Adaptive Random Test Case Generation Based on Multi-Objective Evolutionary Search

Chengying Mao[1,*] Linlin Wen[1] Tsong Yueh Chen[2]

[1] School of Software and IoT Engineering,
Jiangxi University of Finance and Economics, 330013 Nanchang, China
Email: *maochy@yeah.net*

[2] Department of Computer Science and Software Engineering,
Swinburne University of Technology, Melbourne, VIC 3122, Australia
Email: *tychen@swin.edu.au*

***Abstract*—Diversity is the key factor for test cases to detect program failures. Adaptive random testing (ART) is one of the effective methods to improve the diversity of test cases. Being an ART algorithm, the evolutionary adaptive random testing (eAR) only increases the distance between test cases to enhance its failure detection ability. This paper presents a new ART algorithm, MoesART, based on multi-objective evolutionary search. In this algorithm, in addition to the dispersion diversity, two other new diversities (or optimization objectives) are designed from the perspectives of the balance and proportionality of test cases. Then, the Pareto optimal solution returned by the NSGA-II framework is used as the next test case. In the experiments, the typical block failure pattern in the cases of two-dimensional and three-dimensional input domains is used to validate the effectiveness of the proposed MoesART algorithm. The experimental results show that MoesART exhibits better failure detection ability than both eAR and the fixed-sized-candidate-set ART (FSCS-ART), especially for the programs with three-dimensional input domain.**

***Index Terms*—software testing, adaptive random testing, multi-objective evolutionary search, test case, diversity**

I. Introduction

As a black-box testing method, *random testing* (RT) [1], [2] has been widely applied in the software industry due to its simplicity and ease of implementation. Although it has been validated as a cost-effective approach in software testing, there is still much room for improvement in its failure detection ability [3]. In theory, the test cases generated by random testing should be uniformly distributed in the whole input domain of the program under test (PUT). However, in practice, the number of test cases used in software testing is generally not too large as compared with all feasible inputs, making it difficult for these test cases to be truly evenly distributed. To improve the even-spreading of the generated test cases, Chen *et al.* [4] proposed an enhanced version of RT, namely *adaptive random testing* (ART).

In the past two decades, a series of test case generation algorithms for ART have been developed [5], [6]. Generally speaking, most of them use some heuristic techniques to make test cases as far away from each other as possible, and then disperse them as much as possible in the input domain of program under test. For example, the most popular ART algorithm named the *fixed-sized-candidate-set ART* (FSCS-ART for short) [4] employs the *max-min* criterion to select the best candidate whose distance is the farthest in all distances to the already executed test cases as the next test case. The exclusion-based ART (e.g., *restricted random testing*, RRT [7]) is also a typical algorithm, in which the first random input being outside of all exclusion regions of the executed test cases is regarded as the next test case. Meanwhile, the partitioning of a program's input domain using methods such as bisection [8], [9], random partitioning [8], [10], flexible partitioning [11] and dynamic grid partitioning [12], is another important approach to evenly distribute test cases. Furthermore, quasi-random sequences [13], [14] have also been used for ART due to its evenly-spaced characteristics and low computational cost.

In the research community of software testing, evolutionary search has been widely considered as an effective test case generation technique [15], [16], especially for white-box testing. For adaptive random testing, Tappenden and Miller [17] have presented a novel application of genetic algorithm (GA) to generate test cases. According to their experimental report, the proposed *evolutionary adaptive random* (eAR) testing algorithm can generate more evenly-distributed test cases than FSCS-ART, RRT, quasi-random testing and basic RT in some failure scenarios. In their algorithm, for each test input encoded as chromosome in GA, the minimum of the all distances from it to the already executed test cases is viewed as its fitness. Then, the input with the largest fitness is selected as the next test case. Although eAR can generate a test suite with good failure detection effectiveness, it still uses only one diversity metric, that is, the same *max-min* criterion used in FSCS-ART, to select test cases. In fact, the diversity of test cases can be achieved from many aspects [18], [19]. Inspired by the intuition of diversity [5], in this paper, we attempt to extend the search-based ART by considering the diversity of test cases from other perspectives, so that the generated test cases can be closer to the true uniform distribution.

* Corresponding author

978-1-6654-0392-4/20/$31.00 ©2020 IEEE
DOI 10.1109/TrustCom50675.2020.00020

In our work, the test case diversity is achieved from the following three aspects. The first is the "dispersion rule"which attempts to ensure that test cases are far away from each other. Here, the fitness with *max-min* criterion in eAR is adopted to enforce this diversity. The second is the "balance rule", in which the distances from each test case to its top-k nearest neighbors should be balanced. The last one is the "proportionality rule". This rule emphasizes that the ratio of the number of test cases located in a sub-region to the total number of test cases should be consistent with the area (or volume) ratio of the sub-region to the whole input domain. For the above three kinds of test case diversity criteria, three different searching objectives can be designed for selecting test cases. Thus, different from the single-objective optimization in eAR, multi-objective evolutionary search ([20], [21]) is conducted in our method. In the implementation, the well-known evolutionary search of NSGA-II [22] is applied to the above multi-objective optimization problem. In addition, the results of extensive experiments have confirmed that the proposed multi-objective evolutionary search-based ART (hereafter referred to as MoesART) shows better failure detection ability than the single-objective search-based ART, namely eAR.

II. Background

A. Adaptive Random Testing

An input which can cause the potential failures of a program is usually called a *failure-causing input*. In order to design cost-effective software testing methods, quite a few empirical studies have been conducted to investigate the behaviors of program failures and the features of failure-causing inputs [23], [24]. Based on the observations of program failures on the real-life applications, Chen *et al.* summarized the failure-causing inputs into the following three patterns [4]: *block pattern*, *strip pattern*, and *point pattern*. In general, the inputs which can trigger program failures are usually "clustering" into some contiguous regions, namely *failure-causing region*. In most cases, the failure-causing regions are usually formed into some blocks or narrow strips, that is, block pattern or strip pattern. For the point pattern, the failure-causing inputs are scattered to standalone points or some very small groups of points in the input domain. Generally speaking, once a program is finalized to form a specific version, its failure-causing region and failure rate are determined. Here, the *failure rate* [4], [25] is defined as the ratio between the number of failure-causing inputs and the number of all possible inputs of a program, and denoted as θ.

In software testing, the information about failure patterns can bring benefits to test case selection. Due to the "clustering" characteristic of failure-causing inputs, the test cases should be spread as evenly as possible if testers want to reveal the failures as early as possible. Based on this clue, Chen *et al.* proposed an enhanced random testing methodology, called adaptive random testing [4], [5]. During the process of incremental increase of test cases, ART tries to make the test cases as evenly distributed as possible in the whole input domain of a program at any time, that is, achieving diversity in spatial distribution. ART has received much attention in the field of software testing [26], and various algorithms have been developed. Among them, the most representative is candidate-based ART, that is, the fixed-sized-candidate-set ART (FSCS-ART) [4]. It is the first ART algorithm proposed and has exhibited its high effectiveness [27]. The exclusion-based algorithms represented by RRT (i.e., restricted random testing) [7] are another important kind of ART. In this algorithm, in order to avoid the test cases being close together, once a test case is selected for execution, its surrounding region is temporarily set as a forbidden region for test case selection in the next step. Motivated by the principle of proportional sampling (PS) [28], quite a few partitioning-based ARTs, such as ART-B [8], [9], ART-RP [8], and ART-FP [11], have been developed to enrich ART algorithm family.

In addition to the above heuristic strategies, the meta-heuristic based search algorithm has also been applied to generate test cases for ART. For example, Tappenden and Miller [17] adopted the well-defined evolutionary search (namely genetic algorithm, GA) to select test cases. In the search of the next test case, the algorithm seeks the test input that has the maximum distance from all previous test cases. According to the results on simulation experiments, their proposed eAR algorithm is superior to FSCS-ART, RRT, RT, and quasi-random testing for block patterns.

B. Review of eAR algorithm

In essence, software testing can be viewed as taking samples from input domain of a program, and executing these samples in turn to reveal program failures [5]. The goal of sampling is to ensure that the selected test cases are as close to the theoretically uniform distribution as possible in the input domain. Since test cases are generated one by one, it is usually required to have the next test case as far away from the already executed test cases as possible. In order to achieve this goal, Tappenden and Miller developed the evolutionary search-based algorithm (eAR) [17] to convert the sampling of the next test case to the search problem. With the global search ability of evolutionary search, it searches out the next input which is "most" far away from the existing test cases. In their work, the well-known GA is adopted as the backbone technique for the generation of ART inputs.

In the eAR algorithm, the minimum distance from a possible input to all existing test cases is regarded as the optimization objective. In a d-dimensional input domain, each possible input $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ is encoded as a chromosome in GA, where the gene x_i $(1 \leq i \leq d)$ represents the input value in the i-th dimension, and d is the dimensional number of input domain. Suppose E is the set of already executed test cases. The fitness function for the chromosome $\boldsymbol{x}$ in eAR is defined as below.

$$fitness(\boldsymbol{x}, E) = \min_{j=1}^{|E|} \text{dist}(\boldsymbol{x}, tc_j) \tag{1}$$

Here tc_j is the j-th test case in E, and $\text{dist}(\boldsymbol{x}, tc_j)$ represents the Euclidean distance between $\boldsymbol{x}$ and tc_j.

In the process of dynamic incremental generation of test cases, the first test case is generated randomly. Then, the subsequent test cases are selected by GA in the following way. Initially, the inputs in a constant population size are randomly generated and encoded as chromosomes. For each chromosome, its fitness function is measured by Equation (1). Subsequently, the classical operations, such as selection, crossover, and mutation, are performed on the chromosomes. When the evolutionary search process converges, the chromosome with the largest fitness is taken as the next test case. The above search process continues until program failure is detected or the number of test cases reaches a preset size.

The above algorithm has been validated to have better failure detection effectiveness then some basic heuristic ART algorithms. Unfortunately, it considers the diversity of test cases only from one perspective. In fact, if the diversity can be considered from multiple perspectives, the uniform distribution of test cases in the input domain can be better guaranteed.

C. Multi-Objective Evolutionary Search

In the decision-making of many problems in engineering field, it is often necessary to consider more than one objective to find an optimal solution. In this case, the traditional single-objective optimization (or search) methods are not suitable to handle them. If a user wants to make a decision according to multiple criteria, the output solution needs to simultaneously satisfy each individual objective as much as possible. For a nontrivial decision problem, the multiple objectives to be considered are often conflicting. The simplest approach to settle this matter is to convert the multi-objective optimization problem to the conventional single-objective problem. In contrast, a more popular way is to follow the technical route below. Some Pareto optimal solutions are found first, and the trade-off is quantified in satisfying the different objectives to select the most preferred one [29].

Up to present, many methods have been developed to solve this problem, and *evolutionary search* is viewed as a popular approach to generate Pareto optimal solutions [20]. Currently, most multi-objective evolutionary algorithms search for the solutions by applying Pareto-based ranking schemes. The typical frameworks include the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) [22] and Strength Pareto Evolutionary Algorithm 2 (SPEA-2) [30]. Besides genetic algorithm, other algorithms such as particle swarm optimization (PSO) [31] and simulated annealing (SA) [32] are also used to solve multi-objective optimization problems. The main advantage of these evolutionary search-based methods, when applied to optimize multi-objective problems, is that they can find the satisfied "optimal" solution in most cases.

As mentioned earlier, the diversity of test cases can be reflected from many perspectives. Therefore, the test case generation can be modeled as a multi-objective optimization problem. On this basis, the evolutionary search algorithm represented by NSGA-II is used to solve the problem.

III. Multi-Objective Evolutionary Search-Based ART

A. The Overall Solution

In our solution, the process of test case generation still follows the incremental mode of ART. As illustrated in Figure 1, the first test case is randomly generated, and then is used to execute the program under test. At the same time, it is set as the first element in test set E. For the generation of each follow-up test case, the multi-objective evolutionary search is applied for selecting the most appropriate input with respect to the three kinds of diversity perspectives. Once the "best" input is determined, it is decoded as the next test case for execution and is also added into test set E. The above procedure of selecting the next test case is repeated until the stop condition of testing is satisfied. Usually, the stop condition is that a program failure is detected or the number of test cases reaches a specified size.

During the sub-process of the next test case generation, the initial population is randomly generated according to the coding scheme of test input at first. Then, each individual (or chromosome) is evaluated from three perspectives of diversity: diversity about dispersion, diversity about balance, and diversity about proportionality. In our method, NSGA-II [22] is used for the multi-objective evolutionary search. Accordingly, the fast non-dominated sorting approach is adopted to select the partial best individuals. Besides the selection operation, the crossover and mutation operators are also applied for producing an offspring population (or sub-population). Similarly, the individuals in the offspring population can be evaluated by these three diversity metrics. Subsequently, the population in the next round can be formed by picking out the best individuals from the combined pool of the parent and offspring populations. When the stop condition of evolutionary search is met, the global individual with the greatest non-dominated rank and congestion is selected as the next test case.

B. Diversity Metrics of Test Inputs

In the search for the next test case from the input domain, the diversity measure of test inputs plays an important role. Generally speaking, the diversity of a test set can be reflected from several aspects, such as dispersion, balance, and proportionality. If a test set can show good diversities in multiple perspectives, it should be a good set for testing [18]. Suppose the already executed test case set is E, for a given candidate of test input (i.e., an individual in the population) $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$, the degree of it being suitable as the next test case can be measured by the diversity of set $E \cup \{\boldsymbol{x}\}$. That is, if $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ is added to set E to make the expanded set have good diversity, it can be regarded as an appropriate input in the next step of test execution. Here, we refer to the diversity of set $E \cup \{\boldsymbol{x}\}$ as the diversity of candidate $\boldsymbol{x}$ with respect to set E.

In the evolutionary search of test cases, each individual (chromosome) can be viewed as a candidate of the next test case. In this work, we measure the diversity of a candidate from the following three aspects.

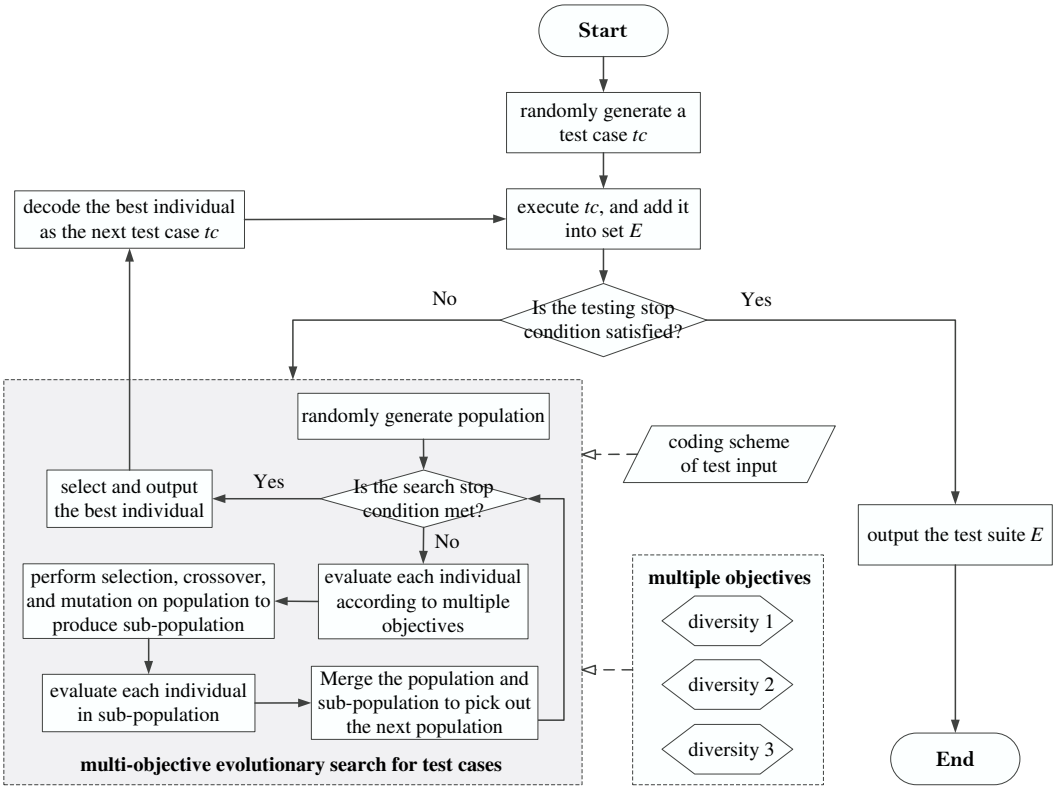

Fig. 1. The overall process of multi-objective evolutionary search-based ART

1) The diversity about dispersion: This metric attempts to spread test cases as widely as possible across the input domain. Thus, for a candidate $\boldsymbol{x}$, if it wants to have a better diversity about dispersion, it should be as far away from the existing test cases (i.e., E) as possible. To measure this diversity, as shown in Equation (1), the minimum distance from the candidate to the all executed test cases can be taken into consideration. Here, we denote this diversity as the first fitness, that is, $fit_1(\boldsymbol{x}, E)$. Obviously, the larger the fitness value is, the better the diversity that will be obtained after the candidate is added to the set E.

2) The diversity about balance: The above diversity metric can make test cases be far away from each other, but it could not ensure an even distribution of them. In fact, while selecting a test case, in addition to keeping it far away from the already executed test cases, it also needs to make the distances from it to its neighbors as balanced as possible in order to achieve a better uniform effect.

Given a candidate $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$, the top-k nearest neighbors can be picked out from the executed test set E through distance computation, here we denote the set of them as $top\text{-}k(\boldsymbol{x})$. For any test case tc_j in $top\text{-}k(\boldsymbol{x})$ $(1 \leq j \leq |top\text{-}k(\boldsymbol{x})|)$, the distance from $\boldsymbol{x}$ to it can be represented as dist($\boldsymbol{x}$, tc_j). Thus, for the candidate $\boldsymbol{x}$, its diversity about the balance of distances from it to its top-k nearest neighbors can be measured as follows.

$$fit_2(\boldsymbol{x}, E) = -\sum_{j=1}^{|top\text{-}k(\boldsymbol{x})|} p_j \cdot \log p_j, \tag{2}$$

$$p_j = \frac{\text{dist}(\boldsymbol{x}, tc_j)}{\sum_{i=1}^{|top\text{-}k(\boldsymbol{x})|} \text{dist}(\boldsymbol{x}, tc_i)} \tag{3}$$

It is not difficult to see that the larger fitness value shown in the above equations means that the candidate of test case will have better diversity.

3) The diversity about proportionality: Based on the theoretical analysis in [28], *proportional sampling strategy* is an effective and practically applicable way to improve the diversity of the selected test cases. The strategy reveals the rule that if testers want to guarantee partition testing to have a strong failure detection ability, the number of test cases selected from each partition should be proportional to the size of the partition. This rule can also be used to guide the test case selection based on evolutionary search. For example, for any sub-region in the input domain, the ratio of the number of test cases located in the sub-region to the total number of test cases should be proportional with the size ratio of the sub-region to the whole input domain.

In our work, each dimension of the input domain is divided into p parts, and the whole input domain is partitioned to p^d cells, where d is the dimensional number of the input domain.

Accordingly, for each candidate $\boldsymbol{x}$ (a.k.a. the individual or chromosome in the population of GA), it is easy to locate its relevant cell. Suppose the number of executed test cases located in the corresponding cell is n_x, the size of the cell is S_x, and the size of whole input domain is S_D. In order to achieve the proportionality, the candidate whose cell has fewer test cases will be preferred to the next use case. Based on this rule, we can define the fitness of a given candidate $\boldsymbol{x}$ with respect to the proportionality as below.

$$fit_3(\boldsymbol{x}, E) = \frac{S_x}{S_D} - \frac{n_x}{|E|} \tag{4}$$

Obviously, the greater fitness value of candidate $\boldsymbol{x}$ means that it will be selected as the next test case with higher probability. Initially, the partition number p is set to 1. When the number of the already executed test cases grows to $\tau \cdot p^d$, p is increased to $p+1$. Here, τ is set to a default value of 5.

C. Some Technical Issues

In this multi-objective evolutionary search-based ART, we focus on the test case generation for numerical programs, and the distance is measured by Euclidean distance. Thus, the coordinates of a test case in the input domain are directly encoded as the genes of a chromosome in genetic algorithm. That is, for an individual $\boldsymbol{x} = (x_1, x_2, \cdots, x_d)$ in the population of GA, x_i $(1 \leq i \leq d)$ represents the coordinate in the i-th dimension of input domain. Of course, if the inputs of a program under test require integers, they can be converted from the original individuals of GA by applying round operation.

In the preparation stage of evolutionary search, the initial population of NSGA-II is randomly generated. During the process of individual evolution, some best individuals will be preserved, while those who are low quality will be discarded. In this work, the quality of an individual is considered from three different perspectives (objectives). However, in the practice of multi-objective optimization, the best individual is not always optimal in every objective. In the framework of NSGA-II, the fast non-dominated sorting approach is used to choose the best individuals in the population for retention. Specifically, all individuals in the population are firstly sorted by iterative traversal, and each individual is assigned a non-domination level. Then, the crowding-distance of each individual is computed by considering the objective functions (i.e., fitness functions) one by one. Finally, the best individuals are chosen according to the following rules: if two individuals have different non-domination levels, the individual with the lower rank (level) is selected preferentially. Otherwise, if both individuals belong to the same level, the individual located in a lesser crowded region (or with a greater crowding-distance) is regarded as the preferred one.

Besides the operations of selection, crossover and mutation, the computation of three fitness functions for each individual is also an overhead worth paying attention to. While computing the first fitness value for individuals, the algorithm needs to calculate the distances to all executed test cases for each candidate. In the second fitness function, for a candidate, it is required to record its top-k nearest neighbors. Since the distances form the candidate to all previous test cases have been measured in the computation of the first fitness function, the information of top-k nearest neighbors can be easily obtained without any extra cost. For the third fitness function, its computation involves the partition of input domain and test case localization. In order to avoid the frequent partitioning, we adopt an intermittent dynamic partitioning strategy instead of re-partitioning for each newcomming of test case. That is, only when the number of test cases increases to τ times the number of cells will the re-partition of the input domain be started. Moreover, the partition number (p) in each dimension changes to $p+1$. Here, τ is a manually preset parameter.

In the evolutionary search, the stop condition of searching for the optimal solution is usually determined by reaching a predefined number of evolution generations. In this mode, if the search can converge to the optimal solution quickly, then many subsequent iterations will be wasted. To overcome this problem, we add another search termination condition, that is, if the optimal solution does not exceed the preset deviation ϵ in the most recent consecutive 20 generations, the search process can be stopped to output the current optimal solution.

IV. Experimental Analysis

A. Experimental Setup

In the field of random testing, the simulation experiments are usually adopted to validate the effectiveness of the related testing methods. In this study, as a common form of software failures, the block failure pattern is taken as a representative to analyze the testing effectiveness of the proposed MoesART method. Specifically, a small square (or cube) is randomly placed as the failure region in the input domain, and the ratio of the size of the small square (or cube) to that of the whole input field is the failure rate. In this experiment, we emphasize the following two cases: two-dimensional and three-dimensional input domains. In each dimension, the value range of test cases is set to -5000 to 5000.

In this work, the key research question to be answered is as follows: compared with single-objective evolutionary search, can multi-objective evolutionary search select test cases with better failure detection ability? Therefore, as the single-objective evolutionary search-based ART algorithm, the eAR algorithm [17] is used as the primary reference method for comparative experiments. In addition, FSCS-ART [4] is the first defined and most popular algorithm in ART family, so we also take it into account for comparison analysis.

When performing the effectiveness comparison between random testing methods, the following three measures are usually considered [33]: *F-measure*, *P-measure*, and *E-measure*. For the three ART algorithms considered in this paper, they all generate test cases in an incremental way. Thus, *F-measure* is taken as an evaluation metric for comparing the above three algorithms. The *F-measure* reflects the expected number of distinct test cases to detect the first program failure. In order to realize the normalization representation, the *F-measure* of an ART algorithm is usually divided by that of basic

RT, and the corresponding ratio is called *F-ratio*. Formally, $F\text{-}ratio$=F_{ART}/F_{RT}, where F_{ART} represents the *F-measure* of an ART method, and F_{RT} is the *F-measure* of basic RT.

In the experiments, we set the same configuration for the common parameters of eAR and MoesART about evolutionary search. That is, the population size was $|P|$=20, the selection operation modes in the two algorithms were ℓ-tournament (ℓ=2), the single-point crossover probability was p_c=0.6, the single-point mutation probability was p_m=0.1, and the maximum number of generations executed in evolution was set to G_{max}=100. In our MoesART, the number of the nearest neighbors of a candidate was $top\text{-}k$=5. For FSCS-ART algorithm, the size of candidate set adopted the commonly recommended value, i.e., s=10. In order to analyze the effect of each algorithm under different failure rates, repeated experiments were carried out for different failure rates, and the failure rate θ varied from 0.1 to 0.0002. In each case of failure rate, 10000 repeated experiments were performed for each algorithm to collect its *F-measures* (*F-ratios*), and the Wilcoxon rank-sum test [34] is adopted to judge the significance of the difference between the two algorithms. All three algorithms in the experiments were implemented in Java, and executed on the Eclipse platform with JDK 1.8. The experiments were conducted on a desktop PC with i7 CPU at 3.6 GHz and 8 GB RAM running under the operating system of Windows 10 64-bit.

B. Experimental Results

1) The results in the case of d=2: As the metric of failure detection effectiveness, the average *F-ratios* of FSCS-ART, eAR and MoesART for generating test cases in the 2-dimensional input domain are shown in Table I. It is easy to see that, for each failure rate in the range of 0.1 to 0.0005, the *F-ratio* of MoesART algorithm is always less than those of the other two algorithms. At the same time, the *F-ratios* of the three algorithms all decrease with the decrease of the failure rate, among which eAR algorithm showed the most obvious downward trend, while the change trends of FSCS-ART and MoesART are similar and relatively stable. Specifically, when the failure rate is equal to 0.1, the difference of *F-ratio* between the eAR and MoesART algorithms is 12.16%. However, when the failure rate decreases to 0.0005, the difference between the *F-ratios* of the two algorithms decreases significantly to 0.37%. In contrast, when θ=0.1, the difference of *F-ratio* between FSCS-ART and MoesART is 1.81%. Subsequently, the difference between them rises slightly with the decrease of θ value. When θ drops to 0.005, the difference remains at about 4%.

In addition to the comparative analysis of the mean value of the *F-ratios* for the three algorithms, the statistical test and analysis are also carried out for MoesART and the other two ART algorithms. The results are shown in Table II. When the failure rate is between 0.1 and 0.01, although MoesART has an advantage of 1.81% to 3.45% in *F-ratio* compared with FSCS-ART, the *p*-value of statistical test is far more than 0.05 in most cases. Only when θ=0.02, *p*-value is equal to 0.0292 ($<$0.05). This implies that, when the failure rate is relatively high, although MoesART has certain advantages over FSCS-ART, the confidence of the superiority is not high. However, when the failure rate is lower than 0.005, the *F-ratio* of MoesART is always lower than FSCS-ART by about 4%, and the confidence of the difference between them is relatively high, that is, the *p*-value is always lower than 0.05.

TABLE I
THE *F-ratio* RESULTS OF THE THREE ART ALGORITHMS IN 2-DIMENSIONAL INPUT DOMAIN

Failure Rates	***F-ratio* (%)**		
	FSCS-ART (***x***)	eAR (***y***)	MoesART (***z***)
0.1	84.71	95.06	**82.90**
0.05	76.53	82.39	**74.04**
0.02	70.21	71.34	**66.76**
0.01	66.79	66.87	**64.01**
0.005	66.29	63.73	**61.86**
0.002	64.18	61.25	**60.30**
0.001	64.38	59.54	**59.00**
0.0005	63.63	60.20	**59.29**
0.0002	61.88	58.59	**58.22**

TABLE II
STATISTICAL ANALYSIS ON THE *F-ratios* OF THE THREE ART ALGORITHMS IN 2-DIMENSIONAL INPUT DOMAIN

Failure Rates	**FSCS-ART (*x*) vs. MoesART (*z*)**		**eAR (*y*) vs. MoesART (*z*)**	
	***x*-*z* (%)**	*p*-value	***y*-*z* (%)**	*p*-value
0.1	1.81	0.6612	12.16	2.99E-72
0.05	2.49	0.9560	8.35	1.25E-37
0.02	3.45	0.0292	4.58	4.19E-15
0.01	2.78	0.5890	2.86	4.57E-08
0.005	4.43	0.0004	1.87	0.0018
0.002	4.50	0.0006	0.95	0.0589
0.001	5.38	1.59E-06	0.54	0.3204
0.0005	4.34	9.37E-05	0.91	0.0200
0.0002	3.66	0.0124	0.37	0.2052

While considering the difference between MoesART and eAR, MoesART shows obvious advantages over eAR in the cases of high failure rate. That is, when θ is between the range of 0.1 to 0.005, the advantage of MoesART with respect to *F-ratio* is always higher than 1.87%, and the *p*-value of Wilcoxon rank-sum test is also far less than 0.05. When the failure rate is lower than 0.002, the advantage of MoesART over eAR about *F-ratio* is less than 1%, and the *p*-value is usually higher than 0.05. Only when the failure rate is 0.0005, the *p*-value of the difference between the two algorithms is lower than 0.05.

2) The results in the case of d=3: As shown in Table III, the average *F-ratio* of our MoesART is also always less than those of the other two algorithms. Moreover, compared to the 2-dimensional case, the advantage of the MoesART algorithm is more obvious. The *F-ratios* of the three algorithms all decrease significantly with the decrease of the failure rate. Among them, the eAR algorithm has the most obvious downward trend, followed by FSCS-ART, and the decline of MoesART is relatively slow.

TABLE III
THE *F-ratio* RESULTS OF THE THREE ART ALGORITHMS IN 3-DIMENSIONAL INPUT DOMAIN

Failure Rates	***F-ratio* (%)**		
	FSCS-ART (***x***)	eAR (***y***)	MoesART (*z*)
0.1	110.34	138.10	**98.83**
0.05	98.91	121.23	**92.18**
0.02	89.16	99.00	**82.96**
0.01	82.78	89.15	**78.76**
0.005	79.95	82.78	**74.60**
0.002	75.96	77.49	**71.41**
0.001	75.01	73.15	**70.05**
0.0005	74.28	70.70	**69.06**
0.0002	72.12	69.14	**68.55**

Comparing FSCS-ART and MoesART algorithms (see Table IV), it is not difficult to see that when the failure rate is equal to 0.1, the *F-ratio* of FSCS-ART is about 11.5 percentage points higher than that of MoesART, and when the failure rate is 0.0002, the gap between them drops to about 3.5%. However, in all cases, the superiority shown by the MoesART algorithm is significant, that is, the *p*-value is always less than 0.05.

TABLE IV
STATISTICAL ANALYSIS ON THE *F-ratios* OF THE THREE ART ALGORITHMS IN 3-DIMENSIONAL INPUT DOMAIN

Failure Rates	**FSCS-ART (*x*) vs. MoesART (*z*)**		**eAR (*y*) vs. MoesART (*z*)**	
	***x*-*z* (%)**	*p*-value	***y*-*z* (%)**	*p*-value
0.1	11.51	2.40E-35	39.27	2.94E-257
0.05	6.73	6.52E-06	29.05	5.30E-203
0.02	6.20	6.86E-05	16.04	1.03E-82
0.01	4.02	0.0471	10.39	6.16E-41
0.005	5.35	3.99E-05	8.18	1.33E-28
0.002	4.55	0.0006	6.08	5.90E-18
0.001	4.96	5.51E-05	3.10	5.49E-08
0.0005	5.22	3.82E-05	1.64	0.0001
0.0002	3.57	0.0036	0.59	0.0696

The difference between the *F-ratios* of eAR and MoesART is also listed in Table IV. When the failure rate is high, MoesART has obvious superiority in *F-ratio*. For example, when θ=0.1, the F-ratio of MoesART is 39.27% lower than that of eAR. When the failure rate decreases, MoesART's advantage also gradually decreases. When θ drops to 0.0002, the difference in *F-ratio* between the two algorithms is only 0.59%. At the same time, the advantage of MoesART is statistically significant in almost all cases. Only when θ=0.0002, the *p*-value of the advantage of MoesART over eAR is slightly higher than 0.05.

C. Experimental Summary

Based on the above observations about the failure detection abilities of three ART algorithms, we can draw the following three conclusions:

- The multi-objective evolutionary search-based ART (MoesART) has better failure detection ability than the single-objective evolutionary search-based ART, namely eAR, and is also better than the most popular FSCS-ART algorithm.
- In the case of two-dimensional input domain (i.e., d=2), MoesART shows a significant advantage over eAR for the programs with high failure rate, and its advantage over FSCS-ART is significant in the situation of low failure rate.
- For the three-dimensional input domain (i.e., d=3), the failure detection effectiveness of the MoesART algorithm is significantly enhanced, and in most cases, it is significantly better than that of eAR and FSCS-ART. Especially, in the case of high failure rate, this advantage becomes more obvious.

V. CONCLUDING REMARKS

With the increasing complexity of functional modules and code structure of software systems, simple and effective testing technologies such as random testing and fuzz testing [35] have been widely used to ensure the quality of software. As an enhanced method of random testing, adaptive random testing disperses the test cases in the input domain as much as possible, so as to ensure the diversity of test cases in spatial distribution. Accordingly, it usually achieves better failure detection effectiveness than random testing. This paper focuses on the ART methods based on evolutionary search, and measures the diversity of test cases from three different perspectives. Based on the principle of dispersion, balance, and proportionality, the ART method based on multi-objective evolutionary search, namely MoesART, is proposed. In each round of test case selection, the NSGA-II framework is adopted to search for the best solution that meets the above three search objectives as the next test case.

To validate the effectiveness of the proposed MoesART method, the experiments on the typical failure pattern in two different dimensions of input domain have been performed. The experimental results confirm that MoesART shows significantly better performance over both the single-objective evolutionary search-based ART (i.e., eAR) and the popular

FSCS-ART, especially in the case of three-dimensional input domain. The above results tell us that diversifying test cases from multiple perspectives can effectively improve their corresponding failure detection ability. Although three effective diversification objectives for test cases are proposed in this paper, they can be further enriched in follow-up research.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous reviewers for their valuable comments regarding this paper. The work was supported in part by the NSFC (Grant No. 61762040), the Natural Science Foundation of the Jiangxi Province (Grant Nos. 20162BCB23036 and 20171ACB21031), and the Jiangxi Social Science Research Project (Grant No. TQ-2015-202).

REFERENCES

[1] R. Hamlet, "Random testing," in *Encyclopedia of Software Engineering*, 2nd ed., J. J. Marciniak, Ed. Chichester: John Wiley and Sons, 2002, pp. 1507–1513.

[2] R. Gerlich, R. Gerlich, and T. Boll, "Random testing: From the classical approach to a global view and full test automation," in *Proc. of the 2nd International Workshop on Random Testing: Co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07)*, November 2007, pp. 30–37.

[3] A. Arcuri, M. Z. Iqbal, and L. Briand, "Random testing: Theoretical results and practical implications," *IEEE Transactions on Software Engineering*, vol. 38, no. 2, pp. 258–277, 2012.

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

[5] T. Y. Chen, F.-C. Kuo, R. G. Merkel, and T. H. Tse, "Adaptive random testing: the ART of test case diversity," *Journal of Systems and Software*, vol. 83, no. 1, pp. 60–66, 2010.

[6] J. Chen, H. Ackah-Arthur, C. Mao, and P. K. Kudjo, "A taxonomic review of adaptive random testing: Current status, classifications, and issues," *CoRR abs/1909.10879*, pp. 1–34, 2019.

[7] K. P. Chan, T. Y. Chen, and D. Towey, "Restricted random testing: Adaptive random testing by exclusion," *International Journal of Software Engineering and Knowledge Engineering*, vol. 16, no. 4, pp. 553–584, 2006.

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

[9] C. Mao and X. Zhan, "Towards an improvement of bisection-based adaptive random testing," in *Proc. of the 24th Asia-Pacific Software Engineering Conference (APSEC'17)*, December 2017, pp. 689–694.

[10] C. Mao, "Adaptive random testing based on two-point partitioning," *Informatica*, vol. 36, no. 3, pp. 297–303, 2012.

[11] C. Mao, X. Zhan, J. Chen, J. Chen, and R. Huang, "Adaptive random testing based on flexible partitioning," *IET Software*, vol. 14, no. 5, pp. 493–505, 2020.

[12] C. Mao, T. Y. Chen, and F.-C. Kuo, "Out of sight, out of mind: A distance-aware forgetting strategy for adaptive random testing," *Science China: Information Sciences*, vol. 60, pp. 092 106:1–092 106:21, 2017.

[13] T. Y. Chen and R. G. Merkel, "Quasi-random testing," *IEEE Transactions on Reliability*, vol. 56, no. 3, pp. 562–568, 2007.

[14] H. Liu and T. Y. Chen, "Randomized quasi-random testing," *IEEE Transactions on Computers*, vol. 65, no. 6, pp. 1896–1909, 2016.

[15] M. Harman and P. McMinn, "A theoretical and empirical study of search-based testing: Local, global, and hybrid search," *IEEE Transactions on Software Engineering*, vol. 36, no. 2, pp. 226–247, 2010.

[16] M. Harman, Y. Jia, and Y. Zhang, "Achievements, open problems and challenges for search based software testing," in *Proc. of the 8th IEEE International Conference on Software Testing, Verification and Validation (ICST'15)*, April 2015, pp. 1–12.

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

[18] T. Y. Chen, "Fundamentals of test case selection: Diversity, diversity, diversity," in *Proc. of the 2nd International Conference on Software Engineering and Data Mining (SEDM'10)*, June 2010, pp. 723–724.

[19] 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.

[20] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Chichester, UK: Wiley, 2001.

[21] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: A tutorial," *Reliability Engineering and System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.

[22] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[23] M. J. P. van der Meulen, P. G. Bishop, and M. Revilla, "An exploration of software faults and failure behaviour in a large population of programs," in *Proc. of the 15th International Symposium on Software Reliability Engineering (ISSRE'04)*, November 2004, pp. 101–112.

[24] C. Schneckenburger and J. Mayer, "Towards the determination of typical failure patterns," in *Proc. of the 4th International Workshop on Software Quality Assurance (SOQUA'07)*, September 2007, pp. 90–93.

[25] W. J. Gutjahr, "Partition testing vs. random testing: The influence of uncertainty," *IEEE Transactions on Software Engineering*, vol. 25, no. 5, pp. 661–674, 1999.

[26] S. Anand, E. K. Burke, T. Y. Chen, J. Clark, M. B.Cohen, W. Grieskamp, M. Harman, M. J. Harrold, and P. McMinn, "An orchestrated survey of methodologies for automated software test case generation," *Journal of Systems and Software*, vol. 86, no. 8, pp. 1978–2001, 2013.

[27] J. Mayer and C. Schneckenburger, "An empirical analysis and comparison of random testing techniques," in *Proc. of the 2006 ACM/IEEE International Symposium on Empirical Software Engineering (ISESE'06)*, September 2006, pp. 105–114.

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

[29] J. S. Arora, *Introduction to Optimum Design*, 4th ed. London, UK: Academic Press, 2017.

[30] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm," Computer Engineering and Networks Laboratory (TIK), Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH), Zurich ETH Zentrum, Zurich, Tech. Rep. TIK-Report 103, 2001.

[31] C. A. C. Coello and M. S. Lechuga, "MOPSO: A proposal for multiple objective particle swarm optimization," in *Proc. of the 2002 Congress on Evolutionary Computation (CEC'02)*, May 2002, pp. 1051–1056.

[32] B. Suman and P. Kumar, "A survey of simulated annealing as a tool for single and multiobjective optimization," *Journal of the Operational Research Society*, vol. 57, no. 10, pp. 1143–1160, 2006.

[33] 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.

[34] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.

[35] P. Godefroid, "Fuzzing: Hack, art, and science," *Communications of the ACM*, vol. 63, no. 2, pp. 70–76, 2020.