Information and Software Technology 51 (2009) 1573–1585

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Is non-parametric hypothesis testing model robust for statistical fault localization? ☆,☆☆

Zhenyu Zhang [a], W.K. Chan [b,*], T.H. Tse [a], Peifeng Hu [c], Xinming Wang [d]

[a] Department of Computer Science, The University of Hong Kong, Pokfulam, Hong Kong
[b] Department of Computer Science, City University of Hong Kong, Tat Chee Avenue, Hong Kong
[c] China Merchants Bank, Central, Hong Kong
[d] Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong

ARTICLE INFO

Article history:
Available online 8 July 2009

Keywords:
Fault localization
Non-parametric
Hypothesis testing
Normality

ABSTRACT

Fault localization is one of the most difficult activities in software debugging. Many existing statistical fault-localization techniques estimate the fault positions of programs by comparing the program feature spectra between passed runs and failed runs. Some existing approaches develop estimation formulas based on mean values of the underlying program feature spectra and their distributions alike. Our previous work advocates the use of a non-parametric approach in estimation formulas to pinpoint fault-relevant positions. It is worthy of further study to resolve the two schools of thought by examining the fundamental, underlying properties of distributions related to fault localization. In particular, we ask: Can the feature spectra of program elements be safely considered as normal distributions so that parametric techniques can be soundly and powerfully applied? In this paper, we empirically investigate this question from the program predicate perspective. We conduct an experimental study based on the Siemens suite of programs. We examine the degree of normality on the distributions of evaluation biases of the predicates, and obtain three major results from the study. First, almost all examined distributions of evaluation biases are either normal or far from normal, but not in between. Second, the most fault-relevant predicates are less likely to exhibit normal distributions in terms of evaluation biases than other predicates. Our results show that normality is not common as far as evaluation bias can represent. Furthermore, the effectiveness of our non-parametric predicate-based fault-localization technique weakly correlates with the distributions of evaluation biases, making the technique robust to this type of uncertainty in the underlying program spectra.

1. Introduction

Software debugging is time-consuming and is often a bottleneck in software development process. It involves at least two crucial steps, namely *fault localization* and *fault correction*. Fault localization identifies the causes of abnormal behaviors of a faulty program. Fault correction modifies the faulty program or data structure to eliminate the effect of the identified faults.

A traditional fault-localization technique consists of setting breakpoints, re-executing the faulty program on the inputs, and examining the corresponding program states [13]. Recently, statistical fault-localization techniques [10,12,14–17] were proposed and reported to be promising. They locate faults by analyzing the statistics of dynamic program behaviors. A *failed run* is a program execution that reveals a failure, and a *passed run* is a program execution that reveals no failure. A statistical fault-localization technique locates a fault-relevant statement (or a faulty statement directly) by comparing the statistical information of program elements in these two kinds of runs. Such program elements can be statements [12] or predicates [14,15].

Because of their statistical nature, these techniques assume that there are statistically enough passed runs and failed runs to locate faults collectively. These techniques build underlying statistical behavior models for the aggregated execution data of selected program elements (which we call *features*), and search for program elements that strongly correlate with the observed program failures. For instance, predicate-based statistical techniques [14–17] locate those program predicates strongly related to faults. A *program predicate* is a Boolean expression about the property of a

☆ This research is supported in part by GRF grants of the Research Grants Council of Hong Kong (Project Nos. 123207 and 716507) and SRG grants of City University of Hong Kong (Project Nos. 7002324 and 7002464).
☆☆ A preliminary version of this paper was presented at the 8th International Conference on Quality Software (QSIC 2008) [8].
* Corresponding author. Tel.: +852 2788 9684; fax: +852 2788 8614.
E-mail addresses: zyzhang@cs.hku.hk (Z. Zhang), wkchan@cs.cityu.edu.hk (W.K. Chan), thtse@cs.hku.hk (T.H. Tse), pfhu@cs.hku.hk (P. Hu), rubin@cse.ust.hk (X. Wang).

system at some program location (such as a statement). CBI [14,15] checks the probability of a predicate to be evaluated to be true in all failed runs as well as the probability in all the runs (irrespectively of whether passed or failed), and measures the *increase* from the former to the latter. CBI uses this increase as a ranking score, which indicates how much the predicate is related to a fault. SOBER [16,17] defines *evaluation bias* to model the chance that a predicate is evaluated to be true in each run. More precisely, if P is a predicate and $\pi(P)$ is the probability that it is evaluated to be true in every run, then $\pi(P)$ is estimated to be $\frac{n_t}{n_t+n_f}$, where n_t is the number of times that P is evaluated to be true and n_f is the number of times that P is evaluated to be false. SOBER then evaluates the difference between the distributions of $\pi(P)$ between passed runs and failed runs, and deems that the larger the difference, the more will P be relevant to a fault.

As indicated in their models, CBI uses means and changes in mean values to estimate the fault relevance of a program predicate; SOBER applies the central limit theorem in statistics to measure the behavioral difference of a predicate between passed runs and failed runs. Typically, a mean value may reasonably represent a distribution if the variable of the distribution tends to cluster around the mean value. *Is it suitable to assume any known distribution in the program behaviors such as the evaluation biases of predicates?* We have conducted an initial study in our previous work [8] and found that evaluation biases may not form normal distributions. Our previous work also proposes to use a standard non-parametric hypothesis testing method to compare the program spectra of passed runs and those of failed runs. We have stipulated our model in the context of predicate-based statistical fault localization, and picked a form of the Mann–Whitney test to determine the degree of difference between the evaluation biases for passed runs and those for failed runs. The degree of difference in such a comparison is used as the ranking score, which indicates how much a predicate is related to a fault. Based on the ranking scores of the predicates, we reorder the predicates accordingly (predicates having higher values in ranking score are deemed to be more suspicious). The empirical results [8] on the Siemens suite show that our technique can be effective and outperforms CBI and SOBER in locating faults.

In view of the above-mentioned initial study, in this paper, we extend our investigation and ask a dual-sided question: Can the feature spectra of program elements be safely considered as normal distributions so that parametric fault-localization techniques can be soundly and powerfully applied? Alternatively, to what extent can such program spectra be regarded as normal distributions? If the answers to these questions are negative, we further ask the following question: Can the effectiveness of non-parametric fault-localization techniques be really decoupled from the distribution shape of the program spectra?

In this paper, we collect the evaluation biases of all the predicates from passed runs and those from failed runs, and conduct normality tests on them. By using standard statistical hypothesis testing, we successfully reject the assumption that normal distribution is commonly exhibited by evaluation biases of predicates. We further investigate the effect of such normality property for predicates on fault-localization techniques. The empirical results show that the effectiveness of our proposal for non-parametric fault localization [8] weakly correlates with the presumed normal distribution of evaluation biases.

The main contribution of the paper is fourfold: (i) It is the first investigation on the normality nature of the execution spectra. The empirical results show that normal distribution is not common for the evaluation biases of predicates. In particular, the results indicate that the chance of the distribution of the evaluation biases of fault-relevant predicates being normal is less likely than that of other predicates. (ii) Such a finding highlights a threat to the construct validity of any empirical study which is based on the assumption that the evaluation biases of predicates form normal distributions. (iii) It proposes a new metric *P*-score to measure the effectiveness of fault-localization techniques. (iv) It investigates the effect of normality for the evaluation biases of predicates on non-parametric fault-localization techniques. The empirical results show that the effectiveness of our non-parametric fault-localization technique weakly correlates with the normality of the underlying distribution of evaluation biases.

The remainder of the paper is organized as follows. Section 2 gives a motivating study. Section 3 revisits the background and sets the scene for the empirical study. Research questions are outlined in Section 4, followed by the experiment in Section 5. A literature review is given in Section 6. Finally, Section 7 concludes the paper.

2. Motivating study

In this section, we use one of the Siemens programs [5] to illustrate our important initial finding on the statistics of program behaviors. Fig. 1 shows the code excerpted from faulty version "v1" of the program "tot_info". In this code fragment, seven predicates are included, labeled as P_1 to P_7. The statement "goto ret1;" (labeled as E_1) is intentionally commented out by the Siemens researchers to simulate a statement omission fault. Locating such a kind of fault is often difficult even if the execution of a failed test case is traced step-by-step.

```
P1:     if ( rdf ≤ 0 || cdf ≤ 0 ) {
                info = -3.0;
                goto ret3;
        }
        :
P2:     for ( i = 0; i < r; ++i ) {
                double sum = 0.0;
P3:             for ( j = 0; j < c; ++j ) {
                        long k = x(i,j);
P4:                     if ( k < 0L ){
                                info = -2.0;
E1:                             /*goto ret1;*/
                        }
                        sum += (double)k;
                }
                N + = xi[i] = sum;
        }
P5:     if ( N ≤ 0.0 ) {
                info = -1.0;
                goto ret1;
        }
P6:     for ( j = 0; j < c; ++j ) {
                double sum = 0.0;
P7:             for ( i = 0; i < r; ++i )
                        sum += (double)x(i,j);
                xj[j] = sum;
        }
        :
        ret1:
```

Fig. 1. Excerpt from faulty version "v1" of program "tot_info" from the Siemens programs.