The New Approach for Software Testing Using a Genetic Algorithm Based on Clustering Initial Test Instances

Roya Alavi\textsuperscript{1*} and Shahriar Lotfi\textsuperscript{2†}

\textsuperscript{1}Computer Engineering Department, Islamic Azad University, Zanjan Branch, Zanjan, Iran
\textsuperscript{2}Computer Science Department, University of Tabriz, Tabriz, Iran

Abstract. A software system testing includes a large set of test cases. Test selection helps to reduce this cost by selecting a small subset of tests that are likely to reveal faults. The test selection helps to reduce cost by selecting a small subset of tests that find to faults. The aim is to find the maximum faults of program using minimum number of test instances. In this paper, the compound algorithm KMGA is used in Siemens programs and the results got better in comparison to the earlier methods.

Keywords: k-means algorithm, clustering, Genetic algorithm and Software testing.

1. Introduction

The process of testing any software system is an enormous task which is time consuming and costly software testing spends almost 50\% of software system development resources. Random test simply runs the inputs and then clarifies the performed structures but it can't extract some of the accessible information from black box [1]. Dynamic test white and black box methods are used in combination to produce finite amount of test instances [2]. Testing involves three main steps: generation a set of test inputs, execution thos inputs on the program under tests, and then checking whether the test executions reveal faults [3]. Testing all input is impossible so we need to choose a subset of tests. The clustering is to determine the main grouping in a set of unabled data, where the objects in each group are indistinguishable under some criterion of similarity. Clustering is an unsupervised classification process fundamental to data mining [4]. Two evaluation metrics were used for to estimate the best test cases: rank and distance metrics [5]. This reveal that it may be move appropriate to use a subset rather than all the ways at hand to evaluate the fault detection capability (of test cases). Until the program is performed with a specific input, the availability of faults leads to the unexpected results.

The rest of the paper is arranged as follow: problem; the basic concepts about software testing, the earlier strategies used in software testing are presented. Proposed method, experimental results and Conclusion is presented.

2. The Problem

An input matrix with \(n\) rows and \(m\) columns are considered as test instances in which \(m\) expresses the number of available components in each sequence of test and \(n\) is the number of test instances. For example in triangle problem, the length of test sequence or \(m\) must be 3. Each element could be number or character but if element is character should be considered decimal equivalent of character as the input of matrix. The output must be a numerical matrix.

\textsuperscript{*} Corresponding author. \textit{E-mail address:} roya.alavi.2009@gmail.com.
\textsuperscript{†} Corresponding author. \textit{E-mail address:} shahriar_lotfi@tabrizu.ac.ir
3. Background

The methods of designing test dynamically are divided into two parts. Black box test is necessary in output evaluation in response to the input data [6,7,8,9]. The analyses of boundary amount in black box test indicates that test instances in which boundary conditions are specified have better results. The white box test is known as logical cover or structural test. There are three approaches to study cluster validity. The first is based on external criteria. The second approach is based on internal criteria and third approach of clustering validity is based on relative criteria, which consist of evaluating the results (clustering structure) by comparing them with other clustering schemes [4,10]. Genetic algorithms choose subsets of population that are called parents. The rules of offspring combination is based on crossover which includes the internal change of values with specific variables and the operation depending on the random variation of value that is called mutation [11,8,12,13]. The input to the algorithm is described as a strain of 0, 1 that called genes.

4. Related Work

The developers run program using one test and observe faults in the program. Later they choose another failing test instance and repeatedly find faults using a symbolic error finding and observe the situation until there is a faulty situation [14,15]. The capability of test is of great importance in designing and a framework including “design” phase has been presented [16]. Baudry suggested the class diagrams and state diagrams are the main model which needs to be analyzed [17]. Earlier for the reason of time and budget shortage, tests were prioritized [5]. The analyses of test point based on black box test formulate three elements: software system, test method and the level of usefulness [18]. Software testers often gather enormous amount of data for a software system which is going to be tested. One can use it to produce automatic test instances randomly using control graph [13,19,20,21,22].

5. KMGA

Totally, the aim of software test is to design a set of minimum number of test instances and clarify the maximum faults [23]. The number of other test cases be developed to achieve some predefined goal of reasonable testing [6]. The objective of measure in clustering is to minimize this measure as we want to minimize the within-cluster scatter and maximize the between-cluster separation in other words, the internal data of groups are pressed and the groups become thoroughly distinct [4,10]. In Fig. 1, internal evaluation of group and external evaluation between two groups are shown as Intra and Inter, respectively [10].

![Fig. 1: Internal and External clustering criteria](26)

The proposed method is KMGA (K-Means Genetic Algorithm) which is a combination of k-means and Genetic algorithms. Among different clustering techniques, k-means method was used. In k-means algorithm, test cases are clustering according to the internal criteria (Intra). The purpose of k-means method is to minimize the sum of distances of every point from their centers [10,24,25]. After clustering, the gained centers are entered to the genetic algorithm as the test instances. The main idea of genetic algorithm is to generate test cases and the aim is focused on this point to generate a better set of these tests [8,11,12,26]. The smaller the number of matrix rows will cause the faster convergence of genetic algorithm [27].
genetic algorithm, the test is chosen when it has the most distance comparing to all previous tests (Inter). If the distance between two input vectors is small, the possibility of sets of faults by each vector is the same [25]. According to the white box and black box testing, the output is designed for each test. KMGA software testing can be seen in Fig. 2. In Fig. 2a, Euclidean distance for internal criteria is calculated as follow by Eq. 1:

$$d(x, c) = \sqrt{\sum_{i=1}^{n} |x_i - c_i|^2} \tag{1}$$

In the Eq. 1 $x_i$ is the $i$th test case and $c_i$ is the center of $i$th cluster. The main problem of clustering algorithm is in the number of clusters which is $K$ and must be determined as a parameter [10].

The $K$-means algorithm is as follows:

1. Assign initial values for cluster means $c_1$ to $c_K$
2. Repeat
3. For $i=1$ to $n$ do
4. Assign each data point $x_i$ to cluster $C_j$ where $\|c_j - x_i\|$ is the minimum
5. End for
6. For $j=1$ to $K$ do
7. Recalculate cluster mean $c_j$ of cluster $C_j$
8. End for
9. Until convergence
10. Return $C$

(a) $K$-means algorithm for internal criteria

The Genetic algorithm is as follows:

1. Initialize(population) // center matrix of $k$-means
2. Evaluate(population)
3. While (stopping condition not satisfied)
4. {
5. Selection(population)
6. Crossover (population)
7. Mutate(population)
8. Evaluate(population)
9. }

(b) Genetic algorithm for external criteria

Fig. 2: KMGA software testing

The number of groups, namely $K$, is determined in the main program, considering the method of conditional decision making in white box test and the number of possible states. In the follow code, the number of possible state is equal 8, because in the 2 line, the number of True and False state is state 2^2 and in 5 line is equal 2^2.

```java
1. Puplic void f (int a,int b)
2.        If( (a>1 )&& (b=0)){
3.                           x= x+1;
4.            }
5.        If ((a==2) ||( b>1)){
6.                           x= x/b;
7.            }
```

The equation of calculating new center in line 7 of Fig. 2a is written in the Eq. 2 that $C_j$ is number of total clusters.

$$c_j = \frac{1}{|C_j|} \sum_{i \in C_j} x_i \tag{2}$$

The 5th line of Fig. 2b Choosing in this method is based random wheel choice. The possibility of being chosen for the population of next generation is $x_i$ which is calculated in Eq. 3:

$$p(x_i) = \frac{f(x_i)}{\sum_{i=1}^{m} f(x_i)} \tag{3}$$

The function $f(x_i)$ in the above equation is the same as Euclidean distance of Eq. 1 that of maximum Euclidean distance used for external criteria. In Line 6 of Fig. 2b indicates crossover. Parent chromosomes are randomly chosen and the crossover possibility is $P_c= 0.9$ (0.0 < $P_c$ < 1.0). In line 7 of Fig. 2b, the operational mutation is in one chromosome that changes to generate an offspring [23,28]. Test cases are chosen randomly with the possibility of mutation $P_m= 0.01$ (0.0 < $P_m$ < 1.0).
6. Evaluation and Experimental Result

The Siemens researchers created a test pool of black box test cases using the category partition method. They then augmented this test pool with manually created white box test cases that in table 1 the statistical information are given about these codes [11,29].

We ran all faulty versions by white box and black box testing and recording their outputs, and compared thos with the expected output [15]. The results of performing algorithm K-means for 4 versions of siemens are provided in table 2. The number of groups after the initial grouping in table 2 indicates the number of initial input population to the genetic algorithm. In table 3 there are the results of genetic algorithm performance. In this table the results of using CBI (Cooperative Bug Isolation) for gathering tests are included. In CBI, using some sets of predicative rules, all tests that contradict to the results of an immense set of tests are chosen [3].

<table>
<thead>
<tr>
<th>program</th>
<th>Fault version</th>
<th>LOC</th>
<th>Test cases</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Print_tokens</td>
<td>7</td>
<td>539</td>
<td>4130</td>
<td>Lexical analyze</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>489</td>
<td>4115</td>
<td></td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>507</td>
<td>5542</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>397</td>
<td>2650</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>299</td>
<td>2710</td>
<td></td>
</tr>
<tr>
<td></td>
<td>41</td>
<td>174</td>
<td>1608</td>
<td></td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>398</td>
<td>1052</td>
<td></td>
</tr>
<tr>
<td>Print_tokens2</td>
<td>5</td>
<td>539</td>
<td>4130</td>
<td>Lexical analyze</td>
</tr>
<tr>
<td>Replace</td>
<td>7</td>
<td>539</td>
<td>4130</td>
<td></td>
</tr>
<tr>
<td>Schedule</td>
<td>9</td>
<td>397</td>
<td>2650</td>
<td></td>
</tr>
<tr>
<td>Schedule2</td>
<td>10</td>
<td>299</td>
<td>2710</td>
<td></td>
</tr>
<tr>
<td>Tcas</td>
<td>41</td>
<td>174</td>
<td>1608</td>
<td></td>
</tr>
<tr>
<td>Tot info</td>
<td>23</td>
<td>398</td>
<td>1052</td>
<td></td>
</tr>
</tbody>
</table>

Table 2: Results run of k-means in Siemens test cases

<table>
<thead>
<tr>
<th>program</th>
<th>Cluster count(k)</th>
<th>Cluster count next of run algorithm</th>
<th>Algorithm Repeat count</th>
<th>Test cases count</th>
<th>Detected fault</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replace</td>
<td>204</td>
<td>12</td>
<td>9</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Schedule</td>
<td>54</td>
<td>48</td>
<td>5</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Schedule2</td>
<td>64</td>
<td>56</td>
<td>5</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Tcas</td>
<td>40</td>
<td>40</td>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

In Fig.3 comparsion done between KMGA test cases and orginal test cases. For example KMGA was coverage 100% of program codes with 20% of test cases and the against, 20% of orginal test cases (Naïve Ranking) coverage to only 30% of programs. In the diagram of Fig. 4, the proposed method is compared with Tarantula, NN (Nearest Neighbor), CT (Cause-Transition).Tarantula method is a colouring way source code statement that to has fault or pass information in any test cases, give to faulty probable for any statement of program [15].
Fig. 3: Comparison KMGA with original test cases

Showing results that KMGA method could recognize faults number with investigation mini size of program lines. In Tarantula method, about 56% of defective versions and their test sequences powerful guidance programmer to fault by few experiment of about 10% of enforceable code, against in KMGA method detected 62% of fault by 1% of test cases that this 1% of test cases coverage 10% of enforceable code. This 62% is a suitable number in comparison with 56% number. A comparison criterion is introduced in Eq. 4 [29]:

\[
fdp = \frac{\text{Num. of suites that can detect Fault F}}{\text{Total Num. of Test Suites Satisfying Criteria}}
\]  

(4)

In the Eq. 4, the numerator expression indicates the number of situations in which faults may occur and the denominator expresses the total test cases [29]. The result obtained from fdp criterion in figure 5 is shown on the basis of table 3 information. For example in Fig. 5a, replace program has fdp equal to 0.014 (83/5542=0.014) and in Fig. 5b it is 0.86 (6/7=0.86) which is a better number comparing to Fig. 5a.

A new criterion is introduced in Eq. 5. In this equation the numerator shows the number of found errors and TP is the representative of test cases that correctly labeled based cases that discover fault. For optimization purposes, the value of M needs to be maximum.

\[
M = \frac{\text{fault number detected}}{TP}
\]  

(5)
In Fig. 6 diagrams, this comparison is quiet clear. For example, in replace program, the M value in the KMGA is 2.4 which is an optimized value in comparison to the value of 0.4 in predicative rule and main tests.

Fig. 6: Comparison four versions with three type test case based M criteria of table 3

7. Conclusions

One of the final purposes of KMGA, is to choose less number of test cases which leads to the fault discovery using dynamic methods. Applying k-means and genetic algorithms, the test cases were used for black box and white box testing non-randomly and carefully to get the most amount of output. In KMGA, the test cases reduced considerably. With 99 percent reduction of cases, only 40 percent of fault decreased. In future, determining K based line of code, the change of function in genetic algorithms and using fuzzy k-means will be improved.

8. References