ESANN 2025 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium) and online event, 23-25 April 2025, i6doc.com publ., ISBN 9782875870933. Available from http://www.i6doc.com/en/.

Evolutionary Fault Localization Based on the Diversity of Suspiciousness Values

Willian de Jesus Ferreira[1], Plinio S. Leitao-Junior[1],
Deuslirio Silva-Junior[1] and Rachel Harrison[2]

1- Universidade Federal de Goias (UFG) - Instituto de Informatica (INF)
Alameda Palmeiras, Quadra D, Campus Samambaia, Goiania, Goias - Brazil

2- Oxford Brookes - School of Engineering, Computing and Maths
Headington Campus, OX3 0BP, Oxford - United Kingdom

Abstract. *Context.* Fault localization (FL) is a software lifecycle activity and its automation is a challenge for researchers and practitioners. *Method.* The study focuses on evolutionary fault localization and introduces a novel Genetic Programming (GP) approach that evolves FL heuristics based on the diversity of the suspiciousness score of program statements – a score to grade how faulty a statement is. *Experimental analysis.* The approach was evaluated against baselines, which include the canonical GP, in benchmarks with real programs and real faults. *Conclusion.* The results showed the competitiveness of the approach through evaluation metrics commonly used in the research field.

1 Introduction

Software is subject to the presence of faults (defects), which impact its quality, production and maintenance costs. Fixing faults requires knowledge about which program statements are likely to be defective. An important source of information about faults is the testing activity, where data (test spectra) can be collected from test case execution.

Test spectra refers to the control flow executed by a set of test cases, as well as the test result (failure or success). A negative test occurs when the program output obtained by executing a test case is different from what was expected, otherwise the test is positive. For instance, the spectrum of a test case records whether the test is positive or negative and distinguishes the statements executed from the statements not executed by the test case.

Intuitively, if a statement is executed in negative tests, but is not executed in positive tests, then such a statement has a greater chance of being faulty (defective) in relation to statements that are executed only in positive tests.

Spectrum-based Fault Localization (SFL) uses the test spectra to automatically compute a score (suspiciousness value) that ranks how faulty each statement is [1]. In fact, an FL heuristic is an equation (formula) to derive such a score for each program statement. The idea is to have an ordering of statements based on their score, in order to guide the tester in locating faults.

Evolutionary fault localization is a research field of SFL that uses the test spectra to automatically generate FL heuristics [2]. In this context, Genetic

Programming (GP) [3] has been applied on the evolution of FL solutions, mainly by deriving competitive heuristics for the software under test [4], [5].

Our investigation focuses on the diversity of scores that rank program elements. If two or more statements have the same suspiciousness value as the faulty one, then it is not possible to precisely determine which of them is the faulty statement. Therefore, the uniqueness of suspiciousness values potentially impact the fault localization effectiveness.

We hypothesize that the uniqueness of suspiciousness values adds value in guiding the development of heuristics through GP. We suggest that our approach may enable GP-evolved heuristics to empirically outperform the canonical GP and other reference baselines. To our knowledge, this is the first time that the uniqueness of suspiciousness values has been explored as a factor to improve GP-based solutions for fault localization.

The paper is structured as follows. Sections 2 and 3 introduces the test spectra and the proposed approach, respectively; the experiments are described in Section 4; Section 5 presents the results of their analysis; and Section 6 concludes and shows future work.

2 Test Spectra and Suspiciousness Values

For simplicity, the behavior of a statement when executing a set of test cases is summarized in four spectrum variables: ***es*** and ***ns*** represent the number of positive tests that execute and do not execute the statement, respectively; ***ef*** and ***nf*** represent the number of negative tests that execute and do not execute the statement, respectively.

A **spectrum entry (SE)** is a quadruple - *[ef, es, nf, ns]* - that represents the values of spectrum variables, concerning a particular program statement. An FL heuristic calculates a suspiciousness value based on the spectrum variables. As a consequence, if two or more statements have the same spectrum entry (e.g. control flow dependence), they will have identical suspiciousness values.

The **uniqueness of test spectrum entries (uSE)** refers to the number of distinct *[ef, es, nf, ns]* quadruples related to all program elements. The **uniqueness of a suspiciousness value (uSV)** refers to the number of distinct suspiciousness values related to all program elements. Note $uSV \leq uSE$ and higher uSV means higher diversity of suspiciousness values.

Finally, the **uSV ratio**, calculated as uSV/uSE, refers to how unique suspiciousness values are with respect to the number of distinct spectrum entries. The ratio ranges from 0.00 to 1.00, and higher values means better diversity of suspiciousness values.

3 Approach

Our approach involves adding the uniqueness suspicion value (uSV) to the GP metaheuristic learning process. This addition aims to guide the training of fault localization methods, promote individuals with higher uSV, and improve the performance of GP-evolved heuristics in fault localization.

Basically, we use an evolutionary strategy in which individuals (solutions) with smaller fitness function are privileged for the genetic operators (crossover, mutation and selection).

On incorporating the uSV effect to the fitness function, we use a common measure (e.g. number of program elements up to first fault location) but divided by $(uSV * w)$ where $w \in \{1, 3, 5, 7\}$. After preliminary analysis, we observed that w equals 3 produced results that were identified as most promising. Then we have proposed one model aiming to assess our approach: $GP/(uSV * 3)$, the canonical GP fitness function is divided by *three times* uSV.

4 Experimental Design

To address the evaluation of the proposal, we set out a research question: Does the uniqueness of suspicion values guide the generation of effective and competitive evolutionary approaches for fault localization?

Regarding the benchmark, we choose *Defects4J*, proposed by [6], a set of Java programs that represent real failures in real programs and that has been explored in the research field.

The baselines includes human-made FL heuristics, such as $Tarantula$ [7] e $Ochiai$ [8], and heuristics evolved by evolutionary approaches such as the canonical GP [2] [9]. Table 1 analyzes the benchmark and shows for each program the average of: the number of statements, the spectrum entries uniqueness (uSE), and the suspiciousness values uniqueness (uSV) for the baseline approaches as well as the proposal. Since there are several faulty versions per program, so the arithmetic mean was calculated. Note GP derives better diversity of suspiciousness values than the others.

Table 1: Test spectrum analysis - *Defects4J* Programs

				uSV	
Program	**Statements**	uSE	**Tarantula**	**Ochiai**	**GP**
Chart	4167,92	70,38	32,58	34,79	57,24
Math	2266,11	38,74	15,85	16,22	21,42
Lang	832,55	23,62	8,93	9,27	15,31
Time	5078,5	371,85	156,92	157,62	295,02
Mockito	1820,54	205,86	89,30	89,97	203,23

To assess the proposed approach, we use two evaluation metrics: **Exam**, the proportion of investigated statements, in relation to the number of program elements, until finding a fault; and **Wasted Effort(wef@n)**, the number of statements investigated until locating a fault, but considering only n first positions of the suspiciousness ranking. In both metrics, lower values are better.

All experiments were conducted on Debian GNU/Linux version 10. The GP implementation was done using the DEAP[1] (Distributed Evolutionary Al-

[1] http://deap.readthedocs.io

gorithms in Python) framework version 1.3.1, and the algorithms were executed with Python[2] version 3.7.3. This setup provided the necessary environment for running the experiments and ensuring consistent results throughout the tests.

In terms of GP parameters, we explored various configurations to balance the trade-off between fault localization effectiveness and the cost of training. The final settings were as follows: a population size of 100 individuals, each initialized randomly with tree structures having a minimum height of 4 and a maximum of 8. A tournament selection operator with a size of 3 was used, alongside a crossover operator with a rate of 0.8, a subtree replacement mutation operator with a rate of 0.07, and a point mutation operator with a rate of 0.03. The stopping criterion was set at 50 generations. For tie-breaking suspiciousness values, we applied a worst-case strategy, assigning all tied elements to the worst tied position. Additionally, to address overfitting, we used 10-fold cross-validation and repeated the experiments 10 times to minimize the stochastic effects.

5 Results

Figure 1 uses *Time*, *Chart* and *Lang* programs to plot the uSV ratio for the three baselines as well as for the approach. It shows that the heuristics based on the *canonical GP* improved the value of the uSV ratio over *Tarantula* and *Ochiai*. In turn, our approach outperformed all baselines, as it reached the maximum value of the uSV ratio (i.e. 1.0), which $GP/(uSV * 3)$ distinguishes all entries in the test spectrum with respect to their suspiciousness values.

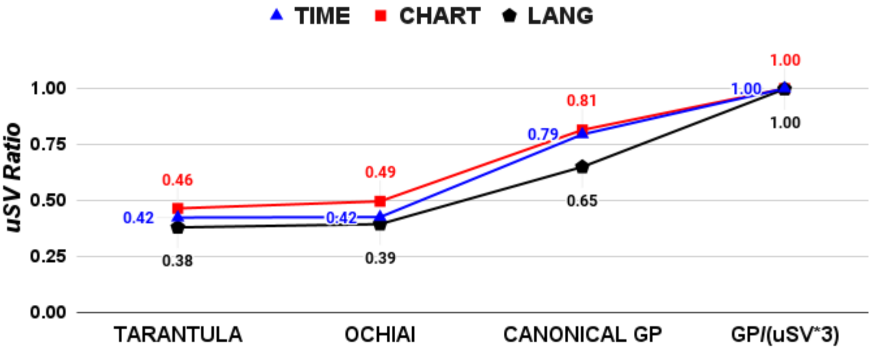

Fig. 1: **uTSE ratio** - *Time*, *Chart* and *Lang* programs.

The Figure 2 shows the results of the *Exam* metric for all approaches. Our approach achieved better effectiveness in relation to all baselines. Regarding our research question, based on all findings we conclude that the maximum value (or approximation) of the uSV ratio enables the fault localization heuristics to have superior effectiveness.

[2] http://python.org

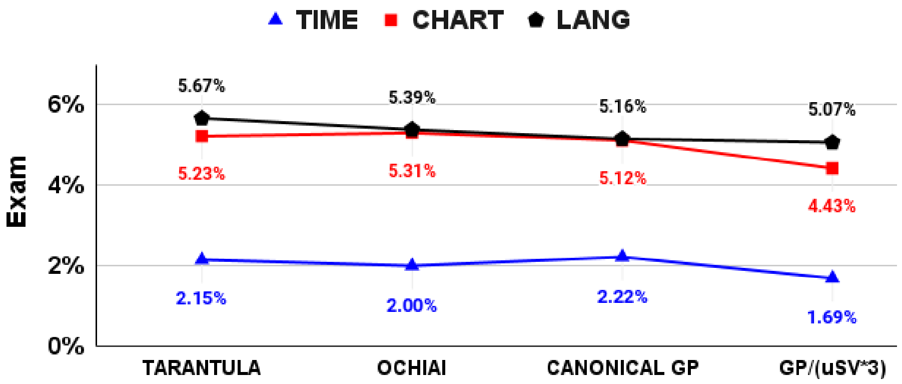

Fig. 2: **Exam** - *Time*, *Chart* and *Lang* programs.

In order to deal with the stochastic effect that is inherent in the evolutionary approaches as well as to raise the findings' reliability, two statistical tests were applied: the Wilcoxon pair comparison test and the Vargha & Delaney $\hat{A}_{12}$ test, as recommended by *Arcuri* and *Briand* [10]. We applied both tests to the results from the evaluation metrics used in the experiment: *Exam* and *Wasted Effort* ($wef@n$). Please see Tables 2 and 3.

The Wilcoxon test shows that the training approach present results for both evaluation metrics with equivalent significance to the baseline (GP meta-heuristic). This finding reveals the competitiveness of the approach.

Table 2: Statistical Analysis *Exam* - Varga e Delaney $\hat{A}_{12}$

Program	Tarantula	Ochiai	GP
Chart	0.47	0.51	0.48
Lang	0.49	0.50	0.51
Math	0.52	0.52	0.51
Mockito	0.59	0.61	0.57
Time	0.52	0.52	0.55

Table 3: Statistical Analysis *Wef@n* - Varga e Delaney $\hat{A}_{12}$

Program	wef@5	wef@10
Chart	0.50	0.50
Lang	0.50	0.53
Math	0.52	0.52
Mockito	0.52	0.53
Time	0.55	0.59

Threats to Validity. We mitigated threats to *internal validity*, i.e. reducing results by chance, by using baselines and evaluation measures used in prior studies. Experiments were run at least 10 times for each instance (program version) in order to deal with stochastic variation and 10-fold cross-validation strategy for generating robust evolutionary GP-evolved heuristics. To deal with *external validity*, i.e. whether the results can be generalized, a benchmark used in many contexts related to software engineering was applied. Finally, to cope with threats to *construct validity*, how well the measurements are actually correlated to what they claim to do, measures are used that are similar to previous studies.

6 Conclusion

This study introduces an approach for evolutionary fault localization, by using the diversity of program statements' suspiciousness values to drive the generation of heuristics, through the Genetic Programming (GP) metaheuristic. The research proposes a metric to assess such a diversity, uSV ratio, which measures how distinct the suspiciousness value of a program statement is from the others.

Regarding the empirical evaluation, the investigation used the *Defects4J* benchmark with real defects and real programs, three baselines that include an evolutionary state of the art (the canonical GP) and evaluation metrics widely used in the research field, aiming to reduce threats to the findings' validity.

We conclude that an approximation to the maximum value of the uSV ratio promotes fault localization effectiveness. It shows the competitiveness of the proposal and we confirmed statistically that the diversity of suspiciousness scores produces superior heuristics for fault localization.

In future work, further experiments are needed, including the use of other benchmarks from the research field and the incorporation of additional data sources (e.g. data flow and mutation spectra).

References

[1] W. Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. A survey on software fault localization. *IEEE Transactions on Software Engineering*, 42(8):707–740, 2016.

[2] Shin Yoo. Evolving human competitive spectra-based fault localisation techniques. In Gordon Fraser and Jerffeson Teixeira de Souza, editors, *Search Based Software Engineering*, pages 244–258, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[3] John R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, USA, 1992.

[4] Ya Zou, Hui Li, Dongcheng Li, Man Zhao, and Zizhao Chen. Systematic analysis of learning-based software fault localization. In *2024 10th International Symposium on System Security, Safety, and Reliability (ISSSR)*, pages 478–489, 2024.

[5] Plinio S. Leitao-Junior, Diogo M. Freitas, Silvia R. Vergilio, Celso G. Camilo-Junior, and Rachel Harrison. Search-based fault localisation: A systematic mapping study. *Information and Software Technology*, 123:106295, 2020.

[6] René Just, Darioush Jalali, and Michael D. Ernst. Defects4j: a database of existing faults to enable controlled testing studies for java programs. In *International Symposium on Software Testing and Analysis*, page 437–440, New York, NY, USA, 2014. ACM.

[7] James A. Jones and Mary Jean Harrold. Empirical evaluation of the tarantula automatic fault-localization technique. In *20th IEEE/ACM International Conference on Automated Software Engineering*, ASE '05, pages 273–282, New York, NY, USA, 2005. ACM.

[8] Rui Abreu, Peter Zoeteweij, and Arjan J. C. van Gemund. An evaluation of similarity coefficients for software fault localization. In *Proceedings of the 12th Pacific Rim International Symposium on Dependable Computing*, Washington, DC, USA, 2006. IEEE.

[9] Jeongju Sohn and Shin Yoo. Fluccs: using code and change metrics to improve fault localization. In *26th ACM SIGSOFT International Symposium on Software Testing and Analysis*, page 273–283, New York, NY, USA, 2017. ACM.

[10] Andrea Arcuri and Lionel Briand. A hitchhiker's guide to statistical tests for assessing randomized algorithms in software engineering. *Softw. Test. Verif. Reliab.*, 24(3):219–250, may 2014.