Journal of Theoretical and Applied Information Technology
15th October 2018. Vol.96. No 19
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

A COMPARATIVE STUDY OF TEST CASE GENERATION AND OPTIMIZATION USING GENETIC ALGORITHM

[1]ITTI HOODA, [2]PROF. DR. RAJENDER SINGH CHHILLAR

[1]Research Schooler, Department of Comp. Sc. and Applications, M.D.U, Rohtak, Haryana, India
[2]Professor, Department Of Comp. Sc. and Applications, M.D.U, Rohtak, Haryana, India
E-mail:-[1]ittihooda01@gmail.com, [2]chhillar02@gmail.com

ABSTRACT

The main consideration of the work is a comparative study and analysis of test case generation and optimization algorithms based on Genetic Algorithm. The work also present how genetic algorithm shows the better result as compared to other techniques by using UML diagram .The comparison is shown between three different techniques which differently consider the genetic algorithm for the purpose of test case generation and optimization. The survey is being presented using the activity diagram and also the activity graph of the Airline Reservation System. The activity diagram of the airline reservation system is first converted into the activity graph. As any graph is the combination of vertices and edges which are connected with some nodes, in the activity graph the nodes actually represents the test path, the test path of the system is being optimized using the optimization technique or we can say using the Genetic Algorithm. The survey conducted is estimated or analysed on various defined factors which are differing for every technique defined in the study. Three algorithms considered are "Test case generation and Optimization using UML models and Genetic Algorithm", "Optimization of Test case generation using Genetic Algorithm" and "Test case retrieval Model by Genetic algorithm with UML diagram". This work with the empirical results prove that the algorithm for test case retrieval model by Genetic Algorithm with UML diagram shows better results as compared to other discussed techniques.

Keywords:*UML, Genetic Algorithm, Optimization, test path, Test case, Mutation,Crossover,GA.*

1. INTRODUCTION

The software system is the part of modern day to day life at present. As the software systems are now applicable for each and every task of life starting from the kitchen systems to almost all kind typical infrastructures required in life which actually works intangibly, that means for enjoying the life all sort of availability provided for which the software is must part as per the present scenario. As per the fact that the usage and also the importance for using the software system is increasing then the same is to be developed more proficiently. Software development is the long process and consist many stages to make it usable for the user in day to day life. Means the simple looking expression is to be translated into the computer words which actually have some line code or we can say interrelated codes. As per the scenario generated in 1960's which actually termed as the software crisis, by that it is quite clear that to have the software system at large scale, it is not very much prone to the bugs or issues. As per the facts the existing system were not efficient for the tasks arising at that time or they were not able to give the predictable output. [1]

The development of the software have may stages which are combined together as software development lifecycle and almost half of the cost of the software development is consumed in the testing part. The cost of testing the software can be reduced to an extent in the case when the predicted errors or bugs are detected at the early development stages of the software development lifecycle. So as ensure the quality and reliability of the software system testing is the most important stage of the software development stages. The software testing process has the following stages or steps:

1. Test data generation,
2. Test data Execution and
3. Evaluation of the test outcomes.

If the practicality of the concern or the above points is concerned then it is hard to carry out the things manually. For reducing the cost and effort required for the testing the things are supposed to be carried out automatically. Test cases are designed to detect and also to resolve the maximum number of defects or bugs from the software system. Just because of the matter of the time and cost the exhaustive type of testing cannot be adopted software testing.

White box testing and behavioural testing are the majorly used testing techniques. White box testing is also being defined as structural testing which is actually for finding the test data from the internal programming and also deriving the structure of the program. In the case of the black box testing strategy the functionality of the code is being verified and doesn't check the internal structure of the application code. In the case of the structural testing strategy the test paths are being verified for the obtained test data provided as input. The term path coverage is being first generated and then different test scenarios are being obtained. After obtaining the test paths the same are then prioritized and along with that the test data for that is path is being generated, which is then followed by the evaluation of the same.[2]

The focus of the work for the analysis of the three defined techniques for test case generation and optimization using genetic algorithm and the analysis specification is being provided using different factors when applied on same execution module.

The further modules of the paper are as under section 2 defines the previous work in the field of test case generation and optimization, section 3 talks about themethodologies discussed and also about the case used for describing the techniques as airline reservation system with the activity diagram and activity graph of the same, section 4 describes the conclusion and the future work on the basis of the study done.

4. LITERATURE REVIEW

[3] Presented the technique for the purpose of assembling the libraries of the application software for increasing the reusability of the modules and making the availability of the libraries closer which are required most frequently. [4] Defined a novalapproach which actually is based on information retrieval and particularly considers the BM25, function for similarity of the documents, which works for automatically detecting the bugs in the application software and also generates the bug reports. Certain labels are then assigned to the bug report on the basis of like if they are reported in the past or are newly generated.

[5] proposed BLUIR, which works on the open source tool of IR, the IR toolkit provides the basic ground for the IR based lo0calization of the bugs. Author evaluated BLUIR for four different projects with approx. 3400 bugs finding in all. As per the consideration of the application scenario the BLUIR matches outperforms the available state-of-art tool, the results by the BLUIR are efficient in the case also when it is not using the concept of similarity function for document consideration.

[6] describes the facts that how the history of the different versions of the same application software can be used to make the upcoming version as error prone as possible. The bug localization by including the base version as the part of the current development the MAP (Mean Average Precision) for the bug detection has being raised by something around 30%.

[7] in the research a contextual searching system is being proposed for software engineering tasks, the proposal actually is the combination of behavioural data and Information retrieval part. [8] proposed a technique for the generation of the test data and procedure which are for the real time application systems and also the best enablers for the practical aspects for MBT. As the future aspects are also considered means in the proposal the MBT technique is being used sketching the testing teams is done.

[9] in the work author conducted a review of the traceability researchers and found that the student artifacts are the mostly considered representative for the counterparts of the organizations which actually are not validated for the organizational representativeness.

[10] considered the basic level of educational level gaming applications and is actually based on two aspects of research: agile development methodology and user-cantered design (UCD), the development is generally considered for the group age of 7-10 years of children. A continuous communication among the stakeholders of the development is considered, the facility is being provided by the agile methodology extreme programming (XP) which is quite help in the case of the development of the series games for the students. In the communication of the stakeholders the end user of the product are also considered. In the case UCD the complete profile of the user is known or is being identified for the purpose that the developed work meets the desired outcome as per the end user, and also motivation, capabilities and needs of the children are considered in the defined approach.

[11] the work tries to provide a specific, upto-date, introductory information about the process making the test case generation process automated and also the comprehensive and the authoritive concerns are well maintained in the approach. A orchestrated kind of review of the techniques which are quite prominent for the automated test data generation is considered and also the same is being reviewed for the segment of self-standing.

[12] proposed a technique for the performance-based problems using the black-box testing strategy which works automatically. The technique is applied and implemented for the medium ranged application software's, majorly for the insurance company as an open-source software. The problems or errors related to the performance of the system are automatically detected and are considered by the experienced testers for confinement of the same.

[13] described a technique termed as Refoqus which is actually based on the concept of the machine learning, the training data for machine learning process are the test cases or some queries along with the valid and tested results of the same, the refoqus is evaluated with the four-line technique which actually are used for the retrieval of the natural language-based documents.

[14] studied and evaluated the researches done by the various researchers related to the domain of software engineering and considered the concerns of the researchers along with there proposals and analysed all for the betterment of the SR process. [15] analysed reported patches for three existing generate-and-validate patch generation systems (GenProg, RSRepair, andAE) and also present Kali, a generate-and-validate patch generation system that only deletes functionality. [16] analysed the usage of the history of pre-version modules of the same application and also considered the tangible changes that are related with one-another and also the analysis of the results are being compromised by noise and bias.

[17] in the study the formal analysis concept approach and information retrieval are combined together for addressing the open source problems of source code. The textual description of the application software is being depicted by the latent semantic indexing which actually is an information retrieval technique and also used for bug reporting for the specific parts of the software code which are been provided as the ranked list of the elements of the software code.

5. TEST CASE GENERATION

3.1. Activity Diagram

The reviewed technique are applied and analysed on the activity diagram of the airline reservation system. As the modelling is just dynamic for which the activity diagram of the application module is being used [18]. The activity diagram is the combination of the nodes and connected edges or we can say that it is the graphical representation of the task which a proper flows of the activities. The diagram in figure depicts the activity diagram of the airline reservation system and similarly the diagram in figure 2 depicts the activity graph of the same.

Specification of the software module:-

1. Enter arrival and departure dates in Airline Reservation System.
2. Enter client's personal Information and search for the availability.
3. choose flight and add reward points if applicable.
4. Hold reservation and complete the payment information as needed.
5. Select seat/seats and carry-on with the payment and an acknowledgement will be shared registered or entered on E- mail.

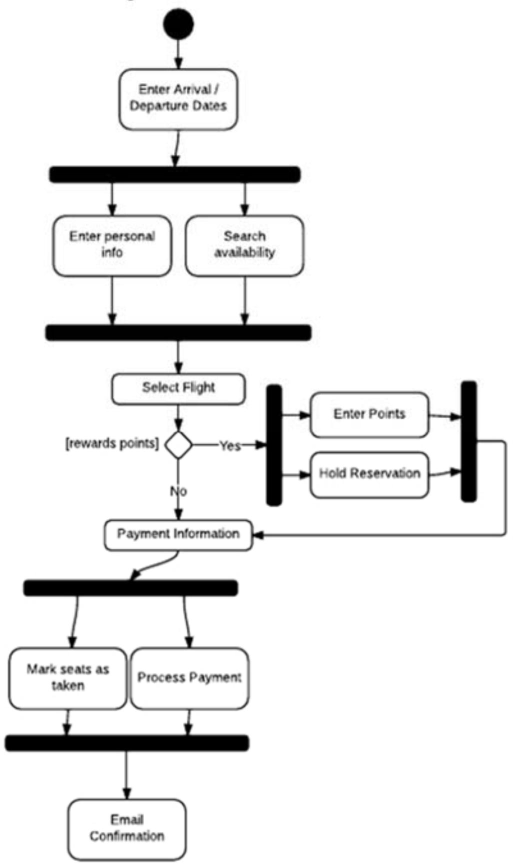


Figure 1. Activity Diagram (AD) for Airline Reservation System.

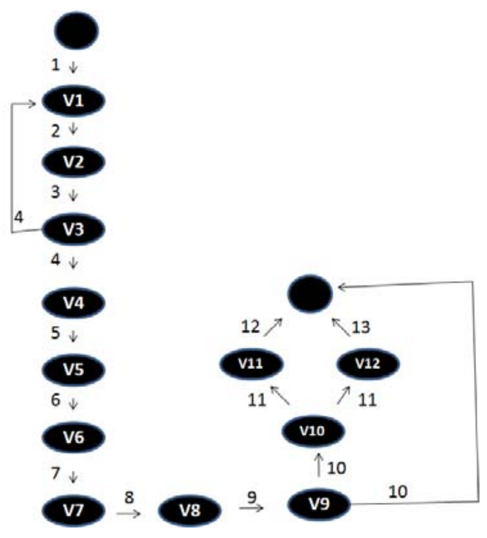


Figure 2. Activity Graph (AG) for Airline Reservation System.

3.2. Test case generation and Optimization using UML models and Genetic Algorithm

The test cases for the software module generated right after gathering all the required information. The optimization of the test cases generated is being done using the evolutionary methodology. The below segment describes the mechanism of obtaining and optimizing the test cases using the genetic algorithm applied on the activity graph. The step by step implementation of the methodology is under [19]:-

Algorithm : GEN-OPT TESTCASES
Input: - Activity Graph (AG)
Output: - Optimized test cases

1. Discover all the scenarios, R= {r1, r2, r3, r4, r5.....} from initial vertex to the final vertex in the Activity Graph.
2. Weights are assigned to scenarios in increasing order from left to right. Individual vertexes are being provided weights as the actual weight of the child vertex which is obtained from weight of the parent node. If in the case any child having more than one parents then weight of that vertex is computed by summing up the weights of the parent's node.
3. Next is to compute the cost (x) of each scenario (path) as the cost of any path is computed by adding the weights of all vertexes on that path.
4. Genetic Algorithm is being applied to the AG.
5. Fitness value computation
    a. For each scenario compute the value of cost(x).
    b. Fitness function is used as F(x)=x*x
    c. Probability is computed for individual as P(i)=F(x)/ΣF(x)

6. For obtaining new generation of solutions best individuals are selected from existing large initial population for mating pool.
    a. Best individual's probability range is grouped into bins, the size of the bin relies on the relative fitness of the solution
    b. Random values are obtained and then verified against the bin where those values related to, choosingthe individuals for the next generation.

7. Crossover is performed on the chromosome pairs by mating two individuals together and applying single point crossover from 4thbit from right.
8. Mutation operation is performed by mutating every third bit from left where the random number generated is less than 0.5.
9. This complete process is rehearsed till the minimization of the fitness value or the maximum number of generations is reached or all the scenarios have been traversed.
10. Test cases are optimized by generating the best scenario as output.
11. End.

The example for the evaluation of the technique is the airline reservation system. The next segment represents the mathematical evaluation of the Genetic Algorithm on the considered example. [19] The possible scenarios generated from the above graph are:-

Scenario1:-
Start:=>V1=>V2=>V3=>V4=>V5=>V6=>V7=>V8=> V9=>end, cost=55

Scenario2:-
Start:=>V1=>V2=>V3=>V4=>V5=>V6=>V7=>V8=> V9=>V10=>V11=>end, cost=78

Scenario5:-
Start:=>V1=>V2=>V3=>V1=>V2=>V3=>V4=>V5=>V6=>V7=>V8=>V9=>V10=>V11=>end,

Scenario3:-
Start:=>V1=>V2=>V3=>V4=>V5=>V6=>V7=>V8=> V9=>V10=>V12=>end, cost=79

Scenario4:-
Start:=>V1=>V2=>V3=>V1=>V2=>V3=>V4=>V5=> V6=>V7=>V8=>V9=>end, cost=64

Scenario5:-
Start:=>V1=>V2=>V3=>V1=>V2=>V3=>V4=>V5=>V6=>V7=>V8=>V9=>V10=>V12=>end, cost=88
These different scenarios become different chromosomes in the population.

Table 1. Fitness Of Initial Population

Scenario No.	Chromosomes		*Y	Probability	Cumulative probability	Associated Probability
1	0110111	5	025	0.0870	0.0870	0-0.2
2	1001110	8	084	0.1750	0.2620	0.2-0.4
3	1001111	9	241	0.1795	0.4415	0.4-0.5
4	1000000	4	096	0.1178	0.5593	0.5-0.6
5	1010111	7	569	0.2177	0.7770	0.6-0.8
6	1011000	8	744	0.2227	1	0.8-1

Table 2. Selection of new generation.

Random No.	Falls into Bin	Selection	Crossover	Mutation
0.8924	6	1011000	1010111	1010111
0.7187	5	1010111	1011000	1011000
0.4376	3	10011111	1000111	1010111
0.6097	5	1010111	1011111	1011111
0.3967	2	1001110	1001000	1011000
0.9126	6	1011000	1011110	1011110

Table3. Fitness Of Initial Population

Scenario No.	Chromosomes	X	X*Y	Probability	Cumulative probability	Associated Probability
1	1010111	7	7569	0.1561	0.1561	0-0.2
2	1011000	8	7744	0.1597	0.3158	0.2-0.4
3	1010111	7	7569	0.1561	0.4719	0.4-0.5
4	1011111	5	9025	0.1861	0.6580	0.5-0.7
5	1011000	8	7744	0.1597	0.8177	0.7-0.9
6	1011110	4	8836	0.1822	1	0.9-1

Figure 4: Selection of new generation.

Random No.	Falls into Bin	Selection	Crossover	Mutation
0.3896	2	1011000	1011000	1001000
0.8846	5	1011000	1011000	1011000
0.4290	3	1010111	1011111	1011111
0.6714	4	1011111	1010111	1010111
0.8761	5	1011000	1011110	1011110
0.9128	6	1011110	1011000	1011000

Table 5:- Fitness Of Initial Population

Scenario No.	Chromosomes	X	X*Y	Probability	Cumulative probability	Associated Probability
1	1001000	2	5184	0.1196	0.1196	0-0.2
2	1011000	8	7744	0.1787	0.2983	0.2-0.4
3	1001111	9	6241	0.1440	0.4423	0.4-0.6
4	1010111	7	7569	0.1747	0.6170	0.6-0.8
5	1011110	4	8836	0.2039	0.8209	0.8-0.9
6	1011000	8	7744	0.1787	1	0.9-1

Table 6. Selection of new generation.

Random No.	Falls into Bin	Selection	Crossover	Mutation
0.4120	3	1001111	1001000	1001000
0.9265	6	1011000	1011111	1011111
0.7635	4	1010111	1011000	1011000
0.3127	2	1011000	1010111	1000111
0.9745	6	1011000	1010111	1010111
0.6793	4	1010111	1011000	1011000

There is a decrement in the difference between the chromosome's value when considered after each and every iteration. Which actually is the representation of the fact of survival of the fittest, the technique ends at the cost value of 88. The further computation shows that the scenario 6 whose cost value is 88 is the optimum path.

3.3. Optimization of Test Case Generation using Genetic Algorithm

[20] The proposed technique is all about the optimization of the work or process, the concept of optimization is being carried out using the concept of Genetic Algorithm. Different project inspection is being considered for the purpose of accommodation of the Genetic Algorithm for optimization. The technique also provides the aspects to have the better results in the efficient time.

The purpose of software testing many researchers have proposed many techniques and assures the better quality of one over another in terms of the quality. For all the techniques there exists some hole of limitation. So as to overcome the issues or limitations in the software testing or specifically saying the issues related to the generation and optimization of the test cases the below study tries to provide better solution for the same for generation and optimization of the test cases.

The evolutionary test data generation techniques are generally good and are for bringing the routinely data which is quite of high quality. The evolutionary techniques are applied on many of the real time application problems. Genetic Algorithm [21] has resulted a good and efficient technique for the optimization and other market related requirements, as per the studies the genetic algorithm is actually a seeking technique. The selection procedure in the GA is based on the real time example of choosing the

man and women who are fit on the basis of the there food selection. Genetic Algorithm can be successfully used with any of the computer software test methodology.

Optimized algorithm

1. Inject the mutant in the program.
2. Generate random test cases.
3. Find the mutation score with the formula, mutation score= (number of mutants found) / (total number of mutants).
4. If the mutant score is satisfactory (Maximum) stop, otherwise go to step 5.
5. Refine the test case using mutation score. Test case, having mutation score 20% or less drops them.
6. Apply Genetic Algorithm Operations on remaining test cases to produce new experiments. Go to step 3.

Algorithm for ab. Where a and b are positive numbers.

```
1.  Power(a, b)
2.  If(a= =1)
3.  Return 1
4.  If(b= =1)
5.  Return a
6.  P=1, i=1
7.  While(i<=b)
8.  {P=P*a
9.  i++}
10. return P
```

Inject four mutants in this program Now algorithm look like this.

```
1.  Power(a, b)
2.  If(a=1)
3.  Return 1
4.  If(b=1)
5.  Return a
6.  P=1, i=1
7.  While(i<b)
8.  {P=P+a
9.  i++}
10. return P
```

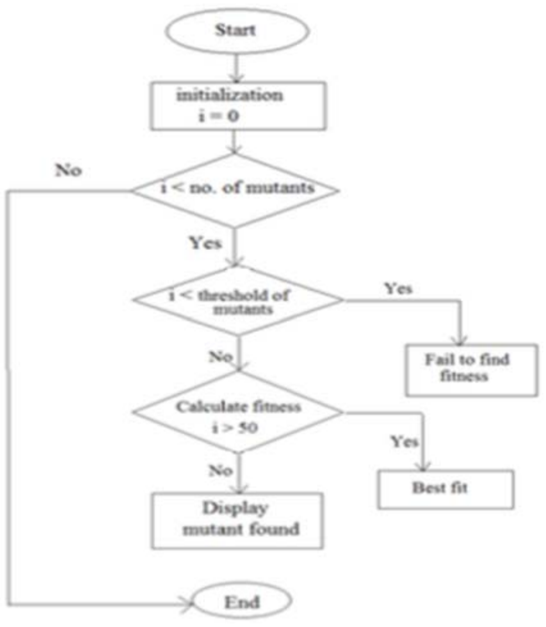


Figure 3. Optimized algorithm flow.

As per indicated in the block diagram of the proposed technique in figure 3, the number of mutants are generated. In the initial stage some of the mutants are added to the program, where the mutants actually defines the type of error and helps to find the optimized test cases. After the addition of the mutants the performance of the test cases is been evaluated which is the initial step of the overall process. In the case when all of the inserted mutants are not detected then query arises that what actually is the number of detected mutants. In the case when the detected number of the mutants are less then that of the minimum number of mutants then the test cases defined are not able to detect the errors properly or just failed.

In the situation of no the fitness function is being computed for estimating the exact number of mutants inserted. In the next case when the number of detected mutants are around fifty percent of the inserted mutants then it the test case is considered as fit and in the case when it is less than 50% then the presentation is all about the number of mutants value. In the case when total number of mutants are detected in the 2nd stage that means the optimal solution is being found and there is no need for going further and also means that the test case is able to detect almost all type of errors in the application software used or is under test.

Just because of the matter of reviewing there is requirement of characterizing the technique which can define the nature of experiment carried. So as to rehash the botches the students are the best propensity. There also exist a case when it is not able consider a class and can also repeat the same for the large part of the module. The related aspects of the experiments are considered as the important point and there is absence of the same in the learning process.

A. Regulated depiction of experiment execution
B. Make particulars of substantial and invalid inputs
C. Understanding directions/level of points of interest
D. Elaborate experiment creation, and not just utilizing the most evident experiment or info
E. Comprehend the motivation behind the framework and current level and connection of testing
F. Characterize a reasonable beginning position for the experiment
G. Understanding test techniques and how to apply them
H. Suppositions, e.g. concerning accuracy and culmination of particulars
I. Experiment assessment (ventures to take to make an unmistakable correlation with expected result ought to be clear)
J. Tidy up after an experiment, repeatability

The above described factors plays an important role for the test cases. These factors are playing very important role in test cases. Further analysis depicts that the elaborating factors have greater influence over the test cases. The elaborating factors have differing impact ratio and are depicted in the table 7.

Table 7. Factors and impact ratio

Factors	Impact(%)
Regulated Depiction	55%
Valid and invalid inputs	38%
Good Detail level	44%
Variation in Test cases	75%
Understand system’s framework	50%
clear beginning position	60%
test design techniques better	80%
accurate suppositions	70%
test case assessment	50%
Tidy up after execution	78%

The system considered for experiments is taken the process of different activities or with the depiction of activities is particular order. For making the way of execution characterized the experiments are considered in small and strides which are extremely itemized which is nearly same as that of code of composing.

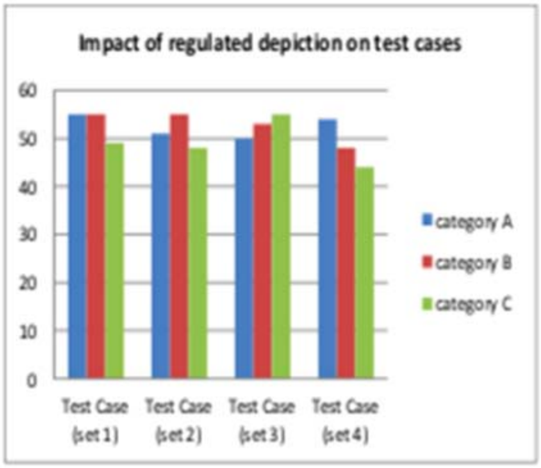


Figure 4. Impact of regulated depiction

Impacts of regulated depiction is being presented in the graph shown in figure 4 for different test cases. The maximum depiction of the impact for the four different sets of test cases is 55%. In the sets of the test cases every of the set contains 3 different test cases where the first regulated depiction of the defined test cases is 55, 55 and 49. In the case of the set 2nd the depiction impact which is regulated are 51, 55 and 48. In the case of the 3rd set the impact of depiction are 50, 53 and 55. The depiction impact for the last set are 54, 48 and 44. Something around the 55% of depiction impacts provides the data which is actually deficient and elements which are subtle so as provide the steps which are trailed unambiguously by the required jumps of the humans and other related personalities those who are taken into consideration at the time of the execution.

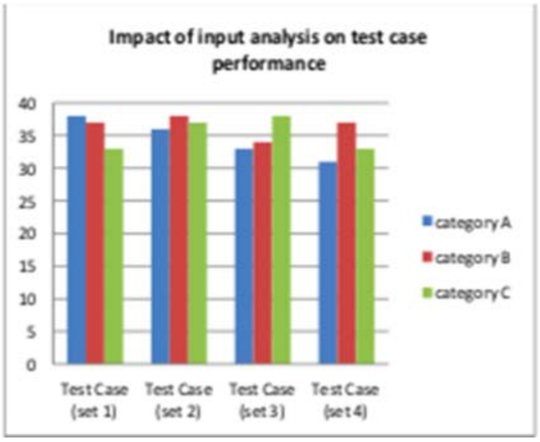


Figure 5. Impact of input analysis

The different test cases are analysed and the graph depicts the representation of the same[29]. From the four different sets of test cases 38% is the maximum impact found for analysis of the input. Similarly every of the set of the test cases contains three different test cases, where the first regulated depiction of the defined test cases is 38, 37 and 33. In the case of the set 2nd the depiction impact which is regulated are 36, 38 and 37. In the case of the 3rd set the impact of depiction are 33, 34 and 38. The depiction impact for the last set are 31, 37 and 33. The things were troubling for the analyser as only 62% were close to the expectation of the examination. The students were at larger extent were selected for getting the information for the examination. The complete data space was not handled by anyone as the impacts remains to be just 38%.

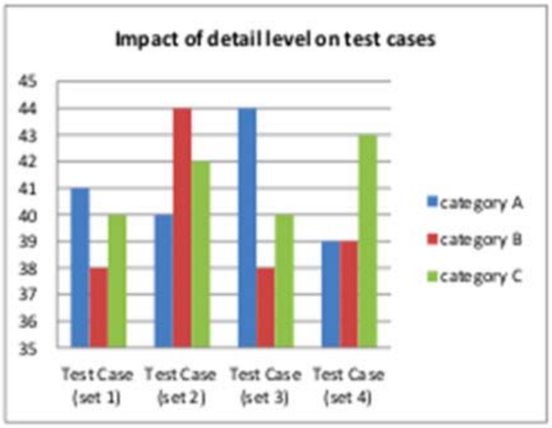


Figure 6. Impact of detail level

The above analysis depicts that the about 29% of the applicants not even read the experimental conveyance and around 15% of the applicants doesn't completed the layout of the experiments, means the impact of depiction is just 44% from the four different sets of the test cases. In the sets of the test cases every of the set contains 3 different test cases where the first regulated depiction of the defined test cases is 41, 38 and 40. In the case of the set 2nd the depiction impact which is regulated are 40, 44 and 42. In the case of the 3rd set the impact of depiction are 44, 38 and 40. The depiction impact for the last set are 39, 39 and 43.

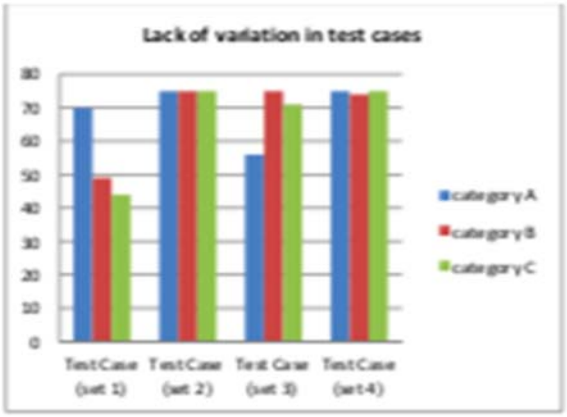


Figure 7. lack of variation

3.4. PROPOSED METHODOLOGY

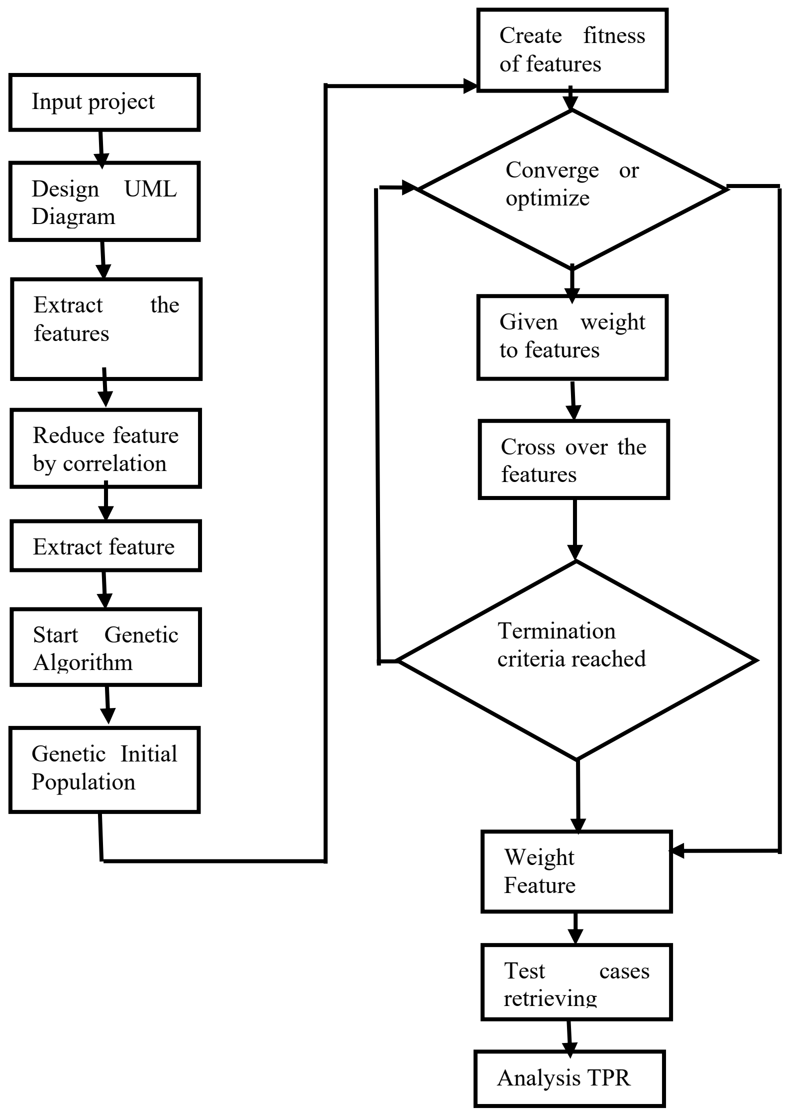


Figure 8. Block diagram of research methodology.

ALGORITHM

Input:

- The UML Diagram[22] to be tested U;
- The flow of graph (CFG) and attributer (ADG) of U;
- The population size (PS)
- Maximum number of Generation (MG);
- Probability of crossover (PX);
- Probability of mutation (PM);

Output:

- TRP of Test Case Retrieval

Begin

<u>Step 1: Initialization</u>

- for i= 1 to PS
- Initialize each feature U_w ϕ;
- Initialize the weight U_w ϕ;

```
    • nRum ← 0;

Step 2: Retrieval test cases
    • While (Test cases not finish)
Begin
    • nRum ← nRum +1
    • for i=1 to PS
    • Put each $U_i$ ← {$w_o$:$w_k$}
    • Current_Population ← Initial_Poplulation
    • No_Of_Generation ← 0;
    • For each individual of current population do
Begin
    • Convert current chromosome according to features;
    • If (current features is optimize) then
    • nfeatures ← nfeatures +1
End If
End For;
    • While (the Best Individual is not Independent feature and No_of_Generations ← MG)

Begin

    • Select set of parents of new population from members of current population using roulette wheel method;
    • Generate New_population using crossover and mutation operations;
    • Current_Population ← New_Population;
    • For each Individual of Current_Population do
      Begin

Given the weight of features
    • nfeatures ← nfeatures + 1

End

    • No_Of_Generation ← No_Of_Generation +1
    • While (Test Case)
    • Weighted features match with use caseDatabase
    • Select Test Cases;
    • Analysis TRP
End while
```

Genetic Algorithm for Feature Selection

The Genetic Algorithm is the part of the heuristic optimization techniques. The solution to the multi-dimensional issues of the optimization is the major usage of the Genetic Algorithm or in the cases where there is no possibility of having the analytic solution.
An optimization problem is denoted as:

$$\bar{x}_{opt} = \arg \operatorname{opt}\{f(\bar{x}) \mid \bar{x} \in C\} \quad (1)$$

where

- f is the cost function,
- C is the set of feasible solutions,
- $\bar{x}$ is a feasible solution and
- $\bar{x}_{opt}$is the optimum weight to features.

In the described approach the set C which is being for the purpose of searching is part to be concerned at most. So as to get the solution which is quite feasible the step-by-step method is considered as one of the most primitive technique for computing the cost function and also for the best solution. This is not a practical scene to be considered. The infinitely defined set is C, The C is finite in the case when the computer is being used because of the fact that in the memory infinite number of combinations are being stored. The number of elements in the set C are always high.

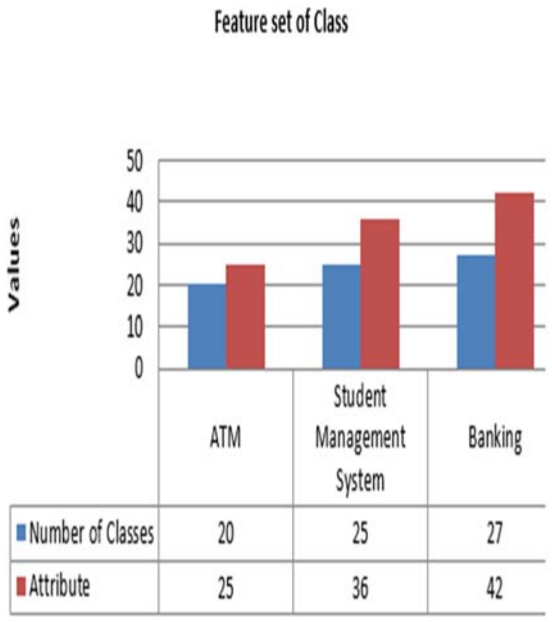
In the defined approach the Genetic Algorithm [23] make use of the natural evolution, in the methodology the chromosomes with highest weight is having more chance of generating new offsprings or to have child[29].

Mutation is being defined as the small alteration in the genetic data. In the weights of the features the changes in the conditions for living are quite easy. Mutation can also prevent from the situation of the degeneration as the optimization deadlock is the analogy of the degeneration.

Using following steps the execution of GA can be described:

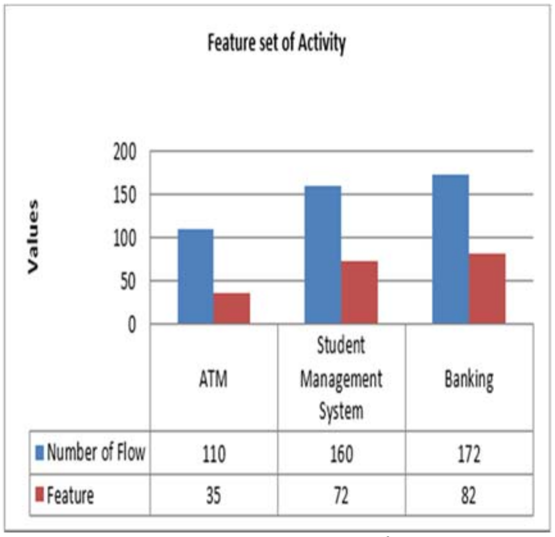
1. Usually random generation is used for the generation of initial population.
2. All features fitness is computed.
3. for parent selection and
4. for offspring generation.
5. By using deletion operator creating new population and in step
6. generated offspring is added.
7. Then mutation takes place.
8. Get back to step 3, in case stopping rule is not satisfied.
9. In the population, the result is the best individual.

Result and Analysis



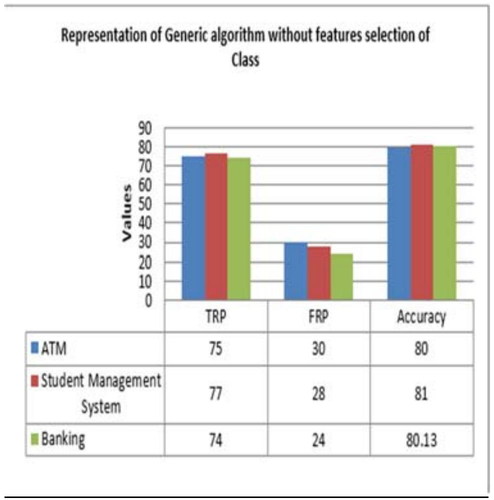
	ATM	Student Management System	Banking
🟦 Number of Classes	20	25	27
🟥 Attribute	25	36	42

Figure 9. Feature set of Class



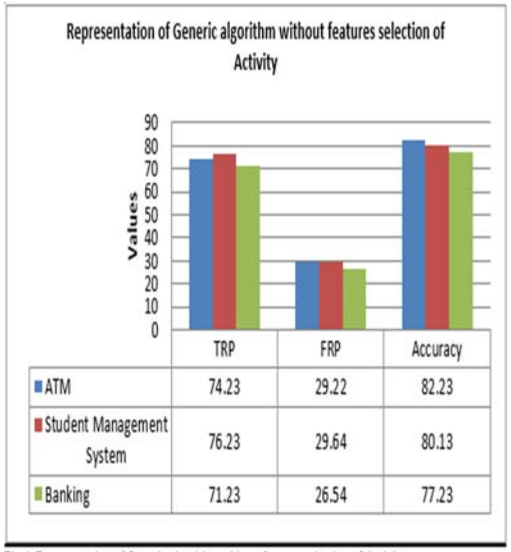
	ATM	Student Management System	Banking
🟦 Number of Flow	110	160	172
🟥 Feature	35	72	82

Figure 10. Features Set of Activity.



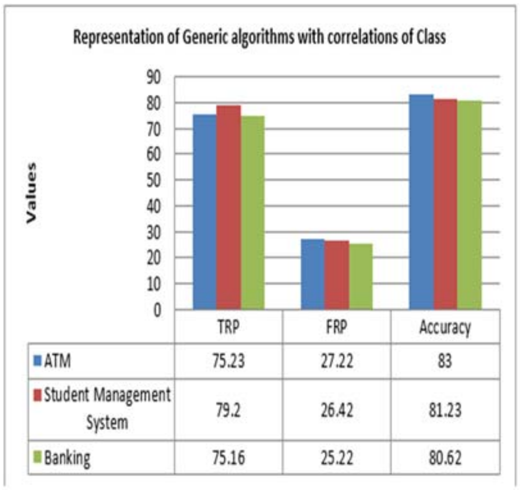
	TRP	FRP	Accuracy
🟦 ATM	75	30	80
🟥 Student Management System	77	28	81
🟩 Banking	74	24	80.13

Figure 11. Representation of Generic algorithm without features selection of class.



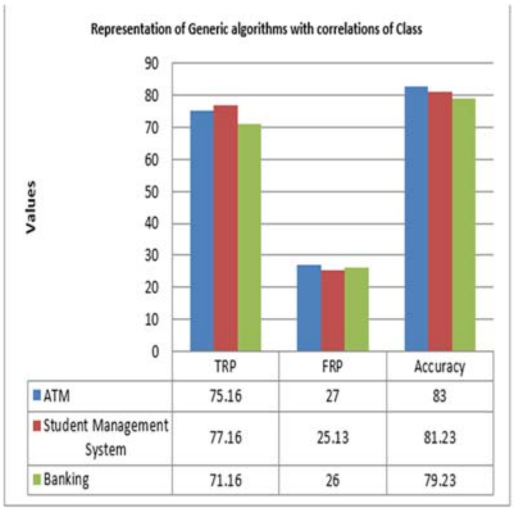
	TRP	FRP	Accuracy
🟦 ATM	74.23	29.22	82.23
🟥 Student Management System	76.23	29.64	80.13
🟩 Banking	71.23	26.54	77.23

Figure 12. Representation of Generic algorithm without features selection of Activity.



	TRP	FRP	Accuracy
🟦 ATM	75.23	27.22	83
🟥 Student Management System	79.2	26.42	81.23
🟩 Banking	75.16	25.22	80.62

Figure 13.Representation of Generic algorithms with correlations of Class.



	TRP	FRP	Accuracy
🟦 ATM	75.16	27	83
🟥 Student Management System	77.16	25.13	81.23
🟩 Banking	71.16	26	79.23

Figure 14. Representation of Generic algorithms with correlations of Class

4. CONCLUSION

Testing is the concept which provides the means of guarantee for the better execution of code of the application software. There are many concerns which are connected with the concept of testing which are considered as the fundamental part of the development cycle, the issues which are attached with the testing part are needed to be resolved at first. The issues can be defined as the compilation error, experimental prioritization, etc., which are considered of the major concern. The major issue with testing of the application software is that how to get the set of cases that bitterly defines almost all aspects of usage of the application software.

The work defined in the paper presents the different techniques for the optimization and generation test cases using the Genetic Algorithm for which three different techniques are used. The discussed and presented techniques are executed on the system module of the airline reservation system whose activity diagram and also the activity graph is presented in the work. The efficiency and also the performance of the proposed methodology and also discussed in the work are analysed and on the basis of the results and discussion of the all three the proposed technique for test case generation and optimization outperforms in the work as it is able to Genetic captures more number of iterations.

The results are efficient and faster as the time consumption is lesser by using GA. Also there is a need to select Fitness function, Best value for Chromosome population, probability for Crossover and Mutation operators in GA which makes it faster. We conclude with the result that correlation show more accuracy than without correlation. In future we can enhance this work with more use cases and optimize by learning and optimization method with GA and can be implemented using neural networks and some more relevant technique to obtain better quality test results by minimizing testing effort.

REFERENCES

[1] Freund, L., & Toms, E. G.,"Contextual search: from information behaviour to information retrieval", *Proceedings of the Annual Conference of CAIS/Actes du congrèsannuel de l'ACSI.*

[2] D. Kundu, M. Sharma, D. Samanta,R. Mall, "System testing for object-oriented systems with test case prioritization", *Journal of Software Testing, Verification and Reliability,* Willey online Library,2009,volume 19, pp. 297-333.

[3] Maarek, Y. S., Berry, D. M., & Kaiser, G. E.,"An information retrieval approach for automatically constructing software libraries", *IEEE Transactions on software engineering*, Vol. 17, No. 8, 1991, pp. 800-813.

[4] Tian, Y., Lo, D., & Sun, C.,"Information retrieval based nearest neighbor classification for fine-grained bug severity prediction", *19th Working Conference on Reverse Engineering,* IEEE, October 2012, pp. 215-224.

[5] Saha, R. K., Lease, M., Khurshid, S., & Perry, D. E.,"Improving bug localization using structured information retrieval",*Automated Software Engineering (ASE), IEEE/ACM 28th International Conference,* November 2013, pp. 345-355.

[6] Sisman, B., &Kak, A. C.,"Incorporating version histories in information retrieval based bug localization", *Proceedings of the 9th IEEE Working Conference on Mining Software Repositories*, June 2012, pp. 50-59.
[7] Wieringa, R. J., "Design science methodology for information systems and software engineering", *Springer,* 2014.
[8] Peleska, J.,"Industrial-strength model-based testing-state of the art and current challenges", *arXiv preprint arXiv:*1303.1006, 2013.
[9] Borg, M.,"Advancing Trace Recovery Evaluation-Applied Information Retrieval in a Software Engineering Context", *arXiv preprint arXiv:*1602.07633, 2016.
[10] Cano, S. P., González, C. S., Collazos, C. A., Arteaga, J. M., & Zapata, S., "Agile Software Development Process Applied to the Serious Games Development for Children from 7 to 10 Years Old", *International Journal of Information Technologies and Systems Approach (IJITSA),* 2015, Vol. 8, No. 2, pp. 64-79.
[11] Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., ...& McMinn, P.,"An orchestrated survey of methodologies for automated software test case generation", *Journal of Systems and Software*, Vol. 86, No. 8, 2013, pp. 1978-2001.
[12] Grechanik, M., Fu, C., &Xie, Q.,"Automatically finding performance problems with feedback-directed learning software testing", *34th International Conference on Software Engineering (ICSE),* IEEE, 2012, June, pp. 156-166.
[13] Bajracharya, S., Ossher, J., & Lopes, C.,"Sourcerer: An infrastructure for large-scale collection and analysis of open-source code", *Science of Computer Programming,* 2014, Vol. 79, pp. 241-259.
[14] Dit, B., Revelle, M., &Poshyvanyk, D.,"Integrating information retrieval, execution and link analysis algorithms to improve feature location in software", *Empirical Software Engineering*, 2014, Vol. 18, No. 2, pp. 277-309.
[15] Cleland-Huang, J., Gotel, O. C., Huffman Hayes, J., Mäder, P., &Zisman, A.,"Software traceability: trends and future directions",*Proceedings of the on Future of Software Engineering, ACM,* 2014, May, pp. 55-69.
[16] Unterkalmsteiner, M., Gorschek, T., Feldt, R., &Klotins, E., "Assessing requirements engineering and software test alignment—Five case studies", *Journal of Systems and Software,* Vol. 109, pp. 62-77.
[17] Herzig, K., Just, S., & Zeller, A, "The impact of tangled code changes on defect prediction models", *Empirical Software Engineering,* 2016, Vol. 21, No. 2, pp. 303-336.
[18] D. Kundu and D. Samant, "A Novel Approach to Generate Test Cases from UML Activity Diagrams" *Journal of Object Technology,* 2009, 8(3), pp. 65-83.
[19] N. Khurana, R.S Chhillar, "Test case generation and Optimization using UML models and Genetic Algorithm",*Procedia Computer Science, Elsevier* Volume - 57, pp, pages 996-1004. (ICRTC 2015)
[20] Ahmed Mateen, Marriam Nazir and Salman Afsar Awan, "Optimization of Test Case Generation using Genetic Algorithm (GA)", *international Journal of Computer Applications* (0975 – 8887) Volume 151 – No.7, pp.6-14, October 2016.
[21] N. Khurana, R.S Chhillar, Usha Chhillar "A Novel technique for Generation and Optimization of test cases using Use Case, Sequence, Activity Diagram and Genetic Algorithm @March 2016 *Journal of Software*, pp 242-250.
[22] Ajay K. Jena, Santosh K. Swain, Durga K. Mohapatra, "Test case generation and prioritization based on UML Behavioral Models", *Journal of Theoretical and applied Information Technology,* Vol 78, No. 3,pp. 336-352.@2015
[23] Nguyen, B. N., Robbins, B., Banerjee, I., &Memon, A.,"GUITAR: an innovative tool for automated testing of GUI-driven software", *Automated Software Engineering*, 2014, Vol. 21, No. 1, pp.65-105.
[24] Choi, W., Necula, G., & Sen, K.,"Guided gui testing of android apps with minimal restart and approximate learning", *ACM SIGPLAN Notices, ACM,*2013, October, Vol. 48, No. 10, pp. 623-640.
[25] Harman, M., "The role of artificial intelligence in software engineering", *Proceedings of the First International Workshop on Realizing AI Synergies in Software Engineering, IEEE,*2012, June, pp. 1-6.
[26] Acher, M., Collet, P., Lahire, P., & France, R. B., "Familiar: A domain-specific language for large scale management of feature

models", *Science of Computer Programming,* 2013, Vol. 78, No. 6, pp. 657-681.

[27] Poshyvanyk, D., Gethers, M., & Marcus, A.,"Concept location using formal concept analysis and information retrieval", *ACM Transactions on Software Engineering and Methodology (TOSEM),* 2014, Vol. 21, No. 4, pp. 23.

[28] Walkinshaw, N., Taylor, R., & Derrick, J.,"Inferring extended finite state machine models from software executions", *Empirical Software Engineering,* 2015, Vol. 21, No. 3, pp. 811-853.

[29] Itti Hooda., Rajender Singh Chhillar., "Test Case Retrieval Model by Genetic Algorithm with UML diagram", *International Journal Of Applied Engineering Research,* Vol 13, No. 7,pp.5167-5174 @2018