How to Effectively Reduce Tens of Millions of Tests: An Industrial Case Study on Adaptive Random Testing

Journal:	*IEEE Transactions on Reliability*
Manuscript ID	TR-2018-168.R2
Manuscript Type:	Special Section on Adaptive Random Testing
Date Submitted by the Author:	16-Apr-2019
Complete List of Authors:	Zhang, Zhiyi; Nanjing University of Aeronautics and Astronautics, Computer Science Wang, Yabin Wang, Ziyuan Qian, Ju; Nanjing University of Aeronautics and Astronautics, College of Computer Science & Technology
Keywords:	Test Reduction, Adaptive Random Testing, Category Choices, Category Selection, Industrial System

How to Effectively Reduce Tens of Millions of Tests: An Industrial Case Study on Adaptive Random Testing

Zhiyi Zhang, Yabin Wang, Ziyuan Wang, *Member, IEEE,* and Ju Qian

***Abstract*—Running and analyzing a large number of tests in an industrial scenario is labor intensive and time consuming. Hence, it is necessary to select smaller number of tests for the purpose of cost reduction as well as fault detection. For a type of non-numeric systems, the Linear-Order Algorithm for Adaptive Random Testing ($LART$) technique was proposed by making tests evenly spread in non-numeric input domains. To further enhance $LART$ in the industrial scenarios where the number of input categories is too large, a new technique called Category Selection Based ART ($CSBART$), in which partial categories are selected to calculate tests' distances to guide $LART$, is proposed in this paper. The fault-coverage effectiveness of $CSBART$ is evaluated via an empirical study on two large scale billing systems with tens of million of test cases, and the results demonstrate the promising performance of the proposed $CSBART$. We also find that, after category selection, $CSBART$ can outperform a more complex and popular N-per Cluster Sampling (NCS) technique that uses K-Means clustering to certain extends.**

***Index Terms*—Test Reduction, Adaptive Random Testing, Category Choices, Category Selection, Industrial System.**

I. Introduction

Test case generation is one of the most vital and difficult points in software testing. For large scale and complex system, such as search engines with millions of searching records or billings system with millions of user records, it needs a large number of test cases to ensure software quality. In this way, since the large size of test numbers, conducting execution and analysis could be prohibitively expense. For example, Huawei Technologies Co. Ltd., which is one of the top communication companies and has developed billing systems for telecommunications operators in many countries, spends much expenditure on software testing. Running and analyzing 50 million tests will cost as long as 50 days. Thus, test reduction techniques are urgently needed to reduce the cost of testing in Huawei.

A number of different test reduction approaches have been proposed to help reduce the cost of testing. Three major test reduction approaches for regression software are test suite minimization, test case selection and test case prioritization [1]. Test suite minimization seeks to identify and eliminate redundant tests. Test case selection selects a subset of the test suite used to test the modified parts of the software. Test case prioritization tries to identify an ordering of tests, which is aimed at satisfying some criteria such as early fault detection.

In order to study these approaches, we made an investigation in Huawei for more than half a year. From the investigation, we found that for an industrial system with million lines of codes, it costed an average of 50 days to test it, including about 2 days of running all tests costs and more than 48 days of analyzing running results costs. The analyzing work includes matching each test with system rules and configurations, trying to modify some configurations and rerunning tests, which is done manually.

To reduce testing cost in Huawei, it is obvious that the amount of analyzed tests should be reduced. However, existing test reduction approaches can not deal with this problem. There are two reasons. One is that test suite minimization [2] needs the access to requirements which are collected from the running environment directly in Huawei. The relations between tests and requirements are not known in advance. It takes a lot of time to match tests with the requirements. For example, A tester in Huawei described the matching work to us as follows. "Even for a group of 3 people with 5 years of working experiences, it takes more than 3 weeks to complete the work. There are too many rules that you can only cover the main parts sometimes." The other is that, there are tens of millions of tests for system being tested in Huawei, so existing test reduction approaches need much cost to reduce tests, which can not reduce testing cost effectively.

The testing team in Huawei was in great demand to find an advanced technique to reduce cost while covering more potential faults. Another limitation on the selection of technique was that codes cannot be instrumented in the consideration of privacy and safety. Most of existing test selection techniques which need control flow or data flow collected by instrumenting codes [2] cannot be used here. One direction to solve this problem is to use a more advanced input based test generation technique to generate less number of tests while finding the same number of or even more faults. Chen *et al.* proposed Fixed-Size Candidate Set ART technique ($FSCS-ART$) to improve the failure detection effectiveness of a classic and popular testing technique RT (Random Testing) [3].

But $FSCS-ART$ has been criticized for its expensive cost. Each time a test is selected, distances between candidate tests and all the selected tests have to be calculated. Therefore the time complexity of $FSCS-ART$ is $O(n^2)$. Most of

Z. Zhang and J. Qian are with Collage of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, 210016 China. Corresponding e-mail: (zyzhang10@nuaa.edu.cn).

Y. Wang is with Software Institute, Nanjing University, Nanjing, Jiangsu, 210093 China.

Z. Wang is with School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing, Jiangsu, 210016 China. Corresponding e-mail: (wangziyuan@njupt.edu.cn).

ART testing techniques can be only applied to numeric inputs. However, an industrial program often has non-numeric inputs. The explanation of non-numeric inputs along with an example is presented in section 2.1. To solve these two problems, Chen *et al.* proposed Linear-Order Algorithm for Adaptive Random Testing [4] (abbreviated as $LART$ here). Although $LART$ can reduce the time complexity of $FSCS-ART$ from $O(n^2)$ to $O(n)$, it still has some limitations when applied to testing the billing system in Huawei. The biggest limitation is that all the categories are considered when distances for test inputs are calculated (we describe the notion "category" in Section 2). But in fact only a few categories are likely related to the cause of a failure, which we call failure related categories, while the other categories have little effects on the execution results. Suppose for a system, we have a test case T_1 with five categories(A,B,C,D,E), and category B and C are failure related categories for a system fail. If we changed the value of A,D or E and get a new test case T_2, there is a high probability that T_1 and T_2 have same execution results, so A,D and E are not failure related categories in this example.

Under above circumstance, we propose a new test reduction approach called Category Selection Based ART ($CSBART$). In our approach, we use $CategoryScore$ functions to identify failure related categories. According to selected categories, $CSBART$ selects tests and uses them to guide $LART$. Moreover, we perform our approach on a system of Huawei. Our experimental results show that it can reduce much time cost and improve the failure detection effectiveness of $LART$.

In our experiment we also compared $CSBART$ with *n-per-cluster* sampling test selection (abbreviated as NCS here) which was a fundamental and popular cluster filtering technique [5]. In order to cluster a huge volume of tests we use a very popular big data clustering method which is K-Means based on MapReduce [6].

To sum, we make the following contributions in this paper:

- We propose to utilize Adaptive Random Testing (ART) strategy to reduce tests in industrial real-life systems. The new proposed Category Selection based Adaptive Random Testing ($CSBART$) technique could be utilized for large-scale industrial programs with tens of millions of tests.
- We apply two category selection techniques, including Mutual Information based Variable-Ranking (MI) and Input-Profile based Variable-Ranking (IP), to help $CSBART$ reduce cost for industrial programs with millions lines of codes. Application results showed that compared with IP, MI could help $CSBART$ to detect faults and touch failure categories more rapidly but spend a bit more time.
- We apply $CSBART$ on two large industry billing systems. To the best of our knowledge, there have not been records that ART was used for large-scale industrial systems. Application results showed that compared with existing $LART$ and NCS, $CSBART$ could reduce much time cost and detect fault more rapidly.

Our paper is structured as follows. In the Section 2, we introduce some background techniques, as well as the testing process of the target system in Huawei. Section 3 discusses a motivation example of our approach. Section 4 proposes our new $CSBART$ method in detail. Section 4 reports our experimental studies, including experiment implementation, evaluation, experimental results and analysis, and industry review. Section 5 summarizes the lessons learned from the case study in Huawei. Section 6 discusses the threats to validity of our study. Section 7 presents some related work. Section 8 summarizes this paper.

II. Background

A. Industrial Settings

The system under test in Huawei is called Call Detail Records billing system (CDR billing system), which is used to replace the old billing system in Huawei. If a legacy system needs to be substituted for a new system, the new system must be fully tested. The information collected for testing includes documents, configurations and user information. One of the most important information is configurations. Configuration information includes necessary parameters used to calculate the charges for the CDRs. A small mistake in this step may lead to many failed tests. Faults occurring in the process of configuration settings is very difficult to be found.

TABLE I: Format of two tests

Time	*Area code*	...	*Number*
20150712	089	...	15850770934
20150712	041	...	15850770992

Table I shows an incomplete example of two tests, i.e. CDR. The new system, i.e. the CDR billing system developed by Huawei calculates billing result for each test. One test is a vector of string containing some information about one time of communication. In this paper one dimension of a vector is called a category and each element is called a choice [7]. Most categories in a vector are non-numerical, such as "Area code" used for identifying a specific area and "Number" used for recording the calling number. In this paper, "non-numerical inputs" refers to categorical or discrete data. "Area code" had 125 choices and "Number" had 112 choices making nearly 125*112 situations need to be considered when analyzing these two categories. The problem became even more complex when up to 200 categories with each category having 10 choices need to be taken into consideration. It takes more than 3 weeks to match tests with detailed requirements during the procedure of running and analyzing tests. Non-numeric categories are categories that exclude choices sets represented by numeric contents. The operations which are permitted only on numeric choices will generally be meaningless on non-numeric choices. Although the formats of "Area code" and "Number" are digits, they are non-numerical. Mathematical operators operated on them are meaningless.

Figure 1 shows the detailed testing procedure in Huawei. A legacy system had been working for years. Its running results were considered as the expected outputs. First, CDRs generated in one month were collected and charged using the legacy system. Then these CDRs were converted manually

不懂

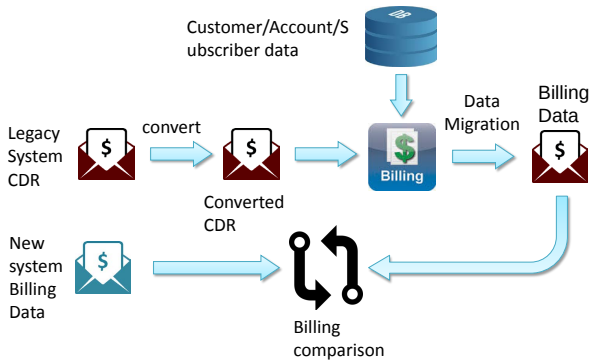

Fig. 1: Original testing process

from the legacy system's formats to the new system's formats by experienced programmers. Then, these experienced programmers randomly selected some CDRs and checked whether the conversion is right. CDRs in new formats were used for testing the new system. The outputs of the legacy system and the new system were collected and compared one by one. If two outputs for the same initial test were different, this test executed in the new system was failed.

B. Adaptive Random Testing

Random testing, which is an input-domain software testing technique, picks up test cases randomly one by one from input domain [8]. To speed up fault detection of random testing, Chen *et al.* firstly proposed Adaptive Random Testing (ART) technique [9], which is an enhanced version of random testing. The main idea of ART is that test cases should be evenly spread across the input domain by controlling distance among selected test cases. During software testing, if previous test cases have not revealed any failure, new test cases should be selected from input domain randomly as far as possible away from those existing test cases. In the case where failure regions in input domain is compact, which means that failure-causing inputs cluster into regions, ART could reveal failure more randily than random testing. There are many different implemented versions of ART. The best known one is Fixed Size Candidate Set Adaptive Random Testing ($FSCS$-ART) algorithm [10]. The detailed process of $FSCS$-ART algorithm includes: 1) Generate first test cases randomly to form a test set E; 2) Generate k candidate test cases c_1, c_2, ..., c_k, and calculate $dist(c_i, E)$ that is distance from the i-th candidate test case to all existing test cases (i=1, 2, ..., k); 3) Choose the candidate test case with the greatest distance to existing test cases.

Here we adopt the concept of categories and choices which was originally proposed as part of the category-partition technique [7], and suppose there are total g categories named A_1, A_2, ..., A_g. In original version $FSCS$-ART algorithm, the distance from candidate test case c to the set of all existing test cases E should be calculated by Equation 1 if n test cases have been selected into E and executed.

$$dist1(c, E) = \sum_{j=1}^{n}(\sum_{i=1}^{g} D(i, j)) \quad (1)$$

Where the $D(i, j) \in \{0, 1\}$ indicates whether the choices of the i-th category in candidate test case c and the j-th executed test case e_j in E are the same. It could be detailed defined as Equation 2:

$$D(i, j) = \begin{cases} 0 \ if \ r_i^c = r_i^{e_j} \\ 1 \ if \ r_i^c \neq r_i^{e_j} \end{cases} \quad (2)$$

Where $r_i^{e_j}$ indicates the choice of the i-th category in e_j, and r_i^c indicates the choice of the i-th category in c. It is obvious that, the number of comparison operations when calculating $dist(c, E)$ is $n \times g$. Such a cost may become huge with the growth of n.

In order to reduce such a comparison cost in distance calculating, Chen *et al.* proposed Linear-Order Algorithm for Adaptive Random Testing ($LART$) technique for discrete non-numeric input domain [4]. $LART$ generates and maintains a vector S=< $s_1^0, s_1^1, ..., s_1^{h_1}; s_2^0, s_2^1, ..., s_2^{h_2}; s_g^0, s_g^0, ..., s_g^{h_g}$ >. Here g is the number of categories, which means each test has g categories $A_1, A_2, A_3, ..., A_m, ..., A_g$. And h_m is the number of choices of m-th category ($1 \leq m \leq g$), which means that the m-th category has totally h_m choices. The term s_m^t in the vector S is the number of previously selected test cases where the choice of the m-th category is the t-th choice ($1 \leq t \leq h_m$). And the term s_m^0 in the vector S is the number of the previously selected test cases where the choice of the m-th category is not determined. For example, $s_4^3 = 5$ means there are totally 5 previously selected test cases where the choice of the 4-th category is the 3-rd choice. In $LART$, the distance between c and E is calculated by the Equation 3:

$$dist2(c, E) = \sum_{i=1}^{g}(n - s_i^{r_i^c}) \quad (3)$$

It is obvious from Equation 3 that, the number of calculation operations is g. After selecting a new test case into E, the counts will be updated in S vector. According to Equation 3, the cost of maintaining the vector S is g. Equation 1 and Equation 3 could output the same results, but Equation 3 could reduce calculating costs compared with Equation 1.

Besides the test case generation, the idea of ART could be utilized for test case prioritization problem. In this paper, in order to speed up failure detection and test case analysis, we attempt to adopt the idea of ART on test case selection problem in an industrial case.

III. Motivation Example

In ART algorithm, the cost of calculating distances among test cases may be huge. Some techniques, such as $LART$ and etc, have been proposed to reduce the cost of distance calculation. However, in some cases where there are too many input categories (e.g. hundreds or more), the $LART$ algorithm may still not be applied since the cost of calculation rises with the number of categories linearly. A possible option is

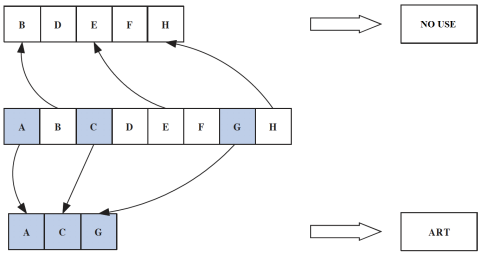

Fig. 2: ART by partial categories

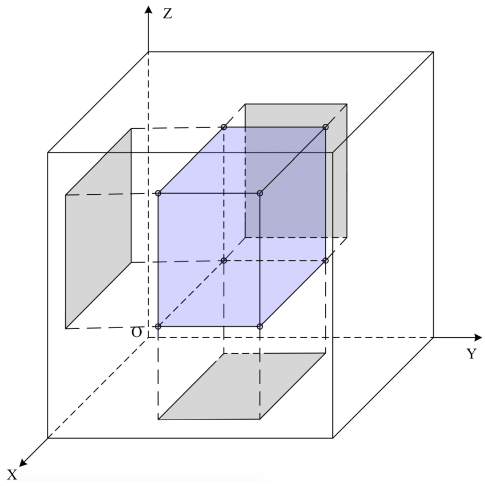

Fig. 3: Projection Strategy for Category Selection

to adopt category selection on traditional $LART$ technique. As the illustration of Figure 2, for a given system with many categories, partial important categories could be selected from all categories according to some criteria. The $LART$ algorithm will be applied on these selected important categories, while other categories will not be taken into consideration.

Which categories should be selected into consideration? In order to achieve the goal of ART that speed up fault detection, the failure-related categories should be selected here. Take Figure 3 for example, for a system with total 3 categories (X, Y, Z), the input domain may be displayed as a cube in the Figure. There is a tridimensional failure region, which means that all test cases in such a failure region must trigger the failure of the system, in the input domain. On each plane including XOY, XOZ, and YOZ, there is a projection of tridimensional failure region to form an area. And on each coordinate axis, there is also a projection of tridimensional failure region to form a segment. The larger the length of the projection segment on a coordinate axis, the easier to find the failure region on corresponding coordinate axis. While the smaller the length of the projection segment on a coordinate axis, the harder to find the failure region on corresponding coordinate axis. We should pay more attention on those coordinate axis with smaller length projection segment, since the values of these corresponding categories will determine whether the failure could be triggered. These "key" categories could be named as failure-related categories in this paper.

Such approach, we name it as Category Selection Based Adaptive Random Testing ($CSBART$), may help us to reduce the cost of the testing of CDR billing system. Figure 4 illustrates an example to show how category selection works in ART. Each table in Figure 4 represents a test. In the tables, the left column indicates all the categories, and the right column indicates the corresponding choices. In this example, B and C are failure related categories. After a failed test T_1 is selected, in order to find more faults, T_3 should be selected because both choices of B and C (b_2 and c_2) are different from these of T_1 (b_1 and c_1),respectively. However, if using $LART$, it considers all categories (from A to E) for calculating distances when select next test case. So the distance between T_1 and T_4 are 3 because there are three different choices of categories (A, D and E), while the distance between T_1 and T_3 are 2. According the results, T_4 will be selected since the distance of T_4 is longer than of T_3. But in fact, T_3 should be a better choice. So non-failure related categories may interrupt us from selecting fault revealing tests and should be removed during test selection in Huawei.

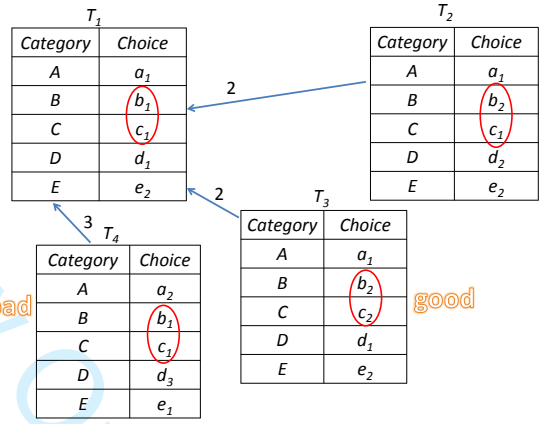

Fig. 4: An example to show why category selection is necessary

IV. Our Approach $CSBART$

The framework of our proposed approach can be found in Figure 5. We randomly select a test and add it to selected test set. For each test in selected test set, we execute it and identify whether it is pass or fail. Next, two category selection techniques are used to identify failure related categories, which guide to select specific number of tests. When selecting tests according to the idea of $LART$, the distance between a test c and existing tests E could be defined as a new form since category selection technique is applied.

Moreover, we describe the detailed procedures of our approach in Algorithm 1. $dist3$ is defined as following:

$$dist3(c, E) = \sum_{i \in SCS} (n - s_i^{r_i^c}) \quad (4)$$

The input of Algorithm 1 is all of original tests, and the output is a selected tests set. There are also some abbreviations

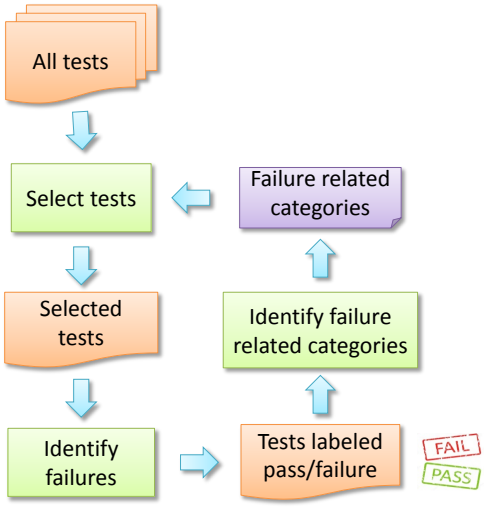

Fig. 5: Our approach

in our algorithm. S represents a vector used in $LART$, which we have introduced in detail in Section II.B. CSA is the abbreviation of category score array, which contains scores assigned to each category. SCS means the size of candidate set, which has been used in $LART$. k is an integer defined by testers.

Step 1 *Select intial tests*: At the initial step, we should select a test set containing $TN1$ tests. The value of $TN1$ could be set by industrial testers according to test requirements. To do this, one test is randomly selected (line 4-6) and to be added to selected test set E (line 13). Then S structure described in section 2.2 is updated (line 14). Next k candidate tests are selected (line 7) and the distances between E and each candidate test are calculated (line 8-10). The test e_n with the largest distance with E is selected and to be added to E (line 11-13). Then S is updated (line 14). This procedure will be iterated until $TN1$ tests have been selected.

Step 2 *Identify failures*: Tests are executed on old and new systems respectively, to identify passed and failed tests (line 16). Since old system is a stable version and has been used for long time, we believe that the execution result on it is right. For a test, if the execution result on new system is the same as on old system, this test is called a passed test. Otherwise, if execution results on old and new systems are different, whether the result on new system is interrupted, blocked or other abnormal execution result, this test is considered as a failed test. Moreover, difference abnormal execution results do not affect our approach, because we only use pass or fail execution results to select failure related categorise in our approach.

Step 3 *Identify failure related categories and select tests*: We call the $CategoryScore$ function to score each category and fill CSA (line 18-20). For each category one test e_n has a $<$ *choice, result* $>$ pair denoted by $CPFP$ indicating a pair of the choice of e_n and

Algorithm 1 $CSBART$ procedure

Input: Original Tests;
Output: Selected Tests;
Initialize S by setting each elements to 0
CSA = [] // CSA contains scores assigned to each category
Set $n = 0$ and $E = \{\}$
$SCS = \{1, 2, ...g\}$ (g is the number of categories)
Define an integer $k > 0$ as the number of candidates to be selected
1: Randomly select a test e_0 and add it into E
2: **while** specific number of tests (TN_1) have not selected **do**
3: Increment n by 1
4: **if** n = 1 **then**
5: Randomly select a test e_n
6: **else**
7: Randomly select k candidates $c_1, c_2, ..., c_k$
8: **for** each $c_{u(1<=u<=k)}$ **do**
9: Calculate $dist2(c_u, E)$ // using all categories
10: **end for**
11: Select c_o and set test $e_n = c_o$, $\forall c_u$ in $c_1, c_2, ..., c_k$, $dist2(c_o, E) >= dist2(c_u, E)$
12: **end if**
13: Add e_n into E
14: Update S by incrementing each s_i^j if jth choice of category A_i is contained in e_n
15: **end while**
16: Run all tests in E and get the pass or failure label for each test
17: **while** Termination condition not satisfied **do**
18: **for** each category A_i **do**
19: $CPFPS_i$ = Get $CPFPS$ for A_i in E
20: $CSA[i]$ = $CategoryScore(CPFPS_i)$
21: **end for**
22: Rank CSA according to scores
23: SCS = Get top M ranked categories according to CSA
24: **while** specific number of tests (TN_2) not selected **do**
25: Increment n by 1
26: Randomly select k candidates $c_1, c_2, ..., c_k$
27: **for** each $c_{u(1<=u<=k)}$ **do**
28: Calculate $dist3(c_u, E)$ // using top M ranked categories in SCS
29: **end for**
30: Select c_o and set test $e_n = c_o$, $\forall c_u$ in $c_1, c_2, ..., c_k$, $dist3(c_o, E) >= dist3(c_u, E)$
31: Add e_n into E
32: Update S by incrementing each s_i^j if jth choice of category A_i is contained in e_n
33: **end while**
34: Run last TN_2 tests in E and get the pass or failure label for each test
35: **end while**

the running result of e_n (pass/failure). A $CPFP$ set is denoted by $CPFPS$. For one category A_i, we collect a $CPFPS_i$ in E (line 19). $CPFPS_i$ will work as the input of a $CategoryScore$ function (line 20). We use two kinds of $CategoryScore$ functions. One approach is Input-Profile proposed by Yan et al. [26] and the other approach is Mutual Information based Variable-Ranking (MI) [9]. Then CSA is ranked to get top M categories to construct the new SCS (line 22-23). Nextaccording to top M categories, we use $LART$ to select $TN2$ tests(line 24-33). The value of $TN2$ also could be set by industrial testers according to test requirements. After that, $TN2$ selected tests are executed to identify whether they are pass or fail(line 34). This process continues until termination condition are satisfied. Termination condition are set according to system testing requirements, such as code coverage, special number of tests and so on.

In data-driven systems, tests do not merely define the state of some objects, but actually define the control flow of the system [11], so $CSBART$ can be used for these systems to reduce the cost of testing data driven systems with large amount of tests. Tests presented as unstructured texts may need to be transformed to the tests with format of categories and choices in our approach. In this paper, we selected one system in Huawei as a representative of large scale data driven systems, and did our experiments on this system. Specially, since $CSBART$ use relationships between (categories, choices) and system functions to select tests, it can be used for other systems if this system under test could provide input domain information and execution information that used to finding these relationships.

A. Category Selection Technique: Input-Profile

Algorithm 2 shows the detailed procedure of Input-Profile category selection technique. The input $CPFPS_i$ is obtained from Algorithm 1. In Algorithm 2, firstly, two arrays $FT = \{ft_1, ft_2, ..., ft_{h_i}\}$ and $PT = \{pt_1, pt_2, ..., pt_{h_i}\}$ are initialized with each element set to 0. h_i is the number of choices for category A_i. IP traverses each $CPFP$ in $CPFPS_i$ of category A_i and updates FT and PT (line 1-7). Then choices occurring more or equal times in failures than in passes are decided to be failure related choices. The $ScoreFunction$ of IP identifies failure related choices and sets scores of the category according to the ratio of failure related choices (line 8-9). Lastly, all categories are ranked according to the scores, and then testers select certain number of categories with largest values.

The following example shows how IP selects categories. In Table II there are four categories A, B, C and D. The left column shows the categories, and the right column shows the corresponding choices. Choices in each test are shown in Table III(a) in which each column represents a test. The first line indicates whether the test is failed or passed. F indicates a failed test. P indicates a passed test. Table III(b) shows the frequency of choices of category B occurring in failed tests and passed tests. The first column shows the choices. The second column shows the number of failed tests that

Algorithm 2 $CategoryScore - IP$ procedure

Input: $CPFPS_i$;
Output: Score of the category A_i;
Initialize PT and FT
1: **for** each $CPFP_j$ in $CPFPS_i$ **do**
2: **if** $result = Pass$ **then**
3: Update PT by incrementing each pt_k if kth choice of category A_i is contained in $CPFP_j$
4: **else**
5: Update FT by incrementing each ft_k if kth choice of category A_i is contained in $CPFP_j$
6: **end if**
7: Identify failure related choice according to PT and FT
8: Set Score = ratio of failure related choices
9: **end for**

cover the specific choices. For example, there are six tests that the value of category B is set to b_1, five of them are pass and one of them is fail. Since the number of passed tests are larger than of failed tests, according to the definition of failure related choices, b_1 is not failure related. b_2, b_3 and b_4 in Table III(b) are failure related. IP takes the ratio of failure related choices as the score of one category. Since category B has three failure-related choices, and one choice that is not failure related, so the score of category B is 3/(1+3) = 0.75. The scores of category A, C and D is 0.33, 0.5 and 0.33, respectively, so the categories are ranked as B, C, A and D. If testers want to select two categories, category B and C will be selected.

TABLE II: Categories and choices in inputs

Category	*Choices*
A	a_1, a_2, a_3
B	b_1, b_2, b_3, b_4
C	c_1, c_2
D	d_1, d_2, d_3, d_4, d_5

B. Category Selection Technique: Mutual Information Based Variable Ranking

$Variable\ Ranking$[12] makes use of a scoring function to assign one score to each category. A higher score is indicative of a more failure related category in this paper. A lot of criteria have been proposed to be the scoring functions. Shi et al. introduced information theory to evaluate the ability of tests to reveal faults[13] and gained good effects. In this paper we also chose information theory as the base of the scoring function.

$$MI(FT, PT) = \sum_{x_i} \sum_{y} \log \frac{P(X = x_i, Y = y)}{P(X = x_i)P(Y = y)} \quad (5)$$

Algorithm 3 shows the detailed procedure of Mutual-Information category selection technique. The input $CPFPS_i$ is obtained from Algorithm 1. Algorithm 3 gets FT and PT using the same way in Algorithmic 2. Then equation 5 works as the $ScoreFunction$ in Algorithm 3. In equation 5, X

TABLE III: An example to determine failure related categories

(a) Test inputs

	F	F	F	F	P	P	P	P	P
A	a_1	a_2	a_1	a_2	a_2	a_2	a_3	a_3	a_3
B	b_1	b_2	b_3	b_4	b_1	b_1	b_1	b_1	b_1
C	c_1	c_2	c_3	c_3	c_4	c_1	c_2	c_2	c_2
D	d_1	d_3	d_2	d_1	d_1	d_2	d_2	d_3	d_1

(b) Determine failure related category

B	F	P	*Failure related choice*
b_1	1	5	Not
b_2	1	0	Yes
b_3	1	0	Yes
b_4	1	0	Yes

Algorithm 3 $CategoryScore - MI$ procedure

Input: $CPFPS_i$;
Output: Score of the category A_i;
Initialize PT and FT
1: **for** each $CPFP_j$ in $CPFPS_i$ **do**
2: get FT and PT using the same way in Algorithmic 2
3: Identify failure related choice according to FT and PT
4: **end for**
5: Set Score = $ScoreFunction(FT, PT)$

indicates a category. x_i indicates a choice of category A_i. Y indicates the running results of the corresponding tests. y indicates whether the test is passed or failed. $P(X = x_i, Y = y)$ indicates the probability that the running result of the test is y while its category A_i's choice is x_i. $P(X = x_i)$ indicates the probability that the category A_i's choice is x_i. A larger score means that category X is more related to whether the test is passed or failed. We also take Table III(b) as an example to show how MI is calculated. There are totally 4 failures and 5 passes. 6 tests contains b_1. Among all the tests only one failure contains b_1. 5 passes contain b_1. In this case $P(X = b_1, Y = F) = 1/9$, $P(X = b_1, Y = P) = 5/9$, $P(X = b_1) = 6/9$, $P(Y = F) = 4/9$. $P(Y = P) = 5/9$. MI can be calculated using these intermediate results and all categories are ranked according to the scores, then certain number of categories with largest values will be selected.

V. Experimental Studies

A. Research questions

In this paper, we study the following four research questions. ARTsum

- RQ1. Can $CSBART$ outperform RT and $LART$ in terms of fault detection capability?
- RQ2. Can $CSBART$ outperform NCS in terms of fault detection capability?
- RQ3. Can $CSBART$ outperform RT and $LART$ in terms of time cost?
- RQ4. Can $CSBART$ outperform NCS in terms of time cost?

Our first research question is used to identify that when comparing with RT and $LART$, whether $CSBART$ could perform better in terms of fault detection capability. We select RT and $LART$ for two reasons. One is the object systems contain tens of million test cases, so the ideas of RT and $LART$ are suitable for selecting tests in Huawei. The other is that RT and $LART$ are two popular test case generation techniques. In our experiments, we use their ideas to select tests. Moreover, NCS is a popular test selection technique [5], so we compare the fault detection capability of $CSBART$ and NCS in the second research question. In order to verify the efficiency of $CSBART$, we also compared the time cost between $CSBART$ with RT, $LART$ and NCS in the third and fourth research questions, and this is what Huawei cares about in system testing.

B. Evaluation

In our experiments, the fault detection capability includes two aspects. One is the fault coverage and the other is the speed of fault detection. In the industry testing scenario where our experiments was conducted, a huge volume of failed tests was caused by the same faults. Running and analyzing a specific subset of tests was enough to reveal the most faults. Which failed test was caused by which fault should be known in advance, to study whether $CSBART$ can improve $LART$ on the ability of selecting tests that can reveal more faults. Note that this information was only used to evaluate $CSBART$ not to run $CSBART$. Running $CSBART$ did not need the information of detailed relations between tests and detected faults. Running $CSBART$ only needed the information of whether a test passes or fails.

During the evaluation process all the tests were analyzed to get corresponding billing rules. Each billing rule was analyzed to get which categories and choices by which the rule can be represented. Passed and failed tests with similar billing rules were compared in order to find the corresponding faults and generate the descriptions about the faults. A fault description consisted of 4 parts: 1. the billing rule the test was concerned; 2. the combination of choices related to this billing rule; 3. the configurations related to the billing rule; 4. how wrong configuration made the fault. We used these four parts to code and identify fault descriptions. After doing this each failed test was related to a certain ID of fault description. The note is that running $CSBART$ does not need all these analyzing work described above. Running $CSBART$ only needs to know whether a test passes or fails. All the analyzing work is only required to evaluate not to run $CSBART$.

A variable fault coverage rate was used to evaluate the performance of different approaches ($LART$, $CSBART$ and NCS). Fault coverage rate was calculated as equation 6. In this equation MF is the number of faults which selected tests can reveal. TF is the total number of faults. The higher

coverage rate indicates a better ability to find tests that can reveal faults.

According to faults descriptions we got some categories that were related to the faults. These categories are called failure related categories in this paper. We wanted to test the ability of $CSBART$ to find these failure related categories. Another variable failure categories coverage rate was used to evaluate the ability of $CSBART$ to identify failure related categories. Failure categories coverage rate was calculated as equation 7. In this equation SFC indicates the number of selected failure related categories. FC indicates the number of all failure related categories.

$$Fault\ coverage\ rate = \frac{MF}{TF} \tag{6}$$

$$Failure\ categories\ coverage\ rate = \frac{SFC}{FC} \tag{7}$$

To evaluate the speed of fault detection of different approaches, we calculate the average percent of faults detected ($APFD$) [14] of each approach using equation 8.

$$APFD = 1 - \frac{\sum_{i=1}^{m} TF_i}{nm} + \frac{1}{2n} \tag{8}$$

In equation 8, n is the number of selected tests, m is the number of faults, and TF_i is the the number of tests that have been executed, when the fault i is firstly discovered by selected tests. Obviously, the value range of $APFD$ should be (0, 1). The maximum value of $APFD$ is $1 - \frac{1}{2n}$, if all faults are detected by the first test case. The minimum value of $APFD$ is $\frac{1}{2n}$, if all faults are detected by the last test case. The larger the value of $APFD$ is, the faster the fault detection speed of the approach is.

Moreover, time cost is very important in industry, and the experts in Huawei hopes testing systems with less time cost, so we use time cost to evaluate each technique. The less time the technique uses, the better it is.

C. Experiment implementation

In our experiments, two Call-Detail-Records billing systems, which are from Vietnam and Paraguay, are used as experiment objects. Both projects of old versions have been applied in practice for some years, so there are sufficient tests to evaluate $CSBART$. Moreover, since these two systems are from two different countries, tests used in our experiments have high diversity, which could ensure the accuracy of experiment results. There are about fifty-million tests in our experiments.

We used three steps to select tests from original tests. In the first step, we select a test case randomly at first. Then we selected test one by one from original tests to perform category selection. The new selected test has largest distance with tests which have been selected, and the distance between are calculated using all categories in this step. According to the experiences of testers in Huawei, k in Algorithm 1 is set to be ten. If we need to select N_F tests finally from original tests, $TN1$ in Algorithm 1 is set to be $\frac{N_F}{10}$. For example, if one-million tests are selected finally from fifty-million tests, we will select one-hundred-thousand tests in the first step. In the second step, we run tests that were selected in the first steps in projects of new versions, and get execution results(pass of fail). In the last step, according to results, we ranked categories related to faults using IP and MI, respectively. A suitable technique needed to be chosen to handle a huge volume of tests in order to rank categories. We chose MapReduce, which is a parallel data processing framework, to implement MI and IP. Moreover, there were only few studies on how many categories should be selected when using *Variable Ranking* techniques. Most of the studies were based on experiences [15]. According to the experiences of testers in Huawei, we selected top 20 ranked categories. Lastly, we selected tests until termination condition was satisfied. We still selected new test which has largest distance with tests which have been selected, but the distance between tests are calculated only using 20 categories which we selected in the second step. In our experiments, the termination conditions are the number of tests are one-million, two-million, three-million, ... , ten-million, respectively.

For RT, we selected tests randomly from original tests, and the number of tests is the same as $CSBART$. For $LART$, we calculated the distance between tests using all categories, and selected new test which has largest distance with tests which have been selected until certain number of tests have been selected.

For NCS, we clustered original tests using K-Means algorithm. According to [16], the number of clusters is set to $\sqrt{\frac{N_t}{2}}$. N_t is the number of original test cases. In our experiments, the number of original test cases is fifty-million, so the number of cluster is five-thousand. Then we randomly selected the same number of tests from each cluster, and the whole number of tests are the same as tests selected by $CSBART$. For example, if we selected one-million tests by $CSBART$, when using NCS, two-hundred tests should be randomly selected from each cluster. For $NCS{-}IP$ and $NCS{-}MI$, according to [17], we selected two-thirds of original tests (about thirty-three million tests) to select categories related to faults. Based on the 20 selected categories, remaining one-third tests (about seventeen-million tests) were clustered using K-means algorithm, and the number of clusters is set to 4132. Lastly, we selected tests with the same size as tests selected by $CSBART$, and for each cluster, the number of tests are same. Moreover, in order to get a more precise coverage rate, we repeated all experiments 30 times and calculated the average value.

D. Experimental Results and Analysis

The experiments were conducted on the tests collected from two countries which had different failure ratios 11% and 1%. In order to answer *RQ1*, we compared $CSBART{-}IP$, $CSBART{-}MI$, $LART$ and RT. The results from tests with failure ratio reaching 11% is shown in Figure 6. The X axis indicates the number of selected tests. The Y axis indicates the fault coverage rate. From Figure 6, we find that when selecting same number of tests, $CSBART{-}MI/IP$ could achieve higher than RT and $LART$. For example, $CSBART{-}IP$ can get 10% fault coverage rate higher than $LART$ at best

while $CSBART-MI$ can get about 20% higher in average. Moreover, we also find that $CSBART-MI$ is better than $CSBART-IP$, which means $CSBART-MI$ could achieve higher than $CSBART-IP$ when their selected test rates are the same.

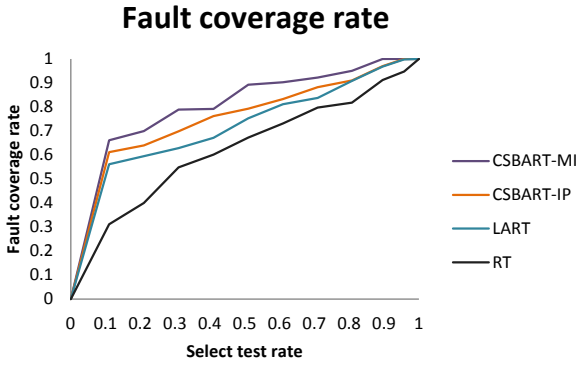

Fig. 6: Fault coverage rate from tests with failure ratio reaching 11%

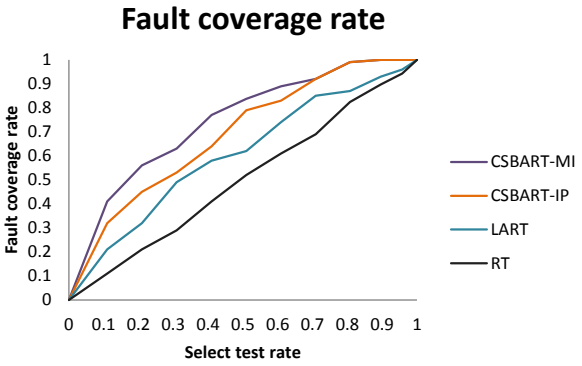

Fig. 7: Fault coverage rate from tests with failure ratio reaching 1%

Figure 7 shows the results from tests with failure ratio reaching 1%. Both $CSBART-IP$ and $CSBART-MI$ can find more faults than RT and $LART$ when their selected test rates are the same. For example, if 10% of tests were selected, the tests selected by $CSBART-IP$ and $CSBART-MI$ can reveal more than 30% and 40% of faults, respectively. When using RT and $LART$, the selected tests can only find about 5% and 16% of faults. Moreover, from figure 7, we find that if selected test rates are the same, $CSBART-MI$ could also achieve higher than $CSBART-IP$.

All the following results were collected from experiments on the tests with 1% failure ratio. In the experiments, we called certain combination of choices that can reveal a failure pattern. Many failure patterns may be caused by one fault. In the beginning, tests that reveal faults generating relative large number of failure patterns were selected to cause a faster increasing of fault coverage rate. But after tests that reveal these types of faults selected, the remaining tests that reveal faults generating less failure patterns were very hard to be selected. We analyzed the distributions of failure patterns to see why $CSBART$ could achieve higher than $LART$ when selected test rates are the same. In Figure 8, the X axis indicates the IDs of failure patterns, and the Y axis indicates the percent of failed tests relating to the corresponding failure patterns. As there were too many failure patterns, we just list top 11 of them. The number of failures relating to other failure patterns was too small. From Figure 8, we can see that there were some patterns relating to much more tests than others. These patterns were more likely to be selected while other failure patterns were less likely to be selected using $LART$. By introducing category selection, $CSBART$ had a higher ability to find failures relating to failure patterns occupying rather small percentage of failures.

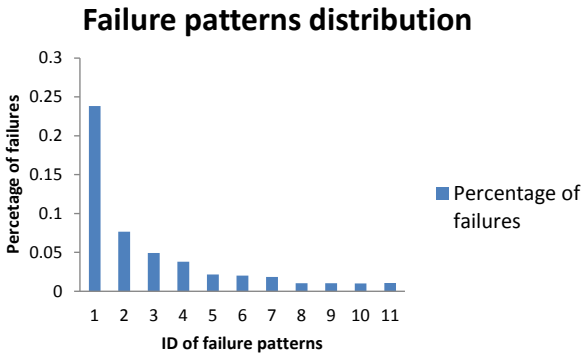

Fig. 8: Distribution of failure patterns

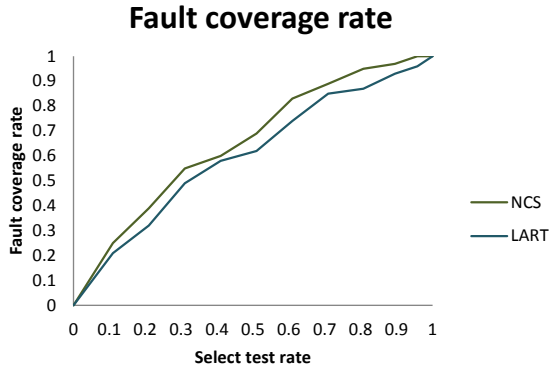

Fig. 9: Comparing $LART$ with NCS

In order to answer *RQ2*, $CSBART-MI/IP$ was compared with NCS. Figure 10 shows that under the guidance of category selection, the tests selected by $CSBART-MI$ could find more faults than by NCS, while the fault coverage of tests selected by $CSBART-IP$ and NCS are almost the same. When using category selection during NCS, we found that the effect of $NCS-IP/MI$ is as similar as $CSBART-IP/MI$, which is shown in Figure 11 and Figure 12. Moreover, we compared NCS with $LART$. Figure 9 shows that, without guidance of category selection, the tests selected by NCS

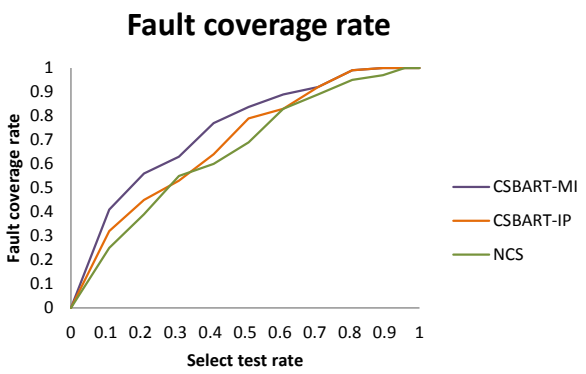

Fig. 10: Comparing $CSBART-MI$, $CSBART-IP$ with NCS

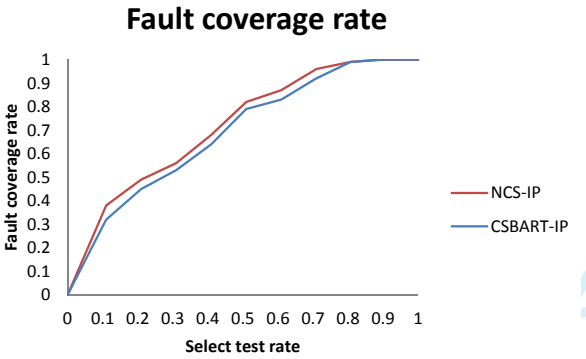

Fig. 11: Comparing $CSBART-IP$ with $NCS-IP$

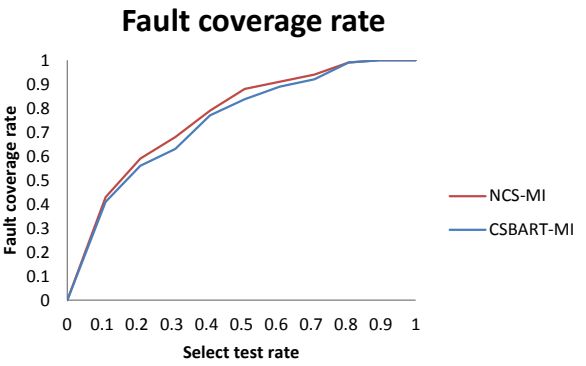

Fig. 12: Comparing $CSBART-MI$ with $NCS-MI$

still found more faults than by $LART$, but the difference is not obvious.

In summary, under the guidance of category selection, $CSBART$ could select tests with higher fault coverage than NCS, especially when MI was used. But the performance of $CSBART-IP/MI$ was as similar as $NCS-IP/MI$. However, consider the time cost (Table VI), NCS took more than 58361 seconds to cluster one million tests, which was nearly thirty times longer than $CSBART-IP/MI$ spent. For $NCS-IP/MI$, they needs to executed two-thirds of original

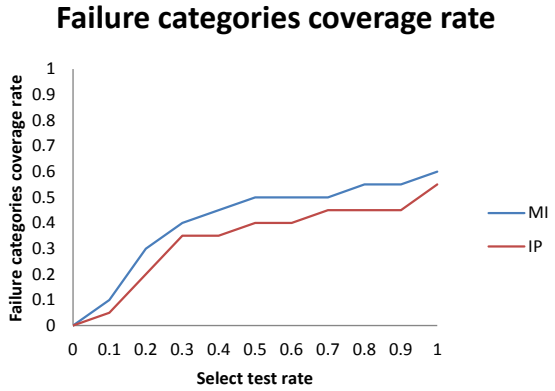

Fig. 13: Failure categories coverage rate for MI and IP

tests (about thirty-three million tests) to select categories related to faults, and the execution time cost are about thirty hours, so the time cost is too large. Based on the above analysis, we judge that $CSBART-IP/MI$ is better than $NCS-IP/MI$.

To evaluate the speed of fault detection of each approach, we calculated the $APFD$ values of them. Table IV shows the results of $APFD$ values. From Table IV, we find that $CSBART$ has larger $APFD$ value than $LART$ and RT, which means it could detect faults faster than $LART$ and RT. The $APFD$ value of $CSBART-MI$ is larger than NCS, but slightly less than $NCS-MI$. The $APFD$ value of $CSBART-IP$ is slightly less than NCS and $NCS-MI$. So the fault detection speed of $CSBART-MI/IP$ are similar as of $NCS-MI/IP$.

TABLE IV: $APFD$ value of each approach

	$CSBART-MI$	$CSBART-IP$	RT	$LART$
$APFD$	0.739	0.687	0.498	0.584
	NCS	$NCS-MI$	$NCS-IP$	
$APFD$	0.690	0.745	0.696	

In order to see whether $CSBART$ can achieve better performance, we perform $Mann-Whitney\ U\ Test$ with respect to $CSBART$. Table V shows the results of p values. The results show that $CSBART$ can get significantly better performance than $LART$ when a better category selection (MI) method is used. Although NCS took much more time than $CSBART$, it was not significantly better than $CSBART$.

TABLE V: $Mann-Whitney\ U\ Test$ results

	$LART$	NCS	$NCS-IP$	$NCS-MI$
$CSBART-MI$	**0.048**	0.152	-	0.350
$CSBART-IP$	0.204	0.423	0.346	-

Moreover, we want to see whether MI and IP can truly find the real failure related categories. Each time a number of tests were selected, we calculated failure categories coverage

rate. The results are shown in Figure 13. Before 30% of tests being selected the increasing speed of coverage is relatively high. After 30% of tests being selected the speed decreases. After 50% of tests being selected, failure categories coverage rate can only reach up to 51%. MI is more capable in finding failure related categories than IP according to the results.

TABLE VI: Time cost(s)

$\#Tests$	RT	$CSBART-MI$	$CSBART-IP$	$LART$	NCS
1000000	3	1832	1759	2240	58361
2000000	7	3520	3391	4550	117643
3000000	10	5402	5232	6724	169506
4000000	13	7932	6784	8948	233988
5000000	16	9980	8832	11210	288805
6000000	19	10029	10986	12347	357671
7000000	23	13309	11389	15607	410013
8000000	27	16100	14933	19295	466956
9000000	30	18900	16033	21450	519026
10000000	33	21086	18493	23043	600413

In order to answer RQ3 and RQ4, we recorded the time that $LART$, $CSBART-MI$, $CSBART-IP$, RT and NCS spent on selecting tests from one million to ten million, respectively. The results are shown in Table VI. The "time cost" is the time of executing algorithms to select tests, expect for test case execution. We excluded test execution for two reasons. One is that all executed tests in algorithms($CSBART-MI$ and $CSBART-IP$)would be selected from the original tests, so the execution time cost of these tests is not an overhead of our algorithms, and we counted it in the execution time cost of all selected tests. The other is that the execution time cost is very small (2 days) compared with the cost of whole testing work.

However, we don't compare the "time cost" between $CSBART$ and $NCS-IP/MI$. This is because for $NCS-IP/MI$, we executed two-thirds of original tests (about thirty-three million tests) to select categories related to faults, and selected tests from the remaining one-third tests (about seventeen million tests), so all executed tests in algorithms were not selected as the final tests, which means the execution time cost of these tests is an overhead of $NCS-IP/MI$. In our experiments, the execution time cost in algorithms is very large(about thirty hours), which means the performances of $NCS-MI$ and $NCS-IP$ are poor, so it is meaningless to compare compare the "time cost" between $CSBART$ and $NCS-IP/MI$.

From the Table VI, we find that $CSBART$ used less time to select tests than $LART$ and NCS. NCS took more nearly thirty times longer than $CSBART-IP/MI$. However, it spends much more time than RT. This is because when using random selection, we just need to generate certain number of random numbers, so the time cost is very small. But from experiment results, the fault detection capability of tests selected by RT is very weak.

In summary, from our experiments results, for RQ1, we consider that $CSBART$ outperforms RT and $LART$ in terms of fault detection capability. For RQ2, under the guidance of category selection, $CSBART$ outperforms NCS in terms of fault detection capability, especially when MI was used. But the performance of $CSBART-IP/MI$ was as similar as $NCS-IP/MI$. For RQ3, $CSBART$ outperforms $LART$ in terms of time cost, but spent much more time than RT. For RQ4, $CSBART$ outperforms NCS in terms of time cost.

E. Industry Review

In order to review $CSBART$, a review meeting which involved 6 testers with more than 5 years of working experiences was held to collect their opinions. 10% of tests were selected by $CSBART$. The data, evaluated in the meeting, included: (1) 51 faults revealed by the selected tests; (2) 20 categories that selected; (3) percentage of covered requirements; (4) the time cost of analyzing original tests and selected tests.

According to the collected data some questions were discussed in the meeting:

(1) Whether critical faults can be revealed when relatively small number of tests (10%) were selected? To answer this question we need to first determine whether a fault is critical or not. The vote was conducted by 6 testers. The fault receiving more than 4 votes was considered as a critical fault. Among 51 revealed faults, 2 faults were determined to be critical. The testers agreed that $CSBART$ can find tests that reveal critical faults when relatively small number of tests were selected.

(2) Whether $CSBART$ can find important categories? Experienced testers defined important categories as the categories related to the implementation of the business requirements. Important categories were determined in the same way as that was used to determine critical faults. Among 20 selected categories 16 categories were considered as important. The selected important categories included user type, special billing service indicator, duration of communication, service priority and etc. 2 categories, special billing service indicator and service priority were ignored by experts. Such overlook resulted in longer time to be spent to find 2 critical faults. The testers agreed that $CSBART$ can find important categories and categories experts ignored.

(3) Whether selected tests can cover the main requirements? The requirements coverage had been calculated before the meeting was held. Tests were analyzed by testers to see which requirements can be matched with each test to find the faults. The results showed that selected tests can cover 91% of the business requirements. Business requirements describe customer focused billing business logic. These requirements do not include the performance or technique architectures. Testers also reviewed uncovered requirements and found that these requirements involved a very small number of users. Only about 0.032% users paid attention to the requirements. The testers agreed that tests selected by $CSBART$ can cover the main requirements.

(4) Whether analyzing selected tests can reduce the time cost of analyzing tests? To answer this question a lot of preparing work was performed before the meeting was held. Testers evaluated each 10% of tests selected by

$CSBART$ and calculated the time cost for analyzing selected tests to find specific faults. The time cost of finding each fault in the original testing process was also recorded for comparison. The summarized data is presented in Table VII.

TABLE VII: The time cost of analyzing tests that is reduced by $CSBART$

Fault coverage rate	Original cost (Day)	Percentage of tests selected by $CSBART$	cost of test selection (Hours)	Cost of analyzing selected tests (Day)	Reduced time (Day)
41%	11	10%	1.2	2	9
56%	18	20%	3	6	11.9
63%	26	30%	4.3	11	14.8
77%	29	40%	5.6	16	12.8
84%	31	50%	7	18	12.7
89%	34	60%	7.5	22	11.7
92%	35	70%	8	26	8.7
99%	47	80%	9.8	39	7.6
100%	55	90%	11	49	5.5

In Table VII, the first column shows the percentage of faults revealed, the second column shows the original time cost for finding the corresponding faults, while the third column shows the number of selected tests revealing corresponding faults. Time cost of test selection and analyzing selected tests are presented in fourth and fifth columns respectively. The last column shows the reduced time.

Table VII shows that $CSBART$ can reduce 10 days of analyzing tests in average. 14.8 days can be reduced at best when 63% of faults were found. Before 89% of faults found, $CSBART$ can reduce more time cost - analyzing tests reaching 11.7 days in average. After that the number of reduced days was decreased. $CSBART$ can reduce 7.3 days in average in order to find remaining 11% faults. The testers reviewed the remaining faults and found that these faults had little connection to the corresponding test inputs. Testers need to do more analysis by comparing passed tests with failed tests to find faults. Nevertheless in such case there were still 5.5 days reduced. The testers agreed that although most of the tests need to be executed, tests analyzing time can be reduced.

Two critical faults were found by the tests selected by $CSBART$. Since the Confidentiality Agreement with Huawei, we only described these two critical faults and related categories experts ignored as following:

(1) A special billing service was only needed in some cases. It should be activated when needed. One category special billing service indicator shows whether the service should be activated. In the project we studied the service was disabled incorrectly leading to large deviation of billing results. The testers in Huawei ignored the category and did not check whether the service was activated. This made them spend more than 1 day to find the fault.
(2) Different billing services used different billing rules to calculate prices for each test. Different billing services run in different priority which was recorded in one category called the service priority. In the project we studied one fault was that the priority of services was not configured correctly. The fault leaded to prices of large number of tests calculated using wrong services. The testers ignored the priority category and did not check the corresponding configuration. This made the testers spend almost 2 days to find the fault.

VI. Lessons learned

Before we cooperated with Huawei, we have learnt a model about how to bridge academia and industry and deal with practitioners [18], which helped us greatly for our cooperation. The first step of the model is how to identify potential improvement areas based on industry needs, so we were on site and understood the different roles and goals of industrial practitioners. Moreover, we helped industrial practitioners to achieve their goal friendly, which we considered as the starting point of identifying research problems. In our study, the time pressure and the limited budgets are two important factors that need to be considered in a higher priority in Huawei. How to help different roles to solve the problems brought by time pressure and limited budgets are also common industry problems deserved to be researched.

The second step is formulating a research agenda. We considered that when collaborating with industry partners, how to communicate with them. Since the background knowledge of the industry partners is different from of researchers, it is not easy for industrial partners and researchers to communicate smoothly. So we learned the industry background knowledge, company and domain-specific vocabulary, and get consistent understanding on some important concepts with industry partners. Using the vocabulary of industrial partners to let them understand the rationality and effectiveness of proposed approach can greatly enhance their confidence and participation. This also makes the communication two-way instead of one-way.

The next step is finding a candidate solution in cooperation with industry. When applying research results into industry cases, we need to solve the problem of unifying the knowledge between academics and industry. The process of unifying the knowledge can also be seen as the process of knowledge exchanging. The knowledge from academic often represents as mathematical models. Mathematical models cannot be directly transferred to an industrial achievement without an abstracting process in which variables are extracted from the industry problem. In this paper, every category of the test input can be abstracted as different variables. The successful or failed execution of a test case can be also seen as one variable. In this way the category selection problem is converted into the problem of measuring the correlation of two variables.

During above process, which can be seen as the process of substituting mathematical models with expert knowledge, it is worth noted that it is not necessary to fully simulate the details of expert knowledge. Professional testers in Huawei use their knowledge of business logic to select the important categories and analyze test inputs. Complex and trivial business logic makes it very hard to describe all the business logic with

one mathematical model directly. In this paper, the correlation between categories and pass or failure information of tests is chosen as a substitution of the business logic. Moreover, we presented our method to Huawei experts, and improved it based on the feedback and ideas. Before $CSBART$ be used in the billing system, we have tested it in a lab environment. Only if test results are satisfactory to us and Huawei experts, we can apply $CSBART$ in the billing system.

The last step is evaluation. In order to evaluate an approach in an industry environment, both objective and subjective approaches should be used. Subjective evaluation is a good complement to objective evaluation. In this paper objective approach was used to evaluate the fault coverage rate of $CSBART$. Subjective evaluation was performed by testers to see which faults were critical and which selected categories were important. The testers also evaluated selected tests by $CSBART$ to see whether analyzing selected tests can reduce the time cost.

Besides the above model we used during the cooperation with Huawei, we still have some insights on the domain part of the collaboration. The main insight is how practitioners and researchers both perceived the value of the approach from a practical perspective. At first, there were some mismatching views between practitioners and researchers regarding the goals of the research. The purpose of practitioners in Huawei is reducing the test cost of billing system, so they do not focus on whether the solution is new or old techniques, which is different from research purpose. To overcome this problem, we try to apply some new technologies to solve industrial problems. In this paper, we propose a new test selection method called $CSBART$ by improving $LART$ with category selection approaches to reduce time cost of billing system, which have been bothering Huawei for a long time. $LART$ could initially meet the needs of Huawei, but the cost is expensive. Category selection approaches could help reduce test cost when using $LART$. So $CSBART$ could satisfy both requirements of practitioners and researchers in our paper.

VII. Related work

A. Test optimization

In this section we give a brief introduction to the studies of test suite minimization, test case selection and test case prioritization. Horgan and London proposed ATAC [19], [20], a test suite minimization method. ATAC applied data-flow analysis and linear programming to solve the test case minimization problem. However it is too expensive to analyze the data flow of a billing system with more than one million lines of codes in Huawei. Offutt *et al.* treated the test suite minimization problem as a set cover problem and measured requirement coverage [21]. Tallam and Gupta developed a greedy approach based on the Formal Concept Analysis of the relations between tests and test requirements by introducing the delayed greedy approach [22]. In Huawei matching all the tests with the corresponding requirements took more than three weeks. Other minimization approaches primarily considered code-level structural coverage. Marre and Bertolino treated test suite minimization as a problem of finding a spanning set over a graph [23].

Existing test selection techniques mainly used control flow or data flow techniques [1]. Fischer *et al.* presented an approach using Integer Programming to solve the selection problem for testing FORTRAN programs [24], [25]. One weakness of this approach is that it has to deal with control-flow modifications. The test dependency matrix depends on the control-flow structure of SUT. If a program's control-flow structure changes, the tests dependency matrix has to be updated by executing all the tests. In their researches, several times of control flow modifications will lead to several times of executing all the tests. Several times of two days need to be spent if this approach is used in Huawei. Other research about metamorphic testing on test case selection is also too time consuming [26] as the expensive cost of matching requirements and tests in Huawei.

The aim of test case prioritization is to find an ideal execution ordering of tests, so that some criteria can be satisfied as early as possible. Test case prioritization was first proposed by Wong *et al.* [27]. However, it can only be applied to tests that are already selected by a test case selection technique. Zhenyu Chen *et al.*[28] used logic coverage to perform test case prioritization.

B. Adaptive Random Testing

The main idea ART is achieving an even spreading of test cases across the input domain. For numeric input domains, the distance metric is usually used to measure "far apart". The most popular and cited ART algorithm is the Fixed-Sized-Candidate-Set ART ($FSCS-ART$) [3]. In this method, the distance between each candidate test and tests that have been executed is calculated. The test with the largest distance is then selected as the next test case. Since $FSCS-ART$ is not straightforward for non-numeric input domains, Kuo *et al.* [29] and Merkel *et al.* [30] have proposed a generic distance metric based on the concept of categories and choices. They classify black-box input domain into several categories, and use partitioning to generate functional tests. Ciupa *et al.* [31] proposed a specific distance metric for object-oriented software called $ARTOO$, which extends the applicability of the basic ART algorithm to object-oriented software. $ARTOO$ use the technique of clustering to calculate the object distance, which is a measure of how different two objects are, including an elementary distance between the direct values of the objects, a distance between the types of the objects, and recursive distances between the fields of the objects, to reduce the distance computation overheads. However, the studies in [29] and [30] are not suitable in our experiment. Since there are many categories and choices in billing system, the test cost using the studies in [29] and [30] is very expensive, so we used failure related category to reduce cost. $ARTOO$ needs object distance which we can't get from billing system.

In the previous studies of ART, the most frequently used metric to compare the effectiveness of ART and RT is F-measure. To compared RT and ART, Ciupa *et al.* [31] used real-life faulty versions of object-oriented programs, Lin *et al.* [32] used six open source Java software artifacts with seeded faults and Zhou *et al.* [33] used four numerical programs

written in C from GNU Scientific Library with seeded faults. The experiment showed that the average F-measure of ART are smaller than of RT. Though Arcuri *et al.* [34] exposed the practical limitations of ART, T.Y. chen *et al.* [35] think ART will have a better F-measure performance than RT at later stages of software development through studies of [36] and [37].

Moreover, there exist general techniques applied to ART algorithms to reduce their cost of test case generation. mirror ART [38] divides the input domain into several partitions, one partition is referred to as the source partition and the other partitions are referred to as the mirror partitions. Test cases generated in the source partition are mapped into the mirror partitions, and new test cases are generated in the the mirror partitions. Another technique is called forgetting ART [39]. It uses only a portion or a constant number of already executed test cases to determine the next test case instead of using all already executed test cases. However, these cost reducing techniques focused on systems with numeric input domains. In our paper, we applied failure related category on $LART$ for systems non-numeric input domains.

VIII. Threats to Validity

Threats to internal validity are concerned with the uncontrolled factors that may be also responsible for the results. In this paper, the correction of conversion made from the old CDRs to the new ones may influence the results. To ensure the correction, conversions have been tested by experts in Huawei. And the new system would check whether the format of new CDRs are correct. If the format of a new CDRs is not correct, it cannot be executed by the new system. Moreover, there are many ART techniques and category selection approaches. The result may be affected by the selection of ART techniques and category selection approaches may influence the results. As we known, most of ART techniques are used for continuous input domain. However, the application scenario in our method is that the input domain of the system is non-numeric, so we propose $CSBART$ by improving $LART$. We used two popular category selection approaches IP and MI to select fault related categorise. These techniques help reduce the internal threats to our experiment results to certain extends.

Threats to external validity are concerned with whether the findings in our experiments are generalizable for other settings. The selection of experiment objects and data is a key point for our study. The purpose of our method is to solve the problem of testing billing system in Huawei, so we select this system as our experiment object in our method. The CDRs data we used are really industrial data, which means that our method can be applied in industry. Moreover, though we only apply our method in billing system in Huawei, it also could be used in other lager scale system as long as the input domain of the system is non-numeric.

Threats to construct validity are concerned with whether the measurement in our experiments reflect real-world situations. The evaluation metric is a major concern that may affect the experimental results. In this paper, we used real fault coverage and time cost as evaluation metrics, which are two of the most important things that industry concerns. Moreover, we use $Mann-WhitneyUTest$ [40], which is one the most widely used methods to test for two independent samples, to test whether $LART$ is better than NCS. All of these could help us reduce construct validity.

Threats to conclusion validity are concerned with that can lead researchers to reach an incorrect conclusion about a relationship in the observations. To reduce the threats to the conclusion of our experiment, we consider full time cost of our method including test selection time, analysis time and execution time. Compared time cost of original test process, our method could reduce time cost. Moreover, to further verify experiment conclusions, 6 test-experts in Huawei reviewed our application results, and all of them agreed with that our method could reduce time cost.

IX. Conclusion

In this paper, the $CSBART$ technique which uses category selection to perform test reduction on tens of millions of industrial tests is proposed. The results were promising. Two category selection approaches MI and IP were used in $CSBART$. The results showed that the effect of MI was better than IP. We also compared $CSBART$ with NCS using K-Means clustering. The results showed that $CSBART$ can get better performance than NCS which is more complex. We find that category selection is a good way to improve $LART$ when the tests have the formats of categories and choices.

Though $CSBART$ performs better on billing system in Huawei, there are still some limitations of $CSBART$. $CSBART$ only be applied on industrial systems which have a clear input domain and the distance between tests could be calculated. Moveover, failure region in industrial systems should be compact. For one fault, there should specific number of tests that could detect this fault. In the future, we try to overcome these hindrances of $CSBART$ practical application. We will also study more category selection approaches to improve $CSBART$'s ability of selecting faults revealing tests.

Acknowledgment

The work is supported, in part, by National Key R&D Program of China (Grant No. 2018YFB1003900), by Key Laboratory of Safety-Critical Software (Nanjing University of Aeronautics and Astronautics), Ministry of Industry and Information Technology (Grant No. XCA17007-04), by Natural Science Foundation of China (Grant No. 61772259) and by the Science and Technology Planning Project of Jiangsu Province (Grant No. BY2016003-02).

References

[1] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, no. 2, pp. 67–120, 2012.
[2] G. Rothermel, M. J. Harrold, J. Von Ronne, and C. Hong, "Empirical studies of test-suite reduction," *Software Testing, Verification and Reliability*, vol. 12, no. 4, pp. 219–249, 2002.
[3] T. Y. Chen, F.-C. Kuo, D. Towey, and Z. Q. Zhou, "A revisit of three studies related to random testing," *Science China Information Sciences*, vol. 58, no. 5, pp. 1–9, 2015.

[4] A. C. Barus, T. Y. Chen, F.-C. Kuo, H. Liu, R. Merkel, and G. Rothermel, "A cost-effective random testing method for programs with non-numeric inputs," *IEEE Transactions on Computers*, vol. 65, no. 12, pp. 3509–3523, 2016.
[5] W. Dickinson, D. Leon, and A. Podgurski, "Finding failures by cluster analysis of execution profiles," in *Proceedings of the 23rd international conference on Software engineering*. IEEE Computer Society, 2001, pp. 339–348.
[6] W. Zhao, H. Ma, and Q. He, "Parallel k-means clustering based on mapreduce," in *Cloud Computing*. Springer, 2009, pp. 674–679.
[7] 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, 1988.
[8] P. Bourque, R. E. Fairley *et al.*, *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*. IEEE Computer Society Press, 2014.
[9] K. P. Chan, T. Y. Chen, and D. Towey, "Restricted random testing," in *Software QualityłECSQ 2002*. Springer, 2002, pp. 321–330.
[10] T. Y. Chen, H. Leung, and I. Mak, "Adaptive random testing," in *Advances in Computer Science-ASIAN 2004. Higher-Level Decision Making*. Springer, 2005, pp. 320–329.
[11] E. S. Raymond, *The art of Unix programming*. Addison-Wesley Professional, 2003.
[12] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *The Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.
[13] Q. Shi, Z. Chen, C. Fang, Y. Feng, and B. Xu, "Measuring the diversity of a test set with distance entropy."
[14] Z. Wang and L. Chen, "Improved metrics for non-classic test prioritization problems." in *SEKE*, 2015, pp. 562–566.
[15] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping, "Use of the zero norm with linear models and kernel methods," *The Journal of Machine Learning Research*, vol. 3, pp. 1439–1461, 2003.
[16] N. Martin and H. Maes, *Multivariate analysis*. Academic press London, 1979.
[17] M. Hall, G. Holmes *et al.*, "Benchmarking attribute selection techniques for discrete class data mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 15, no. 6, pp. 1437–1447, 2003.
[18] T. Gorschek, P. Garre, S. Larsson, and C. Wohlin, "A model for technology transfer in practice," *IEEE software*, vol. 23, no. 6, pp. 88–95, 2006.
[19] J. R. Horgan and S. London, "A data flow coverage testing tool for c," in *Assessment of Quality Software Development Tools, 1992., Proceedings of the Second Symposium on*. IEEE, 1992, pp. 2–10.
[20] ——, "Data flow coverage and the c language," in *Proceedings of the symposium on Testing, analysis, and verification*. ACM, 1991, pp. 87–97.
[21] A. J. Offutt, J. Pan, and J. M. Voas, "Procedures for reducing the size of coverage-based test sets," in *In Proc. Twelfth Int'l. Conf. Testing Computer Softw*. Citeseer, 1995.
[22] S. Tallam and N. Gupta, "A concept analysis inspired greedy algorithm for test suite minimization," in *ACM SIGSOFT Software Engineering Notes*, vol. 31, no. 1. ACM, 2005, pp. 35–42.
[23] M. Marré and A. Bertolino, "Using spanning sets for coverage testing," *Software Engineering, IEEE Transactions on*, vol. 29, no. 11, pp. 974–984, 2003.
[24] K. F. Fischer, "A test case selection method for the validation of software maintenance modifications," in *Proceedings of COMPSAC*, vol. 77, 1977, pp. 421–426.
[25] K. Fischer, F. Raji, and A. Chruscicki, "A methodology for retesting modified software," in *Proceedings of the National Telecommunications Conference B-6-3*, 1981, pp. 1–6.
[26] D. Towey, Y. Dong, C.-A. Sun, and T. Chen, "Metamorphic testing as a test case selection strategy," *Science China Information Sciences*, pp. 1–2, 2016.
[27] W. E. Wong, J. R. Horgan, S. London, and A. P. Mathur, "Effect of test set minimization on fault detection effectiveness," in *Software Engineering, 1995. ICSE 1995. 17th International Conference on*. IEEE, 1995, pp. 41–41.
[28] C. Fang, Z. Chen, and B. Xu, "Comparing logic coverage criteria on test case prioritization," *Science China Information Sciences*, vol. 55, no. 12, pp. 2826–2840, 2012.
[29] F.-C. Kuo *et al.*, *On adaptive random testing*. Swinburne University of Technology, Faculty of Information & Communication Technologies, 2006.
[30] R. G. Merkel *et al.*, "Analysis and enhancements of adaptive random testing," Ph.D. dissertation, Swinburne University of Technology, 2005.
[31] 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*. ACM, 2008, pp. 71–80.
[32] Y. Lin, X. Tang, Y. Chen, and J. Zhao, "A divergence-oriented approach to adaptive random testing of java programs," in *Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2009, pp. 221–232.
[33] B. Zhou, H. Okamura, and T. Dohi, "Enhancing performance of random testing through markov chain monte carlo methods," *IEEE Transactions on Computers*, vol. 62, no. 1, pp. 186–192, 2013.
[34] A. Arcuri and L. Briand, "Adaptive random testing: An illusion of effectiveness?" in *Proceedings of the 2011 International Symposium on Software Testing and Analysis*. ACM, 2011, pp. 265–275.
[35] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B. Cohen, W. Grieskamp, M. Harman, M. J. Harrold, P. Mcminn, A. Bertolino *et al.*, "An orchestrated survey of methodologies for automated software test case generation," *Journal of Systems and Software*, vol. 86, no. 8, pp. 1978–2001, 2013.
[36] T. Y. Chen and R. Merkel, "An upper bound on software testing effectiveness," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 17, no. 3, p. 16, 2008.
[37] T. Y. Chen, F.-C. Kuo, and Z. Q. Zhou, "On favourable conditions for adaptive random testing," *International Journal of Software Engineering and Knowledge Engineering*, vol. 17, no. 06, pp. 805–825, 2007.
[38] T. Y. Chen, F.-C. Kuo, and R. Merkel, "On the statistical properties of testing effectiveness measures," *Journal of Systems and Software*, vol. 79, no. 5, pp. 591–601, 2006.
[39] K. P. Chan, T. Y. Chen, and D. Towey, "Forgetting test cases," in *null*. IEEE, 2006, pp. 485–494.
[40] N. Nachar *et al.*, "The mann-whitney u: A test for assessing whether two independent samples come from the same distribution," *Tutorials in Quantitative Methods for Psychology*, vol. 4, no. 1, pp. 13–20, 2008.

Zhiyi Zhang received the M.Sc degree and the Ph.D. degree both in Software Engineering from Nanjing University, Nanjing, China, in 2011 and 2016, respectively. He is currently an assistant professor in the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. His research interests include software engineering and software testing.

Yabin Wang is a Ph.D. student at the Software Institute, Nanjing University, underthe supervision of Prof. Zhenyu Chen and Prof. Bin Luo. His research interest is soft-ware testing. He visited the University of Texas at Dallas as a visiting student underthe supervision of Prof. Eric Wong in 2012.

Ziyuan Wang is an Associate Professor at School of Computer, Nanjing University of Posts and Telecommunications. He received his B.S. in Mathematics and Ph.D. in computer science from Southeast University. He worked as a Postdoctoral Researcher at Department of Computer Science and Technology, Nanjing University. His research interest mainly focuses on automated software testing techniques.

Ju Qian received the BSc and PhD degrees in computer science from Southeast University, China, in 2003 and 2009, respectively. He was a visiting scholar at Stony Brook University in 2014. He is currently an Associate Professor in the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, China. His research interests include program analysis, software testing, debugging, and evolution.

Dear Editors and Reviewers

Thank you for your valuable comments. We have significantly revised the paper based on your comments. The major changes include:

- Modified confusing descriptions (including contribution, background, experiment study and so on) in this paper, making it easy to be understood.
- Modified experimental studies, including research questions, evaluation methods, experiment implementation and experimental results and analysis.
- Fixed some presentation issues.

Also submitted along with the revised paper is a response detailing how each of the comments has been carefully addressed.

We sincerely hope that the revised paper and the response can satisfy all the concerns that the reviewers had and help you make a favorable decision on the acceptance of this paper. All the contents in the response with significant revision are highlighted in blue for your convenience.

Please let us know if you need any additional information.

Sincerely,
Zhiyi Zhang
Yabin Wang, Ziyuan Wang, and Ju Qian

1. Response to Reviewer 1

Question 1

First of all, it is still not very clear how the experiments were designed. Section V.A appears to mention some details of experimental settings, but it is not sufficient, nor written in a scientific way.

Answer 1

Thanks for the suggestions. We have modified Section V to make it easy to be understood. Section V.A introduces three research questions. Section V.B proposes the methods to evaluate our method "*CSBART*". Section V.C shows how the experiments were designed, including objects selectin, experiments setting and so on. Experiment results and analysis are illustrated in Section V.D. Six testers in Huawei hold a review meeting and the discussion results are presented in Section V.E.

Question 2

The writing style of research questions is weird. Research questions are the fundamental part of experimental studies, so they have to be defined in a very rigid and formal way. For example, the current writing of RQ1 "Whether CSBART can beat RT and LART" may be rewritten to "Can and to what extent CSBART outperform RT and LART in terms of fault coverage rate?". Similar problems must be addressed for RQ2 and RQ3.

Answer 2

Thank you for pointing out this issue. We have modified our research questions as following,

"RQ1. Can *CSBART* outperform *RT* and *LART* in terms of fault detection capability?
RQ2. Can *CSBART* outperform *NCS* in terms of fault detection capability?
RQ3. Can *CSBART* outperform *RT* and *LART* in terms of time cost?
RQ4. Can *CSBART* outperform *NCS* in terms of time cost? "

RT and *LART* are two popular test case generation techniques, and their ideas are suitable for selecting tests from fifty-million tests in Huawei. *NCS* is a popular test selection technique. So we compared them with *CSBART* in terms of fault detection capability (including fault coverage and speed of fault detection) in RQ1 and RQ2. Moreover, since time cost is very important in industry and in order to verify the

efficiency of *CSBART*, we also compared the time cost of these techniques in RQ3 and RQ4.

Question 3

Moreover, all the experimental settings must be defined explicitly and clearly.

Answer 3

Thanks for the suggestions. We have modified Section V to make it easy to be understood. All the experimental settings, such as the number of selected categories, experiments repeat times and so on, have been shown in Section V.C.

Question 4

In addition to the research questions, what are the object(s)? how and why were the objects selected? what are the independent/dependent variables? what is the detailed experimental procedure, e.g. how random test cases were generated (which is a non-trivial task for complex systems)? what is the environment for experiments? The reviewer could find some information for part of the above questions, but not all. More importantly, all the information must be given in a well-structured way to avoid any misunderstanding.

Answer 4

In our experiments, two Call-Detail-Records billing systems, which are from Vietnam and Paraguay, are used as experiment objects. We select these two systems for two reasons. One is that both projects of old versions have been applied in practice for some years, so there are sufficient tests to evaluate *CSBART*. The other is that these two systems are from two different countries, so tests used in our experiments have high diversity, which could ensure the accuracy of experiment results. There are about fifty-million tests in our experiments.

The detailed experimental procedure has been shown in Section V.C. "For *RT*, we selected tests randomly from original tests, and the number of tests is the same as *CSBART* ". We acknowledge that random test case generation is a non-trivial task for complex systems because the input domain of complex systems is hard to be confirmed. However, in this paper, we SELECT tests from original tests, rather than generate tests. Moreover, the input domain of objects have been confirmed, so random test case selection is not a non-trivial task in our experiments.

Because the telephone bill data involves privacy, all test data cannot be copied and could only run in the production environment of local telephone operators. The production environment of local telephone operators is not open to us. We can only ensure that all experiments are in the same production environment.

We have modified Section V, including research questions, evaluation methods, experiment implementation and experimental results and analysis, to make it easy to be understood by readers.

Question 5

Secondly, though the writings for contributions have been improved, they still need further revisions. For Contribution 1, the authors claim that the new technique can be used for large-scale systems with lots of test cases. In the response letter, they further stated that "ART techniques (including FSCS-ART, LART and so on) can only be applied to industrial systems with LIMITED SIZE of test suite." It is not so appropriate to have such a statement without appealing evidence and justification. The reviewer might agree that this is a problem for the naive FSCS-ART, but LART was proposed for addressing this problem. The authors must be careful making any claim.

Answer 5

We are sorry for no making ourselves understood in the previous response letter. We agree that *LART* could be applied to industrial systems with lots of test cases. However, *LART* is still NOT suitable for large-scale systems in Huawei for two reasons. One reason is that *LART* needs to consider all categories when calculating the distance between test cases, but for the under-test systems in Huawei, there are too many categories, so it will cost much time when using *LART*. The other reason is that each category reflects the characteristics of a dimension of the test case, in fact, only a few categories are related to test fail, and other categories have little effect on running results of tests. So we propose a new technique *CSBART* for large-scale systems with lots of test cases.

Question 6

In addition, what is the fundamental difference between Contributions 2 and 3? The current writings mentioned the time reduction and performance improvement of CSBART in both contributions. Then why did the authors repeat this contribution twice? The only difference is that Contribution 2 also mentioned the difference between MI and IP techniques, but is it a major contribution of the paper?

Answer 6

Contribution 2 is about how to select categories. In our method, since selected categories has a great impact on the results of *CSBART*, so how to score categories is very important. In this paper, we apply two methods *MI* and *IP* to score categories, which could help *CSBART* to detect faults and touch failure categories more rapidly.

Contribution 3 is about the application of *CSBART*. We have used *CSBART* in two industrial real-life systems with tens of millions of tests in Huawei and our results showed *CSBART* could reduce much time cost and detect fault more rapidly.

Question 7

Overall, I still appreciate the work conducted in the paper, especially because it is the application of ART into industrial real-life systems. However, the writing must be significantly improved to make the work more understandable and thus more acceptable by the community.

Answer 7

Thank you very much for pointing out this issue. We have carefully proofread our paper and fixed as many such presentation issues as possible.

==========

2. Response to Reviewer 2

The current version has well addressed my previous concerns with the work. I thus recommend to accept the current version of the paper.

Answer

Thanks for the comment.

==========

3. Response to Reviewer 3

Question 1

Sufficient contribution?
The extension is fairly trivial and while the large-scale industrial system ensures a realistic setting and interesting data it is not clear that the proposed algorithm is an advance over already published techniques.

Answer 1

In previous work, considering testing cost, *ART* techniques (including *FSCS-ART* and so on) can only be applied to industrial systems with LIMITED SIZE of test suite. For large-scale industrial programs, existing *ART* techniques are not suitable because cost is very expensive. *LART* could be applied to industrial systems with lots of test cases. However, *LART* is still NOT suitable for large-scale systems in Huawei for two reasons. One reason is that *LART* needs to consider all categories when calculating the distance between test cases, but for the under-test systems in Huawei, there are too many categories, so it will cost much time when using *LART*. The other reason is that each category reflects the characteristics of a dimension of the test case, in fact, only a few categories are related to test fail, and other categories have little effect on running results of tests.

To solve this problem, in our paper, we proposed "*CSBART* ", which could be used for large-scale industrial programs with tests of tens of millions firstly. What' more, our work means that *ART* could also be used for large-scale industrial real-life systems. Comparing with test sizes of industrial programs in previous work, tens of millions is a greatly improvement of test size. Moreover, our technique has been used in Huawei with hundreds of millions of tests. So in our opinion, our contribution of this paper is sufficient.

==== Major comments on sections ===

Question 2

Writing and language is a problem throughout to the level that it detracts from understanding. I give more detailed examples below but the writing is far from being acceptable for publication at present. Problems are too numerous to list them all; language needs to be improved throughout, probably with the help of native English people.

Answer 2

Thanks very much for pointing out this issue. We have carefully proofread our paper and fixed as many such presentation issues as possible.

Question 3

There is also a lack of precision in how algorithms and technical details are expressed. This could partly/mainly be a language problem (see above) but it can also be said to be sloppy or lacking rigor. As one example, in Algorithm 1 you state the "Input:" to be "Test input". This does not help clarify which information your method needs. As another example, in algorithm 2 you use FT to count the number of times something passed while PT is used to counter the number of times something failed (probably a typo?

Answer 3

Thank you for pointing out this issue. We have carefully proofread our paper and fixed as many such presentation issues as possible. For example, in algorithm 1, we have modified the "Input:" to be "Original input". And we modified "generate" in line 4, line 6 and line 25 in algorithm 1 to "select". In algorithm 2, we have changed to using PT to count the number of times tests passed while FT is used to counter the number of times tests failed.

Question 4

Descriptions are also frequently unnecessarily complex. For example, the category scoring procedure IP seems to simply be calculating the percentage of choices of values for the category that always lead to failure. By not describing this in simple terms but rather stating an algorithm for it without giving an intuitive explanation the paper wastes a lot of space as well as time of the reader. However, you do not state details about how you ranked the categories given the values returned by the scoring functions (selected the ones with largest or smallest values).
Another example of unnecessary complexity: It is not clear why you needed to implement MapReduce for selecting between on the order of 10 million test cases. Most modern-day computers should be able to process such datasets even fully in memory.

Answer 4

Thanks for the suggestions. We have modified our paper and tried our best to fix unnecessarily complex descriptions.

In our method, since selected categories has a great impact on the results of *CSBART*, so we use an algorithm to have a more accurate and detailed description of the scoring procedure *IP*, to make readers understand the selection method better. Moreover, whether using *IP* or *MI*, all categories are ranked according to the scores, and then testers select certain number of categories with largest values.

We give an example to explain the algorithm, which could help readers to understand our method easier. The example is as following,

TABLE III: An example to determine failure related categories

(a) Test inputs

	F	F	F	F	P	P	P	P	P
A	a_1	a_2	a_1	a_2	a_2	a_2	a_3	a_3	a_3
B	b_1	b_2	b_3	b_4	b_1	b_1	b_1	b_1	b_1
C	c_1	c_2	c_3	c_3	c_4	c_1	c_2	c_2	c_2
D	d_1	d_3	d_2	d_1	d_1	d_2	d_2	d_3	d_1

(b) Determine failure related category

B	F	P	*Failure related choice*
b_1	1	5	Not
b_2	1	0	Yes
b_3	1	0	Yes
b_4	1	0	Yes

For example, there are six tests that the value of category *B* is set to b_1, five of them are pass and one of them is fail. Since the number of passed tests are larger than of failed tests, according to the definition of failure related choices, b_1 is not failure related. b_2, $b_3$and b_4 in Table III (b) are failure related. IP takes the ratio of failure related choices as the score of one category. Since category *B* has three failure-related choices, and one choice that is not failure related, so the score of category *B* is 3/(1+3)=0.75. The scores of category *A*, *C* and *D* is 0.33, 0.5 and 0.33, respectively, so the categories are ranked as *B*, *C*, *A* and *D*. If testers want to select two categories, category *B* and *C* will be selected.

We implement MapReduce for two reasons. One is that only few PCs in Huawei can process such datasets, but the process still needs a lot of time which is not accepted by Huawei. The other is that we consider the scalability of the method. With the application of systems, the size of tests is becoming bigger and bigger. We hope our method is still valid when the number of tests is much larger than 10 million. Actually, the number of tests of undertest systems in Huawei has been reached hundreds of millions right now, and our method is still valid. But to make this paper be read easier, we have deleted the introduction of MapReduce.

Question 5

The problem and situation you focus on is one that Combinatorial Interaction Testing (CIT) has been targeting for quite some years now. It is not clear that ART is the most natural or only type of testing technique applicable here and I expect you to clarify why you have not compared to and evaluated your results against/with CIT. At least you need to discuss this early, when motivating your choice of looking into speeding up ART-based techniques and also to describe and discuss related work on CIT and how it relates to your results.

Answer 5

Thanks for the suggestion. We have not considered to compare *CSBART* with *CIT* for two reasons. One is that *CIT* is usually used for test case generation, but *CSBART* is used for test case selection. The other is though the idea of *CIT* can be used for test case selection, but there are too much choices and categories in tests, so the choice and category coverage will be low if we use *CIT*.

Question 6

Also your description of the industrial setting for your investigation is not clearly natural for ART since you do not seem free to generate new test data freely; rather you can only select from the test data that was already in the recorded data from the one month of use. This discrepancy should be avoided and you need to more clear explain if you are doing simply selection among the recorded data or if you can actually generate new tests / test data freely.

Answer 6

Thanks for the suggestion. Our method is used for test case selection from existing tests. Though *ART* is originally used for test generation, we use the idea of *ART* for test selection and the new selected test has the largest distance with tests which have been selected. We have modified description of our paper, such as the industrial setting, the Algorithm 1 and so on, to avoid the discrepancy.

Question 7

You don't describe LART in a good way in Section IIB. A reader would need to go to the original paper to understand it in detail. You talk about maintaining the S vector of counts of the number of times a category has been chosen for a certain category. But you don't describe for which set of test cases these counts are to be updated. For the testcases in E or for the set of candidate test cases etc? Your introduction of LART also doesn't build any intuition about why LART is constructed in this way and why it leads to reductions. Since I know about ART and understands more about the context I can guess what you mean but a reader without specific such knowledge would likely struggle.

Answer 7

Thanks for the suggestion. We have modified Section II.B to make reader understand easier. In *LART*, S vector records the number of times a choice has been chosen for a

certain category, so it counts for the tests that have been selected in E, which means for test cases in E, the counts will be updated in S vector. Moreover, in our paper, *LART* is constructed in the way that showed in [1]. In Equation 1, the number of comparison operations when calculating *dist(c,E)* is n * g, while the number is g in Equation 3, so *LART* leads to reductions.

Question 8

It is not clear how your method will cope with well-tested systems such as when the category selection methods get very little data for selecting failure-revealing categories. If very few test cases are expected to fail (which is more likely in regression testing where the daily changes to a system are relatively small compared to the size of the system overall) there is not enough information to rank the categories.

Answer 8

Our method is not suitable for systems without enough information to rank the categories. We do not consider this situation for two reasons. One is that in Huawei, most time is consumed in the analysis of failed tests, such as 2 days of running all tests costs and more than 48 days of analyzing failed tests costs (we have written in Section I). If there are only few failed test cases, the time cost for analysis is also small, so it do not need our method in this situation. The other is that the undertest systems in Huawei is complex and large-scale, so it is very difficult to update every day. These systems are usually updated once a week or every month, so they have enough information to rank categories, where our method is suitable for it.

Question 9

Details are lacking of how you used NCS in the empirical evaluation.

Answer 9

In the second paragraph of Section V.C, we show the *NCS* details in our experiments. "For *NCS*, we clustered original tests using K-means algorithm. According to [16], the number of clusters is set to $\sqrt{\frac{N_t}{2}}$, N_t is the number of original test cases. In our experiments, the number of original test cases is fifty-million, so the number of cluster is five-thousand. Then we randomly selected the same number of tests from each cluster, and the whole number of tests are the same as tests selected by *CSBART*. For example, if we selected one million tests by *CSBART,* when using *NCS*, two-hundred tests should be randomly selected from each cluster. "

Question 10

Your empirical evaluation lack detailed comparison of the studied techniques. It is not clear what you mean when you say that on average one technique was 10% better than another. You should probably use area-under-the-curve or similar, quantitative methods to evaluate how they compare to each other.

Answer 10

Thanks for the suggestions. We have given the *APFD* values of *CSBART, RT, LART* and *NCS*. From the results, we could find that find that *CSBART* has larger *APFD* value than *LART* and *RT*, which means it could detect faults faster than *LART* and *RT*. The *APFD* value of *CSBART−MI* is larger than *NCS*, but slightly less than *NCS−MI*. The *APFD* value of *CSBART−IP* is slightly less than *NCS* and *NCS−MI*. So the fault detection speed of *CSBART−MI/IP* are similar as of *NCS−MI/IP*.

Moreover, we compare these techniques with fault detection capability, including fault coverage and fault detection speed. We also explain what on average one technique was 10% better than another means. For example, when using fault coverage to evaluate technique *A* and *B*, after they select the same number of tests (such as 10%), if tests selected by *A* find 25% faults while tests selected by *B* find 15% faults, we say that *A* technique was 10% better than *B*. In this version, we have showed the detailed comparison, such as "*CSBART−IP* can get 10% fault coverage rate higher than *LART* at best", so that the readers could understand our paper better.

Question 11

You also don't give details on the time needed for NCS and compared to NCS your method has basically the same results. It is not clear that it is an advance.

Answer 11

We have shown the details on the time needed for *NCS* in Table VI. When consider fault coverage and *APFD*, the results of *CSBART* are similar as of *NCS*. But if we consider time cost, from Table VI, we could find that *NCS* needs too much time (nearly thirty times of *CSBART*) to select tests, which is unacceptable to Huawei. So in summary, we believe that our method is an advance.

Question 12

The times to select the tests seem small and can only be considered large in relation to the time it takes to run a test. This information is missing.

Answer 12

Thank you for pointing out this issue. We have showed the time cost of selecting tests in Table VI. The times to select the tests are small in our experiments. Most of time cost are spent on analysis. So we need to select part of tests to reduce analysis cost. From Table VII, we can find that when using our method, the time cost has been reduced from 5.5 days to 14.8 days.

Moreover，because each technique selects the same number of tests, the times of executing selected tests are similar. Compared with the times to select the tests and analysis, this differences are negligible, so we don't compared the time of executing selected tests.

==== Minor comments on sections ===

Question 13

Not clear what the weights on the edges in Fig 4 represents.

Answer 13

The weights on the edges in Fig 4 represents the number of different choices compared to T_1. For example, compared with T_1, T_2 has two different choices (b_2 and d_2), so weight on the edge connecting T_1 and T_2 is 2.

=== Language issues ===

Question 14

Language issues:
* Summary of language/writing:
There are many problems with your writing and language which makes it hard to fully understand the paper. I urge you to improve your English writing skills and to involve a native English speaker to help you review your papers before submitting them.

Example of language issues of different types below:

* Simplistic/non-scientific/imprecise language; rephrase:
Abstract: When you say "select partial tests", it is not precise what you mean with partial tests since it is more likely to be run as only running parts of tests rather than running fewer of them. I don't yet know what you mean when you write this but you need to avoid such ambiguous language and expressions.

You use "non-numerical inputs" throughout to refer to categorical/discrete data. This is not directly wrong but it is misleading since the opposite of non-numerical data is often seen as structured data such as XML files, graphs, etc. But that is not the focus of your technique. So you should clarify this early in the paper.

You repeatedly say that there are 10s of millions of tests in the industrial system you investigate but it is not clear if you refer to test executions or test cases or something else. You need to be clear and precise.

Incomprehensible/ambiguous sentence; rephrase:

Wrong tempus/tempii & simplistic language; reword:
Abstract: "tests in industrial scenario", either "tests in an industrial scenario" or "tests in industrial scenarios"

Change wording in following terms/words/sentences:
- "which is done by manual" => "which is done manually"
- "To our known" => "To the best of our knowledge"
- "It is obviously from Equation 3" => "It is obvious from Equation 3"

Typos:
- Abstract: "lure" in "lure numbers". Do you mean huge or large?

Answer 14

Thank you very much for pointing out these issues. We have carefully proofread our paper and fixed as many such presentation issues as possible.

Reference

[1] A. C. Barus, T. Y. Chen, F.-C. Kuo, H. Liu, R. Merkel, and G. Rothermel, "A cost-effective random testing method for programs with non-numeric inputs," IEEE Transactions on Computers, vol. 65, no. 12, pp. 3509– 3523, 2016.

==========