I J C T A, 9(2) 2016, pp. 859-863

Bug Triage Based on Ant System with Evaporation Factor Tuning

V. Akila[1], V. Govindasamy[2] and S.Sharmila[3]

ABSTRACT

Bug Triaging is an important aspect of Bug Management in Open Source Systems. Bug Triaging pertains to assignment of a newly opened bug to an appropriate developer for resolution. This task is complicated in an open source environment because of the nature of the Open Source Development. Open Source Software Development comprises of the developers who are volunteers. The volunteers are distributed at different geographical locations. Further, the developers may become inactive after some time and their expertise may change. This necessitates the need for a robust automatic Bug Triage System . This paper presents an Automatic Bug Triage system based on Ant System with Evaporation Factor Tuning. Evaporation of the pheromone is based on the power law. This assists in optimizing the ant system for bug triage. The proposed system was evaluated using the parameters -Prediction Accuracy and Path Similarity.

Keywords: Bug Triaging; Evaporation Factor; Power Law

I. INTRODUCTION

Open Source Software is an example of open source development and often compared to user-generated content or open-content movements. While Open Source Software (OSS) is becoming more widespread and popular, its maintenance has become an important issue. The OSS developers and users are distributed geographically and therefore management of the defect reports and source code is an important factor for the success of the project. Open Source Software Development has opened up rich repositories pertaining to software development process like versioning repository, bug repository and email repository which were previously unavailable to the public. In particular, OSSs employ the two kinds of management systems: Defect Management System (DMS) and a Version Management System (VMS). The purpose of a DMS is to establish process for managing defect reports. DMS has mechanism for reporting and assigning the bug reports and modification requests. Furthermore, DMSs provide flexible possibilities for tracking and controlling and assigning bugs to developers to resolve them.

A software bug is an error, flaw, failure, or fault in a computer program or system that produces an incorrect or unexpected result. Software maintenance contributes to 50% of the software development cost. Issue management is a part of software maintenance [1][3]. Bug triaging was an important aspect of issue management.

In the case of popular Open Source Software Systems, the DMS receives an increasing number of reports every day. Therefore, the task of triaging the incoming reports therefore consumes an increasing amount of time[9]. Bug fixing is crucial in the final quality of the software products. When bug(s) are filed in a bug report, assigning it to the most capable and competent developer is important in reducing the cost and time in a bug fixing process. This assignment process is referred to as bug triaging [4], [6]. The application

[1] Assistant Professor, Dept. of C.S.E, Pondicherry Engineering College, India
[2] Assistant Professor, Dept. of I.T, Pondicherry Engineering College, India
[3] Dept. of C.S.E, Pondicherry Engineering College, India

which receives hundreds of bug reports a day; ideally, each bug gets assigned to a developer who can fix it in the least amount of time[8]. This process of assigning bugs is known as bug assignment[10].

II. RELATED WORK

The Bug Management process allows the project team to collect evaluate, correct, and track reported bugs and issues[1][2][5][7]. When a bug is first reported, it is registered as unconfirmed and managed by Test Manager. Test Manager is responsible to periodically go through any new bugs that may have been reported and for accepting or rejecting new bug. The test manager is also in charge of assigning accepted bugs to developers and set priority and severity of the bug based. When a bug is fixed by a member of the development team, it is considered as resolved. The testing team uses the information provided by reporter to verify that a bug has been fixed in a later build of the product. After successful verification, the bug is closed; however when there are still issues unresolved, the bug might be reopened for further development.

Once a bug report has been assigned to the developers, the developers can reassign the bug to other developers. This process is called as bug tossing. Representing those relationships among the developers in the form of graph is called as bug toss graph [2][7]. The current bug triage system rely on weight breadth first search algorithm (WBFS)[1]. WBFS captures the bug tossing relation in a static graph. WBFS is a rigid which does not suit the nature of open source software developer network. So, an Ant system is used to discern the right developer for an open bug. The ant system is optimized to mimic human forgetting behavior by the use of evaporation factor tuning. The main objective of the paper is to automate the bug assignment process in a optimized way. It focuses on reducing software evolution cost and effort. The evaporation factor tuning is performed based on power law which is the basis of human forgetting.

III. PROPOSED ALGORITHM

Ant Colony Optimization (ACO) is an evolutionary algorithm in which ant colonies assist in locating shortest routes to food sources. One of the factors influencing the behaviour of ants and its performance is the pheromone evaporation. Here, the pheromone evaporation mechanism in ACO through the exponential generating function is discussed.

Ant routing algorithm (ARA) is a probabilistic technique for solving computational problems . The bug reports are extracted from the DMS system. Any new bug is assigned to a new developer. If that developer is unable to resolve the bug then the bug is passed on to another developer. This tossing behaviour is modelled in a tossing graph. A set of ants are set randomly to traverse the tossing graph. The behaviour of ants seeking a path between their colony and a source of food is imitated for the tossing graph [11][12][13][14] . In the natural world, ants (initially) wander randomly, and upon finding food return to their colony while laying down pheromone trails. If other ants find such a path, they are likely not to keep travelling at random, but to instead follow the trail, returning and reinforcing it if they eventually find food[16][17][18]. The pheromone trail starts to evaporate with time, thus reducing its attractive strength. The more time it takes for an ant to travel down the path and back again, the more time the pheromones have to evaporate. A short path, by comparison, gets traced over more frequently, and thus the pheromone density becomes higher on shorter paths than longer ones. Pheromone evaporation also avoids convergence to a locally optimal solution.

Evaporation Factor Tuning In Bug Triage

The evaporation factor tuning in the proposed system is based on the Ebbinghaus scientific study of memory [Note on the power law of forgetting]. The Ebbinghaus scientific study of memory concludes that the human forgetting of facts follow a power law function[19]. In the proposed system the evaporation factor tuning of the ants after each iteration is modeled as a power law function.

The rate of decay is a constant with respect to time.

$$\rho = \sum_{i=0}^{n} e^{-b*i}$$

Rate of decay slows down as strength of memory increases with time.

Let 'i' – be the index of the steps taken by the ant to reach the resolver. So if the path is traversed in the beginning of the iteration then the forgetting is less and if the path is traversed only in the later part of the iteration then the for getting is more. The pheromone the paths are tuned according to the power law.

N	No of ants
n	No of steps
D	Developer Network
E	Evaporation Factor
D*	Bug fixing developer
P	Pheromone

Step 1: Initialize N,n,P=0.5,i=0
Step 2: Initialize Vector V
Step 3: Assign N ants to starting nodes
Step 4: Assign Transition Probability (TP) to each link
Step 5: Each ant selects the next node according to TP and the P is incremented
Step 6: Store i in V
Step 7: Increment i
Step 8: Repeat Step 4 to Step 7 until ant reaches D*
Step 9: Update E with value from V

Figure 1: Algorithm for Evaporation factor Tuning

IV. SIMULATION RESULTS

The bug reports of Eclipse project from the year 2010 to 2013 were used for the experiments. The experiments were conducted in a system with Pentium4 Processor and 320 GB Hard Disc. The experimental environment comprises of Netbeans 7.2, Oracle and JDK. The metrics used are Path similarity and Prediction Accuracy. The results of the experiments for the parameters - Path Similarity and Prediction Accuracy are depicted in Figure.2 and Figure.3. It can be seen that the proposed system outperforms the existing system based on WBFS.

V. CONCLUSION AND FUTURE WORK

The assignment of bug reports is still primarily a manual process. Often bugs are assigned incorrectly to a developer or need to be discussed among several developers before the developer responsible for the fix is identified. These situations naturally lead to bug tossing, i.e., A bug report being reassigned from one developer to another. In this paper, we analysed the bug reports and their detailed activities from the Eclipse and Mozilla projects. We found that it takes a long time to assign and toss bugs. In addition, some bugs have a long tossing length, which means they are passed among many developers before the bug is actually fixed.

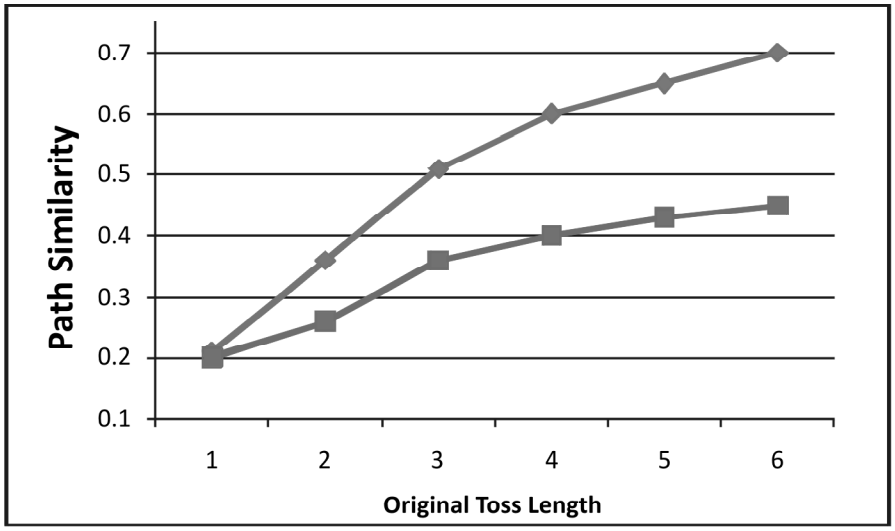

Figure 2: Graph for Path Similarity

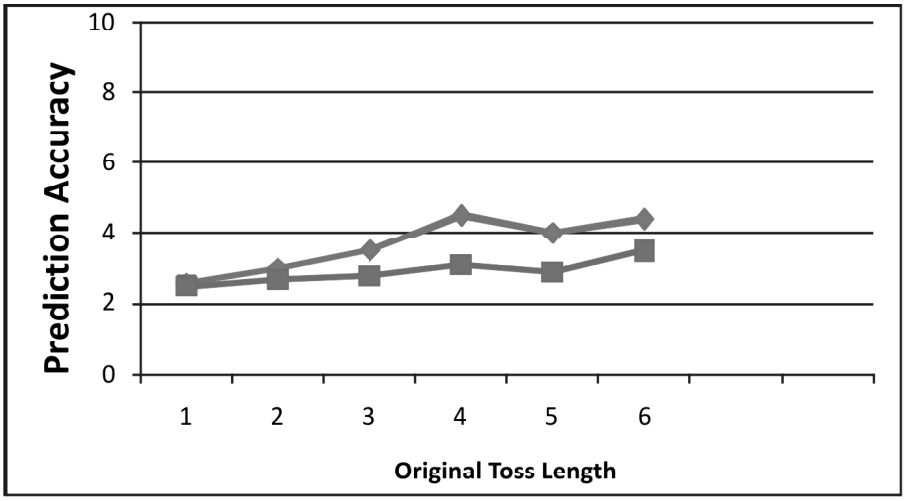

Figure 3: Graph Prediction Accuracy

To improve the bug assignment process and reduce unnecessary tossing steps, we proposed an ant route algorithm, which is used to reduce the tossing path length and improve the prediction accuracy. In addition, we compared the result obtained from the evaporation factor. The evaporation factor will be updated to default constant based on the number of times the ant uses the same path. If a path was used more frequently the updating occurs in the database. In this case the type of bug does not occur frequently thus the evaporation factor increases and it becomes maximum .Therefore, the path is removed from memory space. Hence we obtain the optimal path.

REFERENCES

[1] Pamela Bhattacharya, Iulian Neamtiu and Christian R.Shelton, "Automated, highly-accurate, bug assignment using machine learning and tossing graphs" The Journal of Systems and Software, vol.85, pp :2275-2292, May 2012.

[2] G. Jeong, S. Kim, and T. Zimmermann, "Improving bug triage with bug tossing graphs," in Proceedings of the the 7th joint meeting of the European software engineering conference and the ACM SIGSOFT symposium on The foundations of software engineering. ACM, 2009, pp. 111–120.

[3] D. Cubranic and G. C. Murphy, "Automatic bug triage using text categorization," in SEKE, 2004.

[4] Anvik, J., Hiew, L. and Murphy, G. C. Who Should Fix This Bug? In Proceedings of International Conference on Software Engineering. IEEE CS Press, 2006, pp.361-370.

[5] N. Bettenburg, R. Premraj, T. Zimmermann, and S. Kim, "Duplicate bug reports considered harmful. really?" in ICSM, 2008.

[6] John Anvik, Gail c. Murphy,"Reducing the Effort of Bug Report Triage: Recommenders for Development-Oriented Decisions " Journal of ACM Transactions on Software Engineering and Methodology (TOSEM) Vol 20 Issue 3, August 2011.

[7] Olga Baysal, Michael W. Godfrey, Robin Cohen , "A Bug You Like: A Framework for Automated Assignment of Bugs", IEEE 17th International Conference on Program Comprehension, 2009.

[8] P. Hooimeijer and W. Weimer. Modeling bug report quality. In ASE '07: Proceedings of the twenty-second IEEE/ACM International Conference on Automated Software Engineering, pp. 34–43, 2007.

[9] Zhang, T. and Lee, B. How to Recommend Appropriate Developers for Bug Fixing? In Proceeding of the 36th Annual IEEE International Computer Software and Application Conference(COMPSAC'12). IEEE CS Press, pp. 170-175, 2012.

[10] Xuan, J., Jiang, H., Ren, Z. and Zou, W. Developer Prioritization in Bug Repositories. In Proceeding of the 34th International Conference on Software Engineering(ICSE'12). IEEE Press, pp. 25-35, 2012.

[11] Akila V, Zayaraz G, and Govindasamy V, "Effective Bug Triage–A Framework," in International Conference on Communication and Convergence, pp. 114-120, 2014.

[12] V. Akila, G. Zayaraz, V. Govindasamy, "Bug Triaging Based on Ant Systems", International Journal of Bio Inspired Computing, Inderscience Publishers, Vol. 7, No. 4, 2015.

[13] GovindasamyV, Akila V, Banu priya , "Bug Triaging Using Multi-Attribute Bug Tossing Graph", Discovery Journal, 101-106, 2015.

[14] V. Akila, G. Zayaraz, V. Govindasamy, "Bug Triage In Open Source System- A Review", International Journal Of Collaborative Enterprise, Inderscience Publishers, Pp.299 – 319, 2014.

[15] Y. Wang and J. Xie, "An Adaptive Ant Colony Optimization Algorithm and Simulation", Journal of System Simulation, vol. 14 no. 1, pp. 31-33, 2012.

[16] G. Qin and J. Yang, "An improved ant colony algorithm based on adaptively adjusting pheromone", International Journal of Information and Control, vol. 31, no. 3, pp. 198-201, 2012.

[17] C. Chen and Z. Luo, "Rapid Algorithm Based on Ant Colony Algorithm", Computer Engineering, vol. 33, no.6, pp. 206-207, 2007.

[18] Loftus, G. R., "Evaluating forgetting curves" , *Journal of Experimental Psychology: Learning, Memory, & Cognition*, pp. 397–406, 1985.