Similarity-based regression test case prioritization

Rongcun Wang
School of Computer Science and Technology
China University of Mining and Technology
Xuzhou, 221116, China
Email:rcwang@hust.edu.cn

Shujuan Jiang
School of Computer Science and Technology
China University of Mining and Technology
Xuzhou, 221116, China
Email:shjjiang@cumt.edu.cn

Deng Chen
Hubei Provincial Key Laboratory of Intelligent Robot
Wuhan Institute of Technology
Wuhan, 430073, China
Email:chendeng8899@hust.edu.cn

***Abstract*—With the continuous evolution of software systems, test suites often grow very large. Rerunning all test cases may be impractical in regression testing under limited resources. Coverage-based test case prioritization techniques have been proposed to improve the effectiveness of regression testing. The original test suite often contains some test cases which are designed for exercising production features or exceptional behaviors, rather than for code coverage. Therefore, coverage-based prioritization techniques do not always generate satisfactory results. In this context, we propose a global similarity-based regression test case prioritization approach. The approach reschedules the execution order of test cases based on the distances between pair-wise test cases. We designed and conducted empirical studies on four C programs to validate the effectiveness of our proposed approach. Moreover, we also empirically compared the effects of six similarity measures on the global similarity-based test case prioritization approach. Experimental results illustrate that the global similarity-based regression test case prioritization approach using Euclidean distance is the most effective. This study aims at providing practical guidelines for picking the appropriate similarity measures.**

***Keywords*—regression testing, test case prioritization, similarity measures**

I. Introduction

Regression testing is a very time-consuming and expensive activity. It accounts for more than 50% of the cost of software maintenance [1]. With the continuous evolution of software systems, test suites grow very large. The execution of the whole test suite consumes more time and resources. The high testing cost conflicts with constrained time and resources. Many test case prioritization techniques have been proposed to solve the contradiction [2], [3]. They aim at rescheduling the execution order of test cases to detect faults as early as possible.

Coverage-based prioritization techniques have gained wide attention. Most of these techniques resort to use greedy or metaheuristic search algorithms [5] to maximize the possible coverage. The original test suite often contains some test cases which are designed for exercising production features or exceptional behaviors, rather than for code coverage [6]. Therefore, coverage-based prioritization techniques do not always generate satisfactory results.

More recently, similarity-based test case prioritization techniques, incorporating clustering-based [7], ART-based [8] and other similarity-based prioritization [9], have been developed. Similarity-based test case prioritization techniques assume that test case diversity aids to detect more faults[15], [13]. Clustering-based prioritization techniques assume that the test cases within the same cluster have the same fault detection capability. Clustering-based prioritization techniques significantly depend on the number of clusters. Similarly, ART-based prioritization technique selects a test case to be prioritized from a subset of all remaining test cases. In other words, the method does not assure global test case diversity. Therefore, we propose a global similarity-based test case prioritization approach. Our approach rearranges the execution order of test cases from the global perspective. More importantly, our proposed approach is nonparametric.

We empirically evaluate the effects of six similarity measures, including Jaccard Index (JI), Gower-Legendre (GL), Soka-Sneath (SS), Euclidean distance (ED), Cosine similarity (CS) [10], and Proportional distance (PD) [11]metric, on the global similarity-based prioritization algorithm over 4 programs written in the C language. One way variance analysis (ANOVAs) [12] is used to analyze the statistical difference between different similarity measures. The results illustrate that Euclidean distance is superior to other similarity measures in terms of fault detection rate and standard deviation. The global similarity-based prioritization algorithm using Euclidean distance outperforms random prioritization and the additional function coverage prioritization. Our proposed approach is comparable to the best coverage-based prioritization techniques, i.e., the additional branch coverage prioritization technique. This study provides a practical guide for picking the optimal similarity measures.

The rest of this paper is organized as follows: Section II describes our approach. Experimental design and results analysis are presented in Section III and Section IV. The threats to validity are discussed in Section V. Section VI summarizes related works. The conclusions are described in Section VII.

II. Methodology

A. Overview

Figure 1 summarizes similarity-based test case prioritization techniques, which mainly include four steps:

(1) Instrumentation

DOI reference number: 10.18293/SEKE2015-115

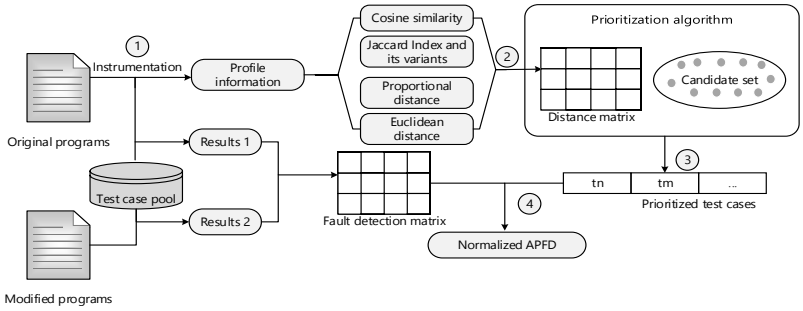


Fig. 1. Overview of similarity-based test case prioritization

With the dynamic instrumentation tool gcov, we collect execution profiles and construct branch coverage vectors.

(2) Distance calculation

The distance between pair-wise test cases is calculated by a certain distance measure.

(3) Test case prioritization

Test cases are prioritized based on the distance between pair-wise test cases.

(4) Evaluation

The fault detection effectiveness of a prioritized test suite is evaluated based on the relation between faults and test cases.

B. Similarity measures

All branches covered by a test case, can be represented as a binary branch coverage vector V: $< v_1, v_2, \ldots, v_n >$, where v_i is 0 if the ith branch is covered, otherwise 1. Similarly, the vector can also be implemented with numeric entries, i.e., v_i represents the number of times that ith branch is executed.

1) Cosine similarity: The binary branch coverage vectors generated by executing test case t_1 and t_2 are $X :< x_1, x_2, \ldots, x_n >$ and $Y :< y_1, y_2, \ldots, y_n >$, respectively. The similarity between t_1 and t_2 is defined as follows:

$$s(t_1, t_2) = \frac{X^t \cdot Y}{\|X\|\|Y\|}, \tag{1}$$

where X^t is a transposition of vector X, and $\|X\|$ is the Euclidean norm of vector X. Similarly, $\|Y\|$ is the Euclidean norm of vector Y. In essence, s is the cosine of the angle between X and Y. For Cosine similarity, the corresponding dissimilarity is defined as $d(t_1, t_2)$=1$-s(t_1, t_2)$.

2) Euclidean distance: The Euclidean distance between test case t_1 and t_2 is defined as follows:

$$d(t_1, t_2) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}. \tag{2}$$

3) Proportional distance: Let $P :< p_1, p_2, \ldots, p_n >$ and $Q :< q_1, q_2, \ldots, q_n >$ stand for two coverage vectors implemented with numeric entries by executing t_1 and t_2. The proportional distance between t_1 and t_2 is defined as follows:

$$d(t_1, t_2) = \sqrt{\sum_{i=1}^{n} (\frac{|p_i - q_i|}{max_i - min_i})^2}, \tag{3}$$

where max_i and min_i represent the maximum and minimum number of times that the ith branch is executed over all tests in the test suite, respectively. When the difference between max_i and min_i is equal to 0, the corresponding term is also equal to 0.

4) Jaccard Index and its variants: The similarity between t_1 and t_2 based on Jaccard Index and its variants is defined as follows:

$$s(t_1, t_2) = \frac{X \cdot Y}{X \cdot Y + w(\|X\|^2 + \|Y\|^2 - 2(X \cdot Y))}, \tag{4}$$

where $X \cdot Y$ is the inner product of X and Y. When w is equal to 1, the above formula is called Jaccard Index. If w=2 and 1/2, this formula is called Soka-Sneath measure and Gower-Legendre measure, respectively. For Jaccard Index and its variants, the corresponding distance is $d(t_1, t_2) = 1 - s(t_1, t_2)$.

C. Prioritization Algorithm

The global similarity-based test case prioritization algorithm (GSTCP) selects a test case from all not yet prioritized test cases, rather than a candidate set of them [8]. Its pseudo-code is described in Algorithm 1.

This algorithm randomly selects a test case t_k from the original test suite T, deletes it from T, and adds it to the tail of a prioritized sequence P. The distance from test case t in T to t_k is viewed as the minimal set distance from t to P. Function $dist$ is responsible for the calculation of the distance between two test cases. The process of test case prioritization mainly includes two steps. The first step is to seek a test case t_i in T that has the maximum distance from the last test case in P (Line 9-Line 13). Test case t_i is added to the tail of P and deleted from T. The second step is to update the minimal set distance from test case t in T to P by comparing the distance from t to t_i and the set distance from t to P before adding t_i to P (Line 15-Line 19).

In the process of prioritizing test cases, this algorithm needs to calculate the distance between remaining test cases in T and the last test case in P. The number of times that this algorithm calculates the distance gradually decreases from $n-1$ to 1. The time complexity and space complexity of GSTCP are $O(n^2)$ and $O(n)$, respectively. Compared with the ART-based prioritization algorithm, GSTCP significantly reduces time complexity.

Algorithm 1: GSTCP

Input : A test suite $T : \{t_0, t_1, \cdots, t_{n-1}\}$
Output: A prioritized sequence $P : \langle p_0, p_1, \cdots, p_{n-1}\rangle$

$P \leftarrow \emptyset$; *double* $dist\_min[n]$;
randomly select test case t_k from T;
$max \leftarrow k$; $T \leftarrow T \backslash \{t_{max}\}$; $P \leftarrow \langle t_{max}\rangle$;
for $i \leftarrow 0$ *to* $n-1$ **do**
  $dist\_min[i] = dist(t_j, t_{max})$;
end
repeat
  $max\_dist = 0.0$;
  for $i \leftarrow 0$ *to* $n-1$ **do**
    if $dist\_min[i] > max\_dist$ **and** $t_i \in T$ **then**
      $max\_dist = dist\_min[i]$; $max \leftarrow i$;
    end
  end
  add t_{max} to the tail of P; $T \leftarrow T \backslash \{t_{max}\}$;
  for $i \leftarrow 0$ *to* $n-1$ **do**
    if $dist\_min[i] > dist(t_i, t_{max})$ **and** $t_i \in T$ **then**
      $dist\_min[i] = dist(t_i, t_{max})$;
    end
  end
until T *is empty*;
return P

III. Empirical Study

A. Research Questions

In the empirical study, we address the following two specific research questions.

RQ1: Do different similarity measures have significant effects on the global similarity-based test case prioritization algorithm?

RQ2: Can the most effective global similarity-based prioritization technique be as effective as coverage-based prioritization techniques?

B. Subject Program

Our experiments were conducted over four subject programs [1], incorporating 2 small-sized Siemens programs and 2 medium-sized UNIX utilities. Descriptive information about the selected programs is presented in Table I, where the lines of codes are calculated by the tool "**SLOCCount**" [2].

TABLE I
Descriptive information for subject programs

Program Name	Fault Versions	Lines of Code	Test Suite Size
print_tokens	7	341-343	4130
print_tokens2	10	350-355	4115
make	33	12609-17153	1043
sed	18	4711-9204	370

[1]http://sir.unl.edu/php/showfiles.php
[2]http://www.dwheeler.com/sloccount

We eliminated the fault versions whose faults cannot be detected by any test case. Likewise, if a fault can be detected by more than 20% of test cases, the fault is also excluded.

C. Evaluation Metric

The average percentage of faults detected (*APFD*) [18] is commonly used to evaluate the effectiveness of a prioritization technique implemented on a whole test suite. Let T be a test suite containing n test cases and let F be a set of m faults exposed by T. *APFD* is defined as follows:

$$APFD = 1 - \frac{TF_1 + TF_2 + \ldots + TF_m}{nm} + \frac{1}{2n}, \quad (5)$$

where TF_i is the first test case in a prioritized test suite that detects fault i.

The application of *APFD* assumes that the whole test suite can be run and find all of the faults. Only a part of test suite is often executed in regression testing [14], [19]. In this sense, *APFD* is unsuitable to measure the effectiveness of a prioritization technique implemented on a part of test suite. Therefore, we use a metric, called Normalized *APFD* [20], which utilizes information on both fault finding and time of detection. Let TF_i be the first test case in the prioritized and reduced test suite T' of T that detects fault i. Let n' be the size of T'. Normalized *APFD* is defined as follows:

$$NAPFD = p - \frac{TF_1 + TF_2 + \ldots + TF_m}{n'm} + \frac{p}{2n'}, \quad (6)$$

where p represents the quotient of the number of faults detected by the prioritized and reduced test suite divided by the number of faults detected in the whole test suite. If a fault i is never detected by T', TF_i is set 0.

Since all prioritization algorithms in this study have the nature of randomness, we repeated 100 times for every prioritization algorithm with different random seeds. The mean $NAPFD$ values of every sample were calculated.

IV. Experimental Results

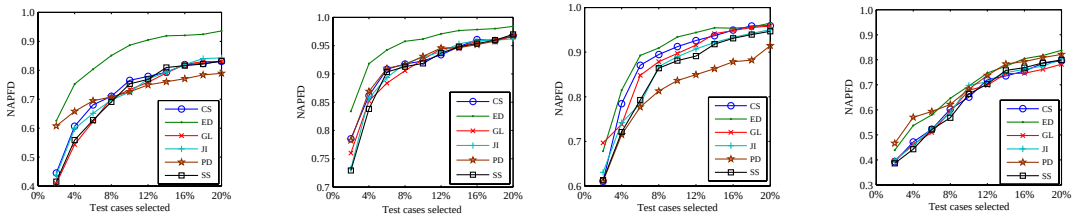
We took 10 samples starting from 2% to 20% with the increment of 2% for every program so as to look deeper into the statistical difference between different similarity measures.

A. Experiment 1

1) Experimental Results: The experimental results are shown in Figure 2, where the x-axis of each graph indicates the number of tests selected; while the y-axis shows the mean *NAPFD* of each similarity measure.

From Figure 2, Cosine similarity performed better or as good as Jaccard Index in terms of *NAPFD*. Likewise, Euclidean distance also performed consistently better than Jaccard Index for smaller samples. Particularly, Euclidean distance outperformed other similarity measures for all samples over the program *print_tokens* and *print_tokens2*. The results of Jaccard Index were very close to those of its variants.

Table II summarizes the standard deviations for all similarity measures. The minimal standard deviations are reported highlighting cells in gray shade. From Table II, Euclidean distance



(a) print_tokens (b) print_tokens2 (c) schedule (d) schedule2

Fig. 2. NAPFDs of six similarity measures using the global similarity-based test case prioritization algorithm

TABLE II
THE STANDARD DEVIATIONS OF DIFFERENT SIMILARITY MEASURES BASED ON THE GLOBAL SIMILARITY PRIORITIZATION ALGORITHM

Program	SM	Sampling proportion									
		2%	4%	6%	8%	10%	12%	14%	16%	18%	20%
print_tokens	CS	0.145	0.125	0.100	0.116	0.084	0.086	0.065	0.070	0.070	0.079
	ED	0.098	0.085	0.083	0.062	0.055	0.045	0.043	0.037	0.039	0.035
	PD	0.121	0.109	0.113	0.109	0.110	0.109	0.125	0.106	0.108	0.109
	JI	0.136	0.112	0.108	0.093	0.099	0.086	0.068	0.067	0.078	0.074
	GL	0.117	0.137	0.105	0.108	0.094	0.083	0.085	0.077	0.075	0.065
	SS	0.145	0.122	0.111	0.094	0.099	0.087	0.089	0.077	0.073	0.069
print_tokens2	CS	0.099	0.065	0.050	0.050	0.036	0.034	0.029	0.020	0.024	0.019
	ED	0.066	0.036	0.027	0.025	0.015	0.013	0.012	0.009	0.008	0.007
	PD	0.139	0.079	0.059	0.060	0.051	0.032	0.039	0.039	0.029	0.031
	JI	0.109	0.070	0.050	0.040	0.030	0.035	0.023	0.023	0.022	0.023
	GL	0.094	0.058	0.059	0.051	0.039	0.029	0.030	0.021	0.016	0.015
	SS	0.097	0.080	0.048	0.044	0.044	0.033	0.028	0.026	0.024	0.017
make	CS	0.277	0.180	0.099	0.084	0.057	0.055	0.045	0.033	0.031	0.024
	ED	0.167	0.133	0.079	0.058	0.046	0.051	0.036	0.035	0.046	0.025
	PD	0.280	0.265	0.220	0.241	0.233	0.177	0.158	0.196	0.156	0.116
	JI	0.285	0.239	0.240	0.132	0.172	0.122	0.118	0.145	0.049	0.074
	GL	0.213	0.204	0.178	0.137	0.138	0.193	0.053	0.096	0.049	0.068
	SS	0.269	0.253	0.189	0.124	0.131	0.152	0.136	0.113	0.076	0.070
sed	CS	0.055	0.064	0.056	0.055	0.053	0.054	0.048	0.040	0.038	0.042
	ED	0.038	0.042	0.040	0.031	0.042	0.033	0.031	0.035	0.026	0.032
	PD	0.062	0.540	0.054	0.041	0.043	0.048	0.039	0.406	0.034	0.043
	JI	0.062	0.052	0.055	0.053	0.041	0.045	0.048	0.051	0.043	0.041
	GL	0.058	0.056	0.043	0.060	0.042	0.052	0.051	0.052	0.047	0.041
	SS	0.051	0.061	0.050	0.053	0.051	0.058	0.038	0.043	0.044	0.040

generated smaller standard deviations than other similarity measures with the same sampling proportion. This means that the application of Euclidean distance reduces the risk of missing faults in practice.

2) Experimental Analysis: We conducted ANOVAs to verify whether different similarity measures generate the significant effects on the global similarity-based prioritization algorithm at 5% significance level. Having executed ANOVAs, we find that the p-value was less than 0.05 for every sample across every subject program. In other words, different measures have significant effects on the global similarity-based prioritization algorithm. We further conducted multiple comparisons so as to seek which similarity measures can produce higher *NAPFD*s.

Table III shows the results of multiple comparisons between pair-wise similarity measures, where each element (m, n) denotes the "win/tie/loss" (win: the number of times that the m-th row measure performs significantly better than the n-th column measure; tie: the number of times that there is no significant difference between the m-th row measure and the n-th column measure; loss: the number of times that the m-th row measure performs significantly worse than the n-th column measure) value.

TABLE III
THE SUMMARIZED RESULTS OF MULTIPLE COMPARISONS BETWEEN PAIR-WISE SIMILARITY MEASURES OVER 40 DIFFERENT EXPERIMENTAL SETTINGS (4 PROGRAMS × 10 SAMPLING RATES)

	CS	ED	GL	JI	PD
ED	(27/12/1)	-	-	-	-
GL	(1/31/8)	(1/9/30)	-	-	-
JI	(1/32/7)	(1/8/31)	(2/34/4)	-	-
PD	(6/20/14)	(2/9/29)	(8/19/13)	(9/19/12)	-
SS	(0/32/8)	(0/8/32)	(1/36/3)	(3/35/2)	(12/20/8)

From Table III, Euclidean distance performed better than other similarity measures. In the best case, Euclidean distance outperformed Soka-Sneath on 32 settings, while Soka-Sneath did not outperformed it on any setting. In the worst case, Euclidean distance outperformed Cosine similarity on 27 settings,

while Cosine similarity outperformed it only on 1 settings. The results generated by Jaccard Index were very close to those of its variants. The plausible explanation is that the topologies of Jaccard Index and its variants are very similar.

B. Experiment 2

1) Experimental Results: We mainly compared the global similarity-based prioritization using Euclidean distance with random prioritization (RP), the additional function coverage prioritization (AF), the additional statement coverage prioritization (AS), and the additional branch coverage prioritization (AB) [4]. Figure 3 shows the $NAPFD$s of different prioritization techniques. Table IV summarizes the standard deviations for different prioritization techniques.

TABLE V
THE SUMMARIZED RESULTS OF MULTIPLE COMPARISONS BETWEEN DIFFERENT PRIORITIZATION TECHNIQUES OVER 40 DIFFERENT EXPERIMENTAL SETTINGS (4 PROGRAMS $\times$ 10 SAMPLING RATES)

	RP	AF	AS	AB
AF	(24/16/0)	-	-	-
AS	(40/0/0)	(22/13/5)	-	-
AB	(40/1/0)	(25/9/6)	(14/26/0)	-
ED	(39/1/0)	(23/14/3)	(10/22/8)	(10/18/12)

2) Experimental Analysis: Table V shows the results of multiple comparisons between different prioritization algorithms. Our approach performed significantly better than RP, and AF in terms of $NAPFD$. In general, the global similarity-based prioritization algorithm using Euclidean distance was equal to AS. ED outperformed AS on 10 settings, while AS outperformed it on 8 settings. Particularly, our proposed approach was comparable to the best coverage-based prioritization algorithm, i.e., AB. ED outperformed AB on 10 settings, while AB outperformed it on 12 settings.

Additionally, our proposed approach is superior to other prioritization techniques with respect to the standard deviation of $NAPFD$. This means that it can yield more reliable results. In summary, the global similarity-based prioritization algorithm using Euclidean distance is very promising as a candidate for regression test case prioritization.

V. THREATS TO VALIDITY

This section discusses the potential threats to validity.

Threats to internal validity are from the effects of instrumentation and the sampling rates. Therefore, we collected profile information using a professional tool gcov. Additionally, we took 10 samples for every subject program and reported the mean value of every sample.

Threats to external validity for this study concern the representativeness of the programs utilized. Although the four programs are from different domains with different characteristics, they may not be representative of all other programs. The threat can be addressed by selecting larger scale and more representative industrial programs in future work.

Threats to construct validity may be affected by the evaluation metric. As previous studies, *NAPFD* is used to evaluate the effectiveness of different prioritization techniques. This may be insufficient for evaluating the effectiveness of different combinations. There may be other metrics which are more relevant to this study.

VI. RELATED WORK

Ledru *et al*. used string distances for test case prioritization [9]. They empirically evaluated the effects of string edit distances on test case prioritization. Test cases were ordered before executing them. Similarly, Hemmati *et al*. introduced a family of similarity-based test case selection techniques for model-based testing [13], [15]. The above two methods were applied to black-box testing. On the contrary, our approach uses dynamic profile information generated by executing test cases to reschedule the execution order.

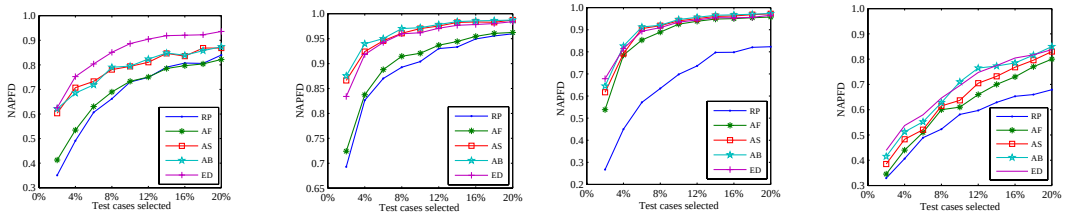
Clustering algorithms have been also applied to test case prioritization. Yoo *et al*. [7] combined clustering with expert knowledge to achieve scalable prioritization. The process of prioritization depended on human efforts, which is different from ours. Dickinson *et al*. [11] presented distribution-based filtering and prioritizing techniques incorporating sampling methods. Clustering-based prioritization techniques significantly depend on a parameter, i.e., the number of clusters. On the contrary, our approach is nonparametric, i.e., it dose not depend on the number of clusters.

More recently, similarity-based algorithms have been applied to regression test case prioritization. Krishna *et al*. [16] used Levenshtein distance to similarity-based test prioritization. Jiang *et al*. [8] proposed a new family of coverage-based ART techniques and used Jaccard Index to measure the distance between pair-wise test cases. Fang *et al*. [17] introduced several similarity-based test case prioritization techniques based on the edit distances of ordered sequences. However, we represented the execution profile that a test case exercised into a branch coverage vector, rather than an ordered sequence [17]. Differently from Jiang *et al*. [8] and Krishna *et al*. [16], where only one similarity measure was used, we empirically evaluated six similarity measures.

VII. CONCLUSIONS

In this paper, we proposed a global similarity-based test case prioritization algorithm based on the comparison of similarity measures between test cases in a given test case suite. By ANOVAs, we find that different measures have significant effects on the global similarity-based test case prioritization algorithm. Particularly, Euclidean distance perform better than other five similarity measures in terms of *NAPFD* and standard deviation. Therefore, we recommend Euclidean distance. Moreover, the global similarity-based prioritization algorithm using Euclidean distance outperforms random prioritization and the additional function coverage prioritization with respect to $NAPFD$. It is comparable to the additional branch coverage. Since our proposed approach generates smaller standard deviation, it can provide more reliable results.

For future work, much more similarity measures will be empirically evaluated. Least but not last, we will collect more



(a) print_tokens (b) print_tokens2 (c) schedule (d) schedule2

Fig. 3. NAPFDs of different test case prioritization techniques

TABLE IV
THE STANDARD DEVIATIONS OF DIFFERENT PRIORITIZATION ALGORITHMS

Program	TCP	Sampling proportion									
		2%	4%	6%	8%	10%	12%	14%	16%	18%	20%
print_tokens	RP	0.155	0.149	0.135	0.126	0.113	0.105	0.095	0.089	0.087	0.084
	AF	0.141	0.136	0.120	0.101	0.102	0.085	0.080	0.087	0.077	0.069
	AS	0.128	0.114	0.097	0.084	0.092	0.085	0.074	0.076	0.077	0.062
	AB	0.112	0.106	0.090	0.087	0.085	0.077	0.079	0.076	0.069	0.063
	ED	0.098	0.085	0.083	0.062	0.055	0.045	0.043	0.037	0.039	0.035
print_tokens2	RP	0.138	0.123	0.119	0.095	0.087	0.086	0.084	0.073	0.074	0.065
	AF	0.130	0.107	0.098	0.083	0.074	0.077	0.052	0.048	0.044	0.039
	AS	0.085	0.079	0.072	0.070	0.063	0.057	0.056	0.041	0.031	0.009
	AB	0.075	0.068	0.054	0.057	0.049	0.036	0.030	0.021	0.020	0.008
	ED	0.066	0.036	0.027	0.025	0.015	0.013	0.012	0.009	0.008	0.007
make	RP	0.260	0.225	0.194	0.156	0.121	0.116	0.105	0.096	0.073	0.082
	AF	0.236	0.173	0.132	0.107	0.091	0.079	0.052	0.036	0.025	0.022
	AS	0.190	0.141	0.093	0.080	0.055	0.035	0.049	0.026	0.013	0.011
	AB	0.172	0.152	0.095	0.072	0.053	0.034	0.038	0.039	0.012	0.015
	ED	0.167	0.133	0.079	0.058	0.046	0.051	0.036	0.032	0.046	0.025
sed	RP	0.126	0.113	0.095	0.097	0.073	0.052	0.061	0.045	0.038	0.029
	AF	0.126	0.090	0.072	0.063	0.044	0.040	0.032	0.030	0.021	0.020
	AS	0.086	0.073	0.069	0.043	0.024	0.022	0.035	0.017	0.025	0.021
	AB	0.072	0.065	0.052	0.030	0.027	0.021	0.022	0.018	0.015	0.017
	ED	0.038	0.042	0.043	0.040	0.031	0.042	0.033	0.031	0.026	0.032

coverage information test cases exercised and construct differently structural profiles. The effects of differently structural profiles on test case prioritization will also be empirically evaluated.

REFERENCES

[1] M. J. Harrold and A. Orso. Retesting software during development and maintenance. In *FOSM'08*, pp. 99-108, 2008.

[2] W. E. Wong, J. R. Horgan, S. London, and H. Agrawal. A stduy of effective regression testing in practice. In *ISSRE'97*, pp. 230-238, 1997.

[3] G. Rothermel, R. Untch, C. Chu, and M. J. Harrold. Test case prioritization: an empirical study. In *ICSM'99*, pp. 179-188, 1999.

[4] S. Elbaum, A. G. Malishevsky, and G. Rothermel. Test case prioritization: a family of empirical stdudies. *IEEE Trans. Softw. Eng.*, 28(2):159-182, 2002.

[5] Z. Li, M. Harman, R. M. Hierons. Search algorithms for regresion test prioritization. *IEEE Trans. Softw. Eng.*, 33(4):225-237, 2003.

[6] G. Rothermel, M. J. Harrold, J. V. Ronne, and C. Hong. Empirical studies of test suite reduction. *Softw. Test. Verif. Rel.*, 12: 219-249, 2002.

[7] S. Yoo, M. Harman, P. Tonella, and A. Susi. Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge. In *ISSTA'09*, pp. 201-212, 2009.

[8] B. Jiang, Z. Y. Zhang, W. K. Chan, and T. H. Tse. Adaptive random test case prioritization. In *ASE'09*, pp. 233-244, 2009.

[9] Y. Ledru, A. Petrenko, and S. Borody. Prioritizing test cases with string distances. *Automat. Softw. Eng.*, 19(1):65-95, 2012.

[10] R. Xu and D. Wunsch. A surven of clustering algorithms. *IEEE Trans. Neural Networ.*, 16(3):645-678, 2005.

[11] W. Dickinson, D. Leon, and A. Podgurski. Finding failures by cluster analysis of execution profiles. In *ICSE'01*, pp. 339-348, 2001.

[12] H. Scheffe. *The Analysis of Varianc*. John Wiley and Sons, New York, 1993.

[13] H. Hemmati, A. Arcuri, and L. Briand. Achieving scalable model-based testing through test case diversity. *ACM Trans. Softw. Eng. Methodol.*, 22(1):1-42, 2013.

[14] J. Jones, and M. Harrold. Test suite reduction and prioritization for modified condition/decision coverage. *IEEE Trans. Softw. Eng.*, 29(3):193-209, 2003.

[15] H. Hemmati and L. Briand. An industrial investigation of similarity measures for model-based test case selection. In *ISSRE'10*, pp. 141-150, 2010.

[16] M. Krishna, M. Koyuturk, A. Grama, and S. Jagannathan. PHALANX: A graph-theoretic framework for test case prioritization. In *SAC'08*, pp. 667-673, 2008.

[17] C. Fang, Z. Chen, K. Wu, Z. Zhao. Similarity-based test case prioritization using ordered sequences of program entities. *Softw. Quality J.*, 22(2):335-361, 2014.

[18] G. Rothermel, R. Untchm, C. Chu, and M. J. Harrold. Prioritizing test cases for regression testing. *IEEE Trans. Softw. Eng.*, 27(10):929-948, 2001.

[19] A. Smith, J. Geiger, G. Kapfhammer, and M. Soffa. Test suite reduction and prioritization with call trees. In *ASE'07*, pp. 539-540, 2007.

[20] X. Qu, M. B. Cohen, and K. M. Woolf. Combinatorial interaction regression testing: a study of test case generation and prioritization. In *ICSM'07*, pp. 255-264, 2007.