GAT 和 GT 的流程如图 1 所示.与 GP 相同的是,本文方法也是基于选择的目标路径生成测试数据;与 GAT 选择测试数据相同的是,本文方法选择路径,也基于遗传算法;GP 和 GT 也有共同点,即 GP 选择路径和 GT 选择测试数据,均采用贪心算法.此外,上述方法均采用基于 VC++6.0 实现的简单遗传算法,生成测试数据[26],其中,GAT 和 GT 根据测试数据选择结果,以未覆盖分支为目标,依次生成测试数据;在生成覆盖某一分支的测试数据过程中,也检查该测试数据是否覆盖其他分支.如果该测试数据覆盖某一分支,那么,将该分支从未覆盖的目标集合中删除.对于本文方法与 GP,根据选择的路径,以没有覆盖的路径作为目标,生成测试数据.

本文求解路径选择方法的遗传操作如第 3 节所述,w_1=0.01,w_2=1,w_3=0.1;对比方法的遗传操作分别为轮盘赌选择、单点交叉和单点变异.无论是本文方法还是对比方法,交叉和变异概率均分别为 0.8 和 0.15.此外,种群规模均为 50,用于路径或测试数据选择的最大进化代数为 3 000,用于生成覆盖每个目标路径测试数据的最大进化代数为 5 000.

为了比较不同方法的性能,采用如下指标:

目标分支覆盖率(r_{con}),为测试数据覆盖的目标分支数与所有目标分支数的比值.

生成的测试数据数($|T'|$),即生成的测试数据集中包含的测试数据个数.

已有测试数据比率,是指用于回归测试的测试数据集中包含 T 中测试数据的个数,与用于回归测试的数据集中测试数据的个数比值.

测试数据生成时间($t_{generate}$),是指生成测试数据消耗的时间.

执行时间(t_{total}),是测试数据或路径选择与测试数据生成所消耗的时间总和.

等待时间,是指测试数据或路径选择后,生成测试数据所需的时间与执行已有测试数据所需时间的差.

为了减少遗传算法的随机性对不同方法性能的影响,每种方法和每一测试数据集均独立运行 10 次,针对每一指标,求取某一方法 200 次实验的平均值.

5.3 实验结果

表 2 和表 3 列出了相应的实验结果,表中,加粗显示的数据为最优结果.由表 2 可以看出,路径或测试数据选择的时间均较短,而用于测试数据生成的时间则相对较长.

Table 2 Target branches coverage and the number of generated test data of different methods in each tested program

表 2 不同方法对 PG1~PG6 测试的目标分支覆盖率以及生成测试数据数

被测程序	GT		GAT		GP		GAP	
	r_{con} (%)	$\|T'\|$	r_{con} (%)	$\|T'\|$	r_{con} (%)	$\|T'\|$	r_{con} (%)	$\|T'\|$
PG1	96.25	1.05	**96.67**	0.92	92.5	1.15	94.59	**0.65**
PG2	100	0.15	100	0.15	100	0.25	**100**	**0.15**
PG3	94.29	2.15	95.36	1.57	85.71	1.95	**97.5**	**1.43**
PG4	100	1.1	100	1.28	99.16	1.35	**100**	**0.84**
PG5	97.90	2.45	97.64	2.6	93.71	3.05	**98.58**	**2.1**
PG6	95.48	3.9	95	4.26	95.44	4.45	**96.9**	**3.18**

Table 3 Perform time and test data generation time of different methods in each tested program(s)

表 3 不同方法对 PG1~PG6 测试的方法执行时间以及测试数据生成时间(s)

被测程序	GT		GAT		GP		GAP	
	t_{total}	$t_{generate}$	t_{total}	$t_{generate}$	t_{total}	$t_{generate}$	t_{total}	$t_{generate}$
PG1	0.63	0.63	0.72	0.55	0.72	0.72	**0.51**	**0.29**
PG2	**0.37**	0.37	0.67	0.38	0.53	0.53	0.98	**0.42**
PG3	2.24	2.24	**1.75**	1.42	2.13	2.13	2	**1.32**
PG4	1.22	1.22	1.57	1.33	1.78	1.78	**1.16**	**0.85**
PG5	39.84.	39.84.	42.27	41.82	56.65	56.65	**23.99**	**22.61**
PG6	61.67	61.67	67.41	66.92	73.61	73.61	**38.18**	**35.83**

(1) 对问题 1 的回答

由表 2 可以看出,相比其他方法,本文方法用于回归测试的测试数据集达到了很高的目标分支覆盖率.

① 由路径选择的结果我们发现,对于每一程序,采用本文方法选择的路径子集,均能够覆盖所有的目标分支.这说明,本文提出的路径选择模型和求解方法是可行的.

② 与其他方法相比,本文将分支覆盖问题转化为路径覆盖问题,使得需要生成的测试数据数减少.因此,在测试数据生成方面,本文方法具有显著的优势.此外,本文方法形成的回归测试数据集,分支覆盖率优于其他 3 种方法,在程序 PG2~PG6 中达到了最大的目标分支覆盖率.这意味着,本文建立的路径选择问题的模型适合回归测试.但是,对于程序 PG1,本文方法选择的未有测试数据覆盖的部分路径难以覆盖,且没有优化测试数据的生成过程,因此,目标分支覆盖率比 GAT 略低.这表明,对于某些程序的回归测试,如果优先选择容易覆盖的路径,那么,能够进一步提升所提方法的性能.

(2) 对问题 2 的回答

由表 2 和图 3 可以看出,相比其他方法,本文方法在达到很高分支覆盖率的同时,需要生成的测试数据最少.

① 对规模较小的程序 PG1~PG3,需要生成的测试数据最多仅为 1.43 个;对规模较大的程序 PG4~PG6,最多仅需生成 3.18 个测试数据.这充分说明,本文设定的路径选择问题的目标函数是合适的.

② 由图 3 可以看出,与其他方法相比,对于每一程序,本文方法的已有测试数据比率是最高的.这说明,本文方法对已有测试数据集的利用充分,有助于减少需要生成的测试数据.

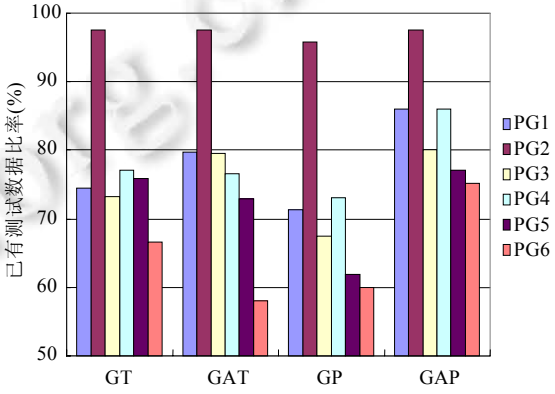


Fig.3 Comparison of existing test data ratio among different methods

图 3 不同方法已有测试数据比率的比较

(3) 对问题 3 的回答

结合表 2 和表 3 可以看出,相比其他方法,本文方法在达到很高分支覆盖率的同时,具有较少的时间消耗.

① 对于规模较大的程序 PG4~PG6,本文方法的执行时间最短;对于规模较小的程序 PG1,虽然本文方法的分支覆盖率为 94.58%,略低于 GAT,但是,本文方法的执行时间少于 GAT;对于程序 PG2 和 PG3,本文方法的执行时间略高于最优方法,但覆盖率是最高的.

② 对于每一程序,不同方法选择路径或测试数据的时间都很短,使得执行时间的大小主要取决于测试数据的生成.在这些方法中,本文方法的测试数据生成时间最短,随着程序规模的扩大,本文方法的优势更加明显.这说明,本文方法的实用性更强.

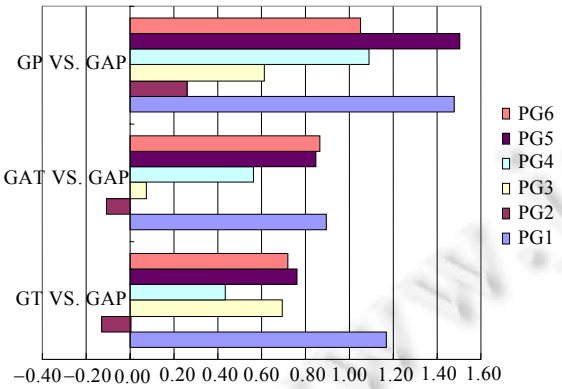


Fig.4 Extra times of waiting time of other methods than our method

图 4 相对本文方法,其他方法的额外等待时间倍数

③ 鉴于测试数据的生成过程可以和已有测试数据的验证过程并行执行,而前者较为耗时,因此,验证已有测试数据之后,需要等待新的测试数据生成.通过实验我们发现,与其他方法相比,对于大部分被测程序,本文方法由于应用较多已有的测试数据验证被测程序,使得等待生成测试数据的时间较短.

将每一对比方法的等待时间减去本文方法的等待时间,再除以本文方法的等待时间,能够得到该方法相对本文方法的额外等待时间倍数,结果如图 4 所示.由图 4 可以看出,对于程序 PG2,虽然本文方法的测试数据生成等待时间多于 GT 和

GAT,但是,等待时间相差并不大.对于其他程序,每一对比方法的等待时间明显多于本文方法,额外等待时间倍数最高为 1.51.

上述实验结果和问题的回答体现了本文方法的优势,但在衡量本文方法与其他方法在回归测试效率方面是否有显著性差异时,还需要通过多次实验采集到的样本之间的差异来推断不同方法总体之间的区别.本文采用 Z 检验的方法,通过 200 次实验结果,对不同方法的分支覆盖率(r_{con})和回归测试执行时间(t_{total})进行预测.不同程序、不同方法运行得到的分支覆盖率和回归测试执行时间是随机变量,这是因为,它们的值都受许多随机因素的影响,在大量实验中,它们近似服从正态分布.比较不同方法随机变量的平均值,如果分支覆盖率越高,回归测试所用的时间越短,那么,该方法回归测试效率越高.

以 PG5 为例,令本文方法和 GAT 的 t_{total} 随机变量的均值为 μ_1 和 μ_2,给出利用 Z 检验方法进行比较 μ_1,μ_2 的具体过程.鉴于样本方差是总体方差的无偏估计,采用样本方差的值当作对总体方差的估计值,也就是用样本标准差的值作为对总体标准差的估计值,得到 σ_1=10.25 和 σ_2=17.39.样本容量 n_1=n_2=200, $\overline{t_{total\,1}}$=23.99, $\overline{t_{total\,2}}$=42.27,显著性水平 α 的值为 0.01.

第 1 步.建立原假设 $H_0:\mu_1 \geqslant \mu_2$ 以及相反假设 $H_1:\mu_1 < \mu_2$.

第 2 步.设置统计量 $Z_1=\dfrac{\overline{t_{total\,1}}-\overline{t_{total\,2}}}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_2^2}{n_2}}}$.

第 3 步.给定拒绝域 $Z_1=\dfrac{\overline{t_{total\,1}}-\overline{t_{total\,2}}}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_2^2}{n_2}}} \leqslant -Z_\alpha$;

第 4 步.计算统计量的值 $Z_1=\dfrac{\overline{t_{total\,1}}-\overline{t_{total\,2}}}{\sqrt{\dfrac{\sigma_1^2}{n_1}+\dfrac{\sigma_2^2}{n_2}}}=\dfrac{23.99-42.27}{\sqrt{\dfrac{10.25^2}{200}+\dfrac{17.39^2}{200}}} \approx -12.8$;

第 5 步.得出结论,因为 $Z_1 \leqslant -Z_\alpha$, 落在拒绝域内,所以,拒绝 H_0,接受 H_1.即认为本文方法执行时间的期望值显著性小于 GAT.这说明,与 GAT 相比,本文方法回归测试的总时间明显小于 GAT,回归测试成本消耗优于 GAT.鉴于此,在表 4 中,GAP 与 GAT 在 PG5 关于 t_{total} 的显著性分析结果为 Y.

对于 6 个程序的显著性分析结果的比较情况见表 4.其中,本文方法显著性优于其他方法标记为 Y,否则,标记为 N,由表 4 可以看出,在简单程序中,由于各种方法分支覆盖率都较高,本文方法未体现出明显优势,但是在较大规模的 PG4~PG5 中,本文方法分支覆盖率显著高于其他方法.而在分支覆盖率接近时,在绝大多数情况下,本文方法需要的执行时间明显少于其他方法.

Table 4 Result of significance analysis

表 4 显著性分析结果

被测程序	GAP vs. GT		GAP vs. GAT		GAP vs. GP	
	r_{con} (%)	t_{total}	r_{con} (%)	t_{total}	r_{con} (%)	t_{total}
PG1	N	Y	N	Y	Y	Y
PG2	N	N	N	N	N	N
PG3	Y	Y	Y	N	Y	Y
PG4	N	Y	N	Y	N	Y
PG5	Y	Y	Y	Y	Y	Y
PG6	Y	Y	Y	Y	Y	Y

综合以上实验结果与分析,能够得出如下结论:本文方法能够达到很高的目标分支覆盖率,回归测试需要的耗时短,需要生成的测试数据少,且更适合于较大规模程序的测试.

5.4 多次回归测试中的可重用性分析

根据帕累托准则,软件缺陷具有类聚性,那么,因缺陷做出的软件维护,使得程序控制结构的变化也具有重复性.鉴于此,对某个程序维护版本的选择性回归测试,其中的路径选择或者测试数据选择结果,可以重复利用

到该程序其他版本的回归测试中.显然,这对执行频度较高的回归测试是较有价值的,这将进一步降低回归测试的成本.

为了验证本文方法在多次回归测试中的可重用性,并与GAT方法比较,首先,采用变异测试方法模拟维护行为,使用5个变异算子(ABS,AOR,LCR,ROR和UOI)对表1程序的不同位置实施变异,每个变异体相当于程序的一个维护版本;然后,每个被测程序随机挑选出100个变异体,仍然使用第5.1节中的20个不同的已有测试数据集,分别采用本文方法以及GAT,进行20次连续选择性回归测试,每次的测试需求为:覆盖变异影响到的分支;最后,对每个程序的若干需生成测试数据的变异体,分别统计本文方法以及GAT使用20个不同已有数据集的重用率,记为 *Rate*,*Rate* 可以表示为

$$Rate = \frac{ReuseCount}{AffectCount} \tag{7}$$

具体计算方法如下:

首先,对每一程序的100个变异体,统计目标分支未被已有数据全部覆盖的变异体数,记为 *AffectCount*,这些变异体需要生成一些新的测试数据.

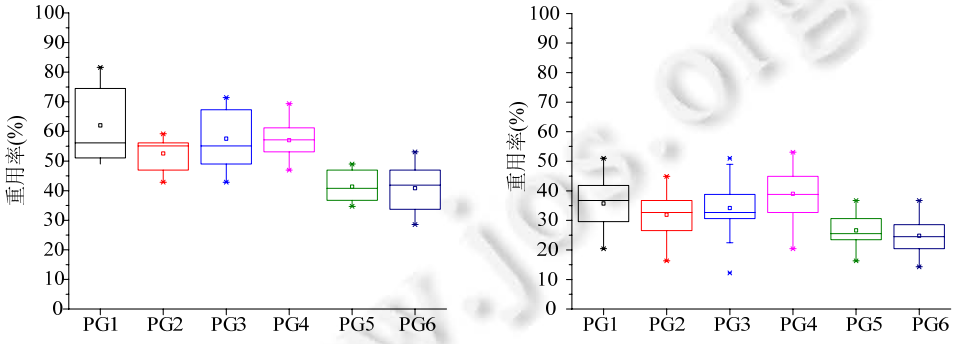
接着,对需生成测试数据的变异体依次生成测试数据,如果某个变异体的已有数据穿越的路径覆盖的分支与其他先生成测试数据的变异体相同,那么,在使用本文方法进行回归测试时,利用相应已测试变异体的路径选择结果,如果已有测试数据覆盖的分支与其他已测试变异体相同,则在使用GAT进行回归测试时,利用对应的测试数据选择结果.

然后,统计可以利用先测试变异体对象选择结果的个数,记为 *ReuseCount*.

最后,按照式(7),计算本文方法以及GAT的重用率,结果如图5所示.

由图5可以看出,通过本文方法选择的路径,在不同程序的多次回归性测试中,其重用率显著高于GAT方法选择的测试数据.在PG1程序中,重用率最高达到81.6%,在PG6程序中,重用率最低也达到28.5%.平均而言,在6个程序中,能够达到51.8%,这表明,采用本文方法可以减少在多次回归测试中的路径选择的时间,方法的可重用性高.

对于本文方法,分析在不同程序中的重用率我们发现,对于规模较小的程序,如图5所示,本文方法的重用率较大,对于规模较大的程序,本文方法的重用率较低.这进一步说明,当软件缺陷分布相对集中,或软件维护内容在较小的范围内时,采用本文提出的路径选择方法所选择的路径可重复利用的价值高,这在执行频度较高的回归测试中,具有很大的应用价值.



(a) 本文方法在6个程序中20次实验的重用率分布 (b) GAT方法在6个程序中20次实验的重用率分布

Fig.5 Distribution of reuse rate of our method and GAT in each tested program under 20 times continuous regression tests

图5 在6个程序中,本文方法与GAT方法,20次连续回归测试的重用率分布

5.5 有效性分析

本文主要针对分支覆盖问题,研究在取得较高分支覆盖率的同时,回归测试选择测试数据以及生成测试的时间较少,对于其他结构元素覆盖问题,还需要重新设定目标函数并给出相应的求解问题方法.

在外部有效性方面,对于分支覆盖问题,影响本文方法适用性的一个主要问题是:本文方法选择的被测对象以及相应的已有测试数据集的代表性.虽然本文选择的被测对象包含规模大小不同的 6 个 C 语言程序,但是,与工业程序相比,代码行数以及函数个数仍有差距.因此,选择维护后的工业程序作为测试对象可能会影响本文方法的性能.此外,本文不同被测程序的 20 个已有测试数据集,是从已满足程序覆盖准则测试数据集中随机选择出来的,其他测试数据集的产生方法,也有可能带来不同的实验结果.在后续的研究中,我们将继续选择较大规模的工业程序进行回归测试,并根据实际的维护情况,构造已有测试数据集,以进一步验证本文方法的有效性.

在内部有效性方面,本文用于求解路径选择问题的遗传算法参数设置并没有达到最优.针对路径选择问题,本文用遗传算法求解,并设计了个体评价函数、交叉、变异以及个体修复算子.在对不同的程序进行测试时,采用了相同的遗传参数,但针对具体程序,这些遗传参数有可能不是最优的,这将影响本文方法的性能.此外,本文在通过多目标优化选择需要覆盖的路径时,没有考虑这些路径的分布及其覆盖难度,有可能使得覆盖选择路径所需生成的测试数据难度增大,导致回归测试成本增加,选择一些较易生成测试数据的路径,将继续提高本文方法的性能.在实验分析方面,本文主要考察了分支覆盖率以及回归测试的执行时间,当需要生成新的数据时,选用不同的测试数据进化生成方法对本文的实验结果有较大影响,当测试数据生成的效率能够进一步提升时,这将降低本文方法的优势.

对本文所提方法的安全性进行分析,虽然本文方法以较低的测试开销达到了较高的分支覆盖率,期望可以检测出修改后程序内的缺陷,但是,本文方法仍然属于面向结构覆盖准则的测试数据选择以及生成方法,使用本文方法选择以及生成的测试数据对修改程序测试后,有可能修改程序仍然存在隐藏的缺陷,这就需要使用强度更高的结构覆盖准则或者使用面向缺陷检测的测试数据生成方法,在将来的研究中,我们将考虑把缺陷检测能力作为路径选择问题优化模型的目标之一,以增强本文方法的缺陷检测能力.

6 总结及下一步工作

本文将回归测试中的分支覆盖问题转化为路径覆盖问题,并基于选择的路径,生成用于回归测试的数据集,以覆盖程序维护影响的分支.为了选择需要覆盖的路径,将路径选择问题建模含有 3 个目标的优化问题;采用遗传算法生成路径时,设计具有针对性的个体编码方法、遗传算子以及修补算子.

将所提方法应用于 6 个不同规模的被测程序中,并与已有方法比较.实验结果表明,本文方法能够达到很高的目标分支覆盖率,回归测试需要的耗时短,需要生成的测试数据少,且更适合于较大规模程序的测试.

需要说明的是,本文仅通过 6 个基准程序验证所提方法的有效性.这些基准程序的规模还比较小,难以充分说明本文方法的可扩展性.因此,在今后的研究中,需要采用规模更大的实际工业程序来测试所提方法的性能.此外,本文在通过多目标优化选择需要覆盖的路径时,没有考虑这些路径的分布及其覆盖难度,使得覆盖选择路径所需生成的测试数据难度增大,从而限制了测试数据生成的效率.鉴于此,基于选择路径的分布和覆盖难度,建立路径选择问题新的(约束)多目标优化模型,并寻求有针对性的求解方法,也是需要进一步研究的问题.

致谢 感谢各位审稿专家以及本刊编辑的辛勤工作.

References:

[1] Rothermel G, Harrold MJ. Analyzing regression test selection techniques. IEEE Trans. on Software Engineering, 1996,22(8): 529−551. [doi: 10.1109/32.536955]

[2] Yoo S, Harman M. Regression testing minimization, selection and prioritization: A survey. Journal of Software Testing, Verification and Reliability, 2012,22(2):67−120. [doi: 10.1002/stv.430]

[3] Gu Q, Tang B, Chen DX. A test suite reduction technique for partial coverage of test requirements. Chinese Journal of Computers, 2011,34(5):879−888 (in Chinese with English abstract). [doi: 10.3724/SP.J.1016.2011.00879]

[4] Shi A, Gyori A, Gligoric M, Zaytsev A, Marinov D. Balancing trade-offs in test-suite reduction. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (SIGSOFT FSE 2014). New York: ACM Press, 2014. 246−256. [doi: 10.1145/2635868.2635921]

[5] Rothermel G, Untch RJ, Chu CY, Harrold MJ. Prioritizing test cases for regression testing. IEEE Trans. on Software Engineering, 2001,10(27):929−948. [doi: 10.1109/32.962562]

[6] Zhang ZY, Chen ZY, Xu BW, Yang R. Research progress on test case evolution. Ruan Jian Xue Bao/Journal of Software, 2013,24(4):663−674 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4379.htm [doi: 10.3724/SP.J.1001.2013.04379]

[7] Rothermel G, Harrold MJ. A framework for evaluating regression test selection techniques. In: Proc. of the 16th Int'l Conf. on Software Engineering (ICSE 1994). Sorrento: IEEE Computer Society Press, 1994. 201−210. [doi: 10.1109/ICSE.1994.296779]

[8] Smith AM, Kapfhammer GM. An empirical study of incorporating cost into test suite reduction and prioritization. In: Proc. of the 24th ACM Symp. on Applied Computing (SAC 2009). New York: ACM Press, 2009. 461−467. [doi: 10.1145/1529282.1529382]

[9] El-Hamid WSA, El-Etriby SS, Hadhoud MM. A general regression test selection technique. World Academy of Science, Engineering and Technology, 2010,4(2):893−897.

[10] Yoo S, Harman M. Using hybrid algorithm for Pareto efficient multi-objective test suite minimisation. Journal of Systems and Software, 2010,83(4):689−701. [doi: 10.1016/j.jss.2009.11.706]

[11] Sampath S, Bryce R, Memon A. A uniform representation of hybrid criteria for regression testing. IEEE Trans. on Software Engineering, 2013,39(10):1326−1344. [doi: 10.1109/TSE.2013.16]

[12] Panichella A, Oliveto R, Penta MD, Lucia AD. Improving multi-objective test case selection by injecting diversity in genetic algorithms. IEEE Trans. on Software Engineering, 2015,41(4):358−383. [doi: 10.1109/TSE.2014.2364175]

[13] Nardo DD, Alshahwan N, Briand L, Labiche Y. Coverage-Based regression test case selection, minimization and prioritization: A case study on an industrial system. Software Testing, Verification and Reliability, 2015,25(4):371−396. [doi: 10.1002/stvr.1572]

[14] Suri B, Singhal S. Understanding the effect of time-constraint bounded novel technique for regression test selection and prioritization. Int'l Journal of System Assurance Engineering and Management, 2015,6(1):71−77. [doi: 10.1007/s13198-014-0244-3]

[15] Gligoric M, Eloussi L, Marinov D. Practical regression test selection with dynamic file dependencies. In: Proc. of the Int'l Symp. on Software Testing and Analysis (ISSTA 2015). New York: ACM Press, 2015. 211−222. [doi: 10.1145/2771783.2771784]

[16] Ye N, Chen X, Peng J. Automatic regression test selection based on activity diagrams. In: Proc. of the 5th Int'l Conf. on Secure Software Integration & Reliability Improvement Companion (SSIRI-C 2011). Jeju Island: IEEE Computer Society Press, 2011. 166−171. [doi: 10.1109/SSIRI-C.2011.31]

[17] Kumar A, Tiwari S, Mishra KK. Generation of efficient test data using path selection strategy with elitist GA in regression testing. In: Proc. of the 3rd IEEE Int'l Conf. on Computer Science and Information Technology (ICCSIT 2010). Chengdu: IEEE Computer Society Press, 2010. 389−393. [doi: 10.1109/ICCSIT.2010.5564915]

[18] Santelices RA, Chittimalli PK, Apiwattanapong T, Orso A, Harrold MJ. Test-Suite augmentation for evolving software. In: Proc. of the 23th IEEE/ACM Int'l Conf. on Automated Software Engineering (ASE 2008). L'Aquila: IEEE Computer Society Press, 2008. 218−227. [doi: 10.1109/ASE.2008.32]

[19] Xu ZH, Kim Y, Kim M, Rothermel G, Cohen MB. Directed test suite augmentation: Techniques and tradeoffs. In: Proc. of the ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering (SIGSOFT FSE 2010). Santa Fe: ACM Press, 2010. 257−266. [doi: 10.1145/1882291.1882330]

[20] Xu ZH, Cohen MB, Rothermel G. Factors affecting the use of genetic algorithms in test suite augmentation. In: Proc. of the 12th Annual Conf. on Genetic and Evolutionary Computation (GECCO 2010). Portland: ACM Press, 2010. 1365−1372. [doi: 10.1145/1830483.1830734]

[21] Kim Y, Xu ZH, Kim M, Cohen MB, Rothermel G. Hybrid directed test suite augmentation: An interleaving framework. In: Proc. of the 7th Int'l Conf. on Software Testing, Verification and Validation (ICST 2014). Cleveland, OH: IEEE Computer Society Press, 2014. 263−272. [doi: 10.1109/ICST.2014.39]

[22] Xu ZH, Kim Y, Kim M, Cohen MB, Rothermel G. Directed test suite augmentation: An empirical investigation. Software Testing, Verification and Reliability, 2015,25(2):77−114. [doi: 10.1002/stvr.1562]

[23] Benedusi P, Cmitile A, De Carlini U. Post-Maintenance testing based on path change analysis. In: Proc. of the Int'l Conf. on Software Maintenance (ICSM'88). Santa Fe: IEEE Computer Society Press, 1988. 352−361. [doi: 10.1109/ICSM.1988.10187]

[24] Gong DW, Zhang WQ, Yao XJ. Evolutionary generation of test data for many paths coverag based on grouping. Journal of Systems and Software, 2011,84(12):2222−2233. [doi: 10.1016/j.jss.2011.06.028]

[25] Xie XY, Xu BW, Shi L, Nie CH. Genetic test case generation for path-oriented testing. Ruan Jian Xue Bao/Journal of Software, 2009,20(12):3117−3136 (in English with Chinese abstract). http://www.jos.org.cn/1000-9825/580.htm [doi: 10.3724/SP.J.1001.2009.00580]

[26] Gong DW, Yao XJ. Testability transformation based on equivalence of target statements. Neural Computing & Applications, 2012,21(8):1871−1882. [doi: 10.1007/s00521-011-0568-8]

[27] Mao CY, Lu YS. A simplified method for generating test path cases in branch testing. Journal of Computer Research and Development, 2006,43(2):321−328 (in Chinese with English abstract). [doi: 10.1360/crad20060220]

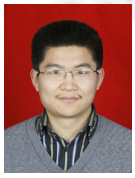
[28] Do H, Elbaum S, Rothermel G. Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Software Engineering, 2005,10(4):405−435. [doi: 10.1007/s10664-005-3861-2]

附中文参考文献:

[3] 顾庆,唐宝,陈道蓄.一种面向测试需求部分覆盖的测试数据集约简技术.计算机学报,2011,34(5):879−888. [doi: 10.3724/SP.J.1016.2011.00879]

[6] 张智轶,陈振宇,徐宝文,杨瑞.测试用例演化研究进展.软件学报,2013,24(4):663−674. http://www.jos.org.cn/1000-9825/4379.htm [doi: 10.3724/SP.J.1001.2013.04379]

[27] 毛澄映,卢炎生.分支测试中测试路径用例的简化生成方法.计算机研究与发展,2006,43(2):321−328. [doi: 10.1360/crad20060220]



吴川(1980－),男,江苏盐城人,博士生,讲师,CCF 会员,主要研究领域为软件测试.

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

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