

一种混合测试用例约简和生成的方法

李 益, 吴军华

(南京工业大学 计算机科学与技术学院, 江苏 南京 211816)

摘 要: 软件测试是软件开发的重要的环节, 如何实现路径覆盖、提高测试效率是测试研究的关键问题. 从测试需求着手, 通过路径分析进行测试需求的约简; 利用条件概率判定条件语句间的相关性以检测不可达路径; 综合遗传算法和萤火虫算法, 根据层接近度和分支距离设计适应度函数, 根据萤火虫位置的更新进行遗传操作, 从而减少冗余的测试用例, 提高算法收敛性. 将所提方法应用于基准程序, 并与同类方法比较可知, 该方法生成路径覆盖测试用例具有高效性.

关键词: 路径覆盖; 需求约简; 不可达路径; 遗传算法; 萤火虫算法

中图分类号: TP311

文献标识码: A

文章编号: 1000—7180(2018)06—0017—05

An Approach Hybridized Test Case Reduction and Generation

LI Yi, WU Jun-hua

(College of Computer Science and Technology, Nanjing Tech University, Nanjing 211816, China)

Abstract: Software testing plays a vital role in software development life cycle. How to achieve path coverage, increase testing efficiency is the heart of the matter. This paper starts with test requirements for test requirements reduction by path analysis, and checks the dependency among conditional statements by calculating conditional probability to detect infeasible paths in program. In addition, this paper combines genetic algorithm(GA) with firefly algorithm(FA), calculates fitness function by branch distance and layer approach, considers the location update of FA as genetic operation of GA, thus reducing redundant test cases and improving the astringency of the algorithm. The proposed method is applied to benchmark and industrial programs, and is compared with other alternative approaches. The experimental result demonstrates that the proposed method is efficient in generating test cases for path coverage.

Key words: path coverage; requirement reduction; infeasible path; genetic algorithm; firefly algorithm

1 引言

软件测试是保证软件质量的有效手段. 软件测试任务复杂并且测试代价较大. 目前, 已经有很多人使用智能算法对自动化生成测试用例进行了研究. Mann M 等使用遗传算法和模拟退火算法组合自动生成测试数据^[1]; 在文献[2]中, Lu L 等在上下文感知应用程序中使用蚁群算法来自动生成测试用例; 文献[3]使用 ROBDD 图和 PSO 算法提出一种测试用例自动生成的最佳方案. 以上这些

方法都极大地提高了测试的效率, 但是这些方法大多只考虑了测试用例间的关系, 因此它们的计算复杂度相对较高, 有时甚至都不能得到最佳的测试用例集.

在软件测试中, 测试需求的规模通常与测试用例的数量和测试执行的效率息息相关. 如果测试用例的生成存在偏差, 例如覆盖不完全等, 就会导致测试需求约简不完备^[4-5].

本文提出了一种测试需求约简和测试用例生成优化方法. 图 1 给出本文体系结构流程图.

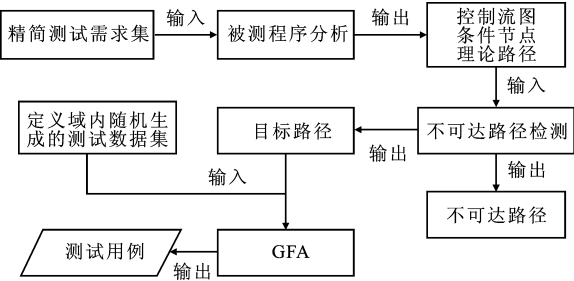


图 1 体系结构流程图

2 测试需求集的获取和约简

测试需求是软件系统中能被一个或多个测试用例验证的条目或事件. 测试需求的约简能够减少测试过程的盲目性, 提高测试效率.

测试需求的约简基于控制流模型来实现, 通过控制流图(Control Flow Graph, CFG)判断相邻测试需求间的关系, 由此得到精简的测试需求集.

定义 1 测试需求集 $R = \{r_i\} (i = 1, \dots, 15)$, 其中 r_i 表示 CFG 中相邻节点的边.

每一个测试需求对应一组变量的范围, 满足该范围的用例即表示该测试需求, 由此得到整个测试程序的测试需求集.

测试需求间关系包括等价关系 ($\equiv$) 和包含关系 ($\subseteq$). 如图 2 所示, 执行 r_2 必定会执行 r_3 , 执行 r_3 也必定执行过 r_2 , 则称 $r_2 \equiv r_3$; 执行 r_2 必定执行过 r_1 , 但执行 r_1 了未必会执行 r_2 , 则称 $r_2 \subseteq r_1$.

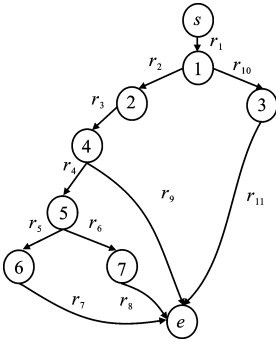


图 2 示例控制流图

定义 2 若 $a \equiv b$ 或 $b \subseteq a$, 则称 a 可以约简 b , 表示为 $a \supseteq b$.

对于图 2 所示的 CFG 中的需求集 R 及每个需求间的关系, 采用如下测试需求约简算法对测试需求集 R 进行约简.

测试需求约简算法流程如下.

输入待测程序, 执行的每个语句相当于一个需求. 从第二个语句开始, 判断与前一个语句的关系,

确定是否能被约简. 算法伪代码如下.

Algorithm 1

```
01      int i = 2;
02      Set all = {1, 2, ..., end};
03      Set contained = {};
04      while(i < end) { //i直到语句结束
05          if (isBranchStart(i)) { //分支开始
06              i  $\supseteq$  i - 1;
07              contained.put(i - 1);
08          }
09          else if (isBranchEnd(i)) { //分支结束
10              i  $\supseteq$  i + 1;
11              contained.put(i + 1);
12          }
13          else { //顺序执行
14              i  $\supseteq$  i - 1;
15              contained.put(i - 1);
16          }
17          i++;
18      }
19      result = all - contained;
```

根据上述算法 1, 可以得到约简后的测试需求集为 $\{r_7, r_8, r_9, r_{11}\}$. 由此可以看出, 该基于控制流图的测试需求约简算法将原本的 11 个测试需求约简成了 4 个测试需求, 极大地约简了测试需求集的规模, 减少测试用例生成所需的计算工作量, 提高测试的效率.

选取 6 个不同类型和难度的程序作为被测程序, 其中包括 Triangle 程序^[6]、Postcode 程序^[5]以及 Smoker detector 程序^[7]等, 这些程序被广泛应用于评价不同测试方法的性能. 将它们约简前后的测试需求数进行对比, 实验对比图如表 1 所示.

表 1 测试需求约简对比

原需求数	19	23	32	47	54	65
约简后需求	5	9	14	16	20	22

根据表 1 可以看出, 上述的测试需求约简方法, 可将测试需求约简一半以上; 同时, 原测试需求数越多, 约简效率越高.

3 基于遗传萤火虫算法的测试用例生成

遗传算法 (Genetic Algorithm, GA) 在测试用例生成中已有多种实践^[8-10], 但其局部搜索能力差, 容易过早收敛. 本文提出了一种融合 GA 和萤火虫算法 (Firefly Algorithm, FA) 的测试用例生成方法, 首先检测出待测程序的不可达路径, 将所有可达路径作为待覆盖目标路径, 通过算法生成覆盖所有目标路径的测试用例集.

3.1 不可达路径检测

在软件测试中, 路径覆盖是一种常用的测试充分性准则, 该准则要求采用合适的方法生成测试数据以覆盖给定的目标路径, 但不可达路径的存在为路径覆盖测试数据生成问题的求解带来了很大问

题. 因此,有必要采用合适的方法,及时检测程序包含的不可达路径,从而提高软件测试的效率. 在检测出所有的不可达路径之后,在后续的路径覆盖测试中,考虑的目标路径仅为可达路径,因此本文将路径覆盖测试数据生成问题转化为融合 GA 和 FA 的测试数据生成问题.

设被测程序 G 如图 3 所示,及其所对应的控制流图如图 4 所示.

```
s      void example (int a,int b){
1      int c, w;
2      if (a>0) {
3          c = a;
          }
4      else {
          c = b;
          }
5      if (a + b - c > 0) {
6          w = a - b;
          }
7      else {
          w = b - a;
          }
8      if (w > 0) {
9          system.out.println (TRUE);
          }
10     else {
          system.out.println (FALSE);
          }
e      }
```

图 3 示例程序 G

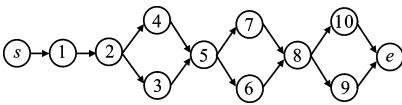


图 4 G 的控制流图

$$p[z_1][z_2]\cdots[z_m]=\frac{\sum_{k=1}^N(z_1^k)\lambda_1(1-z_1^k)^{1-\lambda_1}(z_2^k)\lambda_2(1-z_2^k)^{1-\lambda_2}\cdots(z_m^k)\lambda_m(1-z_m^k)^{1-\lambda_m}}{\sum_{k=1}^N(z_1^k)\lambda_1(1-z_1^k)^{1-\lambda_1}(z_2^k)\lambda_2\cdots(1-z_{m-1}^k)^{1-\lambda_{m-1}}}$$

(3)

式中, m 表示条件个数; $z_i \in \{0,1\}; i \in \{1,2,3\}$. 当 $z_i = 1$ 时表示为每条用例执行时通过第 i 个条件语句节点,当 $z_i = 0$ 时表示不通过该节点.

综上,通过判断图 4 中条件语句的相关性及通过式(2)和式(3)计算两两条件语句相关性概率和三个条件语句相关性概率,可得如下结果如表 2 和表 3 所示.

表 2 两两条件语句相关性概率

$\begin{matrix} k \\ (i,j) \end{matrix}$	(2, 5)	(2, 8)	(5, 8)
0	0	0. 678 6	0. 452 4
1	0. 363 6	0. 181 8	0. 5

定义 3 条件语句 s_i 和 s_j 的相关性

$$R_{s_i,s_j}=i,j,i,j\in\{0,1\}$$

(1)

式中, $i,j = 1$ 时,条件语句取真分支; $i,j = 0$ 时,条件语句取假分支.

1. 若 $i,j = 1,1$,则语句对 (s_i,s_j) 具有真-真相关性;
2. 若 $i,j = 1,0$,则语句对 (s_i,s_j) 具有真-假相关性;
3. 若 $i,j = 0,1$,则语句对 (s_i,s_j) 具有假-真相关性;
4. 若 $i,j = 0,0$,则语句对 (s_i,s_j) 具有假-假相关性.

定义 4 条件语句两两相关性概率

$$p_{i,j}(k)=\begin{cases}\frac{\omega_1}{\varphi}, & k=0 \\ \frac{\omega_2}{N-\varphi}, & k=1\end{cases}$$

(2)

式中, φ,ω_1 和 ω_2 分别为统计量 $\Phi=\sum_{k=1}^NZ_i^k,\Omega_1=\sum_{k=1}^NZ_i^k\sum_{k=1}^NZ_j^k$ 和 $\Omega_2=\sum_{k=1}^N(1-Z_i^k)\sum_{k=1}^NZ_j^k$ 的取值. $k \in \{0,1\}$, k 取 1 或取 0 分别表示条件语句 i 执行条件下,条件语句 j 执行或不执行的概率.

定义 5 多条件语句的相关性概率

表 3 三个条件语句相关性概率

$[z_1][z_2][z_3]$	000	001	010	011
ρ	0. 285 7	0. 714 3	NaN	NaN
$[z_1][z_2][z_3]$	100	101	110	111
ρ	1	0	0. 875	0. 125

如由表 2 和表 3 可知,条件语句对 (2,5) 存在假-假相关性;三个条件语句 (2,5,8) 具有真假-假相关性. 由此可得示例程序 G 的不可达路径有如下几条:

- $P_1=s,1,2,3,5,7,8,9,e$
- $P_2=s,1,2,4,6,7,8,10,e$
- $P_3=s,1,2,4,6,7,8,9,e$

3.2 基于遗传萤火虫算法的测试用例生成方法模型

基于遗传萤火虫算法 (Genetic Firefly Algorithm, GFA) 的测试数据自动生成方法思想是:假设待覆盖目标路径数量为 m , 目标是寻找至少 m 个测试数据以覆盖所有的 m 条路径. 也就是找到使得 $f_1(X), f_2(X), \dots, f_m(X)$ 分别取得最小值的解, 由此, 寻找覆盖目标路径的测试数据生成问题就转化为如下多目标优化问题:

$$\min(f_1(X), f_2(X), \dots, f_m(X)) \tag{4}$$

由上式可以看出, 如果存在 X , 满足 $f_i(X) = 0$, 则 X 就是覆盖目标路径的测试数据.

GFA 算法中将 GA 中的适应度函数 $f(X)$ 作为 FA 中萤火虫的亮度函数, FA 中萤火虫的位置更新看成是 GA 中的遗传操作.

3.2.1 初始化种群

根据变量定义域, 随机产生初始种群. 待测程序输入参数个数为 n , 一组变量就是覆盖一条路径的测试用例. 以这些数据为输入, 运行插桩后的待测程序, 易得到初始覆盖的路径集 S .

在图 4 中被测程序分析得到的理论路径集 E 中除去由不可达路径检测得到的不可达路径集 I , 剩下的即为目标路径集 T , 即:

$$E = I \cup T \tag{5}$$

T 中减去初始覆盖的路径集 S 后即待覆盖的目标路径集 TD . 无重复保留能够覆盖 S 中路径的测试用例作为最终测试用例集的一部分.

3.2.2 适应度函数设计

计算以个体 X 为输入所覆盖的路径 $P(X)$ 与目标路径 P_0 之间的接近程度, 记为 $f_i(X)$, 并以此作为适应度, 用层接近度和分支距离这两个指标来衡量

$$f_k(X) = \text{appr}_{p_k}(X) + \text{dist}_{p_k}(X) \tag{6}$$

适应度函数以 X 为输入覆盖的路径 $P(X)$ 与目标路径 P_0 之间的接近程度记为 $f_k(X)$, 并以此作为萤火虫的亮度. 通常地, $f_k(X)$ 越小, 则萤火虫亮度越大, 表明该值越优, 亦即 $P(X)$ 与目标路径越接近; 特别地, $f_k(X) = 0$, 则 X 即为期望的测试数据.

3.2.3 遗传操作

遗传操作包括萤火虫的个体选择和位置更新.

假设萤火虫 i 比萤火虫 j 的亮度大, 则萤火虫 j 需向萤火虫 i 移动, 萤火虫 i 对萤火虫 j 的吸引力为

$$\beta_{ij} = \beta_0 * \exp(-\gamma r_{ij}^2) \tag{7}$$

式中, β_0 为最大吸引力, γ 为光吸收系数, r_{ij} 为萤火虫 i 到萤火虫 j 的笛卡尔距离.

在每个萤火虫各自的决策半径内, 对每个萤火虫进行两两比较, 以确定移动方向, 如图 5 所示.

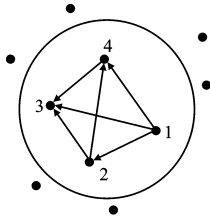


图 5 萤火虫移动方向

以图 5 为例, 计算出每个点 (即每个萤火虫) 在各自决策半径内可能移动的方向组成有向图, 并且将每个点的入度加入到选择次数数组 $C[k] (k = 1, 2, \dots, m)$ 内, 如表 4 所示.

根据数组 $C[k]$ 计算个体选择概率

$$p_k = \frac{\sum C[k]}{\sum * (m - 1)} \tag{8}$$

式中, $\sum$ 为数组 $C[k]$ 中所有元素之和. 如表 4 所示.

表 4 选择次数及对应概率

编号	1	2	3	4
$C[k]$	0	1	3	2
p_k	6/18	5/18	3/18	4/18

再将个体选择概率转换成对应的“饼状图” (m 个小块), 如图 6 所示, 其中每个小块大小对应每个个体概率所占比例.

最后, 随机生成一个 $(0, 1)$ 之间的数, 若在 $(\sum_{j=1}^{i-1} p_j, \sum_{j=1}^i p_j]$ 范围内, 则选中第 i 块进行位置更新. 若选取的萤火虫 j 为决策半径内最亮的点时, 则更新萤火虫 j 本身, 即随机选取 j 的第 k 个变量 $(1 \leq k \leq n)$ 进行变换, 通过 $\pm \Delta$ 看其对应亮度值的变化, 变大则更新 $x_k (x_k = x_k \pm \Delta)$, 否则不变.

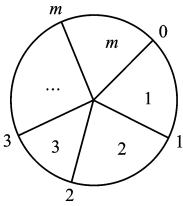


图 6 个体选择概率

将选取的萤火虫 j 向其决策半径内最亮的点萤火虫 i 移动, 其位置更新公式为

$$X_j = X_j + \beta_{ij} (X_i - X_j) + \alpha * \left(\text{rand} - \frac{1}{2} \right) \quad (9)$$

式中, X_i 和 X_j 为萤火虫 i 和 j 所处的位置; $\alpha \in [0,1]$ 为步长因子;rand 为 $[0,1]$ 上服从均匀分布的随机因子.

3.2.4 算法终止判定

检查是否找到覆盖所有目标路径的测试用例集,或者是否达到预先设定好的最大迭代次数. 若已覆盖所有目标路径或者尚未找到覆盖所有目标路径的测试用例集,但是已经达到最大迭代次数,算法终止,输出测试用例集.

3.2.5 算法步骤

使用 GFA 算法分成路径覆盖测试用例的步骤如下:

Step 1:根据精简测试需求集,在定义域内随机生成初始测试用例集,作为初始种群,并记录路径覆盖情况;

Step 2:检查是否满足算法终止条件,若满足,则转至 Step 5;

Step 3:以个体为输入运行插桩后的被测程序,计算个体适应度函数;

Step 4:遗传操作,即萤火虫的选择和位置更新操作,产生新一代种群,转 Step 2;

Step 5:停止进化,输出测试用例集.

4 实验

为了验证方法的有效性,本文设计了实验方案,并给出实验结果和分析. 所有实验程序均采用 Java 语言编写,并在 eclipse 中运行.

实验选取待测程序为三角形分类程序,设定不同的种群规模、最大迭代次数、变量取值范围,将本文提出的 GFA 与文献[11]提出的 F-method 方法进行了对比. F-method 方法使用不可达路径检测模型,结合遗传算法,优化测试用例生成效率. 实验数据如表 5 所示.

表 5 三角形分类程序实验对比

种群规模	定义域	GFA		F-method	
		迭代次数	运行时间	迭代次数	运行时间
50	(0, 100]	60.11	132	106.17	294
100	(0, 100]	36.07	90	63.74	187
100	(0, 500]	313.65	2 886	482.52	3 676

通过对比 GFA 和 F-method 的实验数据可以看出,种群规模越大,平均迭代次数越小;定义域越大,平均迭代次数越大. 通过对比 GFA 和 F-method

可以发现,种群规模和定义域相同情况下,GFA 比 F-method 的平均迭代次数更少,效率更高,说明本文方法对测试数据的生成效率有所改进.

5 结束语

在软件测试中,测试需求的数量直接影响着测试用例的生成和测试执行的效率,测试需求的约简有助于减少测试用例产生时的计算工作量并提高测试的效率. 软件测试中的路径覆盖是一种常用的测试充分性准则,本文提出的方法就是通过控制流图的中相邻需求间关系来约简测试需求集,采用条件概率的方式找出所有不可达路径,在排除不可达路径后,以所有可达路径条件为目标路径,考虑生成覆盖所有目标路径的测试用例,再综合遗传算法和萤火虫算法改进测试用例生成的方法,力求更快更好地生成覆盖目标路径的测试用例集.

参考文献:

[1] Mann M, Sangwan O P, Tomar P, et al. Automatic goal-oriented test data generation using a Genetic algorithm and simulated annealing[C]// International Conference Cloud System and Big Data Engineering. China,Chengdu, IEEE, 2016:83-87.

[2] Lu L, Zeng M. Ant colony optimization based test case generation method in context-aware applications [J]. Journal of Computational Information Systems, 2015, 11(10):3641-3648.

[3] Kalaei A, Rafe V. An optimal solution for test case generation using robdd graph and pso algorithm[J]. Quality & Reliability Engineering International, 2016, 32(7):2263-2279.

[4] Gong D, Yao X. Automatic detection of infeasible paths in software testing[J]. Software Iet, 2010, 4 (5):361-370.

[5] 夏春艳, 张岩, 宋丽. 基于节点概率的路径覆盖测试数据进化生成[J]. 软件学报, 2016, 27(4):802-813.

[6] 巩敦卫. 测试数据进化生成理论及应用[M]. 北京:科学出版社, 2014.

[7] Mcminn P. Evolutionary search for test data in the presence of state behaviour[J]. University of Sheffield, 2005 (12) :41-46.

[8] Aleti A, Grunske L. Test data generation with a Kalman filter-based adaptive genetic algorithm[J]. Journal of Systems & Software, 2015, 103(C):343-352.

5 结束语

标准的CSO在寻优能力上优于其他的优化算法,但因其小鸡粒子容易陷入局部最优,使算法易出现早熟停滞现象^[12]。虽然在处理低维度函数问题时,能较快地收敛到全局最优,但在处理高维多峰函数时,CSO常常陷入局部最优无法跳出。本文模拟人类扬长避短的优点,让公鸡粒子向优秀个体学习的同时,在自身停滞不前的情况下向较差粒子反向学习以跳出局部最优。通过实验验证,改进算法在寻优能力,稳定性,高效性等方面均得到了提高。

参考文献:

- [1] 李振璧,王康,姜媛媛. 基于模拟退火的改进鸡群优化算法[J]. 微电子学与计算机, 2017, 34(2):30-33.
- [2] 李俊,汪冲,李波,等. 基于扰动的精英反向学习粒子群优化算法[J]. 计算机应用研究, 2016, 33(9): 2584-2587.
- [3] 张建春,康凤举,梁洪涛,等. 基于鸡群优化的粒子滤波算法研究[J]. 系统仿真学报, 2017, 29(2):295-300.
- [4] Chen S, Yan R. Parameter estimation for chaotic systems based on improved boundary chicken swarm optimization[C]// International Symposium on Optoelectronic Technology and Application 2016. Beijing, China, International Society for Optics and Photonics, 2016:101571K.
- [5] 孔飞,吴定会. 一种改进的鸡群算法[J]. 江南大学学报(自然科学版), 2015, 14(6):681-688.
- [6] 黄霞,叶春明,郑军. 混合改进搜索策略的鸡群优化算法[J/OL]. (2017-03-04). [2017-07-05]. <http://kns.cnki.net/kcms/detail/11.2127.TP.20170304.1727.004.html>.
- [7] 肖亮,刘思彤. 基于认知多样性变异的鸡群算法协同优化异步实现[J]. 计算机科学, 2017, 44(6A):99-104.
- [8] Chen Y L, He P L, Zhang Y H. Combining penalty function with modified chicken swarm optimization for constrained optimization[C]// First International Conference on Information Sciences, Machinery, Materials and Energy. Pairs: Atlantis Press, 2015: 1899-1907.
- [9] 崔东文. 鸡群优化算法-投影寻踪洪旱灾害评估模型[J]. 水利水电科技进展, 2016, 36(2):16-23.
- [10] 徐仪勋,李旺,李东东,等. 基于改进鸡群算法的非侵入式家电负荷分解[J]. 电力系统保护与控制, 2016, 44(13):27-32.
- [11] 夏学文,刘经南等. 具备反向学习和局部学习能力的粒子群算法[J]. 计算机学报, 2015, 38(7):1397-1407.
- [12] Hedar J. Test functions for unconstrained global optimization[EB/OL]. [2017-07-05]. http://www-optim.amp.i.kyoto-u.ac.jp/member/student/hedar_files/Test_GO_files/Page364.htm.

作者简介:

张慕雪 女, (1992-), 硕士研究生. 研究方向为智能优化算法、数据挖掘. E-mail: zhangmuxue@foxmail.com.

张达敏 男, (1967-), 博士, 教授. 研究方向为优化计算、网络拥塞控制.

杨菊靖 女, (1993-), 硕士研究生. 研究方向为数据挖掘.

朱陈柔玲 女, (1993-), 硕士研究生. 研究方向为数据挖掘.

(上接第21页)

- [9] Khan R, Amjad M. Automatic test case generation for unit software testing using genetic algorithm and mutation analysis[C]// IEEE up Section Conference on Electrical Computer and Electronics. Jaipur, Rajasthan India, IEEE, 2015:1-5.
- [10] Khan R, Amjad M, Srivastava A K. Generation of automatic test cases with mutation analysis and hybrid genetic algorithm[C]// International Conference on Computational Intelligence & Communication Technology. Ghaziabad, India, IEEE, 2017:1-4.

- [11] 丁蕊,董红斌,张岩,等. 基于关键点路径的快速测试用例自动生成方法[J]. 软件学报, 2016, 27(4): 814-827.

作者简介:

李益 女, (1993-), 硕士研究生. 研究方向为软件测试. E-mail: 249401266@qq.com.

吴军华 女, (1965-), 博士, 副教授, 硕士生导师. 研究方法为软件工程.