How to Do Tie-breaking in Prioritization of Interaction Test Suites?

Rubing Huang*, Jinfu Chen
School of Computer Science and Telecommunication Engineering
Jiangsu University
Zhenjiang, Jiangsu 212013, P.R.China

Rongcun Wang, Deng Chen
School of Computer Science and Technology
Huazhong University of Science and Technology
Wuhan, Hubei 430074, P.R.China

Abstract—The prioritization of interaction test suites has received more attention in the field of combinatorial interaction testing, especially when testing resources are limited to allow the part of combinatorial test cases to be executed. Many strategies have been proposed to prioritize interaction test suites according to different evaluation measures. However, most of these strategies may face a challenge to choose more than one “best” candidate with the largest evaluation measure value. In this case, there is a tie among all “best” candidates. How to do tie-breaking? Intuitively speaking, random tie-breaking could be a reasonable choice, which has also been applied to many research papers. In this paper, we investigate different tie-breaking techniques including random tie-breaking, first-element tie-breaking, last-element tie-breaking, higher-strength tie-breaking, and lower-strength tie-breaking, and also conduct experiments on a well-known prioritization strategy of interaction test suites, namely interaction coverage based prioritization, in order to present a guideline of choosing tie-breaking techniques for testers in practical testing. The experimental results show that although no tie-breaking technique always performs best, in many cases random tie-breaking and last-element tie-breaking have best performance, so that they would be best choices for testers in the prioritization of interaction test suites.

Keywords—Combinatorial interaction testing, interaction test suite, test case prioritization, tie-breaking, guideline

I. INTRODUCTION

Combinatorial interaction testing (CIT) [1] aims at generating an interaction test suite [1], in order to identify failures that are caused by parameter interactions. Intuitively speaking, combinatorial interaction testing presents a tradeoff between testing effectiveness and testing efficiency, because it only focuses on interaction coverage of fixed strengths (the levels of interaction among parameters) rather than that of all strengths.

When an interaction test suite $T$ has been already constructed by CIT, traditional CIT would directly run $T$ but does not consider the execution order of test cases in $T$. However, due to limited test resources in practice, only part of test cases in $T$ could be executed. In such case, the execution order of test cases in $T$ would be critical for the whole testing process, because a well-prioritized order of test case execution may be able to identify failures earlier, and thus enable fault characterization, diagnosis and revision as earlier as possible [1]. The process of determining the order of test cases in $T$ is generally called test case prioritization. A prioritized interaction test suite is also called an interaction test sequence. As shown in Figure 1, the testing process involved in the dashed box is the traditional CIT, and an enhanced testing process adds test case prioritization between test suite construction and test suite execution in the traditional CIT [2].

To date, many strategies have been proposed to guide the prioritization of interaction test suite according to different evaluation measures, for example interaction coverage based prioritization [2–8] and incremental interaction coverage based prioritization [9, 10]. However, most of prioritization strategies may face a challenge $^1$: During the prioritization process, there may exist more than one “best” candidate such that they have the same largest evaluation measure value. In this case, there is a tie among all “best” candidates. How to do tie-breaking? In this paper, we investigate different tie-breaking techniques such as random tie-breaking [5, 7, 10], first-element tie-breaking, last-element tie-breaking, higher-strength tie-breaking, and lower-strength tie-breaking, and also conduct experiments on the ICBP prioritization algorithm to analyze the effectiveness of tie-breaking techniques, so as to present a guideline of choosing tie-breaking techniques for testers in practical testing. The experimental results indicate that although there is no best tie-breaking technique, in many cases random tie-breaking and last-element tie-breaking have best performance, so that they would be best

$^1$Some prioritization strategies do not face this challenge, for example random test case prioritization [6], because it only prioritizes an interaction test suite at a random manner. In this paper, therefore, we assume that our study focus on the prioritization strategies with such challenge.

![Figure 1. Traditional CIT with test case prioritization [2].](image-url)
choices for testers in the prioritization of interaction test suites.

The remaining part of this paper is organized as follows: Section 2 simply introduces some background information about CIT and test case prioritization. Section 3 presents some related work about our study. Section 4 investigates various methods that can be used to prioritize arbitrary interaction test suites. Section 5 conducts some empirical studies to evaluate different prioritization methods investigated in Section 4. Section 6 analyzes the application range of each method. Finally, Section 7 concludes and discusses future work.

II. BACKGROUND

In this section, some background information will be described, including CIT, interaction test suite, and test case prioritization.

A. CIT and Interaction Test Suite

CIT aims at covering some specific combinations of parametric values by using an interaction test suite with the small number of test cases, so as to detect failures that are triggered by parameter interactions.

To clearly describe some notions, we first model the test object in CIT. Assume the SUT has \( k \) parameters that constitute a parameter set \( P = \{p_1, p_2, \ldots, p_k\} \), and each parameter \( p_i \) has some valid values or levels from the finite set \( V_i \) (\( i = 1, 2, \ldots, k \)). In practice, parameters may represent any factors that influence the performance of the SUT, such as components, configuration options, user inputs, etc. Suppose \( \mathcal{C} \) be the set of constraints on combinations of parameter values.

**Definition 1 (Test profile).** A test profile, denoted as \( TP(k, |V_1||V_2|\cdots|V_k|; \mathcal{C}) \), is the model of the SUT, including \( k \) parameters, \( |V_i| \) parameter values for each \( i \)-th parameter, and value combination constraints \( \mathcal{C} \).

Unless specifically stated, the definitions illustrated in this paper are based on a \( TP(k, |V_1||V_2|\cdots|V_k|; \mathcal{C}) \).

**Definition 2 (Covering array).** A covering array (CA), denoted \( CA(N; \tau, k, |V_1||V_2|\cdots|V_k|) \), is an \( N \times k \) matrix, which satisfies the following properties: (1) each column \( i \) (\( 1 \leq i \leq k \)) contains elements only from the set \( V_i \); and (2) the rows of each \( N \times \tau \) sub-matrix cover all possible \( \tau \)-tuples (referred to as \( \tau \)-wise value combinations or \( \tau \)-wise value schemas [1]) from the \( \tau \)-columns at least once.

Here, \( \tau \) is called strength. Since the strength of a covering array is fixed, a covering array at strength \( \tau \) is also usually called a \( \tau \)-wise covering array.

In a covering array, each row represents a test case; while each column represents a parameter. Generally speaking, in the field of combinatorial interaction testing, a covering array represents an interaction test suite. In this paper, therefore, we assume that a covering array is equivalent to an interaction test suite.

B. Test Case Prioritization

Test case prioritization seeks to schedule test cases so that those with the highest priority, according to some criterion, are executed earlier in testing than lower priority test cases. When testing resources are limited or insufficient to execute all test cases in a test suite, a well-designed execution order of test cases seems especially significant. A prioritized test suite is generally called a test sequence. The problem of test case prioritization is defined as follows [11].

**Definition 5 (Test case prioritization).** Given a tuple \((T, \Omega, f)\), where \( T \) is a test suite, \( \Omega \) is the set of all possible permutations of \( T \), and \( f \) is a function from \( \Omega \) to real numbers, the test case prioritization problem is to find a test sequence \( S \in \Omega \) such that:

\[
(\forall S')(S' \in \Omega)(S' \neq S)[f(S) \geq f(S')].
\]

According to Rothermel’s investigations [11], there are many possible goals of prioritization (that is, different functions of \( f \)). For example, a well-known function, namely average percentage of faults detected (APFD), is related to fault detection, which is widely used in regression testing. The APFD function needs to obtain the fault-detection capability of each executed test case.

III. Tie-Breaking Techniques in Prioritization of Interaction Test Suites

In this section, we investigate different tie-breaking techniques used in the prioritization of interaction test suites. In order to describe techniques clearly, we apply different tie-breaking techniques to a well-known prioritization strategy of interaction test suites, namely interaction coverage based prioritization (in short ICBP). Figure 2 presents the detailed algorithm of ICBP, from which the “best” element is chosen from candidates in \( T \) as the next test case in \( S \) such that it covers the largest number of \( \tau \)-wise value combinations that have not yet covered by \( S \).

Consider an interaction test suite \( T \) (its original test sequence \( T' \) is obtained according to its generation order). In order to describe the prioritization strategy for algorithm ICBP, that is, ICBP uses the \( \tau \)-wise interaction coverage, or uncovered \( \tau \)-wise value combinations distance (in short UVC\( \tau \)) [12] to guide the prioritization, and an interaction test sequence of \( T \)

```
Input: A \( \tau \)-wise CA, denoted as \( T \).
Output: An interaction test sequence \( S \) of \( T \).
1. Initialize \( S \);
2. while (\( T \) is not empty)
   3. Select the “best” element \( e \) from \( T \).
   4. Remove \( e \) from \( T \).
   5. Insert \( e \) into \( S \);
6. end_while
7. return \( S \).
```

Figure 2. The detailed algorithm description of ICBP.
prioritized by ICBP. We will describe different tie-breaking techniques as follows.

A. Random Tie-breaking

Obviously, random tie-breaking is an intuitive technique, because it randomly breaks the tie among all “best” candidates. For example, two combinatorial test cases $tc_1$ and $tc_2$, in $T$, have the same $UVCD_2$ value, ICBP will randomly choose one from $tc_1$ and $tc_2$ as the next test case in $S$. In other words, $tc_1$ and $tc_2$ have the same probability to be selected.

B. First-element Tie-breaking

The first-element tie-breaking technique denotes that among all “best” candidates, ICBP would select a candidate as the next test case such that it occurs in the original test sequence $T'$ first.

C. Last-element Tie-breaking

Similar to the first-element tie-breaking technique, Last-element Tie-breaking chooses the element as the next test case such that is occurs in the original test sequence $T'$ lastly when facing the challenge of more than one “best” candidate.

As we know, ICBP uses the fixed strength $\tau$ to guide the prioritization of interaction test suites. When there are two candidates $tc_1$ and $tc_2$ of which have the largest $UVCD_2$ value, the strength $\tau$ is unavailable to distinguish $tc_1$ and $tc_2$. In this case, it is reasonable to consider other strength to assist ICBP in choosing the next test case. Therefore, we consider the following two cases: (1) the tie-breaking technique using the strength higher than $\tau$, namely Higher-strength tie-breaking; and (2) the tie-breaking technique using the strength lower than $\tau$, namely lower-interaction-coverage tie-breaking. We will describe them in turn.

D. Higher-strength Tie-breaking

When the strength $\tau$ is unavailable to assist ICBP in selecting the next test case from the “best” candidates, the Higher-strength tie-breaking technique uses the strength $(\tau + 1)$ for ICBP to further calculate each “best” candidate, and then choose one as the next test case in $S$. It can be noted that if strength $(\tau + 1)$ is also unavailable (that is, after using higher interaction coverage, there still exists more than one “best” candidate), ICBP will randomly choose an element from the remaining “best” candidates as the next test case in $S$.

E. Lower-strength Tie-breaking

Compared to the Higher-strength tie-breaking technique, Lower-strength tie-breaking uses the strength $(\tau - 1)$ to further guide the prioritization when strength $\tau$ is unavailable. Similar to Higher-strength tie-breaking, if strength $(\tau - 1)$ is also unavailable, ICBP chooses one element from the remaining “best” candidates as the next test case at a random manner.

Among all tie-breaking techniques, first-element tie-breaking and last-element tie-breaking are deterministic, which means that given an original interaction test sequence, each of them could obtain a unique interaction test sequence. However, other three tie-breaking techniques are nondeterministic, because they may randomly choose an element from all “best” candidates as the next test case.

IV. Empirical Case Study

In this section, an empirical case study is presented to analyze the effectiveness of different tie-breaking techniques in the prioritization of interaction test suites, according to fault detection. We have designed the empirical study to answer the following research questions:

RQ1: Among all tie-breaking techniques, which technique is best for the prioritization of interaction test suites?

RQ2: How to choose the tie-breaking technique in practical testing?

A. Setup

We use a medium-sized real-life program from a lexical analyzer system (flex), which is obtained from the Software Infrastructure Repository (SIR) [13]. This program involves five versions, and contains 9,581 sin 11,470 uncommented lines of C language code, which are also augmented with a seeded fault library. In this paper, we have used 34 seeded faults. According to Petke’s investigation [8], the test profile of program flex is $TP(9, 2^6 3^2 5^1, \emptyset)$, where $\emptyset \neq \emptyset$.

The original covering arrays were generated by two widely-used tools: Advanced Combinatorial Testing System (ACTS) [14]; and Pairwise Independent Combinatorial Testing (PICT) [15], both of which are supported by greedy algorithms, and respectively implemented by the In-Parameter-Order (IPO) method and the one-test-at-a-time approach. We focused on covering arrays with strength $\tau = 2, 3, 4, 5$. Since some tie-breaking techniques involve randomization such as random tie-breaking, we ran the experiment 100 times for each interaction test suite and report the average.

In practical testing, testing resources may be limited to allow only part of interaction test sequence to be executed. In this study, therefore, we consider different budgets by considering different percentages ($p$) of each interaction test sequence, for instance, $p = 5\%, 10\%, 25\%, 50\%, 75\%$, and $100\%$.

B. Metrics

Generally speaking, APFD has been used to evaluate different prioritization techniques. However, it has two requirements [2, 6, 9, 10], so as to fail to be used as the evaluation metric. Therefore, in this study we used an alternative of APFD, namely Normalized APFD (or NAPFD)
in order to evaluate the fault detection rate for each tie-breaking technique.

C. Results and Analysis

Table I presents the NAPFD metric values for ICBP with different tie-breaking techniques when executing the certain percentage of each interaction test sequence. It can be noted that the bold datum in this table is largest in each sub-column. From this table, we can have the following observations.

1) Among all tie-breaking techniques, which technique is best for the prioritization of interaction test suites? According to data shown in Table I, it can be clearly seen that there is no best tie-breaking technique for the prioritization of interaction test suites. In other words, each tie-breaking technique performs best in some cases. For example, random tie-breaking obtains the best NAPFD metric values for ACTS covering arrays at strengths $\tau = 2$ and $\tau = 4$ when $p$ is high; first-element tie-breaking performs best for PICT covering array at strength $\tau = 5$ regardless of $p$ value; last-element tie-breaking has the best performance when prioritizing PICT covering array at strength $\tau = 3$; higher-strength tie-breaking has the best rates of fault detection for ACTS covering arrays at strength $\tau = 3$ when $p$ is low; and lower-strength tie-breaking behaves best NAPFD metric values for PICT covering array at $\tau = 2$ when $p$ is high.

However, on the whole, first-element tie-breaking performs worst in many cases. Therefore, first-element tie-breaking is the last choice. Higher-strength tie-breaking could be a better choice when executing fewer number of combinatorial test cases in the interaction test sequence. Additionally, random tie-breaking and last-element tie-breaking could be best in many cases.

2) How to choose the tie-breaking technique in practical testing? According to the definition of each tie-breaking technique, higher-strength and lower-strength tie-breaking techniques are more time-consuming than other tie-breaking techniques, because they need to count information at higher strength (or lower strength). As discussed before, random tie-breaking and last-element tie-breaking would be better choices according to fault detection rates. As a consequence, in practical testing when testing resources are sufficient, testers would choose the higher-strength tie-breaking technique in the prioritization of interaction test suites, because it needs more prioritization time; when testing resources are limited, random tie-breaking and last-element tie-breaking would be better alternatives, because they has less prioritization time. On the other hand, random tie-breaking and higher-strength tie-breaking are un-deterministic; while last-element tie-breaking is deterministic. Therefore, when testers need to compare their methods with deterministic algorithms, last-element tie-breaking is a better choice; otherwise, random tie-breaking or higher-strength tie-breaking would be better.

D. Threats to Validity

Despite our best efforts, our experiments may face some threats to validity.

The first threat is the selection of experimental data – in this paper only a medium-sized subject program has been used to investigate the effectiveness of different tie-breaking techniques. Additionally, two tools used to construct interaction test suites are widely used but both of them are greedy. Finally, we applied different tie-breaking techniques to only ICBP algorithm. To address this threat, additional

<table>
<thead>
<tr>
<th>Method</th>
<th>Strength</th>
<th>$p$ of ACTS Interaction Test Sequence Executed</th>
<th>$p$ of PICT Interaction Test Sequence Executed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>5%</td>
<td>10%</td>
</tr>
<tr>
<td>Random Tie-breaking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First-element Tie-breaking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last-element Tie-breaking</td>
<td>$\tau = 2$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Higher-strength Tie-breaking</td>
<td>$\tau = 3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower-strength Tie-breaking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Random Tie-breaking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>First-element Tie-breaking</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Last-element Tie-breaking</td>
<td>$\tau = 3$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Higher-strength Tie-breaking</td>
<td>$\tau = 4$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lower-strength Tie-breaking</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table I

The NAPFD metric (%) for ICBP with different tie-breaking techniques for subject program Flex when executing the percentage of interaction test sequence.
studies will be conducted in the future using more real-life programs, more interaction test suite construction tools, and more prioritization algorithms of interaction test suites.

Another threat is the evaluation of experimental results – a metric named APFD or NAPFD was used to evaluate the rates of fault detection for different tie-breaking techniques. The NAPFD metric is commonly used in the study of test case prioritization.

V. CONCLUSIONS AND FUTURE WORK

The prioritization of interaction test suites has been widely studied in recent years, especially when testing resources are limited. During the process of prioritization of interaction test suites, the corresponding algorithm may face a challenge that there may exist more than one “best” candidate, how to choose it from them (or do tie-breaking)? In this paper, we investigate different techniques used to choose the next test case from “best” candidates, including random tie-breaking, first-element tie-breaking, last-element tie-breaking, higher-strength tie-breaking, and lower-strength tie-breaking. An empirical study shows that random tie-breaking and last-element tie-breaking would be the better choices than other tie-breaking techniques for testers.

This study only investigates different tie-breaking techniques on the certain prioritization strategy (that is, interaction coverage based prioritization), it is necessary to apply these tie-breaking techniques to other prioritization strategies of interaction test suites. We will study it in the future.

VI. ACKNOWLEDGMENTS

We would like to thank D. R. Kuhn for providing us the ACTS tool, and the Software-artifact Infrastructure Repository (SIR) [13] which provided the source code and fault data for the subject program (flex). This work is in part supported by the National Natural Science Foundation of China (Grant No. 61202110), the Natural Science Foundation of Jiangsu Province (Grant No. BK2012284), and the Senior Personnel Scientific Research Foundation of Jiangsu University (Grant No. 14JDG039).

REFERENCES