

基于向量相似度的测试用例集约简方法

刘 锋,李 朋,朱二周

(安徽大学 计算机科学与技术学院,安徽 合肥 230601)

摘 要: 测试用例集约简的目的是在保证原有测试用例集覆盖度不变的情况下,使用一定的方法策略,尽可能的缩减测试用例集的数量,从而达到提高软件测试效率、降低测试成本的目标. 本文的向量相似度算法是先利用二元向量的相似性函数,然后计算出测试用例之间的相似度、覆盖度和冗余度,最后根据计算结果选择最佳测试用例,从而得到约简后的最优或近似最优测试用例集. 通过实验表明,该算法与现有的 GRE 算法相比:约简后测试用例集的数量平均降低约 10%,而测试用例集总的冗余度平均降低约 13%.

关键词: 测试用例集约简;软件测试;向量相似度;二元向量

中图分类号: TP311.5

文献标识码: A

文章编号: 1000—7180(2017)03—0035—05

Test Case Reduction Method Based on Vector Similarity Algorithm

LIU Feng,LI Peng,ZHU Er-zhou

(School of Computer Science and Technology, Anhui University, Hefei 230601, China)

Abstract: The purpose of test suite reduction is to improve the efficiency of software testing and reduce the cost of test. At the same time, the coverage of the original test cases should be ensured invariable and the number of test cases could be reduced as possible with the use of a certain strategy. Vector similarity algorithm is that the binary vector similarity function should be used firstly, then the similarity, coverage and redundancy test cases needed be calculated, and the best test cases is selected according to the calculation result, finally we will get the optimal or approximate optimal test cases after the reduction. Compared with the existing algorithm of GRE, experiments show that the number of test cases decreased by about 10% averagely and the whole redundancy rate of test cases decreased by about 13% averagely according to the vector similarity algorithm.

Key words: test suite reduction; software test;vector similarity;binary vector

1 引言

由于软件行业的不断规范化和完善化,也随之带动着软件测试行业的不断发展,软件测试方法也由原始的纯手工测试逐渐地演变为规范化和程序化. 这些都表明了:软件测试在软件开发生命周期中所占的比重也在日益地增大. 在进行软件测试之前,测试人员要依据测试需求来编写相应的测试用例集. 软件测试的目的是为了发现程序中的缺陷,而软件测试的过程就是利用生成的测试用例集 T 对每个测试需求进行测试,从而发现软件中的缺陷. 但

是,随着软件规模的不断增加,使软件开发过程中的迭代、演化需要频繁地进行软件测试,进而使测试需求的数量和与之相应的测试用例的数量也在不断的增加,从而导致了冗余测试用例的出现,这些冗余测试用例严重的影响了软件测试效率,如何选择一个好的测试用例集 T 来提高软件测试的效率和软件质量,已经成为现在软件测试研究的一个重要课题.

自从 1993 年 M. J. Harrold 等人首次提出测试用例集简化的概念后, T. Y. Chen 等人又给出了测试用例集简化问题中的一系列术语^[1]. 从此,测试用例集约简问题引起了很多研究人员的关注,现有的

测试用例集约简方法就是针对原有的测试需求集 R , 生成相应的测试用例集 T ; 然后, 所有满足测试需求的测试用例就构成了初步的测试用例集 T ; 最后, 为了提高测试效率, 需要采用一些算法对这个测试用例集进行约简, 从而尽可能的减少冗余的测试用例. 因此, 测试用例集约简的最终目的: 在满足测试需求的情况下, 尽可能的使用最少的测试用例, 以达到提高测试效率、降低测试成本的效果^[2].

本文提出了一种基于向量相似度^[3]的测试用例集约简方法.

2 测试用例集约简的常用算法及其比较

为了求解该测试用例集约简问题, 研究人员已提出了多种约简方法, 目前常用的测试用例集约简方法主要有: 贪心算法、GRE 算法以及基于遗传算法的方法等.

2.1 贪心算法

文献[1]首次提出了用贪心算法对测试用例集进行约简. 贪心算法执行步骤为: ①从测试用例集 T 中选择一个对测试需求集 R 覆盖度最大的测试用例 t_i , 其中 $t_i = \{\max | R(t_i) | t_i \in T\}$ (假如这样的测试用例不止一个, 可以随机的选择其中一个) 加入测试用例约简集 T' 中; ②将已满足的测试需求和已选择的测试用例从对应的集合中删除; ③然后重新建立新的测试用例与测试需求二者之间的矩阵对应关系; ④最后循环执行步骤①、②和③, 直到测试需求集 R 为空时结束.

贪心算法的特点是: 该算法的运行效率很高, 虽然它的时间复杂度^[1]为 $O(m * n * \min(m, n))$, 但是空间复杂度相对较低; 最重要的是该算法实现起来简单且易于理解, 不过该算法有时会陷入局部最优解, 因此, 它不适合全局范围求最优解.

2.2 GRE 算法

定义 1 在测试用例和测试需求对应关系中, 唯一可以满足某个测试需求的测试用例叫做必不可少测试用例, 使用 $Nec(T)$ 来表示.

定义 2 在测试用例集中, 假如存在 $R(t_j) \in R(t_i)$, 则 t_j 就称为冗余测试用例, 使用 $Red(T)$ 来表示.

GRE 算法^[4-5]是在贪心算法的基础上增加一些限定条件得到的. 首先, GRE 算法在测试用例集中找出所有必不可少测试用例, 再将其加入到测试用例约简集 T' 中; 然后, 将冗余测试用例从测试用例集 T 中删除; 最后, 使用上面提到的贪心算法直至

所有的测试需求都被满足. 该算法的最坏时间复杂度^[6]为 $O[\min(m, n)(m + kn^2)]$. 其中: k 表示一个测试用例最多能满足的测试需求数量. 通常该算法的时间复杂度比贪心算法更好些, 而且能较快速的降低测试用例集约简的问题规模.

2.3 遗传算法

可以把测试用例集约简问题转化为 NP-C 问题^[4]来求解, 遗传算法^[7]是一种自适应的智能搜索算法, 现在很多 NP-C 问题都会优先选择用该算法来解决, 该算法相比贪心算法和 GRE 算法实现起来要复杂的多, 但是效果更好.

本文提出的测试用例集约简方法是基于 GRE 算法和向量相似度计算方法相结合而形成的一个新算法, 所以该算法可以简记为: GRE-VS 算法. 因为 GRE-VS 算法又是在 GRE 算法的基础上增加一些限定条件得到的, 因此, GRE-VS 算法比 GRE 算法在降低测试用例集的数量和总的冗余度方面更高效, 同时能快速的降低测试用例集约简的问题规模, 又因为该算法融合了向量相似度的计算方法, 所以实现起来更简单.

3 向量相似度方法在测试用例集约简问题中的应用

3.1 测试用例集 T 与测试需求集 R 关系的数学表示

使用矩阵 TR 表示测试用例集和测试需求集之间的关系. 现在假设有 n 个测试用例和 m 个测试需求, 则测试用例 - 测试需求的 $n * m$ 维关系矩阵如下所示:

$$TR = \begin{bmatrix} tr_{11} & tr_{12} & \cdots & tr_{1m} \\ tr_{21} & tr_{22} & \cdots & tr_{2m} \\ \cdots & \cdots & \cdots & \cdots \\ tr_{n1} & tr_{n2} & \cdots & tr_{nm} \end{bmatrix}$$

tr_{ij} 表示第 i 个测试用例是否能满足第 j 个测试需求: 如果 $tr_{ij} = 1$ 则能满足; 如果 $tr_{ij} = 0$ 则不能满足.

如何采用一些算法去实现: 用最少测试用例来达到对测试需求的全覆盖, 本文提出的利用向量相似度方法可以很好的解决测试用例集约简中的一些主要问题.

3.2 向量相似度算法的原理

相似度是指两个对象之间的相似程度. 根据不同的研究模型, 可以把相似度的概念进一步细化, 例如: 形状相似度、文本相似度、系统相似度和向量相似度等. 目前常用的相似度计算的方法有两种: 相似

性函数法和距离测量法. 根据测试用例和测试需求之间的关系特点, 本文选择的是相似性函数法, 而按向量中元素的不同, 相似性函数又可以分为: 一般向量相似性函数和二元向量相似性函数.

3.2.1 一般向量相似性函数

一般向量相似性函数常用的有相关系数法、夹角余弦法、广义 Jaccard 系数法和广义 Dice 系数法^[8-9]等等, 经过分析, 这些方法都不适合于测试用例集的约简. 最终, 本文选择的是二元向量相似性函数, 它的原理和实现起来都很简单, 而且能够有效的解决测试用例集约简问题.

3.2.2 二元向量相似性函数

它也称为匹配测度, 所谓二元向量就是函数变量只要(0-1)两种状态, 这种变量称作二元变量. 二元变量可以分为对称和非对称两种, 对称二元变量就是: 二元变量的两种(0-1)状态价值等且权重一样, 否则称为非对称二元变量. 假设有二元变量 $X, Y: X = (x_1, x_2, x_3, \dots, x_n)', Y = (y_1, y_2, y_3, \dots, y_n)'$, 则 X, Y 两个二元向量的匹配数目可以定义为:

$$a = \sum_{i=1}^n x_i y_i \quad (1)$$

$$b = \sum_{i=1}^n (1 - x_i) y_i \quad (2)$$

$$c = \sum_{i=1}^n x_i (1 - y_i) \quad (3)$$

$$d = \sum_{i=1}^n (1 - x_i) (1 - y_i) \quad (4)$$

a 表示在二元向量 X 和 Y 中成功匹配(1-1)的数目; b 表示在二元向量 X 和 Y 中成功匹配(0-1)的数目; c 表示在二元向量 X 和 Y 中成功匹配(1-0)的数目; d 表示在二元向量 X 和 Y 中成功匹配(0-0)的数目;

定义 3 一个新的二元向量乘积运算:

$$\begin{aligned} V &= (v_1, v_2, v_3, \dots, v_n) = X | Y \\ &= (x_1, x_2, x_3, \dots, x_n)' | (y_1, y_2, y_3, \dots, y_n)' \\ &= (x_1 | y_1, x_2 | y_2, x_3 | y_3, \dots, x_n | y_n) \quad (5) \end{aligned}$$

式中, $x_i | y_i$ 表示按位或运算, 即只有 x_i, y_i 同时为 0 时, $x_i | y_i = 0$, 其它情况的运算结果都为 1.

二元向量的常用匹配测量方法:

(1) 简单匹配系数(SMC)

$$\text{Sim}(X, Y) = (a + b + c) / (a + b + c + d) \quad (6)$$

SMC 的分子表示: 成功匹配(1-1)、(0-1)以及(1-0)的数目之和; 分母表示: 所有匹配数目的

和, 这种方法主要适用于分析对称二元向量.

(2) Jaccard 系数

$$\text{Sim}(X, Y) = a / (a + b + c) \quad (7)$$

Jaccard 系数不考虑(0-0)匹配, 适用于非对称二元向量.

(3) Dice 系数

$$\text{Sim}(X, Y) = 2a / (2a + b + c) \quad (8)$$

Dice 系数也不考虑(0-0)的匹配, 但是却增加了(1-1)匹配的权重, 这说明提高了(1-1)匹配的重要性.

3.3 二元向量相似性函数的改进

在测试用例集约简问题中, 选择那个测试用例加入测试用例约简集 T' 中, 主要取决于加入该测试用例 $t_i (i = 1, 2, 3, \dots, m)$ 后, 是否使新的测试用例约简集 T' 覆盖度达到最优, 如果达到最优, 就把 t_i 加入 T' 中并把 t_i 从测试用例集 T 中删除, 否则继续寻找, 直至找到最优的测试用例. 其中, 式(1)可以用来计算测试用例的冗余度; 式(2)可以用来计算测试用例的新增覆盖度; 式(1)、(2)、(3)的和可以用来计算总的覆盖度; 式(4)可以用来计算还未被覆盖的测试需求个数; 式(5)用来表示两个测试用例通过运算得到一个新的测试用例.

3.4 算法步骤

GRE-VS 算法的具体步骤如下:

Step1: 根据测试用例和测试需求建立的 $n * m$ 维矩阵关系, 找出测试用例集 T 中所有必不可少的测试用例 $\text{Nec}(T)$, 然后将其加入到测试用例约简集 T' 中后删除, 同时删除冗余测试用例 $\text{Red}(T)$, 如果 T' 为空, 则跳转到 Step2; 如果 T' 不为空, 则把 T' 中的所有测试用例通过式(5)计算得到一个新的测试用例 V , 然后跳转到 Step4;

Step2: 从测试用例集 T 中选择一个对测试需求集 R 覆盖度最大的测试用例 $V (V = \{\max(R(v)) | v \in T\})$ (若有多个, 则随机选择其中一个) 加入测试用例约简集 T' 中, 然后把 V 从 T 中删除, 再跳转到 Step4;

Step3: 选择新增覆盖度最大的测试用例 t_j , 如果 t_j 有多个, 则比较他们的冗余度, 选择冗余度小的测试用例 t_j , 然后把 t_j 加入 T' 中, 再从 T 中删除 t_j 、新增覆盖度为 0 以及从 R 中删除已经满足的测试需求, 最后重新建立测试用例和测试需求的 $m * n$ 维矩阵关系并跳转到 Step1;

Step4: 通过二元向量相似性函数的匹配公式: 计算出 V 与剩余测试用例集 T 中的每个测试用例的

新增覆盖度、总的覆盖度以及冗余度,此时,如果式(6)的 $\text{Sim}(X,Y)=1$,则跳转到 Step5,否则跳转到 Step3;

Step5: $\text{Sim}(X,Y)=1$ 说明达到测试需求的全覆盖,则结束循环.

图 1 是 GRE-VS 的算法流程图.

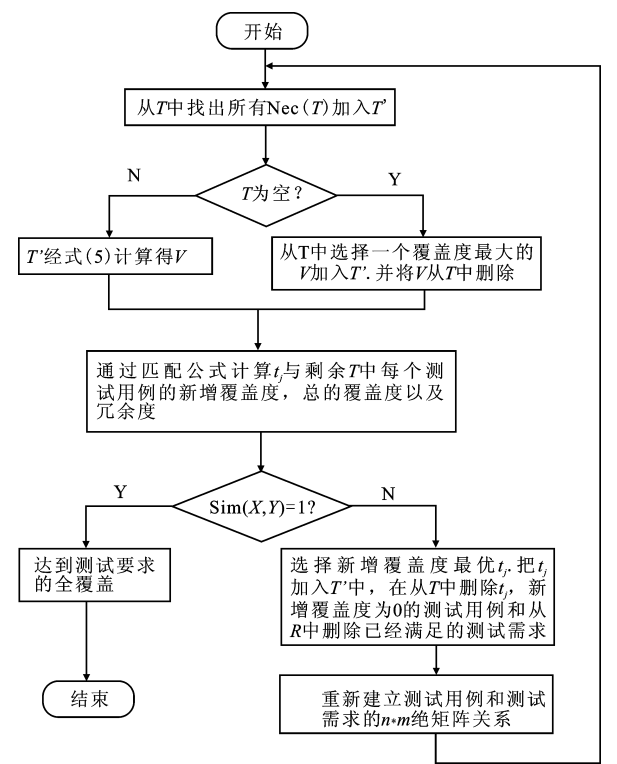


图 1 GRE-VS 流程图

4 实验实例及结果对比分析

为了验证向量相似度方法在测试用例集约简方面的可行性以及向量相似度算法在测试用例集约简过程中的效率,表 1 中给出了测试用例和测试需求之间的 $n \times m$ 维矩阵关系,其中表格中的“*”表示该测试用例满足对应的测试需求;“ ”表示该测试用例不能满足对应的测试需求.从表 1 中可以看出:首

表 1 测试用例与测试需求的关系

T	R													
	r ₁	r ₂	r ₃	r ₄	r ₅	r ₆	r ₇	r ₈	r ₉	r ₁₀	r ₁₁	r ₁₂	r ₁₃	r ₁₄
t ₁	*				*							*		*
t ₂			*						*				*	
t ₃				*			*	*			*	*		
t ₄								*	*		*			
t ₅	*					*				*			*	
t ₆		*			*		*							
t ₇				*				*			*			
t ₈	*									*		*		*
t ₉		*					*							
t ₁₀	*			*	*			*			*		*	

先,通过实验生成了 14 个测试需求,然后又生成了与测试需求相对应的 10 个测试用例,使用这 10 个测试用例对这 14 个测试需求进行测试,最后,通过 JAVA 语言实现了本文提出的 GRE-VS 算法和传统的 GRE 算法代码,并做了 GRE 算法和本文提出的基于向量相似度的 GRE-VS 算法的对比实验.

通过该实验可以证实:本文提出的基于向量相似度的 GRE-VS 算法可以实现测试用例集的约简,表 2 是 GRE 算法和 GRE-VS 算法在进行测试用例集约简过程中产生的实验对比结果.

表 2 测试用例集约简结果

结果	冗余测试用例 Red(T)	必不可少测试用例 Nec(T)	测试用例约简集 T'	总的冗余度 Num
GRE			t ₂ , t ₅ , t ₁₀ , t ₁ , t ₆	6
	t ₇ , t ₉	t ₂ , t ₅	t ₂ , t ₅ , t ₁₀ , t ₃ , t ₆ , t ₈	11
			t ₂ , t ₅ , t ₁₀ , t ₃ , t ₁ , t ₆	11
			t ₂ , t ₅ , t ₁₀ , t ₆ , t ₈	6
GRE-VS	t ₇ , t ₉ → t ₄ , t ₈ , t ₁₀	t ₂ , t ₅ → t ₆	t ₂ , t ₅ , t ₃ , t ₆ , t ₁	5

从表 2 可以看出:在使用基于向量相似度的 GRE-VS 算法对测试用例进行约简之后,最终仅仅需要 5 个测试用例就可以满足所有的测试需求,这样就将原来的 14 个测试用例约简到了 5 个,从而缩减到原来的 35.7%;在没有对测试用例进行约简之前,测试需求总的冗余度是 23,但是,经过使用本文提出的 GRE-VS 算法约简后,最优测试用例约简集 T' 总的冗余度是 5,从而缩减到原来的 20.8%,由此可以得出:本文的 GRE-VS 算法的约简效果非常好.

但是,为了确保实验结果的准确性,又进一步做了测试用例数量为:20、40、80、100,而与之相对应的测试需求数量为:24、160、400、600 的对比实验,表 3 是 GRE-VS 算法和 GRE 算法的实验对比结果.

表 3 GRE-VS 和 GRE 的对比结果

测试用例数量	约简后 T' 中的测试用例数量		总的冗余度 Num	
	GRE-VS	GRE	GRE-VS	GRE
20	7	8	17	24
40	20	22	271	312
80	36	40	727	802
100	40	44	1 586	1 761

从表 2 和表 3 对比结果能得出以下结论:通过使用本文提出的 GRE-VS 算法找出的冗余测试用例和必不可少测试用例都要比使用 GRE 算法多;GRE 算法一般会得到多组可能的测试用例约简集 T',因此得到的

很有可能不是最优或近似最优约简集 T' , 而 GRE-VS 算法一般会得到一组测试用例约简集 T' , 而且它很有可能是最优或近似最优约简集 T' ; GRE-VS 算法与 GRE 算法相比: 约简后测试用例集的数量平均降低约 10%, 而测试用例集总的冗余度平均降低约 13%. 图 2 和图 3 是根据表 3 结果所绘制的折线图:

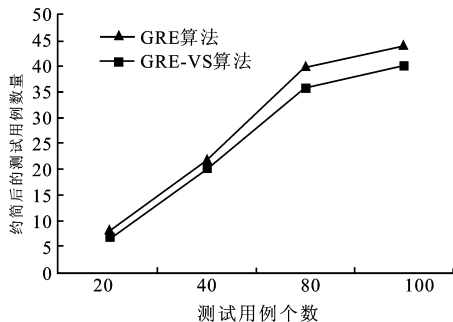


图2 测试用例个数和约简后的测试用例数量的对应关系

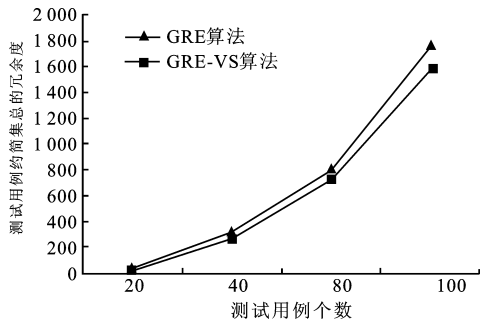


图3 测试用例个数和测试用例约简集总的冗余度对应关系

根据图2和图3的结果可以得出如下结论: 对同样数量的测试用例集进行约简后发现, 使用 GRE-VS 算法要比使用 GRE 算法能得到更少的测试用例数量; 通过对比使用这两种算法得到的测试用例约简集总的冗余度发现, 使用 GRE-VS 算法得到的总的冗余度数量要小于使用 GRE 算法, 尤其是随着测试用例和测试需求的增加, GRE-VS 算法对降低测试用例约简集的冗余度要比 GRE 算法的效果更好。

4 结束语

传统的 GRE 算法首先是选择必不可少测试用例, 然后删除所有的冗余测试用例, 最后用贪心算法对测试用例集进行约简, 直至所有测试需求都满足为止, 该算法只是在一开始进行了必不可少测试用例的选择和删除所有冗余测试用例的操作。但是, 本文提出的基于向量相似度 GRE-VS 算法的思想是: 在每次循环、重新建立测试用例和测试需求的关系后, 都要进行了必不可少测试用例的选择和删除所有冗余测试用例的操

作, 这样就会使测试用例和测试需求的数量迅速降低, 从而实现了将问题规模快速减小的目的, 进而, 最终实现对所有测试需求都覆盖的目标。所以, GRE 算法和 GRE-VS 算法相比, GRE 类似一个静态算法, 而 GRE-VS 相类似一个动态算法。

由于 GRE-VS 算法是使用矩阵存储数据, 所以存储空间开销大; 在一些特殊情况下, 该算法有可能陷入局部最优解。因此, 下一步工作主要是降低 GRE-VS 算法的空间复杂度和尽量避免陷入局部最优解。

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 Transaction on Software Engineering and Methodology, 1993, 2(3):270-285.
- [2] 聂长海, 徐宝文. 一种最小测试用例集生成方法[J]. 计算机学报, 2003, 26(12): 1690-1695.
- [3] Zhang K, Chen H, Liu S, et al. Application of vector similarity method in multi-plan optimization [C]// IEEE International Conference on Information & Automation. China, Macau, 2015:335-338.
- [4] Chen T Y, Lau M F. A new heuristic for test suite reduction[J]. Information and Software Technology, 1998, 40(5/6):347-354.
- [5] Chen S T Y, Lau M F. On the divide-and-conquer approach towards test suite reduction[J]. Information Sciences, 2003, 152(1):89-119.
- [6] Chen T Y, Lau M F. Dividing strategies for the optimization of a test suite [J]. Information Processing Letters, 1996, 60(3): 135-141.
- [7] 马雪英, 盛斌奎, 叶澄清. 用遗传算法的测试用例最小化[J]. 计算机科学, 2007, 34(1): 285-288.
- [8] Michael W Berry, Murry Browne. Understanding search engines mathematical modeling and text retrieval[R]. USA, Mishawaka: University of Tennessee, 1999:31-39.
- [9] William B Frakes, Ricardo Baeza-yates. Information retrieval: data structures and algorithms[M]. USA, Prentice-Hall Inc, 2009: 420-441.

作者简介:

刘 锋 男, (1962-), 教授, 硕士生导师。研究方向为并行计算与云计算。

李 朋 男, (1991-), 硕士研究生。研究方向为程序分析、云计算和智能软件。

朱二周(通讯作者) 男, (1981-), 副教授, 硕士生导师。研究方向为虚拟化与程序分析。E-mail: eezhu@ahu.edu.cn.