2011 International Conference on Computer and Software Modeling
IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore

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

Roya Alavi[1*] and Shahriar Lotfi[2†]

[1]Computer Engineering Department, Islamic Azad University, Zanjan Branch, Zanjan, Iran

[2]Computer Science Department, University of Tabriz, Tabriz, Iran

Abstract. A software system testing includes a larg 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 papar, 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.

* Corresponding author. *E-mail address:* roya.alavi.2009@gmail.com.
† Corresponding author. *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, 1that 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].

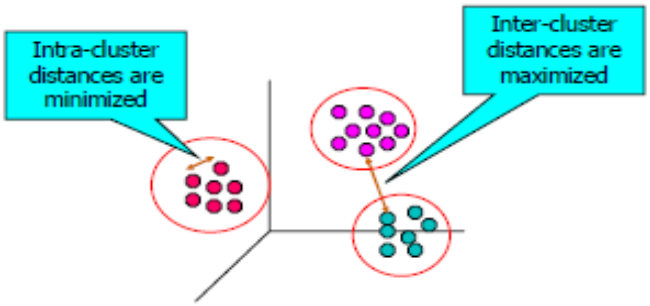

Fig. 1: Internal and External clustering criteria [26]

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]. In

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:

$$d(x,c) = \sqrt[2]{\sum_{i=1}^{v} |x_i - c_i|^2} \quad (1)$$

In the Eq. 1 x_i is the ith test case and c_i is the center of ith 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:	*The Genetic 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*	*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. }*
(a)K-means algorithm for internal criteria	(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.

```
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.

$$c_j = \frac{1}{|C_j|} \sum_{i=c_j} x_i \quad (2)$$

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:

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

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 Pc= 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 Pm= 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 1: Object of analysis

program	Fault version	LOC	Test cases	Description
Print_tokens	7	539	4130	Lexical analyze
	10	489	4115	
Print_tokens2	32	507	5542	Lexical analyze
	9	397	2650	
	10	299	2710	
Replace	41	174	1608	Patternreplacement
	23	398	1052	
Schedule				Priority scheduler
Schedule2				Priority scheduler
Tcas				Altitude separation
Tot_info				Measure Information

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

program	Cluster count(k)	Cluster count next of run algorithm	Algorithm Repeat conut
Replace	204	12	9
Schedule	54	48	5
Schedule2	64	56	5
tcas	40	40	5

Table 3: Results of Genetic algorithm

	Result of GA		CBI		Orginal	
program	Test cases count	Detected fault	Test cases count	Detected fault	Faults	Passed
Replace	7	17	76	26	83	5459
Schedule	9	6	33	6	31	2619
Schedule2	14	8	41	6	32	2678
tcas	7	25	38	26	23	1585
Total	37	56	188	64	169	12341

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].

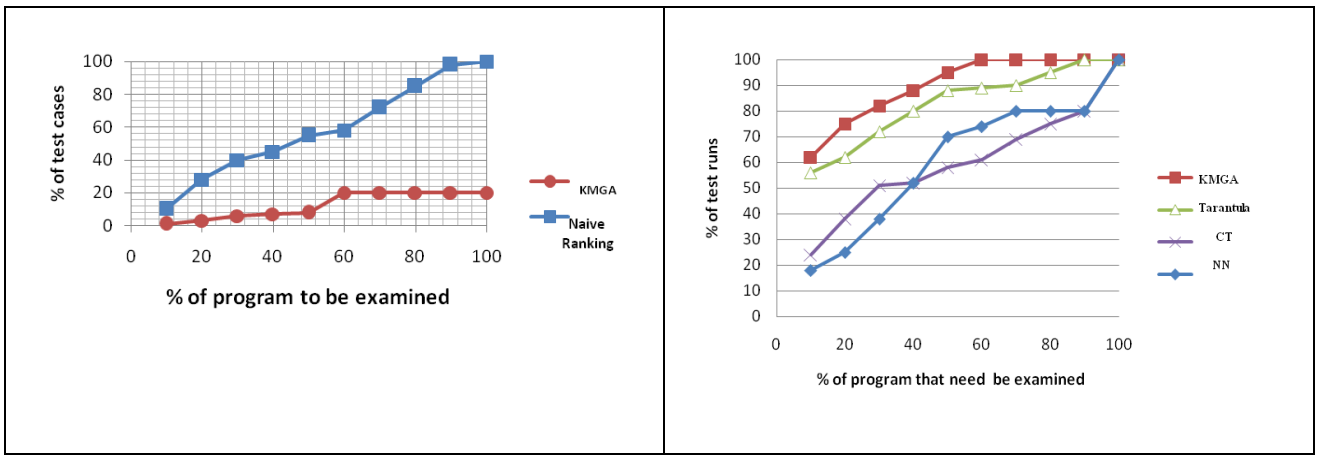

Fig. 3: comparsion KMGA with orginal test cases

Fig. 4: Comparsion effects any one of techniques

Showing results that KMGA method could recognization 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 comparsion with 56% number. A comparison criterion is introduced in Eq. 4 [29]:

$$fdp = \frac{Num.of\ suites\ that\ can\ detect\ Fault\ F}{Total\ Num.of\ Test\ Suites\ Satisfying\ Critria} \tag{4}$$

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.

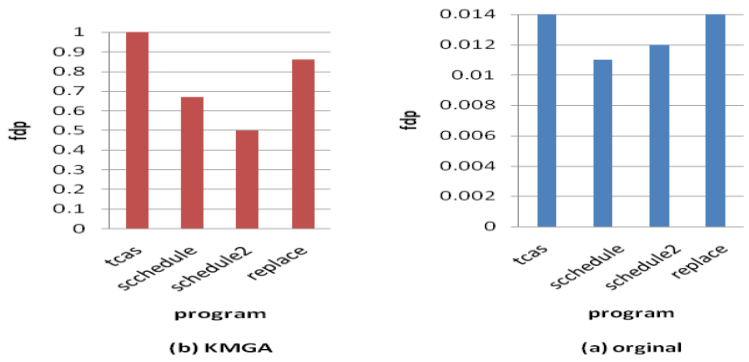

Fig. 5: Comparsion foure versions based fdp criteria of table 3

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 labled based cases that discover fault. For optimization purposes, the value of M needs to be maximum.

$$M = \frac{fault\ number\ detected}{TP} \tag{5}$$

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.

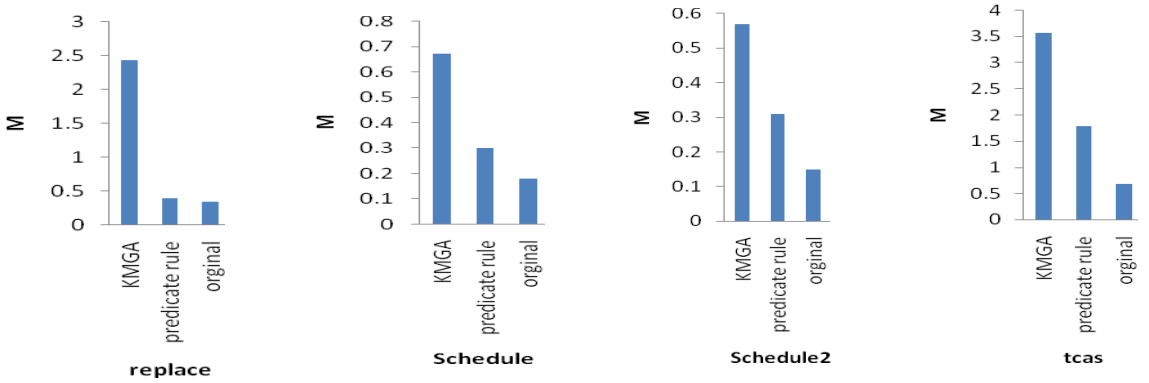

Fig. 6: Comparsion four versions with three type test case based M criteria of table 3

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. Refrences

[1] P. Mcminn. Search-based Software Test Data Generation: A Survey. 2004: 1-58.

[2] H. Y. Chen, T. H. Tse, F. T. Chan and T. Y. Chen. In black and White: An Integrated Approach to Class Level Testing of Object-Oriented programs. *HKU CSIS teach report TR 96-07.* 1997: 1-40.

[3] W. Zheng, M. R. Lyu and T. Xie. Test Selection for Result Inspection via Mining Predicate Rules. 2008: 1-4.

[4] E. Rendon, I. Abandez, A. Arizmendi and E. M. Quiroz. Internal Versus External Cluster Validation Indexes. *International Journal of computers and communications*. 2011: 27-34.

[5] R. C. Bryee and C. J. Colbourn. Prioritized Interaction Testing for Pair-Wise Coverage with Seeding and Constraints. *Information and software technology 48*. 2006: 960-970.

[6] G. J. Myers. The Art of Software Testing. Revised and Updated by Tom Badgett and Todd M.Thomas with Corey Sandler, John Wiley & Sons, Inc, Second Edition. 2004, pp. 1-255.

[7] L. Luo. Software Testing Techniques. Institute for software research international Carnegie mellon university Pittsburgh, PA 15232 USA. 2002: 1-19.

[8] M. R. Girgis. Automatic Test Data Generation for Data flow Testing Using a Genetic Algorithm. *Journal of universal computer science, vol.11.no 6*. 2005(6): 906-914.

[9] Y. K. Malaiya. Antirandom Testing: Gtting the Most Out of Black-Box Testing. *Computer science Dept.Colorado State university, Technical Report CS-96-129*. 1995: 1-14.

[10] Tan, Steinbach, Kumar. Data Mining Cluster Analysis: Basic Concepts and Algorithms. 2004, ch.8,pp. 487-568.

[11] J. T. Alander, T. Mantere, and P. Turunen. Genetic Algorithm Based Software Testing. *Department of Information Technology and Industrial Economics*. 1995: 1-4.

[12] M. Ahmed and I. Hermadi. GA-Based Multiple Paths Test Data Generator. *Computer & operations research 35*. 2008: 3107-3124.

[13] Y. S. Dai, M. Xie, K. L. Poh and B. Yang. Optimal Testing-Resource Allocation with Genetic Algorithm for Modular Software Systems. *The Journal of systems and Software 66*. 2003(66): 47-55.

[14] G. Difatta, S. Leue and E. Stegantova. Discriminative Pattern Mining in Software Fault detection. *SOQUA' 06, Portland, or, USA*. 2006: 1-8.

[15] J. A. Jones and M. Harrold. Empirical Evaluation of the Tarantula Automatic Fault-Localization technique. *ASE' 05, Long Beach California, USA*. 2005: 1-10.

[16] M. Nazir, R. A. Khan and K. Mustafa. Testability Estimation Framework. *International journal of computer applications (0975-8887), volume 2- no. 5*. 2010(5): 9-14.

[17] E. Mulo. Design for Testability in Software Systems. *Faculty EEMCS, TU Delft*. 2007:5-32.

[18] M. Kerstner. Software Test Effort Estimation Methods. 2011: 1-12 .

[19] D.Berndt, J. Fisher, L.Johnson, J.Pinglikar, and A.Watkins. Breeding Software Test cases with Genetic Algorithms. *Proceeding of the 36th Hawaii international conference on system sciences (HICSS03)*, 2002: 1-10.

[20] J. H. Andrews, T. Menzies, and F. H. Li. Genetic Algorithms for Randomized unit Testing. *IEEE transactions on software engineering, vol.1, no.1*. 2001(1): 1-15.

[21] J. Miller, M. Reformat, and H. Zhang. Automated Test data generation using genetic algorithm and program dependences graphs. *Information and software technology 48*.2006: 586-605.

[22] R. P. Pargar, M. Harrold, and R. R. pech. Test-Data Generation Using Genetic Algorithms. *Journal of software testing, verification and reliability*. 1999: 1-9.

[23] P. Srivastava and T. Kim. Application of Genetic Algorithm in Software Testing. *International Journal of software engineering and its applications vol.3.no.4*. 2009(4): 87-95.

[24] P. S. Sandhu, J. Singh, V. Gupta, M. Kaur, S. Manhas and R. Sidhu. A K-means Based Clustring Apporach for finding Faulty Modules in Open Source Software Systems. *World Academy of science, engineering and technology 72*. 2010: 654-658.

[25] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman and A. Y. Wu. An Efficient K-means Clustering Algorithm: Analaysis and Implementation. *IEEE transactions on pattern analysis and machine intelligence, vol.24, no.7*. 2002(7): 881-892.

[26] M. Last, S. Eyal and A. Kandel. Effective Black-Box Testing with Genetic Algorithm. 2005: 1-15.

[27] A. Bouchachia, R. Mittermeir, P. Siclecky, S. Stafiej and M. Zieminski. Nature-Inspired Techniques for Conformance Testing of Object-Oriented Software. *Applied Soft Computing. J*, 2009: 1-16.

[28] T. Takagi, S. Hashimoto, and Z. Furukawa. Operational Profile-based Test Suite Generation using a Genetic Algorithm. *Dept. of Reliability-Based Information system engineering.* 2009:1-2.

[29] L.Zhao. A New Approach for Software Testability Analysis. *ICSE' 06,* Shanghai, china. 2006: 985-988.