International Journal of Computer Engineering & Technology (IJCET)
Volume 7, Issue 1, Jan-Feb 2016, pp. 26-35, Article ID: IJCET_07_01_004
Available online at
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=7&IType=1
Journal Impact Factor (2016): 9.3590 (Calculated by GISI) www.jifactor.com
ISSN Print: 0976-6367 and ISSN Online: 0976–6375

EXPERIMENTAL EVALUATION AND RESULT DISCUSSION OF METAMORPHIC TESTING AUTOMATION FRAMEWORK WITH NOVEL ALGORITHMS

CHITTINENI ARUNA
Research Scholar, Acharya Nagarjuna University, Andhra Pradesh
Assistant Professor, Department of CSE,
KKR & KSR Institute of Technology and Sciences,
Guntur, Andhra Pradesh, India

R. SIVA RAM PRASAD
Research Guide, Department of CSE and Head,
Department of IBS, Acharya Nagarjuna University,
Guntur, Andhra Pradesh, India

ABSTRACT

Metamorphic Testing is an attribute relations based testing, used to mitigate the test oracle problem in testing complex non-testable programs. MTAF stands for Metamorphic Testing Automation Framework, introduced to eliminate the human intervention in creating test cases, mapping the relations, executing the statements and identifying the errors from input programs. MTAF is especially designed to address the test oracle problem of two most popular non-testable program domains are Multi Precision Arithmetic (MPA) and Graph Theory (GT) applications. In this paper, the researcher explains the results of conducted experiments and identified bug information with MTAF. Several Multi Precision Arithmetic and Graph Theory related hidden bugs are discussed in this paper to show the performance of MTAF.

Key words: MTAF, Metamorphic Relations, Metamorphic Testing, Multi Precision Arithmetic, Graph Theory, Hidden bugs.

Cite this Article: Chittineni Aruna and R. Siva Ram Pra. Experimental Evaluation and Result Discussion of Metamorphic Testing Automation Framework with Novel Algorithms. *International Journal of Computer Engineering and Technology*, **7**(1), 2016, pp. 26-35.
http://www.iaeme.com/IJCET/issues.asp?JType=IJCET&VType=7&IType=1

1. INTRODUCTION

Since past decade Metamorphic Testing (MT) emerged as a better solution to the test oracle problem in testing non-testable programs. Test oracle value is an expected output value of the input test cases in testing. Test Oracle value is used to decide the resulted test output value for the given test case is correct or not. Designing the test oracle value for some test cases became cumbersome and even unattainable some times. This problem in testing is called "Test Oracle Problem" and the programs which are suffering from this problem are called non-testable programs. To test these non-testable programs without using test oracle Chen et al [1] introduced "Metamorphic Testing" in 1998. MT is an attribute relations based testing, which transforms the general attributes of the program to Metamorphic Relations to validate the test input results without test oracles. Christian Murphy et al [4] used the metamorphic testing to test the non-testable machine learning algorithms.

CH Aruna and Dr. R Siva Ram Prasad [2 and 3] applied the metamorphic testing to the most popular non-testable program domains are Multi Precision Arithmetic and Graph Theory. They proposed several metamorphic relations for the domain of MPA and GT to alleviate the test oracle problem. Automation is an important aspect for any testing methodology, which makes the testing scalable and reliable. Most of the present testing methodologies are having the respective automation tools to improvise the performance in terms of quality and cost. Since beginning the concept automation was not implemented for metamorphic testing with a comprehensive framework. CH Aruna and Dr. R Siva Ram Prasad proposed a comprehensive framework MTAF [5], to perform the metamorphic testing process without human intervention. This framework is designed with three modules and two novel algorithms, which are explained in the related work section of this paper.

In this paper, CH Aruna and R Siva Ram Prasad described the results of conducted experiments and identified bug information with MTAF. To conduct the experiments with MTAF, researcher selected more than 250 Java programs of Multi Precision Arithmetic and Graph Theory domains. After evaluating these programs with MTAF the obtained results and the revealed bug information is represented as result discussions. Experimental evaluations are denoting that the proposed MTAF revealed different kinds of hidden bugs from the input Java programs.

2. RELATED WORK

In this section, we discuss about the MTAF and its algorithms in brief. MTAF stands for Metamorphic Testing Automation Framework, which is a comprehensive framework to test the non-testable programs automatically without using any test oracle. This framework consists of three modules and two algorithms. The three modules are Base Test Suite Selection Module, Applicability and adoptability analysis module and MT Execution Engine module.

Base Test Suite selection module is the first module, which takes the java programs and specifications as input for testing. Base Test Suites of this module helps to extract the business logic contained executable statements. These statements are given to the second module as input for processing. CFG's of this module represents the program statements as a graph structure in an execution order. At this movement each statement of this graph structure is given to NDI algorithm for evaluation. This algorithm is especially designed for the evaluation of complex arithmetic expressions of input program statements. NDI algorithm [6] split the complex expression to low-level sub expressions for evaluation at each small sub expression level. This process

benefits the testing operation to find the hidden bugs at each sub expression level and to identify the exact error location of input expression.

Metamorphic Relations should be mapped before evaluating each sub expression with NDI algorithm. In order to map the relations against sub expression CH Aruna and R Siva Ram Prasad introduced the STCG algorithm [7]. STCG is used to map the suitable metamorphic relations for the programs expressions for metamorphic testing. Successful test cases are used by this STCG algorithm to generate the follow-up test cases. These follow-up test cases are more efficient than general test cases, which helps to identify the hidden bugs from the program.

Once the mapping of relations done by STCG, the sub expression is evaluated on Metamorphic Testing Execution Engine (MTEE). This engine is equipped with the Java Runtime Environment (JRE) for the evaluation. After the execution of expressions with metamorphic relations, the successful results are redirected to test case repository for next level follow-up test case generations and the failure test case sent to bug repository. At the end the bug repository contains the total revealed bugs of the metamorphic testing process.

Figure1 Home page of MTAF web tool

Infrastructure Setup

The Architectural view of the MTAF Frame work is designed with J2EE as Primary Language [8]. Java Servlets 2.5, JSP2.0 Web Technologies are used to design the server side components and Interactions with the Server. MTAF framework is designed by using the IDE Net beans 8.x [9]. All the User Interface screen and designed with HTML 5.0 and Cascading Style sheets 3 Model. Java Server pages are also a part the User Interface design. Proposed STCG, NDIA algorithms are developed implemented with JAVA programming Language.

The researcher designed and conducted many experiments on MTAF web tool to measure the accuracy, scalability and performance of the proposed MTAF with DELL Intel CORE i5 Processor at 1.70 GHz to 2.4 GHz. 8 GB RAM, 1TB Hard Disc, Windows-10 Operating System as core components.

3. EXPERIMENTAL EVALUATION AND RESULT DISCUSSION OF MTAF

MTAF Web tool is designed by the author, which is a practical implementation for MTAF framework. By using the above mentioned infrastructure this tool is developed and the experiments are conducted. Initially this tool displays the home page with five possible options as shown below:

First option “Home Page” to display the home page environment with MT definition as shown in figure 1. The second option and the third options are used for MPA testing with MT. Fourth option for Graph Theory Testing and fifth option for MT with DSE [9].

A. MPA program Testing Example 1

The selection of link MPA Testing (File) immediately displays the MPA testing screen. Choose file button of this screen helps to select the input java file from system location for Metamorphic Testing. By clicking on this, the researcher chosen a complex MPA program file as input. After testing the input program execution the results of a complex MPA addition expression are shown as:

((4855771.4367823456+2343678077.3537473)+(6138938.1588222+758991499.2290087)+(8545346464.634535335+93539753.43535353))

The above complex expression is fragmented by NDI algorithm into different sub Expressions like

(4855771.4367823456+2343678077.3537473),(6138938.1588222+758991499.2290087) and (8545346464.634535335+93539753.43535353).

To test this complex expression results, each sub expression is evaluated separately and results are verified distinctly. NDI and STCG algorithms of MTAF are working in a co-ordinated way to implement this metamorphic testing process as metamorphic relations identification, mapping, execution and follow-up test case process is applied for each sub expression of the above complex expression like below.

The first sub expression (4855771.4367823456+2343678077.3537473) is evaluated with The three metamorphic relations like (a+b)-(a-b) = 4.687356154707495E9, (a-(a-b)+b) = 4.687356154707495E9, 2(k-a)= 4.687356154707495E9 and results are compared The results generated with the three metamorphic relations are same, so it indicates there is no error in the first sub Expression. Then the next sub expression (6138938.1588222+758991499.2290087) is

evaluated with the same set of metamorphic relations ((a+b)-(a-b)) = 1.5179829984580173E9, (a-(a-b) +b) = 1.5179829984580173E9, 2(k-a)= 1.5179829984580173E9 and results are compared and indicating that there are no errors in the sub expression 2 also. After that the 3rd Sub Expression (8545346464.634535335+93539753.43535353) is evaluated with same set of metamorphic relations ((a+b)-(a-b)) =1.8707950687070656E8, (a-(a-b)+b)= .870795068707068E8, 2(k-a)= 1.8707950687070656E8 and results are compared and noticed that there is an error and generated the error report according to that.

To test this addition operation of above expression, The MTAF considered proven relation a+b = b+a as base test case. The other mapped relation for this addition operation is [(a+b=k) :!(k<=a) && !(k<=b)]. Based on the relation (a+b = k) the generated follow-up test case relation is [a + b = k than ((a+b)-(a-b)) = (a-(a-b)+b)= 2(k-a)]. This mapping process is clearly identifying the bug information .This information helps the developer to find and correct the bugs.

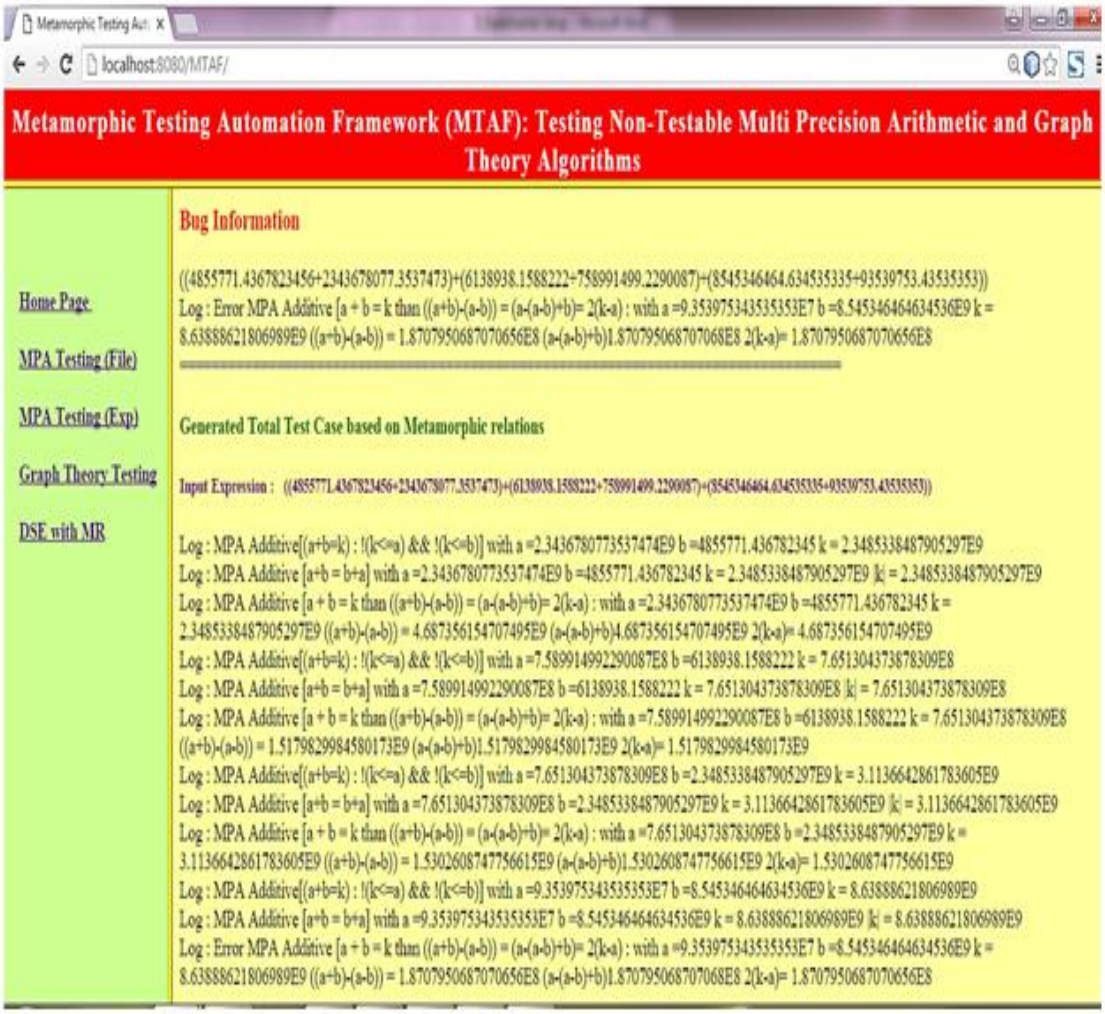

Figure 2 Identified bug information of MPA addition statement

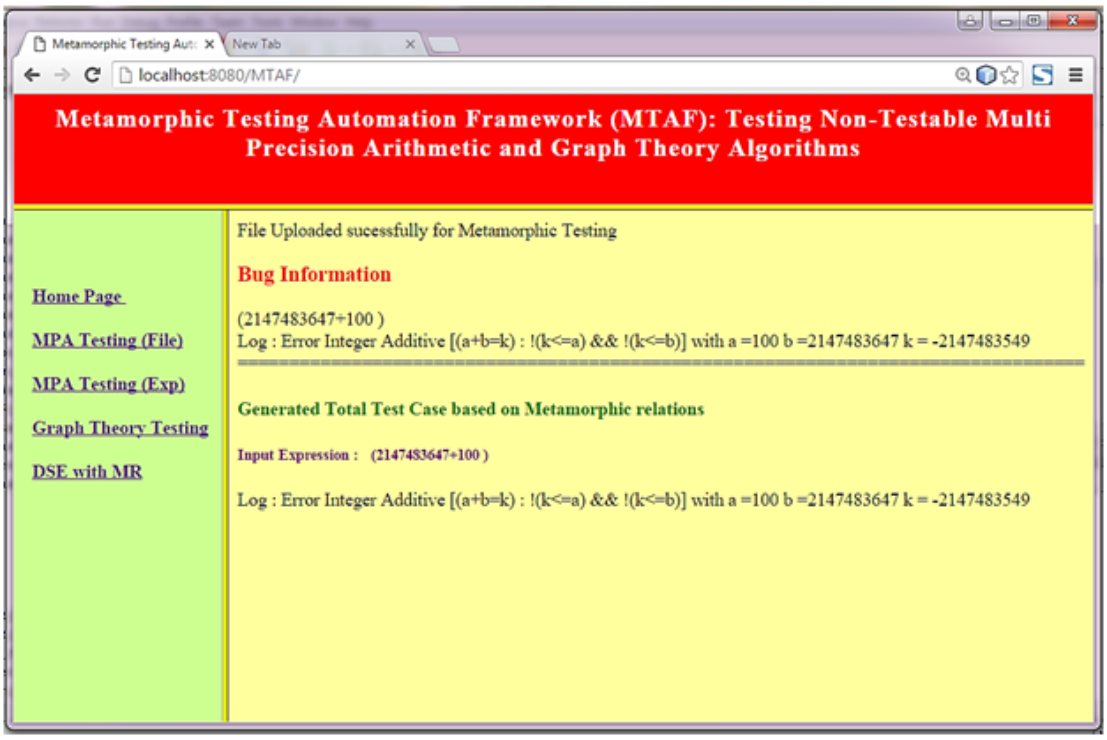

Figure 3 Identified Bug Information of MPA addition statements with overflow

B. MPA program Testing Example 2

The screenshot (figure 3) is displaying another hidden bug that is identified with the proposed MTAF frame work, which is very difficult to identify during other testing methodologies. This hidden bug is integer overflow problem, while dealing with simple arithmetic expressions. This type of error can't be noticed with compiler or any run time runtime environment. The given screen is showing addition operation with the given input values 2147483647 and 100. After evaluating the expression the generated result is '-2147483549'and represented as K. Proposed MTAF tool evaluated this expression with the metamorphic relation [a+b = k : !(K<a) && !(K<b)]. According to that Metamorphic Relation the resultant K value should not be less than given input value 'a'. But the generated resultant value k '-2147483549' is less than 2147483647. This evaluation indicates that, there is a bug in the given Expression. MTAF generated the same bug information in the screen as shown in figure 3.

Like this several bugs are identified with MTAF while evaluating the addition, subtraction, multiplication and division operations of MPA.

C. Graph Theory Testing Example

In this section the researcher represents the experimental results of Graph Theory Applications for creation and validation of shortest path algorithm, and verification of minimal spanning tree with different Screen Shots. To implement this experimentally ,the reseracher created graphs with the user interface, identified the result and validated with Metamorphic Relations. Initially the researcher explained the test results of Shortest Path Algorithm Creation,verification a Validation in different Screens. Later he explored the concepts of Testing for the verification of Minimal Spanning Tree.

To Test, the shortest path validation, author designed an undirected graph with respective distance values as shown below:

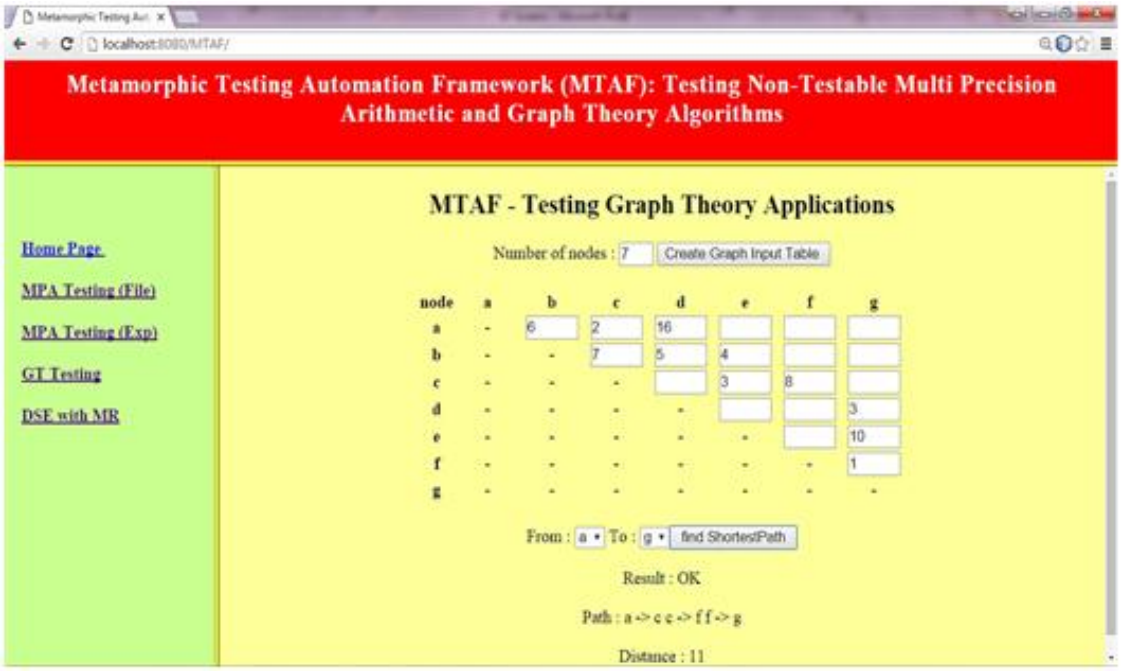

Figure 4 Seven nodes graph with shortest path and distance values

By using our shortest path algorithm implementation, researcher generated the shortest path for the above graph with seven nodes and distance values. Based upon the our implemented algorithm the shortest path resutl is a->c, c->f, f->g with distance value 11.

This shortest path should be verified and validated to assure the relaibility. MTAF generated the below test cases to test the given graph theory shortest path values as shown in below figure 5, 6 and 7.

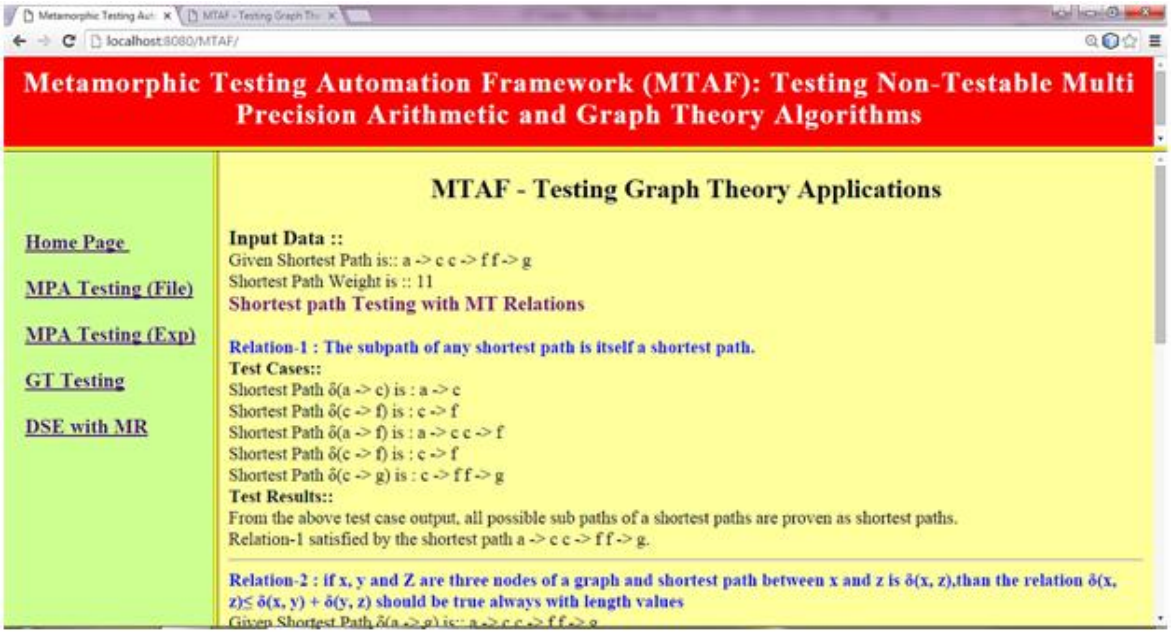

Figure 5 Generated Test Cases set-1 for Shortest Path Validation by MTAF

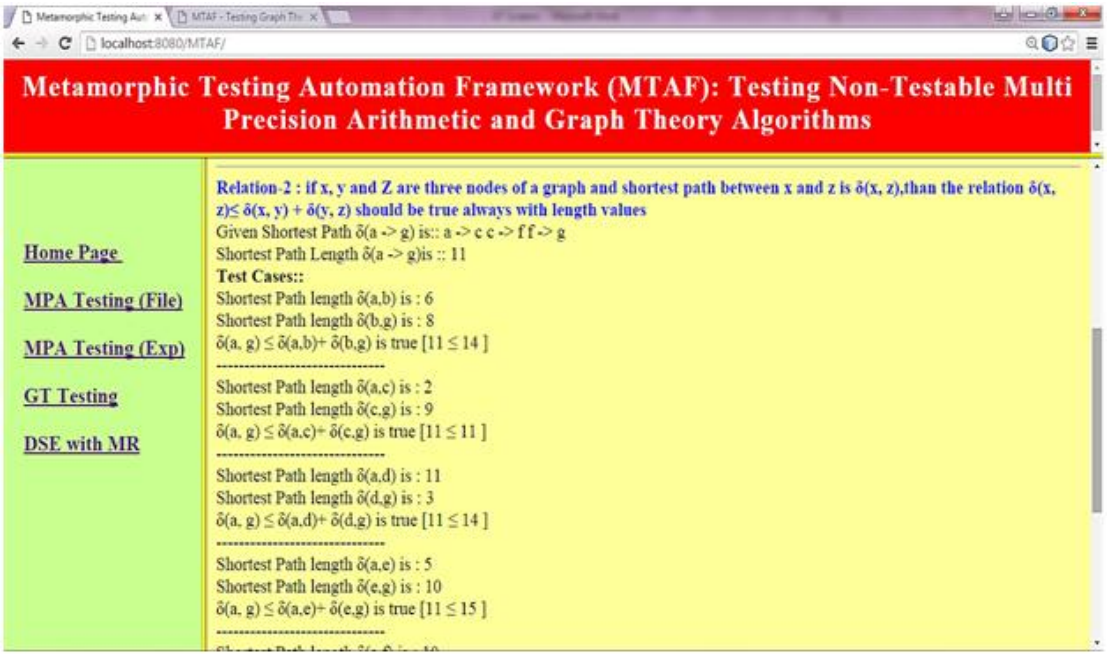

Figure 6 Generated Test cases set-2 for Shortest Path validation

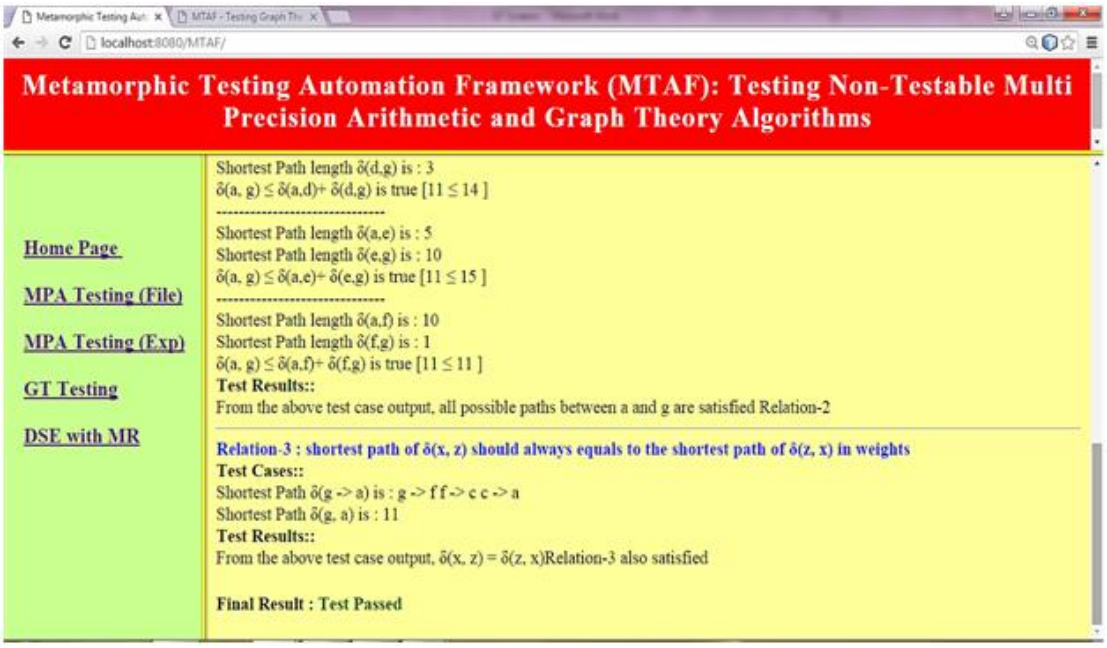

Figure 7 Generated Test cases set-3 and Test Resluts for shortest Path Validation

To Validate the given Shortest Path ,MTAF considered distinct Metamorphic Relations specified for the Shortest Path Algorithm, Test Cases are generated based on these Metamorphic Relations are results are evaluated. The figure 5 is showing the generated test cases for the first Metamorphic Relation is "The sub path of any Shortest path is it self a shortest". The given shortest path input a->c,c->f,f->g satisfies the Metamorphic Relation -1 with all possible node combinations of given shortest path. The whole process is clearly shown in the below screen 5 with all possible values. The given input path is again evlauted against another Metamorphic relation ' The identified shortest path between two nodes must be equal or small than any other shortest paths between these these nodes', Test cases are generated for the evlauated the path as shown in figure 6. After this evaluation then the MTAF

considered the final Metamorphic Relation 'The shortest path and reverse shortest path should be equal', and evaluated the given input a->c,c->f,f->g against this metamorphi relation and test cases are generated. The Generated test cases are shown in figure-7 along with test results. The given shortest path a->c,c->f,f->g is satisfying all the 3 metamorphic relations and identified that there are no errors for given input path. The results of this testing is pass, because of no bugs identified with test case output values. All metamorphic relations are satisfied for this shortest path indicating no error.

Similarly to identify the errors in shortest path of above graph, we updated the given shrtest path from a->c,c->f,f->g to a->c,c->e,e->g. The Proposed MTAF tool automatically generated all possible test cases by mapping with the specified Metamorphic Relations and tested with the given input path for errors. After the evaluation the same shortest path with MTAF and identifed the bugs as shown below screen (figure 8).

The figure 8 is showing the identified bug information along with the proposed metamorphic relations. While calculating the shoretst path between c->g the result is c->f and f->g. The node f is not a part of given shortest path (a->c,c->e,e->g) indicating that the given shortest path is not a valid one. Like this our proposed MTAF is having the capability to identify the errors from shortest path. Researcher of this paper also implemented the metamorphic testing for Minimal Spanning Tree and the Dynamic Symbolic Execution also.

Experiments with MTAF web tool identified several hidden bugs from MPA, Graph Theory input programs without using any test oracle. Test cases generation, mapping and executing is implemented automatically with help of novel algorithms like NDI and STCG algorithms.

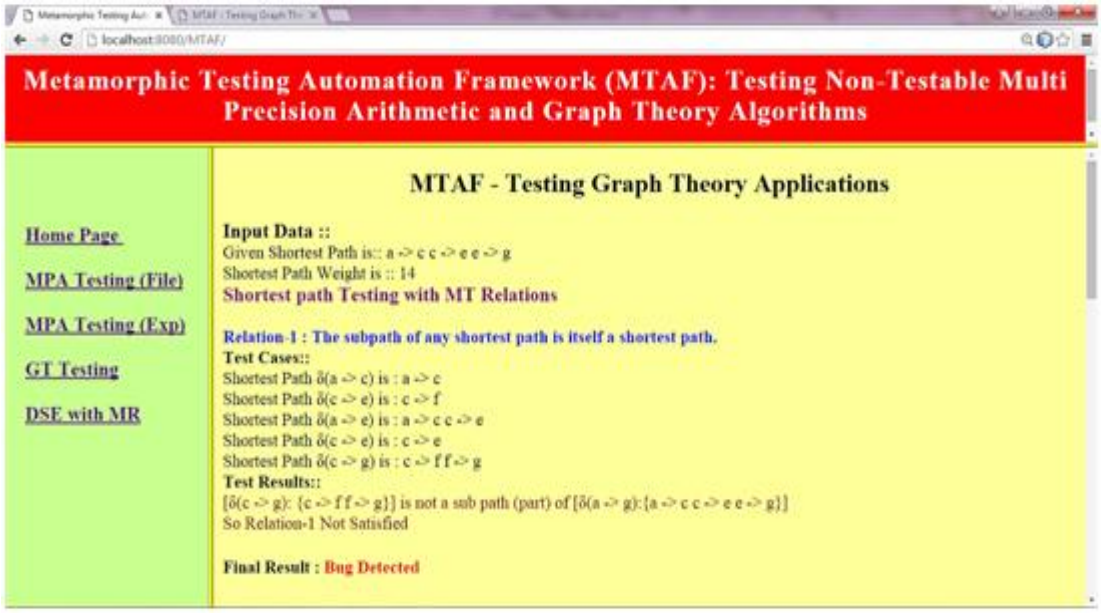

Figure 8 Test cases generated screen for the Shortest path Algorithm

4. CONCLUSION

In this paper, the researcher presented MTAF framework web tool conducted experiments with respective screenshots. Related work briefly explained the MTAF architecture with infrastructure set up. This web tool is implemented by using Java and its relevant technologies to test Java software applications. MTAF interface

evaluated MPA and GT domain errors information presented with proper example set. Discussion at each test example is representing the evaluation process and how the relations are used to find the bugs in a standard manner. Experiments are proven that MTAF is the comprehensive Metamorphic Testing automation tool to identify the hidden bugs from the given input programs.

REFERENCES

[1] T.Y. Chen, S.C. Cheung, and S.M. Yiu. Metamorphic testing: a new approach for generating next test cases. Technical Report HKUST-CS98-01, University of Science and Technology, Hong Kong, 1998.

[2] Ch Aruna, Dr.R.Siva Ram Prasad, Metamorphic relations to improve the test accuracy of Multi Precision Arithmetic software applications, Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on 24-27 Sept. 2014 by IEEE, Delhi, India. pages 2244 – 2248.

[3] Aruna, C.; Prasad, R.S.R., Adopting Metamorphic Relations to Verify Non-Testable Graph theory Algorithms, in Advances in Computing and Communication Engineering (ICACCE), 2015 Second International Conference on, pp.673-678,1-2May-2015.

[4] X. Y. Xie, J. W. K. Ho, C. Murphy, G. Kaiser, B. W. Xu, and T. Y. Chen. Application of metamorphic testing to supervised classifiers. In Proceedings of the 9Th International Conference on Quality Software (QSIC), pages 135–144, Jeju, Korea, 2009. CPS.

[5] Chittineni Aruna & R Siva Ram Prasad, MTAF: A Testing Framework for Metamorphic Testing Automation, ISBN: 978-1-941968-18-5 ©2015 SDIWC.

[6] Chittineni Aruna & R Siva Ram Prasad, NDI Al, an International Conference on 6.

[7] Chittineni Aruna & R Siva Ram Prasad, Successive Test Case Generation (STCG) Algorithm for Metamorphic Testing to generate Follow-up Test Cases, Proceedings of the International Conference on Software Engineering, Mobile Computing and Media Informatics (SEMCMI2015), Kuala Lumpur, Malaysia, 2015. ISBN: 978-1-941968-18-5 ©2015 SDIWC.

[8] Ganesh Prasad and Suresh Prasad Yadav, Spatial Positioning of Fracture In Hard Rock, *International Journal of Advanced Research in Engineering and Technology*, **6**(7), 2015, pp. 37-42.

[9] Mr. Sonar Sanjay Bhagwan and Dr. Samrat O. Khanna. A Uml Model For Automation of Counseling System Using Pure Object Oriented Approch. *International Journal of Computer Engineering and Technology*, **4**(5), 2014, pp. 15-22.

[10] http://www.oracle.com/technetwork/java/javaee/downloads/java-archive-downloads-eesdk-419427.html

[11] https://netbeans.org/downloads/