Adaptive Genetic Algorithm (AGA) Based Optimal Directed Random Testing for Reducing Interactive Faults

[1]Dr. K. Koteswara Rao [2]Mrs Y Saroja [3]Mr N Ramesh Babu [4] Dr G Lalitha Kumari [5]Mrs Y Surekha
[1]Associate Professor, [5]Asst Professor, [4]Sr Asst Professor
[154] Department of CSE, Prasad V Potluri Siddhartha Institute of Technology, VIjayawada,
[2]Associate Professor, Mallareddy College of Engineering for Women Telangana
[3]Assistant Professor,RGKUT,Srikakulam, Andhrapradesh, INDIA
Email: kkrao@pvpsiddhartha.ac.in, sarojaphani@gmail.com, ramesh.nuthakki@gmail.com,
lalithajoy.nuthakki@gmail.com, yalamanchili.surekha@gmail.com

Abstract - The intension of software testing is to finding the errors in software. Software testing is the process of validating and verifying that a program functions correctly. Random testing generates test inputs randomly from the input space of the software under test. To generate random test cases each time, which will contain some similarity. In order to overcome these issues, we will propose a technique for reducing the faults based on optimal test cases generated from directed random testing. In proposed method, we will generate an efficient random testing test case based on the object behavior dependence model. In this research, the optimal inputs will be generated based on Adaptive Genetic Algorithm (AGA) which will reduce the illegal inputs and equivalent inputs. To reduce the fault proneness, AGA uses the coverage metrics of the test cases. Our proposed method will prunes the input space by combining the previous input with the current one and also increase scalability and effectiveness in the era of software testing.

Keywords: software testing, test case, object behavior dependence model, adaptive genetic algorithm, coverage metrics.

1. Introduction

The black-box testing is an unsystematic testing procedure where programs are investigated by producing unsystematic, autonomous inputs. Random Testing (RT) is the essential approach in the midst of different software testing procedure. It is easy in conception. It's frequently uncomplicated to implement, can frequently work out the software beneath experiment in unpredicted manner, and has established efficiency in identifying malfunction (Zhou et al., 2012). In cryptographic function, random number is extensively utilized as key. Since the protection of key entirely rely on the quantity and arbitrariness of itself. Random numbers are very significant to engender for cryptographic applications (Niu et al., 2014). Random number initiators extensively engaged in business functions do not severely promise these necessities (Dionísio et al., 2014). Though, a foremost confront for these process is the uncertain, two-dimensional, and combinatorial gigantic input gap that they have to discover and train mechanically (Malpani & Bassi, 2012).

The fusion process is used for the production of the time test data generation that obtain the compensation of mutually static and dynamic process was made (Tahbildar & Kalita, 2011). Software testing is enormously expensive movement in the software engineering lifecycle, and as such, its computerization maintains to be of elevated apprehension (Mcminn, 2013). To engender experiments that wrap each and every one of the branches in a group, the group must be initiated, and a process identify progression may require to be engender to locate the entity into a definite position (Fraser et al., 2015). Producing unit test set mechanically is an imperative donation towards enhancing software eminence, and procedures like search-oriented software testing active representative implementation can competently generate experiment set accomplishing elevated code exposure (Galeotti et al., 2013).Model-oriented scheme testing of functions through a GUI front-end to be more gainful and competent contrast to their conventional documentation after that replay counterpart (Darab & Chang, 2014). A variation of RT subsists to progress its effectiveness, such as Adaptive Random Testing (ART) ART is employed to analysis arithmetical programs, derived from break down model which encompass three types: block model, strip model, and point model (Putra & Mursanto, 2013). Though, ART is a smaller amount competent than random testing due to the additional mission of make sure yet dispersal of experiment conditions, where the effectiveness is calculated in provisions of the time to produce an experiment condition (Chow et al., 2013). Random test data production, engender experiment data derived from discriminating arbitrary inputs type a number of allocation. Path-oriented and structural methods utilize the program's organize flow diagram for experiment data production; they choose a path, and utilize a procedure such as representative implementation for production of experiment data. Goal-oriented experiment data production methods pick inputs to carry out the elected target, such as report, condition exposure, decision exposure, irrespective of the path in use (Khan & Nadeem, 2013). The origin for

identification is a test collection, and frequently the obtainable experiment collection is not optimized for generating premium analytical reports. Thus, is significant to produce tests to develop identification. There is lots of existing experiment production procedures Search-Based Software Testing (SBST). Global search algorithms are Genetic Algorithms (Campos et al., 2012).

In software growth, debugging is a prolonged mission. Even though a variety of mechanical methods have been projected, they are not effectual adequate. As well, in physical debugging, developers have obscurity in deciding breakpoints. To tackle these troubles and assist developers establish mistake efficiently, interactive mistake localization structure is employed which merge the remuneration of mechanical methods and physical debugging. Ahead of the mistake is establish, the structure constantly suggest inspection points derived from statements' uncertainties, which are premeditated consistent with the implementation information of experiment conditions and the response information from the developer at previous checking points.

2. Related Works

(Bo Yu & Zeliang Pang, 2012) have projected the enhanced standardized intend approach to intend experiment data for manipulating experiment class was offered. Research illustrate that that the enhanced standardized intend approach was constancy and could reinstate the amalgamation experiment process at the instant for the experiment time being restricted derived from the assumption testing.

(Padgham et al., 2013) have projected a representation oriented oracle production process for unit testing conviction aspiration target mediators. They expand a mistake representation derived from the attributes of the center units to detain the kinds of mistake that may be come upon and identify how to mechanically engender a limited, inactive oracle from the mediator intend representation Line mutually the mistake representation and the oracle production by experimenting 14 mediator scheme. Above 400 concerns were elevated, and these were investigated to establish whether they signify actual mistake or were counterfeit positives. They establish that more than 70 percent of problem elevated were analytical of troubles in either intend or the code. Of the 19 test carried out by their oracle, mistakes were establish by the entire but 5 of these test. They also establish that 8 out the 11 mistake category recognized in their mistake representation revealed at least one mistake.

(Bestoun S. Ahmed et al., 2014) have projected a policy to be utilized for GUI practical testing. The policy engenders the necessary experiment conditions for the GUI beneath experiment by the combinatorial design and afterward it eliminates the unnecessary experiment conditions. The Simplified Swarm Optimization theory was exploited to optimize the experiment conditions allowing for the combinatorial intend perception. By engender the experiment conditions the experiment useful to an actual world case analysis to experiment the efficiency of the policy.

(Andrea Arcuri & Lionel Briand, 2012) have projected a numerous theorems illustrating the possibility of arbitrary testing to distinguish relations mistake and contrast the consequences to combinatorial testing when there are no restriction in the middle of the attributes that can be component of a product. Random testing turns out to be even more effectual as the amount of attributes augment and unite in the direction of equivalent efficiency by combinatorial testing. Specified that combinatorial testing requires note worthy computational above your head in the occurrence of hundreds or thousands of attributes, the consequences recommend that there are reasonable situation in which random testing may surpass combinatorial testing in huge scheme.

(Leandro L. Minku et al., 2014) have projected a theoretical investigation contains sensible suggestion derived from it; they obtain an enhanced EA design. That consist of standardize employees' devotion for dissimilar responsibilities to make sure they are not operational eventually; a fitness task that want less pre-defined limitation and supply an obvious gradient towards practicable resolutions; and an enhanced depiction and alteration operator.

(Junpeng Lv et al., 2014) have projected a fusion method that utilize AT and random partition testing (RPT) in an irregular way. The inspiration for that method was that together policies were engaged such that the fundamental computational complication of AT was condensed by initiating RPT into the experiment procedure devoid of concerning the deficiency recognition efficiency. A condition analysis among seven real-life theme programs was obtainable. The investigational consequences exhibit that fresh policy significantly diminishes the computational above your head of the innovative AT policy but still surpass the uncontaminated RT policy and PT policy in conditions of the amount of experiment cases utilized to distinguish and eliminate a specified amount of deficiency.

(Phil McMinn et al., 2012) have projected a fixed reliance study resultant from program segment that can be utilized to maintain search space lessening. That document describes mutually a hypothetical and experimental examination of the function of this method to open source and manufacturing production code. The consequences supply substantiation to maintain the claim that input field lessening comprises a noteworthy consequence on the presentation of limited, worldwide, and fusion search, whereas a simply random search is unaltered.

(Andrea Arcur, 2016) has projected investigate the responsibility that the extent plays in software testing, particularly branch exposure. We illustrate that, on “hard” software testing standard, longer experiment series construct their testing inconsequential. Thus, they dispute that the alternative of the extent of the experiment series was very significant in software testing. Hypothetical study and experimental studies on extensively utilized standard and on trade software were accomplished to maintain our claims.

(Barus, Arlinta C et al., 2016) have proposed another likeness metric to empower multiclass level testing utilized ART. While creating test contributions, they utilized the similitude metric to compute the remove between two arrangements of articles, and between two successions of strategy summons. They coordinate that metric with ART and applied it to an arrangement of open-source programs, with the experimental come about demonstrating that approach outperforms other RT and ART approaches in OOS testing.

(Syed Abdul Moeed and Niranjan Polala, 2017) explored the different sources of input which was lawful or unlawful by utilizing GSA diminishes the illicit input and identical sources of input. Experiments were created arbitrarily to stay away from the equivocalness and furthermore testing was happen based on GSA wellness that could diminish the deficiencies in the system. The yield result ought to be lawful input that demonstrates the fault rate diminishment.

(Chen, Jinfu et al., 2017) showed a direct request ART algorithm for programming with non-numeric sources of input. The key requirements for utilized ART with non-numeric information sources were a suitable "remove" measure. They utilized the ideas of classes and decisions from classification segment testing to detail such a measure. They examine the disappointment recognition adequacy of our system by play out an observational examination programs, utilized two standard measurements - F-measure and P-measure. Our ART algorithm measurably fundamentally out performs RT, and displays execution like RT on three of the four residual projects. The determination overhead of our ART calculation was near that of R.

3. Problem Definition

In this section briefly discussed the problem definition,

- Scalability and effectiveness is an important problem that needs to be considered while testing and it is a critical issue in the software industry.
- Test cases trigger failures and do not directly uncover faults. Existing random testing will not be that much productive in terms of time consuming and cost.
- Existing ART needs higher computation overhead to achieve the even spread of test cases.
- The existing software fault prediction models require software metrics collected with automated tools and fault data belonging to previous software version or similar software project.
- The main disadvantage of random testing is 1) lengthy test case generation 2) it gives equivalent inputs for test cases 3) it creates many illegal inputs.
- Generating minimal test suites that guarantee t-wise coverage is a highly difficult problem which has been the subject of a great deal of research.

Our main objective in this paper is to generate as many test cases as possible in such a way that they help uncover as many faults as many coverage targets as possible. These test cases must be valid for each time it generates. Another objective is to increase scalability and effectiveness in the era of software testing while delivering comparable failure-detection effectiveness.

Our proposal is important in the field of random software test case generation. In this research, we will generate an efficient random testing test case. It will provide a valid test case for each time test cases will generate. Another importance is to reduce the fault proneness it will use uses the coverage metrics of the test cases.

4. Proposed Method

The most fundamental target of the projected procedure is devoted to the prologue of a competent process for reduce the interactive fault in keeping through the finest experiment condition in the direct random testing. In order to address the effectiveness and scalability of random testing t-wise interaction faults are considered. For the principle of constructing the experiment conditions the fresh procedure efficiently utilize the Object Behavior Dependence Model (OBDM). The resultant inputs are delivering to the Adaptive Genetic Algorithm (AGA) when the experiment condition establishment approaches to a last part. In the Adaptive Genetic Algorithm the finest inputs are engendered which depart an extensive manner in cutting back the prohibited inputs also the indistinguishable inputs, in that way dropping the inclination of being fault-prone. The modus operandi of the projected procedure is gracefully revealed in Figure 1. The block illustration of the new-fangled method is excitingly engraved out in the subsequent segment.

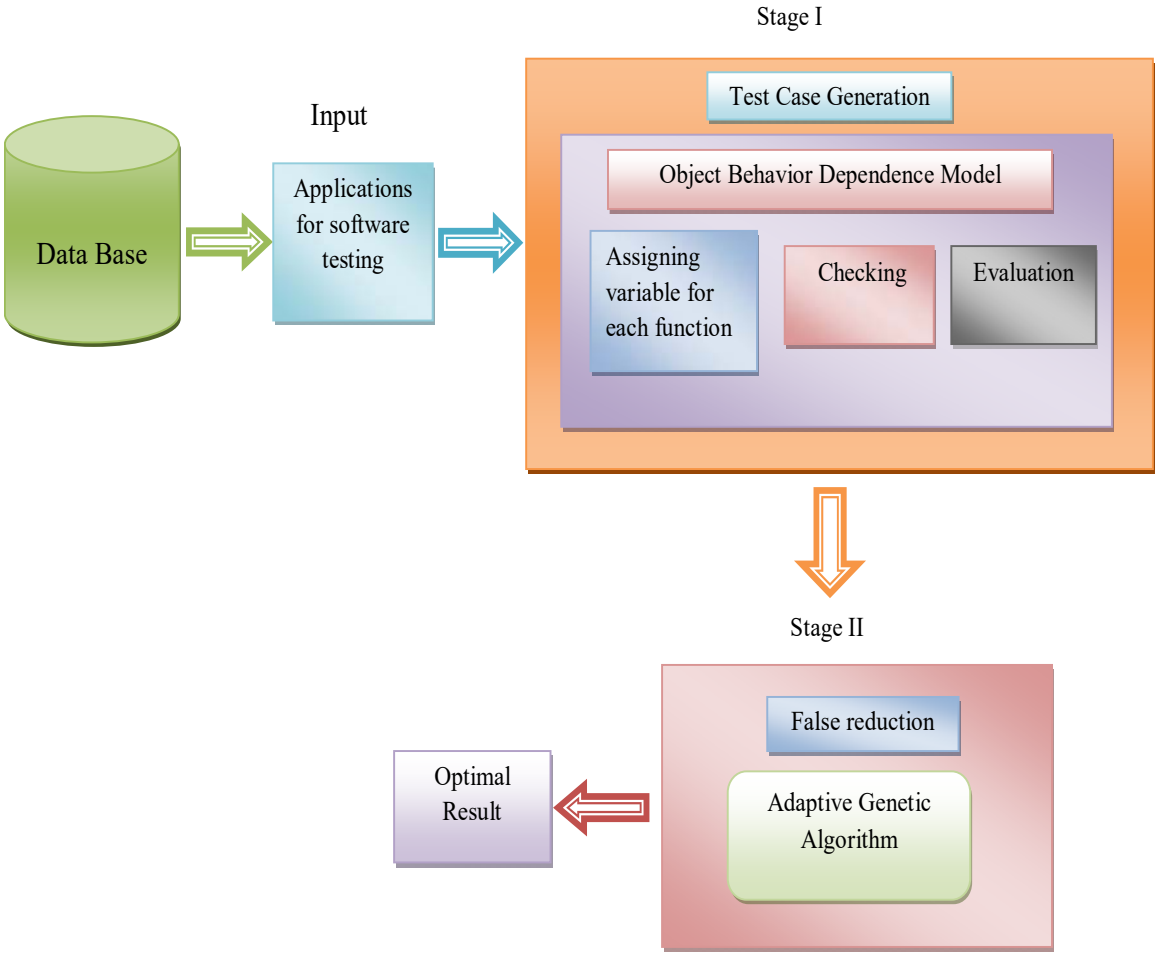

Fig.1 Block diagram of proposed method

Initially, the input application is selected from the database for software testing. After selecting the input application the test cases are generated with the help of Object Behavior Dependence Model (OBDM). Here the test cases are generated by means of feature value from the input application. After selecting the test cases the false reduction is done by adaptive genetic algorithm. In adaptive genetic algorithm the optimal inputs are generated which will reduce the illegal inputs and equivalent inputs. Based on the process the false reduction is done in the proposed method. The novel technique flows through the two phases shown below,

1. Test case generation
   - ❖ Object Behavior Dependence Model (OBDM)
2. False reduction
   - ❖ Genetic Algorithm
   - ❖ Adaptive Genetic Algorithm (AGA)
   - ❖ Particle Swarm Optimization

Stage 1:

4.1 Test Case Generation

The predictable procedure efficiently engenders the experiment conditions in keeping through the Object Behavior Dependence Model (OBDM). The function tackled to authentication, is considered as an input for the object activities reliance representation in the software testing. The detail explanation of the Object Behavior Dependence Model is illustrated in further section,

- ❖ **Object Behavior Dependence Model (OBDM):**

In the progression diagram, the group of nodes distinguishes the objects (O_b) and the group of edges exposes the function (F), where, $F \in S_f$ distinguish the synchronous task, which hold the six characteristic comprehensive beneath and include a direct reliance among the basis and end objects.

$F_{source} \in O_b$ – represents the source of the function

$F_{dest} \in O_b$ - relates to the destination of the function and where $F_{source} \neq F_{dest}$

F_{name} - characterized the name of the function

$F_{BW} \epsilon S_f$ - corresponds to the backward navigable function and where, $F_{BW} \neq F$ and it is indicated as "-".

F_{ER} - signifies the probabilistic execution rate of a function in a Sequence Diagram and where, $0 \leq F_{ER} \leq 1$ and the default value is one.

F_{EER} - corresponds to the expected execution rate of a function in a Sequence Diagram and where, $0 \leq F_{EER} \leq 1$ and the default value is one.

We consider the condition of a division control structure of a basis code, in which the implementation charge of a task is expected to be influenced. Presume a task is in an alteration mutual portion and only when the requirement in the portion is satisfied. If the task is executed surrounded by the equivalent specification portion, afterward the possibility of implementation charge of a task is 0.5. Otherwise, the defaulting value is one. The anticipated implementation charge of a task symbolizes a possibility of the implementation charge of a progression diagram. Consequently, it communicates to the possibility of the implementation time for the entire amount of task in an exact class to the implementation time for the entire amount of task in the whole input function. The utility in a progression diagram is carried out only when it is instigated. The defaulting value of F_{EER} is also one.

Here, each application has the number of function that is used for the generation of test case. Based on the function value, the suggested technique calculates the feature value. These value is represents the OBDM value. The exclusive OBDM procedure essentially endow its consideration on the task and reporting metrics of the function which have been useful for the experiment condition formation, which departs an extensive manner in fascinating the formation of the reproduction and inappropriate test conditions. The proposed method is mainly focus on the functions and coverage metrics of the application that we are used for the test case generation. In our implemented method the function name are represent as a variable. In Fig.2 demonstrates the general system of experiment condition formation is gracefully revealed and is efficiently endorsement by definite occurrence comprehensive

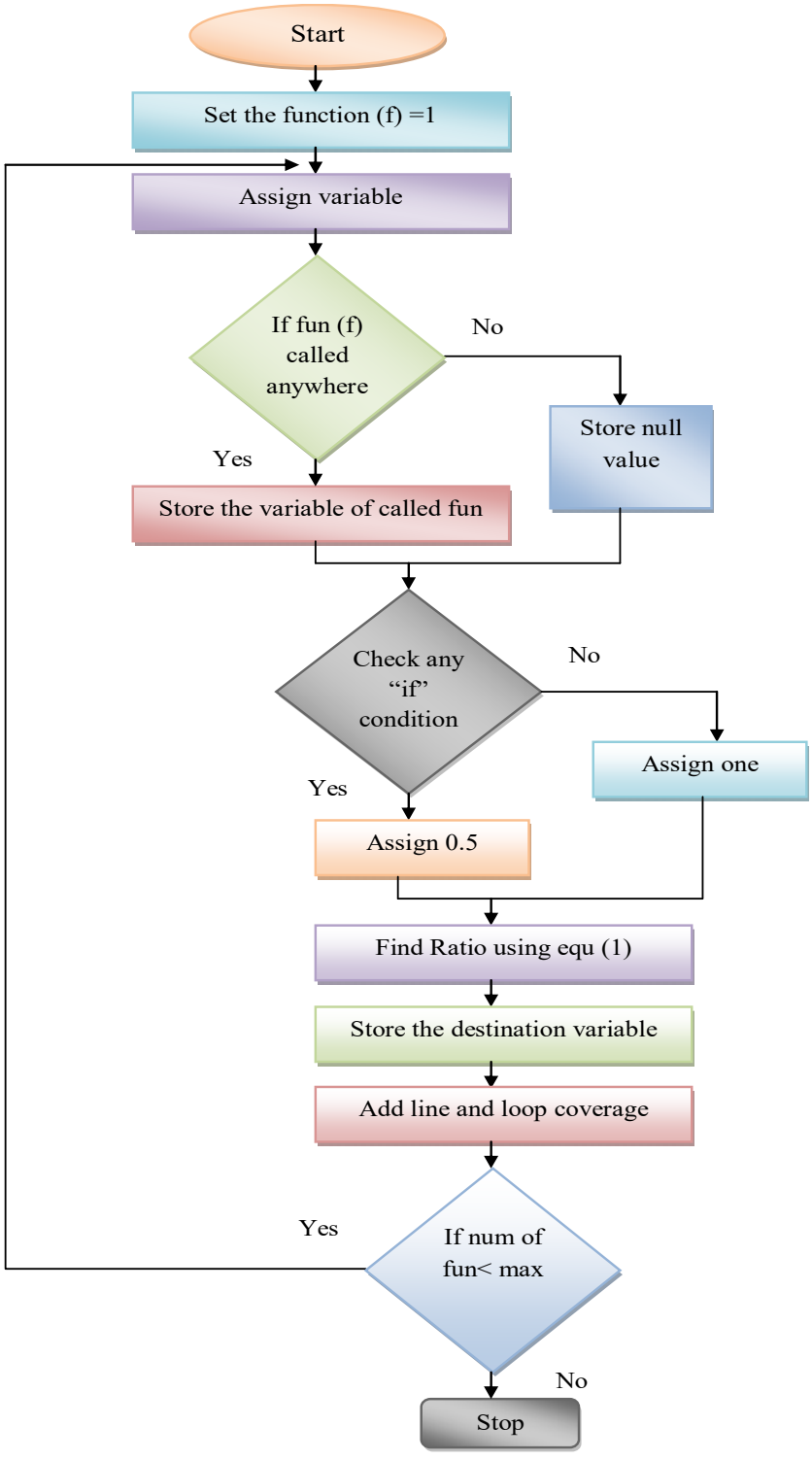

Fig.2 Flowchart for test case generation using OBDM

Figure 2 delightfully portrait the creation system of the experiment condition and each one task is symbolized as the basis task. Consequently, a basis task is representing a variable name and then each task is observed to determine whether it is previously fixed to definite supplementary mission. If it is formerly allocated, after that the connected variable name has to be distributed in the experiment condition, if not, it is dispensed as 'illogical'. In the occurrence, any task acquiring the "if" specification, a value of 0.5 is dispensed in the experiment condition; or else a value of one has to be distributed. Consequently, the particular task proportion is estimated trailed by the depiction of the target task. Currently, the entire experiment conditions are connected to the coverage matrix.

In this condition, a possibility value is dispensed and any task through the "if" specification is distributed a value of 0.5 values and a task among no such situation is selected a value of one.

The Ratio is assess for the individual task as per Equation 1 specified beneath and subsequently the target task is distinguished, trailed by the calculation of the whole experiment conditions to the coverage matrix. Ratio value is defined by the ratio of how much time one particular function call the other function to total function. It is specified in equation (1).

$$Ratio = \frac{how\ much\ time\ call\ the\ other\ function}{Total\ \ function} \tag{1}$$

The epoch-making technique makes use of the line coverage and loop coverage from the coverage metrics.

Line coverage:

Line coverage is also marked as statement, which embrace only the exact conditions. As well, it guesstimates the eminence of the code and makes sure the flow of a various trail in the code in query. It is assessed by the subsequent Equation 2.

$$line\ coverage = \frac{number\ of\ \ lines\ exercised}{total\ number\ of\ \ lines} \tag{2}$$

Loop coverage:

The loop coverage utensil is delegated by the mission of establishing and enlightening whether each one loop body is carry out either zero times, accurately on one occasion or numerous times. It also points out whether loop body is implemented accurately once or numerous times in the condition of 'do-while' loops. Supplementary, the while-loops and for-loops produce numerous presentations. The comparative data is not indicated by further coverage utensils. We consider the condition of one function for an occasion, which hold two programs through four tasks where the task names are symbolized as variables. The specific procedure demonstrated in Table.

Table I: Sample code

Class A	Class B
A1 **If** **{** **A2** **B1** **}** **A2** **{** **C1** **}**	**B1** **{** **B2** **C1** **}** **B2** **If** **{** **A1** **}**

Examples for line coverage and branch coverage

Sample code:

```
if(cond)
{
   line1();
   line2();
   line3();
   line4();
}
else
{
   line5();
}
```

If your test only exercises the *cond* being true and never runs the else branch you have; So, 4 out of 5 lines covered and 1 out of 2 branches covered. On the whole,

Line Coverage is the percent of lines executed by this test run and Branch Coverage is the percent of branches executed by this test run.

Examples for test case generation

Variable names for the tasks A1, A2, B1, B2 and C1 are delivered as F1, F2, F3, F4 and F5. . In our proposed method, the test case contains source function name, probability value, ratio value, destination function name, line coverage and loop coverage. Test case generation process with the corresponding example is given below,

Test case 1: [F1, -, 0.5, 2/5=0.4, F2] + line coverage + loop coverage

Test case 2: [F1, -, 0.5, 2/5=0.4, F1] + line coverage + loop coverage

Test case 3: [F2, -, 1, 1/5=0.2, F5] + line coverage + loop coverage

Here, F1 represents the initial function name, 0.5 indicates that if a test case contains any if condition or not. If contains means 0.5 will be assigned. If not means the value will be 1. Then the next value 2/5 indicates the ratio value of the test case mentioned in equation 1 and F2 represents the destination function name. Along with these values, line coverage and loop coverage values are used for test case generation.

Likewise, all the test cases are generated.

At this point the input task signifies the banking function. It includes roughly 47 classes and 108 tasks. The whole amount of experiment condition production in phase 1 is to the refrain of 661. In an indistinguishable way, the complete experiment conditions are engendered, which are consequently provided to the adaptive genetic algorithm. In observation of the reality that the experiment conditions are produced each time, it is to be expected enclose definite similarity on each instance. The original procedure efficiently utilizes the adaptive genetic method therefore as to reduce the occurrence of faults in the experiment conditions.

Stage 2:

4.2 False Reduction

The False reduction is fundamentally depicted as the restrain of additional features which are not fundamental for the principle of dispensation. In the highlight process, a spread of experiment condition is produced of which a few are not requiring for the dispensation mission. Supplementary, there is the probably of construction of indistinguishable experiment conditions at each and every time-frame. Consequently it is every part of the additional necessary to utilize the services of definite effectual system to pick out the equivalent experiment conditions. In the epoch-making procedure, the Adaptive genetic method is gracefully engaged for the principle of false reduction which is accomplished in keeping by the optimization. It is escort by the procedure of false reduction among the powerful aid of the Adaptive Genetic Algorithm (AGA). In the existing inquiry, the finest inputs are engendered as per the Adaptive Genetic Algorithm (AGA) which precisely releases its responsibility of lessening the illegitimate inputs and indistinguishable inputs. The roadmap for the inclusive procedure of the Adaptive Genetic Algorithm is excitingly graphed in the subsequent segment,

- **Adaptive Genetic Algorithm (AGA)**

Generally, the Genetic Algorithm symbolizes an inventive adaptive universal search method enchanting signal from the evolutionary data of the inheritance. In this process, the Iterations and the populace are symbolized as the production and the chromosomes correspondingly. In observation of the reality that the meeting charge of the conventional GA is lesser, the AGA is efficiently engaged for significantly rushing the meeting charge, through the powerful aid of the Cauchy alteration as the transformation machinist, which construct its impressive exterior as the supreme nominee in the genetic algorithm for calculating force to the GA mission and also to modify the GA recital.

AGA distinguishes a meta-heuristic procedure devoted to the reduction of the ordinary development system. It is consistently exploited to instigate the elucidation to a number of optimization and investigate problem, that methods activated by ordinary development, like the inheritance, adaptive mutation, selection, and crossover. Supplementary, the input of AGA composes the consequence of experiment condition invention.

Initial Phase

At the outset, the populations of the chromosomes x_i , ($i = 1,2,...N$) are created arbitrarily. N represents the dimension of the population. The chromosome (x_i) encompasses the test cases produced in an arbitrary manner. Here the chromosome x_i is the set of test cases.

Fitness Evaluation

The fitness value of each one limitation is assessed as demonstrated in the subsequent Equation 3 and the chromosome containing the utmost fitness value is pick out as the finest chromosome. We have already calculated some factors for each test case such as OBDM value, fault proneness ratio, line coverage, loop coverage etc. These metrics are used to estimate fitness which is given below.

$$fitness_i = \sum_{i=1}^{N} x^i{}_{if\ value} + x^i{}_{ratiovalue} + x^i{}_{linecoverage} + x^i{}_{loopcoverage} \quad (3)$$

Here our problem is maximization of fitness to reduce the interactive faults. The OBDM value will presents the object behavioral dependency value.

Selection of Chromosomes

Single or several parent chromosomes are pick out in keeping by the '$N/2$' finest chromosomes possessing the maximum fitness value and the innovative explanation is engendered.

Crossover

The solitary peak crossover is implemented at the crossover charge of (C_r) and consequently ('$N/2$',) issue are accomplished. In the condition of each and every crossover task, (NC_r) genes are exchanged among the comparative parents.

Mutation

Mutation and Crossover are the two important genetic operation which will help for solution convergence. Here the adaptive behavior of GA will be presented in terms of Adaptive mutation.

Here we are using Cauchy's mutation for adaptiveness in GA. The entities are concerned probabilistically to convey a radical modify in themselves. Whereas utilizing the alteration machinist, there is a possibility that definite original qualities emerge on description of the alteration in the chromosome. The Cauchy transformation is successfully exploited to alter the entities as per Equation 4 exposed. The alteration is executed in keeping by the fixed transforming possibility. Where the Cauchy alteration is achieved, illogical variable 'x' symbolize a Cauchy allocation. The Cauchy allocation task is distinct as per the subsequent Equation 4,

$$F(x) = \frac{1}{2} + \frac{1}{\pi}\arctan(x) \quad (4)$$

Here x represents the test cases.

Updation of solution

When the transformation task is ended, innovative chromosome is engendered and subsequently the current chromosome is surrogated by innovative chromosome. This system is identified as the modernizing of explanation. If the fitness value of innovative chromosome surpasses that of the existing chromosome, the innovative chromosome is elected as the finest chromosome.

Termination Criteria

The procedure is carried on till it satisfies the termination criteria.

The flowchart for the novel technique is elegantly exhibited in Figure 3.

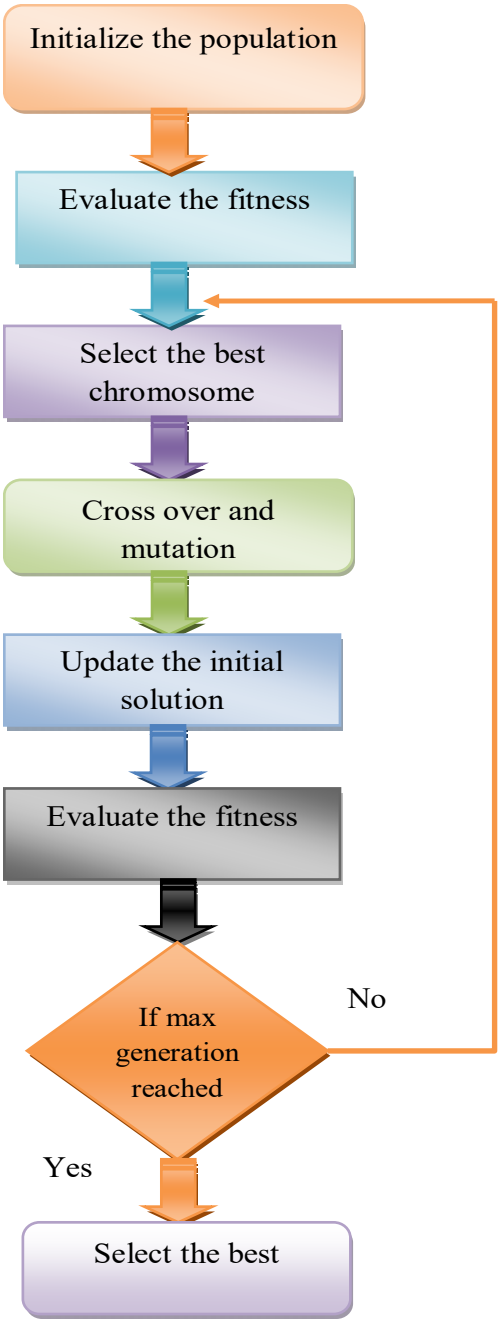

Fig.3 Flowchart for the proposed Adaptive genetic algorithm

The most favorable product is contrasted by the complete task subsequent to accomplishing the conclusion from the adaptive genetic algorithm. Some task which does not give up the finest consequence has to be instantaneously detached from the function. Hence the adaptive genetic algorithm is capable to accomplish substantial lessening in the mistake charge rely on the best experiment condition.

In the inventive procedure, the task of the AGA is the selection of the finest ideal experiment conditions which are proper for the input function. Consequently, the counterfeit lessening is accomplished by means of the optimization algorithm and the optimized conclusion is attained rely on the fitness value selected the AGA process. The experiment conditions provided to the optimization algorithm are practiced in keeping by means of the fitness values. The experiment conditions encompassing better fitness values signify the bug free function. Therefore, through the aid of the AGA procedure, the essential experiment conditions are attained for the inventive procedure.

5. Result and Discussion

In our experiment, we have utilized the adaptive genetic algorithm is used for Reducing. Interactive Faults Based on Optimal Test Cases in Directed Random Testing**.** The implementation is done in the JAVA platform and the results are given as follows.

Based on the fitness value of the parameter chosen, the whole process labors in Genetic algorithm. For the additional processing the parameter with high fitness value is chosen. For the Adaptive Genetic Algorithm the fitness values of the chromosomes are computed for dissimilar iteration and the results are charted. By seeing the Table 2 the fitness value for the suggested technique verified to be superior to the technique where GA is applied

Table 2: Fitness comparison for different iterations using our proposed AGA and GA.

Iterations	Fitness values		
	Adaptive Genetic Algorithm	Genetic Algorithm	PSO
25	650	547	621
50	630	520	610
75	573	495	558
100	567	475	496

The fig 4 given below shows the graphical representation of fitness value using the Adaptive and normal GA. In this, the fitness value that are obtained from adaptive genetic algorithm as well as that we obtain from Genetic algorithm are being plotted. From the graph it is evident that our proposed method using adaptive GA delivers better fitness value when compared to that of the GA.

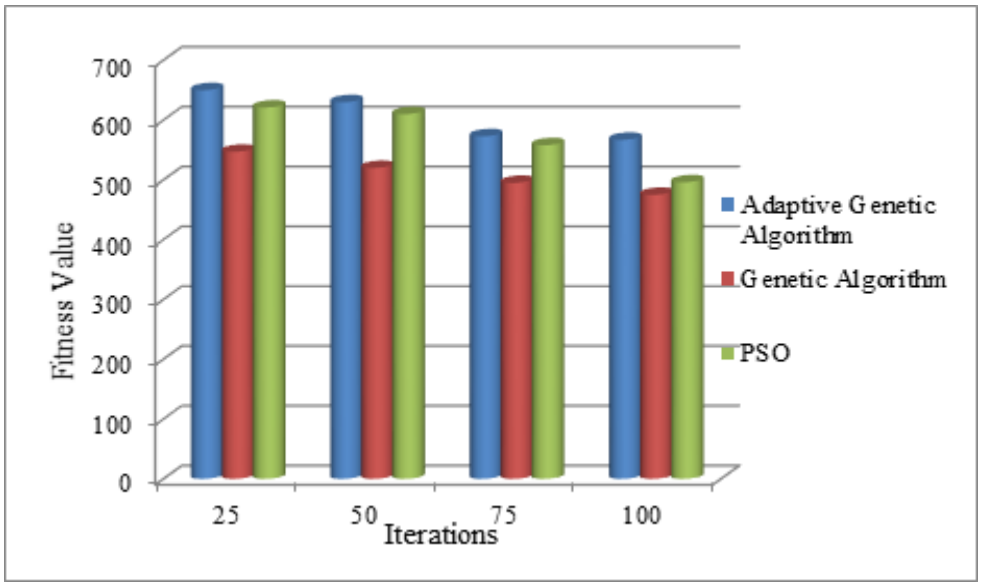

Fig 4: Graphical representation of Fitness value for different iterations using Adaptive GA and GA

The table 3 given below shows the test count value obtained before and after optimization. The test case counts are to be reduced in order to select optimal test cases. The optimized result shows that test cases that are not relevant to the required process are being ignored which is evident from the Test case count obtained after optimization

Table 3: Test count value obtained before and after optimization.

Iterations	Test case count	
	Without AGA	With AGA
25	661	434
50	661	463
75	661	437
100	661	414

The fig 5 shows the graphical representation of Test count value which is obtained before and after optimization. Here the test cases that are generated before and after optimization are plotted. From the graph it is evident that the Test case count has been reduced to a greater extend when compared to those obtained before optimization.

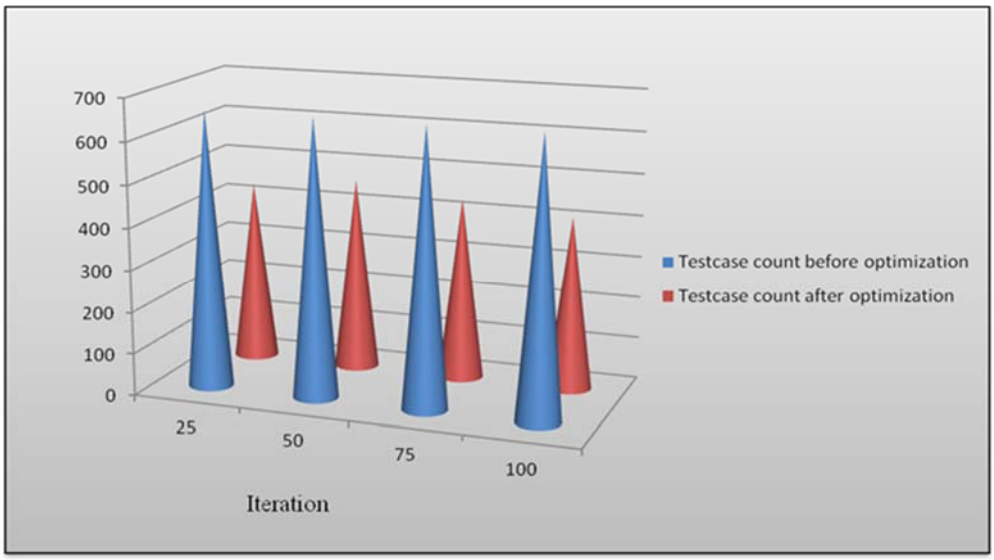

Fig 5: Graphical representation for Test count value obtained before and after optimization.

The table 4 given below shows the test case count obtained from the different existing techniques and our proposed AGA method. We have obtained better test case count when compared with other methods.

Table 4: Test case count for proposed and existing methods

Iterations	Test case count		
	AGA	GA	PSO
25	434	475	497
50	463	470	491
75	437	461	462
100	414	452	449

The fig 6 given below shows the graphical representation of Test case count for proposed and existing methods.

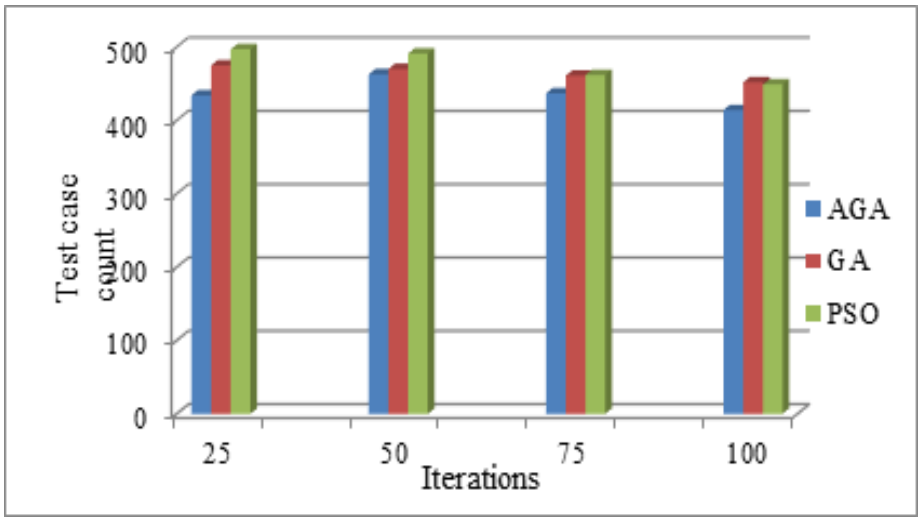

Fig 6: Graphical representation for Test case count for proposed and existing methods

The table 5 given below shows the time and memory usage of our proposed methodology. For each iteration, the corresponding time and memory usage are calculated and the results are tabulated. By reducing the interactive faults here, we reduce the execution time and memory usage. When the iteration increases, the time usage and memory usage are reducing automatically.

Table 5: Time and Memory usage for different iterations using proposed AGA.

Iteration	Time usage (sec)	Memory usage (Bytes)
25	4256	4586136
50	4399	3001288
75	4525	2973680
100	4502	2188312

The table 6 given below shows the time usage obtained from the different existing techniques and our proposed AGA method. We have obtained better time usage when compared with other methods. The time usage is calculated the difference between the process start system time and the process end system time. The table 6 is tabulated in beneath,

Table 6: Comparison of Time usage in proposed and existing method.

Iteration	Time usage (sec)		
	AGA	GA	PSO
25	4256	4521	5214
50	4399	4698	5365
75	4525	4724	5368
100	4502	4792	5741

The fig 7 given below shows the graphical representation of time usage for proposed and existing methods.

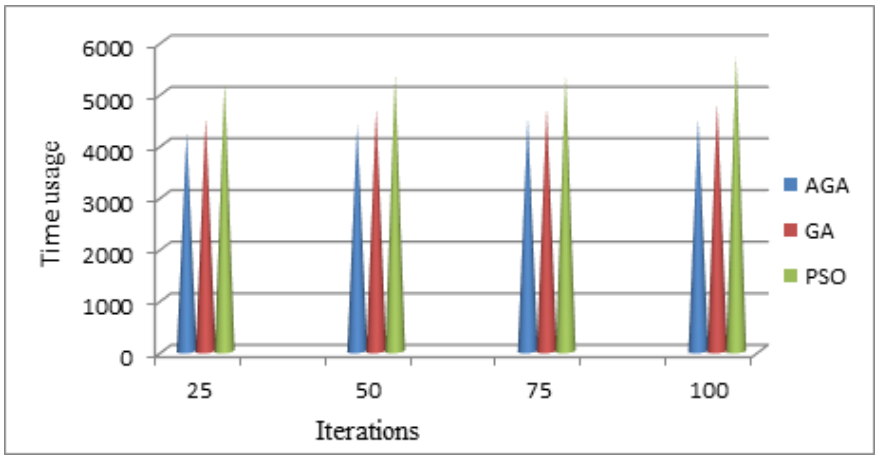

Fig 7: Graphical representation of Time usage for different iterations

The faults detected from the proposed method of random testing can be compared with other existing testing techniques which is presented in Table 7

Table 7: Fault detected from various testing methods

Testing Methods	Faults Detected (in %)
Random Testing	95%
Regression Testing	92%

From the table 7, we observe that the interactive faults can be highly detected from the random testing than other testing like regression testing. If the test suite is large, then RT will give better results. The real time application of the proposed technique is used to analyze in a new light the applicability of Combinatorial Interaction Testing (CIT). In real-world industrial systems with thousands or even hundreds of features have been analyzed.

6. Conclusion

In the article, an inventive procedure for reduce the interactive mistake is instigated which reliant on finest experiment conditions in the intended for unsystematic is experiment. The original method is engaged to reduce the interactive mistakes as per the adaptive genetic algorithm. The adaptive genetic algorithm constructs the finest product which significantly diminishes the prohibited inputs. By an eye on decrease the flaws; the original method efficiently utilizes the coverage metrics. The encouraging product reveal the reality that the innovative procedure emerges by flying colors in receiving liberate of the indistinctness of subjectively formed experiment conditions and producing the finest product. In the days to arrive, it is probable to recognize the experiment condition allocation metrics as experiment condition choice principle for accomplishing the greater coverage experiment conditions quickly. Supplementary, definite further inventive meta-heuristic investigate utensils will be exploited to engender the flaw-free experiment conditions.

References

[1] Zhou, Z. Q., Sinaga, A., & Susilo, W. (2012, January). On the fault-detection capabilities of adaptive random test case prioritization: Case studies with large test suites. In System Science (HICSS), 2012 45th Hawaii International Conference on (pp. 5584-5593). IEEE.
[2] Niu, X., Wang, Y., & Wu, D. (2014, August). A Method to Generate Random Number for Cryptographic Application. In Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP), 2014 Tenth International Conference on (pp. 235-238). IEEE.
[3] Dionísio, J., Mota, T., Pinto, I., & Niehus, M. (2014). Real Time Random Number GeneratorTesting. Procedia Technology, 17, 534-541.
[4] Malpani, P., & Bassi, P. (2014, September). Analytical & empirical analysis of external sorting algorithms. In Data Mining and Intelligent Computing (ICDMIC), 2014 International Conference on (pp. 1-6). IEEE.
[5] Tahbildar, H., & Kalita, B. (2011). Automated software test data generation: direction of research. International Journal of Computer Science and Engineering Survey, 2(1), 99-120.
[6] Mcminn, P. (2013). An identification of program factors that impact crossover performance in evolutionary test input generation for the branch coverage of C programs. Information and Software Technology, 55(1), 153-172.
[7] Fraser, G., Arcuri, A., & McMinn, P. (2015). A memetic algorithm for whole test suite generation. Journal of Systems and Software, 103, 311-327.
[8] Galeotti, J. P., Fraser, G., & Arcuri, A. (2013, November). Improving search-based test suite generation with dynamic symbolic execution. In Software Reliability Engineering (ISSRE), 2013 IEEE 24th International Symposium on (pp. 360-369). IEEE.
[9] Darab, M. A. D., & Chang, C. K. (2014, October). Black-box test data generation for gui testing. In Quality Software (QSIC), 2014 14th International Conference on (pp. 133-138). IEEE.
[10] Putra, I. P. E. S., & Mursanto, P. (2013, September). Centroid Based Adaptive Random Testing for object oriented program. In Advanced Computer Science and Information Systems (ICACSIS), 2013 International Conference on (pp. 39-45). IEEE.
[11] Chow, C., Chen, T. Y., & Tse, T. H. (2013, July). The art of divide and conquer: An innovative approach to improving the efficiency of adaptive random testing. In Quality Software (QSIC), 2013 13th International Conference on (pp. 268-275). IEEE.
[12] Khan, S. A., & Nadeem, A. (2013, April). Automated test data generation for coupling based integration testing of object oriented programs using evolutionary approaches. In Information Technology: New Generations (ITNG), 2013 Tenth International Conference on (pp. 369-374). IEEE.
[13] Campos, J., Abreu, R., Fraser, G., & d'Amorim, M. (2013, November). Entropy-based test generation for improved fault localization. In Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering (pp. 257-267). IEEE Press.
[14] Yu, B., & Pang, Z. (2012). Generating Test Data Based on Improved Uniform Design Strategy. Physics Procedia, 25, 1245-1252.
[15] Padgham, L., Zhang, Z., Thangarajah, J., & Miller, T. (2013). Model-based test oracle generation for automated unit testing of agent systems. IEEE Transactions on Software Engineering, 39(9), 1230-1244.
[16] Ahmed, B. S., Sahib, M. A., & Potrus, M. Y. (2014). Generating combinatorial test cases using Simplified Swarm Optimization (SSO) algorithm for automated GUI functional testing. Engineering Science and Technology, an International Journal, 17(4), 218-226.
[17] Arcuri, A., & Briand, L. (2012). Formal analysis of the probability of interaction fault detection using random testing. IEEE Transactions on Software Engineering, 38(5), 1088-1099.
[18] Minku, L. L., Sudholt, D., & Yao, X. (2014). Improved evolutionary algorithm design for the project scheduling problem based on runtime analysis. IEEE Transactions on Software Engineering, 40(1), 83-102.
[19] Lv, J., Hu, H., Cai, K. Y., & Chen, T. Y. (2014). Adaptive and random partition software testing. IEEE Transactions on Systems, Man, and Cybernetics: Systems, 44(12), 1649-1664.
[20] McMinn, P., Harman, M., Lakhotia, K., Hassoun, Y., & Wegener, J. (2012). Input domain reduction through irrelevant variable removal and its effect on local, global, and hybrid search-based structural test data generation. IEEE Transactions on Software Engineering, 38(2), 453-477.
[21] Arcuri, A. (2012). A theoretical and empirical analysis of the role of test sequence length in software testing for structural coverage. IEEE Transactions on Software Engineering, 38(3), 497-519.
[22] Kempka, J., McMinn, P., & Sudholt, D. (2013, July). A theoretical runtime and empirical analysis of different alternating variable searches for search-based testing. In Proceedings of the 15th annual conference on Genetic and evolutionary computation (pp. 1445-1452). ACM.
[23] Arts, T., Gerdes, A., & Kronqvist, M. (2013, September). Requirements on automatically generated random test cases. In Computer Science and Information Systems (FedCSIS), 2013 Federated Conference on (pp. 1347-1354). IEEE.
[24] Barus, A. C., Chen, T. Y., Kuo, F. C., Liu, H., Merkel, R., & Rothermel, G. (2016). A cost-effective random testing method for programs with non-numeric inputs. IEEE Transactions on Computers, 65(12), 3509-3523.
[25] Syed Abdul Moeed and Niranjan Polala,(2017) "Interactive Faults Detection (Ifd) In Acceptance Testing Based On Optimal Test Cases Using Gravitational Search Algorithm (Gsa),"International Journal of Pure and Applied Mathematics Volume 115(8), 559-564.
[26] Chen, J., Kuo, F. C., Chen, T. Y., Towey, D., Su, C., & Huang, R. (2017). A Similarity Metric for the Inputs of OO Programs and Its Application in Adaptive Random Testing. IEEE Transactions on Reliability, 66(2), 373-402.