IEICE TRANS. INF. & SYST., VOL.E97–D, NO.4 APRIL 2014

PAPER

New Metrics for Prioritized Interaction Test Suites

Rubing HUANG[†,††a)], *Student Member*, Dave TOWEY[†††], Jinfu CHEN[††b)], *and* Yansheng LU[†], *Nonmembers*

SUMMARY Combinatorial interaction testing has been well studied in recent years, and has been widely applied in practice. It generally aims at generating an effective test suite (an interaction test suite) in order to identify faults that are caused by parameter interactions. Due to some constraints in practical applications (e.g. limited testing resources), for example in combinatorial interaction regression testing, prioritized interaction test suites (called interaction test sequences) are often employed. Consequently, many strategies have been proposed to guide the interaction test suite prioritization. It is, therefore, important to be able to evaluate the different interaction test sequences that have been created by different strategies. A well-known metric is the *Average Percentage of Combinatorial Coverage* (shortly $APCC_\lambda$), which assesses the rate of interaction coverage of a strength λ (level of interaction among parameters) covered by a given interaction test sequence S. However, $APCC_\lambda$ has two drawbacks: firstly, it has two requirements (that all test cases in S be executed, and that all possible λ-wise parameter value combinations be covered by S); and secondly, it can only use a single strength λ (rather than multiple strengths) to evaluate the interaction test sequence – which means that it is not a comprehensive evaluation. To overcome the first drawback, we propose an enhanced metric *Normalized* $APCC_\lambda$ (*NAPCC*) to replace the $APCC_\lambda$. Additionally, to overcome the second drawback, we propose three new metrics: the *Average Percentage of Strengths Satisfied* (*APSS*); the *Average Percentage of Weighted Multiple Interaction Coverage* (*APWMIC*); and the *Normalized APWMIC* (*NAPWMIC*). These metrics comprehensively assess a given interaction test sequence by considering different interaction coverage at different strengths. Empirical studies show that the proposed metrics can be used to distinguish different interaction test sequences, and hence can be used to compare different test prioritization strategies.

key words: *combinatorial interaction testing, test case prioritization, prioritized interaction test suite (or interaction test sequence), interaction coverage, metrics*

1. Introduction

Combinatorial interaction testing, a black-box testing method, has been well researched, and has been applied to practical systems [1]. It focuses on constructing an effective test suite (an interaction test suite) to identify failures triggered by the interactions among k parameters of the software under test (SUT). Here, parameters may represent any factors that affect the running of the SUT, such as user inputs, configuration options, etc.; and each parameter may have several valid values. In fact, combinatorial interaction testing provides a tradeoff between testing effectiveness and efficiency, because it only requires coverage of certain key parameter values combinations, rather than all possible combinations – which may lead to the exponential explosion. For instance, τ-wise ($1 \leq \tau \leq k$) combinatorial interaction testing – where the τ (strength) refers to the level of interaction among parameters – aims at constructing an interaction test suite to cover all possible τ-tuples of parameter values (or named τ-wise parameter value combinations).

Due to limited testing resources in practical applications where combinatorial interaction testing is used, for example in combinatorial interaction regression testing [2], the execution order of combinatorial test cases can be critical: potentially failure-revealing test cases in the interaction test suite should be executed as early as possible. In other words, a well-designed order of test case execution may be able to detect failures earlier, and thus enable earlier fault characterization, diagnosis and revision [1]. To improve testing efficiency, interaction test suite prioritization has been employed to prioritize the test suite according to some strategy [1]. In general, a prioritized interaction test suite is referred to as an interaction test sequence.

To date, many strategies have been proposed to guide the prioritization of interaction test suites [2]–[13], including branch-coverage-based test prioritization [2] and random test case prioritization [8]. Given an interaction test suite, different interaction test sequences can be obtained by using different test prioritization strategies. Therefore, it is critical to evaluate these interaction test sequences in order to choose the best one for further testing or to assess the effectiveness (or efficiency) of each test prioritization strategy.

A well-known metric used to evaluate interaction test sequences from the perspective of the rate of interaction coverage at a fixed strength (denoted as λ, where $1 \leq \lambda \leq k$), is the *Average Percentage of Combinatorial Coverage* ($APCC_\lambda$) [8]. The $APCC_\lambda$ metric, however, may face a couple of drawbacks when evaluating different interaction test sequences prioritized by different prioritization strategies. The first drawback is that the $APCC_\lambda$ metric has two requirements: (1) that all test cases in the interaction test sequence should be executed; and (2) that all parameter value combinations at a given strength should be covered by the interaction test sequence. Additionally, a further drawback for $APCC_\lambda$ is that it uses only a single strength rather than

Manuscript received July 31, 2013.
Manuscript revised November 3, 2013.

[†]The authors are with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, Hubei 430074, P.R. China.
[††]The authors are with the School of Computer Science and Telecommunication Engineering, Jiangsu University, Zhenjiang, Jiangsu 212013, P.R. China.
[†††]The author is with Division of Computer Science, The University of Nottingham Ningbo China, Ningbo, Zhejiang 315100, P.R. China.
a) E-mail: rbhuang@hust.edu.cn
b) E-mail: jinfuchen@ujs.edu.cn

DOI: 10.1587/transinf.E97.D.830

Table 1 A test profile denoted as $TP(5, 3^5)$ for the "Research Field" option shown Fig. 1.

Parameters	Fundamentals(A)	Communications(B)	Electronic(C)	Information & Systems(D)	Transactions (1976-1990)
Values	*INVITED*	*INVITED*	*INVITED*	*INVITED*	*Computers*
	Acoustic	*Sensing*	*Components*	*Algorithms*	*Physics*
	Education	*Terminals*	*Materials*	*Software Engineering*	*Mathematics*

multiple strengths to assess each interaction test sequence, and consequently, it may not have a comprehensive evaluation of a given interaction test sequence, potentially failing to distinguish different interaction test sequences of a particular interaction test suite. We present a motivating example to illustrate this in the following section.

Motivated by the above, in this paper, we first propose an enhancement to $APCC_\lambda$, named *Normalized* $APCC_\lambda$ ($NAPCC_\lambda$), which attempts to alleviate the difficulties associated with $APCC_\lambda$'s two requirements. To overcome the second drawback of $APCC_\lambda$, by considering different interaction coverage at different strengths, we next propose three new metrics: *Average Percentage of Strengths Satisfied* (*APSS*); *Average Percentage of Weighted Multiple Interaction Coverage* (*APWMIC*); and *Normalized APWMIC* (*NAPWMIC*). These three metrics aim to provide a comprehensive evaluation of interaction test sequences. We present the results of empirical studies to investigate the performance of the proposed metrics for different interaction sequences prioritized by different test prioritization strategies, showing that the metrics are reasonable and practical. Moreover, we also provide a comparative analysis of the different metrics, which may serve as guidelines to assist testers in their selection.

The rest of this paper is organized as follows: Section 2 reviews some preliminaries including combinatorial interaction testing, interaction test suite prioritization, and the $APCC_\lambda$ metric. Section 3 proposes the new metric, named *Normalized* $APCC_\lambda$, as an enhancement of $APCC_\lambda$. Section 4 proposes three new metrics used to comprehensively evaluate prioritized interaction test suites. Section 5 describes some empirical studies to analyze the proposed metrics. Section 6 presents a comparison of the different metrics for interaction test sequences, and finally, some conclusions and future work are given in Sect. 7.

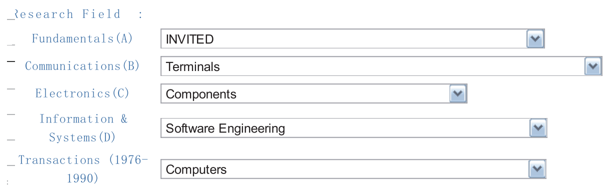
2. Preliminaries

In this section, some related work is described, including combinatorial interaction testing, prioritization of interaction test suites, and the *Average Percentage of Combinatorial Coverage* ($APCC_\lambda$) metric.

2.1 Combinatorial Interaction Testing

Combinatorial interaction testing is a widely used black-box testing method for detecting interaction defects in software systems.

Assume that a system under test (SUT) has k parameters that constitute a parameter set $P = \{p_1, p_2, \cdots, p_k\}$, which may represent any factors that affect the running of



Research Field :

Fundamentals(A): INVITED

Communications(B): Terminals

Electronics(C): Components

Information & Systems(D): Software Engineering

Transactions (1976-1990): Computers

Fig. 1 The option "Research Field" from the IEICE homepage.

the SUT, such as user inputs, configuration options, etc., and each parameter p_i has discrete valid values or levels from the finite set V_i. Let C be the set of constraints on parameter value combinations, and R be the set of interaction relations among parameters. For convenience, in the remainder of this paper, a *combination of parameters* will be referred to as a *parameter combination*, and a *parameter value combination* or a *combination of parameter values* as a *value combination*.

Definition 1 (Test profile): A $TP(k, |V_1||V_2|\cdots|V_k|, C)$ is a test profile, which contains information about a combinatorial test space of the SUT, including k parameters, $|V_i|$ values for each i-th parameter, and value combination constraints C.

Since the value combination constraints have no impact on our current study, they are not considered in this paper. Therefore, a $TP(k, |V_1||V_2|\cdots|V_k|, C)$ can be abbreviated as $TP(k, |V_1||V_2|\cdots|V_k|)$. Our definitions are based on a $TP(k, |V_1||V_2|\cdots|V_k|)$ notation.

For example, Fig. 1 presents a screen dump of the "Research Field" options from the homepage[†] of the IEICE (The Institute of Electronics, Information and Communication Engineers). There are five sub-options (parameters), each of which has various possible values. Due to page limitation, we choose only three representative values for each parameter, allowing us to summarize the information in Fig. 1 as a five-parameter system where each parameter has three values, as in Table 1. Therefore, the test profile of the "Research Field" option can be described as $TP(5, 3^5)$.

Definition 2 (Test case): A test case is a k-tuple $(v_1, v_2, \cdots, v_k)$, where $v_1 \in V_1, v_2 \in V_2, \cdots, v_k \in V_k$.

As shown in Table 1, a test case of the "Research Field" option is: (*INVITED*, *Terminals*, *Components*, *Software Engineering*, *Computers*).

Given a $TP(k, |V_1||V_2|\cdots|V_k|)$, all the possible test cases form an exhaustive interaction test suite $T_{all} = V_1 \times V_2 \times \cdots \times V_k$, the size of which is $|T_{all}| = |V_1| \times |V_2| \times \cdots \times |V_k|$.

Definition 3 (λ-wise value combination): A λ-wise value

[†]http://search.ieice.org/bin/search.php?lang=E

Table 2 A covering array $CA(12, 2, 5, 3^5)$ for the system in Table 1.

Test No.	Fundamentals(A)	Communications(B)	Electronic(C)	Information & Systems(D)	Transactions (1976-1990)
1	*INVITED*	*INVITED*	*Materials*	*INVITED*	*Mathematics*
2	*INVITED*	*Sensing*	*Components*	*Algorithms*	*Computers*
3	*INVITED*	*Terminals*	*INVITED*	*Software Engineering*	*Mathematics*
4	*Acoustic*	*INVITED*	*Components*	*Software Engineering*	*Physics*
5	*Acoustic*	*Sensing*	*INVITED*	*INVITED*	*Mathematics*
6	*Acoustic*	*Terminals*	*Materials*	*Algorithms*	*Computers*
7	*Education*	*INVITED*	*INVITED*	*Algorithms*	*Physics*
8	*Education*	*Sensing*	*Materials*	*Software Engineering*	*Physics*
9	*Education*	*Terminals*	*Components*	*INVITED*	*Computers*
10	*INVITED*	*Terminals*	*Materials*	*INVITED*	*Physics*
11	*Education*	*Sensing*	*Components*	*Algorithms*	*Mathematics*
12	*Acoustic*	*INVITED*	*INVITED*	*Software Engineering*	*Computers*

combination is a k-tuple $(\widehat{v_1}, \widehat{v_2}, \cdots, \widehat{v_k})$ involving λ parameters with fixed values (called fixed parameters), and $(k - \lambda)$ parameters with arbitrary, allowable values (called free parameters), where $0 \leq \lambda \leq k$ and

$$\widehat{v_i} = \begin{cases} v_i \in V_i, & \text{if } p_i \text{ is a fixed parameter;} \\ \text{“}-\text{”}, & \text{if } p_i \text{ is a free parameter.} \end{cases} \tag{1}$$

Generally speaking, a λ-wise value combination is also known as a λ-value schema [1], and λ is called the strength. When $\lambda = k$, a λ-wise value combination becomes a test case for the SUT as each parameter is a fixed parameter. For ease of description, we define $CombSet_\lambda(tc)$ as the set of all λ-wise value combinations covered by the test case tc. Obviously, a test case tc with k parameters contains C_k^λ λ-wise value combinations, that is, $|CombSet_\lambda(tc)| = C_k^\lambda$. To simplify the problem, each "–" is omitted from the λ-wise value combinations. Therefore, if $tc = (v_1, v_2, \cdots, v_k)$ where $v_i \in V_i$ $(i = 1, 2, \cdots, k)$, then the $CombSet_\lambda(tc)$ can be described as:

$$CombSet_\lambda(tc) = \{(v_{j_1}, v_{j_2}, \cdots, v_{j_\lambda}) | 1 \leq j_1 < j_2 < \cdots < j_\lambda \leq k\}. \tag{2}$$

Given an interaction test suite T, the $CombSet_\lambda(T)$ can be written as:

$$CombSet_\lambda(T) = \bigcup_{tc \in T} CombSet_\lambda(tc). \tag{3}$$

If $T = T_{all}$, then $|CombSet_\lambda(T)|$ is fixed, and can be described as $\sum_{i_1=1}^{k-\lambda+1} \ldots \sum_{i_\lambda = i_{\lambda-1}+1}^{k} (|V_{i_1}| \ldots |V_{i_\lambda}|)$.

Definition 4 (Covering array): An $N \times k$ matrix is a τ-wise $(1 \leq \tau \leq k)$ covering array denoted as $CA(N; \tau, k, |V_1||V_2| \cdots |V_k|)$, which satisfies the following properties: (1) each column i $(i = 1, 2, \cdots, k)$ contains only elements from the set V_i; and (2) the rows of each $N \times \tau$ sub-matrix cover all τ-wise value combinations from the τ columns at least once.

The interaction relation set R has elements with the same size for a $CA(N; \tau, k, |V_1||V_2| \cdots |V_k|)$, that is, $R = \{\{p_{j_1}, p_{j_2}, \cdots, p_{j_\tau}\} | p_{j_1} \in P, p_{j_2} \in P, \cdots, p_{j_\tau} \in P, 1 \leq j_1 < j_2 < \cdots < j_\tau \leq k, \ \tau \text{ is fixed}\}$ and $|R| = C_k^\tau$.

For example, to exhaustively test all possible value combinations for the system shown in Table 1, we would require $3^5 = 243$ test cases. However, as shown in Table 2, a covering array $CA(12, 2, 5, 3^5)$ requires only 12 test cases to cover all 2-wise value combinations.

Table 3 The summary of the prioritization of interaction test suite.

Strategy	Reference
Pure prioritization	[2], [5], [6], [8], [9], [10], [13]
Re-generation prioritization	[2], [3], [4], [7], [9], [11], [12]

As noted previously, a covering array is an interaction test suite, but not vice versa.

Although interaction test suite construction has been to be proven to an NP-Complete problem [14], many strategies and tools have been developed in recent years. Most of these strategies can be classified into four categories: algebraic methods, greedy algorithms, recursive algorithms, and heuristic search algorithms (see [1] for more details).

2.2 Prioritization of Interaction Test Suites

Interaction test suite prioritization aims to create a test case execution order for the test suite according to some criteria (eg. statement coverage), so that test cases with higher priority are executed earlier in testing. A prioritized interaction test suite is called an interaction test sequence, and a well-prioritized interaction test sequence may improve the likelihood of detecting faults earlier [1]. This problem is defined as follows [1]:

Definition 5 (Interaction test suite prioritization): Given a tuple (T, Ω, g), where T is an interaction test suite, Ω is the set of all possible interaction test sequences obtained by permutating test cases in T, and g is a function from Ω to real numbers, the problem of interaction test suite prioritization is to find an $S \in \Omega$ such that:

$$(\forall S')\ (S' \in \Omega)\ (S' \neq S)\ [g(S) \geq g(S')]. \tag{4}$$

In Eq. (4), g is a function to evaluate an interaction test sequence S by returning an award value, with higher values implying better sequences. Different functions can be used to evaluate interaction test sequences according to different criteria, such as fault detection [15], statement coverage [16], branch coverage [16], block coverage [16], and so on.

The prioritization of interaction test suites has been extensively researched in recent years, and according to [2], it can be divided into two categories: (1) *Pure prioritization*: re-prioritizing test cases in the interaction test suite; and (2) *Re-generation prioritization*: prioritization during the generation of the interaction test suite. Table 3 summarizes some details of the literature on interaction test suite prioritization.

2.3 Average Percentage of Combinatorial Coverage

As discussed, in Eq. (4) there have been many implementations of function g, proposed according to different criteria. For example, if g is related to fault detection, a corresponding function (metric) may be the *Average of the Percentage of Faults Detected* (*APFD*) [15], which measures the weighted average of the percentage of faults detected over the life of the suite; if g is related to statement coverage, then a corresponding metric may be the *Average Percentage of Statement Coverage* (*APSC*) [16], which measures the rate at which an interaction test sequence covers the statements. However, since it is not normally possible to know the number of faults triggered by a test case in advance, use of the *APFD* metric may not be possible. Additionally, due to a possible lack of source code, it is also not always possible to obtain the statement coverage covered by a test case, thus hindering use of the *APSC* metric.

To evaluate interaction test sequences, Wang *et al.* [8] proposed the *Average Percentage of Combinatorial Coverage* ($APCC_\lambda$) metric, which measures the rate at which an interaction test sequence covers value combinations at a given strength λ. Given a λ-wise covering array $T = \{tc_1, tc_2, \cdots, tc_n\}$ with size n, and an interaction test sequence $S = \langle s_1, s_2, \cdots, s_n\rangle$, formed by permuting T, where $s_i \in T$ and $s_i \neq s_j, i, j = 1, 2, \cdots, n, i \neq j$, then the definition of $APCC_\lambda$ is as follows:

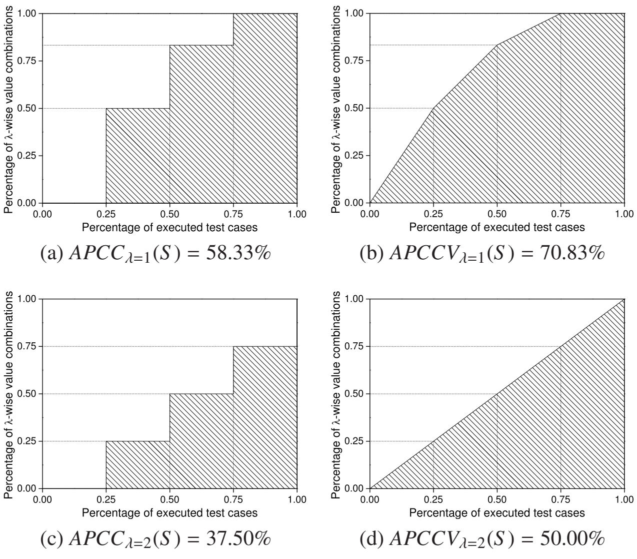
$$APCC_\lambda(S) = \frac{\sum_{i=1}^{n-1} |\bigcup_{j=1}^{i} CombSet_\lambda(s_j)|}{n \times |CombSet_\lambda(T_{all})|}. \quad (5)$$

In order to unify the definition of $APCC_\lambda$ with *APFD* [15] and *APSC* [16], and to facilitate the introduction of other new metrics, we present a variant of $APCC_\lambda$, named the *APCC Variant* ($APCCV_\lambda$) metric, as follows:

$$APCCV_\lambda(S) = 1 - \frac{TC_1 + TC_2 + \cdots + TC_m}{n \times m} + \frac{1}{2n}, \quad (6)$$

where $m = |CombSet_\lambda(T_{all})|$, and TC_i $(i = 1, 2, \cdots, m)$ is the minimum number of test cases in S required to achieve the i-th λ-wise value combination.

Both the $APCC_\lambda$ and $APCCV_\lambda$ metrics measure how quickly an interaction test sequence achieves λ-wise interaction coverage. They both measure the area under the curve for the plot of increasing λ-wise interaction coverage for an interaction test sequence. Actually, both $APCC_\lambda$ and $APCCV_\lambda$ metrics use the piecewise function to describe the increase of interaction coverage at a specific strength, however, they use different plot curves: the $APCC_\lambda$ metric uses the ladder chart; while the $APCCV_\lambda$ metric uses the line chart. The main reason for this, as discussed in [8], is that the piecewise function used by $APCC_\lambda$ is discrete (that is, a step function), but the piecewise function used by $APCCV_\lambda$ is continuous (a linear function). Nevertheless, they are actually equivalent. In other words, given two interaction test sequences, S_1 and S_2, if one metric determines that the test sequence S_1 is better than S_2 (denoted $S_1 \succ S_2$), the other metric will also have the same determination.



(a) $APCC_{\lambda=1}(S) = 58.33\%$ (b) $APCCV_{\lambda=1}(S) = 70.83\%$ (c) $APCC_{\lambda=2}(S) = 37.50\%$ (d) $APCCV_{\lambda=2}(S) = 50.00\%$

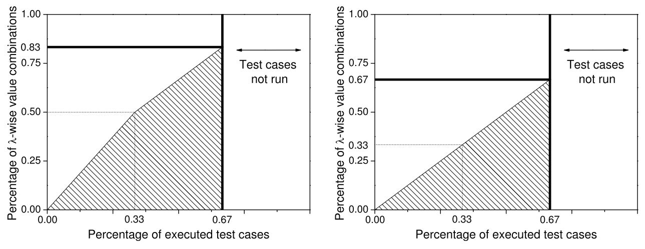
Fig. 2 An example of $APCC_\lambda$ and $APCCV_\lambda$ at different λ values.

To illustrate $APCC_\lambda$ and $APCCV_\lambda$, we present an example based on $TP(3, 2^3)$. Let an interaction test suite be $T = \{tc_1 = (0, 2, 4), tc_2 = (0, 3, 5), tc_3 = (1, 2, 5), tc_4 = (1, 3, 4)\}$, and an interaction test sequence of T be $S = \langle tc_1, tc_2, tc_3, tc_4\rangle$. Figure 2 (a) shows $APCC_\lambda(S)$, and Fig. 2 (b) shows $APCCV_\lambda(S)$, both for $\lambda = 1$; while Fig. 2 (c) shows $APCC_\lambda(S)$, and Fig. 2 (d) shows $APCCV_\lambda(S)$ for $\lambda = 2$. The shaded region area in each figure corresponds to each $APCC_\lambda(S)$ or $APCCV_\lambda(S)$.

3. Normalized APCCV

As shown in Eqs. (5) and (6), the $APCC_\lambda$ and $APCCV_\lambda$ metrics have two requirements: (1) all test cases in S should be executed – n is equal to $|S|$; and (2) all λ-wise value combinations should be covered by S – $CombSet_\lambda(S) = CombSet_\lambda(T_{all})$, i.e., S is a λ-wise covering array. In other words, if not all test cases in S are executed, or if S does not cover all possible λ-wise value combinations, then the $APCC_\lambda$ and $APCCV_\lambda$ metrics may fail to objectively evaluate the rate of λ-wise interaction coverage of S. To overcome these two requirements, we next propose a new metric called *Normalized* $APCCV_\lambda$ ($NAPCCV_\lambda$) as follows.

Given an interaction test suite $T = \{tc_1, tc_2, \cdots, tc_n\}$ of size n, and an interaction test sequence $S = \langle s_1, s_2, \cdots s_n\rangle$ of T, where $s_i \in T$ and $s_i \neq s_j$ $(i, j = 1, 2, \cdots, n), i \neq j$, the formula for calculating the $NAPCCV_\lambda(S)$ is:



(a) $NAPCCV_{\lambda=1}(S_1) = 45.83\%$ (b) $NAPCCV_{\lambda=2}(S_1) = 25.00\%$

Fig. 3 An example of $NAPCCV_\lambda$ at different λ values.

$$NAPCCV_\lambda(S) = p - \frac{TC_1 + TC_2 + \cdots + TC_{m'}}{n' \times m'} + \frac{p}{2n'}, \tag{7}$$

where TC_i $(i = 1, 2, \cdots, m')$ has the same meaning as for the $APCCV_\lambda$ metric (Eq. (6)). However, $m' = |CombSet_\lambda(S)|$; n' is the number of executed test cases in S $(n' \leq n)$; and p is the number of λ-wise value combinations covered by the first n' test cases in S divided by the number of λ-wise value combinations covered by S. If a λ-wise value combination, i, is not covered by the first n' test cases, we set $TC_i = 0$.

Similar to the $APCCV_\lambda$ metric, $NAPCCV_\lambda$ also presents an evaluation of the rate of λ-wise interaction coverage of S. However, it broadens the application range of the $APCCV_\lambda$ metric: although $APCCV_\lambda$ evaluates interaction test sequences by ordering covering arrays, $NAPCCV_\lambda$ evaluates the sequences by permutating any interaction test suites – not only covering arrays. Additionally, although $APCCV_\lambda$ requires that all test cases in the test sequence be executed, $NAPCCV_\lambda$ does not have this requirement.

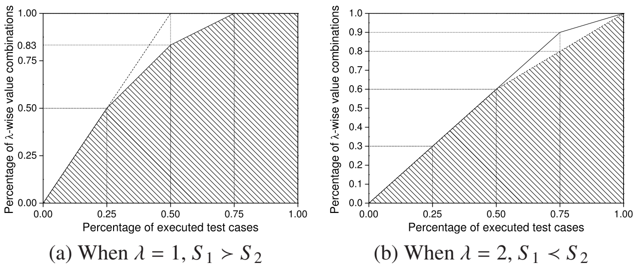
To illustrate the $NAPCCV_\lambda$ metric more clearly, an example is given: we suppose $S =< tc_1 = (0, 2, 4), tc_2 = (0, 3, 5), tc_3 = (1, 2, 5) >$, and due to limited testing resources, we suppose that only two test cases in S are executed – $S_1 = \langle tc_1, tc_2 \rangle$ and $n' = 2$. As shown in Fig. 3, When $\lambda = 1$, we have $m' = 6$ and $p = \frac{CombSet_{\lambda=1}(S_1)}{CombSet_{\lambda=1}(S)} = \frac{5}{6} = 0.83$, and therefore $NAPCCV_{\lambda=1}(S_1) = 45.83\%$; when $\lambda = 2$, we have $m' = 9$ and $p = \frac{CombSet_{\lambda=2}(S_1)}{CombSet_{\lambda=2}(S)} = \frac{6}{9} = 0.67$, and therefore $NAPCCV_{\lambda=2}(S_1) = 25.00\%$.

4. New Metrics for Interaction Test Sequences

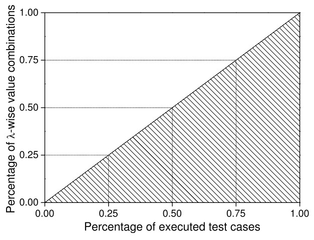
In this section, some new metrics to evaluate interaction test sequences are proposed. These metrics aim at overcoming one of the drawbacks associated with the $APCC_\lambda$ and $NAPCCV_\lambda$ metrics. We first present a motivating example to illustrate the problem, and then propose new metrics to overcome it.

4.1 Motivating Example

Given a test profile denoted as $TP(3, 2^3)$, and an interaction test suite $T = \{tc_1 = (0, 2, 4), tc_2 = (1, 3, 5), tc_3 = (0, 3, 5), tc_4 = (1, 3, 4)\}$, let two interaction test sequences of T be $S_1 = \langle tc_1, tc_2, tc_3, tc_4 \rangle$ and $S_2 = \langle tc_1, tc_3, tc_4, tc_2 \rangle$.



(a) When $\lambda = 1$, $S_1 \succ S_2$ (b) When $\lambda = 2$, $S_1 \prec S_2$



(c) When $\lambda = 3$, $S_1 \equiv S_2$

Fig. 4 A motivating example.

In Fig. 4, the dashed line represents interaction coverage at a given strength covered by the interaction test sequence S_1, and the full line represents interaction coverage for S_2. Each figure corresponds to each strength λ, and the shaded region is the overlap between the S_1 and S_2 regions.

When $\lambda = 1$, as shown in Fig. 4 (a), the area of the region for S_1 is larger than that for S_2. $NAPCCV_{\lambda=1}(S_1) = 75.00\%$ and $NAPCCV_{\lambda=1}(S_2) = 70.83\%$ – so, $NAPCCV_{\lambda=1}(S_1) > NAPCCV_{\lambda=1}(S_2)$ (that is, $S_1 \succ S_2$). Similarly, it can be seen from Fig. 4 (b) that when $\lambda = 2$, $NAPCCV_{\lambda=2}(S_1) = 55.00\% < NAPCCV_{\lambda=2}(S_2) = 57.50\%$ (denoted $S_1 \prec S_2$). When $\lambda = 3$, as shown in Fig. 4 (c), we can see that $NAPCCV_{\lambda=3}(S_1) = NAPCCV_{\lambda=3}(S_2) = 50.00\%$ (denoted $S_1 \equiv S_2$).

According to the above example, we can conclude that the $NAPCCV_\lambda$ metric can be successfully used to evaluate the rate of λ-wise interaction coverage for a given interaction test sequence. However, as a metric, it may not be effective at distinguishing different interaction test sequences permutated from the same interaction test suite: as shown in Fig. 4, for example, when $\lambda = 1$, $S_1 \succ S_2$; when $\lambda = 2$, $S_1 \prec S_2$; and when $\lambda = 3$, $S_1 \equiv S_2$. Therefore it may not be possible to use $NAPCCV_\lambda$ to decide whether S_1 or S_2 is the better interaction test sequence. For a given interaction test suite T, although there are many strategies to create different interaction test sequences of T, in practice, only one test sequence is required. In other words, some metrics based on a single strength such as $APCCV_\lambda$ and $NAPCCV_\lambda$, may fail to evaluate and compare two interaction test sequences. Additionally, previous studies (e.g. [17], [18]) reported that different faults are triggered by different interaction coverage (or different strengths). The above facts motivate us to research new metrics that can evaluate and distinguish different interaction test sequences, and that can do so by comprehensively considering multiple interaction coverage at multiple strengths.

4.2 Properties of Interaction Test Sequences

For clarity description, we first define a few terms. Based on $TP(k, |V_1||V_2|\cdots|V_k|)$, given an interaction test sequence S with size n, denoted $S = \langle s_1, s_2, \cdots, s_n\rangle$, we define a function $F_\lambda(S, i)$, where $1 \leq i \leq n$, which returns the set of λ-wise value combinations covered by the first i test cases in S, that is, $F_\lambda(S, i) = \bigcup_{j=1}^{i} CombSet_\lambda(s_j)$. Intuitively speaking, $|F_\lambda(S, 1)| = C_k^\lambda$, because the first test case can cover C_k^λ λ-wise value combinations; while $F_\lambda(S, n) = CombSet_\lambda(S)$, because all test cases in S are executed.

Property 1: If $F_\lambda(S, \varphi) = CombSet_\lambda(S)$ where $1 \leq \varphi \leq n$, $F_{\lambda'}(S, \varphi) = CombSet_{\lambda'}(S)$ where $1 \leq \lambda' < \lambda \leq k$.

***Proof* 1:** If $F_\lambda(S, \varphi) = CombSet_\lambda(S)$ where $1 \leq \varphi \leq n$, it can be concluded that the first φ test cases in S can cover all possible λ-wise value combinations that have been covered by S. We suppose that the first φ test cases in S form an interaction test sub-sequence S', therefore, $CombSet_\lambda(S') = CombSet_\lambda(S)$. As a consequence,

$$CombSet_{\lambda'}(CombSet_\lambda(S')) = CombSet_{\lambda'}(CombSet_\lambda(S)), \tag{8}$$

where $1 \leq \lambda' < \lambda \leq k$. Since

$$CombSet_{\lambda'}(CombSet_\lambda(S')) = CombSet_{\lambda'}(S'), \tag{9}$$

Eq. (8) converts to the following formula:

$$CombSet_{\lambda'}(S') = CombSet_{\lambda'}(S), \tag{10}$$

which indicates that S' covers all possible λ'-wise value combinations that have been covered by S. In other words, $F_{\lambda'}(S, \varphi) = CombSet_{\lambda'}(S)$, where $1 \leq \lambda' < \lambda \leq k$.

Based on this, we propose a new metric for interaction test sequences, *Average Percentage of Strengths Satisfied* (*APSS*), which considers different interaction coverage at different strengths.

4.3 Average Percentage of Strengths Satisfied

Based on $TP(k, |V_1||V_2|\cdots|V_k|)$, given an interaction test sequence S with size n, denoted $S = \langle s_1, s_2, \cdots, s_n\rangle$, we define a function $f_\lambda(S)$ returning the minimum number of test cases in S required to cover all possible λ-wise value combinations that have been previously covered by S. More specifically, $f_\lambda(S) = \varphi$, where φ satisfies the following conditions: (1) $F_\lambda(S, \varphi) = CombSet_\lambda(S)$; and (2) $F_\lambda(S, \varphi - 1) \subset CombSet_\lambda(S)$.

The function $f_\lambda(S)$ has the following properties:

Property 2: $1 \leq f_1(S) \leq f_2(S) \leq \cdots \leq f_k(S) = n$.

***Proof* 2:** On the one hand, it is obvious that $f_1(S) \geq 1$ because $S \neq \emptyset$, and $f_k(S) = n$ because no redundancy exists in S. On the other hand, according to Property 1, it can be seen that if all λ'-wise value combinations previously covered by S have not been covered by an interaction test sub-sequence S' of S, S' has not also covered all λ-wise value combinations previously covered by S, where $1 \leq \lambda' < \lambda \leq k$. In other words, S' generally requires more test cases to cover all λ-wise value combinations previously covered by S than to cover all value combinations at strengths lower than λ, that is, it can be concluded that $f_1(S) \leq f_2(S) \leq \cdots \leq f_k(S)$. As a consequence, $1 \leq f_1(S) \leq f_2(S) \leq \cdots \leq f_k(S) = n$.

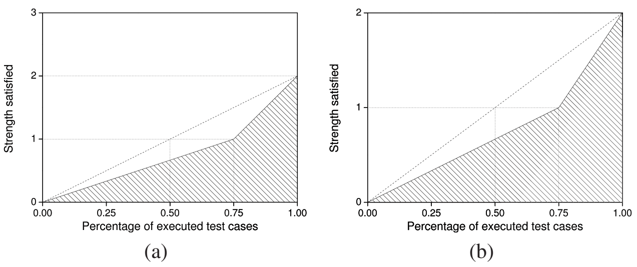


Fig. 5 An example of the *APSS* metric.

Property 3: If S is an interaction test sequence formed by permutating a covering array, then $f_\lambda(S) \geq \max_{i=1}^{k}\{|V_i|\}$, where $1 \leq \lambda \leq k$.

***Proof* 3:** This property is intuitive, because any interaction test sub-sequence of S requires at least $\max_{i=1}^{k}\{|V_i|\}$ test cases in S in order to cover all possible 1-wise value combinations, that is, $f_1(S) \geq \max_{i=1}^{k}\{|V_i|\}$. According to Property 2, it can be concluded that $f_\lambda(S) \geq \max_{i=1}^{k}\{|V_i|\}$, where $1 \leq \lambda \leq k$.

Given an interaction test sequence S, of size n, its *Average Percentage of Strengths Satisfied* (*APSS*) is defined as follows:

$$APSS(S) = 1 - \frac{f_1(S) + f_2(S) + \cdots + f_k(S)}{n \times k} + \frac{1}{2n}. \tag{11}$$

To explain this definition, we consider again the example used in the Sect. 4.1. As shown in Fig. 4, it is difficult to determine whether S_1 or S_2 is the better interaction test sequence, in terms of the $NAPCCV_\lambda$ metric. Figure 5 (a) shows the *APSS* values for S_1 (dashed line) and S_2 (full line), with the shaded region representing the overlap between the S_1 and S_2 areas. From this figure, it can be clearly seen that the *APSS* value of S_1 is greater than that of S_2 — $APSS(S_1) = 33.33\%$; $APSS(S_2) = 25.00\%$ — based on which, we can conclude that S_1 is a better interaction test sequence ($S_1 \succ S_2$), according to the *APSS* metric.

In Eq. (11), since the $f_k(S)$ value of any interaction test sequence is equal to n, the formula can be rewritten as follows:

$$APSS(S) = 1 - \frac{f_1(S) + f_2(S) + \cdots + f_{k-1}(S)}{n \times (k-1)} + \frac{1}{2n}. \tag{12}$$

Consequently, the *APSS* values for S_1 and S_2 can be represented in Fig. 5 (b), where $APSS(S_1) = 50.00\%$ and $APSS(S_2) = 37.50\%$.

Unlike the $NAPCCV_\lambda$ metric, *APSS* not only considers the strength λ, but also considers other strengths from 1 to $(k-1)$. However, the *APSS* metric mainly focuses on the rate of strengths satisfied for an interaction test sequence, neglecting the rate of interaction coverage at each strength. For example, given an interaction test suite T, with two interaction test sequences, S_1 and S_2, although both of them may require the same number of test cases to first cover all possible λ-wise value combinations previously covered by T (i.e., $f_\lambda(S_1) = f_\lambda(S_2)$), however, their rates of λ-wise interaction coverage may differ — $NAPCCV_\lambda(S_1) \neq NAPCCV_\lambda(S_2)$. Motivated by this, it is necessary and reasonable to propose a new metric for interaction test sequences by not only considering different strengths, but also considering the rates of different interaction coverage.

4.4 Average Percentage of Weighted Multiple Interaction Coverage

In this section, we propose the *Average Percentage of Weighted Multiple Interaction Coverage* (*APWMIC*) metric, which is used to evaluate and distinguish different interaction test sequences permutated from a given interaction test suite. The *APWMIC* metric combines benefits of both the $NAPCCV_\lambda$ and *APSS* metrics, because it not only considers the rate of interaction coverage, but also takes account of different strengths.

Before introducing the *APWMIC* metric, we initially define a function $h(s_i)$ which returns a reward value for an executed test case s_i $(i = 1, 2, \cdots, n)$ in the given interaction test sequence $S = \langle s_1, s_2, \cdots, s_n\rangle$. Its definition is as follows:

$$h(s_i) = \sum_{\lambda=1}^{k-1}\left(\omega_\lambda * \frac{|F_\lambda(S,i)|}{|CombSet_\lambda(S)|}\right), \tag{13}$$

where $\omega_\lambda(\lambda = 1, 2, \cdots, k-1)$ is a constant weight for each λ that should be assigned in advance, satisfying the following two requirements: (1) $0 \leq \omega_\lambda \leq 1.0$ $(\lambda = 1, 2, \cdots, k-1)$; and (2) $\sum_{\lambda=1}^{k-1} \omega_\lambda = 1.0$.

Intuitively, $h(s_n) = 1.0$, and the value range of $h(s_i)$ is $[0, 1.0]$, where $i = 1, 2, \cdots, n$. Additionally, $h(s_1) \leq h(s_2) \leq \cdots \leq h(s_n)$.

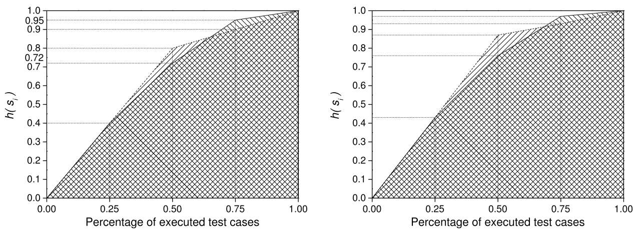
Given a $h(s_i)$ for each s_i in S, the *APWMIC* metric is defined as follows:

$$APWMIC(S) = \frac{1}{n}\sum_{i=1}^{n} h(s_i) - \frac{1}{2n}. \tag{14}$$

However, *APWMIC* has the requirement that all test cases in S be executed, which limits its application, especially when testing resources are limited. To overcome this, we present a variant of *APWMIC* called *Normalized APWMIC* (*NAPWMIC*) as an enhancement.

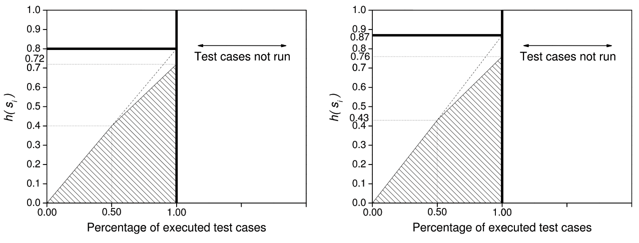
$$NAPWMIC(S) = \frac{1}{n'}\sum_{i=1}^{n'} h(s_i) - \frac{p}{2n'}, \tag{15}$$

where n' is the number of executed test cases in S, and $p =$



(a) Equal weight assignment (b) Empirical weight assignment

Fig. 6 An example of the *NAPWMIC* metric when executing all test cases in each interaction test sequence.



(a) Equal weight assignment (b) Empirical weight assignment

Fig. 7 An example of the *NAPWMIC* metric when executing only some of the test cases in each interaction test sequence.

$h(s_{n'})$. Obviously, if $n' = n$, and $p = 1.0$, then Eq. (15) is equal to Eq. (14).

As shown in Eq. (13), the weight ω_i $(i = 1, 2, \cdots, k-1)$ should be assigned in advance. In this paper, we focus on two weight distributions: (1) equal weight assignment: $\omega_1 = \omega_2 = \cdots = \omega_{k-1} = \frac{1}{k-1}$; and (2) empirical weight assignment. Previous studies (e.g. [17], [18]) have reported that 29% to 82% of faults were caused by 1-wise parameter interactions; 6% to 47% by 2-wise; 2% to 19% by 3-wise; 1% to 7% by 4-wise; and even less by higher than 4-wise parameter interactions. Consequently, we assigned each weight as follows: $\omega_1 = \omega$, and $\omega_{i+1} = \frac{1}{2}\omega_i$, where $i = 1, 2, \cdots, k-2$. For instance, if $k = 3$, then we have $\omega_1 = 0.67$ and $\omega_2 = 0.33$; if $k = 4$, we have $\omega_1 = 0.57$, $\omega_2 = 0.29$, and $\omega_3 = 0.14$.

Here, we look again at the example from Sects. 4.1 and 4.3. When testing resources are sufficient to execute all test cases in each interaction test sequence, as shown in Fig. 6, the dashed line represents S_1 while the full line represents S_2. The areas of the regions with the left and right diagonals are surrounded by the dashed line and the full line, respectively; while the area of the region with the crossed diagonal is the overlap of S_1 and S_2. As shown in Fig. 6 (a), $NAPWMIC(S_1) = 65.00\%$ and $NAPWMIC(S_2) = 64.17\%$, therefore, we can conclude that $S_1 \succ S_2$, according to the *NAPWMIC* metric with equal weight assignment. As shown in Fig. 6 (b), $NAPWMIC(S_1) = 68.33\%$ and $NAPWMIC(S_2) = 66.39\%$, therefore, $S_1 \succ S_2$, according to the *NAPWMIC* metric with empirical weight assignment.

When limited testing resources mean that only some of the test cases in each interaction test sequence can be executed — e.g. 50% in this example, as shown in Fig. 7, where the shaded region is the overlap between the S_1

and S_2 regions — we may have the following results: Fig. 7 (a) shows that *NAPWMIC*(S_1) = 40.00% and *NAPWMIC*(S_2) = 37.92%, when the equal weight assignment is used; Fig. 7 (b) shows that *NAPWMIC*(S_1) = 43.33% and *NAPWMIC*(S_2) = 40.56%, when the empirical weight assignment is used. Therefore, we can conclude that $S_1 \succ S_2$ when executing 50% of the test cases in each interaction test sequence, irrespective of the weight assignment method.

5. Empirical Studies

In this section, we report on some empirical studies conducted to investigate the feasibility and practicability of the proposed metrics.

5.1 Setup

We selected two test profiles, denoted $TP(7, 2^4 3^1 6^1 16^1)$ and $TP(8, 2^6 9^1 10^1)$ [2], which are derived from real-life programs — a module from a lexical analyzer system (flex), and a configuration model for GNUzip (gzip). The interaction test suites were constructed using two well-known tools — *Advanced Combinatorial Testing System* (ACTS) [19] and *Pairwise Independent Combinatorial Testing* (PICT) [20] — both of which are supported by greedy algorithms, and implemented by the *In-Parameter-Order* (IPO) and the *one-test-at-a-time* methods, respectively [1].

As discussed earlier, an interaction test suite is not necessarily a covering array, however, in our empirical studies we used covering arrays generated by ACTS and PICT to represent interaction test suites. We focus on covering arrays with strength $\tau = 2, 3, 4, 5$. The size of each covering array is given in Table 4. Additionally, to obtain different interaction test sequences, we used four prioritization strategies that have commonly used in the field of combinatorial interaction testing — (1) `Original`: generating the interaction test sequence according to the test case order of the corresponding interaction test suite; (2) `Random`: prioritizing the interaction test suite in a random manner [2]; (3) `ICBP`: prioritizing the interaction test suite based on interaction coverage of a given strength, namely, *Interaction Coverage Based Prioritization* (ICBP) [2], [5], [6], [8], [9]; and (4) `IICBP`: prioritizing the interaction test suite based on different interaction coverage by incrementally updating the strength and starting with a strength of 1, namely, *Incremental Interaction Coverage Based Prioritization* (IICBP) [13].

Since randomization was used in the `Random`, `ICBP`, and `IICBP` prioritization strategies, we ran each interaction test suite 100 times for each strategy, and reported the average of the results.

5.2 Results

The *APSS* and *NAPWMIC* metrics were used as the evaluation measures to calculate different interaction test sequences prioritized by different prioritization strategies.

Table 4 Sizes of covering arrays for two test profiles.

Tool	Strength	$TP(7, 2^4 3^1 6^1 16^1)$	$TP(8, 2^6 9^1 10^1)$
ACTS	$\tau = 2$	96	90
	$\tau = 3$	289	180
	$\tau = 4$	578	632
	$\tau = 5$	1728	1080
PICT	$\tau = 2$	96	90
	$\tau = 3$	293	192
	$\tau = 4$	744	592
	$\tau = 5$	1658	1237

5.2.1 Study 1: The *APSS* Metric

Table 5 shows the *APSS* metric values for different interaction test sequences prioritized by different prioritization strategies on the two test profiles. Based on the experimental results, we have the following observations.

1. According to the *APSS* metric, among all prioritization strategies for interaction test suites, `IICBP` performs best, followed by `ICBP`.
2. For each prioritization strategy (`Original`, `Random`, `ICBP`, and `IICBP`), the *APSS* value generally increases with the increase in the strength value.
3. When interaction test suites (or covering arrays) are constructed using the ACTS tool, `Original` generally has the lowest *APSS* values; when constructed using the PICT tool, `Random` performs worst, in terms of the *APSS* metric. The *APSS* values are particularly low for `Original` when using the ACTS tool: for $TP(7, 2^4 3^1 6^1 16^1)$ when $\tau = 2, 3, 4$; and for $TP(8, 2^6 9^1 10^1)$ when $\tau = 2, 3$ — all have values of less than 2.00%.

Observation 1 is easily explained: The main reason is that `IICBP` prioritizes a given interaction interaction suite according to interaction coverage at a low strength λ ($\lambda = 2$), and then incrementally updates the strength value. More specifically, `IICBP` aims at achieving an interaction coverage at low strength as soon as possible, resulting in it requiring the lowest number of test cases to cover all value combinations at a given strength. Therefore, `IICBP` has the highest *APSS* metric value. `ICBP` uses interaction coverage at a fixed strength τ to guide its prioritization process, which guarantees that it needs the lowest number of test cases to cover all possible τ-wise value combinations. However, it cannot guarantee interaction coverage at other strengths. Finally, `Original` does not prioritize any interaction test suites, and `Random` does not use any information to guide the prioritization.

Regarding Observation 2, when the strength τ increases, the size of the τ-wise covering array T increases dramatically. According to Property 2, $f_\lambda(S) \le |S|$ in each interaction test sequence S of T where $1 \le \lambda < \tau$; while $f_{\lambda'}(S)$ is generally equal to $|S|$ where $\tau \le \lambda' \le (k-1)$. Consequently, when τ is larger, for the same prioritization strategy, *APSS* also becomes larger.

The main reason underlying Observation 3 is related

Table 5 The *APSS* metric for different prioritization strategies for each interaction test suite (%).

Tool	Strength	$TP(7, 2^4 3^1 6^1 16^1)$				$TP(8, 2^6 9^1 10^1)$			
		`Original`	`Random`	`ICBP`	`IICBP`	`Original`	`Random`	`ICBP`	`IICBP`
ACTS	$\tau = 2$	1.39	10.24	13.84	14.41	1.83	10.56	13.00	13.25
	$\tau = 3$	1.38	17.22	23.15	26.93	1.71	13.55	14.08	20.48
	$\tau = 4$	1.44	23.46	26.85	35.80	19.41	22.09	32.42	34.88
	$\tau = 5$	23.08	35.41	48.15	54.78	20.18	28.21	39.35	45.61
PICT	$\tau = 2$	12.15	9.91	13.99	14.41	11.67	10.34	13.10	13.25
	$\tau = 3$	16.15	17.39	24.04	27.05	15.59	13.63	16.07	20.73
	$\tau = 4$	25.20	25.18	34.15	40.46	26.60	21.55	32.17	34.37
	$\tau = 5$	37.89	34.83	47.15	54.48	33.74	29.47	40.94	46.36

Table 6 The *NAPWMIC* metric for different prioritization strategies when executing all test cases in each interaction test sequence (%).

Metric	Tool	Strength	$TP(7, 2^4 3^1 6^1 16^1)$				$TP(8, 2^6 9^1 10^1)$			
			`Original`	`Random`	`ICBP`	`IICBP`	`Original`	`Random`	`ICBP`	`IICBP`
$NAPWMIC_{equal}$	ACTS	$\tau = 2$	59.49	63.13	65.44	65.16	60.53	63.86	65.50	65.35
		$\tau = 3$	60.55	69.42	71.90	71.51	63.12	66.15	69.58	68.13
		$\tau = 4$	60.40	73.33	75.12	74.89	70.32	74.01	76.42	75.89
		$\tau = 5$	71.14	82.08	84.60	84.52	70.62	77.21	79.31	79.14
	PICT	$\tau = 2$	64.79	63.59	65.67	65.53	64.93	63.75	65.41	65.38
		$\tau = 3$	71.04	69.99	72.35	73.09	68.43	67.39	68.95	68.51
		$\tau = 4$	77.23	75.96	78.38	78.22	75.36	73.92	75.99	75.71
		$\tau = 5$	83.12	81.81	84.23	84.19	80.20	78.77	80.90	80.73
$NAPWMIC_{empirical}$	ACTS	$\tau = 2$	68.27	77.94	81.39	81.33	73.78	81.12	83.77	83.73
		$\tau = 3$	69.16	86.48	88.77	88.92	75.08	85.18	87.04	87.34
		$\tau = 4$	69.27	90.51	91.63	92.18	84.77	92.46	93.94	94.01
		$\tau = 5$	79.23	95.31	96.22	96.51	83.24	94.60	95.45	95.79
	PICT	$\tau = 2$	80.27	78.33	81.69	81.63	82.59	81.12	83.78	83.80
		$\tau = 3$	87.65	86.85	89.57	89.22	87.32	86.23	87.91	88.07
		$\tau = 4$	92.47	91.96	93.46	93.71	93.04	92.24	93.72	93.83
		$\tau = 5$	95.56	95.20	96.11	96.40	95.66	95.12	96.04	96.24

to the different mechanisms implementing ACTS and PICT. Given a $TP(k, |V_1||V_2| \cdots |V_k|)$ with $|V_1| \geq |V_2| \geq \cdots \geq |V_k|$, ACTS constructs a τ-wise covering array using *horizonal growth*, and then *vertical growth*. During the *horizontal growth* process, ACTS first builds a τ-wise test set for the first τ parameters, meaning it needs at least $1 + (|V_1| - 1) \prod_{i=2}^{\tau} |V_i|$ test cases to cover all possible 1-wise value combinations. Additionally, when $\tau = 2$ or 3, according to the explanation of Observation 2, the ACTS `Original` test sequences may have very low *APSS* metric values. Therefore, it potentially has the low rate of covering all 1-wise value combinations, which may lead to the low *APSS* value. However, PICT uses the *one-test-at-a-time* strategy [1] to construct a τ-wise covering array by choosing an element from candidates such that it covers the largest number of uncovered τ-wise value combinations. In other words, PICT has a similar mechanism to `ICBP` and `IICBP`.

5.2.2 Study 2: The *NAPWMIC* Metric

Since there were two weight assignment methods, for ease of presentation, we use $NAPWMIC_{equal}$ to refer to *NAPWMIC* with equal weight assignment, and $NAPWMIC_{empirical}$ for *NAPWMIC* with empirical weight assignment. Two cases were considered: (1) when testing resources are sufficient to run all test cases in each interaction test sequence; and (2) when testing resources limit running to only 50% of all test cases in each test sequence. Case (1) is presented in Table 6, and case (2) in Table 7.

Based on the experimental data shown in Tables 6 and 7, we have the following observations:

1. The trends for the results when executing all test cases (Table 6) are similar to those when executing only 50% of the test cases (Table 7), but with the $NAPWMIC_{equal}$ scores generally being lower than those for $NAPWMIC_{empirical}$.
2. `Original` and `Random` are the two prioritization strategies with worst overall performance: When interaction test suites are generated with ACTS, `Original` performs worst, followed by `Random`; but when generated by PICT, `Random` has the lowest $NAPWMIC_{equal}$ and $NAPWMIC_{empirical}$ values, followed by `Original`.
3. The `ICBP` and `IICBP` are the two prioritization strategies with the best performance, with both producing very similar $NAPWMIC_{equal}$ and $NAPWMIC_{empirical}$ scores.
4. As the strength value increases, both the $NAPWMIC_{equal}$ and $NAPWMIC_{empirical}$ values also tend to increase, for all test prioritization strategies.

Observations 2 and 4 are basically consistent with Observations 1 and 2 in Study 1 (Sect. 5.2.1), so here we focus the analysis on the other Observations, 1 and 3. For Observation 1, a possible explanation relates to the characteristics of the different interaction test sequences and weight assignments. For Observation 3, `IICBP` focuses on different inter-

Table 7 The *NAPWMIC* metric for different prioritization strategies when executing 50% of all test cases in each interaction test sequence (%).

Metric	Tool	Strength	$TP(7, 2^4 3^1 6^1 16^1)$				$TP(8, 2^6 9^1 10^1)$			
			`Original`	`Random`	`ICBP`	`IICBP`	`Original`	`Random`	`ICBP`	`IICBP`
$NAPWMIC_{equal}$	ACTS	$\tau = 2$	38.17	42.33	45.67	45.17	43.35	43.35	46.16	45.90
		$\tau = 3$	39.82	51.48	54.81	54.24	47.40	47.40	51.31	49.35
		$\tau = 4$	39.84	57.45	60.15	59.93	50.50	58.40	61.66	60.79
		$\tau = 5$	48.29	69.94	73.26	73.22	49.82	63.14	65.92	65.75
	PICT	$\tau = 2$	44.56	42.81	45.89	45.62	44.90	43.22	46.01	45.86
		$\tau = 3$	53.43	52.05	55.13	54.73	50.33	48.60	50.93	50.10
		$\tau = 4$	62.25	60.93	64.10	63.88	59.87	58.10	60.92	60.43
		$\tau = 5$	71.06	69.60	72.74	72.80	66.93	65.19	68.03	67.80
$NAPWMIC_{empirical}$	ACTS	$\tau = 2$	51.19	63.16	68.65	68.55	58.84	68.08	72.98	72.93
		$\tau = 3$	52.62	76.77	80.33	80.70	60.93	75.16	78.31	78.42
		$\tau = 4$	52.89	83.53	85.43	86.46	71.07	86.78	89.10	89.27
		$\tau = 5$	60.98	91.45	92.92	93.52	68.65	90.29	91.65	92.35
	PICT	$\tau = 2$	66.64	63.57	69.14	69.03	70.52	68.04	72.95	72.95
		$\tau = 3$	78.54	77.26	80.85	81.06	78.38	76.56	79.37	79.43
		$\tau = 4$	86.37	85.76	88.19	88.69	87.44	86.36	88.72	88.95
		$\tau = 5$	91.75	91.21	92.74	93.33	91.99	91.16	92.68	93.10

Table 8 Comparison of different metrics for an interaction test sequence S.

Metric	Requirements		Factors considered	
	Is S a λ-wise covering array?	Are all test cases in S executed?	Strength π	Rate of π-wise interaction coverage
$APCC_\lambda(S)$	yes	yes	$\pi = \lambda$	yes
$APCCV_\lambda(S)$	yes	yes	$\pi = \lambda$	yes
$NAPCCV_\lambda(S)$	no	no	$\pi = \lambda$	yes
$APSS(S)$	no	no	$\pi = 1, 2, \cdots, (k-1)$	no
$APWMIC(S)$	no	yes	$\pi = 1, 2, \cdots, (k-1)$	yes
$NAPWMIC(S)$	no	no	$\pi = 1, 2, \cdots, (k-1)$	yes

action coverage at different strengths, but during the prioritization process it only considers a single strength to guide the selection of each next test case; similarly, `ICBP` uses strength τ to prioritize a given interaction test suite: Neither metric considers multiple strengths at the same time. Since *NAPWMIC* (both $NAPWMIC_{equal}$ and $NAPWMIC_{empirical}$) considers different interaction coverage at different strengths at the same time, therefore, `ICBP` and `IICBP` may perform similarly.

In summary, the proposed metrics (including *APSS* and *NAPWMIC*) can successfully evaluate and distinguish different interaction test sequences, and thereby evaluate different test prioritization strategies. Therefore, we conclude that the proposed metrics are both reasonable and practical.

5.3 Threats to Validity

To the best of our knowledge, there are four main potential threats to the validity of this study.

The first potential threat relates to the representativeness of the test profiles: in empirical studies we considered two test profiles derived from the real-life programs, which are widely used but limited. The second potential threat is the representativeness of the interaction test suites: we used ACTS and PICT to construct the test suites, but both of them belong to the category of greedy methods. The third potential threat is the representativeness of weight assignments for the *NAPWMIC* metric: we only used two weight assignments, both of which may be considered quite simple. The final potential threat is the representativeness of the test prioritization strategies used to prioritize the interaction test suites: we only adopted four prioritization strategies, all of which have frequently been applied in the study of constructing interaction test sequences.

Further studies will be conducted to address these potential threats: in particular, a greater number of test profiles, different interaction test suite construction tools, different weight assignments for *NAPWMIC*, and other test prioritization strategies will all be considered.

6. Comparison of Different Metrics for Interaction Test Sequences

In this section, we briefly present a comparison of the different metrics for evaluating interaction test sequences.

For an interaction test sequence S, Table 8 shows a comparison of different metrics for evaluating S from the perspectives of requirements, and factors considered. As shown in the table, it can be seen that the metrics proposed in this paper ($NAPCCV_\lambda$, *APSS*, *APWMIC*, and *NAPWMIC*) have advantages over previous metrics *APCC* and *APCCV*: either reducing requirements, or considering additional factors for the evaluation of S. We next describe each of these comparisons in turn.

1. $APCCV_\lambda(S)$ VS $APCC_\lambda(S)$: $APCCV_\lambda(S)$ is a variant of $APCC_\lambda(S)$, therefore, they are equivalent.
2. $NAPCCV_\lambda(S)$ VS $APCC_\lambda(S)$: Although both use the same factors to evaluate S, $NAPCCV_\lambda(S)$ does not have

the two requirements that $APCC_\lambda(S)$ has (that all test cases in S should be executed; and that all possible λ-wise value combinations should be covered by S).

3. $APSS(S)$ VS $APCC_\lambda(S)$: Similar to $NAPCCV_\lambda(S)$, $APSS(S)$ does not have the two requirements that $APCC_\lambda(S)$. However, $APSS(S)$ does not consider the rate of interaction coverage at a given strength; while $APCC_\lambda(S)$ does not use multiple strengths to evaluate S.
4. $APWMIC(S)$ VS $APCC_\lambda(S)$: $APWMIC(S)$ overcomes one of requirements of $APCC_\lambda(S)$, and it uses additional factors to evaluate S.
5. $NAPWMIC(S)$ VS $APCC_\lambda(S)$: $NAPWMIC(S)$ not only overcomes the two requirements of $APCC_\lambda(S)$, but also considers other factors to evaluate S that are not been used by $APCC_\lambda(S)$.

Although both $APSS(S)$ and $NAPWMIC(S)$ overcome the drawbacks associated with $APCCV(S)$ (or $APCC(S)$), $APSS(S)$ does face a couple of challenges that $NAPWMIC(S)$ does not: unlike $NAPWMIC(S)$, as shown in Table 8, $APSS(S)$ neglects the rate of interaction coverage at each strength, and, as discussed in Sect. 4.3, may fail to evaluate and compare two interaction test sequences. In practice, when choosing which of $APSS(S)$ or $NAPWMIC(S)$ to use, we suggest the following: (1) when evaluation time is limited, choose $APSS(S)$ instead of $NAPWMIC(S)$, because the calculation time for $APSS(S)$ is less than that for $NAPWMIC(S)$; (2) when time allows, use either $APSS(S)$ or $NAPWMIC(S)$; and (3) when $APSS(S)$ fails to evaluate and compare two interaction test sequences due to above reasons (See Sect. 4.3), $NAPWMIC(S)$ is a better choice.

7. Conclusion and Discussion

In this paper, to overcome some shortcomings of previous metrics [8], we have proposed several metrics to help guide prioritization of interaction test suites. According to empirical studies, it is both reasonable and practical to use the proposed metrics to evaluate and distinguish different interaction test sequences of a given interaction test suite, and thereby evaluate different prioritization strategies.

Recently, Petke *et al.* [21] presented a variant of the $APCC_\lambda$ metric, namely the *Average Percentage of Covering-array Coverage* metric, which is similar to the $APCCV_\lambda$ metric. It does not require that all possible value combinations should be covered by the interaction test sequence S. However, it still requires that all test cases in S should be executed, and only focuses on a single strength to evaluate S.

Although other factors can applied to metrics used for evaluating interaction test sequences, such as test case cost and test case weight [2]–[4], [8], in this paper we only considered the characteristics of the interaction test suite (i.e., interaction coverage information). It will be interesting to apply additional factors to the proposed metrics, so as to allow a deeper evaluation of interaction test sequences.

Acknowledgment

We would like to thank D. R. Kuhn for providing us with the ACTS tool. We would also like to thank T. Y. Chen for the many helpful discussions. This work is partly supported by the National Natural Science Foundation of China (Grant No. 61202110), and Natural Science Foundation of Jiangsu Province (Grant No. BK2012284).

References

[1] C. Nie and H. Leung, "A survey of combinatorial testing," ACM Comput. Surv., vol.43, no.2, pp.11:1–11:29, Jan. 2011.

[2] X. Qu, M.B. Cohen, and K.M. Woolf, "Combinatorial interaction regression testing: A study of test case generation and prioritization," Proc. 23rd International Conference on Software Maintenance, pp.255–264, Paris, France, 2007.

[3] R.C. Bryce and C.J. Colbourn, "Test prioritization for pairwise interaction coverage," Proc. 1st International Workshop on Advances in Model-based Testing, pp.1–7, St. Louis, Missouri, USA, 2005.

[4] R.C. Bryce and C.J. Colbourn, "Prioritized interaction testing for pairwise coverage with seeding and contraints," Information and Software Technology, vol.48, no.10, pp.960–970, 2006.

[5] R.C. Bryce and A.M. Memon, "Test suite prioritization by interaction coverage," Proc. Workshop on Domain Specific Approaches to Software Test Automation, pp.1–7, Dubrovnik, Croatia, 2007.

[6] R.C. Bryce, S. Sampath, and A.M. Memon, "Developing a single model and test prioritization strategies for event-driven software," IEEE Trans. Softw. Eng., vol.37, no.1, pp.48–64, 2011.

[7] X. Chen, Q. Gu, X. Zhang, and D. Chen, "Building prioritized pairwise interaction test suites with ant colony optimization," Proc. 9th International Conference on Quality Software, pp.347–352, Jeju, Korea, 2009.

[8] Z. Wang, L. Chen, B. Xu, and Y. Huang, "Cost-cognizant combinatorial test case prioritization," Int. J. Software Engineering and Knowledge Engineering, vol.21, no.6, pp.829–854, 2011.

[9] X. Qu, M.B. Cohen, and G. Rothermel, "Configuration-aware regression testing: an empirical study of sampling and prioritization," Proc. 17th International Symposium on Software Testing and Analysis, pp.75–86, Seattle, WA, USA, 2008.

[10] R. Huang, J. Chen, T. Zhang, R. Wang, and Y. Lu, "Prioritizing variable-strength covering array," Proc. IEEE 37th Annual Computer Software and Applications Conference, pp.502–511, Kyoto, Japan, 2013.

[11] S. Fouché, M.B. Cohen, and A. Porter, "Towards incremental adaptive covering arrays," Proc. 6th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp.557–560, Dubrovnik, Croatia, 2007.

[12] S. Fouché, M.B. Cohen, and A. Porter, "Incremental covering array failure characterization in large configuration spaces," Proc. 18th International Symposium on Software Testing and Analysis, pp.177–188, Chicago, IL, USA, 2009.

[13] R. Huang, X. Xie, D. Towey, T.Y. Chen, Y. Lu, and J. Chen, "Prioritization of combinatorial test cases by incremental interaction coverage," Int. J. Software Engineering and Knowledge Engineering, 2013, To appear.

[14] G. Seroussi and N.H. Bshouty, "Vector sets for exhaustive testing of logic circuits," IEEE Trans. Inf. Theory, vol.34, no.3, pp.513–522, 1988.

[15] S. Elbaum, A.G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," IEEE Trans. Softw. Eng., vol.28, no.2, pp.159–182, 2002.

[16] Z. Li, M. Harman, and R. Hierons, "Search algorithms for regres-

sion test case prioritization," IEEE Trans. Softw. Eng., vol.33, no.4, pp.225–237, 2007.

[17] D.R. Kuhn and M.J. Reilly, "An investigation of the applicability of design of experiments to software testing," Proc. 27th Annual NASA Goddard/IEEE Software Engineering Workshop, pp.91–95, Greenbelt, Maryland, 2002.

[18] D.R. Kuhn, D.R. Wallace, and A.M. Gallo, "Software fault interactions and implications for software testing," IEEE Trans. Softw. Eng., vol.30, no.6, pp.418–421, 2004.

[19] Y. Lei, R. Kacker, D.R. Kuhn, and V. Okun, "Ipog/ipod: Efficient test generation for multi-way software testing," Software Testing, Verification, and Reliability, vol.18, no.3, pp.125–148, 2008.

[20] J. Czerwonka, "Pairwise testing in real world: Practical extensions to test case generators," Proc. 24th Pacific Northwest Software Quality Conference, pp.419–430, Portland, Oregon, USA, 2006.

[21] J. Petke, S. Yoo, M.B. Cohen, and M. Harman, "Efficiency and early fault detection with lower and higher strength combinatorial interaction testing," Proc. 12th Joint Meeting on European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp.26–36, Saint Petersburg, Russia, 2013.

Rubing Huang was born in 1984. He received the B.E. degree in Computer Science and Technology from Zhengzhou University of Light Industry (China), in 2007. He is an M.E.-Ph.D. candidate in Computer Science and Technology at Huazhong University of Science and Technology (China). His current research interests include software testing and software maintenance. He is a member of ACM, IEEE, IEICE, and IEEE Communications Society.

Dave Towey was born in 1974. He received the B.A. and M.A. degrees in Computer Science, Linguistics and Languages from the University of Dublin, Trinity College, in 1997 and 2000, respectively, and received the Ph.D. degree in Computer Science from the University of Hong Kong in 2006. He is an assistant professor in the Division of Computer Science, The University of Nottingham Ningbo China, prior to which he was with Beijing Normal University–Hong Kong Baptist University: United International College, China. His current research interests include software testing, software design, and technology in education. He is a member of both the IEEE and the ACM.

Jinfu Chen was born in 1978. He received the B.E. degree from Nanchang Hangkong University, Nanchang, China, in 2004, and Ph.D. degrees from Huazhong University of Science and Technology, Wuhan, China, in 2009 respectively, both in Computer Science. He is currently an associate professor in the School of Computer Science and Telecommunication Engineering, Jiangsu University. His major research interests are software engineering, services computing and information security. He is a member of ACM, IEEE, and China Computer Federation.

Yansheng Lu was born in 1949. He received the B.E. in Mathematics Science from Wuhan University, Wuhan, China, in 1985. He is currently a professor and Ph.D. supervisor in the School of Computer Science and Technology, Huazhong University of Science and Technology. His major research interests are software engineering, database system and data mining.