VOL. 10, NO. 1, JANUARY 2015 ISSN 1819-6608
ARPN Journal of Engineering and Applied Sciences


www.arpnjournals.com

AN EFFICIENT SOFTWARE DEFECT PREDICTION MODEL USING OPTIMIZED TABU SEARCH BRANCH AND BOUND PROCEDURE

Pandiyan G.[1] and P. Krishnakumari[2]
[1]RVS College of Arts and Science, Coimbatore, India
[2]Department of MCA, RVS College of Arts and Science, Coimbatore, India
E-Mail: ganeshanpandiyan@gmail.com

ABSTRACT

Software fault localization is considered to be one of the most tedious procedures that involves larger amount of time during the debugging of program. With this, there arises an increasing desire for software fault localization to be practiced with minimum amount of human intervention. This resulted with the design of several methods, each of which provides means to address the issues related to software fault localization to be more significant in its own notable and innovative manner. While the automatic structure based fault localization using genetic programming retains the program by avoiding a specific error, but failed to repair new types of bugs and programs. Most of the present software fault localization method overcomes the individual software failures and faults. However, an in-depth insight into the work reveals that, localization does not support several combinations of heuristics faults while performing software component testing. Finally, high automatic fault localization demand led to the proposal and development of software functionality on predicting the faults at an earlier stage with minimal prediction time. To overcome the defect on software fault localization, Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB) is proposed in this paper. TSFL-PBB divides the work into two phases. The first phase identifies (i.e.,) search doubtful software programming code which contain bugs (i.e.,) faults using Meta-heuristic Tabu search method. The mathematical operational based optimization checks with the immediate neighbors to handle different combinations of heuristics faults. The second phase of the TSFL-PBB software engineering model develops the Path Branch and Bound procedure. The branch and bound procedure in TSFL-PBB uses the travelling salesperson operation on localizing the faults at a faster prediction rate with higher readability and maintainability of software quality. Experiment is performed on factors such as fault prediction rate, processing time, repair cost.

Keywords: tabu search fault localization, software engineering, path branch, bound procedure, travelling salesperson, meta-heuristic faults.

1. INTRODUCTION

With the increase in the size of the software program, the idea of software fault localization or the ability to detect bugs present in it has received widespread attention in the field of software engineering. Many researchers have contributed in the field of software fault localization and several methods and mechanisms have been developed. Generic method for Automatic Software Repair GenProg [1] used an enhanced form of genetic programming models that in a way highly sustained with the required functionality for user defined fitness function. GenProg effectively detected the repair. The key to the method that it did not introduced new vulnerabilities. Though, substantial amount of bugs were repaired, but there was no room for new bugs.

Computer Software Configuration Items (CSCI) [2] not only analyzed the fault but also evaluated and measured different faults spread throughout the system. In addition, the method was proven to be advantage in terms of coding faults, requirement faults and finally the common fault types. However, an overall analysis of the work revealed that the faults measured in CSCI did not fully supported different combinations of heuristics faults during software component testing.

Different types of distributed systems can be adopted using Service Oriented Architecture (SOA). A fault localization framework called as the Business Process Execution Language (BPEL) [3] was developed to effectively measure and locate the faults. The framework proved to be more efficient in terms of largest number of seeded faults. However, a fully automatic system to detect the faults was not identified and addressed. To address this problem several fault prediction metrics were addressed in [4] based on three stages namely, planning, conducting and reporting stage. But, influential metrics remained unaddressed. To address issues related to influential metrics, in the purview of fault prediction, multi linear regression map and stepwise linear regression was used in [5] to minimize the reduced set of influential terms. Though, fault prediction rate was improved, but left room for non linear regression model.

Over the last few decades, significant amount of fault-localization models have been introduced on the basis of statistical analysis of software program constructs. In [6], the enhancement of fault localization algorithms like, Tarantula, Ochiai, and Jaccard fault were studied in existence with web applications with the aid of source mapping. The method proved to be effective with respect to maximal fault-localization effectiveness. However, measures were not taken to address client side java script code for test driven model.

In [7], Test Driven Development practice was introduced that in a way identified and located the bugs present in the software program at an earlier stage.

However, the accuracy with which the method was proven to be in relation with other statistical based algorithms remained unaddressed. To improve the level of accuracy, artificial neural network was introduced in [8] to locate and identify the defected codes. The method was proved to be not only accurate but also highly reliable in locating the faults at relatively lesser time period.

To overcome the shortcomings of the above methods, this paper provides an insight into the design of an efficient model called as the Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB) to overcome the defect on software fault localization in software program. The contributions of the proposed model include the following: To overcome the defect on software fault localization using Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB). To identify the bugs or faults present in software programming code using Meta-heuristic Tabu search method. To efficiently handle different combinations of heuristics faults using mathematical operational based optimization checks with the immediate neighbors. To localizes the faults at a faster predicted rate with higher readability and maintain the software quality using branch and bound procedure with the aid of travelling salesperson operation. The organization of the paper is as follows.

Section 1 provides the framework for software fault localization based on the software program construct. Section 2 includes a detailed comparison with other state-of-the-art methods. Section 3 details the Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB), Tabu based fault search and branch and bound procedure for fault localization with the help of a near architecture diagram and algorithmic description. Section 4 provides the experimental setups followed by Section 4 including the result analysis. Finally, Section 6 includes the concluding remarks.

2. RELATED WORKS

During the development of software projects, more like 50 percent of the project cost is spent during the software program testing and debugging. On the other hand, software fault localization hugely depends on two main factors namely, passed and failed runs. At the same time, the passed runs are highly prone to coincidental correctness whereas vast number of bug reports is produced during the failed runs.

F Only [9] developed an effective model that in a way highly relied upon the failed runs to identify and locate the software faults. However, the consumption of resources increased with the increase in the size of the software program. To address this issue, Spectrum based Fault Localization (SFL) [10] was introduced that efficiently identified the bugs and was proved to be more feasible and effective. However, multiple types of faults remained unaddressed. Ordinary Least Squared (OLS) based prediction model was introduced in [11] to identify and detect various types of faults using K-Nearest Neighbor regression. Though software fault prediction rate was improved, with the increasing size of software program constructs, the rate at which the software fault was predicted was relatively less.

Software defect prediction measures and locates the defective modules present in the software program construct. With the aid of software detect prediction quality of the software and the efficiency of testing can be improved in a significant manner. In [12], data mining models like clustering and classification were introduced for efficient detection and prediction of errors present in the software program construct. Though the methods were proved to be efficient in terms of detection rate, the means for improving the software quality was not addressed. To address the issues related to software reliability, statistical and machine learning methods were introduced in [13] to improve the software prediction rate using random forest and bagging. However, the product properties or the size of the software program construct was not taken into consideration.

In the recent years, there has been an increasing interest and need for certain amount of precise method for measuring the software size and quality assurance related to medical technology devices. A framework was designed in [14] to measure the software quality related to medical technology and the analysis of quality. However, the method was proved to be a specific model designed effectively for medical practitioners. A fault detection model was introduced in [15] for radar system that analyzed the faults using command parameters. However, transient faults remained unaddressed. In order to address the transient faults, Software Error Detection using Software Redundancy (SEDSR) [16] was designed that effectively served as an alternative for hardware based models. The method not only detected control flow but also identified the data errors at relatively lesser interval of time.

Based on the methods and models discussed above, we provide an insight into the tabu search software fault localization with path branch and bound procedure to localize the faults in software projects.

3. TABU SEARCH SOFTWARE FAULT LOCALIZATION WITH PATH BRANCH AND BOUND PROCEDURE

The goal of the proposed work is to localize the faults (i.e.,) bugs in software projects using Meta-heuristic Tabu Search optimization method. The objective behind the design of Meta-heuristic Tabu Search optimization method is to identify the bugs and localize the faulty statement at minimum time interval. TSFL-PBB software engineering model is based on localizing the faults on different software components. With this, the faults present in small and large size software components are identified. The Meta-heuristic Tabu based Fault Searching is illustrated in Figure-1.

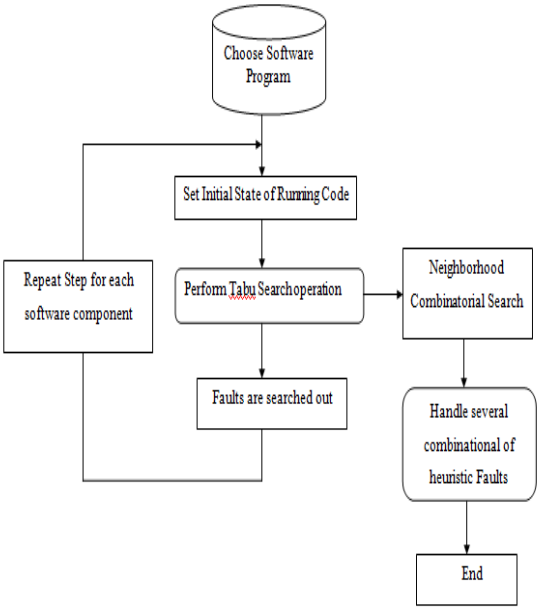

Figure-1. Meta-heuristic tabu fault search optimization.

As illustrated in Figure-1, TSFL-PBB software engineering model initially chooses the software programs. The chosen software program runs the code to identify the bug report and the bugs are identified with the aid of Tabu search method. Tabu based software fault searching method uses the neighborhood combinatorial search optimization procedure that efficiently handles several combination of heuristic faults on software projects. With this, the fault localization solution is determined by analyzing through the neighborhood software statements. The neighborhood software statements are iteratively checked out and finally improve the localization procedure at earlier stage of software project testing.

The Tabu list in TSFL-PBB software engineering model contains the set of rules that efficiently filter out the faulted code statement from the software program. In the simplest form, Tabu list in TSFL-PBB software engineering model also includes the recent past visited program code information for easy identification of faulty statement at an earlier stage. The second phase of the work is to develop the Path Branch and Bound procedure. The Path Branch and Bound procedure of TSFL-PBB model uses the breadth first search based travelling salesperson operation.

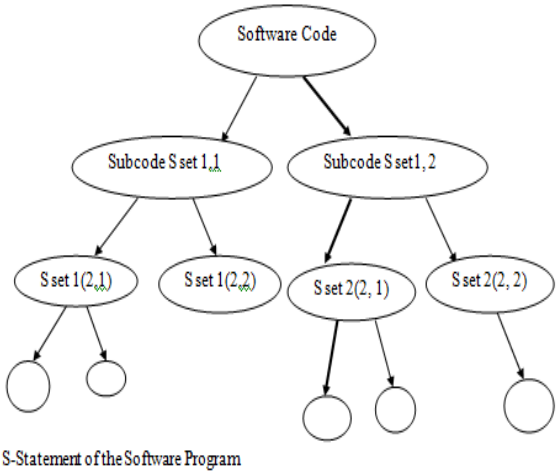

Figure-2. Branch and bound using travelling salesperson operation.

As illustrated in Figure-2, the branch and bound using travelling salesman operation consists of a root software code, that are further divided into sub code. Followed by this, each sub code is further divided into sub set and so on. The Branch and bound method not only handle the single fault but also several combinational meta-heuristic fault problems with faster prediction rate.

The branch in branch and bound method perform splitting procedure whereas the bounding operation evaluates both the upper and lower bound with minimum value range. The travelling salesperson operation with linear program bounds the faulted program code for faster prediction rate. The architecture diagram of Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB) model is depicted in Figure-3.

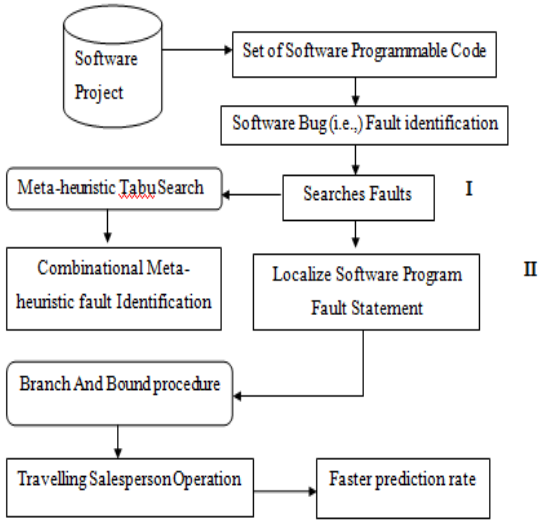

Figure-3. Flow diagram of TSFL-PB software engineering model.

Fault Localization of software projects is clearly carried out in two phases of the proposed work. The initial phase of the work is to identify the faulty software program statement on several meta-heuristic combinations. The Tabu search carries the Tabu list in TSFL-PBB model, where the previously visited software component statement is used for easier prediction of faulty statement. The bugs in the software projects are clearly described using the branch and bound procedure. The branch and bound procedure with TSFL-PBB model uses the tree structure to identify the faults at faster rate. The larger the amount of software fault prediction improves the software repair rate. The forthcoming subsections discusses in detail about the two phases, the meta-heuristic tabu based fault search and branch and bound procedure for fault localization.

3.1. Meta-heuristic tabu based fault search

The first phase measures and searches the faults in the software program using meta-heuristic tabu based fault search. The Tabu based Fault search generates tests on the input software programs to identify the faults. Let us assume ' ' to be the vector for the given software program statement, then the set of values for program statement is measured in order to search the fault rate.

Tabu search performs the search by iterating with the sub codes of the software program statement. Tabu based fault search also uses the branch and bound procedure to exactly localize the fault rate. As a result, branch and bound process, regenerate the search to several larger software components and finally reaches the feasible fault prediction solution. The Tabu based fault searching procedure is described as,

//Tabu based Software Fault Search
Begin
Step-1: Initialize set of software programmable code
Step-2: Assume Tabu list T = (1, 2, 3,….,n), where T is the previous visited program statement
Step-3: Repeat for each software Component
Step-4: Find the Neighborhood Combinational Search in distant software component zones
Step-5: If =0

Step-6: Then no faults on the software program statement
Step-7: Else
Step-8: ← 1
Step-9: End If
Step-10: While (Neighborhood Combinations 'P') = High Quality
Step-11: P= Arbitrary neighborhood with Better Quality of software
Step-12: End While
End

The Tabu based fault search depends on the neighborhood program statements to produce optimized fault localization solution in TSFL-PBB software engineering model. Tabu based fault search uses the if-else procedure to identify whether the code is faulty or not. Secondly, the while procedure used on satisfying the neighborhood combination condition (i.e., Step 10) attain better software repair rate. The neighborhood combinational condition is expressed as,

$$If (||S\text{-}S|| \leq \Theta\ f(S)\ then\ feasible\ region \quad (1)$$

The feasible region is measured through (1), where the software program statement 'S' is compared with the predefined statement ' '. The faulty condition produces the result as,

$$If\ f(S^*) \leq f(S)\ then\ faulty\ region \quad (2)$$

Mathematical operational as described above is used in TSFL-PBB software engineering model to attain the best optimized software maintenance by predicting the bugs on software projects or programs. Once the faults are efficiently searched, the fault localization is performed which is discussed in detail in the forthcoming subsections.

3.2. Branch and bound procedure for fault localization

Upon successful fault searching in software program, fault localization is performed using branch and bound procedure. Branch and bound in the proposed work uses the breadth first search with pruning concept to easily predict the faults on software components and uses the Tabu list data structure information. The child nodes or the software sub code is analyzed to localize the fault paths on the software projects.

Branch and Bound procedure divides the problem into sub problem and localize the exact fault occurrence on the software component. The branching is based on different programmable software component with linear program code for faster prediction rate. The branch as described in (2) is called lower branch of inequality. The higher branch of inequality is

$$f(S^*) \geq f \quad (3)$$

The higher branch of splitting improves the tree count by '1'. The lower branch tracks the faulted software component and localizes the accurate faulty condition. The lower bound reaches the upper branch in TSFL-PBB software engineering model with minimal processing time. The minimal time on processing reaches the faster fault prediction.

3.2.1 Bounding operation

Travelling salesperson operation is applied in TSFL-PBB software engineering model to perform prioritized based Tabu list data structure information. The prioritization increases the localization at faster rate. The initial bound of the branch and bound tree sums up the

outgoing edge programmable sub code to reduce the processing time.

Bounding Operation = Subcode S(set1, 2+ S set 2(2, 1) +...) (4)

The bounding operation in TSFL-PB software engineering model is performed in (4) using the information in Figure-2. The fault code statement on the software from root to the leaf of the tree structure is analyzed and identifies the faulty software components. The defects in the software projects are identified effectively through the TSFL-PBB software engineering model.

4. EXPERIMENTAL EVALUATION

To localize the faults, Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB) is proposed and performs the experimental evaluation using JAVA platform. Software Engineering model based fault localization takes the software programs of size 50 MB which includes two datasets, Arcene Data Set and Lung cancer dataset from UCI repository. Arcene data and Lung cancer data interrelated and then the software fault are localized while running the program code.

Arcene contains the 7000 real variables, with 3000 probes which totally of 10000 variables. Lung cancer dataset demonstrate the power of the most favorable discriminate plane with 3 kinds of pathological lung cancer. The information on the individual variables clearly describes about the attribute information, taking on integer values 0-3. The program code with bugs is identified at initial stage using TSFL-PB software engineering model. Proposed TSFL-PB model is compared against existing Generic method for Automatic Software Repair (GenProg) [1] and Computer Software Configuration Items (CSCI) [2] to predict faults. The experiment is conducted on the factors such as fault prediction rate, software repair rate, processing time, repair cost time and so on.

The fault prediction rate is obtained by comparing the software statement and with the predefined statement ‘ ’

$$\left.\begin{array}{l} if\ (||S - S*|| \le \theta) then\ feasible\ region \\ \quad FPR = 0 \\ else\ if\ f(S*) \le f(S) then\ faulty\ region \\ \quad FPR = FPR + 1 \\ \quad endif \\ endif \end{array}\right\} \quad (5)$$

Upon satisfaction of the first if statement, the fault prediction rate is set to zero whereas by meeting the second condition, the fault prediction rate gets incremented by one. The fault prediction rate is measured in terms of percentage (%).

The software repair rate of a software program is defined as the probability that the software program or a module experiences the first repair or has repair has occurred one or more time during the time interval . The software repair rate is measured in terms of MB.

$$SRR = S_{repair} \quad (6)$$

Repair cost time in TSFL-PBB software engineering model refers to the time taken to perform the repair operation using breadth first search. The repair cost time is measured in terms of milliseconds. The Processing time involved in the processing of the entire software fault localization using TSFL-PBB software engineering model refers to the time complexity involved during the branch and bound process given as

$$Processing_{time} = Time\ Complexity(Bounding\ Operation) \quad (7)$$

5. RESULTS ANALYSIS OF TSFL-PBB

The result analysis of Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB) model using software programs with Arcene and Lung cancer dataset extracted from UCI repository is compared with existing Generic method for Automatic Software Repair (GenProg) [1] and Computer Software Configuration Items (CSCI) [2] to predict faults. The fault prediction rate obtained using JAVA and comparison is made with two other methods, namely GenProg [1] and CSCI [2].

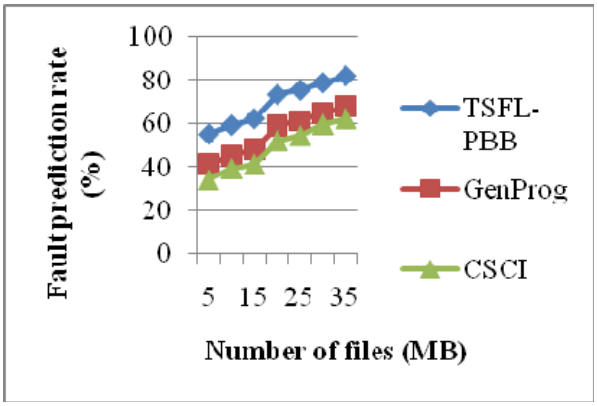

Figure-4. Comparison diagram showing the fault prediction rate over TSFL-PBB, GenProg and CSCI.

Figure-4 shows that the proposed TSFL-PBB model provides higher fault prediction rate when compared to GenProg [1] and CSCI [2]. This is because of the application of Meta-heuristic Tabu search method based on the neighborhood software statements that are iteratively checked that finally improves the rate of fault prediction at earlier stage and therefore improving it by 16

- 25 % compared to GenProg. Moreover, the Tabu based software fault searching utilizes the neighborhood combinatorial search optimization for efficient handling of several combination of heuristic faults on software projects by improving fault prediction rate by 23 - 38 % compared to CSCI.

The comparison of software repair rate is presented with respect to different file sizes of range 5 - 35 MB. With increase in the number of images, the software repair rate also gets increased, but at certain point the repair rate is not observed to be linear. It is because of the errors present in the software program differs and is not linear.

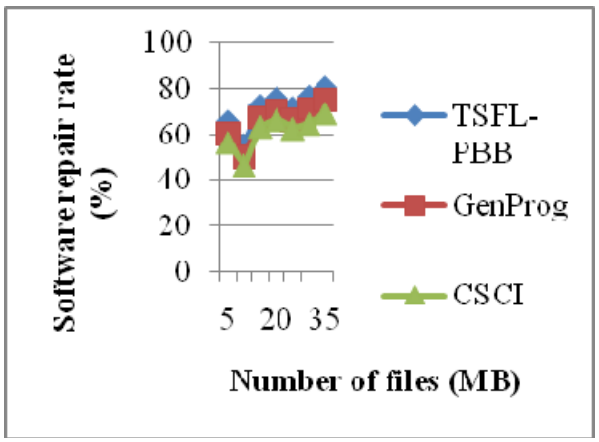

Figure-5. Line chart showing the software repair rate with respect to different file sizes.

To measure and evaluate the performance of the software repair rate, comparison is made with two other existing works Generic method for Automatic Software Repair (GenProg) [1] and Computer Software Configuration Items (CSCI) [2] to predict faults. In Figure-5, the file size is varied between 5 and 35 MB. From the Figure it is illustrative that the software repair rate is higher or increased using the proposed TSFL-PBB model when compared to the two other existing works. This is because with the application of Path branch and bound procedure that uses the breadth first search based travelling salesperson operation to handle both the single fault and combinational meta-heuristic fault problems improving the software repair rate by 6 - 9 % compared to GenProg. Furthermore, using prioritized based Tabu list data structure information, prioritization increases the localization at faster rate increasing the software repair rate by 12 - 16 % than when compared to CSCI.

The repair cost time efficiency for TSFL-PBB model is elaborated in Table-3. We consider the method with software components of size 2 to 14 for experimental purpose using JAVA platform.

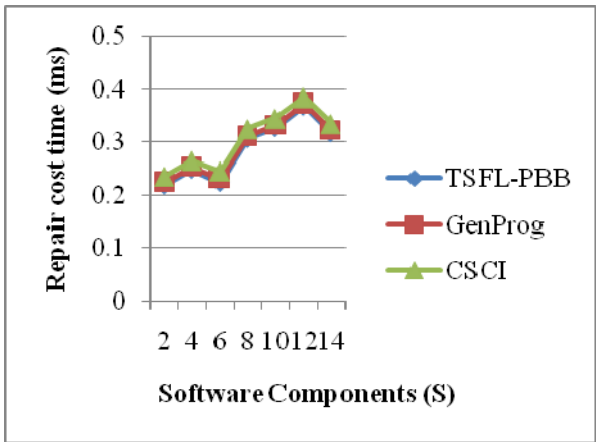

Figure-6. Line chart representing the rate of repair cost time with respect to software components.

In Figure-6, we depict the repair cost time attained using different software components of size 2 to 14 for experimental purposes. From the Figure, the value of repair cost time achieved using the proposed TSFL-PBB model is lower when compared to two other existing works Generic method for Automatic Software Repair (GenProg) [1] and Computer Software Configuration Items (CSCI) [2] to predict faults. Besides we can also observe that by increasing the size of the software components, the time taken to perform the repair cost gets decreases using all the methods. But comparatively, it is lower using the TSFL-PBB model because with the application of branch and bound procedure by improving the time taken for repair cost by 1 - 3 % compared to GenProg. With the aid of breadth first search the branch and bound effectively prunes and easily predict the faults on software components, thus minimizing the software repair cost and therefore the repair cost time by 3 - 9 % compared to CSCI.

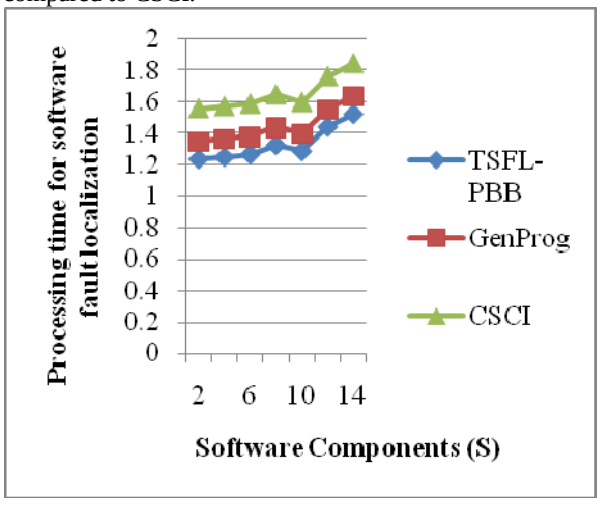

Figure-7. Line chart representing the processing time for software fault localization with respect to software components.

Figure-7 illustrate the processing time involved during software feature localization versus the number of software components in the range of 2 50 14 for experimental purpose conducted using JAVA. From the Figure we can note that the processing time attains 8.45 % improved for the software component of size 8 when compared to GenProg [1] and 24.45 % improved when compared to CSCI [2] which shows that there is a significant improvement using the proposed TSFL-PBB model. This is because the Tabu list in TSFL-PBB software engineering model includes the set of rules to filter out the faulted code statement from the software program. With this, the Tabu list in TSFL-PBB software engineering model includes recent past visited program code information for easy identification of faulty statement at an earlier stage improving the processing time by 7 - 9 % compared to GenProg. In addition, with the application of travelling salesman operation, the initial bound of the branch and bound tree sums up with the outgoing edge programmable sub code to reduce the processing time by 21 - 26 % when compared to CSCI.

6. CONCLUSIONS

Software fault detection and localization has become the key for debugging software programs to increase the fault prediction rate and improve the amount of software repair cost at relatively less amount of time.

In this work, the performance measure and effects of software fault detection and localization model is presented and analyzed to identify and overcome the individual software failures and faults. For this, a model called Tabu Search Fault Localization with Path Branch and Bound procedure on Software Engineering (TSFL-PBB) is structured based on the software program. This improves the prediction rate and identifies the errors present in the software program and greatly localizes the faults (i.e.,) bugs in software projects. First, we study the use of Tabu based fault search to identify the faults with the aid of branch and bound process and localize the fault rate in an exact manner. Second, we design branch and bound procedure for different programmable software component with the aid of breadth first search to improve the prediction rate. The branch and bound tree is finally integrated with the outgoing edge programmable sub code to reduce the processing time in an efficient manner using the software program of size 50 MB which includes two datasets, Arcene Data Set and Lung cancer dataset from UCI repository. The experiment conducted using the data set shows that the DIARETDB1shows that the TSFL-PBB achieves up to 16 percent improvement on software repair rate compared to the state-of-the-art methods.

REFERENCES

[1] Claire Le Goues, ThanhVu Nguyen, Stephanie Forres and Westley Weimer. 2012. GenProg: A Generic Method for Automatic Software Repair. IEEE Transactions on Software Engineering. 38(1).

[2] Maggie Hamill and Katerina Goseva-Popstojanova. 2009. Common Trends in Software Fault and Failure Data. IEEE Transactions On Software Engineering. 35(4).

[3] Chang-ai Sun, Yi Meng Zhai, Yan Shang and Zhenyu Zhang. 2013. BPELDebugger: An effective BPEL-specific fault localization framework. Information and Software Technology, Elsevier.

[4] Danijel Radjenovic, Marjan Hericko, Richard Torkar and Ales Z ivkovic. 2013. Software fault prediction metrics: A systematic literature review. Information and Software Technology.

[5] Rinkaj Goyal1, Pravin Chandra and Yogesh Singh. 2013. Identifying influential metrics in the combined metrics approach of fault prediction. Springer plus.

[6] Shay Artzi, Julian Dolby, Frank Tip and Marco Pistoia. 2012. Fault Localization for Dynamic Web Applications. IEEE Transactions On Software Engineering. 38(2).

[7] Massimo Ficco, Roberto Pietrantuono and Stefano Russo. 2011. Bug Localization in Test-Driven Development. Hindawi Publishing Corporation Advances in Software Engineering Volume 2011.

[8] Xiuhua Yuan, YiqiangWang and Yan Gu. 2013. Software Fault Location of CNC System Based on Similar Path Set and Artificial Neural Network. Hindawi Publishing Corporation Advances in Mechanical Engineering Volume 2013.

[9] Zhenyu Zhang, W.K. Chan and T.H. Tse. 2012. Fault Localization Based Only on Failed Runs. IEEE Computer Society. 5(6).

[10] Xiaoyuan Xie, W. Eric Wong, Tsong Yueh Chen and Baowen Xu. 2012. Metamorphic slice: An application in spectrum-based fault localization. Information and Software Technology, Elsevier.

[11] Rinkaj Goyala, Pravin Chandraa and Yogesh Singha. 2014. Suitability of KNN Regression in the Development of Interaction Based Software Fault Prediction Models. 2013 International Conference on Future Software Engineering and Multimedia Engineering, Elsevier.

[12] Ms. Puneet Jai Kaur and Ms. Pallavi. 2013. Data Mining Models for Software Defect Prediction. International Journal of Software and Web Sciences (IJSWS).

[13] Ruchika Malhotra and Ankita Jain. 2012. Fault Prediction Using Statistical and Machine Learning

Methods for Improving Software Quality. Journal of Information Processing Systems. 8(2).

[14] Nizam Uddin Ahameda, Kenneth Sundaraja, R. Badlishah Ahmadb, Matiur Rahmanc and Asraf Alib. 2012. A framework for the development of measurement and quality assurance in software-based medical rehabilitation systems. International Symposium on Robotics and Intelligent Sensors, Elsevier.

[15] Zhu Jie-zhong, Yao Yong-lei and Chen Su-ting. 2012. Proposal for the Software Development of a Radar Power Fault Detection System. 2012 International Conference on Medical Physics and Biomedical Engineering, Elsevier.

[16] Seyyed Amir Asghari, Atena Abdi, Hassan Taheri, Hossein Pedram and Saadat Pourmozaffari. 2012. SEDSR: Soft Error Detection Using Software Redundancy. Journal of Software Engineering and Applications.