Essay of <Evolving Human Competitive Spectra-Based Fault Localisation Techniques> by Shin Yoo

1. What is the software engineering problem authors are trying to solve?

Spectra-Based Fault Localisation (SBFL) is a class of fault localization techniques that uses program spectra, i.e., a summary of program's execution trace, to predict the likelihood of each program statement containing the fault. [1] [2] [3] Authors are aiming to assist debugging using SBFL which applies risk evaluation formulae. Risk evaluation formulae in SBFL are applied to program spectra and ranking statements according to the predicted risk.

Here, the performance of a SBFL depends mostly on the quality of the risk evaluation formula. However, designing risk evaluation formulae often requires intuition, since there is no guarantee that one formula is optimal for all classes of faults. So these process are done by human software engineers.

2. What are the technical contributions of the paper?

In this paper, authors tried to evolve risk evaluation formulae. To achieve that goal, they used Generic Programming (GP) approach. Their GP approach uses program spectra from four Unix utilities (flex, grep, gzip, and sed) from Software Infrastructure Repository [4] and the location information of 92 injected faults. It was the first evolutionary approach to generating risk evaluation formulae, and this empirical evaluation shows that GP-generated risk evaluation formulae can outperform those designed by human. This phenomena behaves widely, and even equally behaves or exceeds an existing formula that has been proven to be optimal against a specific program structure.

---

[1] Abreu, R., Zoeteweij, P., van Gemund, A.J.C.: On the accuracy of spectrum-based fault localization. In: Proceedings of the Testing: Academic and Industrial Conference Practice and Research Techniques - MUTATION, pp. 89–98. IEEE Computer Society (2007)

[2] Jones, J.A., Harrold, M.J.: Empirical evaluation of the tarantula automatic faultlocalization technique. In: Proceedings of the 20th International Conference on Automated Software Engineering (ASE 2005), pp. 273–282. ACM Press (2005)

[3] Jones, J.A., Harrold, M.J., Stasko, J.: Visualization of test information to assist fault localization. In: Proceedings of the 24th International Conference on Software Engineering, pp. 467–477. ACM, New York (2002)

[4] Do, H., Elbaum, S.G., Rothermel, G.: Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact. Empirical Software Engineering 10(4), 405–435 (2005)

3. Why do you like this work, or are interested in this work?

The problem spectrum for SBFL can be represented as a matrix of n rows and 4 columns. Each row (e_p, e_f, n_p, n_f) corresponds to individual statement of System Under Test (SUT). First two counters represent the number of times the corresponding program statement has been executed by tests, with pass and fail as a result respectively; the last two counters represent the number of times the corresponding program statement has not been executed by tests, with pass and fail as a result respectively. Then the risk evaluation formula uses these four counters with only five operators: +, -, *, / (where the result is 1 if denominator is equal to 0), and $\sqrt{}$ (where it takes the absolute value of the parameter). I surprised that SBFL uses these simple formulae (which may be realized as a metric), since they are just a simple algebraic formulae, and still these simple metrics are effective, even as an optimal methods. And then I like this work since it achieves a significant improvement up to 5.9 times, where the evolved formulae remains as algebraic and still have exponents not exceeding 4. I like these achievements which seems simple but effective.