International Journal of Latest Trends in Engineering and Technology
Vol.(7)Issue(4), pp.170-173
DOI: http://dx.doi.org/10.21172/1.74.022
e-ISSN:2278-621X

A REVIEW: BUG LOCALIZATION USING ANT COLONY OPTIMIZATION

Birinderjit Singh Kalyan[1]

Abstract— Bug localization is the task to locate the source code entities which are relevant from the bug report. Manual bug localization is a time consuming task as developers has to go through thousands of source code entities to locate the relevant one. Current research provides methods as various IR techniques, classifiers, combination of classifiers to improve bug localization. Due to the increasing size and complexity of current software applications, automated solutions for bug localization can significantly reduce human effort and software maintenance cost. This can be done with the help of an Ant Colony Optimization Technique.

Keywords— Bug Localization, Information Retrieval (IR), Ant Colony Optimization (ACO), Pheromone Trails.

I. INTRODUCTION

Bug Localization is a way of finding exactly the location of bugs in the program. It is a process of identifying the specific location or region of source code at various granularity levels such as the directory path, file, method or statement that is faulty and needs to be modified to repair the defect. Bug localization is a routine task in software maintenance. A technique for Bug localization based on a character n-gram based Information Retrieval (IR) model framed the problem of bug localization as a relevant document search task for a given query and investigate the application of character-level n-gram based textual features derived from bug reports and source-code file attributes. The IR models implemented and evaluate its performance on dataset downloaded from two popular open-source projects (JBoss and Apache). After conducting a series of experiments to validate this hypothesis and present evidences to demonstrate that the approach is effective. The accuracy of the approach is measured in terms of the standard and commonly used SCORE and MAP (Mean Average Precision) metrics for the task of bug localization.

An IR system typically begins with three-step preprocessing: text normalization, stop word removal, and stemming. Normalization involves removing punctuation, performing case-folding, tokenizing terms, etc., ultimately defining the initial vocabulary in which queries and documents will be represented. Next, a set of extraneous terms identified in a stop word list (e.g., “to”, “the”, “be”, etc.) are filtered out in order to improve efficiency and reduce spurious matches. Finally, stemming conflates variants of the same underlying term (e.g., “ran”, “running”, “run”) to improve term matching between query and document.

The success of IR-based bug localization is dependent on effectively matching the bug report to source files needing to be fixed. As discussed in Section II, even preprocessing issues can significantly impact IR accuracy. The classic IR challenge lies in effectively recognizing

[1] *Department of Electrical and Electronics Enginnering Chandigarh University, Ghauran*

important terms in the query and document, and assigning each greater weight for matching.[3]

Ant Colony Optimization: ACO technique comes under the swarm intelligence [Daniel Markel and Martin Middendorf. It is used in various dynamic applications. Ants started from nest in the search of food which is away from nest. Each ant follow different path to reach to the food source and secrete pheromone liquid at the path as a mark to attract other ants. Ants choose the path depending upon the pheromone and path is marked after collecting food. So at the end shortest path has the highest probability. The act of making trail of pheromone liquid is very useful to find out good food source direction. Moreover, ACO deals with a process in which decreasing in amount of pheromone deposited on every path by the time is known as trail pheromone evaporation. When they complete their search to find out best result or to reach final destination, they update their trail to attract other ants. Each conspiratorial problem defines its own updating criteria depending on its own local search and global search respectively. Ants are able to find shortest path on the basis of pheromone information laid on ground by other ants of same colony. Ant searching for food goes for all possible trails, but in last chooses the trail with largest deposit pheromone and update according to the latest pheromones. There is population of ants in ants system, which is working as agent to find out shortest path and communicate with other ants. When reached its destination, the direction of path it follow is based upon the amount of pheromone it detects and is made by decision probabilistic.

II. ANT COLONY OPTIMIZATION FLOWCHART (ACO)

A scheme of an ACO flowchart in figure 1 is given in the following.

1. Represent the solution space by a construction graph.
2. Set ACO parameters and initialize pheromone trails.
3. Generate ant solution from each ants walk on the construction graph
4. Update pheromone intensities.
5. Go to step 3, and repeat until convergence or termination conditions are fulfilled.

The large majorities of these applications are to *NP-hard* problems; that is, to problems for which the best known algorithms that guarantee to identify an optimal solution have exponential time worst case complexity.

The use of such algorithms is often infeasible in practice, and ACO algorithms can be useful for quickly finding high-quality solutions. ACO algorithms are based on a parameterized probabilistic model the *pheromone model*, is used to model the chemical pheromone trails.

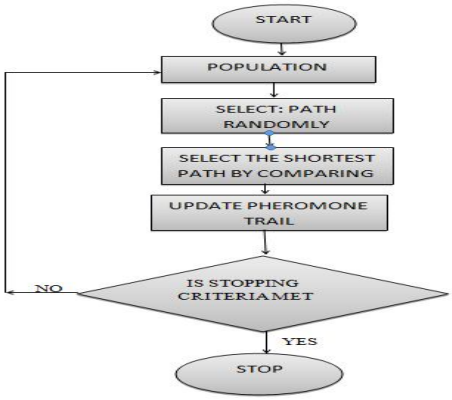

Figure 1: Flow chart of ACO

III. ADVANTAGES OF ANT COLONY OPTIMIZATION

ACO displays powerful robustness.
2. It has an advantage of distributed computation which avoids premature convergence.
3. It is adaptive in nature and can adapt changes easily.
4. It gives positive feedback which leads to discovery of good solutions.
5. It can be used in dynamic applications.
6.To communicate with each other ants use pheromone liquid.
7. The whole process is done in an organized manner.

IV. DIFFERENT BUG-LOCALIZATION TECHNIQUES

A) Static Analysis : Static analysis of software program source codes is a kind of formal analysis where all possible paths of the source code are explored. This kind of analysis strongly depends on the syntax and the semantics of the underlying programming language. This kind of analysis is performed without actually executing the program. Some of the prominent methods of static analysis are Model Checking [Clarke et al. 1999], Data-Flow Analysis and Abstract Interpretation [Cousot and Cousot 1977]. By a straightforward reduction of the Halting Problem [Halting Problem 1930] it is possible to prove that (for any Turing complete language) finding all possible run-time errors in an arbitrary program (or more generally any kind of violation of a specification on the final result of a program) is undecidable: there is no mechanical method that can always answer truthfully and faithfully whether a given program may or may not exhibit runtime errors. In spite of this inherent limitations there are a few tools that implements static analysis based software verification method : BLAST [Henzinger et al. 2002], Clang [Lattner 2007] and Microsoft's SLAM Toolkit [Microsoft Research 2010] are among the most prominent and mostly used tools. One of the most severe limitation of this method is that it often suffers from capacity limitations as reported widely in literature. Intuitively static analysis based methods try to explore all possible executions of the program. Since the number of executions is infinite (even for very simple and relatively small program), hence static analysis based methods often do not scale. In this survey static analysis and related methods are not our prime objective. In the next section we introduce statistical analysis based bug localization briefly.

*B) Statistical Analysis :*To alleviate the capacity related issues, researchers have studied program bug localization methodologies based on dynamic analysis. This methodology has a common framework in all of its variants : the program source code is co-executed along with a large number of possible test cases and execution traces are dumped. Then, suitable machine learning and data mining algorithms are applied to extract out likely invariants and possible bug locations in the software program. Clearly, even if a large number of test suites are used to generate the trace data, it is still impossible to execute all possible paths in the program. However, this makes the problem of invariant generation and bug localization more tractable. But obviously it comes for a price; some of the invariants generated may be spurious which are true for those test cases but do not hold good in general for the program. Also, in case of bug localization it may give rise to false positives. Over the past decades several such dynamic trace based software bug localization techniques have been proposed by the researchers. Some of those techniques use a hybrid approach where knowledge learned from dynamic traces is used to prune significant portion of the state space of static analysis making it more tractable. holmes [Chilimbi et al. 2009] is one such method. In some other methods, different state-of-the-art data mining algorithms along with rigorous statistical analyses are performed over the dynamic traces to locate possible bug locations in the program. CBI [Liblit 2007], [Liblit et al. 2005], SOBER [Liu et al. 2005], [Liu et al. 2006] and DES [Hu et al. 2008] are some of the prominent statistical analysis based bug localization techniques. This survey by no means is as comprehensive as the numerously vast and diverse works that have been done in the domain of software bug localization. In this survey paper

we present the main ideas and the mathematical background of the statistical analyses based software bug localization approach which has evolved over the past 10 years.

IV. APPLICATIONS OF ANT COLONY OPTIMZATION

ACO is used in routing problem i.e. traveling salesman problem, vehicle routing and sequential ordering.

1. It is applied to solve job scheduling problem, project scheduling and applied in multilevel framework also.
2. It is also used to solve many assignment problems like frequency assignment, graph coloring and quadratic assignment problem.
3. ACO is also applied in the field of networking like optical network routing, connection oriented routing etc.
4. It can be used to solve knapsack problem and set covering problems and applied in fuzzy systems.
5. Nearest neighbor node choosing rule can be solved by using ant colony optimization.[2]

V. CONCLUSION

Selecting a persuasive bug localization methodology generally requires expert knowledge regarding the program. Bug Localization is carried out in order to find location of bugs and to achieve fault-free program. In this we have discussed Bug localization by n-gram character model. We have also discussed about the advantages and workflow of Ant Colony Optimization Technique. To render help to the software tester in three aspects – i.e. minimum reach time, localization path, and optimal solution. From our observations, we conclude that by using Ant colony Optimization technique we can easily localize the shortest path for finding a bug and obtain minimum reach time by optimizing the path of a bug.

VI. REFERENCES

[1] Ferdian Thug, “Bug Localizer: Integrated tool support for bug localization”[,pp. 767-770, FSE 2014 Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, 2014.

[2] Sujatha .S.R ,Dr.M.Siddappa “A study on efficient localization methods using swarm intelligence” ISSN 2320-7345, June 2014.

[3] Sangeeta Lal and Ashish Sureka, “A Static Technique for Fault Localization Using Character N-Gram Based Information Retrieval Model”; Proceedings of ISEC '12, Feb. 22-25, 2012 Kanpur, UP, India.

[4] Nicholas Giuseppe, James A. Jones, “On the Influence of Multiple Faults on Coverage-Based Fault Localization”; ISSTA ’11, July 17–21, 2011, Toronto, ON, Canada.

[5] S. Rao and A. Kak, “Retrieval from Software Libraries for Bug Localization: A Comparative Study of Generic and Composite Text Models,” Proc. Eighth Working Conf. Mining Software Repositories, pp. 43-52, 2011.

[6] V.Selvi, Dr.R.Umarani, "Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques”, Volume 5– No.4, , International Journal of Computer Applications (0975 – 8887) August 2010.

[7] W. Eric Wong, and Vidroha Debroy, “Software Fault Localization”; IEEE Reliability Society Annual Technology Report 2009.

[8] Zheng, T. G, Huan, H, & Aaron, Ant Colony System Algorithm for Real-Time Globally Optimal Path Planning of Mobile Robots. Acta. Automatica. Sinica, 33(3), 279-285. 2007.

[9] Dorigo M and Blum C,” Ant colony optimization theory: A survey”, Theoretical Computer Science, Volume344, Issues 2-3, November 2005.

[10] Blum C,” Ant colony optimization: Introduction and recent Trends”, Physics of Life Reviews, Volume 2, Issue 4, December 2005.

[11] Dorigo M and Stutzle T,” Ant Colony Optimization”, MIT Press 2004.