2012 IEEE 36th International Conference on Computer Software and Applications

Adaptive Random Test Case Generation for Combinatorial Testing

Rubing Huang*, Xiaodong Xie†, Tsong Yueh Chen‡, Yansheng Lu*
*College of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
†College of Computer Science and Technology, Huaqiao University, Xiamen 362021, China
‡Faculty of Information and Communication Technologies, Swinburne University of Technology, Hawthorn 3122, Australia
Email: huangrubing1984@smail.hust.edu.cn; xiaodongxie@hqu.edu.cn; tychen@swin.edu.au; lys@mail.hust.edu.cn

***Abstract*—Random testing (RT), a fundamental software testing technique, has been widely used in practice. Adaptive random testing (ART), an enhancement of RT, performs better than original RT in terms of fault detection capability. However, not much work has been done on effectiveness analysis of ART in the combinatorial test spaces. In this paper, we propose a novel family of ART-based algorithms for generating combinatorial test suites, mainly based on fixed-size-candidate-set ART and restricted random testing (that is, ART by exclusion). We use an empirical approach to compare the effectiveness of test sets obtained by our proposed methods and random selection strategy. Experimental data demonstrate that the ART-based tests cover all possible combinations at a given strength more quickly than randomly chosen tests, and often detect more failures earlier and with fewer test cases in simulations.**

***Keywords*-Software testing; combinatorial testing; random testing; adaptive random testing; restricted random testing; uncovered t-wise combinations distance.**

I. Introduction

Software testing, a major approach to software quality assurance, is widely considered as a crucial activity throughout the process of software development. Many software testing methods have been designed at achieving high testing effectiveness and reducing cost of software testing by selecting test cases appropriately. Since system under test (SUT) generally has an extremely large number of test inputs, it is reasonable and practical to choose a portion of the test inputs as test cases such that software failures can be effectively identified. Many software testing approaches have been proposed to guide the test case selection.

Random testing (RT) is a fundamental testing strategy, by simply selecting test cases in a random manner from the input domain. RT allows statistical quantitative estimation of software's reliability and has efficient algorithms associated with the generation of random test cases [1]. RT has been popularly used in different scenarios to detect software failures, such as SQL database systems [2], UNIX utility programs [3], etc.

It has been found however, that program inputs causing software to exhibit failure behaviors, namely failure-causing inputs, tend to be clustered into contiguous failure regions. Therefore, it is intuitively appealing that failures in the SUT can be more effectively and efficiently identified by requiring the test cases to be more evenly distributed, and far separated from each other. On the basis of such an intuition, an innovative testing methodology proposed by Chen et al. [4], namely, adaptive random testing (ART). The variety of ART algorithms have been developed to fulfill an even spread of test cases based on different notions, such as ART by distance [5], ART by exclusion [6], ART by perturbation [7], ART by partitioning [8], ART based on evolutionary search algorithms [9], etc. In fact, ART achieves test case diversity within the subset of test cases executed at any one time [10].

ART has been successfully used in numeric programs and in some non-numeric programs such as object-oriented programs [11]. However, not much work has been done on the effectiveness of ART test suites of the combinatorial test spaces (that is, the inputs consist of parameters, respective values, and constraints on value combinations). As we know, combinatorial testing (CT), a black-boxing testing technique, is widely used in combinatorial test spaces to generate an effective test suite (named CT test suite) for detecting interaction faults that are triggered by interactions among parameters in the SUT [12–14]. A large number of CT algorithms and tools have been proposed, such as AETG [15], IPO [16], EXACT [17], GA-based construction methods [18], etc. (see [19] for more details). CT actually provides a tradeoff between testing effectiveness and efficiency. For example, given a SUT with k parameters, 2-wise testing (or pairwise testing) only test all the 2-wise combinations of parametric values rather than k-wise.

In this paper, we propose an alternative methodology based on ART to generate CT test suites, which actually achieves the mechanism of CT. It involves two algorithms according to two implementations of ART: (1) fixed-size-candidate-set ART (abbreviated as FSCS) [5] and (2) restricted random testing (RRT) [6]. We also investigate the effectiveness of ART in combinatorial test spaces by analyzing the corresponding generated test suites in terms of their sizes and failure detection capabilities. The empirical results illustrate that CT test suites created by ART-based approaches cover all possible combinations at a given strength more quickly than test suites generated by random selection strategy, and also often identify more failures earlier and with fewer test cases than RT.

0730-3157/12 $26.00 © 2012 IEEE
DOI 10.1109/COMPSAC.2012.15

The remainder of this paper is organized as follows: Section 2 briefly introduces some background information on CT and ART, and gives a distance measure to evaluate test cases of the combinatorial test spaces. Section 3 proposes a new family of ART-based algorithms for CT test suite generation. Section 4 evaluates the effectiveness of CT test suites generated by our new methods, as compared with random selection technique. Section 5 presents the discussion and conclusion.

II. Background

In this section, some background about CT and ART will be presented. And a distance measure for test cases of the combinatorial test spaces will also be explained in this section.

A. Combinatorial Testing

Combinatorial testing (CT) aims at covering some specific combinations of parametric values with as fewer test cases as possible, and detecting the program failures that are triggered by parameters and their interactions.

Suppose that a system under test (SUT) has k parameters (or factors) P_1, P_2, ..., P_k, which may represent user inputs or configuration parameters, and each parameter P_i has discrete valid values (or levels) from the finite set V_i. Let R be the set of interaction relations existing among parameters, and C be the set of constraints on parameter value combinations.

Definition 1 (Test profile). The test profile denoted as $TP(k; |V_1||V_2|...|V_k|; R; C)$ is about the information on a combinatorial test space of the SUT, including k parameters, relative $|V_i|(i = 1, 2, ..., k)$ values for the i-th parameter, interaction relations R among parameters, and constraints C on combinations of parametric values.

Each interaction relation in R is related to the same number of parameters and the constraint set C is not taken into consideration in this paper (that is, $C = \emptyset$), unless explicitly stated. Therefore, the test profile is abbreviated as $TP(k; |V_1||V_2|...|V_k|)$.

To illustrate some notions and definitions in detail, we give an example of product-assessment system (PAS) with the input parameters shown in Table I. In PAS, there are four parameters and three values for each parameter, that is, $k = 4$, $V_1 = \{Blue, Green, black\}$, $V_2 = \{Big, Medium, Small\}$, $V_3 = \{Circle, Square, Triangle\}$, $V_4 = \{China, USA, Japan\}$, $|V_1| = |V_2| = |V_3| = |V_4| = 3$, and the test profile for the PAS is expressed as $TP(4; 3^4)$.

Table I
A Test Profile for the Product-Assessment System

P_1:Color	P_2:Size	P_3:Sharp	P_4:Country
Blue	Big	Circle	China
Green	Medium	Square	USA
Black	Small	Triangle	Japan

Table II
$MCA(9; 2, 4, 3^4)$ for the PAS

Test	P_1:Color	P_2:Size	P_3:Sharp	P_4:Country
t_1	Blue	Big	Circle	China
t_2	Blue	Medium	Square	Japan
t_3	Blue	Small	Triangle	USA
t_4	Green	Big	Triangle	Japan
t_5	Green	Medium	Circle	USA
t_6	Green	Small	Square	China
t_7	Black	Big	Square	USA
t_8	Black	Medium	Triangle	China
t_9	Black	Small	Circle	Japan

Definition 2 (Test case). For a test profile denoted by $TP(k; |V_1||V_2|...|V_k|)$, a k-tuple $(t_1, t_2, ..., t_k)$ is a test case for SUT, where $t_i \in V_i$ $(i = 1, 2, ..., k)$.

For instance, a 4-tuple $(Green, Big, Triangle, China)$ is a test case for PAS. There are totally $3^4 = 81$ test cases in the PAS.

In CT, a mixed-level covering array (MCA) [20] usually stands for a CT test suite.

Definition 3 (Mixed-level covering array). An $N \times k$ matrix is called a mixed-level covering array (MCA) denoted as $MCA(N; t, k, |V_1||V_2|...|V_k|)$ for the test profile $TP(k; |V_1||V_2|...|V_k|)$ when this matrix satisfies the following properties: (1) each column $i(1 \le i \le k)$ contains only elements from the set V_i; (2) the rows of each $N \times t$ submatrix cover all t-tuples (referred as t-wise combinations) of values from the t columns at least once.

In the MCA, t is called strength, and each row stands for a test case while each column represents each parameter of the SUT. For instance, Table II presents an $MCA(9; 2, 4, 3^4)$ for the PAS. The MCA has only nine test cases and covers all pairwise combinations of parametric values.

To reduce the cost of CT, many researchers have focused on algorithms to generate the optimal test suite with the minimal number of test cases. But unfortunately, it has been proved that the problem of the MCA generation is NP-Complete [21]. However, there are many approaches for constructing the MCAs and tools have been developed in recent years. Most of the approaches could be categorized into four classes: greedy algorithm, heuristic search algorithm, mathematic design method, and recursive algorithm [19].

B. Adaptive Random Testing

Adaptive random testing (ART) [4] enhances random testing (RT) by even-spreading test cases across the whole input space. There are many different implementations of ART, such as fixed-sized-candidate-set ART (FSCS) [5], restricted random testing (RRT) [6], evoluationary ART [9],

etc. In this paper, FSCS and RRT are used in combinatorial test case generation.

1) FSCS: The fixed-size-candidate-set ART (FSCS) [5] implements ART based on distance, which makes use of two sets of test cases, namely the candidate set and the executed set. The candidate set is a set of test inputs that are randomly chosen from the input domain; while the executed set is a set of test cases that have been executed but without exhibiting any failure. The executed set is initially empty and the first element is randomly selected from the input domain. The executed set is then incrementally updated with the selected element from the candidate set until a failure is revealed. From the candidate set, the element that is farthest away from all test cases in the executed set, is selected as the next test case.

2) RRT: Restricted random testing (RRT) [6] is ART by exclusion, and implements the notion of exclusion as follows. The first test case is randomly selected from the input domain according to a uniform distribution. After generating $m(m \geq 1)$ executed test cases, RRT defines an exclusion region around each executed test case. The size of each exclusion region is represented as $R \cdot d/m$, where R is referred to as the target exclusion ratio that should be designed by testers in advance, and d is expressed as the size of the input domain (which refers to the hyper-volume of the hyper-rectangular input domain). According to a uniform distribution, random test case candidates are chosen successively from the input domain until one candidate is outside all exclusion regions, which will then be selected as the next test case. In other words, RRT generates the next test case of which the distance against each test in the executed test set is greater than a certain value r (for example, r can be written as $\sqrt[3]{3R \cdot d/4\pi m}$ in a three-dimensional input domain).

To implement strategies of "*farthest away*" in FSCS and "*outside the exclusion region*" in RRT to determine the next test case, it is necessary to define distance measures for candidates (of how different candidates are) in advance. There are some definitions of distance for test inputs in ART algorithms, such as Euclidean distance for numeric inputs, object distance for object-oriented inputs [11, 22], etc. Due to the characteristics of the combinatorial test space, we should redefine distance for tests, which will be presented in the next subsection.

C. A Distance Measure for Tests in Combinatorial Spaces

Generally, a distance measure is often used to calculate the difference between two test inputs. For example, in an n-dimensional input domain, for two test inputs $a = (a_1, a_2, ..., a_n)$ and $b = (b_1, b_2, ..., b_n)$, Euclidean distance between a and b can be computed as $\sqrt{\sum_{i=1}^{n}(a_i - b_i)^2}$. However, for $TP(k; |V_1||V_2|...|V_k|)$, its parameters and corresponding values are finite and discrete. The distance for a candidate test case is relevant to the difference of its parametric values against those of each test case in the executed test suite. Besides, parameter value combinations at a given strength t should be taken into account. Hence, in this paper, a distance measure named uncovered t-wise combinations distance is used to measure the difference for test inputs of the combinatorial test spaces.

Definition 4 (Uncovered t-wise combinations distance). Uncovered t-wise combinations distance (abbreviated as the UCD) is defined as the number of t-wise combinations of parametric values of a test case that have not been covered by the executed test cases.

That is to say, for the executed test set $E = \{t_1, t_2, ..., t_s\}$ and a candidate test case c_h, let t be strength and $EC_t(t_i)$ be a set of t-wise combinations covered by the test case t_i. Therefore, $UCD_t(c_h, E) = |EC_t(c_h) \setminus \bigcup_{i=1}^{s} EC_t(t_i)|$.

Intuitively speaking, UCD is used to measure a test input against a test set rather than another test input. To illustrate this distance measure in detail, we present an example. Given an executed test suite $E = \{t_1 = (Blue, Big, Circle, China), t_2 = (Blue, Big, Circle, USA)\}$ and a candidate test case $c = (Black, Small, Circle, China)$ in the PAS, the uncovered 2-wise combinations distance of c is 5 (that is, $UCD_{t=2}(c, E) = 5$) while its uncovered 3-wise combinations distance is 4 (that is, $UCD_{t=3}(c, E) = 4$).

III. ART-Based CT Test Suite Generation

In this paper, we propose a new family of methods to

Input: A test profile, $TP(k; |V_1||V_2|...|V_k|)$, and strength t.
Output: An $MCA(N; k, t, |V_1||V_2|...|V_k|)$, that is, a CT test suite E.

1: Set $E = \{\}$, $SC = \{$ all possible t-wise combinations derived from $TP(k; |V_1||V_2|...|V_k|)$ $\}$, and candidate set size, r.
2: Randomly generate a test case t_c, according to $TP(k; |V_1||V_2|...|V_k|)$ by the uniform distribution.
3: Store t_c into E, and remove all t-wise combinations covered by t_c from SC.
4: **while** ($SC \neq \emptyset$)
5: Set $C = \{\}$.
6: Randomly generate r test inputs according to $TP(k; |V_1||V_2|...|V_k|)$ by the uniform distribution, and add them into C.
7: Find an element $c_h (1 \leq h \leq r)$ from C, which has the longest UCD to E.
8: Set $t_c = c_h$.
9: Store t_c into E, and remove all t-wise combinations covered by t_c from SC.
10: **end_while**
11: Return E, and exit.

Figure 1. Pseudo-code of the FSCS-CT algorithm

generate CT test suites (MCAs) based on the concept of ART, and we call it ART-CT. We mainly take advantage of two implementations of ART, which are FSCS [5] and RRT [6]. Detailed FSCS-based and RRT-based algorithms of generating CT test suites are shown as follows.

A. FSCS-Based CT Test Suite Generation

As shown in Figure 1, the FSCS-based algorithm of CT test suite generation (abbreviated as the FSCS-CT) is a one-test-at-a-time approach. Similar to FSCS [5], it also makes use of two sets, the candidate set (C) and the executed set (E). C contains r test inputs randomly selected from the input domain, from which the next test case will be chosen; while E stores all already executed test cases. For ease of description, let $E = \{t_1, t_2, ..., t_s\}$ be the executed set, $C = \{c_1, c_2, ..., c_r\}$ be the candidate set, and strength be t. The criterion is to select the element c_h as the next test case such that for all $j \in 1, 2, ..., r$, $UCD_t(c_h, E) \geq UCD_t(c_j, E)$. The process runs until all t-wise value combinations of parameters are covered by test cases in the executed set.

B. RRT-Based CT Test Suite Generation

Input: A test profile, $TP(k; |V_1||V_2|...|V_k|)$, and strength t.
Output: An $MCA(N; k, t, |V_1||V_2|...|V_k|)$, that is, a CT test suite E.

1: Set the size of exclusion regions r, $E = \{\}$, and $SC = \{$ all possible t-wise combinations derived from $TP(k; |V_1||V_2|...|V_k|)$ $\}$.
2: Randomly generate a test case c_h, according to $TP(k; |V_1||V_2|...|V_k|)$ by the uniform distribution.
3: Store c_h into E, and remove all t-wise combinations covered by c_h from SC.
4: **while** ($SC \neq \emptyset$)
5: Adjust r according to generated test cases.
6: Randomly generate a test input $c_h^{'}$ according to $TP(k; |V_1||V_2|...|V_k|)$ by the uniform distribution.
7: **while** ($UCD_t(c_h^{'}, E) \leq r$)
8: Randomly generate a test input $c_h^{''}$, according to $TP(k; |V_1||V_2|...|V_k|)$ by the uniform distribution.
9: Set $c_h^{'} = c_h^{''}$.
10: **end_while**
11: Set $c_h = c_h^{'}$.
12: Store c_h into E, and remove all t-wise combinations covered by c_h from SC.
13: **end_while**
14: Return E, and exit.

Figure 2. Pseudo-code of the RRT-CT algorithm

Similar to FSCS-CT, the RRT-based algorithm of CT test suite generation (RRT-CT) also generates test cases one-test-at-a-time. When RRT-CT generates a new test case, the executed set (E) and the numeric value (r) are required. E stores all already executed test cases; while the r stands for the size of exclusion regions. The detailed algorithm is illustrated in Figure 2. For an illustrative purpose, suppose $E = \{t_1, t_2, ..., t_s\}$ be the executed set and strength be t. The criterion is to select the randomly chosen candidate c_h as the next test case such that $UCD_t(c_h, E) > r$. The process runs until test cases in the executed set cover all possible t-wise value combinations of parameters.

The FSCS-CT and RRT-CT algorithms take advantage of the principle of ART, so as to implement the mechanism of CT. Both of them involve the intuition of ART, which means that they achieve the diversity of test cases throughout the combinatorial input domain.

IV. Empirical Studies

In this section, some empirical investigations were conducted to compare the effectiveness of ART-CT algorithms (including FSCS-CT and RRT-CT) against RT, which present a assessment of whether FSCS-CT and RRT-CT are effective. To evaluate CT test suites (MCAs), many metrics have been proposed, such as overlap of the MCA [23], the size of the MCA, etc. In this paper, two metrics are used for assessing MCAs generated by FSCS-CT and RRT-CT algorithms compared to random selection technique: (1) the size of the MCA, (2) the rate of failure detection. Both of them take the number of test cases into account. The MCA size is to evaluate the number of test cases of which the whole possible t-wise combinations are covered while the failure detection rate is to assess the number of test cases of which the failures are detected. The main reason is that testers may only run the partial test cases in the MCA in reality due to limited resources.

A. Experimental Setup

In our experiments, we designed four test profiles as four system models, so as to analyze the performance of ART-CT methods (FSCS-CT and RRT-CT). The first two test profiles are $TP(10; 2^{10})$ and $TP(10; 2^5 3^5)$ that have commonly been used in previous studies [23]. The third designed test profile (that is, $TP(7; 2^4 3^1 6^1 16^1)$) is from a real-world application, which is a real configuration model of a lexical analyzer system. To design diverse test profiles, we give the fourth test profile $TP(5; 2^2 10^2 15^1)$.

On the other hand, in the algorithm of FSCS-CT, the size of candidate set is a constant value (r) and should be assigned in advance. In [5], Chen et al. set the size of the candidate set, r, as 10 in their experiment due to not much difference in the size of test suites required to detect the first failure when $r \geq 10$. However, since the selected metrics are the MCA size and failure detection rate in our studies, the

value of r should be reset. In general, larger r means smaller size of the MCA, and may bring higher failure detection rate for the MCA. For ease of evaluation, we set $r = 10$ and $r = 50$ in our experiments, respectively.

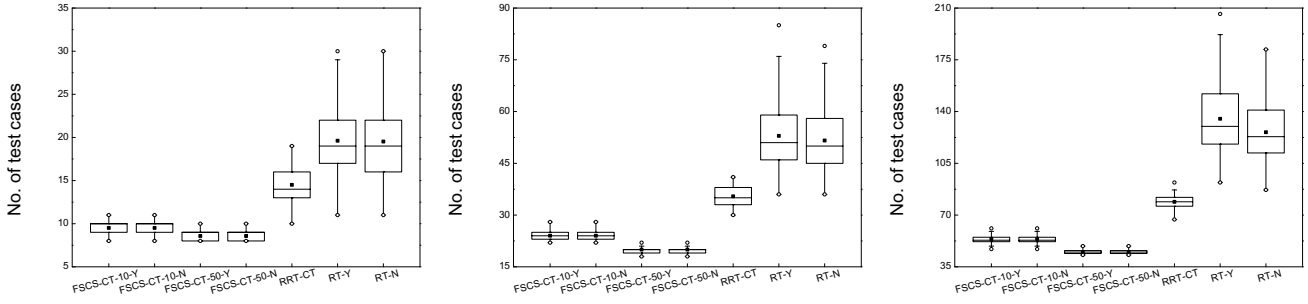
In addition, in the RRT-CT algorithm, r that defines the size of exclusion region should also be discussed. In [6], r is related with the size of the exclusion region and the number of generated test cases, and for example, in a two-dimensional input domain, r can be described as $\sqrt{A/(2m\pi)}$ where $A(A = R{\cdot}d)$ is the area of the exclusion region and m is the number of generated test cases. In other words, r varies along with the number of generated test cases. However, in RRT-CT, r is related to the number of t-wise combinations of parametric values. For ease of explanation, we take account of the simplest situation in our studies (that is, r keeps unchanged and is equal to 0), which means that the RRT-CT randomly chooses an element as the next test case such that its UCD is no less than 1. Nevertheless, RRT-CT may potentially perform better than RT with or without replacement. RT with replacement may generate many duplicated test cases while RT without replacement may choose a test input which does not cover any not yet covered t-wise combinations, as the next test case. However, this is not the case for RRT-CT, that is, test cases generated by RRT-CT are not duplicated and each of them covers at least one not yet covered t-wise combinations.

B. Sizes of MCAs Generated by ART-CT Algorithms

For four test profiles described above, we respectively generated MCAs at different strengths ($t = 2, 3, 4$) using two ART-CT algorithms (FSCS-CT and RRT-CT), and compared their sizes with those of MCAs obtained by random selection (RT).

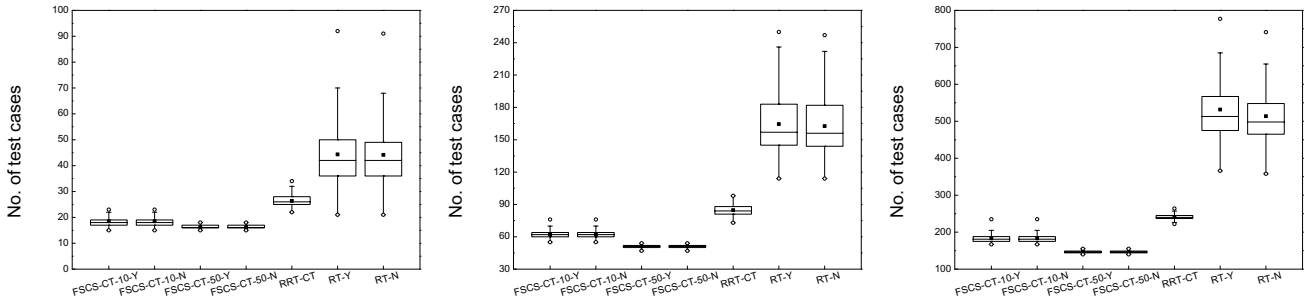
As we know, all previous ART works [5–11] are based on an assumption of selection of test cases with replacement, as no significant difference of selection with and without replacement exists in the number of test cases to detect the first failure (that is, F-measure [5]) while approaches without replacement are more time-consuming than those with replacement [24]. However, ART-CT algorithms, whether FSCS-CT or RRT-CT, take account of the number of test cases covering all possible value combinations of parameters at a given strength rather than F-measure. Therefore, we also investigated the impact of selection with and without replacement on the sizes of MCAs generated by FSCS-CT and RT (apart from RRT-CT as it does not generate duplicate test cases during the testing processing).

Since test case selection strategies, such as FSCS-CT, RRT-CT and RT involve randomization, we independently generated 100 MCAs for each technique and gathered statistics of sizes of above MCAs. For ease of expression, `FSCS-CT-10-Y` represents the FSCS-CT algorithm of the candidate size $r = 10$ with replacement



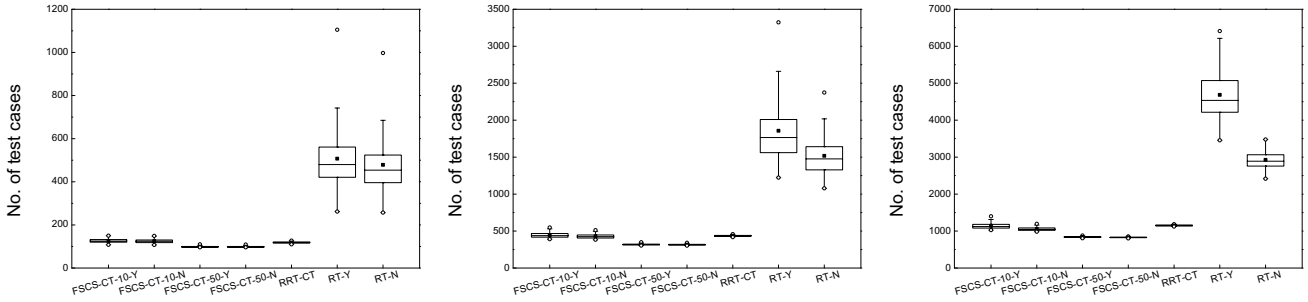
(a) $t = 2$ (b) $t = 3$ (c) $t = 4$

Figure 3. Sizes of MCAs generated by ART-CT and random selection technique on $TP(10; 2^{10})$



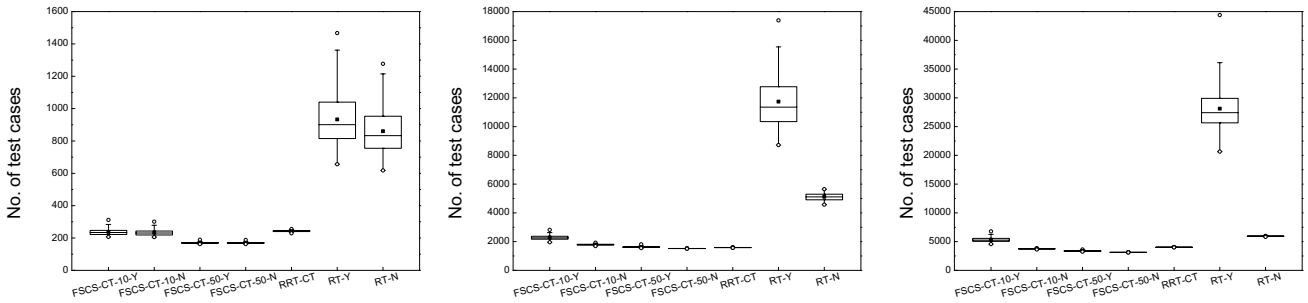
(a) $t = 2$ (b) $t = 3$ (c) $t = 4$

Figure 4. Sizes of MCAs generated by ART-CT and random selection technique on $TP(10; 2^5 3^5)$



(a) $t = 2$ (b) $t = 3$ (c) $t = 4$

Figure 5. Sizes of MCAs generated by ART-CT and random selection technique on $TP(7; 2^4 3^1 6^1 16^1)$



(a) $t = 2$ (b) $t = 3$ (c) $t = 4$

Figure 6. Sizes of MCAs generated by ART-CT and random selection technique on $TP(5; 2^2 10^2 15^1)$

while `FSCS-CT-10-N` stands for that without replacement. `FSCS-CT-50-Y` and `FSCS-CT-50-N` can be explained as the FSCS-CT algorithm of $r = 50$ with and without replacement; while `RT-Y` and `RT-N` are denoted for RT with and without replacement respectively. Figures 3-6 summarize the sizes of MCAs generated by different techniques on the above four designed test profiles.

Based on the experimental data, we have the following observations.

- Compared to the selection of test cases with replacement, the selection without replacement has no impact or little impact on the sizes of MCAs generated by the FSCS-CT algorithm; while it significantly reduces the sizes of random MCAs, especially for some test profiles. Taking $TP(5; 2^2 10^2 15^1)$ at strength $t = 4$ for example, there are at most 6000 (=$2^2 \times 10^2 \times 15$) distinct test cases. However, as shown in Figure 6(c), RT with replacement requires 20644 to 44398 test cases as many of these test cases are duplicated. From the perspective of the number of generated test cases, RT with replacement performs much poorer than exhaustive testing.
- The sizes of MCAs of different strengths ($t = 2, 3, 4$) generated by ART-CT algorithms are much smaller than those of MCAs created by RT on each test profile, which is also applicable for the comparison of ART-CT algorithms with replacement and RT without replacement. In other words, ART-CT MCAs at strength t cover all possible t-wise ($t = 2, 3, 4$) combinations of parameter values quickly than random MCAs.
- The FSCS-CT method performs slightly better than the RRT-CT algorithm in terms of the sizes of corresponding MCAs.
- The size of the candidate set r in FSCS-CT influences sizes of the generated MCAs. Obviously, the sizes of MCAs generated by FSCS-CT are smaller when r is larger.

The first observation can be explained as follows. In FSCS-CT, if an element chosen from the candidate set C as the next test case is involved in the executed set E, none of elements in C covers a uncovered combination (or combinations) of parametric values at a given strength, but not vice versa. In other words, a candidate from C may not be an element in E even though it covers no uncovered combinations. However, RT consists of no more than repeatedly choosing test cases randomly from the combinatorial test spaces, which may lead to generate a great number of duplicate test cases. In summary, FSCS-CT has fewer duplicate test cases than RT, and the selection of test cases without replacement should be used in combinatorial test spaces even though it is generally more expensive than that with replacement.

The second and third observations are explained as follow-

ing. The ART-CT algorithms (both FSCS-CT and RRT-CT) select the next test case that covers uncovered combinations of a given strength as much as possible; while RT simply generates test cases at random from combinatorial test spaces. As a result, the ART-CT MCAs achieve test cases more diversely than RT MCAs. On the other hand, the FSCS-CT algorithm selects an element from the candidate set as the next test case such that it covers the largest number of uncovered combinations of parameter values at the given strength while RRT-CT randomly chooses an element as the next test case such that it covers at least a uncovered parameter value combination. Hence, FSCS-CT performs better than RRT-CT in terms of the size of the MCA.

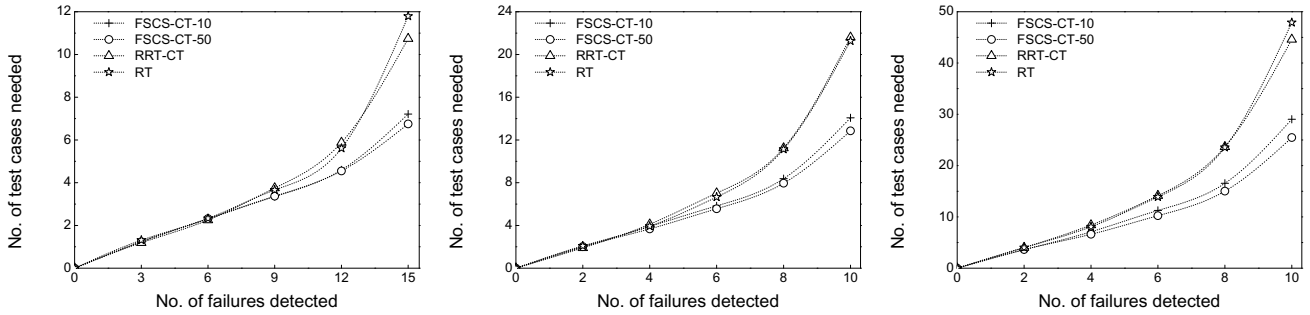
The forth observation is consistent with the intuitive expectation. It has been observed by Chen et al. [5] that when the size of the candidate set is high, the original FSCS algorithm performs well.

C. Failure Detection Capabilities of ART-CT Algorithms

We conducted four simulations using the above four designed test profiles to simulate as four system models, so as to analyze the rate of failure detection of ART-CT algorithms (including FSCS-CT and RRT-CT). For each simulation, the number of 2-wise through 4-wise failures are generated in a random manner. For all techniques of test case generation discussed in this subsection, test cases are generated for testing until all simulated failures are detected.

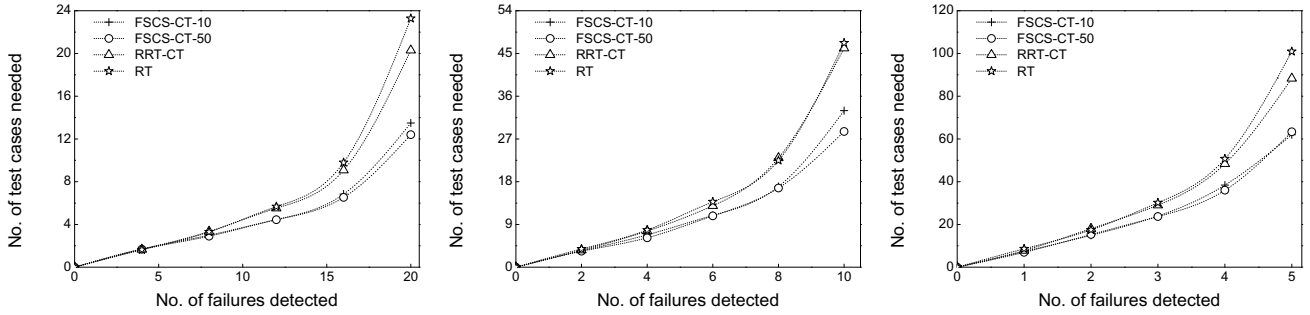
As for the distribution in the number of failures in each simulation, we arrange some failures at lower strengths according to some results reported in [12, 13]. For example, in [13], Kuhn et al. investigated interaction failures by analyzing the faults reports of several software projects, and concluded that 6% to 47% of failures are 2-wise, 2% to 19% of failures are 3-wise, 1% to 7% are 4-wise failures, and even fewer are failures beyond 4-wise interactions. Hence, we distribute fifteen 2-wise, ten 3-wise, and ten 4-wise interaction failures for $TP(10; 2^{10})$ while we distribute twenty 2-wise, ten 3-wise, and five 4-wise interaction failures for $TP(10; 2^5 3^5)$. In addition, thirty 2-wise, twenty 3-wise and fifteen 4-wise interaction failures are distributed for $TP(7; 2^4 3^1 6^1 16^1)$; while fifty 2-wise, twenty-five 3-wise and fifteen 4-wise interaction failures are distributed for $TP(5; 2^2 10^2 15^1)$. On the other hand, since all test case generation techniques involve some randomness and the interaction failures are randomly generated, we ran each simulation 100 times for each strategy of generating test cases and reported the average of the results.

As discussed in previous subsections, approaches without replacement rather than those with replacement are reasonable in combinatorial test spaces. Hence, we decide to carry out our simulations for strategies of test case selection without replacement. For ease of description, `FSCS-CT-10`



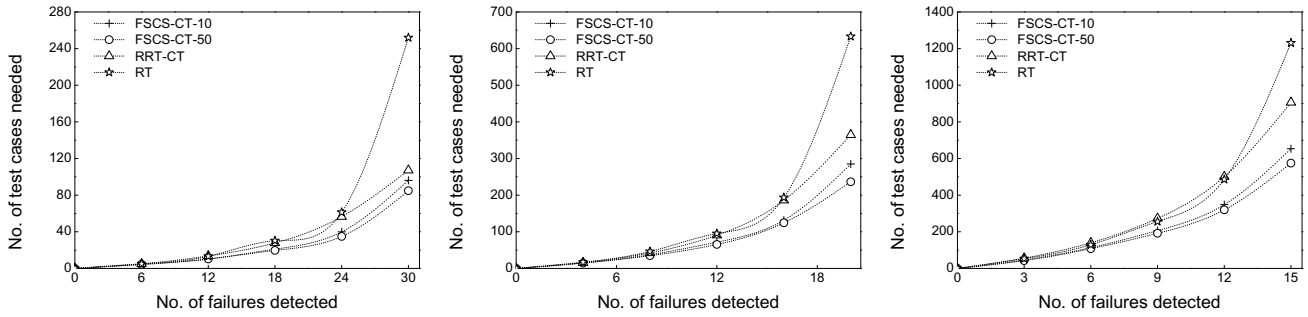
(a) 15 2-wise failures (b) 10 3-wise failures (c) 10 4-wise failures

Figure 7. Rate of interaction failure detection on $TP(10; 2^{10})$



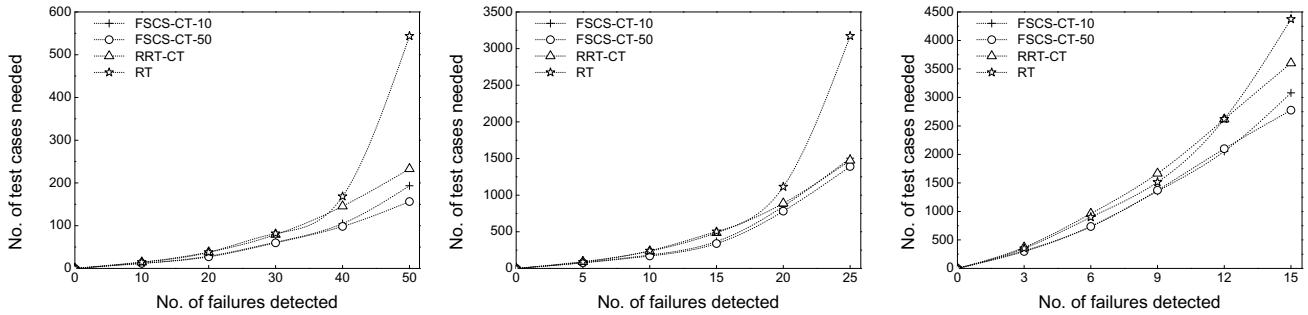
(a) 20 2-wise failures (b) 10 3-wise failures (c) 5 4-wise failures

Figure 8. Rate of interaction failure detection on $TP(10; 2^5 3^5)$



(a) 30 2-wise failures (b) 20 3-wise failures (c) 15 4-wise failures

Figure 9. Rate of interaction failure detection on $TP(7; 2^4 3^1 6^1 16^1)$



(a) 50 2-wise failures (b) 25 3-wise failures (c) 15 4-wise failures

Figure 10. Rate of interaction failure detection on $TP(5; 2^2 10^2 15^1)$

represents the FSCS-CT algorithm of the candidate set size $r = 10$, `FSCS-CT-50` stands for FSCS-CT of $r = 50$ without replacement, `RRT-CT` represents the RRT-CT algorithm, and `RT` stands for random test selection. Figures 7-10 report four simulation results on FSCS-CT ($r = 10$ and $r = 50$), RRT-CT and RT, respectively.

Based on simulation results, we observe the followings.

- In FSCS-CT, the performance of `FSCS-CT-50` is better than that of `FSCS-CT-10`, which means that the failure detection capability of FSCS-CT depends on the size of the candidate set.
- According to the number of test cases required to identify all injected interaction failures, FSCS-CT has the best performance for each test profile at different strengthes, followed by RRT-CT. RT is much less effective. Taking $TP(7; 2^4 3^1 6^1 16^1)$ for example, the 3-wise interaction failures are triggered in 286 tests with `FSCS-CT-10`, 237 tests with `FSCS-CT-50`, 365 tests with RRT-CT, and 634 tests with RT. The `FSCS-CT-10` identifies all 4-wise failures in 654 test cases, `FSCS-CT-50` finds all in 575 tests, RRT-CT triggers all in 907 tests; however, RT finds all 4-wise failures in 1231 test cases.
- From the perspective of rate of failure detection, FSCS-CT is quickest for 2-wise, 3-wise, and 4-wise interaction failures of each test profile while RT is much less effective. As for RRT-CT, it performs less than FSCS-CT, and better than RT in most cases. In other words, ART-CT algorithms often catch more failures earlier with the smaller number of test cases than random selection strategy.

The above three observations are consistent with the previous observations reported in the last subsection. They have been observed that ART-CT algorithms distribute test cases more evenly than RT, which is illustrated not only in the sizes of their generated MCAs but also in their failure detection capabilities. On the other hand, the failure detection effectiveness of RRT-CT is less than FSCS-CT, as the exclusion region size r is constant and is equal to 0. If r can be modified according to the number of already generated test cases or the number of uncovered combinations at a given strength, RRT-CT may perform better.

V. Discussion and Conclusion

Adaptive random testing (ART) [4] was proposed to enhance the failure detection capability of random testing (RT) by evenly spreading test cases throughout the input domain, and has been demonstrated that its effectiveness significantly outperforms that of RT in numeric areas [5–9] and in some non-numeric areas such as object-oriented programs [11]. However, the effectiveness of ART for the

combinatorial test spaces has not been reported. In this paper, we proposed a novel methodology (ART-CT) in order to generate combinatorial test suites according to the principles of ART, aiming at achieving the mechanism of combinatorial testing (CT). It makes use of two versions of ART named fixed-size-candidate-set ART (FSCS) [5] and restricted random testing (RRT) [6] respectively. Therefore, the ART-CT strategy is divided into two categories: (1) the FSCS-based algorithm CT test suite generation (FSCS-CT), and (2) the RRT-based algorithm of generating CT test suites (RRT-CT). We also investigated CT test suites generated by FSCS-CT and RRT-CT algorithms, as compared with random test suites. Our experimental results showed that ART-CT approaches (both FSCS-CT and RRT-CT) not only bring a higher rate in covering all possible combinations at a given strength, but also reveal more failures earlier with fewer test cases than RT.

The original greedy algorithms for CT test suite generation, such as AETG [15] and TCG [25], also make use of candidate tests and then select one of the candidates as the next test case until chosen test cases cover all combinations of parameter values at a given strength. However, their strategies of constructing candidate tests differ from the FSCS-CT algorithm. Taking AETG [15] for example, let a test profile be $TP(k; |V_1||V_2|...|V_k|)$, strength be t, and k parameters be P_1, P_2, ..., P_k respectively. The AETG algorithm firstly selects a parameter P_i and a value v from V_i for P_i such that parameter value appears in the largest number of uncovered t-wise combinations, and then sets $f_1 = P_i$ while the remaining parameters are chosen in a random order (Let a order of parameters be f_1, f_2, ..., f_k). After that, a value is assigned for each parameter (except f_1) according to the above order. To achieve the parameter value assignment, a value is chosen for the next parameter such that it and already assigned parameter values cover the largest number of uncovered t-wise combinations. In contrast with AETG, FSCS-CT builds the candidates in a random manner. Hence, the FSCS-CT algorithm is much less expensive than AETG with respect to the candidate test construction.

Bryce et al. [23] recently proposed a technique namely *adaptive distance-based testing*, where Hamming distance and uncovered combinations distance are used to generate CT test suites. During the process of generating the next test case in *adaptive distance-based testing*, parameters are ordered at random, and each parameter is assigned to a value of which the distance (Hamming distance or uncovered combinations distance) against the previously generated test cases is maximal. However, our ART-CT methods select the next test case by directly calculating the distance for each candidate test input rather than computing the distance for each parameter to generate the next test case. Similar to *adaptive distance-based testing*, ART-CT is not a static strategy of generating CT test suites, which means that testers do not require to run an entire test set. Instead, test cases generated by ART-CT are adaptive. Besides, ART-CT is also flexible for changing systems, which means that it can generate the next test case adaptively when system parameters are added, removed, or temporarily unavailable. In other words, testers do not need to regenerate a new test suite for testing when system changes.

The original ART for non-numeric programs and ART-CT have the same mechanism of constructing parameters and their values for each program [10]. Both of them are based on the concepts of categories and choices proposed by Ostrand et al. [26] for the *category-partitioning strategy*. Moreover, they both are ART-based, which means that they have the similar procedure of generating test cases. However, ART-CT aims at covering all possible combinations of parametric values at a given strength as soon as possible; while the original ART for non-numeric areas proposed in [10] is to detect the first failure as soon as possible. Their distance strategies for choosing the next test case are different.

As a study, we only proposed CT test suite generation based on FSCS [5] and RRT [6]. Apparently, various ART algorithms can also be used to construct test cases in the combinatorial test spaces. It is worthwhile to further study the applicability of different ART implementations. On the other hand, distance measures for determining the next test case are not limited to UCD, which means that there are many ways to improve the effectiveness of ART-CT approaches by introducing better distance measures. In fact, the distance measure is a *difference* measure or *dissimilarity* measure. In addition, Chen et al. [10] argued that dissimilarity plays a key role in revealing failures. To apply ART effectively in combinatorial test spaces, therefore, it is promising to further investigate alternative approaches used for measuring the dissimilarity. The other aspects of future work include evaluation of ART-CT methods against real programs and how to enhance the effectiveness of RRT-CT by introducing a novel strategy of adjusting its exclusion region size r.

ACKNOWLEDGEMENT

This work is supported in part by the National Science Foundation of China (Grant No. 61103053), and Australian Research Council.

REFERENCES

[1] P. G. Frankl, R. G. Hamlet, B. Littlewood, and L. Strigini, "Evaluating testing methods by delivered reliability," *IEEE Transactions on Software Engineering*, vol. 24, no. 8, pp. 586–601, August 1998.

[2] H. Bati, L. Giakoumakis, S. Herbert, and A. Surna, "A genetic approach for random testing of database systems," in *Proceedings of the 33rd International*

Conference on Very Large Data Bases (VLDB '07), September 2007, pp. 1243–1251.

[3] B. P. Miller, L. Fredriksen, and B. So, "An empirical study of the reliability of unix utilities," *Communications of the ACM*, vol. 33, no. 12, pp. 32–44, December 1990.

[4] T. Y. Chen, T. H. Tse, and Y. T. Yu, "Proportional sampling strategy: a compendium and some insights," *Journal of System and Software*, vol. 58, no. 1, pp. 65–81, December 2001.

[5] T. Y. Chen, H. Leung, and I. K. Mak, "Adaptive random testing," in *Proceedings of the 9th Asian Computing Science Cnference (ASIAN '04)*, December 2004, pp. 320–329.

[6] K. P. Chan, T. Y. Chen, F.-C. Kuo, and D. Towey, "A revisit of adaptive random testing by restriction," in *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC '04)*, September 2004, pp. 78–85.

[7] J. Mayer, "Lattice-based adaptive random testing," in *Proceedings of the 20th IEEE/ACM international Conference on Automated software engineering (ASE '05)*, December 2005, pp. 333–336.

[8] T. Y. Chen, R. Merkel, G. Eddy, and P. K. Wong, "Adaptive random testing through dynamic partitioning," in *Proceedings of the 4th International Conference on Quality Software (QSIC '04)*, September 2004, pp. 79–86.

[9] A. Tappenden and J. Miller, "A novel evolutionary approach for adaptive random testing," *IEEE Transactions on Reliability*, vol. 58, no. 4, pp. 619–633, December 2009.

[10] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse, "Adaptive random testing: The art of test case diversity," *Jounal of System and Software*, vol. 83, no. 1, pp. 60–66, January 2010.

[11] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "Artoo: Adaptive random testing for object-oriented software," in *Proceedings of the 30th International Conference on Software Engineering (ICSE '08)*, May 2008, pp. 71–80.

[12] D. R. Kuhn and M. Reilly, "An investigation of the applicability of design of experiments to software testing," in *Proceedings of the 27th Annual NASA Goddard/IEEE Software Engineering Workshop (SEW-27 '02)*, December 2002, pp. 91–95.

[13] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Transaction on Software Engineering*, vol. 30, no. 6, pp. 418–421, June 2004.

[14] C. Yilmaz, M. B. Cohen, and A. A. Porter, "Covering arrays for efficient fault characterization in complex configuration spaces," *IEEE Transactions on Software Engineering*, vol. 32, no. 1, pp. 20–34, January 2006.

[15] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The aetg system: An approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, no. 7, pp. 437–444, July 1997.

[16] K. C. Tai and Y. Lei, "A test generation strategy for pairwise testing," *IEEE Transaction on Software Engineering*, vol. 28, no. 1, pp. 109–111, January 2002.

[17] J. Yan and J. Zhang, "Backtracking algorithms and search heuristics to generate test suites for combinatorial testing," in *Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC '06)*, September 2006, pp. 385–394.

[18] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *Proceedings of the 28th Annual International Computer Software and Applications Conference (COMPSAC '04)*, September 2004, pp. 72–77.

[19] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computer Survey*, vol. 43, no. 2, pp. 11:1–11:29, January 2011.

[20] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn, and J. S. Collofello, "Variable strength interaction testing of components," in *Proceedings of the 27th Annual International Conference on Computer Software and Applications (COMPSAC '03)*, November 2003, pp. 413–418.

[21] G. Seroussi and N. Bshouty, "Vector sets for exhaustive testing of logic circuits," *IEEE Transactions on Information Theory*, vol. 34, no. 3, pp. 513–522, May 1988.

[22] I. Ciupa, A. Leitner, M. Oriol, and B. Meyer, "Object distance and its application to adaptive random testing of object-oriented programs," in *Proceedings of the 1st International Workshop on Random Testing (RT '06)*, July 2006, pp. 55–63.

[23] R. C. Bryce, C. J. Colbourn, and D. R. Kuhn, "Finding interaction faults adaptively using distance-based strategies," in *Proceedings of the 18th IEEE International Conference and Workshops on Engineering of Computer-Based Systems (ECBS '11)*, April 2011, pp. 4–13.

[24] T. Y. Chen and F.-C. Kuo, "Is adaptive random testing really better than random testing," in *Proceedings of the 1st international workshop on Random testing (RT '06)*, July 2006, pp. 64–69.

[25] Y. W. Tung and W. Aldiwan, "Automating test case generation for the new generation mission software system," in *Proceedings of the 2000 IEEE Aerospace Conference*, March 2000, pp. 431–437.

[26] T. J. Ostrand and M. J. Balcer, "The category-partition method for specifying and generating fuctional tests," *Communications of the ACM*, vol. 31, no. 6, pp. 676–686, June 1988.