Bi Objective Optimization for Bug Triage

V. Akila, Dr. G. Zayaraz, R. Anitha

Dept. of Computer Science and Engineering, Pondicherry Engineering College

Abstract- Open Source Software development is self-organizing and dynamic in nature. Software maintenance is subject to error, cost and time overruns. Issue tracking is the major component of software maintenance. The bug triager, is an individual who chooses what to do with an approaching bug report and to whom to prescribe the bug. The number of bug reports submitted day by day increases. The physical prescribing work becomes troublesome and fault prone. The bug prescribed to a developer may get tossed to another developer. The Bug Tossing Graph codifies the tossing relation that exists among the developers. Automated support is needed to find the optimal set of developers who can address the bug. The existing techniques that work on Bug Toss Graph relies on the Weight based Breadth First Search (WBFS) Algorithm. This technique tries to reduce the number of hops between the first assignee of the bug to the final developer, but ignores the transition probability information present in the edges. To this end, this paper presents a technique based on Bi-objective Optimization that maximizes transition probability and minimizes the number of hops. In order to validate the proposed technique experiments were conducted using the bug dataset of Eclipse project. The various parameters including (i) Precision, (ii) Recall and (ii) Mean Steps to Resolve (MSTR) are considered. The results indicate that the proposed technique is better than the existing techniques.

Keywords: Open Source Software; Bug Tossing Graph, Bug Triage, and Optimization.

I. Introduction

Software engineering is even today based on instinct and encounter. Software archives hold an abundance of data about software ventures. Source control stores, bug archives, filed correspondence, sending logs, code archives are examples of software archives that are normally accessible for most software ventures [1]. If knowledge inferred from the software archives can be put in use in software engineering then it can lead to considerable reduction in the cost incurred and time spent. Bug archives track the history of bug reports or characteristic demands that are accounted by clients and developers. The designers and clients submit bug reports to a bug storehouse (e.g. Bugzilla) in Open Source programming improvement. Bugzilla and Jira are examples of bug archives. Bug archives give a dataset of issue reports for a software enterprise. Individuals assume diverse roles as they collaborate through reports. The individual who submits the report is the reporter of the bug [2]. The triager is the individual who judges if the report is serious and who assigns developers to bug reports. A contributor might additionally comment on the solution to the bug. Finally, a developer may resolve the bug and perform a commit operation. Bug Triaging involves classifying the bug, checking for validity, assigning severity level and most importantly, assigning the bug to the appropriate developer [2].

The activity information in the bug archive gives information on the activities that happened during the lifetime of a bug. It includes the reporter information, assignee information, tossing information and committing information. This information for a set of bug reports is captured in a Bug Toss Graph which is based on Markov Model. The nodes are the developers and the edges are the tossing relations that exist among them. The weights on the edges are the transition probability that reflects the frequency of tosses among the developers.

II. Related Work

Cubranic and Murphy et al. [3] were the first to propose the idea of utilizing content grouping strategies to semi-computerize the methodology of bug assignment. They separated words from the title and summary

 ISBN : 978-81-929742-0-0

of the bug report. A Naïve Bayes classifier was employed. The classifier recommends one or more potential developers for settling the bug. Anvik et al. [2] improved the machine learning approach by using some filter techniques. Bug reports labeled invalid were not considered while collecting the data. The data was trained using supervised machine learning algorithms.

Xuan et al. 4] considered the priorities of developers in bug repositories. Developer prioritization provides the knowledge of developer priorities to assist software task especially the task of bug triage. Developer prioritization was made using the socio-technical approach particularly using the comments made by the developers.

Syed et al. [5] proposed a technique which is based on Term to Document Matrix (TDM). Feature selection can lead to reduction in dimensionality of the TDM. Latent semantic indexing method works well for the dimensionality reduction. Jeong et al. [6] were the first to propose the idea of using Bug Tossing Graph for developer recommendations. The Tossing Graph was constructed based on the bug tossing history Tossing Graph which is based on the Markov Model. Then Weighted Breadth First Search (WBFS) algorithm was employed to detect the bug resolver from the Tossing Graph. WBFS utilizes the graphs and maximizes the tossing path reduction.

Liguo Chen et al. [7] presented an approach based on bug tossing history and by using textual similarities between bug reports. They identified duplicate bugs by calculating textual similarities with vector space model. This approach consists of three steps: First the Tossing Graph is calculated and then, the textual similarities are calculated and finally, the sub-graph is obtained using bug similarities. In order to reduce the search failure rate, the vector space model is applied to calculate the textual similarities.Pamela Battacharrya et al. [8] has proposed the idea of machine learning and Bug Tossing Graph to recommend potential developers for bug fixing. WBFS algorithm was employed to find the shortest path. They used different classifiers for machine learning method and they found that Naïve Bayes was the best among the different classifiers. Attributes corresponding to the product component information for a bug are added and the Tossing Graph is modeled as a multi-feature Tossing Graph because tossing probability alone is insufficient.

From the survey, it is clear that the machine learning methods coupled with Tossing Graph i.e. hybrid techniques give the best results. In machine learning method, Naïve Bayes classifier has been widely applied. In Open Source Software Systems, WBFS algorithm is used is used predominantly.

III. Research Contribution

The flow of the automated bug triaging system based on Bi-objective Optimization is illustrated in Figure .1. The bug reports are extracted from the bug repository. The Bug Toss Graph is constructed based on the activity data of bug reports. The transition probabilities are calculated. In order to reduce the tossing path length and to maximize the transition probability, the bi-objective shortest path algorithm is employed.

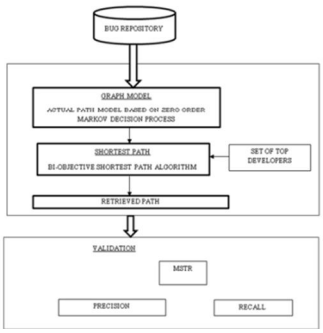

Figure 1. Flow diagram of Bi-Objective Optimization for Bug Triage

The procedure for Bi-objective Optimization is given in Figure.2. The adjacency matrix of the Bug Toss Graph is given as input to the Bi-objective Optimization as TPM[w]. The Non-Empty Paths (NEP) and Optimized Path are initialized to ϕ. The Maximum Weighted Matrix is determined. For all path in NEP(i), NEP(i) is checked whether it belongs to the maximum weighted matrix or not. If the above condition is satisfied, then the NEP(i) is added to the optimized path otherwise the NEP(i) is added to the total path[9].

```
Procedure Bi-Objective(TPM[w])
Begin
//Initialize Non Empty Paths NEP=ϕ, Optimized path OP(i)=ϕ
Step 1:  Get the total path P(i)
Step 2:  Determine M= Max(TPM [w])
Step 3:  Determine non empty paths NEP(i) ie P(i)≠ϕ
Step 4:  if NEP(i)∈ M
Step 5:          OP(i)=OP(i)+NEP(i)
Step 6:  Else
Step 7:          P(i)=P(i)+NEP(i)
Step 8:  return OP(i)
End
```

Figure 2. Procedure for Bi-Objective Optimization

IV. Implementation and Results

To perform the experiment, the bug reports from the available repositories of Eclipse project are extracted. These bug reports have been fixed between 2010-01-01 and 2012-12-31.These bug reports are saved in Comma Separated Value (CSV) format. The bug reports which have been fixed or resolved with a vote count of five were considered.

The bug reports are downloaded and split into training set and test set. The developer's activity data was processed and the names of those developers who changed the status of bug reports to "RESOLVED" are extracted. Based on the activity data of bug reports, the Bug Tossing Graph was constructed as an actual path model. The transition probabilities are calculated in order to reduce the tossing path length. The Bi-objective shortest path algorithm that maximizes the transition probability as well as reduces the number of hops is used to predict the optimal path between the assignee and resolver. The predicted path is cross validated against the test data using the parameters: (i) precision, (ii) recall and (iii) Mean Steps To Resolve (MSTR). The experiment was carried by varying the test data set by 10% , 20% and 30% for the total bug reports.

From the Figure.3a, Figure.3b and Figure.3c it is clear that the Bug Triage based on Bi-Objective optimization performs consistently better than WBFS algorithm in terms of MSTR.

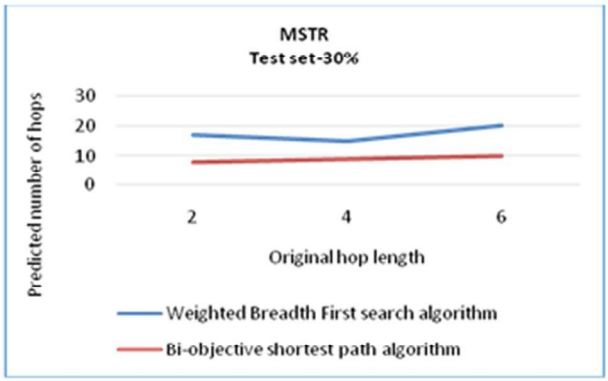

Figure.3a. MSTR for 30% Test Data

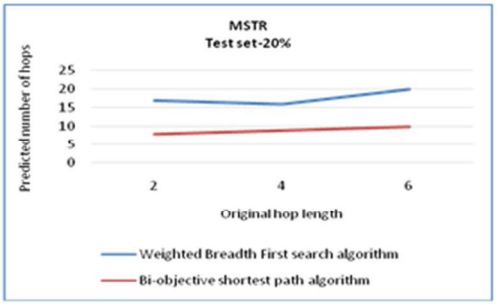

Figure.3b. MSTR for 20% Test Data

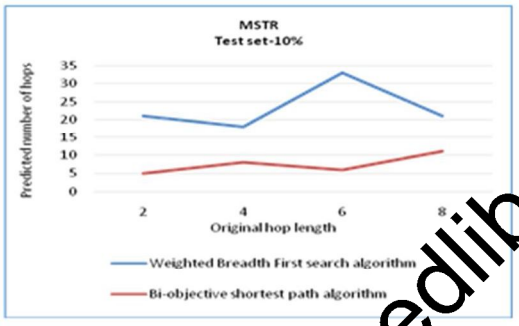

Figure.3c. MSTR for 10% Test Data

From the Figure.4a, Figure.4b and Figure.4c it is clear that the Bug Triage based on Bi-Objective optimization performs consistently better than WBFS algorithm in terms of Precision parameter.

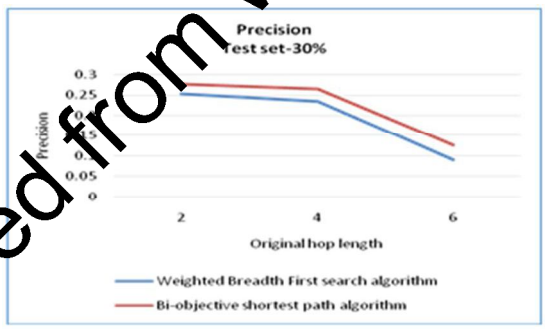

Figure.4a. Precision for 30% Test Data

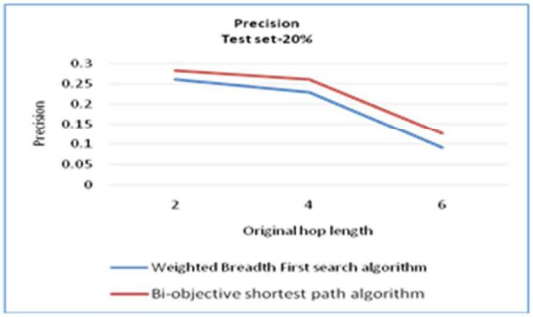

Figure.4b. Precision for 20% Test Data

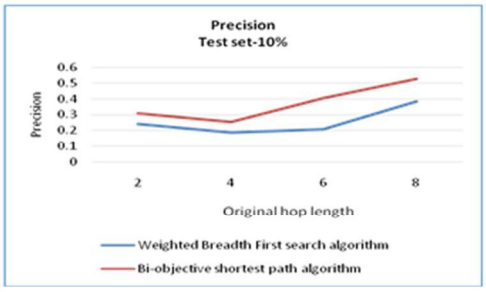

Figure.4c. Precision for 10% Test Data

From the Figure.5a, Figure.5b and Figure.5c it is clear that the Bug Triage based on WBFS algorithm performs consistently better than Bi-Objective optimization in terms of Recall parameter.

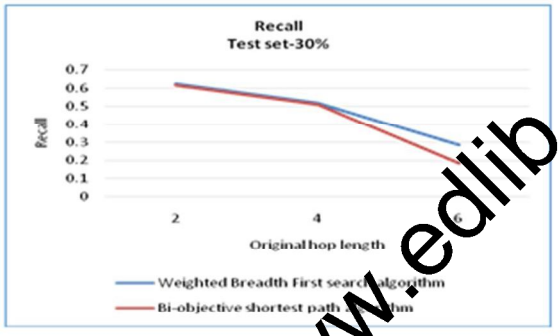

Figure.5a. Recall for 30% Test Data

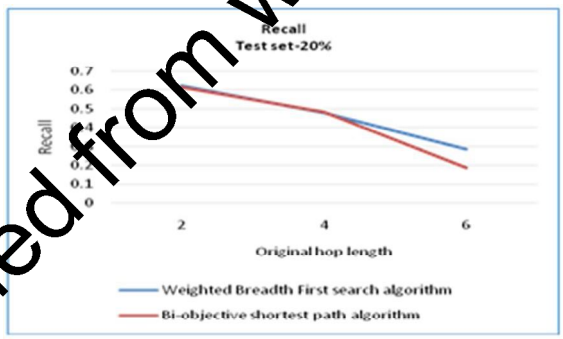

Figure.5b. Recall for 20% Test Data

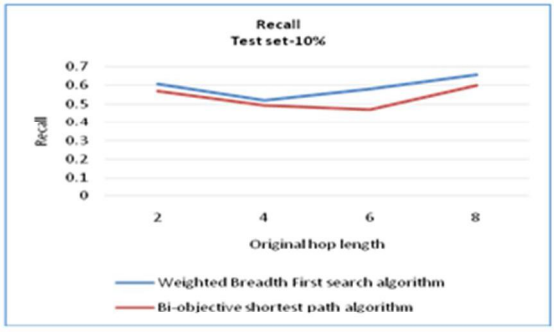

Figure.5c. Recall for 10% Test Data

V. Conclusion

Bug Triaging is a very important step in Software Maintenance. Herein, the bugs get assigned to a developer for resolving and if that developer is unable to resolve that bug then it is given to another developer. This information is captured in Tossing Graphs. The existing techniques for bug triaging focus on weighted breadth first search methods for reducing the number of hops. In this paper, it is proposed to employ Bi-objective shortest path algorithms over the tossing graph to find the optimal developers. The Bi-objective Optimization method was evaluated on the bug data set of eclipse project. The results indicate that Bi-objective Optimisation gives better results than the WBFS algorithm. The future natural progression of this system is to explore solutions from the field of search based software engineering for bug triaging.

References

1. Ahmed E. Hassan, "The Road Ahead for Mining Software Repositories", In Proceedings of the Frontiers of Software Maintenance, pp.48-57,2008.
2. J. Anvik, L. Hiew, and G. C. Murphy, "Who should fix this bug?" In Proceedings of International Conference on Software Engineering (ICSE), pp. 361–370, 2006.
3. D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," In Proceedings of the Sixteenth International Conference on Software Engineering & Knowledge Engineering (SEKE'2004), 2004.
4. Jifeng Xuan, He Jiang, Zhilei Ren, Weiqin Zou,"Developer Prioritization in Bug Repositories" In Proceedings of International Conference on Software Engineering(ICSE), pp.25-35, 2012.
5. Syed Nadeem Ahsan, Javed Ferzund and Franz Wotawa, "Automatic Software Bug Triage System (BTS) Based on Latent Semantic Indexing and Support Vector Machine", In Proceedings of Fourth International Conference on Software Engineering Advances, pp.216-221,2009.
6. Gaeul Jeong, Sunghun Kim and Thomas Zimmermann, "Improving Bug Triage with Bug Tossing Graphs", In Joint 12th European Software Engineering Conference (ESEC) and 17th ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp.111–120, Aug 2009.
7. Liguo Chen, Xiaobo Wang and Chao Liu, "Improving Bug Assignment with Bug Tossing Graphs and Bug Similarities", International Conference onBiomedical Engineering and Computer Science (ICBECS), pp.421-425, Apr 2010.
8. Pamela Bhattacharya, Iulian Neamtiu and Christian R. Shelton, "Automated, highly accurate, bug assignment using machine learning and tossing graphs", Journal of Systems and Software, vol.85, pp: 2275-2292, May 2012.
9. Andrea Raith and Matthias Ehrgott, " A Comparison of Solution Strategies for Bi-objective Shortest Path Problems", Journal of Computers & Operations Research, Vol.36, Issue 4, pp.1299–1331, April 2009.