

Pairwise test case generation with harmony search, one-parameter-at-a-time, seeding, and constraint mechanism integration

Aminu Aminu Muazu^{1,2}, Ahmad Sobri Hashim¹, Umar Danjuma Maiwada^{1,2}, Umar Audi Isma'ila¹,
Muhammad Muntasir Yakubu^{1,3}, Muhammad Abubakar Ibrahim⁴

¹Computer and Information Sciences Department, Faculty of Science and Information, Technology Universiti Teknologi PETRONAS, Perak, Malaysia

²Computer Sciences Department, Faculty of Natural and Applied Science, Umaru Musa Yar'adua University, Katsina, Nigeria

³Department of Information Technology, Faculty of Engineering and Technology, Federal University Dutsin-ma, Katsina, Nigeria

⁴Department of Computer Science, School of Primary Education Sciences, Federal Collage of Education, Katsina, Nigeria

Article Info

Article history:

Received Oct 27, 2023

Revised Feb 16, 2024

Accepted Mar 5, 2024

Keywords:

Combinatorial interaction testing

Harmony search algorithm

One-parameter-at-a-time approach

Pairwise testing

Software testing

Test case generation

ABSTRACT

Pairwise testing is a method for identifying defects through combinatorial analysis. It involves testing all possible combinations of input parameters in pairs within a system, ensuring that each pair is tested at least once. The field of test case generation is highly active in the realm of combinatorial interaction testing. Research in this area is particularly encouraged, as it falls under the category of non-deterministic polynomial-time hardness. A big challenge in this field is the combinatorial explosion problem. It is about finding the best test suite that covers all possible combinations of interaction strength. In this paper, we present the task of discovering a pairwise test set as a search problem and introduce an innovative testing tool referred to as pairwise test case generation in harmony search algorithm with seeding and constraint mechanism (PHOSC). Experimental results show that PHOSC performs better compared to some existing pairwise strategies in terms of test suite size. Additionally, PHOSC provides a comprehensive framework and serves as a research platform for the generation of pairwise test sets employing the harmony search algorithm. It adopts an approach that focuses on one parameter at a time (OPAT) and incorporates seeding and constraint mechanisms at the same time, thereby enhancing the efficiency and effectiveness of the testing process.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Aminu Aminu Muazu

Computer and Information Sciences Department, Faculty of Science and Information Technology

Universiti Teknologi PETRONAS

32610 Seri Iskandar, Perak Darul Ridzuan, Malaysia

Email: aminu.aminu@umyu.edu.ng

1. INTRODUCTION

Software engineering methods enhance reliability across numerous applications [1]. Software development companies constantly strive to attract customers in a competitive environment, aiming to enhance the quality of their products through rigorous testing [2], [3]. There are various testing methods, including category partitioning, equivalence partitioning, and domain testing which are based on the idea of dividing the input space into subsets. The idea behind partition testing is to identify and test a representative set of inputs that are likely to exhibit similar behavior. This is based on the assumption that if a particular input behaves in a certain way, other inputs within the same partition will likely behave similarly. Another

approach is random testing, where test cases are chosen randomly from an input distribution, like a uniform distribution, without using information from the specification or previous test cases.

Presently, many researchers are focused on defining an optimal strategy based on an alternative method known as combinatorial testing [4]–[6]. Combinatorial testing aims to uncover faults that result from interactions between parameter values [7]. This approach has the potential to reduce scheduled costs and time while enhancing the effectiveness of software testing across various configuration systems. One of the most prominent strategies in this regard is t-way testing, with pairwise testing being a noteworthy example [8], [9]. Combinatorial testing has seen the adoption of numerous valuable techniques that include uniformly, variably, or input-output interaction [10].

The pairwise technique in software testing is known for its effectiveness in providing test results [11]. However, there are two main approaches within this technique: one-test-at-a-time (OTAT) and one-parameter-at-a-time (OPAT) [12]. Interestingly, currently, none of these approaches are being used to support seeding or constraints within a metaheuristic method, which could improve software quality and create better test suites. Seeding support helps set boundaries for tests, ensuring they meet specific requirements, potentially leading to desired combinations in sampled test data. On the other hand, constraint support removes undesirable combinations, optimizing the test suite size by eliminating invalid combinations that do not fit the domain semantics. The harmony search as a metaheuristic draws inspiration from the idea that the primary objective of music is to seek and create a state of perfect harmony [13], [14]. This concept is borrowed from the practices of musicians who, when composing harmony, rely on their vast memory of musical pitches. Musicians continuously explore various possible combinations of these stored musical pitches. Numerous research studies have emphasized that when the harmony search algorithm is compared to other techniques, it demonstrates remarkable efficiency in tackling complex optimization problems [15].

Consequently, as the number of software inputs increases, the configurations for pairwise testing grow significantly [16], [17]. This poses challenges when there are limitations like time, cost, and resources. Testing all generated cases becomes impractical within extensive input domains, making it difficult to manage the risk of faults resulting from variable interactions. An effective approach involves dividing each domain into segments, particularly in the OPAT method. This method selects representative values from each segment to generate a smaller set of test cases. The assumption is that using representative values balances the risk of missing interactions while keeping the testing process within a reasonable budget.

In the context of search-based software engineering, which deals with optimizing processes in software engineering, many recent studies have turned to metaheuristics mechanisms to address the challenge of generating pairwise testing, commonly referred to as the combinatorial explosion problem [18]–[20]. According to [21], all pairwise strategies fall into three categories: algebraic-based, computational-based, or metaheuristic-based. The algebraic-based strategies employ mathematical functions in generating test suites, while the computational-based strategies eliminate the constraints of algebraic methods, resulting in higher costs due to considering all possible combinations of space. On the other hand, metaheuristic-based strategies use nature-inspired algorithms as the foundation for pairwise strategies.

The TConfig pairwise strategy employs a recursive construction method, utilizing orthogonal arrays to construct test suites [22]. In-parameter-order (IPO) [23] represents the inaugural pairwise strategy incorporating the one-parameter-at-a-time approach, designed specifically for systems with more than one parameter. At its core, it operates by constructing a pairwise test suite for the initial two parameters and systematically expanding it to cover the first three parameters, continuing this process until it encompasses the last parameter. Pairwise independent combinatorial testing (PICT) [24] produces all designated interaction tuples and randomly chooses their associated interaction combinations to build test cases within the comprehensive test suite. Besides facilitating variable strength, it also accommodates constraints.

In 2007, a more comprehensive iteration of the IPO strategy emerged, known as in-parameter-order-general (IPOG), which integrated both horizontal and vertical algorithms. The primary enhancement involved accommodating combinatorial t-way expansion in the combinations of parameter values to ensure an optimal test size and efficient execution time. Automatic efficient test generator (AETG) [25] stands out as one of the early strategies to implement the one-test-at-a-time method. It creates multiple test case options, choosing one strategically to cover the most uncovered tuples. Importantly, several versions of AETG have been created, such as the pairwise mAETG_SAT [26] strategy. Unlike AETG, modified automatic efficient test generator with satisfiability technique (mAETG_SAT) specifically incorporates support for constraints.

SA_SAT, a variation of simulated annealing (SA), employs a binary search algorithm to identify the most suitable test case in each iteration, adding it to the final test suite [26]. Moreover, SA is a variation of the Metropolis algorithm, where temperatures transition from higher to lower states. SA involves two key stochastic methods: one for generating solutions and another for accepting them. The iterative process of applying the SA algorithm to a discrete optimization problem includes comparing the values of the current and new solutions. Over time, SA has demonstrated efficiency in solving combinatorial optimization

problems like t-way testing [26]. It encompasses support for pairwise and variable strength interactions, addressing constraints as well.

Jenny [27] utilizes a greedy algorithm to create the interaction test suite. The process begins by forming a test suite that addresses one-way interactions. Subsequently, the suite is expanded to encompass two-way interactions and is iteratively extended until all t-way interactions are accounted for. Alsewari *et al.* introduced two strategies, pairwise harmony search algorithm-based strategy (PHSS) [28] and harmony search strategy (HSS) [29], both designed to facilitate pairwise testing. While PHSS is primarily focused on pairwise testing, HSS goes beyond supporting variable strength interactions and constraints with higher interaction strength. Notably, both strategies utilize the harmony search algorithm and employ a one-test-at-a-time approach in constructing optimal test cases.

Late acceptance hill climbing based strategy (LAHC) [30] supports pairwise and higher interaction strengths, extending up to $t \leq 4$. Categorized as a metaheuristic-based strategy, LAHC utilizes the late acceptance hill-climbing algorithm concept to generate a test suite while considering constraints. pairwise migrating birds optimization strategy (PMBOS) [31] is a strategy focused on pairwise testing, employing the migrating birds optimization concept for the generation of test cases. Pairwise choice function based hyper-heuristic (PCFHH) is a pairwise strategy that employs three criteria for selecting from four low-level heuristics, referred to as the choice function, throughout the search process [32]. The introduction of the pairwise artificial bee colony algorithm (PABC) strategy is documented in [33]. PABC is integrated into the artificial bee colony algorithm to ensure consistent interaction strength, particularly when t is set to 2.

The introduction of the bat-inspired testing strategy (BST) strategy aimed to support both pairwise testing and constraints [34]. It utilizes a one-test-at-a-time approach, creating test cases within the bat algorithm mechanism. In study [8], a novel approach is outlined for producing distinct test cases known as genetic and particle swarm optimization (GASPO). GASPO strategy leverages the genetic algorithm and particle swarm optimization, specifically tailored for succeeding pairwise testing. The pairwise gravitational search algorithm strategy (PGSAS) is designed to emphasize pairs of elements (2-way). Significantly, PGSAS stands out as the most recent strategy specifically tailored for pairwise testing [35].

While the literature acknowledges the existence and utility of pairwise strategies, the simultaneous application of seeding and constraint support, along with metaheuristic methods and a one-parameter-at-a-time approach for test case generation, has received limited attention. Consequently, we present the task of discovering a pairwise test set as a search problem and introduce an innovative testing tool called Pairwise test case generation in harmony search with OPAT, seeding, and constraint supports (PHOSC). The PHOSC provides a comprehensive framework and serves as a research platform for the generation of pairwise test sets employing the harmony search algorithm. It adopts an approach that focuses on the OPAT approach and incorporates seeding and constraint mechanisms, thereby enhancing the efficiency and effectiveness of the testing process.

The paper's structure is organized as follows: following the introduction, section 2 outlines the pairwise approach for test case generation. Section 3 describes the proposed PHOSC strategy. Section 4 is dedicated to evaluation, presenting results, and facilitating discussion. Lastly, section 5 concludes the paper, summarizing its findings and contributions in the field of test case generation and pairwise testing strategies.

2. PAIRWISE TEST CASE GENERATION

A test case comprises a collection of conditions with diverse combinations of input values, executed within a specific scenario to verify the functionality of a completed software configuration system [36], [37]. Finding the best possible final test suite to cover every possible combination of a given interaction strength is a challenge in this field, and the combinatorial explosion problem is one such problem. Consequently, test case generation stands out as the most dynamically advancing research area within combinatorial t-way testing, with the added complexity of being classified as NP-hard. In response to this challenge, researchers are increasingly focusing on the development of various pairwise strategies to tackle this issue, aiming to derive an optimal test suite solution [38]. The formulation for generating test cases in combinatorial interaction testing can be articulated as: $F = K_1 \times K_2 \times K_3 \times K_4 \times K_5 \dots \times K_n$, where F represents the total number of test cases generated, $K_1 \times K_2 \times K_3 \times K_4 \times K_5 \dots \times K_n$ represents the number of possible values for each input parameter. In this equation, you multiply the number of possible values for each parameter to calculate the total number of test cases required to cover all possible combinations of these parameters. This approach helps ensure exhaustive testing while with the combinatorial testing method, we normally minimize the total number of test cases required, which is especially valuable in situations with many parameters and possible values [8], [39].

Pairwise testing, alternatively referred to as all-pairs testing, is a method employed to test a software configuration system using combinatorial techniques [40]. This approach involves selecting combinations in a step-by-step manner to ensure that all possible pairs of parameter values are incorporated into the final test

set. For certain types of systems, pairwise testing is an effective way to find faults caused by interactions between two or more input parameters. This is especially useful for systems with a large number of possible configurations [41]. A substantial body of evidence suggests that most software failures result from unintended pairwise interactions between system parameters. As indicated in [34], one primary focus in pairwise testing is to ensure that each pairwise interaction is covered by at least one test case in the final set. As a result, combinatorial testing utilizes a compact test suite to encompass all potential parameter values and their combinations, thereby detecting potential faults in a configuration system. Consequently, t-way techniques systematically generate a concise test suite designed to maximize coverage of interaction tuples based on the specified t-way coverage criteria [34].

For further clarification, we will elaborate on a simplified configuration system of the United Bank for Africa (UBA) Mobile-App login screen, as depicted in Figure 1, to illustrate the pairwise testing scenario. Additionally, Table 1 will demonstrate the parameter values of the configuration. The configuration comprises four parameters with two values each as their symbolic value representation is depicted in Table 2. The exhaustive number of test cases required for these four parameters is calculated as 2^4 , resulting in 16 test cases as shown in Table 3. If each test case takes three minutes to complete, exhaustive testing would take nearly 48 minutes. Moreover, if each test case costs USD30, the cost of completing exhaustive testing would amount to USD480. Considering this, it is noteworthy that this is a relatively simple configuration system, how long would it take or spend to complete the exhaustive testing of a complex configuration or the entire UBA Mobile-App? For extensive industries dealing with intricate configurations involving thousands of parameters and values, addressing this issue would result in substantial expenditures, effort, manpower, resources, and time dedicated to testing their configurations.

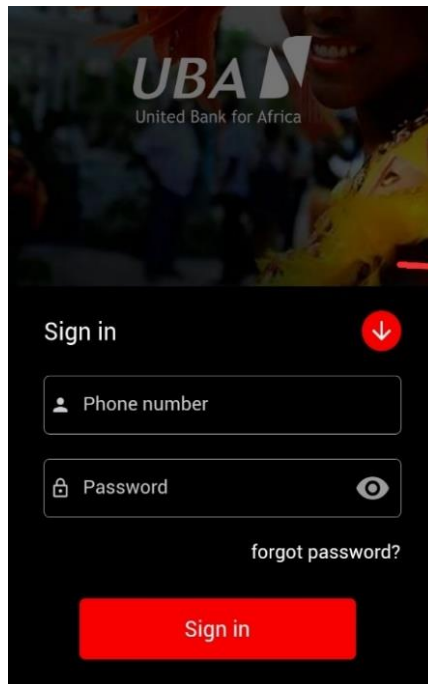


Figure 1. UBA Mobile-App login screen

Table 1. Parameter’s values for UBA Mobile-App login screen

Parameters	Phone number	Password	Forgotten password?	Sign in
Values	Enter	Enter	Click	Click
	Not enter	Not enter	Not click	Not click

Table 2. Symbolic values for UBA Mobile-App login screen

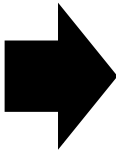
Parameters	N	P	F	S
Values	N ₁	P ₁	F ₁	S ₁
	N ₂	P ₂	F ₂	S ₂

Table 3. Exhaustive testing of the UBA Mobile-App login screen

Parameters	Test case															
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
N	N ₁	N ₁	N ₁	N ₁	N ₁	N ₁	N ₁	N ₁	N ₂	N ₂	N ₂	N ₂	N ₂	N ₂	N ₂	N ₂
P	P ₁	P ₁	P ₁	P ₁	P ₂	P ₂	P ₂	P ₂	P ₁	P ₁	P ₁	P ₁	P ₂	P ₂	P ₂	P ₂
F	F ₁	F ₁	F ₂	F ₂	F ₁	F ₁	F ₂	F ₂	F ₁	F ₁	F ₂	F ₂	F ₁	F ₁	F ₂	F ₂
S	S ₁	S ₂	S ₁	S ₂	S ₁	S ₂	S ₁	S ₂	S ₁	S ₂	S ₁	S ₂	S ₁	S ₂	S ₁	S ₂

Applying pairwise testing to this configuration can significantly reduce the total number of test cases, leading to cost and time savings to the total number of 8 test cases (50% reduction from exhaustive testing) as depicted in Table 4. The bold cells in Table 4 show the parameters with don't care values. In t-way testing, “don't care values” are specific parameter values that do not significantly impact the test outcome. Any valid value can be assigned to these parameters without affecting the test case's effectiveness. This concept allows efficient testing by focusing on critical interactions among relevant parameters while allowing flexibility for non-essential values. Based on the pairwise test applied to the UBA Mobile-App login configuration, it minimizes the number of test cases from exhaustive 16 to 8 (50%), which can lead to reduced time consumption, cost, and resources. As a result, our proposed PHOSC strategy will generate a pairwise test case that can reduce time consumption, cost, and resource utilization.

Table 4. Pairwise testing of UBA Mobile-App login screen

Parameter combinations (2-way)	Parameter				Parameter	Parameter			
	N	P	F	S		N	P	F	S
N and P	N ₁	P ₁	F ₁	S ₂		N ₁	P ₁	F ₁	S ₂
	N ₁	P ₂	F ₁	S ₁		N ₁	P ₂	F ₁	S ₁
	N ₂	P ₁	F ₁	S ₂		N ₂	P ₁	F ₁	S ₂
	N ₂	P ₂	F ₁	S ₂		N ₂	P ₂	F ₁	S ₂
N and F	N ₁	P ₁	F ₂	S ₂		N ₁	P ₂	F ₂	S ₂
	N ₁	P ₂	F ₂	S ₂		N ₂	P ₁	F ₁	S ₁
	N ₂	P ₁	F ₂	S ₂		N ₁	P ₁	F ₂	S ₂
	N ₂	P ₂	F ₂	S ₂		N ₂	P ₂	F ₂	S ₁
N and S	N ₁	P ₁	F ₁	S ₁	Test case=8				
	N ₁	P ₂	F ₁	S ₂					
	N ₂	P ₁	F ₁	S ₁					
	N ₂	P ₂	F ₁	S ₂					
P and F	N ₁	P ₁	F ₂	S ₂					
	N ₁	P ₂	F ₂	S ₁					
	N ₂	P ₁	F ₂	S ₂					
	N ₂	P ₂	F ₂	S ₁					
P and S	N ₁	P ₁	F ₁	S ₁					
	N ₁	P ₂	F ₁	S ₂					
	N ₂	P ₁	F ₁	S ₁					
	N ₂	P ₂	F ₁	S ₂					
F and S	N ₁	P ₁	F ₂	S ₁					
	N ₁	P ₂	F ₂	S ₂					
	N ₂	P ₁	F ₂	S ₁					
	N ₂	P ₂	F ₂	S ₂					

3. PHOSC TEST CASE GENERATION ENGINE

PHOSC follows a framework closely resembling the first strategy, IPO, which implements the one-parameter-at-a-time approach. However, PHOSC introduces a distinct concept for test case generation. Unlike the original IPO, PHOSC simultaneously incorporates both horizontal and vertical growth algorithms. In addition, it does so in a manner that supports both seeding and constraint mechanisms within the harmony search algorithm to enhance the efficiency and effectiveness of the testing process. The PHOSC framework is illustrated in Figure 2.

The OPAT approach is used when exhaustive testing is impractical due to time, cost, and resource constraints. To address this, input domains are divided into segments, and representative values are chosen to generate a reduced set of test cases [42]. The goal is to balance the risk of interaction among components while staying within budget constraints. In the context of supporting seeding and constraints within sampled test data, seeding support enables users to define specific boundaries that a test must adhere to. Consequently, this feature significantly enhances the software's quality. However, some combinations might not be valid based on domain rules and must be removed from the test set (constraint support). This will make the test suite size optimal [43].

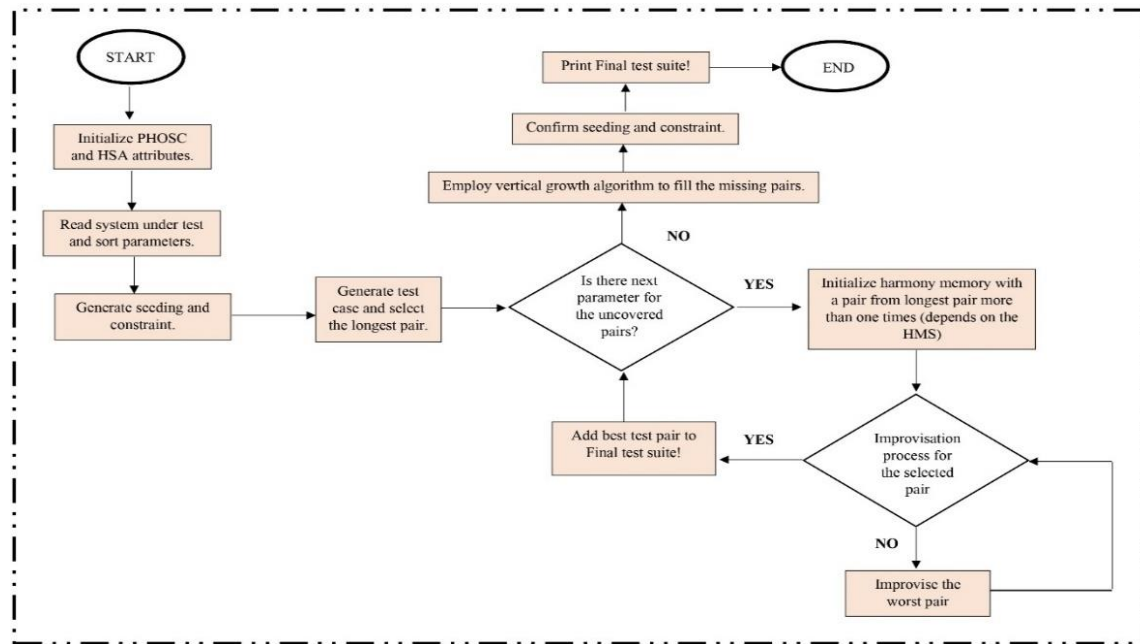


Figure 2. PHOSC framework

The harmony search algorithm comprises four stages: initializing harmony memory, creating a new solution through improvisation, updating the harmony memory, and verifying the stopping condition [13]. The concept behind the harmony search algorithm draws on the techniques used by musicians to create harmonious compositions. Musicians often have a vast repository of music pitches in their memory, and they try out various combinations of these pitches to arrive at perfect harmony. The harmony search algorithm was developed based on the observation that the ultimate objective of any music is to search for and achieve a state of perfect harmony, and it utilizes a similar approach to find optimal solutions to complex problems. A significant number of research studies have highlighted that when compared with traditional techniques, the harmony search algorithm has demonstrated its efficiency in dealing with complex optimization problems [44].

Moreover, a correlation exists between musical terminology and optimization [44]. Each musician or instrument aligns with a decision variable, and the tones produced by musical instruments correspond to decision variable values. The creation of a new harmony, formed by incorporating tones from each instrument, corresponds to the solution of an optimization problem. The aesthetic quality of harmony matches the solution's objective function value. The gradual improvement of a pleasing musical harmony corresponds to the refinement of a solution vector. As a result, PHOSC is embedded in the HSA method to produce an optimal test suite. The PHOSC pseudocode is presented in Figure 3, which depicts the integration of seeding, constraint, OPAT approach, and HSA mechanisms.

Referring to Figure 3, the PHOSC pseudocode initiates from lines 2 to 12, initializing crucial attributes for both PHOSC and the harmony search algorithm. Key attributes, such as the seeding list, constraint list, next pair, and final test set, are taken into account in PHOSC. The attributes of the seeding list and constraint list underscore the significance of seeding support, facilitating the incorporation of valuable combinations, and constraint support, aimed at eliminating undesirable ones. These functionalities contribute to the overall enhancement of software quality, and the size of the test suite will reach an optimal level. The next pair attribute is defined to accommodate pair combinations of each parameter value at each time unit until all parameter values are exhausted. Finally, the test set attribute stores all generated test cases, contributing to the comprehensive coverage and effectiveness of PHOSC's testing process. Meanwhile, for the harmony search algorithm, attributes including harmony memory size, number of iterations, improvisation count, harmony memory considering rate, and pitch adjustment rate are considered. The harmony memory and harmony memory size attribute work together, involving the setup of harmony memory and its size. Harmony memory comprises the initial pair combination of the first two parameters, serving as our solution vectors that can be added to the harmony memory. For improvisation, a new test pair is generated randomly based on two parameters: harmony memory considering rate and pitch adjustment rate. This process is designed to obtain an optimal solution, updating the harmony memory with the best solution. However, the attribute number of iterations determines the stopping condition for PHOSC.

Algorithm: PHOSC Pseudocode	Input: System under testing in 2-way strength
	Output: Final test set
<ol style="list-style-type: none"> 1. Start. 2. PHOSC attributes settings: 3. Define the Seed list 4. Define the Constraint list 5. Define the Next pair 6. Define the Final test suit 7. Harmony search algorithm attributes settings: 8. Define Harmony memory 9. Define Harmony memory size 10. Define Harmony memory considering rate 11. Define Pitch adjustment rate 12. Define the Number of iterations 13. Read the system under testing. 14. Translate actual values into integer values. 15. Create seedings and store them in a Seed list. 16. Create constraints and store them in a Constraint list. 17. Create a combination for the next pair. 18. Employ a one-parameter-at-a-time horizontal growth algorithm. 19. Initialize harmony memory size, adding each pair from the next pair through iteration to check the uncovered interaction. 20. If all interactions are covered, then add the selected pair from harmony memory for subsequent iteration. 21. Else improvise. 22. If harmony memory considering rate is less than 0.85, then do local improvisation. 23. If the pitch adjustment rate is less than 0.30, then adjust a value from the current parameter. 24. Else do global improvisation. 25. Update the harmony memory 26. Employ one-parameter-at-a-time's vertical growth algorithm to populate any missing pairs. 27. Add the next pair to the final test set 28. Check constraints from the final test set. 29. Confirm seeding from the final test set. 30. Print the final test set 31. End 	

Figure 3. PHOSC strategy's pseudocode

Subsequently, PHOSC reads the system under testing from a test file (line 13). To facilitate the selection of the next pair (line 17), the first two parameters are selected with the actual parameter values being converted to integer values at line 14. Given that the sampled test data may include specific combinations favorable or practical for the system under testing, adherence to seeding support becomes necessary. Conversely, as the sampled test data may also include specific combinations that are unwanted or not feasible for the system under testing, it is crucial to adhere to constraint support. As a result, lines 15 and 16 then create seeding and constraints data, respectively. From lines 18 to 25, our PHOSC will employ both the one-parameter-at-a-time horizontal growth algorithm and harmony search algorithm mechanism in expanding and optimizing the pairs of the generated test pair from the next pair attribute for the remaining parameter. The harmony memory (line 19) will accommodate all the generated optimized test pairs from the next pair attribute before adding them to the final test set.

The improvisation parameter plays a crucial role in a harmony search by specifying the maximum number of iterations that the algorithm undergoes in its pursuit of an optimal solution. It determines the extent to which the algorithm explores the solution space and refines its search over multiple iterations. The improvisation can either be done locally or globally. If it is discovered that not all interactions are covered (lines 21 to 23), improvisation takes place in PHOSC. In our PHOSC, there's an 85% chance of local improvisation at line 14, involving a 70% adjustment of a value from the current parameter at line 15, given that the harmony memory considering rate is less than 0.85 and the pitch adjustment rate is less than 0.30. Notably, there is a 15% chance of global improvisation (line 24). Following improvisation, PHOSC updates the harmony memory with the best solution at line 25. Moreover, PHOSC will utilize the vertical growth algorithm of one parameter at a time to populate any missing pairs from the optimized test set generated in line 26. Lastly, PHOSC will verify and validate the constraints and seeding before printing the final test set from lines 27 to 31.

4. EVALUATION AND DISCUSSION

In this section, we present the evaluation and discussion of our experimental results in three subsections. The first is benchmarking evaluation, followed by the statistical analysis of the benchmarking evaluation results. Finally, we discuss our conducted experimental observations. Noticeably the PHOSC strategy is written in Java and runs on an HP laptop with Windows 10. The laptop has 8 GB of RAM and an Intel Core i5-5200U CPU @2.20 GHz. Moreover, test suite sizes are considered in our experiments, as a result, we used the best values for each strategy's test results in all comparisons. Cells marked with an asterisk in the tables show the best test size performance, while "NA" means the result for that configuration is unavailable.

4.1. Benchmarking evaluation

Our main objective is to assess the efficiency of our PHOSC strategy by benchmarking it against established pairwise strategies. The experiments are conducted using recognized benchmarking configurations in three groups: PHOSC compared to existing pairwise strategies, PHOSC compared to existing pairwise strategies with constraint support, and PHOSC compared to existing pairwise strategies with both seeding and constraint supports. The notation for the seeding and constraint lists used in our experiments is represented as (*parameter₁ value, parameter₂ value, parameter₃ value, ..., parameter_n value*). In this representation, a value from each parameter is presented to form a complete test case, which could be either seeded or constrained. In situations where a parameter value is not specified, a default value 'x' will be provided, serving as a “don't care value”.

Firstly, in the context of the experiments conducted using recognized benchmarking configurations published in study [35], we adopted the same configurations and their respective results to compare our PHOSC against existing pairwise strategies. The benchmarking result is presented in Table 5. Secondly, in the context of the experiments conducted using recognized benchmarking configurations published in study [34], we adopted the same configurations and their respective results to compare our PHOSC against existing pairwise strategies with constraint support. The benchmarking result is presented in Table 6. Lastly, in the context of the experiments conducted using recognized benchmarking configurations published in study [34], we appended a seeding list to the adopted configurations independently to evaluate PHOSC due to the absence of pairwise strategies supporting both seeding and constraint. The benchmarking result is presented in Table 7.

Table 5. Experiment of PHOSC against existing pairwise strategies

S/N	Configuration	OTAT										OPAT		
		TConfig	Jenny	PICT	PPSTG	PHSS	PairCS	PairFS	PABC	PKS	DFA	PGSAS	IPOG	PHOSC
1	2 ⁷	7	8	7	6	NA	6	NA	NA	NA	NA	6	7	4*
2	3 ⁷	15	16	16	15	NA	15	NA	15	NA	NA	15	15	12*
3	4 ⁷	28	28	27	26	NA	25	NA	NA	NA	NA	26	29	22*
4	3 ³	10	10	10	9*	9*	9*	9*	9*	9*	9*	9*	11	9*
5	3 ⁴	10	13	13	9*	9*	9*	9*	9*	9*	9*	9*	12	9*
6	3 ⁵	14	14	13	12	NA	11	NA	NA	NA	NA	11	15	10*
7	2 ¹⁰	9	10	NA	8	NA	8	NA	NA	NA	NA	8	NA	6*
8	3 ¹⁰	17	19	18	NA	17	NA	NA	17	16	17	17	20	15*
9	3 ¹³	20	22	20	17*	18	18	18	18	20	17*	20	20	18
10	4 ¹⁰	31	30	31	NA	29	NA	28*	28*	30	30	31	31	32
11	5 ¹⁰	48	45	47	NA	45	NA	42*	43	46	45	48	50	52

Table 6. Experiment of PHOSC against existing pairwise strategies in the presence of constraint support

S/N	Configuration	Constraint List	SA_SAT	mAETG_SAT	PICT	TestCover	LAHC	HSS	BTS	PHOSC
1	3 ³	[(x,1,0), (2,2,x), (x,2,0), (2,x,2), (1,2,2), (0,x,1)]	10	10	10	10	10	10	10	6*
2	4 ³	[(2,x,3), (2,1,x), (0,1,x), (3,3,0)]	17	17	19	17	17	16	16	13*
3	5 ³	[(4,x,2), (4,2,x), (4,x,4), (1,3,x), (1,1,x), (4,3,1)]	26	26	27	30	25	26	25	24*
4	6 ³	[(3,x,1), (2,0,x), (x,3,4), (x,3,1), (3,5,x), (5,4,4), (x,1,2)]	36	37	39	38	36	36	36	34*
5	7 ³	[(1,4,x), (4,x,0), (x,0,5), (6,4,x), (6,3,x), (5,5,3)]	52	52	56	54	53	51	51	48*

Table 7. Experiment of PHOSC pairwise in the presence of seeding and constraint support

S/N	Configuration	Seeding List	Constraint List	PHOSC
1	3 ³	[(1,0,2), (1,2,2)]	[(x,1,0), (2,2,x), (x,2,0), (2,x,2), (1,2,2), (0,x,1)]	7
2	4 ³	[(0,0,0)]	[(2,x,3), (2,1,x), (0,1,x), (3,3,0)]	13
3	5 ³	[(4,4,4), (1,1,2), (0,2,4)]	[(4,x,2), (4,2,x), (4,x,4), (1,3,x), (1,1,x), (4,3,1)]	25
4	6 ³	[(3,3,2)]	[(3,x,1), (2,0,x), (x,3,4), (x,3,1), (3,5,x), (5,4,4), (x,1,2)]	35
5	7 ³	[(0,0,0), (6,6,6)]	[(1,4,x), (4,x,0), (x,0,5), (6,4,x), (6,3,x), (5,5,3)]	48

4.2. Statistical analysis

For the statistical analysis, at a confidence level of 95%, we conducted two statistical analyses on the gathered results to ascertain any significant difference between the PHOSC strategy and other established pairwise t-way strategies. These analyses involve the Wilcoxon signed-rank test and interval plots illustrating the distribution of mean sizes. The Wilcoxon signed-rank test provides a decision outcome. If the asymptotic significance value is at or below 0.05, it leads to rejecting the null hypothesis (H_0), signifying a meaningful distinction between the strategies. Conversely, if the asymptotic significance value surpasses 0.05, it results in retaining the null hypothesis (H_0), indicating no significant difference between the strategies. The decision regarding the null hypothesis is based on the comparison of the results between the two strategies. Additionally, the Wilcoxon signed-rank test provides a Rank outcome that can either be positive, negative, or ties between the two compared strategies. The experiment of non-parametric Wilcoxon signed rank test statistical analysis for the results in Table 5 and Table 6 is presented in Table 8. Moreover, the Interval Plot for experiments in Table 5 and Table 6 results with 95% CL of the mean is presented in Figures 4 and 5 respectively.

Table 8. Experiment of non-parametric Wilcoxon signed rank test statistical analysis.

Wilcoxon Test	Strategy Pair	Ranks			Asymptotic Significant (2-tailed) statistical test	Decision
		Positive	Negative	Ties		
Results of Table 5	PHOSC-TConfig	9	2	0	0.055	Null hypothesis (H_0) retained
	PHOSC-Jenny	9	2	0	0.020	Null hypothesis (H_0) rejected
	PHOSC-PGSAS	7	2	2	0.134	Null hypothesis (H_0) retained
Results of Table 6	PHOSC-SA_SAT	5	0	0	0.038	Null hypothesis (H_0) rejected
	PHOSC-mAETG_SAT	5	0	0	0.039	Null hypothesis (H_0) rejected
	PHOSC-PICT	5	0	0	0.043	Null hypothesis (H_0) rejected
	PHOSC-TestCover	5	0	0	0.038	Null hypothesis (H_0) rejected
	PHOSC-LAHC	5	0	0	0.042	Null hypothesis (H_0) rejected
	PHOSC-HSS	5	0	0	0.041	Null hypothesis (H_0) rejected
	PHOSC-BTS	5	0	0	0.042	Null hypothesis (H_0) rejected

4.3. Experimental observation and discussion

In this section, we will provide a broad overview and discussion of the experimental results. The analysis encompasses both observational insights and statistical findings derived from Table 5, Table 6, Table 7, Table 8, Figure 4, and Figure 5. We will start with a general observation of the benchmarking experiments. Following that, we will delve into the statistical analysis, which includes the Wilcoxon signed-rank test and the description of the result distributions in the Interval Plot.

In reference to Table 5, it is worth mentioning that our PHOSC demonstrated superior performance in nearly all cases, except for specific configurations 3^{13} , 4^{10} , and 5^{10} . In the 3^{13} scenarios, PPSTG and DFA outperformed all, while for 4^{10} , PairFS and PABC showed better results. In the case of 5^{10} , only PairFS surpassed all other strategies. However, most strategies produced the same results for configurations 3^3 and 3^4 . Referring to Table 6, it is evident that our PHOSC exhibited exceptional performance in all instances when the five configurations were executed. Concerning Table 7, the experiment autonomously assessed how well our PHOSC could accommodate seeding and constraint. Regrettably, no pairwise strategies currently exist that support both seeding and constraint.

As per the findings presented in Table 8, the null hypothesis is retained only in two instances when combined with PGSAS and TConfig concerning the results from Table 5. This implies that there is no substantial variation in the size of the produced test suite. On the contrary, when paired with other strategies, the null hypothesis is rejected, indicating a substantial variation. Additionally, in the majority of cases involving Wilcoxon ranking, the positive ranks of our PHOSC outweigh the negative ranks. These results collectively indicate that PHOSC statistically has a better test suite size compared to other pairwise strategies. Based on the results in Figure 4, our PHOSC exhibits the lowest mean value at 17.1818, followed by PGSAS at 18.1818, then TConfig at 19, with Jenny being the least favorable at 19.5455. It is important to note that the results for PICT, PPSTG, PHSS, PairCS, PairFS, PABC, PKS, DFA, and IPOG are disregarded due to incomplete samples. Just like in Figure 4, Figure 5 shows that our PHOSC has the lowest mean value of 25. BTS follows closely with 27.6, then HSS at 27.8. Both LAHC and SAT_SAT score 28.2, while mAETG_SAT comes in at 28.4, TestCover at 29.8, and PICT with the least favorable result at 30.2.

Current pairwise strategies are effective, but it is essential to discuss the PHOSC strategy, which uniquely combines support for seeding, constraints, the OPAT approach, and a metaheuristic method. For example, while TTG supports seeding and constraints, it lacks pairwise and metaheuristic approaches. On the other hand, strategies like LAHC, HSS, and BST are metaheuristic-based but only support constraints. In contrast, PHOSC integrates OPAT, seeding, constraints, and a metaheuristic method simultaneously, aiming

to enhance software quality and reduce the test suite size. Experiments comparing PHOSC with existing pairwise strategies consistently showed PHOSC outperforming others in most cases. However, some strategies like DFA, PABC, PairFS, and PPSTG performed better in three configurations: 3^{13} , 4^{10} , and 5^{10} as depicted in Table 5.

In summary, software testers need to decide the number of test cases based on context, expertise, and judgment, especially considering OPAT, seeding, constraint, and metaheuristic support. The OPAT approach divides input domains, choosing representative values for a reduced set of test cases. Seeding and constraint support allow flexibility in including essential and excluding unnecessary test cases, respectively. With metaheuristic-based support, an optimized final test list is achieved.

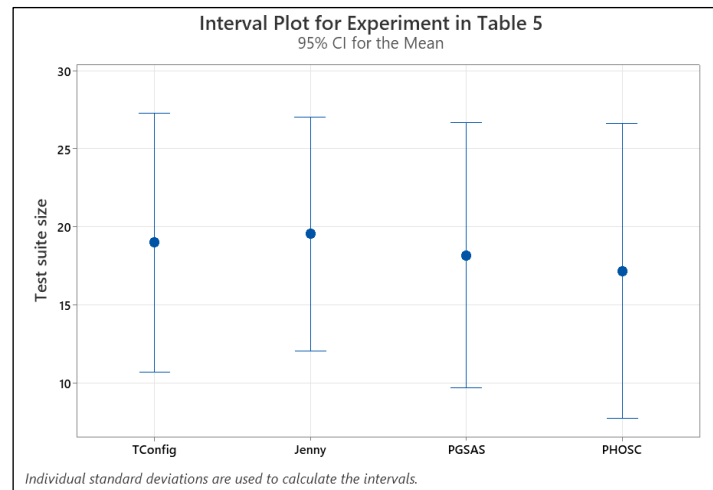


Figure 4. The interval Plot for experiments in Table 5 results in 95% CL of the mean

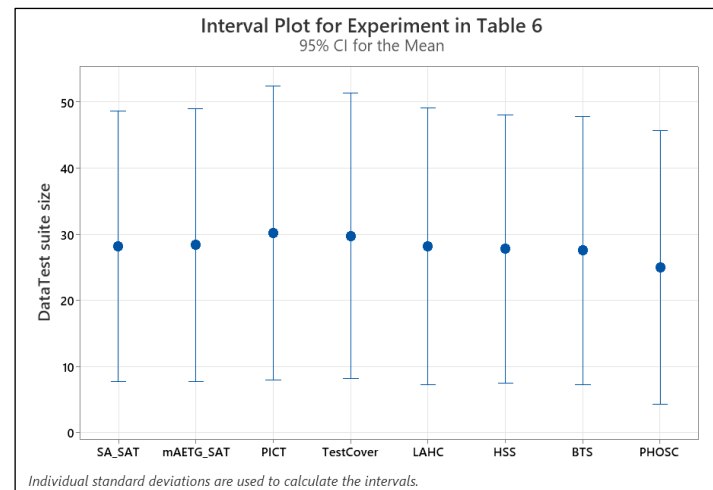


Figure 5. The interval Plot for experiments in Table 6 results in 95% CL of the mean

5. CONCLUSION

In this research, we have discussed and assessed a novel pairwise strategy known as the PHOSC strategy. PHOSC is designed to facilitate both seeding and constraint support within the harmony search algorithm, utilizing a one-parameter-at-a-time approach for test case generation. Our benchmarking results have been promising across almost all benchmarking configurations. Despite achieving research goals related to OPAT, seeding, constraints, and metaheuristic-based support, there are still opportunities for improvement. Our ongoing work involves further enhancing PHOSC to support a t-way with a higher level of interaction strength.

ACKNOWLEDGEMENTS

The authors express gratitude for the support of this research provided by the Ministry of Higher Education (MoHE) Malaysia under the Fundamental Research Grant Scheme FRGS/1/2023/ICT01/UTP/02/2.




REFERENCES

- [1] M. S. Croock, Z. A. Hassan, and S. D. Khuder, "Adaptive key generation algorithm based on software engineering methodology," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 11, no. 1, pp. 589–595, Feb. 2021, doi: 10.11591/ijece.v11i1.pp589-595.
- [2] H. Asfa and T. J. Gandomani, "Software quality model based on development team characteristics," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 1, pp. 859–871, Feb. 2023, doi: 10.11591/ijece.v13i1.pp859-871.
- [3] M. M. Rosli and N. S. M. Yusop, "Evaluating the effectiveness of data quality framework in software engineering," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 6, pp. 6410–6422, Dec. 2022, doi: 10.11591/ijece.v12i6.pp6410-6422.
- [4] A. A. Muazu, A. S. Hashim, A. Sarlan, and M. Abdullahi, "SCIOG: seeding and constraint support in IPOG strategy for combinatorial t-way testing to generate optimum test cases," *Journal of King Saud University - Computer and Information Sciences*, vol. 35, no. 1, pp. 185–201, Jan. 2023, doi: 10.1016/j.jksuci.2022.11.010.
- [5] A. B. Nasser, K. Z. Zamli, N. W. B. M. Nasir, W. A. H. M. Ghanem, and F. Din, "T-way test suite generation based on hybrid flower pollination algorithm and hill climbing," in *ACM International Conference Proceeding Series*, Feb. 2021, pp. 244–250, doi: 10.1145/3457784.3457822.
- [6] J. Chen, J. Chen, S. Cai, H. Chen, C. Zhang, and C. Huang, "A test case generation method of combinatorial testing based on T-way testing with adaptive random testing," in *Proceedings - 2021 IEEE International Symposium on Software Reliability Engineering Workshops, ISSREW 2021*, Oct. 2021, pp. 83–90, doi: 10.1109/ISSREW53611.2021.00048.
- [7] A. A. Muazu, A. S. Hashim, and U. Maiwada, "Enhanced version of seeding and constraint support in IPOG strategy for variable strength interaction T-way testing enhanced version of seeding and constraint support in IPOG strategy for variable strength interaction T-way testing," *Malaysian Journal of Computer Science*, vol. 36, 2023.
- [8] M. Lakshmi Prasad, A. Raja Sekhar Reddy, and J. K. R. Sastry, "GAPSO: optimal test set generator for pairwise testing," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6, pp. 2346–2350, Aug. 2019, doi: 10.35940/ijeat.F8645.088619.
- [9] A. A. Hassan, S. Abdullah, K. Z. Zamli, and R. Razali, "Whale optimization algorithm strategies for higher interaction strength T-way testing," *Computers, Materials and Continua*, vol. 73, no. 1, pp. 2057–2077, 2022, doi: 10.32604/cmc.2022.026310.
- [10] Y. A. Alsariera, Y. Sanjalawe, A. H. Al Omari, M. A. Albawaleez, Y. K. Sanjalawe, and K. Z. Zamli, "Hybridized BA and PSO t-way algorithm for test case generation cloud computing security view project detection DDoS attack approaches against SDN view project hybridized BA and PSO T-way algorithm for test case generation," *International Journal of Computer Science and Network Security*, vol. 21, no. 10, 2021, doi: 10.22937/IJCSNS.2021.21.10.48.
- [11] F. Din and K. Z. Zamli, "Pairwise test suite generation using adaptive teaching learning-based optimization algorithm with remedial operator," in *Advances in Intelligent Systems and Computing*, vol. 843, Springer International Publishing, 2019, pp. 187–195.
- [12] A. K. Alazzawi *et al.*, "Recent T-way test generation strategies based on optimization algorithms: an orchestrated survey," in *Lecture Notes in Electrical Engineering*, vol. 758, Springer Nature Singapore, 2022, pp. 1055–1060.
- [13] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, Feb. 2001, doi: 10.1177/003754970107600201.
- [14] A. Al-Shaikh, B. A. Mahafzah, and M. Alshraideh, "Hybrid harmony search algorithm for social network contact tracing of COVID-19," *Soft Computing*, vol. 27, no. 6, pp. 3343–3365, Jun. 2023, doi: 10.1007/s00500-021-05948-2.
- [15] A. A. Muazu, A. S. Hashim, and A. Sarlan, "Review of nature inspired metaheuristic algorithm selection for combinatorial T-way testing," *IEEE Access*, vol. 10, pp. 27404–27431, 2022, doi: 10.1109/ACCESS.2022.3157400.
- [16] L. Abualigah, A. Diabat, and Z. W. Geem, "A comprehensive survey of the harmony search algorithm in clustering applications," *Applied Sciences*, vol. 10, no. 11, May 2020, doi: 10.3390/app10113827.
- [17] J. Kang, S. Kwon, D. Ryu, and J. Baik, "HASPO: harmony search-based parameter optimization for just-in-time software defect prediction in maritime software," *Applied Sciences*, vol. 11, no. 5, pp. 1–25, Feb. 2021, doi: 10.3390/app11052002.
- [18] J. B. Odili, A. B. Nasser, A. Noraziah, M. H. A. Wahab, and M. Ahmed, "African buffalo optimization algorithm based T-way test suite generation strategy for electronic-payment transactions," in *Lecture Notes in Networks and Systems*, vol. 299, Springer International Publishing, 2022, pp. 160–174.
- [19] H. M. Fadhil, M. N. Abdullah, and M. I. Younis, "Innovations in T-way test creation based on a hybrid hill climbing-greedy algorithm," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 12, no. 2, pp. 794–805, Jun. 2023, doi: 10.11591/ijai.v12.i2.pp794-805.
- [20] E. Pira, V. Rafe, and S. Esfandyari, "A three-phase approach to improve the functionality of T-way strategy," *Soft Computing*, vol. 28, no. 1, pp. 415–435, Apr. 2024, doi: 10.1007/s00500-023-08199-5.
- [21] H. M. Fadhil, M. N. Abdullah, and M. I. Younis, "Combinatorial testing approaches: a systematic review," *Iraqi Journal of Computer, Communication, Control and System Engineering*, pp. 60–79, Dec. 2022, doi: 10.33103/uot.ijccce.22.4.6.
- [22] A. W. Williams, "Software component interaction testing: coverage measurement and generation of configurations," University of Ottawa, 2002.
- [23] K. C. Tai and Y. Lei, "A test generation strategy for pairwise testing," *IEEE Transactions on Software Engineering*, vol. 28, no. 1, pp. 109–111, 2002, doi: 10.1109/32.979992.
- [24] J. Czerwonka, "Pairwise testing in the real world: practical extensions to test-case scenarios," in *Proceedings of 24th Pacific Northwest Software Quality Conference*, 2008, pp. 419–430.
- [25] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: an approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, no. 7, pp. 437–444, Jul. 1997, doi: 10.1109/32.605761.
- [26] M. B. Cohen, M. B. Dwyer, and J. Shi, "Interaction testing of highly-configurable systems in the presence of constraints," in *2007 ACM International Symposium on Software Testing and Analysis*, Jul. 2007, pp. 129–139, doi: 10.1145/1273463.1273482.
- [27] B. Jenkins, "jenny," *Jenny Test Tool*, 2005. <http://www.burtleburtle.net/bob/math/jenny.html> (accessed Sep. 01, 2023).
- [28] A. R. A. Alsewari, "A harmony search based pairwise sampling strategy for combinatorial testing," *International Journal of the Physical Sciences*, vol. 7, no. 7, Feb. 2012, doi: 10.5897/ijps11.1633.
- [29] A. R. A. Alsewari and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing


- strategy with constraints support,” *Information and Software Technology*, vol. 54, no. 6, pp. 553–568, Jun. 2012, doi: 10.1016/j.infsof.2012.01.002.
- [30] A. R. Alsewari, K. Z. Zamli, and B. Al-Kazemi, “Generating t-way test suite in the presence of constraints,” *Journal of Engineering and Technology*, vol. 6, no. 2, pp. 2180–3811, 2015.
- [31] H. L. Zakaria and K. Z. Zamli, “Migrating birds optimization based strategies for pairwise testing,” in *2015 9th Malaysian Software Engineering Conference*, Dec. 2016, pp. 19–24, doi: 10.1109/MySEC.2015.7475189.
- [32] F. Din, A. R. A. Alsewari, and K. Z. Zamli, “A parameter free choice function based hyper-heuristic strategy for pairwise test generation,” in *Proceedings - 2017 IEEE International Conference on Software Quality, Reliability and Security Companion, QRS-C 2017*, Jul. 2017, pp. 85–91, doi: 10.1109/QRS-C.2017.22.
- [33] A. K. Alazzawi, A. A. Ba Homaid, A. A. Alomoush, and A. R. A. Alsewari, “Artificial bee colony algorithm for pairwise test generation,” *Journal of Telecommunication, Electronic and Computer Engineering*, vol. 9, no. 1–2, pp. 103–108, 2017.
- [34] Y. A. Alsariera, H. A. S. Ahmed, H. S. Alamri, M. A. Majid, and K. Z. Zamli, “A bat-inspired testing strategy for generating constraints pairwise test suite,” *Advanced Science Letters*, vol. 24, no. 10, pp. 7245–7250, Oct. 2018, doi: 10.1166/asl.2018.12922.
- [35] K. M. Htay, R. R. Othman, A. Amir, H. L. Zakaria, and N. Ramli, “A pairwise T-way test suite generation strategy using gravitational search algorithm,” in *2021 International Conference on Artificial Intelligence and Computer Science Technology*, Jun. 2021, pp. 7–12, doi: 10.1109/ICAICST53116.2021.9497823.
- [36] B. Swathi and H. Tiwari, “Integrated pairwise testing based genetic algorithm for test optimization,” *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 4, pp. 144–150, 2021, doi: 10.14569/IJACSA.2021.0120419.
- [37] S. I. Khaleel and R. Anan, “A review paper: optimal test cases for regression testing using artificial intelligent techniques,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 2, pp. 1803–1816, Apr. 2023, doi: 10.11591/ijece.v13i2.pp1803-1816.
- [38] A. A. Muazu, A. S. Hashim, A. Sarlan, and U. D. Maiwada, “Proposed method of seeding and constraint in one-parameter-at-a-time approach for T-way testing,” in *2022 International Conference on Digital Transformation and Intelligence*, Dec. 2022, pp. 39–45, doi: 10.1109/ICDI57181.2022.10007210.
- [39] A. S. Hashim, A. Aminu Muazu, M. A. M. Yusof, and N. I. Arshad, “Development of robot to improve learning of programming skills among students,” *Iraqi Journal for Computer Science and Mathematics*, vol. 4, no. 3, pp. 1–11, Jun. 2023, doi: 10.52866/ijcsm.2023.02.03.001.
- [40] A. A. Muazu and U. D. Maiwada, “PWiseHA: application of harmony search algorithm for test suites generation using pairwise techniques,” *International Journal of Computer and Information Technology*, vol. 9, no. 4, Jul. 2020, doi: 10.24203/ijcit.v9i4.23.
- [41] A. A. Muazu, A. S. Hashim, and A. Sarlan, “Application and adjustment of ‘don’t care’ values in t-way testing techniques for generating an optimal test suite,” *Journal of Advances in Information Technology*, vol. 13, no. 4, pp. 347–357, 2022, doi: 10.12720/jait.13.4.347-357.
- [42] D. Gupta and A. Rana, “Fibonacci driven novel test generation strategy for constrained testing,” in *2013 3rd IEEE International Advance Computing Conference (IACC)*, Feb. 2013, pp. 1475–1478, doi: 10.1109/IAdCC.2013.6514444.
- [43] J. M. Sharif, K. Z. Zamli, A. A. Bakar, S. Abdullah, I. S. Isa, and I. R. M. Noordin, “A non-deterministic T-way strategy with seeding and constraints support,” in *2012 IEEE Symposium on Humanities, Science and Engineering Research*, Jun. 2012, pp. 1153–1158, doi: 10.1109/SHUSER.2012.6268795.
- [44] A. A. Al-Omoush, A. A. Alsewari, H. S. Alamri, and K. Z. Zamli, “Comprehensive review of the development of the harmony search algorithm and its applications,” *IEEE Access*, vol. 7, pp. 14233–14245, 2019, doi: 10.1109/ACCESS.2019.2893662.

BIOGRAPHIES OF AUTHORS






Aminu Aminu Muazu    obtained his B.Sc. in computer science from Umaru Musa Yar’adua University, Nigeria, in 2011, followed by an M.Sc. in software engineering from Universiti Malaysia Pahang in 2017. Currently, he is pursuing a Ph.D. in information technology (software engineering) at Universiti Teknologi PETRONAS, Malaysia. His primary research interests lie in software engineering, combinatorial t-way software testing, and optimization algorithm. Aminu has published research papers which are indexed by ISI and SCOPUS. Additionally, he serves as an academic staff member, holding the position of Lecturer I at Umaru Musa Yar’adua University. He can be contacted at email: aminu.aminu@umyu.edu.ng.






Ahmad Sobri Hashim    is a senior lecturer at Universiti Teknologi PETRONAS, specializing in information technology. He holds a Ph.D., M.Sc., and B.Tech. from the same university. Ahmad is passionate about teaching programming, particularly in structured programming and web programming. His research interests encompass IT applications, e-systems, human-computer interaction, educational technologies, learning disabilities, and information systems. With a teaching career since 2014, Ahmad has published numerous research papers since 2009, many of which are indexed by ISI and SCOPUS. Notably, he has received recognition, winning gold medals in prestigious competitions like Malaysia Technology Expo (MTE 2017), International Competition and Exhibition on Computing Innovation (ICE-CInno 2016), Pertandingan Rekaipita dan Inovasi Institusi Pengajian Tinggi Swasta (PERISTIS 2016), and International Invention, Design, and Articulation (i-IDEA 2016). He can be contacted at email: sobri.hashim@utp.edu.my.






Umar Danjuma Maiwada    holds a B.Sc. in computer science from Bayero University Kano and an MSc in computer science from Jodhpur National University, India. Currently working as a Lecturer I at Umaru Musa Yaradua University, Katsina, he is dedicated to education, striving for excellence and precision to contribute effectively. He is pursuing a Ph.D. in the IT Department at Universiti Teknologi PETRONAS. He can be contacted at email: umar.danjuma@umyu.edu.ng.






Umar Audi Isma'ila    is pursuing an MSc in information technology at Universiti Teknologi PETRONAS, Malaysia. He earned his BSc in computer science from Umaru Musa Yar'adua University in 2020. Umar's research focuses on software engineering and blockchain technology. He has published several research papers since 2021, with some papers indexed by Scopus. He can be contacted at email: umar_22005104@utp.edu.my.



Muhammad Muntasir Yakubu    currently works at Federal University Dutsinma in the Department of Information Technology as a Lecturer II, received bachelor of science in computer from Umaru Musa Yar'adua University, Katsina, Nigeria in 2011. He is currently pursuing his PhD. in information technology with the Computer and Information Science Department, Universiti Teknologi PETRONAS (UTP) Malaysia. His research interests are in blockchain technology, software engineering, and data mining. He can be contacted at email: [ymmuhhammad@fudutsinma.edu.ng](mailto:ymmuhammad@fudutsinma.edu.ng).



Muhammad Abubakar Ibrahim    currently works at Federal College of Education Katsina in the Department of Computer Science as a Lecturer II, received bachelor of science in computer from Umaru Musa Yar'adua University, Katsina, Nigeria in 2011. He is currently pursuing his MSc. in computer science with Nigerian Defence Academy Kaduna, Nigeria with the Faculty of Science. He can be contacted at email: muhammadabubakar65@gmail.com.