Contemporary Engineering Sciences, Vol. 7, 2014, no. 30, 1649 - 1655
HIKARI Ltd, www.m-hikari.com
http://dx.doi.org/10.12988/ces.2014.411222

Incremental Spectrum Cloning Algorithm for Optimization of Spectrum-Based Fault Localization

Patrick Daniel and Kwan Yong Sim

Faculty of Engineering, Computing and Science
Swinburne University of Technology, Sarawak Campus
Kuching, Sarawak, Malaysia

Soonuk Seol

School of Electrical, Electronics and Communication Engineering
Korea University of Technology and Education (KOREATECH)
Cheonan, Chungnam, Rep. of Korea

Abstract

Spectrum-based Fault Localization has emerged as a cost effective method to locate faulty code in software during the debugging process. Recent studies have shown that spectra (execution profiles) cloning for fail test cases can effectively improve the performance of certain spectrum-based Fault Localization ranking metrics. However, the amount of cloning required to optimize the performance varies from one program to another. This is because the structure and fault content of each program is unique. Furthermore, both insufficient cloning and excessive cloning may result in performance deterioration. In this paper, we propose an incremental spectra cloning algorithm that will clone the spectra of fail test cases up to an amount that optimizes the fault localization performance of the ranking metrics used for the software debugged. Experiments conducted on faulty versions of real life software have shown that the proposed algorithm could optimize the cost effectiveness of spectrum-based fault localization by reducing the number lines examined to locate the faulty code.

Keywords: Software Engineering, Software debugging, Fault Localization

1 Introduction

Software testing and debugging are the main quality assurance activities in software development life cycle. The goal of software testing is to detect possible faults in software systems. Once a fault is detected, the next follow up activity is software debugging, where software developer examines the information gathered from the testing phase to locate error in the software code that causes the fault. This process is costly as it is both labor-intensive and time consuming. Software developers need to manually trace the test case execution and inspect the software code line by line until the faulty line of code is located. Previous study [1] showed that software testing and debugging could consume 50% to 80% of the overall software project cost. Therefore, apart from software testing, software debugging has also been a focus area of study to reduce software project cost.

In recent years, Spectrum-based fault localization (SBFL) has been studied intensively as an effective technique to rank program codes according to their likeliness to be faulty. It is based on the rational that program codes that have been executed by more fail test cases and less pass test cases are more likely to be faulty, and vice-versa. Therefore, by analyzing the execution profiles (spectra) of test cases, program codes can be ranked for inspection during the debugging process according to their likeliness to fault. This will reduce the amount of codes need to be inspected in order to locate the faulty code.

Numerous ranking metrics such as Jaccard [2], Euclid [3] and Ochiai [4] have been proposed in SBFL studies to rank program codes according to their likeliness to be faulty. For each line of code in a faulty program, the likeliness score is computed using these ranking metrics based on four coefficients, namely, aef, anf, aep, and anp. The first coefficient, aef, represents the number of fail test cases that have executed that line of code. The second coefficient, anf, represents the number of fail test cases that have not executed that line of code. The third coefficient, aep, represents the number of pass test cases that have executed that line of code, while the forth coefficient, anp, represents the number of pass test cases that have not executed that line of code. A good ranking metrics for SBFL should produce a high score for a faulty line of code that have high aef and anp but low anf and aep. The formula for ranking metrics Jaccard, Euclid and Ochiai are shown in Table 1.

Recent studies [5] [6] [7] [8] on the effect of imbalance test suites on SBFL performance found that there are significantly more pass test cases than fail test cases in typical program spectra used for SBFL. However, imbalance composition

Table 1 The Formula of SBFL ranking metrics

Jaccard [2] = $\frac{aef}{aef+anf+aep}$	Euclid [3] = $\sqrt{aef + anp}$	Ochiai [4] = $\frac{aef}{\sqrt{(aef+anf)(aef+aep)}}$

of pass and fail test cases is actually counter-productive for the performance of SBFL ranking metrics which rely on the execution profiles of fail test cases to provide the essential information on the location of faulty code. Based on these studies, cloning of fail test cases have been proposed in [7] to improve the performance of SBFL by constructing a balanced test suite with equal number of pass and fail test cases. Subsequent study in [9] suggests that the performance of SBFL can be further improved by cloning the spectra fail test cases beyond balanced test suites. However, the amount of cloning to achieve optimal performance varies from one program to another as the structure and fault content of each program is unique. Moreover, both insufficient cloning and excessive cloning may result in performance deterioration.

In this paper, we propose an incremental spectra cloning algorithm that will clone the spectra of fail test cases incrementally up to an amount that optimizes the fault localization performance of the SBFL ranking metrics used for the software debugged. 21 cases of experiments conducted on faulty versions from seven real life applications in Siemen Test Suites [10] have shown that the proposed algorithm could optimize the cost effectiveness of spectrum-based fault localization by reducing the number lines examined to locate the faulty code.

2 Incremental Spectrum Cloning Algorithm

Cloning the spectra of fail test cases will increment the value of aef and enf in the SBFL ranking metrics. This will change the metric score for each line of code which indicate its likeliness to be faulty. Intuitively, this will increase the score and ranking of the faulty code which is executed by fail test cases. Therefore, cloning the spectra of fail test cases could improve the performance of SBFL metrics. However, empirical study conducted in [8] also suggested that excessive cloning may also results in performance deterioration. In view of this, we propose a novel cloning algorithm that clones the spectra of fail test cases incrementally (one clone at a time) until the ranking of the program code stabilize and does not change over the past 200 iterations. This incremental spectrum cloning algorithm is outlined in Figure 1.

A *stabilization_count* variable is used in this algorithm to track the number clones added without causing ranking change to the lines of code in the program. In this algorithm, we propose a *stabilization_count* of 200 as the stopping condition for the incremental cloning algorithm because it is unlikely that spectrum cloning of fail test cases will cause further change to the ranking if there have been no change after the past 200 clones have been added to the test suite.

Set *stabilization_count* to 0.
Compute the aep, aef, anp and anf for each line of code.
Compute the SBFL ranking metric score for each line of code.
Sort (Rank) lines of code in descending order according to the SBFL ranking metric score (i.e. likeliness to be faulty).
WHILE *stabilization_count < 200*
    Create one clone for the spectra of the original set of fail test cases in the test suite. ***Add*** the cloned spectra to the test suite.
    Update the count for aef and anf for each line of code.
    Re-compute the ranking metric score for each line of code
    Update the ranking for each line of code according to the re-computed SBFL ranking metric score (i.e. likeliness to be faulty).
    Sort (Rank) lines of code in descending order according to the SBFL ranking metric score (i.e. likeliness to be faulty).
    IF the ranking of ALL lines of code has not changed compare to the previous iteration
        Increment *stabilization_count* by 1,
    ELSE
        Reset *stabilization_count* to 0.
END WHILE

Figure 1 Incremental Spectrum Cloning Algorithm

3 Experiments and Results

In order to evaluate the effectiveness of the proposed algorithm in optimizing the performance of SBFL metrics, experiments have been conducted on faulty versions from seven real life applications in Siemen Test Suites [9]. The faulty versions selected from Siemen Test Suite are print_tokens v5, print_tokens2 v3, replace v2, schedule v4, schedule2 v8, tcas v5, and tot_info v12. Jaccard, Euclid and Ochiai are used to compute the score for each line of code in the faulty program to indicate its likeliness to be faulty. Lines of code in the program are then ranked in descending order for inspection to locate the faulty line code.

The percentage of code inspected (pci) to locate the faulty line of code is used to evaluate the performance of each SBFL ranking metric under the proposed incremental spectrum cloning algorithm. The lower the pci, the better is the performance of an SBFL metric because the less lines of code need to be inspected to locate the faulty code.

The experiment results are presented in Table 2. The performance (measured as pci) of the proposed incremental spectrum cloning algorithm is compared against the performance of the same SBFL ranking metric when no cloning is done. From the results in Table 2, it can be observed that the proposed incremental spectrum cloning algorithm has successfully optimized the performances (measured in pci) of Jaccard and Euclid for all faulty programs used

Table 2 Performance Improvements for SBFL Metrics

SBFL Metrics	Experiments	*Percentage of Code Inspected (pci) to Locate the Faulty Line of Code*						
		print_tokens v5	*print_tokens2 v3*	*replace v2*	*Schedule v4*	*schedule2 v8*	*Tcas v5*	*tot_info v12*
Jaccard	No spectrum cloning	3.73	17.88	6.93	4.62	29.97	19.65	7.14
	Incremental Spectrum Cloning Algorithm	0.36 (stops at 389 clones)	6.29 (stops at 865 clones)	0.36 (stops at 1108 clones)	0.97 (stops at 822 clones)	20.19 (stops at 699 clones)	13.87 (stops at 445 clones)	4.19 (stops at 460 clones)
	% of pci improvement	90.3%	64.8%	94.8%	79.0%	32.6%	29.4%	41.3%
Euclid	No spectrum cloning	13.32	21.4	18.83	7.79	33.88	24.86	15.52
	Incremental Spectrum Cloning Algorithm	0.36 (stops at 1623 clones)	6.29 (stops at 1554 clones)	0.36 (stops at 2031 clones)	0.97 (stops at 831 clones)	20.20 (stops at 546 clones)	13.87 (stops at 495 clones)	4.19 (stops at 530 clones)
	% of pci improvement	97.3%	70.6%	98.1%	87.5%	40.4%	44.2%	73.0%
Ochiai	No spectrum cloning	0.53	19.25	3.20	4.62	33.22	20.23	7.14
	Incremental Spectrum Cloning Algorithm	0.36 (stops at 3079 clones)	7.07 (stops at 3963 clones)	0.36 (stops at 3339 clones)	0.97 (stops at 2663 clones)	20.52 (stops at 2835 clones)	20.23 (stops at 363 clones)	4.19 (stops at 1188 clones)
	% of pci improvement	32.1%	63.3%	88.8%	79.0%	38.2%	0.0%	41.3%

in the experiments. For Ochiai, the performances have also been optimized for all faulty programs except one (Tcas v5) where there is no improvement. The magnitude of improvement is also very encouraging with the highest improvement of 94.8% for Jaccard, 98.1% for Euclid and 88.8% for Ochiai. Overall, the average improvements achieved by the proposed incremental spectrum cloning algorithm are 61.8% for Jaccard, 73.0% for Euclid and 48.9% for Ochiai. It could also be observed that the number of clones made to achieve these improvements ranges from 363 clones to 3339 clones.

4 Discussions and Conclusion

Despite its simplicity in approach, the proposed incremental spectrum cloning algorithm has significantly improved the cost effectiveness of the SBFL because less lines of code need to be inspected in order to locate the faulty code. This is consistent with results from a related study [11] which attempted to optimize the performance of SBFL ranking metrics by assigning higher weighting to fail test cases. The proposed incremental spectrum cloning algorithm not only optimized the performance of SBFL ranking metrics, but also provided a viable solution to the problem on the amount of cloning or additional weighting required to optimize the performance. As for future work, we plan to explore the potential use of the proposed approach in related classification [12] and clustering [13] applications.

References

[1] J. S. Collofello and S. N. Woodfield, Evaluating the effectiveness of reliability-assurance techniques. *Journal of Systems and Software*, **9** (1989), 191–195. http://dx.doi.org/10.1016/0164-1212(89)90039-3

[2] P. Jaccard, Etude Comparative de la Distribution Florale Dans une Portion des Alpes et des Jura. Bull. Soc, *Vaudoise Sci. Nat* (1901), 547 – 579.

[3] E. Krause, Taxicab Geometry. *Mathematics Teacher*, (1973), 695 – 706.

[4] A. Ochiai, Zoogeographic Studies on the Soleoid Fishes found in Japan and its Neighbouring Regions. Bull. Jpn. *Soc. Sci. Fish* (1957), 526 – 530.

[5] C. Gong, Z. Zheng, W. Li and P. Hao, Effects of class imbalance in test suites: an empirical study of spectrum-based fault localization, 2012 IEEE 36th Annual Computer Software and Applications Conference Workshops (2012), 470-475, IEEE. http://dx.doi.org/10.1109/compsacw.2012.89

[6] P. Rao, Z. Zheng, T. Y. Chen, N. Wang and K. Cai, Impacts of Test Suite's Class Imbalance on Spectrum-Based Fault Localization Techniques. 2013 13th International Conference on Quality Software (QSIC) (2013), 260 - 267. http://dx.doi.org/10.1109/qsic.2013.18

[7] Y. Gao, Z. Zhang, L. Zhang, C. Gong and Z. Zheng, A Theoretical Study: The Impact of Cloning Failed Test Cases on the Effectiveness of Fault Localization. 2013 13th International Conference on Quality Software (QSIC) (2013), 288 - 291. http://dx.doi.org/10.1109/qsic.2013.23

[8] P. Daniel and K. Y. Sim, Debugging in the Extreme: Spectrum-based Fault Localization with Limited Test Cases. *International Journal of Software Engineering and Its Applications (IJSEIA)*, **7** (2013), 403 – 412. http://dx.doi.org/10.14257/ijseia.2013.7.5.35

[9] P. Daniel, K. Y. Sim and S. Seol, Improving Spectrum-based Fault-localization through Spectra Cloning for Fail Test Cases Beyond Balanced Test Suite, *Contemporary Engineering Sciences*, **7** (2014), 677 – 682.

[10] H. Do, S. Elbaum, and G. Rothermel. Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and its Potential Impact. *Empirical Software Engineering*, **10** (2005), 405 – 435. http://dx.doi.org/10.1007/s10664-005-3861-2

[11] Y. S. You, C. Y. Huang, K. L. Peng and C. J. Hsu, Evaluation and Analysis of Spectrum-Based Fault Localization with Modified Similarity Coefficients for Software Debugging. In Proceedings of the 2013 IEEE 37th Annual Computer Software and Applications Conference (2013) 180 - 189. http://dx.doi.org/10.1109/compsac.2013.32

[12] Suyeon Yoo and Sehun Kim, Two-Phase Malicious Web Page Detection Scheme Using Misuse and Anomaly Detection. *International Journal of Reliable Information and Assurance*, **2** (2014), 1 - 9.

[13] T. Sharma, B. Kumar, G. S.Tomar, An Efficient Condensed Cluster Stable Election Protocol in Wireless Sensor Networks. *International Journal of Smart Device and Appliance*, **1** (2013), 17 - 28.

Received: October 1, 2014; Published: December 2, 2014