IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received May 22, 2021, accepted June 2, 2021, date of publication June 7, 2021, date of current version June 16, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3087165

A Comparison of Fault Detection Efficiency Between Adaptive Random Testing and Greedy Combinatorial Testing for Control Logics in Nuclear Industrial Distributed Control Systems

CHENG ZONG[1,2,3], YANLIANG ZHANG[2], YONGMING YAO[4], SHIYONG SHUANG[5], AND ZIYUAN WANG[2]

[1]Command and Control Engineering College, Peoples Liberation Army Engineering University, Nanjing 210007, China
[2]School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China
[3]Jiangsu Nuclear Power Corporation, Lianyungang 222042, China
[4]Tongda College, Nanjing University of Posts and Telecommunications, Yangzhou 225127, China
[5]Beijing Jinghang Research Institute of Computing and Communications, Beijing 100074, China

Corresponding author: Ziyuan Wang (wangziyuan@njupt.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1403400, and in part by the Research Project of Tongda College of Nanjing University of Posts and Telecommunications under Grant XK004XZ18006, Grant xk006xz19013, Grant XK004XZ19003, and Grant JG20120024.

ABSTRACT Due to the complexity of the nuclear industrial distributed control system (DCS), input-domain testing techniques, including random testing and combinatorial testing, are usually utilized to test the control logics in nuclear industrial DCS. To improve the fault detection efficiency of random testing, the adaptive random testing technique selects a test case that significantly differs from all existing test cases. Similarly, to improve the fault detection efficiency of combinatorial testing, the greedy combinatorial testing technique adopts a greedy strategy to generate test cases that cover more uncovered tuple-combinations of parametric values. In this paper, we designed an experiment to compare the fault detection efficiency between adaptive random testing technique and greedy combinatorial testing technique for control logics of nuclear industrial DCS. Through the analysis of the fault detection ratios, the f-measure values, and the values of average percent of faults detected (APFD) on two experimental subjects, including the commonly used benchmarks in the field of Boolean-specification testing as well as a group of Boolean expressions extracted from the control logics in nuclear industrial DCS, the experimental results give us the following conclusions: (1) If the test suites' sizes are relatively small, the fault detection efficiencies of the two techniques are very close though there is a slight advantage in adaptive random testing; (2) With the gradual increase of test suites' sizes, the fault detection efficiency of greedy combinatorial testing is beyond adaptive random testing gradually. Such a result can help us select the appropriate testing techniques in the testing of the control logics in nuclear industry DCS.

INDEX TERMS Input-domain testing, fault detection efficiency, adaptive random testing, combinatorial testing, nuclear industrial DCS.

I. INTRODUCTION

The nuclear industrial distributed control system (DCS) is applied in the field of safety, which takes the initiative and protective action to protect the critical core parts of nuclear power units from damage when some crucial parameters of nuclear power units exceed the limitation (such as pressure or pressure difference, temperature or temperature difference, liquid level, etc.), and reduce the probability of nuclear accidents. The fault diagnosis results for the nuclear industrial distributed control system (DCS) show that the root causes of a large number of faults are incorrect design or incorrect implementation of control logics in this type of system [1]. To ensure the correctness of the control logics in the nuclear

The associate editor coordinating the review of this manuscript and approving it for publication was Zhaojun Li.

industrial DCS and to ensure the reliability of the nuclear industrial DCS, it is necessary to conduct high-quality testing on these systems. Due to the complexity of the nuclear industrial DCS, the code-based testing techniques, model-based testing techniques, and fault-based testing techniques are extremely difficult to implement in real scenarios. Therefore, the input-domain testing techniques are usually used to test the control logics in nuclear industrial DCS.

The input-domain software testing techniques include random testing, combinatorial testing, etc [2]. To improve the fault detection speed of random testing, the adaptive random testing technique selects a test case that differs significantly from all existing test cases [3]. Similarly, to improve the fault detection efficiency of combinatorial testing using a small combinatorial test suite, the greedy combinatorial testing technique adopts a greedy one-test-at-a-time strategy to generate test cases that cover more uncovered tuple-combinations of parametric values [4]. When generating test cases one-by-one in adaptive random testing or greedy combinatorial testing, the difference between the current test case and the existing test cases is required to be as large as possible: in the fixed-size-candidate-set adaptive random testing (FSCS-ART) that were the first version of adaptive random testing, the distance between the current test case and existing test cases needs to be maximized [3]; in greedy combinatorial testing, the tuple-combinations of parametric values covered by the current test case and existing test cases need to be different. Therefore, due to such a similarity, which technique is more efficient in fault detection, adaptive random testing technique or greedy combinatorial testing technique?

People have conducted experimental studies on both adaptive random testing and combinatorial testing. In the empirical studies on adaptive random testing, people mainly compare adaptive random testing with pure random testing to verify whether the former testing technique improves fault detection speed [5], [6]. In the empirical studies on combinatorial testing, people usually focus on fault detection ability rather than fault detection efficiency [7]. People have compared the fault detection capabilities of adaptive random testing and combinatorial testing in empirical studies [8], but the comparison of the fault detection efficiencies of them seems to be lacking. When testing a nuclear industrial DCS, the cost of each test case's execution is high since it involves both software and hardware. To reduce testing costs, it is hoped that fewer test cases can be used to detect more faults; furthermore, it is also hoped that the efficiency of fault detection can be improved and faults can be detected faster.

Therefore, we compare fault detection efficiencies of adaptive random testing technique and greedy combinatorial testing technique in this paper. We conduct an experimental study using general-form Boolean-specifications and their mutants as experimental subjects in this paper. The Boolean-specifications used in the experiment are extracted from the well-known TCAS system [9] and a real nuclear industrial DCS, and the mutants are generated using ten mutation operators [10]. By comparing fault detection ratios, f-measure values, and the values of average percent of faults detected (APFD), we found that: (1) If the number of test cases is relatively small, the fault detection efficiencies of adaptive random testing and greedy combinatorial testing are very close, though the former has a slight advantage. (2) In the gradual increase in the number of test cases, greedy combinatorial testing's fault detection efficiency gradually becomes better. (3) If the testing resources are sufficient, the fault detection efficiency of greedy combinatorial testing has obvious advantages.

The main contributions of this paper include (1) Extracting a set of general-form Boolean expressions that reflect control logics in a real nuclear industry DCS; (2) Comparing fault detection efficiencies of adaptive random testing technique and greedy combinatorial testing technique in Boolean-specification testing; (3) Proposing a testing strategy that selects adaptive random testing technique and greedy combinatorial testing technique in real scenarios.

The rest of the paper is organized as follows. Preliminaries about adaptive random testing, greedy combinatorial testing, Boolean-specification testing, and nuclear industrial DCS are introduced in Section 2. Experiments including research questions, experimental designs, experimental results, and experimental findings, are described in Section 3. Threats to validity are analyzed in Section 4. Related works are described in Section 5. Finally, conclusions are given in Section 6.

II. PRELIMINARIES

Some preliminaries, including adaptive random testing, greedy combinatorial testing, and Boolean-specification testing, will be introduced in this section.

A. ADAPTIVE RANDOM TESTING

Adaptive random testing, an enhanced version of random testing, was proposed to improve the fault detection speed. Adaptive random testing is based on the assumption that "test cases near the failure test case are likely to be failure test cases too, and test cases near the pass test case are likely to be pass test cases too." Therefore, when using adaptive random testing methods to generate test cases, it is hoped that the distance between the current test case and the existing test case is as far as possible (or the difference is as large as possible). Currently, people often use Hamming distance to define the distance between two test cases.

The most well-known version of adaptive random testing is called fixed-size-candidate-set adaptive random testing (FSCS-ART) [3]. When using FSCS-ART algorithm to generate test cases, it is necessary to specify number s as the fixed scale of a set of candidate test cases. In each step, s test cases are randomly selected from the input-domain space to form a candidate set of test cases, and then a test case is selected from the candidate set to maximize the minimum Hamming distance between such a test case and all the existing test cases.

B. GREEDY COMBINATORIAL TESTING

Suppose there is n input variables $F = f_1, f_2, \ldots, f_n$, where each f_i has a set of input values $V_i = 1, 2, \ldots, a_i$ for $i = 1, 2, \ldots, n$ respectively. If the cardinalities of the sets of input values for all the input variables are uniform ($a_1 = a_2 = \ldots = a_n = a$), the testing subject is a fixed-level system with an input model $M = a^n$. A τ-way covering array $CA(m; \tau, n, a)$ is an $m \times n$ array on totally a symbols with the property that each $m \times \tau$ sub-array contains all possible τ-tuples from these a symbols at least once. Here τ is called the strength of such a fixed-level covering array. For a given testing subject with input model $M = a^n$, we could obtain a τ-way combinatorial test suite (or a combinatorial test suite with strength τ) from a τ-way a-level covering array by mapping each row in the covering array to a test case.

There are many combinatorial test generation algorithms proposed to generate a combinatorial test suite as small as possible since the small test suite with few test cases consumes less testing resources [11]. There are three kinds of combinatorial test generation algorithms: algebraic methods, greedy algorithms, and meta-heuristic search algorithms [12]. Most of the greedy algorithms use the one-test-at-a-time strategy [13]–[15]. In such a strategy, test cases are generated one-by-one until generated test cases cover all the required τ-tuples. It is usually called greedy combinatorial testing since each test case needs to cover uncovered tuple-combinations of parametric values as many as possible.

When generate a combinatorial test suite using a one-test-at-a-time greedy algorithm, one or more seed test cases can be specified to ensure that they must be included in the test suite. When calling a deterministic combinatorial test generation algorithm, we can diversify the test suite by setting different seed test cases. Besides, when generating a combination test suite with strength τ ($\tau > 2$), we can use the strength-incremental combinatorial testing strategy [16]. It means that a combinatorial test suite with strength 2 is generated and then used as seed test cases to generate a combinatorial test suite with strength 3. Such an operation repeats until a combinatorial test suite with strength τ is generated.

C. BOOLEAN-SPECIFICATION TESTING

A Boolean expression is a string that contains some Boolean input variables, logic operators '$\wedge$' (AND), '$\vee$' (OR), and '$\neg$' (NOT), the brackets '(' and ')'. In a program, Boolean expressions are usually used in predicate statements to determine such a program's execution paths. The purpose of Boolean-specification testing is to detect faults that may occur in these Boolean expressions. This paper mainly focuses on general-form Boolean expressions, which means that they are neither conjunction normal form (CNF) nor disjunction normal form (DNF).

Ten mutation operators are usually utilized to create mutants in the field of fault-based Boolean-specification testing [10]. They are explained in the following:

- `ASF`: associative shift fault is caused by the omission of a pair of brackets.
- `CCF`: clause conjunction fault is caused by replacing an occurrence of an input variable c with ($c \wedge c$'), where c' could be any possible input variable or its negation.
- `CDF`: clause disjunction fault is caused by replacing an occurrence of an input variable c with ($c \vee c$'), where c' could be any possible input variable or its negation.
- `ENF`: expression negation fault is caused by replacing a sub-expression with its negation.
- `LNF`: literal negation fault is caused by replacing an occurrence of an input variable with its negation. It may be called the variable negation fault (VNF).
- `LRF`: literal reference fault is caused by replacing an occurrence of an input variable with another variable or its negation. It may be called the variable reference fault (VRF).
- `MLF`: missing literal fault is caused by the omission of an occurrence of an input variable. It may be called the missing variable fault (MVF).
- `ORF`: operator reference fault is caused by replacing an occurrence of a logic connective $\wedge$ (or $\vee$) with $\vee$ (or $\wedge$).
- `SA0`: the stuck-at-0 fault is caused by replacing an occurrence of an input variable with logic constant FALSE (or 0).
- `SA1`: the stuck-at-1 fault is caused by replacing an occurrence of an input variable with logic constant `TRUE` (or 1).

III. EXPERIMENTS

This section describes our experiments that compare fault detection efficiencies of adaptive random testing and greedy combinatorial testing on general-form Boolean-specifications.

A. RESEARCH QUESTIONS

The following research questions will be answered in our experiments.

Research Question 1: In the case of limited testing resources, which testing technique could generate a test case sequence with higher fault detection efficiency, adaptive random testing technique, or greedy combinatorial testing technique? Here, limited testing resource means that only partial test cases in the exhaustive test set have a chance to run.

Research Question 2: In the case of unlimited testing resources, which testing technique could generate a test case sequence with higher fault detection efficiency, adaptive random testing technique, or greedy combinatorial testing technique? Here, unlimited testing resource means that all the test cases in the exhaustive test set have a chance to run.

B. METRICS

In order to compare fault detection efficiencies of the test case sequences generated by adaptive random testing technique and greedy combinatorial testing technique, three metrics

1	$a \wedge (\neg b \vee \neg c) \wedge d \vee e$
2	$\neg(a \wedge b) \wedge (d \wedge \neg e \wedge f \vee \neg d \wedge e \wedge \neg f \vee \neg d \wedge \neg e \wedge \neg f) \wedge (a \wedge c \wedge (d \vee e) \wedge h \vee a \wedge (d \vee e) \wedge \neg h \vee b \wedge (e \vee f))$
3	$\neg(c \wedge d) \wedge (\neg e \wedge f \wedge \neg g \wedge \neg a \wedge (b \wedge c \vee \neg b \wedge d))$
4	$a \wedge c \wedge (d \vee e) \wedge h \vee a \wedge (d \vee e) \wedge \neg h \vee b \wedge (e \vee f))$
5	$\neg e \wedge f \wedge \neg g \wedge \neg a \wedge (b \wedge c \vee \neg b \wedge d)$
6	$(\neg a \wedge b \vee a \wedge \neg b) \wedge \neg(c \wedge d) \wedge \neg(g \wedge h) \wedge ((a \wedge c \vee b \wedge d) \wedge e \wedge (f \wedge g \vee \neg f \wedge h))$
7	$(a \wedge c \vee b \wedge d) \wedge e \wedge (f \wedge g \vee \neg f \wedge h)$
8	$(a \wedge ((c \vee d \vee e) \wedge g \vee a \wedge f \vee c \wedge (f \vee g \vee h \vee i)) \vee (a \vee b) \wedge (c \vee d \vee e) \wedge i) \wedge \neg(a \wedge b) \wedge \neg(c \wedge d) \wedge \neg(c \wedge e) \wedge \neg(d \wedge e) \wedge \neg(f \wedge g) \wedge$ $\neg(f \wedge h) \wedge \neg(f \wedge i) \wedge \neg(g \wedge h) \wedge \neg(h \wedge i)$
9	$a \wedge (\neg b \wedge \neg c \vee b \wedge c \wedge \neg(\neg f \wedge g \wedge h \wedge \neg i \vee \neg g \wedge h \wedge i) \wedge \neg(\neg f \wedge g \wedge l \wedge k \vee \neg g \wedge \neg i \wedge k)) \vee f$
10	$a \wedge ((c \vee d \vee e) \wedge g \vee a \wedge f \vee c \wedge (f \vee g \vee h \vee i)) \vee (a \vee b) \wedge (c \vee d \vee e) * i$
11	$(\neg a \wedge b \vee a \wedge \neg b) \wedge \neg(c \wedge d) \wedge \neg(g \wedge h) \wedge \neg(i \wedge j) \wedge ((a \wedge c \vee b \wedge d) \wedge e \wedge (\neg i \vee \neg g \wedge \neg k \wedge \neg j \wedge (\neg h \vee \neg k)))$
12	$(a \wedge c \vee b \wedge d) \wedge e \wedge (\neg i \vee \neg g \wedge \neg k \vee \neg j \wedge (\neg h \vee \neg k))$
13	$(\neg a \wedge b \vee a \wedge \neg b) \wedge \neg(c \wedge d) \wedge (f \wedge \neg g \wedge \neg h \vee \neg f \wedge g \wedge \neg h \vee \neg f \wedge \neg g \wedge \neg h) \wedge \neg(j \wedge k) \wedge ((a \wedge c \vee b \wedge d) \wedge e \wedge (f \vee (i \wedge (g \wedge j \vee h \wedge k))))$
14	$(a \wedge c \vee b \wedge d) \wedge e \wedge (f \vee (i \wedge (g \wedge j \vee h \wedge k)))$
15	$(a \wedge (\neg d \vee \neg e \vee d \wedge e \wedge \neg(\neg f \wedge g \wedge h \wedge \neg i \vee \neg g \wedge h \wedge i) \wedge (\neg f \wedge g \wedge l \wedge k \vee \neg g \wedge \neg i \wedge k)) \vee \neg(\neg f \wedge g \wedge h \wedge \neg i) \wedge \neg(\neg f \wedge g \wedge l \wedge k \vee \neg g \wedge$ $\neg i \wedge k) \wedge (b \vee c \wedge \neg m \vee f)) \wedge (a \wedge \neg b \wedge \neg c \vee \neg a \wedge b \wedge \neg c \vee \neg a \wedge \neg b \wedge c)$
16	$a \vee b \vee c \vee \neg c \wedge \neg d \wedge e \wedge f \wedge \neg g \wedge \neg h \vee i \wedge (j \vee k) \wedge \neg l$
17	$(a \wedge (\neg d \vee \neg e \vee d \wedge e \wedge \neg(\neg f \wedge g \wedge h \wedge \neg i \vee \neg g \wedge h \wedge i) \wedge (\neg f \wedge g \wedge l \wedge k \vee \neg g \wedge \neg i \wedge k)) \vee \neg(\neg f \wedge g \wedge h \wedge \neg i) \wedge \neg(\neg f \wedge g \wedge l \wedge k \vee \neg g \wedge$ $\neg i \wedge k) \wedge (b \vee c \wedge \neg m \vee f))$
18	$a \wedge \neg b \wedge \neg c \wedge d \wedge \neg e \wedge f \wedge (g \vee \neg g \wedge (h \vee i)) \wedge \neg(j \wedge k \vee \neg j \wedge l \vee m)$
19	$a \wedge \neg b \wedge \neg c \wedge (\neg(f \wedge (g \vee \neg g \wedge (h \vee i))) \vee f \wedge (g \vee \neg g \wedge (h \vee i)) \wedge \neg d \wedge \neg e) \wedge \neg(j \wedge k \vee \neg j \wedge l \wedge \neg m)$
20	$a \wedge \neg b \wedge \neg c \wedge (f \wedge (g \vee \neg g \wedge (h \vee i)) \wedge (\neg e \wedge \neg n \vee d) \vee \neg n \wedge (j \wedge k \vee \neg j \wedge l \wedge \neg m))$

FIGURE 1. General-form Boolean expressions extracted from the TCAS system.

including fault detection ratio, f-measure, and APFD, will be utilized in our experiment.

Fault detection ratio: For a group of faults, the fault detection ratio is the percentage of faults detected by a given test suite. If there are two test suites of the same size, the higher the test suite's fault detection ratio is, the higher the corresponding test generation technique's fault detection efficiency is.

F-measure value: For a given fault, the f-measure value is the test case index that first triggers such a fault in the test case sequence [17]. In both the case of limited testing resources and unlimited testing resources, the less f-measure value means the higher fault detection efficiency.

APFD value: For a group of faults, the APFD value illustrates how rapidly a test suite detects these faults [18]. If two different test case sequences contain the same number of test cases and detect all the faults in a given set, the higher $APFD$ value means the higher fault detection efficiency.

C. EXPERIMENTAL SUBJECTS

We use two experimental subjects including the commonly used benchmarks in the field of Boolean-specification testing as well as a group of Boolean expressions extracted from the control logics in nuclear industrial DCS.

Firstly, we use 20 general-form Boolean expressions, which are extracted from a well-known TCAS system [9], as experimental subjects. Figure 1 illustrates these expressions. For each expression, we generate mutants by using ten mutation operators described in Section 2.3. There is a total of 24521 mutants; 19131 of them are non-equivalent mutants and are considered faults in our experiments. These original Boolean expressions and their mutants have been widely utilized as benchmarks in the field of Boolean specification testing [19], [20].

Secondly, we use 20 general-form Boolean expressions, which are extracted from a nuclear industrial DCS, as experimental subjects. Table 1 illustrates these expressions and their application scenarios in nuclear industrial DCS. For each expression, we generate mutants by using ten mutation operators described in Section II.C. There is a total of 9381 mutants; 4912 of them are non-equivalent mutants and are considered faults in our experiments.

D. EXPERIMENTAL PROCESS

We conduct two experiments to examine fault detection efficiencies of adaptive random testing and greedy combinatorial testing in the case of limited testing resources and unlimited testing resources.

1) EXPERIMENTAL PROCESS FOR RQ1

The first experiment compares the fault detection ratio and f-measure value of adaptive random testing and combinatorial testing under the condition of limited testing resources. Limited testing resources mean that the numbers of test cases in the test sequence generated by adaptive random testing do not exceed the combinatorial test suite's sizes.

- Step 1: Generate combinatorial test suite. In this experiment, we use the well-known DDA algorithm [21] to generate combinatorial test suites. For each original Boolean expression, assuming that the number of variables is n, we randomly select 10 test cases from all the 2^n possible test cases as seed test cases. For each seed test case, the DDA algorithm can generate three combinatorial test suites with strength 2, 3, and 4, respectively. In other words, for each original Boolean expression, we have 10 different combinatorial test suites with strength 2, 10 test suites with strength 2, and 10 test suites with strength 4. The sizes of these combinatorial test suites are shown in Table 1 and 2.
- Step 2: Generate adaptive random test case sequence. For each original Boolean expression and their corresponding 30 combinatorial test suites generated in the previous step, the number of test cases in each combinatorial test suite is recorded as the upper bound of the number of test cases generated by adaptive random testing. Then we use the FSCS-ART algorithm to generate 30 different adaptive random test suites for each original

TABLE 1. Experimental subjects: 20 general-form Boolean expressions extracted from A nuclear industrial DCS.

NO.	Boolean expressions	Description
1	$\neg d\wedge b\wedge\neg f\vee\neg (e\wedge\neg a) \vee \neg c$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger the manlfunction of reactivity.
2	$\neg f\wedge\neg (d\vee\neg (\neg (a\vee c) \vee b)) \vee e$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger RCPs Trip.
3	$a\vee d\wedge (\neg f\vee\neg c) \vee \neg e\vee\neg b$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger a loss of coolant accident.
4	$b\wedge (f\wedge (e\wedge a)) \vee (d\vee\neg c)$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger a steam pipe break.
5	$(d\wedge\neg f)\wedge b\wedge (\neg (\neg b\wedge d)\wedge c)\wedge \neg (\neg e\wedge\neg d)\wedge (e\wedge f) \wedge a\vee\neg f\vee\neg e\vee e$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger the steam generator tube rupture.
6	$c\vee e\wedge\neg a\wedge (\neg h\vee\neg f) \vee\neg d\wedge\neg b\vee g$	If the control logics in the industrial seismic protection system satisfy the given condition, it will trigger the reactor protection shutdown.
7	$\neg (c\vee f) \wedge \neg e\wedge (\neg (\neg d\vee\neg e) \vee\neg e) \vee\neg b\wedge a$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger the power loss of control rod cabinet.
8	$b\wedge f\vee\neg e\vee d\wedge\neg (\neg h\vee\neg h)\wedge g\wedge\neg (a\vee\neg c)\vee \neg a\vee \neg a\vee f\wedge\neg d\vee\neg h$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will open medium pressure safety injection tank outlet valve.
9	$f\wedge c\wedge\neg (\neg c\vee d)\wedge a\vee (\neg e\vee\neg e)\vee \neg d\vee a\wedge b\wedge (\neg b\vee\neg c) \wedge \neg b\vee\neg a$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will close pressurizer spray valve.
10	$(c\vee\neg b)\wedge \neg (\neg g\vee\neg d)\vee a\vee g\wedge\neg a\vee\neg (\neg a\wedge\neg a)\wedge \neg (f\vee c) \wedge \neg h\vee (\neg e\wedge d) \wedge \neg a\wedge d$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will close steam generator main feedwater isolation valve.
11	$\neg a\wedge f\vee c\vee b\vee e\vee\neg d$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger turbine trip.
12	$\neg (\neg a\wedge d) \vee f\wedge\neg g\vee g\vee b\vee e\vee c$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger steam generator emergeney feedwater system.
13	$h\wedge \left((\neg f\vee\neg g)\wedge \neg a\right)\vee\neg g\wedge\neg a\wedge b\wedge (\neg g\wedge i)\vee e\vee i\vee\neg (\neg (\neg c\vee\neg g) \vee d)$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will start the JMN.
14	$\neg d\vee\neg f\vee (b\vee\neg c) \vee a\wedge e$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will fill the core catcher manually
15	$\neg d\wedge\neg \left(\neg e\wedge\neg (\neg h\wedge g)\right)\wedge \neg e\vee\neg \left((\neg c\vee c)\wedge b\right)\vee \neg a\vee\neg a\vee\neg i\vee\neg c\vee\neg f\wedge h\vee\neg i\vee\neg c$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will start the BRU-A of Non-accident steam generator.
16	$a\wedge\neg f\vee d\wedge (\neg e\vee\neg b) \vee\neg c$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger the steam generator tube rupture, reduce primary pressure.
17	$\neg \left((\neg a\wedge\neg g)\wedge (\neg g\vee\neg d)\right)\vee (\neg e\wedge g)\wedge f\vee \neg (d\vee\neg e) \wedge c\wedge g\vee\neg b\vee\neg h\vee f$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will cut off the main pump on the corresponding loop.
18	$(\neg c\vee a) \vee e\wedge f\wedge d\vee\neg b$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger high-pressure safety injection.
19	$\neg g\vee\neg g\vee\neg (c\vee\neg h)\wedge g\wedge\neg b\wedge (h\vee d)\vee e\vee (\neg a\vee d) \vee\neg e\vee\neg f\vee\neg b$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will close main pump seal water outlet pipeline valve.
20	$\neg a\vee\neg f\wedge\neg c\vee (\neg d\wedge b) \vee\neg (e\vee f) \vee g$	If the control logics in a nuclear industrial DCS satisfy the given condition, it will trigger reactor protection panel s-alarm.

TABLE 2. Size of the combined test set generated for TCAS by the DDA algorithm for each Boolean expression.

Number of input variables	Index of original Boolean expressions	Sizes of combinatorial test suite		
		strength = 2	strength = 3	strength = 4
5	TCAS1	6/6/7	11/12/14	16/20/24
7	TCAS2, TCAS3, TCAS4, TCAS5	7/7/8	14/15/16	30/34/40
8	TCAS6, TCAS7	8/8/8	14/16/18	30/35/40
9	TCAS8, TCAS9, TCAS10	8/8/9	16/17/18	36/38/40
10	TCAS11, TCAS12	8/8/9	16/17/19	39/42/45
11	TCAS13, TCAS14	8/8/9	18/19/21	41/43/46
12	TCAS15, TCAS16, TCAS17	9/9/9	19/20/22	43/46/49
13	TCAS18, TCAS19	9/9/10	20/21/23	47/49/51
14	TCAS20	9/9/10	21/22/23	49/51/53

* In each cell, the three numbers are the minimum, the median, and the maximum.

Boolean expression. Here, each adaptive random test case sequence is paired with a combinatorial test suite.

- Step 3: Collect the data about fault detection. For each mutant, count the proportion that such a fault is killed by the combinatorial test case sequences and the adaptive random test case sequences. Simultaneously, count how many test cases are used when the mutant is killed in a sequence of test cases, and then obtain the f-measure value.

TABLE 3. Size of the combined test set generated for DCS by the DDA algorithm for each Boolean expression.

Number of input variables	Index of original Boolean expressions	Sizes of combinatorial test suite		
		strength = 2	strength = 3	strength = 4
6	DCS1,DCS2,DCS3,DCS4,DCS5,DCS7,DCS9,DCS11,DCS14,DCS16,DCS18	7/7/7	12/13/15	26/28/30
7	DCS12,DCS20	7/7/8	14/15/16	30/34/40
8	DCS6,DCS8,DCS10,DCS17,DCS19	8/8/8	14/16/18	30/35/40
9	DCS13,DCS15	8/8/9	16/17/18	36/38/40

* In each cell, the three numbers are the minimum, the median, and the maximum.

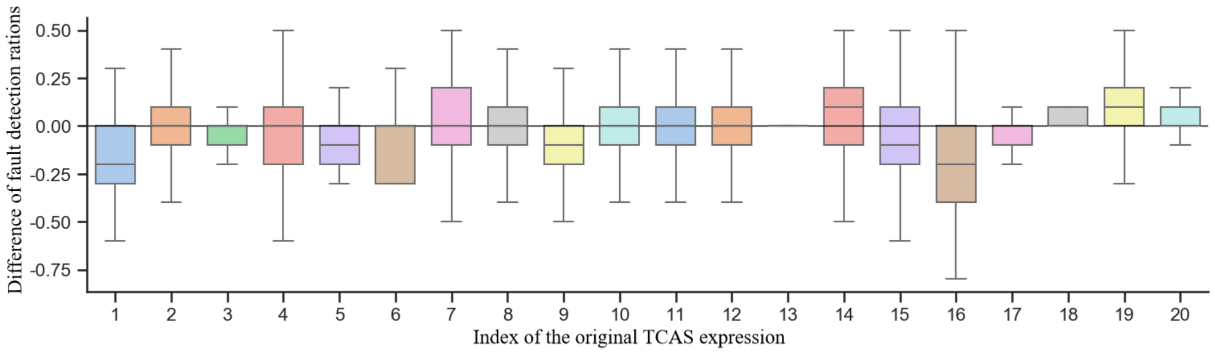

FIGURE 2. Difference of fault detection ratios for TCAS between adaptive random testing and greedy combinatorial testing with strength 2.

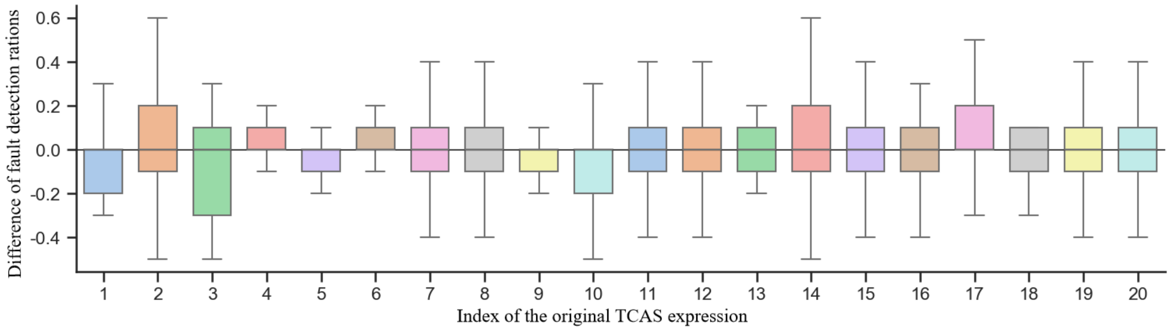

FIGURE 3. Difference of fault detection ratios for TCAS between adaptive random testing and greedy combinatorial testing with strength 3.

2) EXPERIMENTAL PROCESS FOR RQ2

The second experiment compares the f-measure value and the APFD value of the test case sequences generated by adaptive random testing and greedy combinatorial testing under the condition of unlimited testing resources. Unlimited testing resources mean that test cases should be generated indefinitely within the set of all the valid test cases until the test case sequence kills all the mutants.

- Step 1: Count the efficiency of adaptive random testing. For each original Boolean expression, we use the FSCS-ART algorithm to generate 10 different test case sequences. In each test case sequence, test cases are generated one by one until all the mutants of such an original Boolean expression are killed. Record the f-measure value of each mutant and count the APFD value in the process of detecting all the faults. According to the analysis of APFD series metrics [18], if two test case sequences have different sizes, comparing their APFD values may lead to an incorrect result. So, we use a variant version RAPFD [18] in our experiment.
- Step 2: Count the efficiency of combinatorial testing. For each original Boolean expression, we use the strength-incremental combinatorial testing strategy to generate 10 different combinatorial test case sequences. The DDA algorithm is called the strength-incremental combinatorial testing strategy. The combinatorial strength increases gradually until all mutants are killed. Record the f-measure value of each mutant and count the APFD value in the process of detecting all faults.

E. EXPERIMENTAL RESULTS

Experimental results are illustrated in Figure 2 - Figure 17. In Figure 2 - Figure 7 and Figure 10 - Figure 15, we compare the fault detection ratios and f-measure values of adaptive

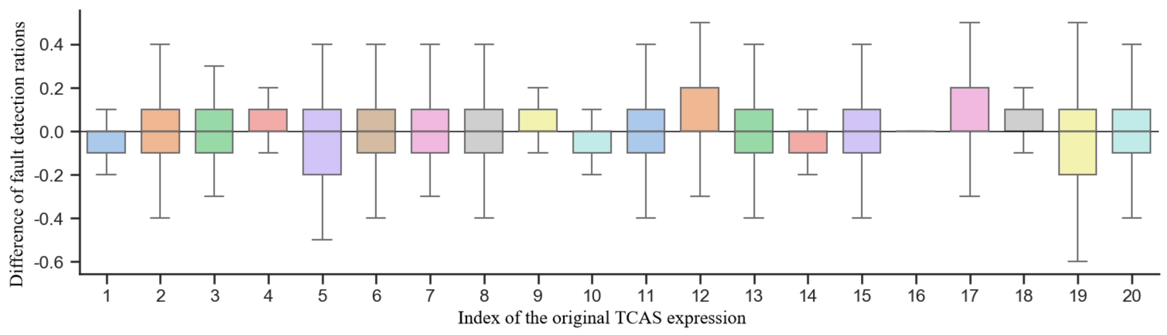

FIGURE 4. Difference of fault detection ratios for TCAS between adaptive random testing and greedy combinatorial testing with strength 4.

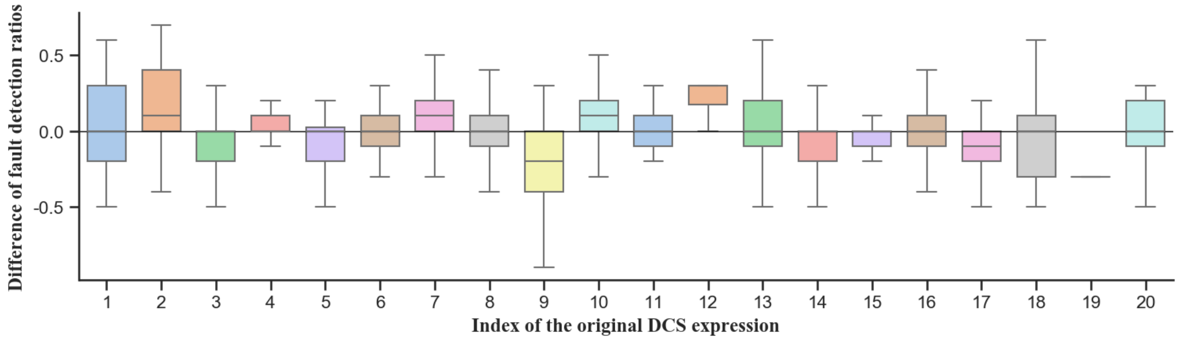

FIGURE 5. Difference of fault detection ratios for DCS between adaptive random testing and greedy combinatorial testing with strength 2.

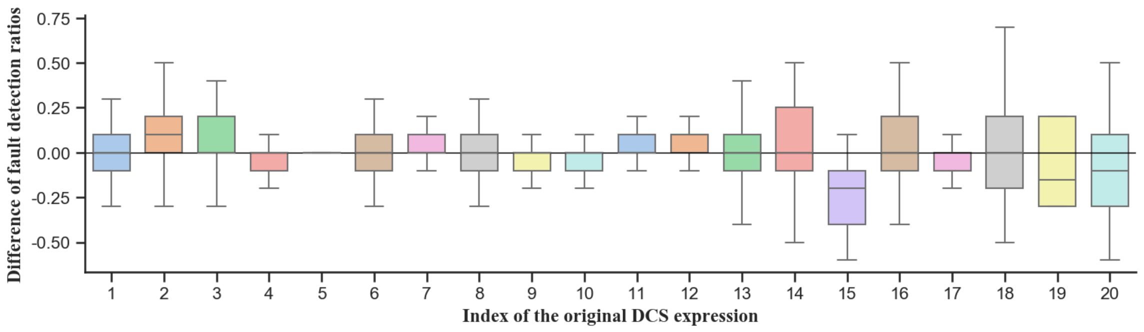

FIGURE 6. Difference of fault detection ratios for DCS between adaptive random testing and greedy combinatorial testing with strength 3.

random testing and greedy combinatorial testing under the condition of limited testing resources. In Figure 8 - Figure 9, Figure 16 - Figure 17 we compare the f-measure values and the APFD values of adaptive random testing and greedy combinatorial testing under the condition of unlimited testing resources.

1) EXPERIMENTAL RESULTS FOR RQ1

In Figure 2 - Figure 13, there are 20 box-graphs in each figure, where each box-graph represents one of the 20 original Boolean expressions extracted from the TCAS system or nuclear industrial DCS system. Figure 2 - Figure 4 show the differences between the fault detection ratios of adaptive random testing and combinatorial testing with strength 2, 3, and 4, respectively in testing the TCAS system. Figure 5 - Figure 7 show the differences between the fault detection ratios of adaptive random testing and combinatorial testing with strength 2, 3, and 4, respectively in testing nuclear industrial DCS system. The values shown in these figures are obtained by subtracting the fault detection ratio of combinatorial testing from the fault detection ratio of adaptive random testing. Figure 8 - Figure 10 show the differences

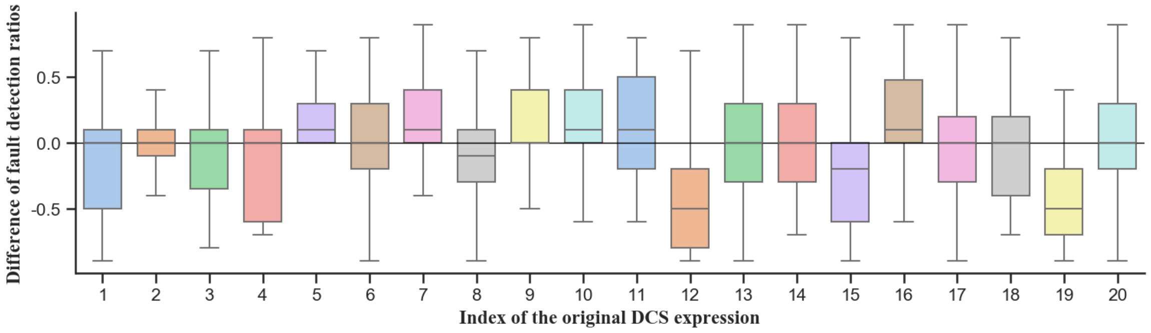

FIGURE 7. Difference of fault detection ratios for DCS between adaptive random testing and greedy combinatorial testing with strength 4.

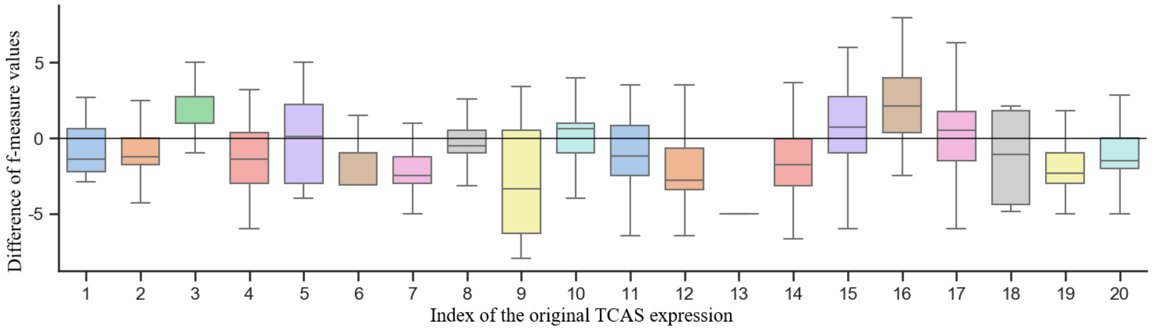

FIGURE 8. Difference of f-measure values for TCAS between adaptive random testing and greedy combinatorial testing with strength 2.

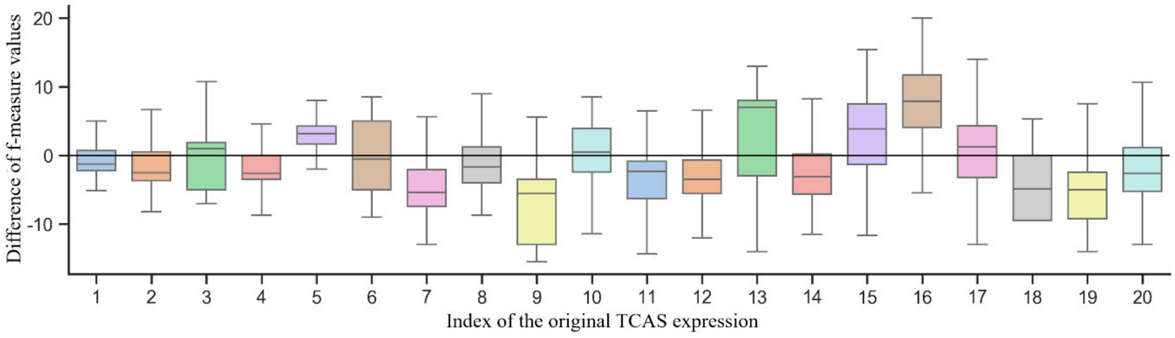

FIGURE 9. Difference of f-measure values for TCAS between adaptive random testing and greedy combinatorial testing with strength 3.

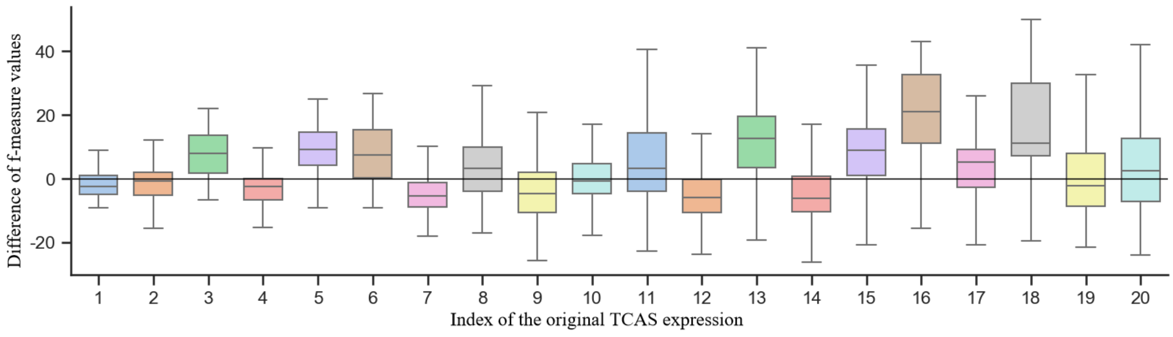

FIGURE 10. Difference of f-measure values for TCAS between adaptive random testing and greedy combinatorial testing with strength 4.

between the f-measure values of adaptive random testing and combinatorial testing with strength 2, 3, and 4, respectively in testing the TCAS system. Figure 11 - Figure 13 show the differences between the f-measure values of adaptive random

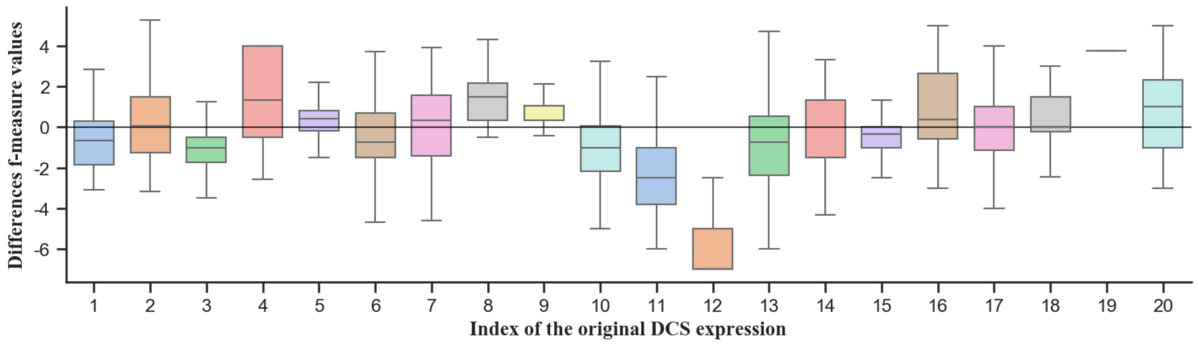

FIGURE 11. Difference of f-measure values for DCS between adaptive random testing and greedy combinatorial testing with strength 2.

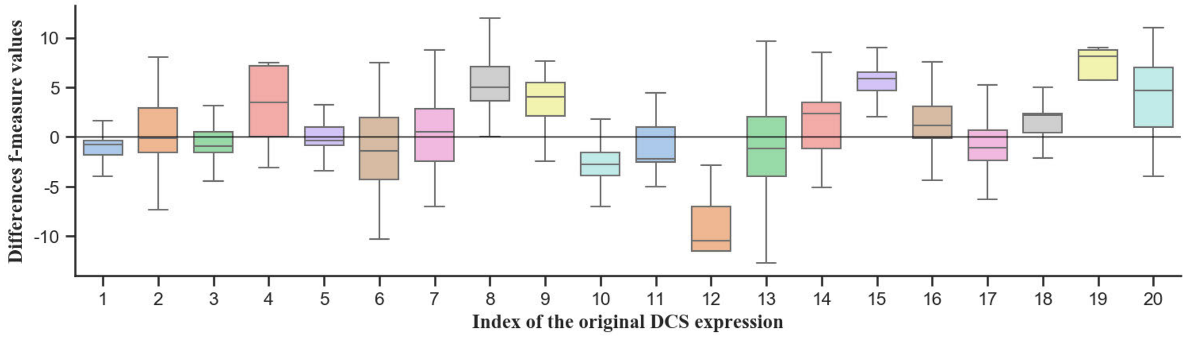

FIGURE 12. Difference of *f* -measure values for DCS between adaptive random testing and greedy combinatorial testing with strength 3.

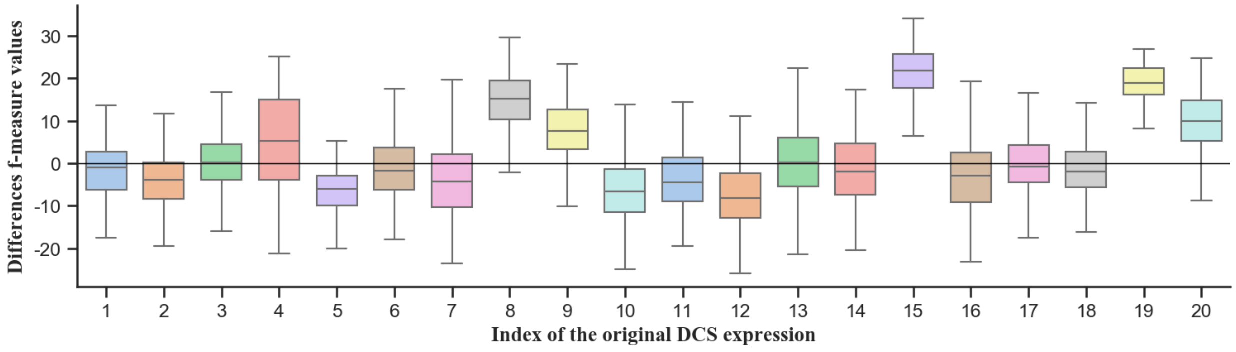

FIGURE 13. Difference of *f* -measure values for DCS between adaptive random testing and greedy combinatorial testing with strength 4.

testing and combinatorial testing with strength 2, 3, and 4, respectively in testing nuclear industrial DCS system. The values shown in these figures are obtained by subtracting the f-measure value of combinatorial testing from the *f* -measure value of adaptive random testing.

Under the condition of limited testing resources, we can answer the first question by comparing the fault detection ratios and f-measure values of adaptive random testing and greedy combinatorial testing. When the strength of combinatorial testing is low and the number of test cases is small, the fault detection efficiency of combinatorial test case sequence is close to that of the adaptive random test case sequence with the same size, though the latter has an advantage. However, when the strength of the combinatorial testing increases and the number of test cases increases gradually, the fault detection efficiency of the combinatorial test case sequence will exceed that of the adaptive random test case sequence with the same size.

2) EXPERIMENTAL RESULTS FOR RQ2

In Figure 14 - Figure 17, there are 20 box-graphs in each figure, where each box-graph represents one of the 20 original Boolean expressions extracted from the TCAS system or nuclear industrial DCS system. Figure 14 shows the differences between the f-measure values of adaptive random testing and strength-incremental combinatorial testing

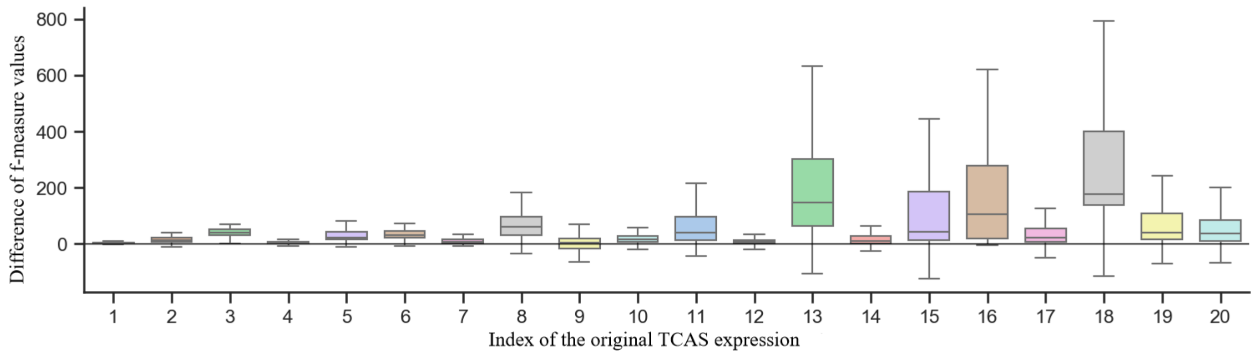

FIGURE 14. Difference of *f* -measure values for TCAS between adaptive random testing and strength-incremental combinatorial testing.

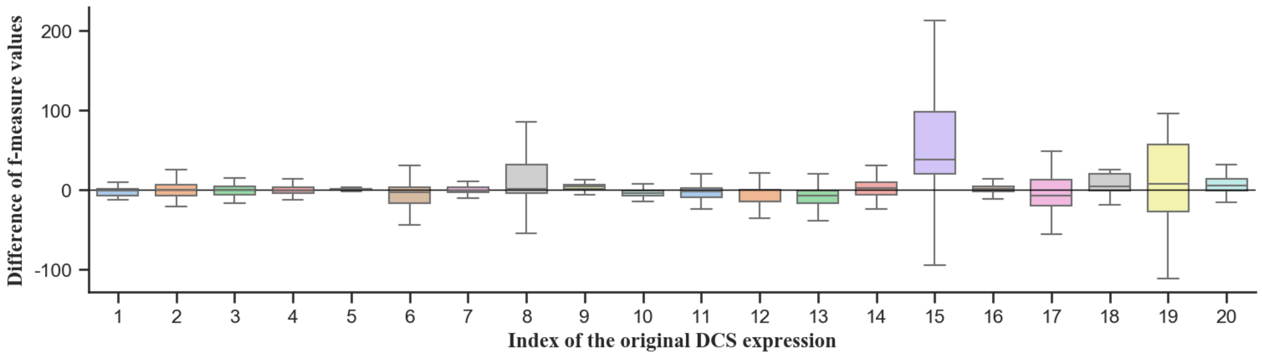

FIGURE 15. Difference of f-measure values for DCS between adaptive random testing and strength-incremental combinatorial testing.

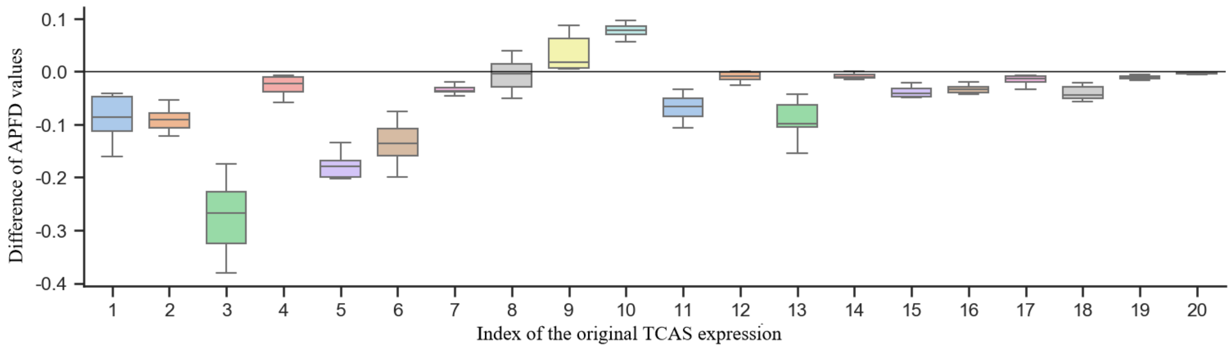

FIGURE 16. Difference of APFD values for TCAS between adaptive random testing and strength-incremental combinatorial testing.

when testing the TCAS system under the condition of unlimited testing resources. Figure 15 shows the differences between the *f* -measure values of adaptive random testing and strength-incremental combinatorial testing when testing the nuclear industrial DCS system under the condition of unlimited testing resources. The values shown in such a figure are obtained by subtracting the *f* -measure value of combinatorial testing from the *f* -measure value of adaptive random testing. Figure 16 shows the differences of the APFD values of adaptive random testing and strength-incremental combinatorial testing when testing the TCAS system. Figure 17 shows the differences of the APFD values of adaptive random testing and strength-incremental combinatorial testing when testing the nuclear industrial DCS system. The values shown in such a figure are obtained by subtracting the APFD value of combinatorial testing from the APFD value of adaptive random testing.

Under the condition of unlimited testing resources, we can answer the second question by comparing *f* -measure values and APFD values of adaptive random testing and greedy combinatorial testing. We find that the fault detection efficiency of combinatorial testing is significantly higher than that of adaptive random testing. This is because the f-measure value of combinatorial test case sequence is usually lower than that

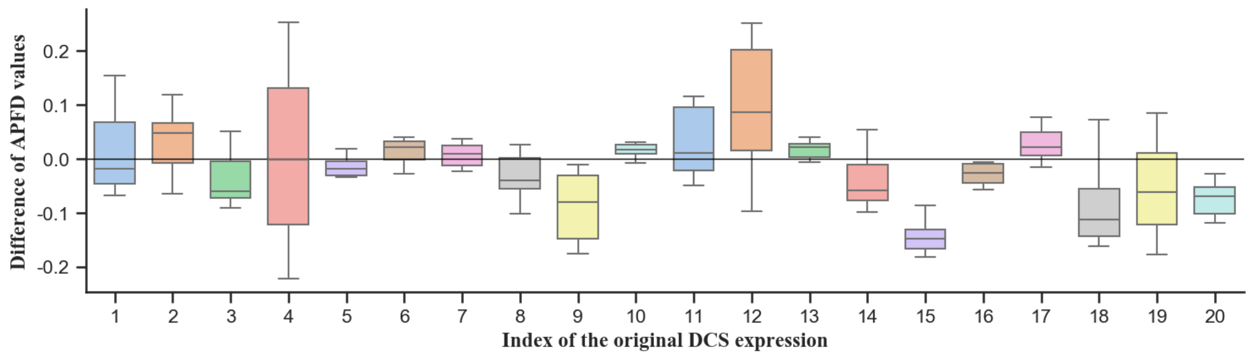

FIGURE 17. Difference of APFD values for DCS between adaptive random testing and strength-incremental combinatorial testing.

of the adaptive random test case sequence, and the APFD value of combinatorial test case sequence is usually higher than that of the adaptive random test case sequence.

F. FINDINGS AND IMPLEMENTATION

By combining the above results, we give the following suggestions about the choice of adaptive random testing technique and greedy combinatorial testing technique:

(1) When testing resources are strictly limited and only a small number of test cases have a chance to run, it is better to choose the adaptive random testing technique. This is because the fault detection abilities and the fault detection efficiencies of the two techniques are close when the number of test cases is small, while the implementation of the adaptive random testing is less difficult.

(2) When the limitation of test resources is loose and a large number of test cases can be run, the greedy combinatorial testing technique can be selected. This is because the fault detection efficiency of greedy combinatorial testing is higher when the number of test cases is massive.

(3) When the available testing resources are difficult to predict, it is impossible to decide how many test cases have the chance to run. In such a scenario, the following hybrid testing strategy could be adopted. Firstly, the adaptive random testing technique is utilized to generate a test set, in which the number of test cases is not greater than the size of the pair-wise test set. Then, these existing test cases are utilized as seed test cases to generate combinatorial test cases with higher strength until the testing resources are exhausted. Such testing strategy that combines two testing techniques can improve the fault detection efficiency of the whole testing process.

IV. THREATS TO VALIDITY

Threats to internal validity are concerned with the uncontrolled factors that may also be responsible for the results. In our experiment, fault detection ratios, f-measure values, and APFD values were collected by running combinatorial test suites and adaptive random test suites on faulty Boolean expressions. There are many combinatorial test generation algorithms, which may generate different test suites and lead to different experimental results. We select the well-known DDA algorithm, which is considered a higher-performance combinatorial test generation algorithm. Furthermore, we seed different seeding test cases to generate rich diversity combinatorial test suites. There are also many different implementations of adaptive random testing, which may generate different test suites and lead to different experimental results. We select the FSCS-ART algorithm, which is the first proposed adaptive random testing algorithm.

Threats to external validity are concerned with whether the results in our experiments are generalizable for other situations. In our experiment, experimental subjects include 20 general-form Boolean expressions extracted from the TCAS system, 19131 non-equivalent mutants generated by ten well-known mutation operators, 20 general-form Boolean expressions extracted from a real nuclear industrial DCS as well as 4912 non-equivalent mutants generated by ten well-known mutation operators. These Boolean expressions extracted from TCAS and their mutants have been widely used as benchmarks in the field of fault-based Boolean-specification testing. The nuclear industrial DCS used in our experiment is a real system that is performed in a nuclear power plant. They could represent realistic cases for comparing the greedy combinatorial testing technique and the adaptive random testing technique.

V. RELATED WORKS

Few related studies compared combinatorial testing and adaptive random testing. Nie *et al.* conducted an experiment on nine programs including FLEX, GREP, GZIP, SED, MAKE, NANOXML,DRUPAL, BUSYBOX, and LINUX. Their results suggested that the fault detection ability of adaptive random testing is comparable to combinatorial testing in 96% of scenarios [8].

However, many scholars designed experiments to compare the fault detection ability and fault detection efficiency of combinatorial testing and random testing. E.g., Kobayashi *et al.* designed an experiment on 20 Boolean expressions extracted from the TCAS system [9]. Balance *et al.* designed an experiment on 100 Boolean expressions with the same numbers of Boolean variables as the TCAS expressions [22]. Vilkomir *et al.* designed an experiment on 1000 Boolean expressions [23]. All these experiments suggest that there is a slight advantage of fault detection ability in combinatorial testing. However, all these works only considered five fault types including ASF, ENF, ORF, VNF, and VRF. Wang *et al.* designed an experiment that considered all ten fault types in Boolean-specification testing [24]. Schroeder *et al.* designed an experiment on the DMAS system and LAS system [25]. Ghandehari *et al.* designed an experiment on the Siemens program suite. Their works reported that there is no significant difference in fault detection ability between combinatorial testing and random testing [26].

VI. CONCLUSION

The adaptive random testing technique has certain similarities with the greedy combinatorial testing technique; the former requires the maximization of the Hamming distance between different test cases, while the latter requires the maximization of the difference of the tuple-combinations covered between different test cases. In order to find the optimal testing strategy for the control logics in the nuclear industrial DCS system, we designed an experiment on general-form Boolean-specifications to compare the fault detection efficiency of such two testing techniques. Experimental results indicate that: (1) If the number of test cases is relatively small, the fault detection efficiencies of adaptive random testing and greedy combinatorial testing is very close, though the former has an advantage; (2) In the gradual increase in the number of test cases, the fault detection efficiency of greedy combinatorial testing becomes gradually better; (3) If the testing resources are sufficient, the fault detection efficiency of greedy combinatorial testing has obvious advantages. Therefore, we can conclude the selection of adaptive random test and greedy combinatorial test: (1) When testing resources are strictly limited and only a small number of test cases have a chance to run, it is better to choose the adaptive random testing technique; (2) When the limitation of test resources is loose and a large number of test cases can be run, it is better to choose the greedy combinatorial testing technique.

Though significant results regarding the comparison of adaptive random testing and greedy combinatorial testing have been presented in this paper, some works are still required in the future. Combined with the results and suggestions given in this paper, in addition to Boolean specification, other related work and experiments will continue in the future.

REFERENCES

[1] C. Zong, S. Huang, E. Liu, Y. Yao, and S.-Q. Tang, "Nowhere to hide methodology: Application of clustering fault diagnosis in the nuclear power industry," *IEEE Access*, vol. 7, pp. 179864–179879, 2019.

[2] P. Bourque and R. E. Fairley, "SWEBOK: Guide to the software engineering body of knowledge (V3.0)," IEEE Comput. Soc., Washington, DC, USA, Tech. Rep., 2014.

[3] T. Y. Chen, T. H. Tse, and Y. T. Yu, "Proportional sampling strategy: A compendium and some insights," *J. Syst. Softw.*, vol. 58, no. 1, pp. 65–81, Aug. 2001.

[4] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: An approach to testing based on combinatorial design," *IEEE Trans. Softw. Eng.*, vol. 23, no. 7, pp. 437–444, Jul. 1997.

[5] R. Huang, W. Sun, Y. Xu, H. Chen, D. Towey, and X. Xia, "A survey on adaptive random testing," *IEEE Trans. Softw. Eng.*, early access, Sep. 23, 2019, doi: 10.1109/TSE.2019.2942921.

[6] H. Hu, W. E. Wong, C.-H. Jiang, and K.-Y. Cai, "A case study of the recursive least squares estimation approach to adaptive testing for software components," in *Proc. 5th Int. Conf. Qual. Softw. (QSIC)*, 2005, pp. 135–141.

[7] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Comput. Surv.*, vol. 43, no. 2, p. 11, Feb. 2011.

[8] H. Wu, C. Nie, J. Petke, Y. Jia, and M. Harman, "An empirical comparison of combinatorial testing, random testing and adaptive random testing," *IEEE Trans. Softw. Eng.*, vol. 46, no. 3, pp. 302–320, Mar. 2020, doi: 10.1109/TSE.2018.2852744.

[9] N. G. Leveson, M. P. E. Heimdahl, H. Hildreth, and J. D. Reese, "Requirements specification for process-control systems," *IEEE Trans. Softw. Eng.*, vol. 20, no. 9, pp. 684–707, Sep. 1994.

[10] Z. Chen, T. Y. Chen, and B. Xu, "A revisit of fault class hierarchies in general Boolean specifications," *ACM Trans. Softw. Eng. Methodol.*, vol. 20, no. 3, p. 13, 2011.

[11] L. Hu, W. E. Wong, D. R. Kuhn, and R. N. Kacker, "How does combinatorial testing perform in the real world: An empirical study," *Empirical Softw. Eng.*, vol. 25, no. 4, pp. 2661–2693, Jul. 2020.

[12] C. Xia, X. Wang, Y. Zhang, and H. Yang, "Using genetic algorithm to augment test data for penalty prediction," *Int. J. Performability Eng.*, vol. 16, no. 7, pp. 1078–1086, 2020.

[13] R. C. Bryce and C. J. Colbourn, "Constructing interaction test suites with greedy algorithms," in *Proc. 20th IEEE/ACM Int. Conf. Automated Softw. Eng. (ASE)*, Long Beach, CA, USA, Nov. 2005, pp. 440–443.

[14] R. C. Bryce, C. J. Colbourn, and M. B. Cohen, "A framework of greedy methods for constructing interaction test suites," in *Proc. 27th Int. Conf. Softw. Eng.*, St. Louis, MO, USA, 2005, pp. 146–155.

[15] R. C. Bryce and C. J. Colbourn, "One-test-at-a-time heuristic search for interaction test suites," in *Proc. 9th Annu. Conf. Genet. Evol. Comput. (GECCO)*, London, U.K., 2007, pp. 1082–1089.

[16] R. Huang, X. Xie, D. Towey, T. Y. Chen, Y. Lu, and J. Chen, "Prioritization of combinatorial test cases by incremental interaction coverage," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 23, no. 10, pp. 1427–1457, Dec. 2013.

[17] T. Y. Chen, F.-C. Kuo, and R. Merkel, "On the statistical properties of the F-measure," in *Proc. 4th Int. Conf. Quality Softw. (QSIC)*, Braunschweig, Germany, 2004, pp. 146–153.

[18] Z. Wang, C. Fang, L. Chen, and Z. Zhang, "A revisit of metrics for test case prioritization problems," *Int. J. Softw. Eng. Knowl. Eng.*, vol. 30, no. 08, pp. 1139–1167, Aug. 2020.

[19] E. Weyuker, T. Goradia, and A. Singh, "Automatically generating test data from a Boolean specification," *IEEE Trans. Softw. Eng.*, vol. 20, no. 5, pp. 353–363, May 1994.

[20] N. Kobayashi, T. Tsuchiya, and T. Kikuno, "Non-specification-based approaches to logic testing for software," *Inf. Softw. Technol.*, vol. 44, no. 2, pp. 113–121, Feb. 2002.

[21] R. C. Bryce and C. J. Colbourn, "A density-based greedy algorithm for higher strength covering arrays," *Softw. Test., Verification Rel.*, vol. 19, no. 1, pp. 37–53, Mar. 2009.

[22] W. A. Ballance, S. Vilkomir, and W. Jenkins, "Effectiveness of pair-wise testing for software with Boolean inputs," in *Proc. IEEE 5th Int. Conf. Softw. Test., Verification Validation (ICSWT)*, Apr. 2012, pp. 580–586.

[23] S. Vilkomir, O. Starov, and R. Bhambroo, "Evaluation of T-wise approach for testing logical expressions in software," in *Proc. IEEE 6th Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Mar. 2013, pp. 249–256.

[24] Z. Wang, Y. Li, X. Gu, X. Zheng, and M. Yu, "Fault detection capabilities of combinatorial testing and random testing for Boolean-specifications," *Int. J. Performability Eng.*, vol. 15, no. 11, pp. 2952–2961, 2019.

[25] P. J. Schroeder, P. Bolaki, and V. Gopu, "Comparing the fault detection effectiveness of N-way and random test suites," in *Proc. Int. Symp. Empirical Softw. Eng. (ISESE)*, Redondo Beach, CA, USA, 2004, pp. 49–59.

[26] L. S. Ghandehari, J. Czerwonka, Y. Lei, S. Shafiee, R. Kacker, and R. Kuhn, "An empirical comparison of combinatorial and random testing," in *Proc. IEEE 7th Int. Conf. Softw. Test., Verification Validation Workshops (ICSTW)*, Mar. 2014, pp. 68–77.

CHENG ZONG was born in Yangzhou, Jiangsu, China, in 1986. He received the B.S. degree in electric engineering and the M.S. degree in control engineering from Southeast University, Nanjing, China, in 2008 and 2014, respectively. He is currently pursuing the Ph.D. degree in software engineering with the Army Engineering University of PLA. His research interests include areas of data mining, industrial control, and failure analysis.

YANLIANG ZHANG was born in Chizhou, Anhui, China, in 1997. He received the B.S. degree in Internet of Things from Anhui Polytechnic University, in 2019. He is currently pursuing the M.S. degree in computer technology with the Nanjing University of Posts and Telecommunications. His research interests include combinatorial testing and deep neural network testing.

YONGMING YAO was born in Yangzhou, Jiangsu, China, in 1987. He received the B.S. degree in communication engineering from the Nanjing University of Posts and Telecommunications, in 2010, and the M.S. degree in computer system architecture from the Xi'an University of Posts and Telecommunications, in 2013. He is currently pursuing the Ph.D. degree in software engineering with the Army Engineering University of PLA. Since 2013, he has been an Assistant Professor with the Software Engineering Department, Tongda College, Nanjing University of Posts and Telecommunications. His research interests include crowdsourced software testing and Android permissions detection.

SHIYONG SHUANG received the B.S. degree in computer science from Information Engineering University, China, in 2005. He is currently a Senior Software Engineer with the Beijing Jinghang Research Institute of Computing, Beijing, China. His research interest includes software testing.

ZIYUAN WANG received the B.S. degree in mathematics and the Ph.D. degree in computer science from Southeast University, Nanjing, China, in 2004 and 2009, respectively. He was a Postdoctoral Researcher with the Department of Computer Science and Technology, Nanjing University, Nanjing. He is currently an Associate Professor in computer science with the School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing. His research interest includes automated software testing techniques.