Journal of Theoretical and Applied Information Technology
15th December 2016. Vol.94. No.1
© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

TEST SUITE MINIMIZATION AND EMPIRICAL ANALYSIS OF OPTIMIZATION ALGORITHMS

[1] SRIVIDHYA. J, [2] DR. R. GUNASUNDARI

[1] Research Scholar, Department of Computer Science, Karpagam University.
[2] Associate Professor & Head, Department of Information Technology, Karpagam University.
E-mail: [1]jsrividhya.ku2011@gmail.com, [2]gunasoundar04@gmail.com

ABSTRACT

Test suite minimization approaches attempt to minimize the huge number of test suites, time and cost. Some of the methods are additionally considered for reusing the test suites at the time of software maintenance which is done by avoiding the redundant test suites from available test suites. Main drawback of these methods are time consuming processes which directs to fault software testing process. The previous studies have presented that in sometime this test suite minimization process reduction is severe. This paper presents three different kinds of test suite minimization approaches such as Gravitational Bee Search Algorithm with Fuzzy Logic (GCSAFL), Non-dominated Sorting Genetic Algorithm (NSGAII) and Ant Colony Optimization (ACO) with Particle Swarm Optimization (PSO). These approaches initially investigate the minimization process and then shows the comparison results in terms of efficiency, time, cost, path coverage, and fault coverage. The experimental results show that a combination of algorithms presents the better results.

Keywords: *Test Suite Minimization, Gravitational Bee Search Algorithm with Fuzzy Logic (GCSAFL), Non-dominated Sorting Genetic Algorithm (NSGAII), Ant Colony Optimization (ACO) with Particle Swarm Optimization (PSO).*

1. INTRODUCTION

Software testing is most essential as well as expensive work in software development process. For software testing process, the test suites are used which is also known as test cases and it can run on the software system to identify the errors. These kinds of test suites are required to define with their specification of requirement. The structure of the test cases is well-defined in IEEE standard [1]. A set of given test cases execution, inputs and expected results are implemented for a specific objective, for example to use a specific program path or to validate the compliance with a particular requirement [2].

Basically, the test suite contains all the test cases which needs to satisfy the all kinds of test requirement. In that case, the software is fully developed but test suite raises greater. This situation becomes unfeasible to run all the test cases which direct to high testing cost. Thus, the test suite minimization approaches are utilized to minimize the test suite and this process directs to reduce the testing cost respectively. The test suite minimization approaches create a representative set from the given original test suite which satisfy the given requirements as original test suite, however, it comprises less number of test cases. Here, the redundant test cases are eliminated from the given test suite [3]. The redundant test case is defined as to satisfy the same requirements used by other test cases.

Therefore, the many researchers have examined the notion that when the number of test cases in the same test suite perform the same components of program, that test suite can be minimized to a reduced test suite that assures the equivalent test suite coverage [4]. The main motivation for this test suite reduction is straightforward which means by minimize the test suite size and it can be used for reducing the cost of managing, validating and executing those test suites over the upcoming releases of the software. A probable drawback of test suite minimization, however, which processes significantly alter the test suite's fault-detection capabilities. This tradeoff between the time required, managing, validating test suites and the fault. It shows that the effectiveness of test suite's fault detection process is central to any kind of decision to apply test suite minimization.

© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Modern studies about test suite minimization process suggest, that the previously proposed test suite minimization processes might create dramatic reduction in test suite size, at minimum cost with the effectiveness of fault-detection process. But they also have some of drawbacks, for example time consuming process. This paper presents new three different kinds of test suite minimization approaches such as Gravitational Bee Search Algorithm with Fuzzy Logic (GCSAFL), Non-dominated Sorting Genetic Algorithm (NSGAII) and Ant Colony Optimization (ACO) with Particle Swarm Optimization (PSO). These approaches initially investigate the minimization process and then show the comparison results of efficiency. This experimental results show that they are unique and quite different from previous researches on comparing five efficiency parameters as objective criteria such as Number of iterations, Execution time, Path coverage, fault coverage and statement coverage, whereas the most of the other researches only compares maximum two or three objective criteria on their results.

Also most of the other researches focus on either test suite minimization or the test-suites prioritization. This proposed method uniquely applies both minimization and prioritization techniques to achieve the most efficient minimized test suite.

This paper is organized as follows. Section 2 explains the related works and research in this specific area. Section 3 explains how the test suite minimization process is achieved by Non-Dominated Sorting Genetic Algorithm (NSGAII). Section 4 talks about a hybrid Particle Swarm-Ant Colony Optimization approach for minimizing the test suites. Section 5 explains the new proposed Gravitational Bee Search Algorithm with Fuzzy Logic (GCSAFL). Section 6 discuss about the comparison results of the above explained three algorithms in terms of Number of iteration, Path coverage, Execution Time, Fault coverage and Statement Coverage and Assumptions and Limitations. Finally the conclusion and future work presented in section 7.

2. RELATED WORK

In [5] author presents a multi-objective test suite minimization process with the consideration of minimum execution time and maximum statement coverage. This proposed work mainly concentrates on including a multi objective minimization approach utilizing clustering techniques and minimal hitting set and also use the weighed distance function for achieving appropriate clusters. The minimal hitting set and mixed variable type are extracted by Hitting Set Directed Acyclic Graph (HS_DAG) algorithm. But it is not always practically possible to compute specific weights for all the objectives, particularly when there are many objectives. Only a relation between objectives is definite sometimes, than their weightage.

In [6] author uses the Extended Finite State Machine (EFSM) model for test case minimization process and in this work analyze the dynamic dependencies such as control dependence and data dependence along with their different interaction patterns. This proposed approach is called as dynamic interaction-based prioritization, modifies the existing approach to improve the fault detection capability and it also considers the optimization process to minimize the resource cost. The contribution of this work is towards prioritization, whereas the minimization results in terms of most common objectives like coverage and size of the test suite were less efficient.

In [7] authors present a method, by modifying an existing heuristic approach for test suite minimization process. In this approach, the random test data generation is done by Genetic algorithm and this test data is given to the minimization process, which is used for minimizing the total number of generated test cases. This process is named as Hybrid Algorithm (HA). In this work, only two objectives size and time of execution are taken into consideration for minimization, but there are possibilities to lose the effectiveness in terms of coverage.

In [8] authors present a novel approach to select a subset of given test cases which uses the set of requirements for data flow testing. To show the effectiveness of the proposed algorithm, both the Bi-Objective Greedy (BOG), and existing Harrold Gupta and Soffa (HGS) algorithms are employed to the create the test suites. This work concentrates on minimization in terms of requirement coverage and size, but there is possibility for lack of fault detection.

In [9] author presents a two levels prioritization method for selecting the test cases and their particular sequence. Initially, analyze the modified code blocks and then present the comparative interaction with the other modules and analyzed. Depends on the number of interacted modules, first level of prioritization is processed. The prioritized

© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

modules are then examined in terms of their criticality. The criticality is classified based on the fault type or the error type in the particular module. Depends on the criticality, the cost is examined to prioritize test cases. At the end of this process, the dynamic programming method is executed to find the appropriate test sequence, thus the regression testing cost is reduced. This work only employs the test case prioritization as EFSM method, with less contribution to minimization.

In [10] author presents the optimal test case generation process which depends on the model driven environment utilizing UML activity diagrams. In the traditional test case generation process, huge number of duplicate test cases are created. This proposed work generates optimal test suite and reduce the generated test suite by the model driven testing process. In this proposed design, the element like modified activity diagram is used for finding the uncommon and common test cases. It focuses on the uncommon test cases for further filtering processes to efficiently utilize the resources. This work focus on reducing the test suites in the test case generation itself, however there are always possibilities to minimize the test suites post generation.

3. NON-DOMINATED SORTING GENETIC ALGORITHM (NSGAII)

The NSGAII is different from traditional Genetic Algorithm (GA) process. The proposed approach implements the Pareto-ranking method in GA process which provides the efficient test suite minimization process. In this process initially the test case are selected by traditional GA and then the ranking process is done by dominance rule from a Pareto front. This work has three different kinds of steps as follows

- Producing test cases
- Formulating multiple objectives
- Finding optimal solution

The initial test suites are generated randomly (10 sample test suites are considered to-be-minimized) and defined as T1, T2, T3,…T10. These test suites are denoted as binary strings and each test case in a binary string is represented as a bit (1 or 0). Each test case has their own coverage X1, X2, X3,…Xn as presented in figure 1. The length of a binary string represents the maximum number of test cases in that test suite. Each binary string creates a test suite and any such test suite in the set can be a possible solution. The initial test suites are then sorted in non-dominated fronts.

	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
T1	1	0	0	1	0	0	0	1	1	1
T2	0	1	1	0	0	1	0	1	1	0
T3	0	0	1	1	0	1	0	1	0	1
T4	1	1	1	0	0	1	0	1	1	0
T5	1	0	1	1	1	0	0	1	0	1
T6	0	1	0	0	1	1	0	0	0	1
T7	1	1	0	0	1	1	0	1	0	1
T8	1	0	1	1	0	0	1	1	1	0
T9	1	0	0	0	1	1	0	1	0	0
T10	1	1	0	0	0	1	1	0	1	0

Figure 1: Chromosome Structure

3.1. Non-dominated Fronts

The obtained test suites are ranked based on two main objectives such as maximum branch coverage and minimum test cases. These objectives are examined by different metrics such as coverage and size respectively. The size is defined as the number of test cases and the coverage is defined as the number of branches. Based on these two different metrics, the non-dominated fronts are created which Pseudo code presented in figure 2.

```
Let F be a front
S1,S2,S3be the test suites
Begin
    Step 1: Compare S1, S2 and S3
      if S1, S2 and S3 are better than in at least in one objective and
                not worse when compared to other test suites in F-1
    Step 2: Form F
    Step 3: F++
End
```

Figure 2: Sorting Test Suites

In the sorted set of test suites, the test suites in the first front are more dominant than the other fronts, the sample coverage of test cases are presented in figure 3.

Coverage/ Test cases	X1	X2	X3	X4
T1		Y		Y
T2	Y	Y		Y
T3	Y			
T4	Y		Y	

Figure 3: Sample Coverage

3.2 Test Suite Detection

© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Detect the least qualified test suite from the available test suites is equal to electing the appropriate test suites. This approach uses the crowding distance operator to select the appropriate test suites. The coverage measure is used for computing the crowding distance of the test suites is defined as follow

$$cd_{cv} = (cv_s+1) - (cv_s-1) \quad (1)$$

Where $c_{\cdot\ c}$ is defined as the crowding distance and c is defined as the coverage, the test suite's measured size is computed by the follows equation.

$$cd_{sz} = (sz_s+1) - (sz_s-1) \quad (2)$$

The overall crowding distance is defined as follows

$$cd = cd_{cv} + cd_{sz} \quad (3)$$

```
Let fs be the front size
    sz be the set size
Begin
for each front F
If( sz>= fz)
        sz=sz-fs
        Insert sz to the set
        F++
Else
        Calculate cd
        Organize test suite in descending order based on cd
        Choose first set of test suites
End
```

Figure 4: Test Suite Minimization

The overall test suite minimization process is presented in the form of Pseudo code in figure 4. This proposed Non-dominated Sorting Genetic Algorithm efficiently minimizes the test suite respectively.

4. HYBRID PARTICLE SWARM-ANT COLONY OPTIMIZATION

This section elaborates, how the test suite minimization process is achieved by heuristic hybrid algorithm named as Particle Swarm Ant Colony Optimization Algorithm (PS-ACO). This hybrid approach aims to remove or reduce the redundant test cases and retain the efficient test suite from the given test suite set. Initially, the particles are randomly distributed to all over the given test suites. The number of user requirements that must be fulfilled is the test suite's position and the displacement between original test suite position and actual test suite position, which is defined as velocity. For instance, consider the test suites having four different test cases which means four requirements as in general is presented in figure 5.

	X_1	X_2	X_3	X_4	Value of test suite (T_i)
T_1	1	0	1	0	1010
T_2	1	0	0	1	1001
T_3	1	1	1	1	1111
T_4	0	0	0	0	0000

Figure 5: Test Suites Values based on Requirements mapping

After this process, each and every test suite (particle) is checked for the satisfaction of user requirements. If test cases are not in a processed test suite which means they does not satisfy the user defined requirements. They have to be reassigned to any other new test suites in that same position or eliminate the specific test suite. After this process, identify the local best (P_t^a) test suite and subsequently find global best (P_t^g) with their objective function $F(P_t^g)$. Considering the test cases X1 and X3 satisfies all the requirements, the corresponding test suites T1 and T3 are the locally best.

The same process is executed for the ants. Consider a particle's current position which is associated with ant's current position. If the ant's current position is better than the particle's, then set that current position of ant as the current position of particle and find the objective function $F(A_t^\beta)$.

Repeating this process until all the given constrains are satisfied, the resulted test suite is more efficient than the original. It is proved that the resulted test suite meets all the requirements during testing which also shows it is more efficient on removing redundant test cases and faults detection.

5. GRAVITATIONAL BEE SEARCH ALGORITHM WITH FUZZY LOGIC (GCSAFL)

This section focus on the GCSAFL system to minimize the test suite by finding the optimal set of test cases which should provide better or same coverage as the original test suites. The test case minimization process is done by Gravitational Bee Search (GBS) algorithm which is integration of two different algorithms such as artificial bee colony and gravitational search algorithm. After finding

© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

the minimized test suites, the Fuzzy operation is used for prioritization. This process is used for efficiently running the optimum test cases at the time of execution. In this approach, a set of test suite is chosen from available test suites. The bees are treated as agents, finds the minimum number of test cases. The bees start the foraging operation on randomly selected test cases and add new test cases. However, after adding the newly discovered test cases, the bees return to their hive and their information is communicated by Gravitational Search process. The proposed GCSAFL algorithm is as follows

GCSAFL Algorithm

Step 1: Initialize the population of test cases $T\_c[T][D]$ by following equation

$$T\_c[i][j] = r * (u - ll) + ll$$

Where u is defined as the Upper bound and ll is defined as the lower bound

Step 2: Initialize $p\_h\_c[T] = 0, t[T] = 0 \forall T\_c\_s$

Step 3: $C = 0$

Step 4: define the objective value (HV) $= \begin{cases} 100 & \text{if any new path} \\ -1 & \text{otherwise} \end{cases}$

Step 5: define the Fitness Function

$$(f) = \begin{cases} 1 / (\mathrm{HV} + 1) & \text{if } H > 0 \\ 1 + \mathrm{fabs}(\mathrm{HV}) & \text{otherwise} \end{cases}$$

Where HV is defined as hive

Step 6:

$$S\_p\_h\_c[T] = \begin{cases} 1 & \text{if covers any new path} \\ 0 & \text{otherwise} \end{cases}$$

Step7:
$M\_t\_c\_w\_h\_o\_v = 100$

Step8:
$D\_it = 1$,
$c\_ = n\_o\_p\_hs\_c$

Step 9: Repeat steps 10 to 19 till either $m < M$

Employed Bee Phase

Step 10: Create new test cases, $s[i]$, in the neighborhood of $T\_c[i][j]$ for the employed bees utilizing the following equation:

*Solution[k] = Test case [i][k] + (Test case[i][k] – Test case[n][k]) * (r-0.5) *2*

Where n is defined as the any test case no. in the given test suite, k is defined as the parameter to change in $T\_c[i][j]$.

Step 11: Compute fitness and Objective values for the neighbor bee.

Step 12: Apply Gravitational Force process between *Test_case[i][j] and solution[i]*.

Step 13 : Increases the count + 1 with newly discovered path

Step 14: If not Increased
$t[i] = t[i] + 1$

End the Phase of Employed Bee

Step 15: Compute the test cases probability values by the fitness values is as follows:

$$P_i = 1 - (f_i/f_m)$$

Where P_i values are standardized into $[0, 1]$

Onlooker Bee Phase

Step 16 : If $P_i > r$

Generate new test cases $s[i]$ using onlookers bees from the defined test cases $T\_c[T][D]$ and update the c_p and v V of new onlooker bees

Step 17: Else

Alter the test case $T\_c[i][j]$

Step 18: Repeat steps 11 to step 14.

E o O B Pha

Scout Bee Phase

© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Step19: if t_1 $[i] > m$

Step 20: Alter the test case T $\_c_1$ $[i][j]$

Step 21:
If
Altered test case does not cover any new path,
Then
Replace the new randomly produced test case which is done by s_1 w d .

E o S_1 B Pha

Step 22: It $= it$ $+ 1$

6. RESULTS AND DISCUSSIONS

This section presents the comparison analysis of the proposed multi objective test suite minimization method GCSAFL with other two existing minimization methods such as NSGAII, and PS_ACO. Here we consider, the number of iterations, path coverage, fault coverage, statement coverage, execution time as multiple objectives of this problem and criteria for this analysis is that the minimized test suites that are better in at least one objective and not worse in any when compared with the other approaches. The GCSAFL method's minimized test suite is compared with NSGAII, PS_ACO's minimized test suites in term of all the above mentioned objectives.

Table 1 shows the Test Cases Results as below.

Table 1: Test Cases Results

Test Case/Faults	Test suite 1	Test suite 2	Test suite 3	Test suite 4	Test suite 5	No. of Faults Covered	Execution Time
Test 1	X			X		2	4
Test 2	X	X	X		X	4	7
Test 3	X			X		2	5
Test 4		X	X	X	X	4	11
Test 5		X	X		X	3	6
Test 6	X	X		X		3	9
Test 7	X		X	X	X	4	5
Test 8		X		X	X	3	8

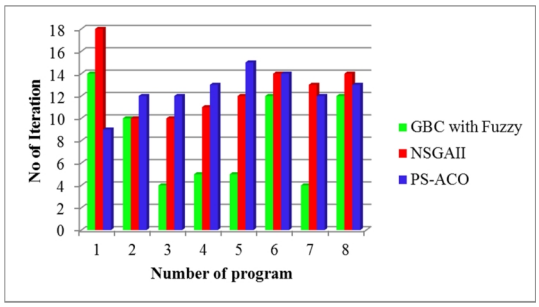

Figure 6: Number of Iteration

It is evidently observed from the Figure 6, that the proposed test case minimization methods GCSAFL, NSGAII and PS_ACO give the promising results in terms of number of iteration, but the GCSAFL shows the efficient iteration results when compared with other two proposed optimization methods such as NSGAII, PS_ACO. The GCSAFL takes the minimum iteration to process for all the test suites respectively.

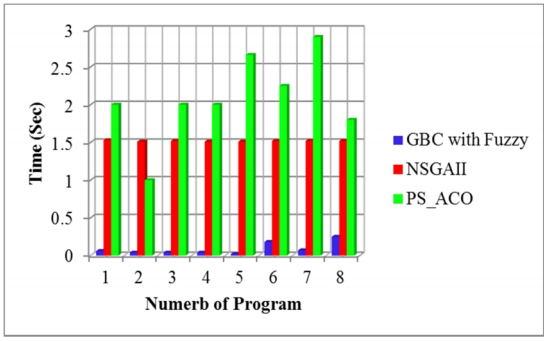

Figure 7: Time

It is evidently observed from the Figure 7, that the proposed test case minimization methods GCSAFL, NSGAII and PS_ACO give the promising results in terms of processing time, but the GCSAFL shows the minimum time when compared with other two proposed optimization methods such asNSGAII, PS_ACO. The GCSAFL takes the minimum time to process for all the test suites respectively.

© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

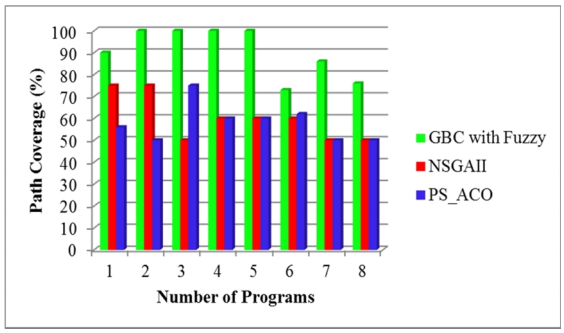

Figure 8: Path Coverage

It is evidently observed from the Figure 8 that the proposed test case minimization methods GCSAFL, NSGAII and PS_ACO give the promising results in term of path coverage, but the GCSAFL shows the maximum path coverage when compared with other two proposed optimization methods such as NSGAII, PS_ACO. The GCSAFL takes the maximum path coverage to process all the test suites respectively. However other two algorithms also show the granted results.

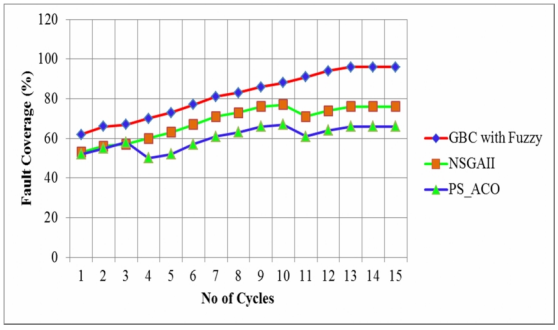

Figure 9: Fault Coverage

It is evidently observed from the Figure 9 that the proposed test case minimization methods GCSAFL, NSGAII and PS_ACO give the promising results in terms of fault coverage, but the GCSAFL shows the maximum fault coverage when compared with other two proposed optimization methods such as NSGAII, PS_ACO. The GCSAFL takes the maximum fault coverage to process all the test suites respectively. However other two algorithms also show the granted results.

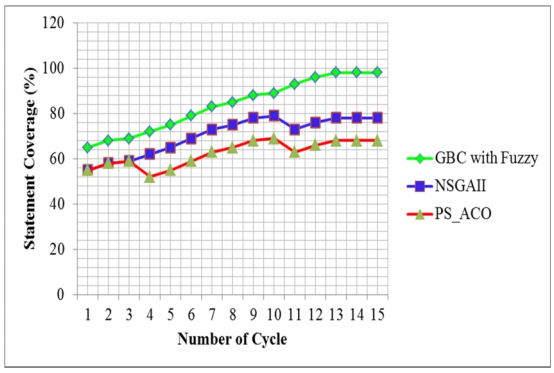

Figure 10: Statement Coverage

It is evidently observed from the Figure 10 that the proposed test case minimization methods GCSAFL, NSGAII and PS_ACO give the promising results in terms of statement coverage, but the GCSAFL shows the maximum statement coverage when compared with other two proposed optimization methods such as NSGAII, PS_ACO. The GCSAFL takes the maximum statement coverage to process all the test suites respectively. However other two algorithms also show the granted results.

Based on the comparison analysis above, the proposed method is approximately 15% to 20% more efficient than the existing test suite minimization methods significant to the number of test cases and execution time. Many times, by many researchers, it is proven that the hybrid approaches are more efficient than the base approaches, likewise the proposed hybrid approach with the secondary prioritization process layer provides promising results. With that said, the current and future researchers needs to focus on hybrid approaches not only within minimization process but also with test case generation and prioritization.

Besides that, there are assumptions and limitations that, the proposed work needs a finalized requirement collection in the Software Requirement Specification document and a test data generator tool or manual process available to create initial test suite. And due to lack of quality of the sample compositions, the research was conducted only on few open source test samples like online booking by using own proprietary testing automation tool ECHO.

7. CONCLUSION AND FUTURE WORK

This paper presents three new different kinds of test suite minimization approaches such as Gravitational Bee Search Algorithm with Fuzzy

© 2005 - 2016 JATIT & LLS. All rights reserved·
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Logic (GCSAFL), Non-dominated Sorting Genetic Algorithm (NSGAII) and Ant Colony Optimization (ACO) with Particle Swarm Optimization (PSO) for efficient test suite minimization process. These approaches investigates the minimization process and then the results are compared in terms of efficiency, time, cost, path coverage, and fault coverage. The proposed method gives the promising results and it gives positive feedback in the evaluation process. The GCSAFL gives the promising results when compared with other two algorithms.

The future work is to implement the proposed algorithm and measure the accuracy and the performance of different type of requirements and applications with real data by using open source of testing automation tools.

REFERENCES

[1] Saif-ur-Rehman Khan, Aamer Nadeem, "TestFilter: A Statement-Coverage Based Test Case ReductionTechnique", IEEE International Multitopic Conference, PP. 275 - 280, 2006.

[2] Shin Yoo, Mark Harman, "Using hybrid algorithm for Pareto efficient multi-objective test suite minimization", The Journal of Systems and Software, Vol.83, PP.689–701, 2010.

[3] Rajvir Singh, Mamta Santosh, "Test Case Minimization Techniques: A Review", International Journal of Engineering Research & Technology (IJERT), Vol. 2 Issue 12, 2013.

[4]Neetu Dabas, Kamna Solanki, "Comparison of Code Coverage Analysis Tools: A Review", International Journal of Research in Computer Applications & Information Technology (IASTER), Vol.1, Issue 1, PP.94-99, 2013.

[5] R.Beena, S.Sarala, "Multi Objective Test Case Minimization Collaborated With Clustering and MinimalHitting Set", Journal of Theoretical and Applied Information Technology, Vol.69 No.1, PP.200- 210, 2014.

[6] Chris Nitin Adonis Petrus, M.S. Razou, M. Rajeev, M. Karthigesan, "Model-Based Test Case Minimization and Prioritization for Improved Early Fault Detection Capability", International Journal of Innovative Technology and Exploring Engineering (IJITEE), Vol.2, Issue-5, PP.205-210, 2013.

[7] P Maragathavalli, S. Kanmani, "Test Suite Minimization using Hybrid Algorithm for GA Generated Test Cases", International Journal of Computers & Technology,Vol. 6, No 1, PP. 279-286, 2013.

[8] Preethi Harris, Nedunchezhian Raju, "A Greedy Approach for Coverage-Based Test Suite Reduction", The International Arab Journal of Information Technology, Vol. 12, No.1, PP. 17-23, 2015.

[9] Monika, Paramjit Singh, "A Dynamic Programming Approach for Fault Optimized Sequence Generation in Regression Testing", International Journal of Scientific & Engineering Research, Vol.4, Issue 9, PP. 2580-2586, 2013.

[10] Hetal J. Thanki, S.M.Shinde, "Test Case Generation and Minimization using UML Activity Diagram in Model Driven Environment", International Journal of Computer & Organization Trends, Vol.9 No.1, PP.41-44, 2014.