Journal of Theoretical and Applied Information Technology
31st December 2023. Vol.101. No 24
© 2023 Little Lion Scientific

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

ENGENDERING FINEST TEST CASES BY USING G-GENETIC ALGORITHM VARIANT SSGA

T J PRASANNA KUMAR[1], JASTI SURENDRA[2], PONNURU ANUSHA[3], DR M. BABU PRASAD[4] MANASA BANDLAMUDI[5] K.PRAVEEN KUMAR[6] ANIL KUMAR PALLIKONDA[7]

[12]Asst Professor, Department of Mechanical Engineering, PVP Siddhartha Institute of Technology, Vijayawada, [3]Asst Professor Dept of CSE, Koneru Lakshmaiah Education Foundation, Vaddeswaram, 522502, [4] Professor, Freshmen Engineering Department, NRI Institute of Technology, Agiripalli
[5]Asst Professor, Department of Information Technology, R.V.R.&J.C. College of Engineering, Chowdavaram, Guntur
[6] Associate Professor, Department of Mechanical Engineering, R.V.R.&J.C. College of Engineering, Chowdavaram, Guntur
[7]Dept. of CSE, PVP Siddhartha Institute of Institute of Technology, Vijayawada, India,

E-mail: tjpk@pvpsiddhartha.ac.in, anusha.ponnuru588@kluniversity.in, bpsirvijayawada@gmail.com, anilkumar.pallikonda@gmail.com

ABSTRACT

Programming Testing is the most difficult occupation among every one of the companions of the business. Thorough programming Testing is never conceivable just streamlined programming testing is conceivable. Thus Programming Testing can be seen as streamlining issue as it fall under NP Hard. Because of the enormous number of experiments that are expected to perform adequate testing of the ideal programming application; the assorted strategies to decrease the test suite is required. One of the normal concentrated on techniques is eliminating the repetitive experiments; the explanation is negligible number of experiments and most extreme number of mistakes disengagement or uncovering. In this examination work review is led to address the use and viability of Consistent satisfy hereditary calculation to reduce the quantity of experiments that don't added unmistakable worth in that frame of mind of test inclusion or where the experiments can't disconnect blunders. Hereditary calculation is used in this work to help in limiting the experiments or upgrading the experiments , where the hereditary calculation produces the fundamental populace arbitrarily, ascertains the wellness esteem utilizing inclusion measurements, and afterward particular the posterity in sequential ages utilizing hereditary tasks determination, get over and change. The hereditary displaying activities are explicit and in light of the activity might shift to typical hereditary calculation. This course of age is rehashed until there is no adjustment of the wellness values for two successive ages, when there is no adjustment of the information age for two emphases so intermingling achieved or a limited experiment is accomplished. The aftereffects of review show the way that, hereditary calculations can essentially lessen the size of the experiments

Keywords: *SSGA, NP Hard, Test Case, Error, Testing.*

1. INTRODUCTION

Irregular testing is a black-box programming testing procedure where projects are tried by producing arbitrary, free data sources. Among the different programming testing procedures, Irregular Testing (RT) is the most crucial methodology. It is basic in idea, frequently simple to apply, can practice the product under test unexpectedly, and has shown adequacy in recognizing disappointments [1]. Arbitrary number is generally utilized in cryptographic applications, which is mostly utilized as key. Since the security of key absolutely relies upon the sum and haphazardness of itself, and vital to create arbitrary numbers for cryptographic applications [2]. Irregular number generators generally utilized in business applications don't rigorously ensure these prerequisites [3]. Notwithstanding, a significant test for these methodologies is the dubious, two-layered, and combinatorial immense info space that they need to investigate and practice consequently [4].

The time test information age utilizing half and half strategy that takes the benefits of both

Journal of Theoretical and Applied Information Technology
31st December 2023. Vol.101. No 24
© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

static and dynamic technique was done [5]. Programming testing stays a very expensive movement in the computer programming lifecycle, and thusly, its mechanization keeps on being of high concern [6]. To create tests that cover every one of the branches in a class, the class should be started up, and a technique call succession might should be produced to place the item into a specific state [7]. Creating unit test suites naturally is a significant commitment towards further developing programming quality, and procedures like pursuit based programming testing dynamic emblematic execution can productively deliver test suites accomplishing high code inclusion [8].Model-based framework testing of utilizations with a GUI front-finish to be more financially savvy and effective contrasted with their customary record-then-replay partners [9]. A change of RT exists to work on its productivity, like Versatile Irregular Testing (Craftsmanship) Craftsmanship used to test mathematical projects, in light of disappointment designs which comprises of three classifications: block design, strip example, and point design [10]. Nonetheless, Craftsmanship is less effective than irregular testing in view of the additional errand of guaranteeing in any event, spreading of experiments, where the proficiency is estimated as far as an opportunity to create an experiment [11].Random age of test information depends on specific contributions of some dissemination. Way situated and underlying methodologies utilize the program's control stream chart for test information age; they select a way, and utilize a procedure like emblematic execution for age of test information. Objective situated test-information age approaches select contributions to execute the chose objective, like proclamation, condition inclusion, choice inclusion, independent of the way taken [12]. The reason for a conclusion is a test suite, and frequently the current test suite isn't streamlined for delivering great symptomatic reports. Thus it is essential to produce tests to further develop finding. There are numerous accessible test age methods Search-Based Programming Testing (SBST). Worldwide inquiry calculations are Hereditary Calculations [13]

Present day frameworks are exceptionally configurable to fulfill different necessities of clients For instance, programming applications running on cell phones can designed with numerous features{type of telephone, working framework and introduced application[14]. Every setup address an alternate item and it might display various disappointments. In modern frameworks, there are ordinarily a huge number of potential designs where perhaps a little sub set of designs can set off disappointments, the inquiry is the way to expand disappointment discovery when testing all configurations is unimaginable[15][16]

Troubleshooting is a tedious undertaking in programming improvement. Albeit different computerized approaches have been proposed, they are not sufficiently viable. Then again, in manual troubleshooting, engineers experience issues in picking breakpoints. To resolve these issues and assist engineers with finding flaws really, intelligent shortcoming confinement structure is utilized which joins the advantages of computerized approaches and manual troubleshooting. Before the shortcoming is found, the structure persistently suggests checking focuses in light of proclamations' doubts, which are determined by the execution data of experiments and the criticism data from the designer at prior really looking at focuses. This course of intuitive issue location makes characteristics of programming to improve to a more prominent broadens[17. The intelligent shortcoming examination makes the cycle pretty much successful in programming testing situation[18]. The significant goal of our proposed strategy is to decrease the intelligent flaws that exist while planning programming. The intuitive shortcoming essentially diminishes the nature of specific programming. Subsequently we have planned a proficient strategy where delicate figuring procedure like versatile hereditary calculation for advancing the experiments that are being produced[19][20]

2. PROBLEM STATEMENT: Versatility and viability is a significant issue that should be considered while testing and it is a basic issue in the product business. Many examinations on certifiable programming are not so normal and this is to a limited extent because of the immense computational time that is expected to complete them. The broadly useful of irregular testing is to produce whatever number experiments as would be prudent so that they help reveal whatever number flaws as numerous inclusion focuses as could be expected under the circumstances. Experiments trigger disappointments and don't straightforwardly reveal issues; from a numerical point of view we can't think about shortcomings as targets. Experiments are picked with the imperative that no less than one experiment is browsed each sub-space. For instance, every usefulness of the product can be considered as an alternate sub-space to test. During the age of experiments, contingent upon the points of interest of the segment procedure, needed to create and run a few experiments to confirm

Journal of Theoretical and Applied Information Technology
31st December 2023. Vol.101. No 24
© 2023 Little Lion Scientific
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

regardless of whether they have a place with parcel. A perception is showing that many program deficiencies bring about disappointments in bordering region of the information area. Craftsmanship deliberately guides, or channels, haphazardly produced applicants, to exploit the possible presence of such sources of info, which endeavor to further develop the disappointment recognition adequacy of arbitrary testing. Locales of the information area where the product produces yields as per determination will likewise be touching. Subsequently, given a bunch of recently executed experiments that poor person uncovered any disappointments, new experiments found away from these old ones are bound to uncover disappointments.

3. PROPOSED METHOD OF IMPLEMETATION AND SOLUTION:

Programming testing is immense and different field, recreating the various prerequisites that product curios should fulfill the various exercises associated with testing and the various levels at which programming can be checked. Inconsistent testing creates test inputs with no obvious end goal in mind from the information space of the product under test. The fundamental element of an irregular test age procedure is that it produces test inputs at erratic from a language structure or some other conventional curio making sense of the info space. The main reason for the proposed strategy is to expand a viable procedure for diminishing intuitive shortcomings in view of ideal experiment in direct arbitrary testing. For the age of experiments the proposed strategy is utilizing Article Conduct Reliance Model (OBDM). The subsequent data sources are taken care of to the GA, Versatile Hereditary Calculation (AGA), and SSGA after the experiment age. In SGA the ideal data sources are created which decline the unlawful information sources and comparable sources of info, and there by it diminishes the shortcoming inclination. The predetermined course of executed strategy is shown in Figure 3.1. The block chart of the proposed technique is shown in beneath,

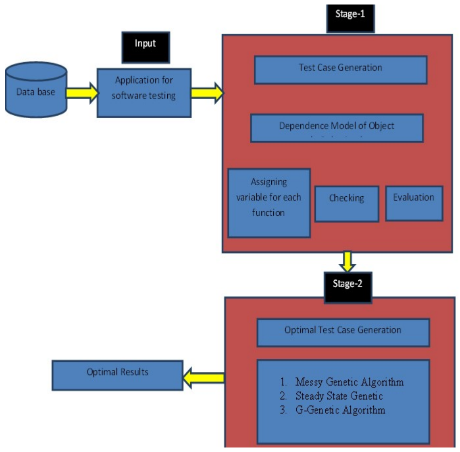

Stage-1
Input
Data base
Application for software testing
Test Case Generation
Dependence Model of Object
Assigning variable for each function
Checking
Evaluation
Stage-2
Optimal Test Case Generation
Optimal Results
1. Messy Genetic Algorithm
2. Steady State Genetic
3. G-Genetic Algorithm

Figure 3.1 Block Diagram Of Proposed Method

Dependency Model: In sequence diagram set of nodes representing objects (O_b) and set of edges that indicate the function (F) where, $F \in S_f$ represents the synchronous function. Function has the following six attributes and has a direct dependency between the source and destination objects.

$F_{source} \in O_b$ - Source of the function

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

F_{name} - Name of the function

$F_{BW} \in S_f$- Backward navigable function and where, $F_{BW} \neq F$ it is denoted as "-".

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

FEER-Expected execution pace of a capability in a Grouping Outline and where, $0 \leq FEER \leq 1$ and the default esteem is 1.We consider a branch control design of a source code, in which the execution pace of a capability might be impacted. Consider a capability is in an alt consolidated piece and just when the condition in the part is fulfilled. In the event that the capability is executed inside this condition part, the likelihood of execution pace of a capability is 0.5. In any case, the default esteem is 1.

The normal execution pace of a capability is a likelihood of the execution pace of a grouping outline. At the end of the day, it is the likelihood of the execution time for the complete number of capabilities in a specific class to the execution time for the all-out number of capabilities in the entire information application. The capability in a grouping graph is executed just when it is enacted. The default worth of FEER is additionally 1

Our proposed technique has two phases to be specific, 1.Test case generation

- ❖ Dependence Model of Object Behavior

2.Optimal test Case Engendering

- ❖ Genetic Algorithm (GA)
- ❖ Steady State Genetic Algorithm (SSGA)

Stage 1:

3.1 Test Case Generation

The proposed strategy creates the experiments in light of the Reliance Model of Item Conduct. The application which we are checking, takes as a contribution for object conduct reliance model in programming testing. Every application has the quantity of capability that is utilized for the age of experiment. The recommended DM technique chiefly centers around the capabilities and inclusion measurements of the application that we are applied for the experiment age. This will try not to create copy and immaterial experiments. The capability name is represented as a variable in our executed strategy. In flowchart (Figure 3.2) the general course of experiment age is outlined and explicitly made cleared underneath for certain models,

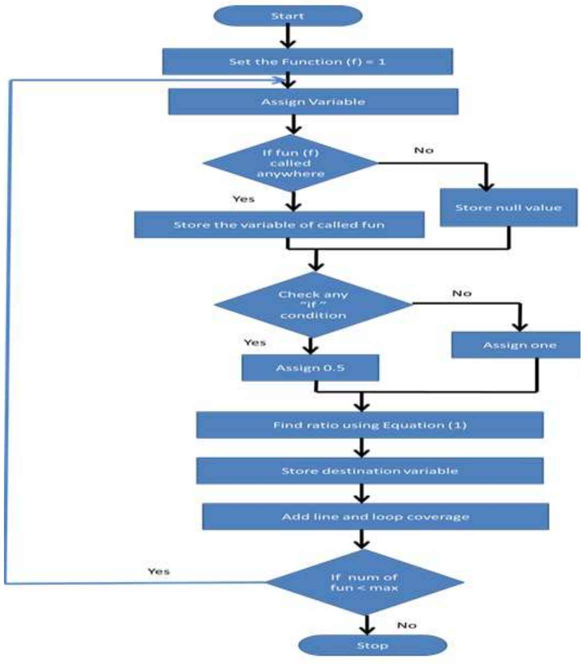


Figure 3.2: Flowchart For Test Case Generation Using Dependency Model

Figure.3.2 delineates the age cycle of experiment. Think about each capability as the source capability. For each source capability, a variable name is indicated, and afterward each capability is confirmed to find whether it is recently dispensed for another undertaking. On the off chance that it has recently distributed, next related variable name ought to be appointed in the experiment, or disaster will be imminent, it is allot as nonsensical. In the event that any capability contains "if" condition it designates 0.5 qualities in the experiment or, more than likely it allocates one. For the given capability, the proportion will be worked out and finally it represents the objective capability. All the experiments are joined to the inclusion measurements. Here we accept the likelihood esteem, any capability comprise "if" condition we appoint 0.5 worth in different cases we allocate one. Proportion worth will be determined for the given capability lastly address the objective capability. Then add all the experiment to the inclusion measurements.

$RV = A/B$ (1)

A= total No Of times called by Other Function

B= Total Functions

The suggested method employs only the line coverage and loop coverage from the coverage metrics.

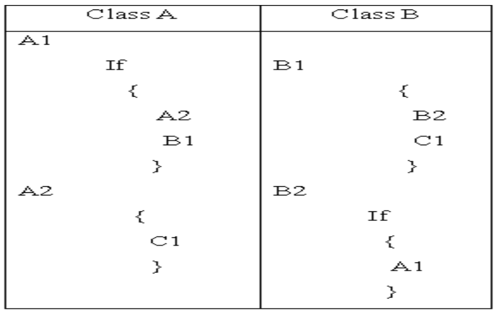
Line coverage:

Line coverage is as well identified as the statement coverage or segment coverage. Only correct conditions are wrapped by line coverage. It as well measures the quality of the code and makes sure the flow of different path in that code.

$$line\ \cos\ erage = \frac{number\ \ of\ \ lines\ \ exercised}{total\ \ number\ \ of\ \ lines}$$

Loop coverage:

This inclusion measurements reports whether each circle body is carried out multiple times, unequivocally once and at least a few times. This measurements reports whether circle body is carried out definitively once and at least a couple of times for do-while circles. Furthermore, too, while-circles and for-circles perform at least a few times. This information isn't accounted by other inclusion measurements. For instance, Ponder one application; it has two classes with four capabilities here the capability names are represented as a variable one. The predetermined cycle showed in table.



Class A	Class B

```
A1
    If
      {
        A2
        B1
      }
A2
      {
        C1
      }
```
	```
	B1
	      {
	        B2
	        C1
	      }
	B2
	    If
	      {
	        A1
	      }
	```

Variable name for the capabilities A1, A2, B1, B2 and C1 are F1, F2, F3, F4 and F5. In the proposed technique, the experiment contains source capability name, likelihood esteem, proportion esteem, objective capability name, line inclusion and circle inclusion. Experiment age process with the comparing model is given underneath,

TC1: [F1, - , 0.5, 2/5=0.4, F2] + line inclusion + circle inclusion

TC2 : [F1, - , 0.5, 2/5=0.4, F1] + line inclusion + circle inclusion

TC3: [F2, - , 1, 1/5=0.2, F5] + line inclusion + circle inclusion

We take the info capability is banking application. It contains almost 47 classes and 108 capabilities. All out number of experiment age in stage 1 is around 661. In next stage coming about experiments are taken care of to the GA, versatile hereditary calculation, SSGA. Since we will deliver experiments in each time, it encases some likeness on each time. The recommended technique utilizes the versatile hereditary calculation to diminish the shortcoming event of the experiments.

Stage 2:

3.2 Optimal Test Case Generation: False reduction is basically defined as the reduction of added parameters which are not required for processing. In the proposed method, we generate enormous test cases in which some of those test cases are not required for processing. Also there may be similar test cases that are being generated at each time interval. In order to select the concerned test cases, we require some efficient techniques. The next stage of the suggested method is false reduction by means of the GA, Messy Genetic Algorithm (MGA) and SGA. Here in the proposed method Steady State Genetic Algorithm genetic algorithm for obtain optimized results. In this research, the optimal inputs will be produced based on Steady State Genetic Algorithm (SGA) which will decrease the illegal inputs and equivalent inputs. The overall procedure of GA, Adaptive Genetic Algorithm and SGA is described beneath,

3.2.1Genetic Algorithm: It is a stochastic calculation: Haphazardness must be fundamental job in GAs. Both determination and propagation need irregular strategies GA generally considers a populace of arrangements keeping in memory in excess of a solitary arrangement at every emphasis offers a ton of benefits

The simple form of GA

1. Start with randomly generated population
2. Calculate the fitness of each chromosome in the population
3. Repeat the following steps until n off springs have been created
    - Select a pair of parent chromosomes from current population
    - With probability P_c crossover the pair at randomly chosen point to form two offspring's
    - Mutate the two offspring's at each locus with probability P_m
4. Replace the current population with the new population
5. Go to step 2

For the most part the underlying populace is created haphazardly then the age is circles over a cycle interaction to cause the populace to develop every emphasis comprises of the accompanying

Selection: The first step consists in selecting individuals for reproduction. This selection is done randomly with probability depending on the relative fitness of individuals

1. Reproduction: This step consists of both recombination and mutation
2. Evaluation: Then the fitness of new chromosome is evaluated
3. Replacement: Individuals from the old population are killed and replaced by new ones

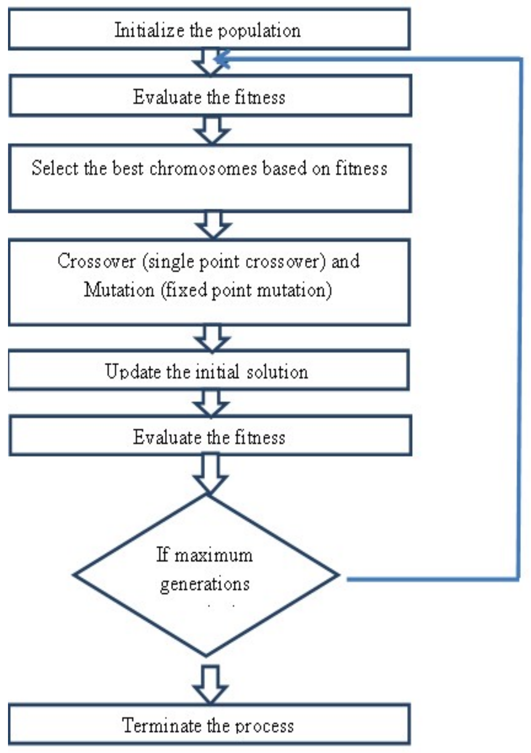

Initialize the population
Evaluate the fitness
Select the best chromosomes based on fitness
Crossover (single point crossover) and Mutation (fixed point mutation)
Update the initial solution
Evaluate the fitness
If maximum generations
Terminate the process

Figure 3.3 Flowchart For Genetic Algorithm

3.2.2 Steady State Genetic Algorithm (SGA):

Hereditary Calculation is versatile worldwide pursuit calculation in view of the developmental information of hereditary qualities. To figure out the improvement issues Hereditary Calculation is an inconsistent pursuit calculation applied. Cycles are represented as age and the populace is represented as chromosomes in hereditary calculation. As the combination pace of the customary GA is low, AGA is used in our proposed strategy to accelerate the union rate. The versatile GA is utilized with the assistance of Cauchy transformation in the change administrator, which will accelerate and upgrade the exhibition of entire GA process.

Versatile GAs are those whose boundaries, for example, populace size, the hybrid likelihood, or change likelihood are shifted while the GA is running. A basic variation could be where the transformation rate changes as indicated by the populace change

Operations of SSGA

Initial Phase: Initially the populations of the chromosomes x_i, $(i = 1,2,...N)$ are generated randomly. N Denotes the size of the population. The chromosome (x_i) contain the test cases randomly generated.

Fitness Evaluation: Fitness value of each parameter is calculated and the chromosome which has the highest fitness value is selected as the best chromosome.

$$fitness_i = \sum_{i=1}^{N} x^i{}_{if\,value} + x^i{}_{ratiovalue} + x^i{}_{linecoverage} + x^i{}_{loopcoverage}$$

Selection of Chromosomes: One or more parent chromosomes are selected based on the '$N/2$' best chromosomes which have maximum fitness and new solution is created.

Crossover: Single point crossover is performed at the crossover rate of (C_r) and hence ('$N/2$') offspring are obtained. In every crossover operation, (NC_r) genes are exchanged between corresponding parents.

Mutation: People are bothered probabilistically to get a change the people. Utilizing transformation administrator, there is a likelihood that another elements could show up because of progress in the chromosome. Cauchy change is utilized to transform the people as per the condition given underneath. Transformation is performed based on pre-decided changing likelihood. On the off chance that the Cauchy change is applied, irregular variable 'x' is a Cauchy conveyance.

In mutation process chromosome values are differed according to the possibility after that produced novel chromosome. Here we randomly mutate and produce the new solution. The replaced genes are the randomly generated genes without any repetition within the chromosome.

Updating: After the transformation cycle, new chromosome is made. Then, at that point, we supplant the ongoing chromosome with new chromosome. This interaction is known as the updation of introductory arrangement. In the event that the wellness worth of novel chromosome is more prominent than the current chromosome, we can pick the clever chromosome as the best chromosome.

Termination Criteria: The process is continued until it meets the termination criteria. Flowchart for the suggested method is illustrated in Figure 3.4 It's shown in beneath

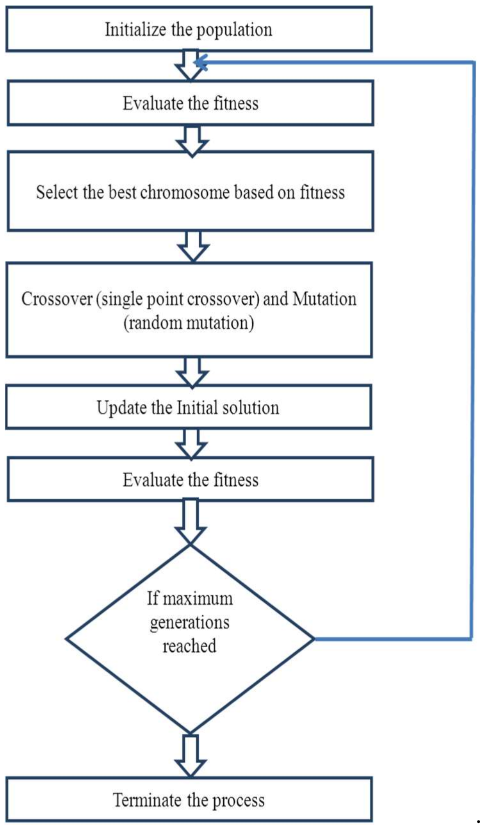

Initialize the population
Evaluate the fitness
Select the best chromosome based on fitness
Crossover (single point crossover) and Mutation (random mutation)
Update the Initial solution
Evaluate the fitness
If maximum generations reached
Terminate the process

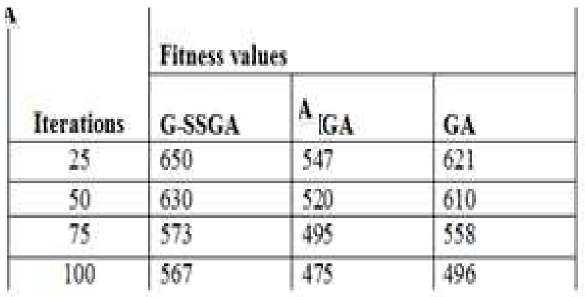
Figure 3.4 Flowchart For The Proposed SSGA

The ideal outcome is differentiated to all the capability in the wake of come by result from the versatile hereditary calculation. In the event that any capability of the application not in the ideal outcome then the subsequent capability can be wiped out from the application. Consequently that versatile hereditary calculation achieves diminishing of shortcomings rate in light of the ideal experiment. This strategy, the course of SSGA is choosing the ideal impeccable experiments which are reasonable for the info application. The bogus decrease is hence acquired utilizing the enhancement calculation. We get the advanced outcome in light of the wellness esteem that we relegate for SSGA. The experiments that we provide for the improvement calculation are handled in light of the wellness values. Experiments with high wellness esteem show the bug free application. Thus by utilizing the SSGA we get required experiments for our proposed strategy.

4. RESULTS AND DISCUSSION

In the experiment, the researchers have utilized the GA, Adaptive Genetic Algorithm and SSGA for Reducing Interactive Faults Based on Optimal Test Cases in Directed Random Testing**.** By observing the Table4.1 the fitness value for the suggested technique is verified to be superior to the technique where SSGA, GA is applied.

Table 4.1: Fitness Comparison For Different



A	Fitness values		
Iterations	G-SSGA	A IGA	GA
25	650	547	621
50	630	520	610
75	573	495	558
100	567	475	496

Iterations using, SSGA and GA.

The Figure 4.1 given below shows the graphical representation of fitness value using the SSGA, and GA. In this, the fitness values that are obtained from adaptive genetic algorithm as well as that we obtain from Genetic algorithm and SSGA are being plotted. From the graph it is evident that the proposed method using SSGA delivers better fitness value when compared to that of the AGA and GA

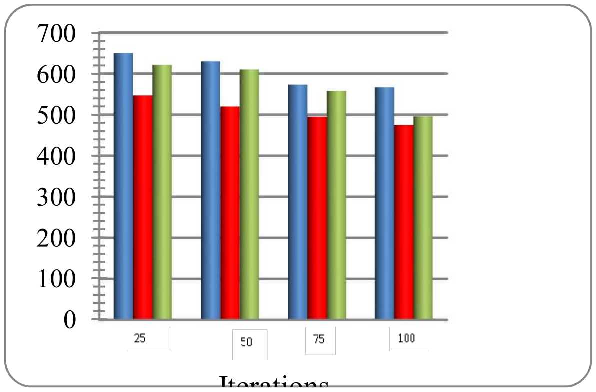


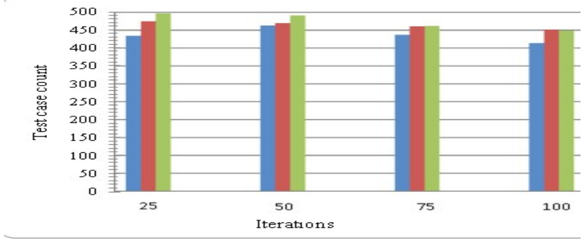
Figure 4.1: Graphical Representation Of Fitness Value For Different Iterations Of SSGA, And GA

The table 4.2 given below shows the test count value obtained 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, here the test cases before optimization are 661

Table 4.2: Test Count Value After Optimization

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

The Figure 4.2 given below, shows the graphical representation of test count value which is obtained after optimization. From the graph, it is an evident that the test case count has been reduced to a greater extend when compared to those obtained before optimization in SSGA only



The table 4.3, 4.4 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 reduce automatically in MGA only

Table4.3: Time Usage For Different Iterations.

Iteration	Time usage (milliseconds)		
	G-SSGA	AGA	GA
25	4256	4521	5214
50	4399	4698	5365
75	4525	4724	5368
100	4502	4792	5741

The Figure 4.3 and Figure 4.4 have given below shows the graphical representation of time and memory usage for different iteration of our proposed technique

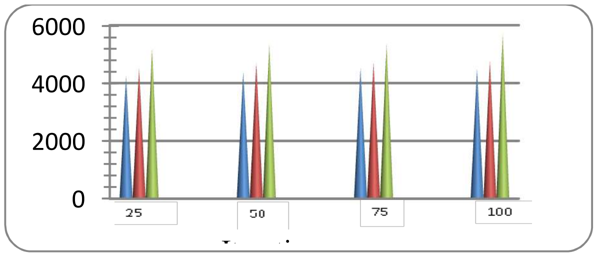


Figure 4.3: Graphical Representation Of Time Usage For Different Iterations

Table 4.4: Memory Usage For Different Iterations

Iteration	Memory usage (bits)		
	G-SSGA	AGA	GA
25	3486136	3576104	3699336
50	3001288	3419328	3535728
75	2973680	3041121	3289761
100	2188312	2974721	3146992

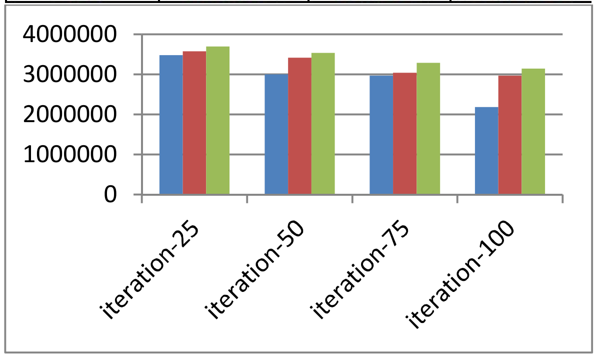


Figure 4.4: Graphical Representation Of Memory Usage For Different Iterations

From the table it is observed that the interactive faults can be highly detected from the random testing rather than other testing's like regression and partition testing. If the test suite is large, then Random Testing will give better results.

6. CONCLUSION AND FUTURE WORK

In this work, the creators have proposed a technique for diminishing intuitive shortcomings in view of ideal experiments in coordinated irregular testing. The carried out strategy is utilized to decrease the intelligent deficiencies in view of G-Consistent State Hereditary Calculation. Here, Consistent State Hereditary Calculation produces the ideal outcome which lessens the restricted information sources. To decrease the issues, the proposed strategy utilizes the inclusion measurements. The outcome shows that our proposed technique eliminates the vagueness of haphazardly created experiments and delivers the ideal outcomes than AGA and GA. In future, specialists can take on experiment circulation measurements as experiment determination standards for getting high inclusion experiments rapidly and can utilize some other met heuristic pursuit devices to deliver impeccable experiments like Half and half and Equal Hereditary Calculation can measure up to variations of Molecule Multitude Advancement.

REFERENCES

[1]. Andrea Arcuri, Iqbal.M.Z. and Lionel Briand, "Formal Analysis of the Probability of Interaction Fault Detection Using Random Testing", IEEE Transactions on software engineering, vol.38, September 2012.

[2]. Andrea Arcuri and Lionel Briand, "Random Testing: Theoretical Results and Practical Implications", IEEE transactions on Software Engineering, vol. 38, no. 2, April 2012.

[3]. James H. Andrews, Tim Menzies, Memberand Felix C.H. Li , "Genetic Algorithms for Randomized Unit Testing", IEEE transactions on software engineering, vol. 37, no. 1, february 2011

[4]. Vahid Garousi, "A Genetic Algorithm-Based Stress TestRequirements Generator Tooland Its Empirical Evaluation", IEEE transactions on software engineering, vol. 36, no. 6, December 2010

[5]. Reza MeimandiParizi, Abdul Azim Abdul Ghani, Rusli Abdullah, and RodziahAtan, "On the Applicability of Random Testing for Aspect-Oriented Programs", International Journal of Software Engineering and its Applications Vol. 3, No. 4, October, 2009.

[6]. Bo Yu, Zeliang Pang, "Generating Test Data Based on Improved Uniform Design Strategy", International Conference on Solid State Devices and Materials Science, vol.25,pp.1245-1252, 2012.

[7]. Lin Padgham, Zhiyong Zhang, John Thangarajah, and Tim Miller, "Model-Based Test Oracle Generation for Automated Unit Testing of Agent Systems", IEEE Transactions On Software Engineering,vol.39,no.9,1230-1244,2013.

[8]. Bestoun S. Ahmed, Mouayad A. Sahib, and Moayad Y. Potrus, "Generating combinatorial test cases using Simplified Swarm Optimization (SSO) algorithm for automated GUI functional testing", International Journal an Engineering Science and Technology,vol.17, pp.218-226,2014.

[9]. Leandro L. Minku, Dirk Sudholt, and XinYa, "Improved Evolutionary Algorithm Design for the Project Scheduling Problem Based on Runtime Analysis", IEEE Transactions On Software Engineering, vol.40,no.1,pp.83-102, 2014.

[10]. JunpengLv, Hai Hu, Kai-Yuan Cai, and TsongYueh Chen, "Adaptive and Random Partition Software testing", IEEE Transactions On Systems Man And Cybernetics: Systems, vol.44, no.12, pp.1649-1664, 2014.

[11]. Phil McMinn, Mark Harman, KiranLakhotia, Youssef Hassoun, and Joachim Wegener, "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, vol.38,no.2,pp.453-477, 2012.

[12]. Andrea Arcur, "A Theoretical and Empirical Analysis of the Role of Test Sequence Length in Software Testing for Structural Coverage", IEEE Transactions On Software Engineering, vol.38,no.3,pp.497-519,2012.

[13]. Tao Yuan, Xi Liu and Way Kuo, "Planning Simple Step-Stress Accelerated Life Tests Using Bayesian Methods", IEEE Transactions on Reliability, Volume: 61,Pp: 254 - 263, March 2012.

[14]. Wu, J."Stress testing software to determine fault tolerance for hardware failure and anomalies", AUTOTESTCON, 2012 IEEE, Sept. 2012.

[15]. Jianhui Jiang and Jipeng Huang, "System Modules Interaction Based Stress Testing model",IEEE transaction on application software,March 2011

[16]. Daning Hu, J.Leon Zhao and Zhimin Hua, "Banking Event Modeling and Simulation in Scenario-Oriented Stress Testing",springer,Volume 108,pp 379-389, 2012

[17]. EduardasBareisa, Vacius Jusas, Kestutis Motiejunas and Rimantas Seinauskas "the non-scan delay test enrichment based on random generated long test sequences", issn 1392 – 124x information technology and control, Vol. 39, No. 4, 2010.

[18]. Bo Zhou, Hiroyuki Okamura and Tadashi

Dohi, "Enhancing Performance of Random Testing Through Markov Chain Monte Carlo Methods", IEEE transactions on computers, journal of latex class files, vol. 6, no. 1, January 2011.

[19]. Ah-Rim-Hom "Measuring Behavioral Dependency for Improving Change Proneness Prediction in UML Based Model" The Journal of Systems and Software 83 (2010) 222–234

[20]. ZhiQuan Zhou, ArnaldoSinaga and Willy Susilo "On the Fault-Detection Capabilities of Adaptive Random Test Case Prioritization: Case Studies with Large Test Suites", Hawaii International Conference on System Sciences, pp.5584-5593, 2012.