一种基于K中心点算法的测试用例集约简方法

陈阳梅 丁晓明
(西南大学软件学院 重庆 400715) (重庆市软件评测中心有限公司 重庆 400715)

摘 要 测试用例集约简的目的是用尽可能少的测试用例充分测试给定的测试目标。引入聚类分析中K中心点(K-medoids)算法的思想将每一个测试用例作为一个结点并寻找其相似性,将得到的聚类分析结果再根据测试需求从各簇中选择测试用例,从而得到约简的测试用例集。仿真实验的结果证明了该方法的可行性和有效性。

关键词 测试用例集约简,聚类分析,K中心点算法,错误检测率

Test Suite Reduction Methods Based on K-medoids

CHEN Yang-mei DING Xiao-ming
(College of Software, Southwest University, Chongqing 400715, China)
(Software Testing Center Co., Ltd. Chongqing, Chongqing 400715, China)

Abstract According to the given testing objectives, test suite reduction aims to satisfy all testing requirements with the minimum number of test cases. This paper introduced the idea of K-medoids algorithm of cluster analysis to finish the test suite reduction. First, treat every test case as a node and find its similarity. And then, according to testing requirements, choose test cases from test case suite. The results of the emulate programs prove that the method is effective and feasible.

Keywords Test suite reduction, Cluster analysis, K-medoids, Fault detection effectiveness

1 引言

在软件测试过程中,测试用例集的好坏直接决定了软件测试的效率高低。为求解测试用例集约简问题,研究人员提出了多种测试用例约简方法,主要包括启发式方法、整数规划方法、需求驱动的方法、基于遗传算法的技术等[2-4],但如何在约简率和错误检测率中寻找到平衡点依然是一个有待解决的难题。

本文引入聚类分析的方法,采用聚类分析中的划分方法K中心点(K-medoids)方法对原始测试用例集进行聚类,根据聚类产生的结果和测试需求集从各簇中选择测试用例,以此构成约简后测试用例集。实验结果分析表明,本文提出的基于聚类分析的测试用例集约简方法能有效地约简测试用例集,并大幅度地降低测试运行代价。

2 问题描述及预备知识

2.1 测试用例集约简

令测试用例集 T 与测试需求集 R 的二元关系 $S(T,R)=\{(t,r)\in TxR:$测试用例 t 满足测试需求 $r\}$,即 $S(T,R)$ 表示测试用例 $t\in T$ 与测试需求 $r\in R$ 的满足关系[1,3]。

如表1所列,测试需求集 $R=\{r_1,r_2,\cdots r_6\}$,测试用例集 $T=\{t_1,t_2,\cdots t_7\}$ 以及集合 R 与集合 T 的关系 $S(T,R)$。

由表1可知,$T_1=\{t_1,t_3,t_7\}$ 即可满足所有的测试需求,并且 T_1 是 T 的子集。在测试用例集 T 中存在着两种测试用例,一种是对于测试来说必不可少的测试用例,另一种是冗余的测试用例,这些测试用例对测试本身来说并没有执行的价值,相反,执行它们反而会增加测试成本。

表1 测试需求与测试用例的满足关系 $S(T,R)$

测试用例 (Test Case)	测试需求(Test Requirements)					
	r_1	r_2	r_3	r_4	r_5	r_6
t_1	√	√	√			
t_2		√	√	√	√	
t_3					√	√
t_4	√	√	√			√
t_5	√		√	√		
t_6		√	√		√	√
t_7				√		

测试用例集约简问题:令测试需求集 $R=\{r_1,r_2,\cdots,r_m\}$,测试用例集 $T=\{t_1,t_2,\cdots,t_n\}$ 满足所有的测试需求 r_i。测试用例集约简问题就是要在测试用例集 T 中找到一个子集 T',并有:如果 T' 的任意真子集 T_1' 都不能实现对测试需求集 R 的充分测试,即 $Req(T')=R$, $\forall T_1'\subset T_1$, $Req(T_1')\neq R$,那么测试用例集 T' 称为满足测试需求集 R 的最小测试用例集,其中,$Req(T')$ 和 $Req(T_1')$ 分别表示测试用例集 T' 和测试用例集 T_1' 所满足的测试需求所组成的集合。

2.2 K中心点(K-mediods)算法

目前已有学者将聚类的思想引入到测试用例集约简问题中,在文献[2]中,作者首先采用CLOPE聚类算法进行聚类,得到聚类结果后再利用分支覆盖和分布式算法相结合的方法

本文受2011年重庆市工业和信息化发展专项资金资助。

陈阳梅(1986-),女,硕士生,主要研究方向为软件测试;丁晓明(1966-),男,副教授,主要研究方向为软件工程、软件测试。

对聚类结果进行抽样，从而得到约简的测试用例集。实验结果表明：在保证错误检测率的情况下，约简率有了明显的提高；在保证约简率的情况下，错误检测率反而有所下降。我国的章晓芳、徐宝文等提出了一种基于测试需求的测试用例集约简算法[4]，其首先从测试需求集的角度对测试需求集进行约简，再利用传统的约简方法进行约简。本文将聚类和充分满足测试需求集两种思想相结合，首先对原始用例集进行聚类，再根据聚类产生的结果和测试需求集从各簇中选择测试用例，以此构成约简后测试用例集。

聚类算法繁多，针对测试需求集特点，并根据划分方法能对大型数据集进行高效分类[7]，以及针对异常数据的敏感处理能力，本文采用聚类算法中划分方法的 K 中心点(K-medoids)算法。

本文实验主要比较了划分方法中的经典算法：K 均值和 K 中心点算法。实验结果如图 1 和图 2 所示。

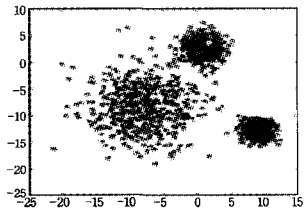

图 1　K 均值(K-mediods)算法实验结果

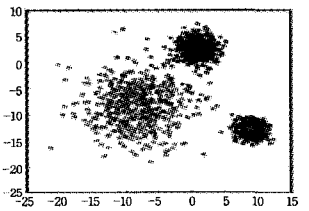

图 2　K 中心点(K-mediods)算法实验结果

由图 1 和图 2 的实验结果可得出，K 中心点算法的结果明显优于 K 均值算法。由于中心点是在已有的数据点中选取的，相对于 K 均值来说，K 中心点不容易受到由于误差之类的原因所产生的影响。

该算法较其他划分方法而言：第一，它能处理任意数据类型的属性；第二，该算法对异常数据不敏感；第三，对于聚类的结果，K 中心点算法的结果准确率很高，并且该算法所需要的训练数据非常少而且容易获得，训练出来的模型也是非常小的。

3　一种基于 K 中心点算法的测试用例集约简方法

3.1　测试用例聚类过程

本文针对当前的测试用例集约简问题，提出了一种基于 K 中心点算法的测试用例集约简方法。K 中心点算法主要的思想：在每个簇 C_i 中选出一个实际的对象来代表该簇，其余的每个对象聚类到与其最相似的代表性对象所在的簇中。

根据聚类使得到的各簇内对象距离尽可能小、各簇间对象距离尽可能大的原则，本文采用 K 个中心点与非中心点的线性距离来评价各测试用例之间的相似或相异度。

设 $T=\{t_1,t_2,\cdots,t_n\}$ 为非中心点测试用例的全部集合，$V=\{v_1,v_2,v_3,\cdots,v_k\}$ 为 K 个聚类中心点。在聚类过程中，需要计算所有非中心点与中心点之间的相异度来替代所需的总代价，即计算非中心点与中心点的欧几里得距离：

$$d(t_j,v_i)=\sqrt{(t_{j1}-v_{i1})^2+(t_{j2}-v_{i2})^2+\cdots+(t_{jm}-v_{im})^2}$$

式中，v_i 表示第 i 个中心点，t_j 表示第 j 个非中心点，则所求问题转化为求所有中心点与非中心点的欧几里得距离之和。

由此，该 K 中心点聚类算法的准则函数为：

$$C(T,V)=\sum_{j=1}^{n}\sum_{i=1}^{k}d(t_j,v_i)$$

聚类过程就是寻找 K 个中心点的过程，在运算过程中，当得到的 K 个中心点不再发生变化时，则算法结束。如图 3 所示，当确定聚类中心点的数目 K 及所容许的误差 ε 后，若相邻两次聚类准则函数 $C(T,V)$ 的差值小于 ε，聚类过程结束。根据所得到的 K 个中心点以及测试需求集 R，在各簇中进行抽样，得到约简测试用例集。

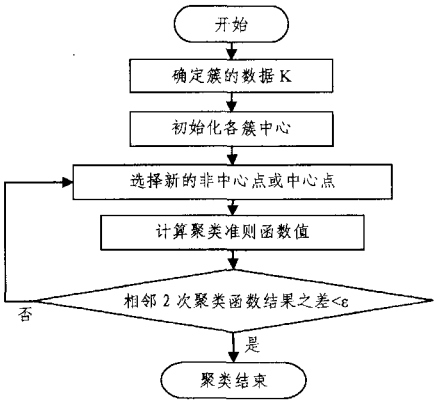

图 3　K 中心点(K-mediods)算法流程

3.2　算法模型及算法实现

如图 4 所示，该图为基于 K 中心点(K-mediods)算法的测试用例集约简模型，该模型形式化地描述了本文所提算法的约简思想。

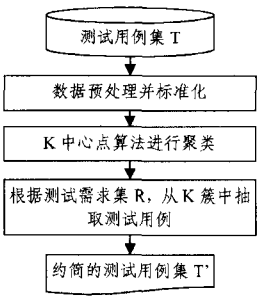

图 4　基于 K 中心算法的测试用例集约简模型

算法　基于聚类分析的测试用例集约简方法(CTSR)

```
DataPrepare:数据预处理,标准化
  Input:原始测试用例集 T1
  Output:T={t1,t2,…,tn}
  Data=getData();
  //对原始的测试用例集进行数据标准化
GetCluster:利用 K 中心点(K-mediods)算法聚类
  Input:T={t1,t2,…,tn}和簇的数目 K
  Output:C={c1,c2,…,ck}
  InitializeCluster();//初始化各簇中心
  GenerateCluster();//运行程序,生成 K 个簇
  CountCost();//计算初始各簇的准则函数值
  While(|C-C'|≥ε)
  {
  for i=k to 0 //选择各个中心点
  U=getNewNCenter();
  for j=n-k to 0 // 选择各个非中心点
```

```
V=getNewNCenter();
GenerateCluster();
C=CountCost();}
```

TestSuiteReduction：根据测试需求集从聚类各簇中抽取测试用例形成约简后测试用例集

```
Input:C={c1,c2,…,ck},R={r1,r2,…,rm},V={v1,v2,…,vk}
Output:T'={t1,t2,…,tr}
getTestCase(){ //抽取测试用例
getV();
//将 K 个聚类的中心加入到测试用例集 T'中
for i=0 to k
    for j=1 to k
if(R(ti)⊂R(tj)) then 将 tj 从 T'中剔除}
SR=(n-r)/n //计算约简率 SR
```

FaultDectect：检测测试用例集检测错误数

```
Input:T'={t1,t2,…,tr}和 T={t1,t2,…,tn}
Ountput:N
for i=1 to r
if(ti 检测出错误) N=N+1;
//统计约简后测试用例集 T'所检测出的错误数
FDE=N/N' //计算错误检测率 FDE
```

4 实例研究及对比实验

在本文的仿真实验中，分别采用经典的 HGS 方法[1]、CLOPE 方法[2]和本文提出的一种基于聚类分析的测试用例集约简方法(CTSR)对原始测试用例集 T 进行了约简测试。HGS 算法是 M. J. Harrold 等人提出的一种高效且经典的测试用例集启发式方法[2-4]；CLOPE 算法引入了 CLOPE 聚类分析算法，聚类后再利用分支覆盖和分布式算法相结合的方法对聚类结果进行抽样，得到约简测试用例集。

仿真实验主要从约简率、错误检测率和有效性 3 个方面进行了比对。在实验过程中，评价体系主要从以下 3 个方面体现[2,5]：

约简率(SR)：$\frac{|T|-|T'|}{|T|}\times 100$，其中 $|T|$ 为原始测试用例集中测试用例数量，$|T'|$ 为约简后测试用例集中测试用例数量。

错误检测率(FDE)：$\frac{|F'|}{|F|}\times 100$，其中 $|F|$ 为原测试用例集 T 检测出的错误数，$|F'|$ 为约简后测试用例集所能检测的错误数。

检错有效性(VAL)：$\frac{LR}{100-SR}$，该检错有效性表示约简后测试用例集中测试用例的平均检错能力。

实验对比结果如表 2 所列。

表 2 测试用例集约简算法结果比较

实验结果	初始测试用例数	HGS	CLOPE	CTSR
test01	40	27	26	25
test02	80	51	50	48
test03	100	63	60	58
test04	200	123	120	115
test05	500	291	287	279

通过表 2 的实验结果显示：本文提出的 CTSR 方法对原始的测试用例集 T 进行了有效地约简，约简率在原有基础上有所提高；通过表 3 的实验结果显示：本文提出的 CTSR 方法在一定程度上提高了测试用例集约简率 SR、错误检测率 FDE 以及检错有效性 VAL，从而验证了本算法是可行的、有效的。

表 3 测试用例集约简率、错误检测率及检错有效性对比结果

实验结果	SR/HGS	SR/CLOPE	SR/CTSR	FDE/HGS	FDE/CLOPE	FDE/CTSR	VAL/HGS	VAL/CLOPE	VAL/CTSR
test01(10)	32.50	33.75	37.50	90.00	92.31	95.00	1.33	1.39	1.52
test02(15)	36.25	37.35	40.00	93.33	90.00	96.67	1.46	1.44	1.61
test03(20)	37.00	39.50	42.00	95.00	93.33	97.50	1.51	1.54	1.68
test04(25)	38.50	40.25	42.50	96.00	97.50	98.00	1.54	1.63	1.70
test05(30)	41.80	42.50	44.20	96.67	96.00	98.33	1.66	1.67	1.79

结束语 本文针对目前的测试用例集约简问题所存在的不足，提出了一种基于 K 中心点算法的测试用例集约简(CTSR)方法，并利用对比仿真实验验证，CTSR 方法能够在提高测试用例约简率的同时有效地提高测试用例的错误检测率。该实验结果表明本算法仍存在着不足，如 K 中心点算法运算代价较高，时间开销较大，进一步的工作可以在以下两个方面开展：(1)优化 K 中心点算法，提高聚类效果；(2)根据测试需求的重要性为每个测试需要进行加权，通过运用聚类分析方法得到更好的聚类结果。

参 考 文 献

[1] 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

[2] Parsa S, Khalilian A, Fazlalizadeh Y. A new algorithm to Test Suite Reduction based on cluster analysis[C]// 2009 2nd IEEE International Conference on Computer Science and Information Technology. 2009: 189-193

[3] 章晓芳，陈林，徐宝文，等. 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

[4] 章晓芳，徐宝文，聂长海，等. 一种基于测试需求约简的测试用例集优化方法[J]. 软件学报，2007，18(4)：821-831

[5] 吴洁，丁晓明. 基于程序切片的测试用例集约简方法[J]. 重庆交通大学学报：自然科学报，2010，29(2)：319-320

[6] 赵书慧. K 中心点算法—PAM 的分析与实现[J]. 福建电脑，2008，24(6)：104-105

[7] 孙吉贵，刘杰，赵连宇. 聚类算法研究[J]. Journal of Software，2008，19(1)：48-61