第 42 卷 第 12 期
Vol. 42 No. 12
计 算 机 工 程
Computer Engineering
2016 年 12 月
December 2016
·体系结构与软件技术·
文章编号：1000-3428(2016)12-0073-05 文献标志码：A 中图分类号：TP391

基于二分 K-means 的测试用例集约简方法

汪文靖[1,2]，冯 瑞[1,2]

（1. 复旦大学 计算机科学技术学院，上海 201203；2. 上海视频技术与系统工程研究中心，上海 201203）

摘 要： 测试用例集约简是软件测试中的重要研究问题之一，目的是以尽量少的测试用例达到测试目标。为此，提出一种新的测试用例集约简方法。应用二分 K-means 聚类算法对回归测试的测试用例集进行约简，以白盒测试的路径覆盖为准则，对每个测试用例进行量化，使每个用例变成一个点。以黑盒测试的功能需求数作为聚类数，在聚类结果的每一簇中，按照离中心点的距离进行排序，依次从每一簇中选择测试用例，直至满足所有测试需求，得到约简的测试用例集。实验结果表明，该方法能有效地减小测试用例集的规模，降低用例集检错率。

关键词： 测试用例集约简；软件测试；二分 K-means 聚类算法；黑盒测试；白盒测试；检错率

中文引用格式： 汪文靖，冯 瑞. 基于二分 K-means 的测试用例集约简方法[J]. 计算机工程，2016，42(12)：73-77，83.

英文引用格式： Wang Wenjing，Feng Rui. Test Suite Reduction Method Based on Bisecting K-means[J]. Computer Engineering，2016，42(12)：73-77，83.

Test Suite Reduction Method Based on Bisecting K-means

WANG Wenjing[1,2]，FENG Rui[1,2]

（1. School of Computer Science and Technology，Fudan University，Shanghai 201203，China；
2. Shanghai Engineering Research Center for Video Technology and System，Shanghai 201203，China）

【Abstract】 Test suite reduction is an important research questions in software test，its purpose is to test as little as possible to achieve the test objective. This paper presents a test suite reduction method based on bisecting K-means. It uses the bisecting K-means clustering algorithm to reduce regression test suite. Regarding the path coverage of white box test as a criterion，each test case is quantified，so that each case becomes a point，the number of black box test functional needs is taken as the number of clusters，each cluster is sorted in the clustering results according to the distance from the center，the test cases from each cluster are selected，until all testing requirements are met and reduced test suite is got. Simulation experimental results show that this method can effectively reduce the size of the test suite，and effectively reduce the impact on the use case set error detection rate.

【Key words】 test suite reduction；software test；bisecting K-means clustering algorithm；black box testing；white box testing；error detection rate

DOI：10.3969/j.issn.1000-3428.2016.12.013

0 概述

随着软件开发的不断迭代更新，需要对软件功能进行回归测试[1]，测试用例集的不断积累，会存在大量冗余测试用例集。由于测试用例集的设计、执行和维护消耗大量的人机资源，因此需要对现有测试用例集进行约简[2]，用尽量少的测试用例就能完成程序测试，进而降低测试成本，提高测试执行效率。

测试用例集约简的方法有多种，如贪心算法[3]、HGS 算法[4]、GE&GRE 算法[5]、GA 算法等启发式方法[6]；基于组合覆盖的方法[7]、0-1 整数规划方法[8]、蚁群算法[9]、模拟退火算法等智能算法[10]；需求驱动的测试用例集约简方法等[11]。上述方法在一定程度上缩减测试用例集的规模，但普遍缺乏对测试用例集的错误检测能力的考虑。如果把测试用例集中的冗余用例全部去掉，将导致一些能检错的测试用例被误删，进而降低测试用例的检错能力[12]。

文献[13]采用 K-means 算法对组合测试生成的测试用例集进行聚类，具有较好的效果，然而该方法缺少聚类类别数以及约简后的测试用例集选择的客观方法，仅根据测试代价选择类别数对测试用例集

基金项目： 国家科技支撑计划项目(2013BAH09F01)；上海市科委科技创新行动计划项目(14511106900)。
作者简介： 汪文靖(1990—)，男，硕士研究生，主研方向为软件测试、模式识别、机器学习；冯 瑞，研究员。
收稿日期： 2015-11-09 **修回日期：** 2016-01-10 **E-mail：** 13210240109@fudan.edu.cn

的规模进行缩减，没有考虑测试用例的充分性和检错率。文献[14]提出 k 中心点的聚类算法，该方法的测试需求考虑了黑盒测试方法，没有结合白盒测试进行研究，对测试用例集的聚类缺少足够的说服力，尽管最终选择测试用例集时考虑了软件需求，但聚类方法无法保证聚类后的每一簇的中心点能完整覆盖所有测试需求。

针对测试用例集的约简问题，本文提出二分 K-means聚类的测试用例集约简方法[15]，为了兼顾检错率，对软件测试中白盒测试和黑盒测试进行了综合考虑。利用白盒测试定义聚类样本的特征，利用黑盒测试的功能需求初始化聚类数量和最终测试用例集选择的标准。对回归测试的测试用例集进行聚类后，在每一类中对用例按照离该类中心点的距离进行排序，从中心点开始依次选择测试用例，直到满足所有功能测试需求。

1 二分 K-means 聚类算法

K-means 聚类算法是一种测试用例集约简的重要方法，该方法能发现给定数据集的 k 个簇，每个簇可以其质心（即簇中所有点的中心）进行描述。首先随机确定 k 个初始点作为质心，然后把数据集中的每一个点分配到离其最近的一个簇中，重新计算每个簇的中心，把该簇中所有点的平均值作为质心。然后重新计算每个点到质心的距离，直到簇的分配结果不再改变为止。该方法具有算法简单、效果明显和容易容易实现等优点，然而由于对 k 个质心的初始选取比较敏感，质心选取不当将容易导致陷入局部最优，在大规模数据上收敛也较慢。

二分 K-means 算法是一种改进的 K-means 算法，通过弱化初始质心的选取减少对最终聚类效果的影响。通过计算误差平方和（Sum of Squared Error，SSE），即 cluster 中每个点到质心的平方差之和，度量聚类的优劣。该值越小表示数据点越接近于质心，簇的内聚性越好，聚类效果也越好。二分 K-means 算法的实现过程是首先将所有点作为一个簇，然后将该簇一分为二。选择能最大程度降低聚类代价函数的簇，并划分为 2 个簇，直至簇的数目等于用户给定的数目 k 为止，该方法能大幅提高算法效率，而且不受初始化问题的影响。二分 K-means 和 K-means 算法的聚类效果如图 1、图 2 所示。

从图 1、图 2 的实验结果可以看出，相对于 K-means算法，二分 K-means 算法更有优势，不会受到随机初始化质心的影响，分成的簇内部更聚集，簇间的可分性更好。

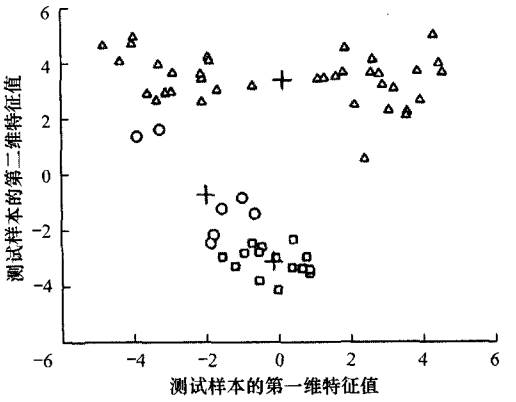

图 1 K-means 算法实验结果

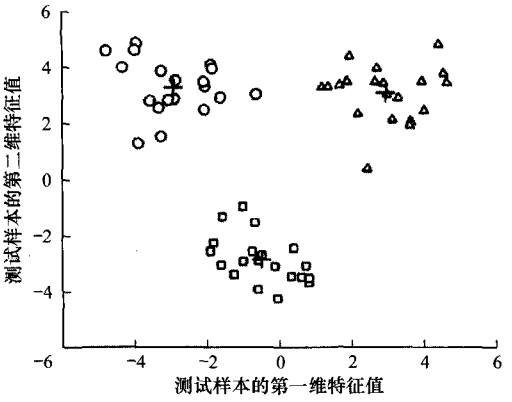

图 2 二分 K-means 算法实验结果

2 二分 K-means 测试用例集约简方法

2.1 测试用例集的转化

白盒测试中有 6 种逻辑覆盖方法，即语句覆盖、判定覆盖、条件覆盖、判定条件覆盖、条件组合覆盖和路径覆盖[16]。语句覆盖是指每条语句至少执行一次。判定覆盖是指每个判定的分支至少执行一次。条件覆盖是指每个判定的条件应取到各种可能的值。判定/条件覆盖是指同时满足判定覆盖和条件覆盖。条件组合覆盖是指每个判定中各条件的每一种组合至少出现一次。路径覆盖是指使程序中每一条可能的路径至少执行一次。

根据白盒测试的思想，上述覆盖方法发现错误的能力呈由弱至强的变化，本文使用路径覆盖方法将测试用例集中的每一个用例转化为一个向量。整个测试用例集转化为一个矩阵，作为二分 K-means 聚类算法的输入。具体转化方法为：将程序经过的路径编号，如果测试用例经过某一条路径，则把相应的分量设置为 1，否则设置为 0。

2.2 测试用例集的聚类

利用二分 K-means 聚类算法对测试用例集进行聚类时，需要选择合理的初试聚类数。借鉴黑盒测试的思

想[17]，将功能测试需求的个数作为聚类的数目 k。值得注意的是，仅利用聚类后的类中心数据进行简约，可能导致类数目不够或聚类效果不好，使类中心数据在测试需求上还有交叉现象，无法保证能覆盖所有功能测试路径，即无法满足测试的充分性。因此，聚类后需要保留测试用例集，并根据每一簇进行收集，依据每个点到质心距离的远近进行排序，以备后用。

2.3　测试用例集的选择

利用二分 K-means 聚类算法对测试用例集进行聚类后，如果直接将每个类的质心作为最终的约简测试用例集，可能会导致 2 个方面问题：1）还存在冗余的测试用例，没有足够约简，即还存在测试用例集的子集能够实现充分测试，满足所有测试需求；2）测试用例不够，无法保证充分测试。因此，需要进一步选择最终的约简测试用例集，本文利用需求驱动的约简方法，以满足所有功能测试需求为目标，当测试用例不够时，依次从每类已经排好序的结果中选择测试用例，直到充分测试。但是，如果测试用例集有冗余，不会直接进行约简，以满足测试需求，会结合测试用例集的检错率一起考虑，如果检错率下降明显，会选择更多的冗余用例，直到检错率达到一定阈值，阈值可以根据待测试软件的特性进行设置，比如软件一旦出错风险就很大的系统，应尽量保持一定的测试用例冗余。

2.4　约简方法流程

基于二分 K-means 算法的测试用例集约简模型如图 3 所示，形式化地描述了约简算法的实现过程。

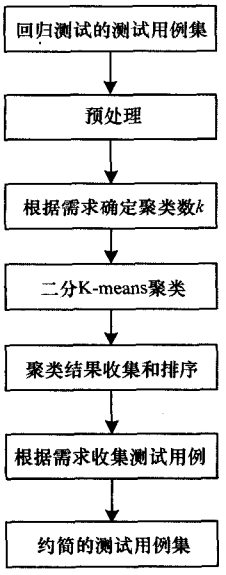

图 3　基于二分 K-means 算法的测试用例集约简模型

首先预处理系统开发和集成过程中收集的测试用例集，按照路径覆盖方法，把每个测试用例变成 0-1序列组成的向量，0 表示不经过该条路径，1 表示经过该路径，进而形成整个测试用例集的二维矩阵，行向量表示用例样本，列向量表示用例特征。根据系统需求测试报告和文档，也支持个性化设置和添加新需求，将需求数量初始化为聚类数目 k。然后利用二分 K-means 算法进行聚类，利用欧氏距离进行相似度计算，将聚类结果存在一个列表中，列表的每一项保存该类的样本，与测试用例一一对应。根据类的质心位置，把所有用例按照离质心的距离进行排序，离质心越近，排在越前面。从每一类的质心开始收集测试用例，把该用例满足的测试功能需求从需求集合中删除，直到需求集合为空，如果选择完所有类的质心，需求集合仍然不为空，接着选择每一类离质心最近的用例，依此类推，直到最终满足测试需求。最后执行测试用例，记录软件执行的结果，每发现一个错误，就写进日志里，并对日志结果进行分析，如果检错率下降明显，尤其是当有重大错误没有检测出来时，适当增加测试用例，以少量的冗余用例来保证软件测试的效果。把这些增加的用例集也并入前面选择出来的用例集，从而得到的用例集为最终约简的用例集。

3　仿真实验设计和分析

为检验算法的有效性，本文按照 2.4 节的算法流程，对已开发的智能视频分析系统的部分功能进行了仿真实验，使用白盒测试的路径覆盖准则设计了多组测试用例，表 1 定义了 27 个测试需求，表 2 在代码中植入 115 个不同程度和类型的错误，其中，12 个严重错误（编号 1 ~ 编号 12）；47 个一般错误（编号 13 ~ 编号 59）；56 个警告（编号 60 ~ 编号 115），并将本文算法（二分 K-means 测试用例集约简，BKTSR）与 HGS 算法、ACA 算法、SAA 算法进行了对比。

评价标准为：

1）测试用例集约简率：

$$SR = \frac{|T| - |T'|}{T} \times 100\%$$

其中，$|T|$ 代表原始测试用例集的个数；$|T'|$ 代表约简之后测试用例集的个数。

2）错误检测丢失率：

$$FL = \frac{|F| - |F'|}{|F|} \times 100\%$$

其中，$|F|$ 代表原始测试用例集能够检测出来错误的个数；$|F'|$ 代表约简之后测试用例集能够检测出来错误的个数。

表 1 系统测试需求

序号	需求	预期结果
1	选择分析系统,显示下一个分析系统的编号	正确显示分析系统编号
2	选择分析系统,点击保存添加正常	保存成功
3	选择守护进程,点击保存添加,守护进程仅可添加一个	只能保存一个守护进程
4	新添加成功的分析进程状态为未配置时,不可直接启动	提示进程未配置不能开启
5	点击修改配置图标切换至相应的系统配置界面	可以正常切换到配置界面
6	点击关闭图标	显示进程未开启不能关闭
7	监控点配置选择摄像头类型输入摄像头的 IP 分辨率和像素点击连接测试	摄像头参数正确情况下显示连接成功
8	选择算法类型如状态判别点击保存	提示监控点保存成功
9	点击启动图标(已配置好的)可开始运行程序	正常启动
10	开启多个识别系统时,可正常工作	支持多个识别系统同时运行
11	点击已有方案,右侧区域显示详细信息	右侧显示已有存储方案详细信息
12	点击新增存储方案或列表中空行,右侧区域显示空白内容	支持重新输入增加存储方案列表
13	点击连接测试可校验是否已连接	提示连接成功
14	选择上传方式大类-DB,类型-1001,可正常设置参数,连接测试正确	支持设置参数,连接测试显示连接成功
15	选择上传方式大类-Socket,类型-3001,可正常设置参数	可正常设置 Socket 下的参数
16	点击开始会看到视频流	海康威视摄像机视频流
17	点击继续视频继续播放	正常继续播放暂停后的视频文件
18	在开始状态下,点击批量截图会截取若干张图片	批量截图截取若干张图片
19	算法类别 4004:座位数目和座位坐标显示正确	座位数目和座位坐标显示正确
20	在左侧画线时,右侧配置信息会显示相应的坐标	右侧配置信息显示相应的坐标
21	算法类别 1001:可配置其他参数	可以配置行驶方向、场景方向和识别融合延迟
22	可进行系统级的各项配置	支持对系统级的各项配置进行修改并保存
23	支持澳盾摄像头输入,可测试连接	输入澳盾摄像头配置参数测试连接成功
24	算法大类支持:车辆卡口,状态判别,人脸识别	支持车辆卡口、状态判别,人脸识别
25	启动时验证加密狗	启动时验证加密狗无加密狗时无法启动程序
26	用户密码用密文管理	支持用户密码以密文管理
27	提取训练图片的特征	支持提取训练图片特征

表 2 代码中植入的错误参数

序号	错误参数	序号	错误参数	序号	错误参数
1	摄像头用户名	41	数据库端口号	81	座位号数目
2	摄像头密码	42	数据库用户名	82	行驶方向
3	识别区域起点 x	43	数据库密码	83	算法 1001 下车牌配置框
4	识别区域起点 y	44	默认行驶方向	84	宽度和高度不填
5	识别区域宽	45	场景方向	85	程序启动配置好的分析系统
6	识别区域高	46	文件路径	86	nginx 的目录
7	摄像头端口号	47	是否自动判定行驶方向	87	添加自动启动功能
8	摄像头 IP	48	配置文件	88	一上来就带着守护进程
9	摄像头类型	49	默认汉字	89	选取了类别之后清空文本说明
10	采集帧率	50	摄像头通道号	90	自动计算上传、存储个数
11	图像宽度	51	识别融合延迟	91	管理系统配置文件数据库密码
12	图像高度	52	最大曝光值	92	重置之后,再点击连接测试
13	识别阈值	53	人脸质量	93	只对某一路监控点启用的功能
14	处理帧率	54	人脸置信度	94	3001,没有 CacheDays
15	最小行人目标矩形宽度	55	类型说明	95	1002,密码密文
16	最小行人目标矩形高度	56	后缀(文件路径)	96	上传方案无连接
17	最大行人目标矩形宽高	57	端口号	97	清理功能问题
18	最大行人目标矩形高度	58	用户名	98	seat 读取不到模型
19	徘徊活动范围因子	59	密码	99	初始化数据库
20	徘徊活动长度因子	60	连接字符串	100	清理每秒存图的功能
21	掩码图像路径	61	人脸识别多服务版重启	101	车辆卡口算法
22	稳定出现次数阈值	62	守护进程启用标志	102	多接受方时 bug

续表 2

序号	错误参数	序号	错误参数	序号	错误参数
23	墙面线起点横坐标	63	区分自动重启和维护重启	103	启动界面会出现黑边
24	墙面线起点纵坐标	64	主程序不存在时不要马上启动	104	方案选择不上传
25	墙面线终点横坐标	65	初始数据无法自动添加	105	axis 连接测试
26	墙面线终点纵坐标	66	跳转控制面板界面启动	106	加密狗时间过期
27	蒙板图像地址	67	图片插入数据库显示	107	每 5 分钟验证一次加密狗
28	最大磁盘占用比率	68	把标题栏上用户管理隐藏	108	开 6 路
29	采集帧率异常阈值	69	进程列表把名称显示全	109	内存泄漏
30	处理帧率异常阈值	70	去掉新增进程中的取消按钮	110	座位状态判别
31	线程异常报警周期(s)	71	模型是否已更改	111	车牌识别 v2
32	监控点异常报警周期(s)	72	新建进程时上传方案拷贝错误	112	无法处理此命令
33	进程异常报警周期	73	存储方案列表显示	113	系统每秒输出结果
34	摄像头连接状态诊断周期	74	方案说明编辑框超长	114	msgqueue wait 应增加期限
35	摄像头断线重连周期(s)	75	数字文本栏输入过滤	115	只有一个 Semaphore
36	方案说明	76	设置 tab 顺序		
37	方案类型	77	监控点配置有一行行距不对		
38	图片存储目录	78	算法大类与算法类别调整		
39	数据库 IP	79	算法配置文件地址显示错误		
40	数据库名	80	画线页面中显示算法类别		

为了对错误的严重程度进行度量,更好地衡量测试效果,对错误的等级进行了划分,计算错误个数的时候进行加权处理,错误等级定义如表 3 所示。

表 3 系统错误等级

错误级别	说明	举例
warn	警告。状态内部数据异常,可能是暂时的	网络短时间不稳定
error	错误。系统发生错误,需要管理员关注并解决,但系统仍能运行	网络长期断线
fatal	严重错误。系统已无法自行解决问题,会关闭程序	加密锁被拔出

其中,严重错误的权重设置为 3,错误的权重设置为 1,警告的权重设置为 0.2。预设的错误总数为:12 × 3 + 47 × 1 + 56 × 0.2 = 94.2,实验结果如表 4 所示。从表 4 的实验结果可以看出,随着原始测试用例的增多,各个算法测试约简率都得到提升,而错误检测丢失率也逐步下降,从最终的 500 个测试用例的一行来看,约简率最好的算法是 SAA 算法(模拟退火算法),约简率达到 49.80%,最差的算法是 HGS 算法,约简率为 48.60%,而本文提出的算法约简率为 49.00%,上下波动为(-0.008,+0.016),由此可见,相比其他算法,本文提出的算法在约简效率上没有明显的差异。对于错误检测的丢失率指标,效果最好的是本文提出的算法,丢失率为2.68%,本文算法以外最好的算法是 HGS 算法,丢失率为7.12%,效果最差的算法是 ACA 算法(蚁群算法),丢失率为 7.38%,效果改善的比例区间为(2.657,2.754),由此可见,本文算法在检错率上比其他算法效果好 2 倍以上。结果表明,本文提出的方法在约简规模上相差不大,但检错能力明显大大增强。

表 4 测试用例集约简率、错误检测丢失率的对比 %

原始测试用例数	测试用例集约简率				错误检测丢失率			
	HGS 算法	ACA 算法	SAA 算法	BKTSR 算法	HGS 算法	ACA 算法	SAA 算法	BKTSR 算法
80	31.25	32.50	35.00	33.75	7.42	7.89	7.58	3.15
120	32.50	34.17	35.00	34.17	7.21	7.68	7.42	2.77
200	35.00	37.00	39.50	36.50	7.20	7.52	7.38	2.71
300	39.33	41.33	42.67	40.33	7.18	7.41	7.20	2.70
500	48.60	49.20	49.80	49.00	7.12	7.38	7.15	2.68

4 结束语

本文使用二分 K-means 聚类算法对回归测试用例集进行了约简,该算法使用测试需求的个数作为聚类的数目,使用白盒测试中路径覆盖的方法转化测试用例,聚类的结果按照离中心点的距离进行排序,约简的测试用例集依据功能需求进行选择。

(下转第 83 页)

[4] Akers S B. Binary Decision Diagrams[J]. IEEE Transactions on Computers Archive,1978,27(6):509-516.
[5] Bryant R E. Graph-based Algorithms for Boolean Function Manipulation [J]. IEEE Transactions on Computers,2001,35(8):677-691.
[6] Berkeley Logic Synthesis and Verification Group. ABC: A System for Sequential Synthesis and Verification[EB/OL]. [2015-08-08]. http://www.eecs.berkeley.edu/alanmi/abc/.
[7] 张敬丽,张会清,代汝勇. 基于 MIC-SVRE 的快速图像匹配算法[J]. 计算机工程,2016,42(1):210-214.
[8] Sharma P K, Kumar M. Implementation of BDDs by Various Techniques in Low Power VLSI Design[J]. International Journal on Recent Trends in Engineering & Technology,2014,10(2):221-228.
[9] Amarú L, Gaillardon P E, Micheli G. An Efficient Manipulation Package for Biconditional Binary Decision Diagrams[C]//Proceedings of Conference on Design, Automation & Test in Europe. Washington D. C., USA: IEEE Press,2014:1-6.
[10] Sharma P K, Kumar M. Implementation of BDDs by Various Techniques in Low Power VLSI Design[J]. International Journal on Recent Trends in Engineering & Technology,2014,10(2):221-228.
[11] Lindgren P, Kerttu M, Thornton M, et al. Low Power Optimization Technique for BDD Mapped Circuits[C]// Proceedings of Asia and South Pacific Design Automation Conference. Washington D. C., USA: IEEE Press,2001: 615-621.
[12] Chaudhury S, Dutta A. Algorithmic Optimization of BDDs and Performance Evaluation for Multi-level Logic Circuits with Area and Power Trade-offs[J]. Circuits and Systems,2011,2(3):217-224.
[13] Najm F N. Transition Density: A New Measure of Activity in Digital Circuits[J]. IEEE Transactions on Computer-aided Design,1993,12(2):310-323.
[14] Wang Xiang, Lu Ying, Zhang Yi, et al. Probabilistic Modeling During Power Estimation for Mixed Polarity Reed-muller Logic Circuits[C]//Proceedings of IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber,Physical and Social Computing. Washington D. C., USA:IEEE Press,2013:1414-1418.
[15] Amarú L,Gaillardon P E,Micheli G. BDS-MAJ: A BDD-based Logic Synthesis Tool Exploiting Majority Logic Decomposition[C]//Proceedings of the 50th Annual Design Automation Conference. Washington D. C., USA: IEEE Press,2013:1-6.
[16] Somenzi F. CUDD: CU Decision Diagram Package-release 2.4.0[EB/OL]. [2015-11-08]. http://www.cs.uleth.ca/~rice/cudd.html.

编辑 刘 冰

(上接第 77 页)

实验结果表明,该方法有效地去掉了冗余测试用例,缩减了测试用例集的规模,加快了测试用例执行的速度,降低了测试成本,同时对测试检错率的影响很小。下一步将从以下方面进行研究:1)优化聚类算法,提高约简效益;2)找到测试需求之间的关系,简化测试需求的个数;3)对约简的测试用例集进一步做白盒测试分析;4)对每一个用例检测出来的错误情况进行分析统计,对检测出来严重错误的用例增加权重或者直接标记,始终保存在用例集中。

参考文献

[1] 章晓芳,陈 林,徐宝文. 测试用例集约简问题研究及其进展[J]. 计算机科学与探索,2008,2(3):236-247.
[2] 孙家泽,王曙燕. 用于测试用例最小化问题的改进 PSO 算法[J]. 计算机工程,2009,35(15):201-202.
[3] Chvatal V. A Greedy Heuristic for the Set-covering Problem[J]. Mathematics of Operations Research,1979, 4(3):233-235.
[4] Harrold M J,Gupta R,Soffa M L. A Methodology for Controlling the Size of a Test Suite [J]. ACM Transactions on Software Engineering and Methodology, 1993,2(3):270-285.
[5] 张 瑞. 基于改进蚁群算法的测试用例集约简技术研究[D]. 广州:华南理工大学,2012.
[6] 张毅坤,赵 明,张保卫,等. 基于区间相容技术与 GA 的测试数据自动生成方法[J]. 西安理工大学学报, 2007,22(4):350-354.
[7] 邹 明. 组合测试用例生成技术研究[D]. 重庆:重庆大学,2012.
[8] 王大伟. 回归测试中基于需求优先级的用例选择算法研究[D]. 杭州:浙江工业大学,2012.
[9] 华 丽. 基于蚁群算法的测试用例集约简技术研究[D]. 重庆:西南大学,2009.
[10] 郑燕妮,李志蜀. 蚁群模拟退火算法在测试用例约简中的应用[J]. 计算机工程,2009,35(2):197-199.
[11] 孙富强,王林章. 多需求驱动的测试用例集约简方法[C]//全国第 20 届计算机技术与应用学术会议暨全国第一届安全关键技术与应用学术会议论文集(下册). 合肥:[出版者不详],2009.
[12] 崔应霞. 基于输入输出关系的综合黑盒测试方法[J]. 计算机工程与设计,2008,28(23):5581-5584.
[13] 冯 霞. 基于 K-means 聚类的组合测试用例生成优化算法[J]. 西安邮电大学学报,2015,20(1):44-48.
[14] 陈阳梅,丁晓明. 一种基于 K 中心点算法的测试用例集约简方法[J]. 计算机科学,2012,39(6):422-424.
[15] 陈平华,陈传瑜. 基于满二叉树的二分 K-means 聚类并行推荐算法[J]. 计算机工程与科学,2015,37(8): 1450-1457.
[16] 封 亮,严少清. 软件白盒测试的方法与实践[J]. 计算机工程,2000,26(12):87-90.
[17] 万年红. 软件黑盒测试的方法与实践[J]. 计算机工程,2000,26(12):91-93.

编辑 索书志