院,江苏徐州 221116;2. 兰州理工大学电气工程与信息工程学院,甘肃兰州 730050;
3. 山东建筑大学计算机科学与技术学院,山东济南 250101)

摘 要: 本文基于语句重要度,提出一种新的变异测试对象选择方法,以减少变异体的数量. 首先,给出反映变异测试对象重要性的3个因素;然后,基于这些因素的重要性,建立评价所选测试对象重要性的指标;最后,基于这些指标的值,选出重要度高的语句作为变异测试对象. 将所提方法应用于8个基准和工业程序的变异测试,实验结果表明,所提方法在维持很高变异测试充分度的同时,显著提高了变异测试的效率.

关键词: 软件测试; 变异测试; 变异体约简; 测试对象选择; 重要语句
中图分类号: TP311 **文献标识码:** A **文章编号:** 0372-2112 (2017)06-1518-05
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2017.06.034

Selecting Objects to be Mutated Based on Statement Importance

GONG Dun-wei[1,2], QIN Bei[1], TIAN Tian[3]
(1. *College of Information and Electrical Engineering, China University of Mining and Technology, Xuzhou, Jiangsu* 221116, *China*;
2. *College of Electrical and Information Engineering, Lanzhou University of Technology, Lanzhou, Gansu* 730050, *China*;
3. *College of Computer Science and Technology, Shandong Jianzhu University, Jinan, Shandong* 250101, *China*)

Abstract: Based on the importance of the statement, a novel method of selecting mutation object is proposed so as to reduce mutants. In this method, three factors that reflect the importance of an object are given, and then an index system of evaluating the importance of an object is established based on the importance of these factors. Finally, statements with high importance are selected as the objects to be mutated according to the values of these indexes. The proposed method is applied to test eight benchmark or industrial programs, and the experimental results suggest that the proposed method significantly improves the efficiency of mutation testing with high mutation sufficiency.

Key words: software testing; mutation testing; mutant reduction; test object selection; important statement

1 引言

软件测试是保证软件可靠性的重要手段[1~3],变异测试是一种面向缺陷的软件测试方法,能够有效的评价和改进测试用例的质量[4]. 变异测试通过向原程序中人为的植入缺陷,以模拟实际的软件缺陷. 每个缺陷副本,称为一个变异体. 设计测试集以尽可能多的揭示植入的缺陷,如果测试用例能够从结果上区分原程序和某一变异体,那么,称该变异体被杀死.

作为一种测试技术,变异分析能够真实的反映实际软件的各种缺陷[5]. 但是,实际软件通常包含规模庞大的代码行、复杂的语句,以及各种变量,显著增加了可变异的位置和可实施的变异算子类型,从而产生为数众多的变异体. 为了杀死这些变异体,必须采用充分的测试用例,反复执行原程序和变异体,降低了变异测试的效率,从而限制了变异测试的应用范围.

本文提出基于语句重要度的变异测试对象选择方法,与以往方法不同,通过分析原程序中语句的成分及其之间的关系,选择部分变异测试对象实施变异操作,减少需要杀死的变异体,从而提高变异测试的效率.

2 相关工作

Zhang 和 Gopinath 等生成所有的变异体之后,通过随机方法,仅选择一定比例的变异体,用于执行变异测试,此即抽样变异测试[6,7].

我们基于弱变异测试准则,通过分析由变异前后

收稿日期:2015-10-24;修回日期:2016-04-23;责任编辑:梅志强
基金项目:国家973重点基础研究发展计划(No. 2014CB046306-2);国家自然科学基金(No. 61375067; No. 61503220)

语句形成的变异分支之间的占优关系,约简被占优的变异分支,仅对非被占优变异分支对应的变异体实施变异测试[8].

Kinits 等根据控制流图中节点之间的控制关系,生成二阶变异体,以减少变异体的数量[9].

Delamaro 等提出一种基于增长模型的变异算子选择方法,不断增加变异算子数量,直到生成的测试用例能够杀死所有的变异体为止,从而得到一个完备的变异算子集[10].

我们曾提出了评价语句覆盖难度的 4 个指标,并给出了这些指标的计算方法,以选择最容易覆盖的语句作为目标语句,从而提高了语句覆盖测试数据生成的效率[11].

Ghiduk 等研究语句覆盖测试时,考察语句之间的占优关系,以非被占优作为语句重要程度的衡量依据,并选择非被占优的语句作为目标语句,减少了需要覆盖语句的数量[12].

上述变异体的约简方法,有的是在变异操作之后删除部分变异体,有的是在变异操作之前选择部分变异算子,从而生成较少的变异体. 已有工作用于测试对象重要程度的衡量,但是,在变异测试方面,基于测试对象重要度的工作还很少且不深入.

3　变异测试对象选择办法

首先,给出反映变异测试对象重要性的 3 个因素:然后,基于这些因素的重要性,建立评价变异测试对象重要性的指标;最后,提出一种变异测试对象选择办法.

3.1　反映变异测试对象重要性的因素

总体上讲,反映变异测试对象重要性的因素主要体现在如下 3 个方面:(1)变异测试对象所在语句的类型;(2)变异测试对象包含关键变量的个数;以及(3)变异测试对象包含的其它变量依赖的变量个数.

首先,考虑变异测试对象所在语句的类型. 一般的,一个程序中,条件语句的影响最大,循环语句次之,影响最小的是空语句. 鉴于此,采用变异测试对象所在语句的类型反映测试对象的重要性,是非常必要的.

记变异测试对象为 o,语句类型集为{表达式语句,条件语句,循环语句,函数调用语句,复合语句,空语句}. 如果 o 位于第 i 种语句类型中,那么,该变异测试对象关于语句类型的重要度为 $\alpha_i(o)$.

然后,考虑变异测试对象包含关键变量的个数. 鉴于条件或循环语句在程序中的重要性,因此,能够影响条件或循环语句的走向的变量称为关键变量. 包含关键变量越多,那么,该对象的变异对程序的执行影响越大. 因此,通过包含关键变量的个数反映对象的重要度,是十分合理的.

对于变异测试对象 o,记该对象包含的关键变量个数为 $\beta(o)$,那么,该变异测试对象关于包含关键变量个数的重要度为 $\beta(o)$.

最后,考虑变异测试对象包含的其它变量依赖的变量个数. 所谓依赖,具体而言,对于程序的 2 个变量 u 和 v,如果在对 v 的赋值时利用了 u,那么,称 v 依赖于 u. 比如,$u=x+1$,$v=u-4$.

对于一个变异测试对象,如果该对象包含的变量有很多其它变量依赖它,那么,对该对象的变异,必然影响其它变量所在的语句,因此,有理由认为,该对象的变异对程序的执行影响大.

对于变异测试对象 o,记该对象中包含的变量为 $v_1,v_2,\cdots,v_m$,依赖 $v_i,i=1,2,\cdots,m$ 的变量个数为 $\gamma(v_i)$,那么,该变异测试对象关于变量依赖个数的重要度为 $\Upsilon(o)=\sum_{i=1}^{m}\gamma(v_i)$.

3.2　变异测试对象重要度评价

记变异测试对象集为 $O=\{o_1,o_2,\cdots,o_n\}$,该集合包含 n 个变异测试对象,在这些测试对象中,有的是重要的,有的对程序的执行没有太大影响. 我们期望从 O 中选择一个子集 O',该子集包含 n' 个最重要的变异测试对象 $o'_1,o'_2,\cdots,o'_{n'}$,$n'\leqslant n$. 对于每一被选择的对象 $o'_i,i=1,2,\cdots,n'$,计算该对象关于上述 3 个因素的重要度,记为 $I(o'_i)$,那么,所选子集 O' 的重要度为:

$$I(O')=\sum_{i=1}^{n'}I(o'_i) \tag{1}$$

为了计算 $I(o'_i)$,考虑上述 3 个因素及其重要度. 我们知道,在上述 3 个因素中,变异测试对象所在语句的类型是最重要的,包含关键变量的个数次之,再次是包含的其它变量依赖的变量个数. 鉴于此,针对上述因素,给出反映它们重要性的权重,分别为 w_1,w_2,w_3,$w_1\geqslant w_2\geqslant w_3\geqslant 0$,$w_1+w_2+w_3=1$,那么,

$$I(o'_i)=w_1\alpha_i(o'_i)+w_2\beta(o'_i)+w_3\Upsilon(o'_i) \tag{2}$$

3.3　变异测试对象选择

首先,从变异测试对象集 O 中随机选择一个子集 O',并计算该子集的重要度;然后,从 O 中选取一定数量的变异测试对象,替换掉 O' 中相同数量的对象. 如果替换后子集的重要度大于原来的子集,那么,更新该子集;最后,将选取的变异测试对象从 O 中删除. 重复上述过程,直到 O 中不包含任何变异测试对象为止. 此时,得到的子集即为具有很大重要度的变异测试对象集. 选择算法的步骤如下:

步骤 1:设置算法的参数取值,包括:O 和 O' 分别包含的变异测试对象的个数 n 和 m、每次从 O 中选取的对象个数 k;

步骤 2:从 O 中随机选择 m 个对象,形成初始的变

异测试对象子集 O',并将这些对象从 O 中删除;

步骤 3:依据式(1)和(2),计算 O'及其所含元素的重要度,并依重要度对 O'的对象降序排列;

步骤 4:如果 O 为空集,转步骤 8;

步骤 5:从 O 中随机选取 k 个变异测试对象,替换 O'中重要度最小的 k 个,形成新的子集 O'',并将这些对象从 O 中删除;

步骤 6:依据式(1)和(2),计算 O''及其所含元素的重要度;

步骤 7:如果 $I(O'') \geqslant I(O')$,令 $O' = O''$;否则;舍弃新选入的语句,保留 O';转步骤 4;

步骤 8:输出 O'中的元素,作为具有重要度的变异测试对象,算法结束.

4 实验

4.1 实验环境

为了评价所提方法的性能,选取 8 个基准和工业程序作为被测程序[13-15],这些程序均采用 Java 语言编写,其基本信息如表 1 所列. w_1, w_2, w_3 的取值分别为 0.5、0.3、0.2. 此外,其它参数的取值见表 2. 采用随机法与本文方法比较,因为到目前为止,还没有看到通过约简变异测试对象,提高变异测试效率的成熟方法.

表 1 被测程序基本信息

ID	测试程序	行数	方法		变异体			程序功能
			总数	可测试数	总数	可测试数	等价变异体数	
J1	Mid	26	1	1	115	115	18	求 3 整数的中间值
J2	TrashAndTakeOut	30	2	2	111	111	29	未知
J3	Triangle	36	1	1	325	325	40	判定三角形类型
J4	WordUtils	173	2	2	262	243	34	字符串操作
J5	DurationFormatUtils	365	9	3	722	576	65	时间格式化
J6	FieldUtils	142	15	3	242	237	28	未知
J7	HelpFormat	416	39	2	301	275	36	未知
J8	NumberUtils	636	47	21	912	780	154	未知
	Sum	1824	116	35	2990	2662	404	

表 2 参数设置

参数	条件语句	循环语句	函数调用语句	复合语句	表达式语句	空语句	包含关键变量的个数	包含的其它变量依赖的变量个数
取值	0.3	0.25	0.15	0.1	0.05	0	0.1	0.05

实验基于 Microsoft Window XP SP3 操作系统和 Eclipse 3.4 集成开发环境. 首先,采用 MuClipse 生成变异体,并通过对日志文件“mutation_log”的解析,自动生成变异分支集;然后,根据选出的变异测试对象,选取对应的变异分支,插入到原程序 P 的相应位置,得到转化后的被测程序 P',采用随机法生成测试用例. 根据弱变异测试准则,能够覆盖变异分支的测试用例,同样能够杀死对应的变异体.

4.2 实验结果与分析

4.2.1 测试对象的约简

表 3 中,“可变异语句数”表示初始变异语句的大小,“选取的语句条数”表示杀死所选变异测试对象生成变异体的测试用例,能够维持很高的变异测试充分度时,选取的变异测试对象个数.

由表 3 可知,(1)在 8 个被测程序中,J3 获得了最大的约简率,为 73.9%;最小约简率来自 J8,为 46.8%;(2)所有程序均达到了很高的约简率,8 个被测程序的平均约简率为 59.8%,也即采用本文的方法,能够约简将近 60% 的被测对象. 因此,所提方法能够显著的减少测试对象的数量.

表 3 本文方法的约简率

ID	可变异语句数	选取的语句条数	约简率(%)
J1	12	5	58.3
J2	12	5	58.3
J3	23	6	73.9
J4	76	28	63.2
J5	132	57	56.8
J6	61	22	63.9
J7	21	9	57.1
J8	356	185	46.8
平均值	-	-	59.8

4.2.2 测试用例的有效性

对于测试对象约简前后的每一被测程序,分别生成覆盖对应变异分支的测试用例,并利用这些测试用

例执行变异测试,结果如表 4 和表 5 所列.

表 4　测试对象约简前,生成测试用例的有效性

ID	语句数	被覆盖的变异分支数	变异分支覆盖率(%)	被杀死的变异体数	变异得分(%)
J1	26	115	100	97	100
J2	30	111	100	82	100
J3	36	317	100	279	98.1
J4	173	237	98.3	201	96.5
J5	365	566	100	497	97.3
J6	142	237	100	207	96.7
J7	416	275	100	256	96.3
J8	636	769	98.6	735	97.0
平均值	–	–	99.6	–	97.6

由表 4 可知,对于 8 个被测程序,测试对象约简之前,生成测试用例的变异分支覆盖率平均为 99.6%,除了 J4 和 J8 的变异分支覆盖率为 98.3% 和 98.6% 之外,其它 6 个程序实现了变异分支全覆盖;在变异得分方面,这些测试用例的平均变异得分为 97.6%,变异得分最小的为 J7,也达到了 96.3%.

表 5　测试对象约简后,生成测试用例的有效性

ID	选取的语句数	被覆盖的变异分支数	变异分支覆盖率(%)	被杀死的变异体数	变异得分(%)
J1	5	115	100	96	99.0
J2	5	110	99.1	82	100
J3	6	317	100	274	96.1
J4	28	236	98.3	199	95.2
J5	57	566	100	495	96.9
J6	22	233	98.3	202	94.7
J7	9	261	94.9	243	91.3
J8	185	739	96.1	706	93.2
平均值	–	–	98.4	–	95.8

从表 5 可知,对于约简测试对象之后的 8 个被测程序,生成测试用例的分支覆盖也高达 98.4%,且有 3 个程序达到变异分支全覆盖;对于变异得分,这些测试用例的平均变异得分也高达 95.8%,且变异得分最小的程序 J7 已经达到 91.3%.

4.2.3　评价体系的有效性

采用随机法,选取与采用本文方法同样数量的语句,作为测试对象;基于这些测试对象,生成测试用例,并对测试对象约简之前的变异体执行变异测试,考察其变异得分;此外,还考察达到与本文方法相同的变异得分,需要选取的语句数,实验结果如表 6 和表 7 所列.

由表 6 可知,本文方法的变异得分高于随机法. 比如,对于程序 J1,本文方法的变异得分为 99.0%,而随机法的变异得分仅为 76.3%;对于所有被测程序,本文方法的变异得分为 95.8%,远高于随机法的 75.1%. 这说明,采用本文方法约简被测对象,基于此生成的测试用例具有更高的变异测试充分度.

表 6　基于不同方法选择被测对象,生成测试用例的变异得分

ID	选取的语句数	本文方法		随机法	
		被杀死的变异体数	变异得分(%)	被杀死的变异体数	变异得分(%)
J1	5	96	99.0	74	76.3
J2	5	82	100	72	87.8
J3	6	274	96.1	213	74.7
J4	28	199	95.2	146	69.9
J5	57	495	96.9	379	74.2
J6	22	202	94.7	166	77.8
J7	9	243	91.3	184	69.3
J8	185	706	93.2	538	71.0
平均值	–	–	95.8	–	75.1

由表 7 可知,为了得到相同的变异得分,本文方法需要变异的语句少于随机法. 比如,对于所有 8 个被测程序,得到 70% 的变异得分,本文需要变异 240 个语句,而随机法需要变异的语句却高达 316;得到 90% 的变异得分,本文需要变异 301 个语句,而随机法需要变异的语句高达 541,这说明,本文提出的评价语句重要性的指标体系是合理的.

表 7　达到某变异得分,不同方法所需的语句数

ID	70%		90%	
	本文方法	随机法	本文方法	随机法
J1	3	5	5	8
J2	3	4	4	8
J3	4	5	6	11
J4	17	30	26	57
J5	37	56	55	117
J6	17	21	20	39
J7	7	10	9	15
J8	152	185	176	286
合计	240	316	301	541

5　结束语

变异测试虽然是一种重要的测试方法,但是,为数众多的变异体影响了变异测试的效率,本文提出一种基于语句重要度的变异测试对象选择方法,与以往方法不同,通过分析原程序中语句的成分及其之间的关

系,选择部分变异测试对象实施变异操作,减少需要杀死的变异体,从而提高变异测试的效率. 将所提方法应用于 8 个被测程序,并与随机法比较,结果表明,采用本文方法是有效的.

需要说明的是,本文方法性能的评价,仅限于 8 个基准和工业程序,今后需要在规模更大的工业程序上,验证所提方法的有效性. 还有,本文方法设置的参数值也未必是最优的,确定各参数的最优取值已经超出了本文的研究范围. 因此,采用合适的方法,确定所需参数的最优值,也是需要进一步研究的内容.

参考文献

[1] 张岩,巩敦卫. 基于搜索空间自动缩减的路径覆盖测试数据进化生成[J]. 电子学报,2012,40(5):1011 – 1016.
Zhang Yan, Gong Dunwei. Evolutionary generation of test data for path coverage based on automatic reduction of search space[J]. Acta Electronica Sinica, 2012, 40(5): 1011 – 1016. (in Chinese)

[2] Mei Jia, Wang Shengyuan. An improved genetic algorithm for test cases generation oriented paths[J]. Chinese Journal of Electronics, 2014, 23(3): 494 – 498.

[3] Tian Tian, Gong Dunwei. Evolutionary generation approach of test data for multiple paths coverage of message-passing parallel programs[J]. Chinese Journal of Electronics, 2014, 23(2): 291 – 296.

[4] 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 Xiangjuan, Gong Dunwei. Mutation testing based on comparison of paths[J]. Acta Electronica Sinica, 2012, 40(1): 103 – 107. (in Chinese)

[5] Jia Y, Merayo M, Harman M. Introduction to the special issue on mutation testing[J]. Software Testing, Verification and Reliability, 2015, 25(5 – 7): 461 – 463.

[6] Zhang J. Scalability studies on selective mutation testing[A]. Proceedings of the 37th IEEE International Conference on Software Engineering[C]. Piscataway: IEEE Press, 2015. 851 – 854.

[7] Gopinath R, Alipour A, Ahmed I, et al. An Empirical Comparison of Mutant Selection Approaches[R]. Eugene: Oregon State University, 2015.

[8] 张功杰,巩敦卫,姚香娟. 基于统计占优分析的变异测试[J]. 软件学报,2015,26(10):2504 – 2520.
Zhang Gongjie, Gong Dunwei, Yao Xiangjuan. Mutation testing based on statistical dominance analysis[J]. Journal of Software, 2015, 26(10): 2504 – 2520. (in Chinese)

[9] Kintis M, Papadakis M, Malevris N. Evaluating mutation testing alternatives: A collateral experiment[A]. Proceedings of 17th Asia Pacific Conference on Software Engineering[C]. Sydney: IEEE Press, 2010. 300 – 309.

[10] Delamaro M E, Deng L, Li N, et al. Growing a reduced set of mutation operators[A]. Proceedings of Brazilian Symposium on Software Engineering[C]. Maceio: IEEE Press, 2014. 81 – 90.

[11] 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[J]. 软件学报,2015,26(8):1925 – 1936.
Gong Dunwei, Zhong Chaoqun, Yao Xiangjuan. Dominant statement(s) selection in testing programs with flag variables[J]. Journal of Software, 2015, 26(8): 1925 – 1936. (in Chinese)

[12] Ghiduk A S, Girgis M R. Using genetic algorithms and dominance concepts for generating reduced test data[J]. Informatica, 2010, 34(3): 377 – 385.

[13] Papadakis M, Malevris N. Automatically performing weak mutation with the aid of symbolic execution, concolic testing and search-based testing[J]. Software Quality Journal, 2011, 19(4): 691 – 723.

[14] Papadakis M, Malevris N. Searching and generating test inputs for mutation testing[J]. Springer Plus, 2013, 2(1): 1 – 12.

[15] Fraser G, Zeller A. Mutation-driven generation of unit tests and oracles[J]. IEEE Transactions on Software Engineering, 2012, 38(2): 278 – 292.

作者简介

巩敦卫 男,1970 年 3 月出生于江苏铜山,中国矿业大学博士,中国矿业大学教授,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

秦 备 男,1990 年 8 月出生于江苏铜山,中国矿业大学硕士研究生,主要研究方向:基于搜索的软件工程.
E-mail:qqin_bei@163.com

田 甜 女,1987 年 1 月出生于山东德州,中国矿业大学博士,山东建筑大学讲师,主要研究方向为软件测试.
E-mail:tian_tiantian@126.com