DOI: 10.15849/IJASCA.241130.15

Int. J. Advance Soft Compu. Appl, Vol. 16, No. 3, November 2024
Print ISSN: 2710-1274, Online ISSN: 2074-8523

Wingsuit Flying Search Optimization Algorithm Strategy for Combinatorial T-Way Test Suite Generation

Nurol Husna Che Rose[1], Rozmie Razif Othman[2], Hasneeza Liza Zakaria[3], Anjila J Suali[4],and ZA Ahmad[5]

[1][2][3][4]Advanced Computing, Centre of Excellence (CoE), Universiti Malaysia Perlis, Malaysia
e-mail:husnarose@unimap.edu.my[1] , rozmie@unimap.edu.my [2], hasneeza@unimap.edu.my[3], anjilajsuali91@gmail.com[4];
[5]Faculty of Electronic Engineering Technology (FTKEN), Universiti Malaysia Perlis, Malaysia
e-mail:zahari@unimap.edu.my[5]

Abstract

Advancement in software development has resulted in complex software applications that encompass various functional and non-functional requirements. Such a complex system usually consists of many inputs either directly from users or from other connected systems or devices. Here, there is a potential for the system to go wrong due to certain combinations of inputs. Combinatorial Testing (or T-Way Testing) is effective in tackling the issue. Numerous studies have proposed strategies in generating T-Way test suite, and current trend indicates that researchers often incorporate metaheuristic algorithms in their proposed strategies. Many recent studies employ parameter optimization algorithms such as (Whale Optimization Algorithm, Particle Swarm Optimization, Gravitational Optimization Algorithm) in generating an optimized T-Way test suite. Often, researchers need to tune the parameters involved in the algorithm before the algorithm can be used for test suite generation. Since the system under test (SUT) can come in various numbers of input, it is impossible to find a single best value for every algorithm parameter. As a result, this paper proposed a T-Way Test suite generator utilizing Wingsuit Optimization Algorithm (a parameter free optimization algorithm) for combinatorial test suite generation. The algorithm learnt by itself as the optimization process progresses and hence eliminates the need for control parameters. Statistical analysis shows that WFS produces a smaller test suite compares to most T-Way strategies and in some cases, the difference between test suite size produce by WFS and other T-Way strategies are insignificant.

Keywords: *Combinatorial testing, Optimization Algorithm, T-Way test suites, Wingsuit Flying Search Algorithm*

Received 14 September 2024; Accepted 20 November 2024

1. Introduction

The swift evolution of technology as well as increasingly complex systems to satisfy human needs and wants require systems to operate efficiently and systematically. A complex system certainly involves numerous interactions between various parameters and is also susceptible to faults and data interaction errors. Therefore, software testing has been recognized as one of the important stages in the software development life cycle. It plays a crucial role for quality assurance, minimizing errors and reducing costs in software development [1]. While exhaustive testing is ideal for examining all potential parameter combinations, this approach is often impractical due to its time-consuming nature, particularly for large and excessively complex systems.

From literature, many researchers suggest the used of combinatorial testing (often refers as T-Way testing where *t* denoted interaction strength) as the replacement of exhaustive testing for multi-parameters system testing. T-Way testing efficiently reduces the number of test cases by excluding tuples comprising combinations of parameters that are covered by a single test case at most[2] . The research on T-Way testing is still progressing and recent trend shows that researchers often employed metaheuristic optimization algorithm in generating T-Way test suite. Various strategies have been introduced by researchers which include Particle Swarm Optimization (PSO), Harmony Search (HS), Cuckoo Search (CS), Flower Strategy (FS), Artificial Bee Colony (ABC), Ant Colony System (ACS), Dragonfly Algorithm, Adaptive Teaching Learning Based Optimization (ATLBO), Whale Optimization Algorithm (WOA), Gravitational Search Algorithm (GSA), Sine Cosine Algorithm (SCA) and many more. These strategies reported to produce an optimal T-Way test suite in many benchmark experiments.

However, according to the No Free Lunch Theorem, no algorithm works best for all possible situation or system configurations. Hence, the integration of optimization algorithms into T-Way testing remains an ongoing and open research endeavor. Motivated by the above problem, this paper introduces a newly developed algorithm called the Wingsuit Flying Search (WFS), which was design in 2020.

The WFS algorithm is a unique global optimization method based on population dynamics and features stochastic search behavior. Inspired by the extreme sport of wingsuit flying, it emulates the flier's objective to reach the lowest point of the terrain, analogous to finding the global minimum in the search space.

Apart from the population size and the maximum number of iterations, this approach offers the advantage of being parameter-free. Additionally, its ability to optimize different sections of the search space independently makes it suitable for parallelized computation [3]. Furthermore, it is considered a simple and lightweight algorithm. While traversing the search space, WFS can rapidly converge on the global optimal solution of the objective function at hand. An extensive computational investigation, encompassing 30 classical and 10 CEC 2020 benchmark functions, showcases the promising capabilities of the WFS algorithm, as it consistently outperforms competing methods across various scenarios [4].
Consequently, WFS has been widely utilized to tackle complex optimization problems across diverse fields, including energy management [5], [6] network communication [7], and condition and fault monitoring [8]. As a new metaheuristic algorithm, it appears the

potential of WFS is not fully explored since the T-Way implementation is not utilized in the algorithm. So, this work has 2 main contributions which are: -

1. the design and implementation of WFS algorithm as the optimization algorithm in T-Way test suite generation.
2. evaluation and comparison of the performance of WFS in generating T-Way test suite across various benchmark experiments.

The organization of this paper is structured as follows: Section 2 discusses related works on T-Way strategies. Section 3 provides background information on the combinatorial T-Way testing strategy. Sections 4 introduce the Wingsuit Flying Search (WFS) Algorithm and its implementation in T-Way testing. Section 5 presents preliminary results and discusses the benchmarking outcomes. Finally, Section 6 concludes this research.

2. Related Work

In general, there are two strategic approaches in T-Way testing which are computational and metaheuristic. The computational-based method is flexible approach derived from the algebraic method. It uses pure computational strategies to construct the test case. The strategies adopting computational can be categorized into two categories which are one-parameter-at-a-time (OPAT) or the one-test-at-a-time (OTAT) strategy approach.

The OPAT strategy employs both horizontal and vertical extension techniques to generate a test suite for a T-Way combination. First, OPAT starts with the initial pair parameter of the system input [9]. Then, it expands horizontally by adding one parameter at each step by prioritizing the parameter that covers the most tuples. The expansion process continues iteratively until all tuples are covered. However, if there are remaining uncovered tuples, OPAT will extend a new test case vertically. IPOG and IPOG-D is among the algorithms that employ the OPAT approach.

Another testing strategy is the OTAT strategy. It constructs a test suite by integrating a single test case that covers the maximum number of tuples at once. Uncovered tuples are addressed by including the next test case and this process repeats iteratively until all tuples are covered. The tool that utilizing OTAT involves Jenny, Pairwise Independent Combinatorial Testing (PICT) and Test Configuration (TConfig).

Apart from the computational approach, there is also another strategy to produce a T-Way test suite by using the metaheuristic approach. Metaheuristic algorithms often mimic diverse sources of inspiration, such as natural phenomena or behaviors[10]. In general, metaheuristic strategies may begin with either a single or population-based random solution. Then, a search space technique either exploration or exploitation is iteratively applied in attempt to improve them. During each iteration the fitness for each population point is calculated and the best candidate solution is selected and added to the final test suite. Metaheuristic can be categorized based on their source of inspiration which is biological or science-inspired.

The examples of biological-inspired metaheuristics include Cuckoo Search (CS), and. Artificial Bee Colony Strategy (ABC), Migrating Bird Optimization (MBO). For science-inspired metaheuristics and hybridization algorithms include the QLearning Sine Cosine

Algorithm (QLSCA), Graph Based Greedy Algorithm (GBGA), Genetic Strategy (GS), Flower pollination algorithm (FPA) and Improved Jaya Algorithm (IJA).

One of the first strategies to use the OPAT approach is the in-parameter-order (IPO) strategy [11]. It is a pairwise strategy (two-way) based on vertical and horizontal extension. It starts by creating a test set for pairs of the first two parameters. Then, it expands the test set to include pairs of the first three parameters, and continues this process until all parameters are covered. If necessary, it will then perform a vertical extension to address any uncovered interactions to address general T-Way support with variant algorithms for optimizing the horizontal and vertical extension. The strategy was later developed into IPOG [12] and IPOG-D [13] which handle general T-Way interactions and optimize both horizontal and vertical extensions.

Then, [14] develops TConfig (Test Configuration) that builds test suites using a recursive process based on orthogonal arrays. Although useful, orthogonal arrays are typically only applicable in tiny configurations. Thus, computationally based techniques that enable very big configurations have attracted a lot of interest.

On the other hand, in 2008, Czerwonka developed the test suite generating technique named Pairwise Independent Combinatorial Testing known as PICT [15]. It is extensively employed by Microsoft for the purpose of software testing. PICT first creates all potential tuples and designates each one as uncovered. The tuples involved in any restrictions declared by the test engineer will be marked as excluded. Next, using the greedy heuristic strategy, one uncovered tuple will be chosen and continued until it is completed. The completed test case will be added to the final test suite, and the tuples it addresses will be marked as covered. This process will keep going until all tuples are covered.

Next, Jenkins introduced a deterministic T-Way generation approach known as 'Jenny' [16]. Jenny utilizes a greedy algorithm to construct a test suite incrementally. In Jenny, each feature has its own T-Way interactions to cover. It starts with one-way interactions (single features), then moves to two-way interactions (combinations of two features), and so on. This means that while one feature covers two-way interactions, another begins working on three-way interactions, and this continues until all interactions are covered in the test suite.

Another computational algorithm is Particle Swarm Optimization (PSO). It was first used in T-Way testing in 2010 to generate a test suite. The Particle Swarm Test Generator (PSTG) [17], mimics the cognitive abilities observed in fish schools and flocks of birds when they search for food. Each member within flocks and schools progresses towards both the most favorable individual position and the optimal global positions. Each time a test case is randomly generated, the velocity is adjusted based on the best test case discovered so far. The particle then progresses to the next improved location and generate the best test case. This process continues until all termination criteria are met. PSO then managed to produce a new variant include DPSO [18] and APSO[19]. The Mamdani-type Fuzzy Inference System (FIS) and Particle Swarm Optimization are hybridized to create adaptive particle swarm optimization (APSO). FIS is used to maximize PSO's parameters.

The Cuckoo Search (CS) are motivated by certain birds' brood parasitic behavior such as the Ani and Guira cuckoos. By increasing the search in nearby areas of the current solution and effectively exploring the whole search area with the use of levy flights, CS offers the

best possible balance between local intensification and global diversification. In 2015, CS was adapted in T-Way [20] by tuning the nest size, the elitism probability, and the repetition parameter.

Then, Migrating Bird Optimization (MBO) was implemented for test data generation in 2016 by [21]. The MBO method looks for the finest test cases by utilizing the energy-saving habits of various long-distance flying birds through neighborhood search. The two unique aspects of MBO are the benefit-sharing system and parallel solution processing to minimize number of test suite.

Meanwhile. Genetic Strategy (GS)[22] is a version of GA that modifies the crossover and mutation operators. By modifying the bit structure and providing fast access to test cases, GS enhances the fitness function's performance. The modifications and the reduction of GA's complexity in the suggested GS reduce the size of the test suite and speed up its creation.

Consequently, Flower Pollination Algorithm (FPA) is inspired by the pollination behavior of flowering plants. It is a process of transferring pollen grains from the male part of one flower (anther) to transferred to the female part (stigma) of another flower by pollinators such as insects, birds, or wind. In 2017, [23] adopts FPA in T-Way named Pairwise Flower Strategy (PairFS). This efficient method with lesser control parameters shows best result in generated pairwise test suite size.

In addition, the Artificial Bee Colony Strategy (ABC) is adopted in T-Way. ABC was created by mimic the eating habits of a colony of honey bees. Honey is represented as test cases and the high-quality honey serve as the test cases with maximum covered interaction element. The Artificial Bee Colony Strategy (ABCVS) is a two-way generation strategy based on the Artificial Bee Colony (ABC) algorithm for a uniform and variable strength test suite [24].

Also, Kidney Algorithm (KA) simulates the function of the kidneys in a living organism. Filtration (local search) and reabsorption (global search) are the two primary steps of KA. To provide a reduced test suite, the Pairwise Kidney Strategy (PKS) was created based on KA in 2018[25]. Urine formation in KA occurs through four primary mechanisms. These procedures include filtration, reabsorption, secretion, and excretion. Solutes and water from the blood are moved to the kidney's tubules in the filtration process. It separates filtered blood (FB) as good solution and waste group (W) for worse solution (local search). The second phase, known as reabsorption, functions as a global search in which test cases in (W) are reevaluated and sent back to FB. Then, in the secretion phase, test cases that have been added to FB are then reviewed, and the test case that has inadequate quality coverage is returned to W. Lastly, test cases in W are eliminated and replaced with newly created test cases during the excretion process.

The Sine Cosine Algorithm (SCA) and the Q-learning algorithm are combined to form the Q-learning Sine Cosine Algorithm (QLSCA)[26]. Instead of using a reward and penalty system to determine the best course of action during runtime, the SCA switching probability method is replaced with the Q-learning technique. To further improve the solution diversity and enable jumping out of local optima, the Lévy flying motion and crossover are incorporate into the QLSCA.

The Graph Based Greedy Algorithm (GBGA) is a competitive greedy algorithm that constructs CAs using a graph representation [27]. It contributes to the construction of a graph representation for the problem of constructing CAs and MCAs, as well as the invention of a competitive greedy algorithm to solve the problem in the graph domain.

Later, a new variant of the Jaya algorithm for generating T-Way test suites has been introduced, known as the Improved Jaya Algorithm (IJA). IJA enhances both the intensification and diversification capabilities by incorporating new search operators such as Lévy flight and mutation operators into the Jaya Algorithm. By applying Lévy flight and mutation operators, IJA aims to improve the search effectiveness of the original Jaya Algorithm. Another variant of the Jaya Algorithm is the Latin Hypercube Sampling Strategy (LHS-JA) [28].

Following that, the Sine Cosine Algorithm (SCA) is a metaheuristic optimization technique inspired by the sine and cosine mathematical functions found by [29]. SCA maintains a balance between exploration and exploitation by combining global and local search strategies.

In addition, Optimization algorithms can also be categorized into parametric and non-parametric. A parametric algorithm means that the algorithm is parameter dependent which requires specific parameters that affect its behavior and performance. This algorithm involves tuning parameters where each specific value will affect the results received [30]. The good thing about this parametric feature is that it provides greater flexibility as it allows the user to fine tune parameters. Also, higher control and customization features as parameters can be adjusted for needs which can lead to more precise results for different scenarios.

However, finding the right values may be time consuming as it requires trial and error which may be time intensive [31]. It is also exposed to the risk of suboptimal performance as poorly tuned parameters can lead to failing and unreliable result. This tuning often requires expertise and making these algorithms harder to apply for non-experts. Examples of parameter optimization algorithms include IPOG, IPOG-D, PICT, TConfig, Cuckoo Search (CS), Artificial Bee Colony Strategy (ABCS), Migrating Bird Optimization (MBO), Genetic Strategy (GS), Flower Pollination Algorithm (PFA), Improved Jaya Algorithm (IJA), DPSO, APSO, Pairwise Flower Strategy (PFS), Kidney Algorithm (KA), and Latin Hypercube Sampling Strategy.

Unlike parameter-dependent algorithms, parameter-free algorithms operate without requiring parameter setup or manual tuning. They often include self-adaptive mechanism based on their operational parameters in the optimization process based on real time feedback [32]. Among the advantages of non-parameter algorithms are its ease of use as it eliminates the need for manual parameter tuning. This makes them more accessible for users without the need for specialized expertise. Also, since they self-tune, the algorithms are robust across applications. The algorithm can perform reliably across various problems and are less sensitive to specific conditions. Nevertheless, this type of algorithm may not be optimal for all problems as it may not achieve the highest level of performance for specialized problems compared to the well-tuned parametric algorithm [33]. Additionally, there may be slower adaptation due to reduced control over the algorithm's behavior. Since the algorithm self-adjusts, users have limited influence over specific actions, which can be

a disadvantage in some scenarios. The QLearning Sine Cosine Algorithm (QLSCA) and the Graph Based Greedy Algorithm (GBGA) are examples of non-parametric algorithms.

3. Overview of T-Way Testing

To illustrate the concept of T-Way interaction testing, let us use a hypothetical example of an online bookstore application depicted in Fig 1. This application comprises four primary components: Type, Author, Publisher, and Price Range. The "Type" component offers three values: Crime and Thriller, Fantasy, and Historical Fiction, whereas the "Author" component presents two possible values: Agatha Jones and Aaron, Jason. Additionally, the "Publisher" component offers two possible values: Bloomsbury Publishing and Faber & Faber. The "Price Range" component also presents two possible values: Under RM50 and RM50 to RM100.

In terms of computation, the potential size of the test suite for a combination can be calculated by multiplying the values of N and K, where N represents the number of test cases and K represents the number of parameter values. If applied to a complex system with numerous parameters, the resulting number of test cases can become unmanageably high and impractical for testing. For instance, consider a system with 15 parameters, each having 3 possible values. In such a scenario, the number of test cases required to cover all system configurations would be 315, which equals 14,348,907 test cases. This illustrates the risk of combinatorial explosion, where the number of test cases grows exponentially with the increase in the number of parameters and their respective values.

To address this challenge, T-Way testing is utilized as an alternative to reduce the number of test cases. It achieves this by examining the interactions between t parameters in every possible combination, thereby significantly reducing the total number of required test cases compared to exhaustive testing. For the online bookstore application, if exhaustive testing is performed on this system, the test suite size count would be 24. This calculation is derived from multiplying the values of 3 (for the "Type" component), 2 (for the "Author" component), 2 (for the "Publisher" component), and 2 (for the "Price Range" component), resulting in 3 * 2 * 2 * 2 = 24 test cases needed to cover all possible application configurations. Nevertheless, with the T-Way approach, only 7 test cases are required to cover all the same configurations. Fig 2 illustrates the generation of the T-Way test suite for the application.

Covering Array (CA) represents the test suite that demonstrates all the interactions of the parameters. It is denoted as CA (N; t, v^p), where: N is the size of the array or the number of produced test cases, t is the interaction strength, p is the input parameter and v is the value of the parameter. However, there are scenarios where the number of values is not consistent across parameters. To represent this in the covering array, we use Mixed Covering Array (MCA), denoted as MCA (N; t, k, (v^1, v^2, ..., v^p)), where N is the size of the array or the number of produced test cases is the interaction strength is the number of parameters and (v^1, v^2, ..., v^p) represents the number of values for each parameter. Referring to Fig 2, the Mixed Covering Array (MCA) for the online bookstore application can be described as MCA (7; 2, 4, (3, 2, 2, 2)). This signifies a test suite of 7 test cases with a covering strength of 2, encompassing 4 parameters. Among these parameters, one has 3 values, while the remaining three parameters each have 2 values.

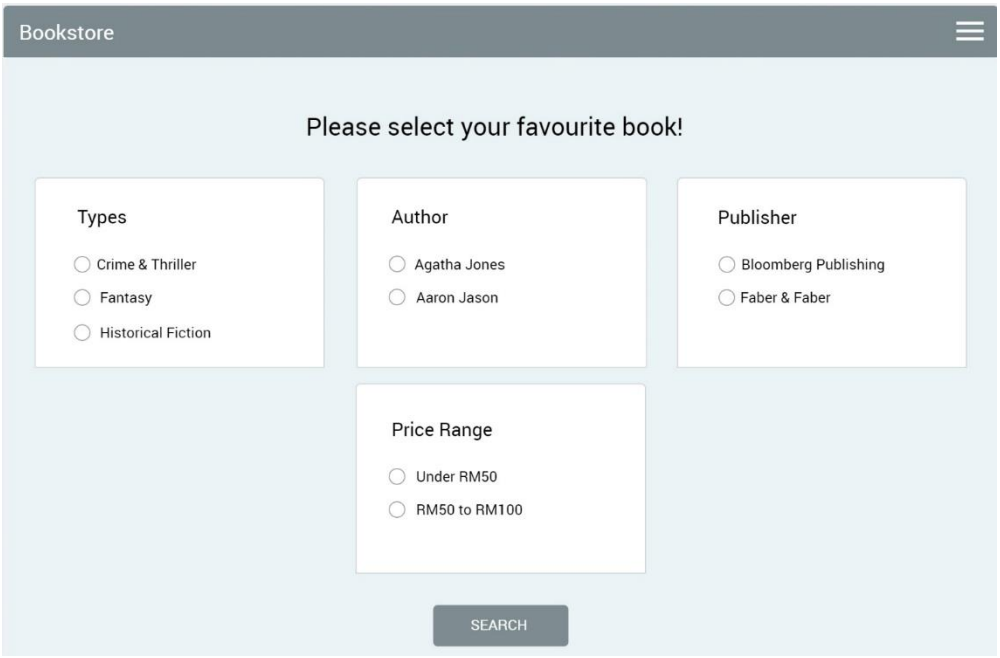

Fig 1: Online Bookstore Application Interface

t = 2

Online Bookstore Application			
Type	**Author**	**Publisher**	**Price Range**
Crime &Thriller	Agatha Jones	Bloomsbury Publishing PLC	Under RM50
Fantasy	Aaroon Jason	Faber & Faber	RM50- RM 100
Historical			

Construction of MCA (7; 2, 3^1,$2^{3)}$

Mixed Covering Array of MCA (7; 2, 3^1,$2^{3)}$

No of Test cases	Type	Author	Publisher	Price Range
1	Crime & Thriller	Agatha Jones	Bloomsbury Publishing	Under RM50
2	Crime & Thriller	Aaron Jason	Faber & Faber	RM50 - RM 100
3	Fantasy	Agatha Jones	Faber & Faber	Under RM50
4	Fantasy	Aaron Jason	Bloomsbury Publishing	RM50 – RM 100
5	Historical	Agatha Jones	Bloomsbury Publishing	Under RM50
6	Historical	Aaron Jason	Faber & Faber	Under RM50
7	Historical	Aaron Jason	Faber & Faber	RM50 - RM 100

Fig 2: T-Way mixed covering array construction for Online Bookstore Application.

4. WFS Optimization Algorithm

The Wingsuit Flying Search (WFS) algorithm is developed by Nermin Covic and Bakir Lacevic in 2020[34]. It introduces a novel approach to global optimization by drawing inspiration from the daring sport of wingsuit flying. Essentially, the WFS algorithm will start by getting a preliminary understanding of the search space. The flier representing the algorithm and the act of flying will be symbolizing the algorithm execution.

During the flight, the wingsuit flier will aim to land on the optimal landing spot which is considered as global minimum. The flier will try to avoid high and rough area and navigates towards wide and smooth surrounding. Over time, the flier will then gradually approach the designated area and landing safely. The act of unveiling flier arms for wing movement and adjustment of the flier legs towards global minimum can be interpret as algorithm's exploration and exploitation phases of the flier towards the targeted landing region. The flier is considered to have found the termination points once it successfully lands. It thereby automatically ends the algorithm's search.

The Wingsuit Flying Search (WFS) algorithm works in 3 phase which are first, generating initial points; secondly, determining neighborhood size for each point; and thirdly, generating neighborhood points. In the first phase, the N initial points (where N represents the population size) are located using the Halton Sequence. At the first iteration (m=1), each point will be in *n*-dimensional box. The point will be placed randomly and separated equally with the same initial discretization step distance. The initial positions of all points are visualized in Fig 3.

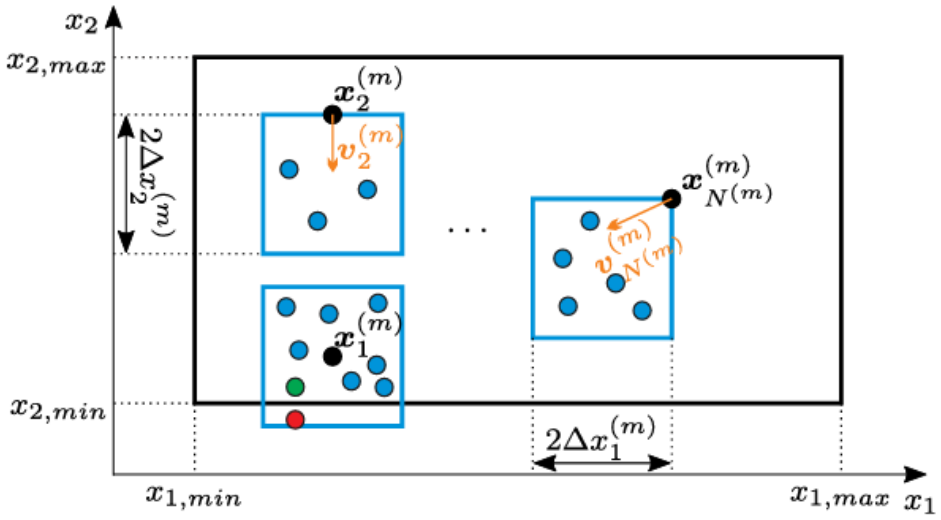

Fig 3: The layout of first N(m) points and their Halton neighborhood in 2D space.Source: [35]

Then, in the second phase, the neighborhood size for each point is determined. Starting from the second iteration (*m*=2), the points will be sorted in descending order and the first point is assumed as the best point which means that the point is assigned with the largest number of neighbors points ($P\_max^{(m)}$). The neighborhood size for each point is determined using the *equation (1)* and *equation (2)*. The equations are described as follows;

a) Calculate the neighborhood size using equation (1).

$$P^{(m)}(i) = P_{max}^{(m)} \left[\left(1 - \frac{i - 1}{N^{(m)} - 1}\right)\right] \tag{1}$$

b) Calculate the number of nodes using equation (2).

$$N^{(m)} = \frac{2N}{P_{max}^{(m)}} \tag{2}$$

The largest neighborhood size, $P\_max^{(m)}$ can be calculated based on *equation (3)*. The $\alpha^{(m)}$ or the search sharpness of the *m*-th iteration can be derived from the formula of $\alpha^{(m)}=1-v\text{^}(-(m-1)/(M-1))$. Where *M* denotes as the maximum iteration for the algorithm and *v* is the flier's velocity. The flier velocity (*v*) is set to a default value since it is reported to have insignificant impact on algorithm performance [35]

c) Calculate the largest neighborhood size using equation (3).

$$P_{max}^{(m)} = \alpha^{(m)} N \tag{3}$$

After the neighborhood size is assigned for each point, the new neighborhood point can be generated. The generation of neighborhood points heavily relies on the discretization step. This is due to the flier altitude decreases as it approaches the land, and similarly, the discretization step also decreases. The discretization step, $\Delta \boldsymbol{x}$, is calculated using *Equation (4)*. From the equation, the discretization step reduces as $\alpha^{(m)}$ approaches 1 towards the end of the iteration when *m* is equal to *M*.

d) Calculate the discretization step using equation (4).

$$\Delta \boldsymbol{x}^{(m)} = (1 - \alpha^{(m)})\, \Delta \boldsymbol{x}^{(1)}, \qquad m \geq 2 \tag{4}$$

The boundary of the neighborhood points is determined through the selection from the three sets algorithm outlined in *Equation (5)*. The neighborhood vector ($v_{k,i}^{(m)}$) will determine the direction for the boundary of the neighborhood points.

e) Calculate the boundary of neighborhood points using equation (5).

$$\begin{aligned}
S_{k,1}(x_i^{(m)}) &= \{ x_{k,i}^{(m)} - \Delta x_k^{(m)}, \; x_{k,i}^{(m)} \}, \quad \text{if } v_{k,i}^{(m)} < 0; \\
S_{k,2}(x_i^{(m)}) &= \{ x_{k,I}^{(m)}, \; x_{k,i}^{(m)} + \Delta x_k^{(m)} \}, \quad \text{if } v_{k,i}^{(m)} < 0; \\
S_{k,3}(x_i^{(m)}) &= S_{k,1}(x_i^{(m)}) \cup S_{k,2}(x_i^{(m)}), \quad \text{if } v_{k,i}^{(m)} = 0;
\end{aligned} \tag{5}$$

Fig 4 illustrates the neighborhood points represented by blue dots, while the potential neighborhood points are represented by white dots in the boundary. Here, *v* represents the vector subtraction between the current point and the current best location.

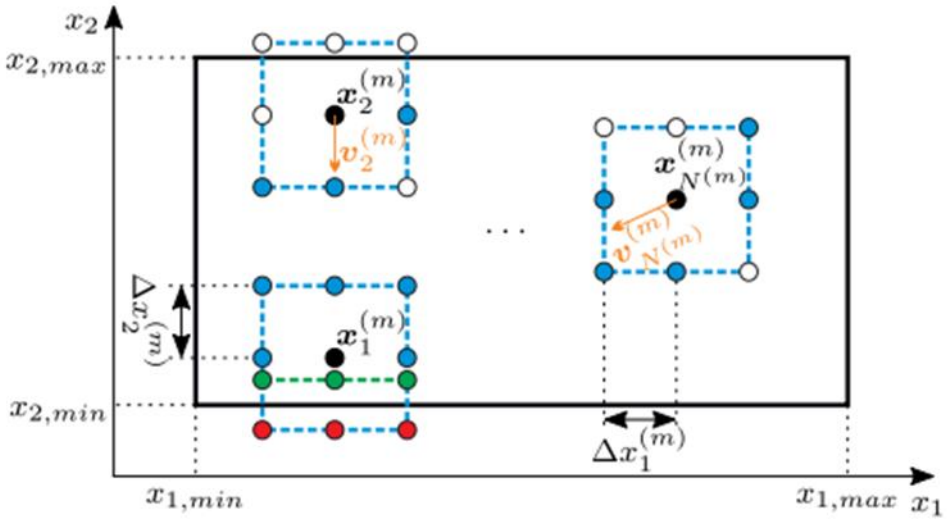

Fig 4: Generated Neighbourhood Points,
Source: (Covic & Lacevic, 2020)

At the end of each iteration, all new points will be added to $N(m)$ new points until reaching the maximum iteration(M). The point with the highest fitness will be considered as the output of the WFS algorithm.

Algorithm 1: Pseudocode of WFS

1. Initialize the required parameters.
2. Generate the initial points in an n-dimensional box using the Halton Sequence.
3. While m-iteration equals 2 and m < M do
   - 3.1 Determine the neighborhood size for each point using equations (1) and (2).
   - 3.2 Calculate the neighborhood point of each point using the discretization step result from equation (3).
   - 3.3 Calculate the new discretization steps using equation (4).
   - 3.4 Calculate the boundary using equation (5).
   - 3.5 Generate the neighborhood points using the current N and equations (1) and (2).
   - 3.6 Calculate the fitness for each point.
   - 3.7 Sort the points according to the fitness values in descending order.

4. end while
5. Return the best point (0).

5. Experiment and Discussion

In this experiment, WFS in T-Way will be benchmarked against existing T-Way strategies based on the test suite size value produced. A desktop PC with Windows 10, 2.40 GHz Intel(R) Core TM i5-1135G7, CPU with 4 GB of RAM was used to run the experiments along with the implemented WFS in Java 13.0.1 programming language using Eclipse software. The experiments are categorized into the following four groups.

A. Comparing WFS with the currently available strategies using CA (t, v7), the number of parameters remained constant, and their values varied. In addition, the interaction strength t ranged from 2 to 6. Test suite size results of the strategies are extracted from the published studies [21], [36], [37], [38].

B. Comparing WFS with the existing strategies using CA (t, 3p), the number of parameters was varied, and their values remained constant. In addition, the interaction strength t varied from 2 to 6. Test suite size results of the strategies are extracted from the published studies [38], [39], [40].

C. Comparing WFS with the existing strategies using CA (N; 2, 2, p), where the interaction strength, t is 2 and value, v is 2, but parameter, p is varied from 3 to 15 and additionally 50 and 100 to test higher configurations. Test suite size results of the strategies are extracted from the published studies [20], [21], [25], [38].

D. Comparing WFS with the existing strategies using CA (N; (t, 2, 10), where p = 10 and v = 2 but the interaction strength, t is varied from 2 to 10. Test suite size results of the strategies are taken from the published studies[21], [38], [41], [42].

All algorithms will be used through several experiments as stated and produce various test suite sizes. The experiments are repeated 30 times and the smallest test suite result value has been chosen as the best test suite size. The results have been included in Table 1 until Table 4. Decisions that are not available are marked as NA as in the published article. Also, in Table 5 until Table 7 shows the result for the Friedman Test of the algorithm.

Based on the results in Table 1, metaheuristic algorithms such GS, CS, ABVCS and WFS succeeded in producing the smallest test suite for all the elements in the first t configuration, t=2. For t=3, GS is the most superior with most test suite results being the smallest, followed by ABVCS and WFS.

Regarding the system configuration for CA (t, v7) where v varied from 2 to 7 and t varied from 2 to 6 as shown in Table 2, GS maintains its performance in producing the best number of test suite sizes for t=2. DPSO, APSO, CS and WFS are algorithms that are competitive with each other by producing results that are almost the same as each other. The test suite size for CS was initially more or less the same as DPSO, APSO and GS finally overtake at t=3, 4 and 5.

Concerning the results produced in Table 3, PKS outshine other algorithms by giving the best test suite size especially during p=7. It works side by side by giving results that are almost the same as the PSO, MBO, CS, FPA, and ABC algorithms. WFS is seen to provide the best test suite coverage so that p=50 while other algorithms cannot provide information based on data in published papers. PKS outperforms the lowest mean rank value followed by WFS for this experiment. As for result of system configuration with CA ($t,2^{10}$) where t varies from 2 to 9, WFS monopolizes the best test suite size especially at t= 7, 8 and 9. FPA also gives almost the same decision as WFS but gives a decision the best at t=5. Meanwhile, Jenny and TConfig give less impressive results, especially at t=4, 5 and 6.

Table 1. Test suite size performance for CA (t, 3^p) where p varied from 3 to 12 and t varied from 2 to 6

CA(t,3^P)		Pure computation strategies					AI-based strategies							
t	p	Jenny	TConfig	PICT	IPOG-D	IPOG	GBGA	QLSCA	GS	DPSO	APSO	CS	ABVCS	WFS
		Best	Best	Best	Best	Best	Best	Best	Best	Best	Best	Best	Best	Best
2	3	**9**	10	10	15	**9**	**9**	**9**	**9**	NA	**9**	**9**	**9**	**9**
	4	13	10	13	15	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**	**9**
	5	14	14	13	15	15	12	**11**	**11**	**11**	**11**	**11**	**11**	**11**
	6	15	15	14	15	15	14	14	13	14	**12**	13	13	13
	7	16	15	16	15	15	15	**14**	**14**	15	15	**14**	15	15
	8	17	17	16	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**	**15**
	9	18	17	17	**15**	**15**	16	**15**	**15**	**15**	16	**15**	16	16
	10	19	17	18	21	15	16	**15**	16	16	17	17	17	17
	11	17	20	18	21	17	17	**16**	**16**	17	NA	18	17	17
	12	19	20	19	21	21	18	**16**	**16**	**16**	NA	18	18	18
3	4	34	32	34	**27**	32	29	**27**	**27**	NA	**27**	28	**27**	28
	5	40	40	43	45	41	39	39	**38**	41	41	**38**	**38**	**38**
	6	51	48	48	45	46	45	**33**	43	**33**	45	43	44	46
	7	51	55	51	50	55	49	49	49	**48**	**48**	**48**	49	51
	8	58	58	59	**50**	56	54	52	54	52	**50**	53	54	56
	9	62	64	63	71	63	58	**56**	58	**56**	59	58	58	61
4	5	109	97	100	162	97	87	**81**	90	NA	94	94	98	96
	6	140	141	142	162	141	133	**129**	**129**	131	129	132	135	133
	7	169	166	168	226	167	156	**150**	153	**150**	154	154	157	160
5	6	348	305	310	386	305	**273**	NA	301	NA	NA	304	**273**	298
	7	458	477	452	678	466	433	NA	**432**	NA	NA	434	433	440
6	7	1089	**921**	1015	1201	**921**	982	NA	963	NA	NA	973	982	963
	8	1466	1515	1455	1763	1493	NA	NA	1399	NA	NA	**1401**	NA	1412

Received 14 September 2024; Accepted 20 November 2024

Table 2: Test suite size performance for CA (t, v^7) where v varied from 2 to 7 and t varied from 2 to 6

CA (t, v^7)		Pure computation strategies					AI-based strategies					
t	p	Jenny	TConfig	PICT	IPOG-D	IPOG	QLSCA	GS	DPSO	APSO	CS	WFS
		Best	Best	Best	Best	Best	Best	Best	Best	Best	Best	Best
2	2	8	7	7	8	7	7	**6**	7	**6**	**6**	7
	3	16	15	16	15	15	15	**14**	**14**	15	15	15
	4	28	28	27	32	29	**23**	24	24	25	25	27
	5	37	40	40	45	45	**34**	36	**34**	35	37	40
	6	55	57	56	72	55	48	52	**47**	NA	NA	57
	7	74	76	74	91	**49**	64	68	64	NA	NA	78
3	2	14	16	15	14	16	15	**12**	15	15	**12**	13
	3	51	55	51	50	55	49	49	49	**48**	49	51
	4	124	**112**	124	114	**112**	**112**	116	**112**	118	117	123
	5	236	239	241	252	237	**215**	221	216	239	223	242
	6	400	423	413	470	420	**364**	374	365	NA	NA	418
4	2	31	36	32	40	35	31	27	34	30	27	**25**
	3	169	166	168	226	167	**149**	153	150	153	155	156
	4	517	568	529	704	614	477	486	**472**	**472**	487	513
5	2	57	56	57	80	60	NA	**51**	NA	NA	53	**51**
	3	458	477	452	678	466	NA	**432**	NA	NA	439	442
	4	1938	**1792**	1933	2816	**1792**	NA	1821	NA	NA	1845	1861
6	2	87	**64**	72	96	**64**	NA	65	NA	NA	66	66
	3	1087	**921**	1015	1201	**921**	NA	963	NA	NA	973	955
	4	6127	NA	5847	5120	**4096**	NA	5608	NA	NA	5610	5610

Table 3: Test suite size performance for CA (2, 2^p) where p varied from 3 to 15, 50 and 100

CA(2,2^P)	Pure computation strategies					AI-based strategies strategies					
p	Jenny	TConfig	PICT	IPOG	PSO	MBO	CS	FPA	ABC	PKS	WFS
	Best	Best	Best	Best	Best	Best	Best	Best	Best	Best	Best
3	5	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**	**4**
4	6	6	**5**	6	6	6	**5**	6	**5**	**5**	6
5	7	**6**	7	8	**6**	**6**	**6**	**6**	**6**	**6**	**6**
6	8	7	**6**	8	7	7	**6**	7	7	**6**	**6**
7	8	9	7	8	7	7	7	7	7	**6**	7
8	8	9	8	8	8	**7**	8	8	8	**7**	**7**
9	**8**	9	9	10	**8**	**8**	**8**	**8**	**8**	**8**	**8**
10	10	9	9	10	**8**	**8**	**8**	**8**	**8**	**8**	**8**
11	9	9	9	10	9	**8**	**8**	**8**	9	**8**	**8**
12	10	9	9	10	9	**8**	9	9	9	**8**	**8**
13	10	9	9	10	9	9	NA	NA	9	**8**	**8**
14	10	**9**	10	10	**9**	**9**	NA	NA	**9**	**9**	**9**
15	10	**9**	10	10	10	**9**	NA	NA	**9**	**9**	**9**
50	NA	NA	NA	NA	NA	NA	**12**	NA	NA	NA	**12**
100	NA	NA	NA	NA	**15**	NA	**15**	NA	NA	NA	NA

Table 4: Test suite size performance for CA (t,2^{10}) where t varied from 2 to 9

CA(t,2^{10})	Pure computation strategies					AI-based strategies			
t	Jenny	TConfig	PSO	CS	IMTS	FPA	IJA	LHS-JA	WFS
	Best	Best	Best	Best	Best	Best	Best	Best	Best
2	10	9	**8**	**8**	**8**	**8**	**8**	**8**	**8**
3	18	20	17	**16**	**16**	17	NA	NA	**16**
4	39	45	37	**36**	38	39	NA	NA	38
5	87	95	82	**79**	80	82	**79**	79	84
6	169	183	158	**157**	159	160	159	159	160
7	311	NA	NA	NA	NA	NA	297	301	**291**
8	521	NA	NA	NA	NA	NA	502	515	**500**
9	788	NA	NA	NA	NA	NA	584	584	**574**
10	**1024**	NA	NA	NA	NA	NA	1024	**1024**	**1024**

Table 5: Wilcoxon test for the result reported on Table 1 and Table 2

Algorithm	Ranks			Test Statistics	
	WFS>	WFS<	WFS =	Asymp. Sig (2-Tailed)	Null Hypothesis
Jenny	5	34	4	< 0.01	Reject the null hypothesis
TConfig	7	29	6	0.007	Reject the null hypothesis
PICT	4	34	5	< 0.01	Reject the null hypothesis
IPOG-D	8	32	3	<0.01	Reject the null hypothesis
IPOG	11	23	9	0.15	Accept the null hypothesis
QLSCA	23	4	6	<0.01	Reject the null hypothesis
DPSO	13	10	7	0.337	Accept the null hypothesis
APSO	16	3	8	0.01	Reject the null hypothesis
CS	8	20	12	0.182	Accept the null hypothesis

To ensure that there is a significant difference between WFS and other algorithms, a non-parametric statistical test, the Wilcoxon Signed Rank Test, is conducted. The null hypothesis suggests that the median difference between paired observations is zero,

Received 14 September 2024; Accepted 20 November 2024

indicating no effect or difference. Table 5 presents the test statistic and the ranks of WFS compared with other optimization algorithms. In the "Ranks" section, "WFS >" indicates the number of times WFS has a larger test case than the other algorithms. "WFS <" indicates the number of times WFS has a smaller test case, and "WFS =" means the number of times WFS produces the same test case value as the compared algorithm. For the test statistic, the Asymp. Sig (2-Tailed) value determines the relevance of an algorithm. If the Asymp. Sig (2-Tailed) value is less than 0.05, the null hypothesis is automatically rejected. This means that the statistical test has provided sufficient evidence to conclude that the null hypothesis is unlikely to be true.

The result shows that WFS mostly produces better results when 6 out of 9 tests show that WFS produces a smaller number of test suites. WFS perform better test cases compared to Jenny, TConfig, PICT, IPOG-D, IPOG and CS algorithm. Even though the number of test cases that produced by WFS is exceeded more compared to QLSCA, DPSO and APSO, but Asymp. Sig (2-Tailed) value is still significant and the difference in test suite size between WFS and other T-Way strategies is minimal.

6. Conclusion

This paper presents the first implementation of WFS in T-Way testing for test suite generation. The results from various experiments indicate that WFS is both promising and competitive. Furthermore, WFS has ranked among the top algorithms, performing on par with other metaheuristic approaches. Due to the nature of this algorithm, WFS start by the process of a flier to land on earth and represent as the global optimum. However, as the flier become nearer to the global optimum, the search sharpness will increase constantly towards 1 for every iteration. As a result, the discretization step also will constantly decrease. Therefore, the search space become smaller and the WFS algorithm will focuses more on exploitation as the algorithm progress. The algorithm eventually will become pure exploitation at the end of the searching process. As a result, the searching process in WFS algorithm has a potential to stuck at the local optima.

To fully explore the potential of WFS in T-Way implementation, it may be beneficial to hybridize it with a global metaheuristic algorithm as the future work. This, in turn, will provide a balance between exploration and exploitation in finding the optimal number of test cases for combinatorial t-way test suite generation. Examples of potential global optimization algorithms that provide good exploration capabilities include Simulated Annealing, Genetic Algorithm, and Lévy Flight. Future work will extend the application of WFS in T-Way testing to include variable strength and input-output relationship testing. Additionally, combining WFS with other existing algorithms could be explored to further enhance its performance in T-Way testing.

ACKNOWLEDGEMENTS
The author would like to acknowledge the support from the Fundamental Research GrantScheme (FRGS) under a grant number of FRGS/1/2023/ICT01/UNIMAP/02/1 from the Ministry of Higher Education Malaysia.

References

[1] N. Anwar and S. Kar, 2019"Review Paper on Various Software Testing Techniques & Strategies," *Global Journal of Computer Science and Technology*, vol. 19, no. 2, pp. 43–49, doi: 10.34257/gjcstcvol19is2pg43.

[2] M. Z. Z. Ahmad, R. R. Othman, M. S. A. R. Ali, N. Ramli, M. W. Nasrudin, and A. A. A. Halim, 2021"A Tuned Version of Ant Colony Optimization Algorithm (TACO) for Uniform Strength T-way Test Suite Generator: An Execution's Time Comparison," *J Phys Conf Ser*, vol. 1962, no. 1, doi: 10.1088/1742-6596/1962/1/012037.

[3] N. Ramli, R. R. Othman, and M. S. A. R. Ali, "Optimizing combinatorial input-output based relations testing using Ant Colony algorithm," in *2016 3rd International Conference on Electronic Design, ICED 2016*, 2017. doi: 10.1109/ICED.2016.7804713.

[4] S. Zhao, T. Zhang, S. Ma, and M. Chen, 2022"Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications," *Eng Appl Artif Intell*, vol. 114, doi: 10.1016/j.engappai.2022.105075.

[5] V. Prasanna Moorthy, S. Siva Subramanian, V. Tamilselvan, S. Muthubalaji, P. Rajesh, and F. H. Shajin, 2022"A hybrid technique based energy management in hybrid electric vehicle system," *Int J Energy Res*, vol. 46, no. 11, pp. 15499–15520, doi: 10.1002/er.8248.

[6] B. Venkatesh, P. Sankaramurthy, B. Chokkalingam, and L. Mihet-popa, 2022"Managing the Demand in a Micro Grid Based on Load Shifting with Controllable Devices Using Hybrid WFS2ACSO Technique," *Energies (Basel)*, vol. 15, no. 3, doi: 10.3390/en15030790.

[7] W. K. Ahmed, M. N. bin M. Warip, W. K. Abduljabbar, and M. Elshaikh, 2022"Ws-Olsr: Multipoint Relay Selection in Vanet Networks Using a Wingsuit Flying Search Algorithm," *International Journal of Computer Networks and Communications*, vol. 14, no. 6, pp. 37–49, doi: 10.5121/ijcnc.2022.14603.

[8] Y. Mao, F. Xu, X. Zhao, and X. Yan, 2021"A gearbox fault feature extraction method based on wingsuit flying search algorithm-optimized orthogonal matching pursuit with a compound time-frequency atom dictionary," *Journal of Mechanical Science and Technology*, vol. 35, no. 11, pp. 4825–4833, doi: 10.1007/s12206-021-1002-5.

[9] A. A. Muazu, A. S. Hashim, A. Sarlan, and U. D. Maiwada, 2022"Proposed Method of Seeding and Constraint in One-Parameter-At-a-Time Approach for t-way Testing," *2022 International Conference on Digital Transformation and Intelligence, ICDI 2022 - Proceedings*, no. Icdi, pp. 39–45, doi: 10.1109/ICDI57181.2022.10007210.

[10] K. Z. Zamli, R. R. Othman, M. I. Younis, and M. H. Mohamed Zabil, "Practical adoptions of T-way strategies for interaction testing," in *Communications in Computer and Information Science*, 2011. doi: 10.1007/978-3-642-22203-0_1.

[11] Y. Lei and K. C. Tai, "In-parameter-order: A test generation strategy for pairwise testing," in *Proceedings - 3rd IEEE International High-Assurance Systems Engineering Symposium, HASE 1998*, 1998. doi: 10.1109/HASE.1998.731623.

[12] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, 2007"IPOG: A general strategy for T-way software testing," *Proceedings of the International Symposium and Workshop on Engineering of Computer Based Systems*, pp. 549–556, doi: 10.1109/ECBS.2007.47.

[13] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, 2008"IPOG-IPOG-D: Efficient test generation for multi-way combinatorial testing," *Software Testing Verification and Reliability*, vol. 18, no. 3, pp. 125–148, doi: 10.1002/stvr.381.

[14] A. W. Williams, "Determination of Test Configurations for Pair-Wise Interaction Coverage," 2000. doi: 10.1007/978-0-387-35516-0_4.

[15] J. Czerwonka, 2008"Pairwise Testing in the Real World: Practical Extensions to Test-Case Scenarios," *Proceedings of 24th Pacific Northwest Software Quality Conference*, pp. 419–430.

[16] B. Jenkins, "jenny: a pairwise testing tool," Jenny Test Tool. [Online]. . Accessed: Jul. 29, 2024. [Online]. Available: https://burtleburtle.net/bob/math/jenny.html

[17] B. S. Ahmed, 2014"PSTG : A t-way strategy adopting particle Swarm Optimization PSTG : A T-Way Strategy Adopting Particle Swarm Optimization," no. May, doi: 10.1109/AMS.2010.14.

[18] H. Wu, C. Nie, F. C. Kuo, H. Leung, and C. J. Colbourn, 2015"A Discrete Particle Swarm Optimization for Covering Array Generation," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 575–591, doi: 10.1109/TEVC.2014.2362532.

[19] T. Mahmoud and B. S. Ahmed, 2015"An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use," *Expert Syst Appl*, vol. 42, no. 22, doi: 10.1016/j.eswa.2015.07.029.

[20] A. B. Nasser, A. R. A. Alsewari, and K. Z. Zamli, 2015"Tuning of cuckoo search based strategy for T-way testing," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 19.

[21] H. L. Zakaria and K. Z. Zamli, "Migrating Birds Optimization based strategies for Pairwise testing," in *2015 9th Malaysian Software Engineering Conference, MySEC 2015*, 2016. doi: 10.1109/MySEC.2015.7475189.

[22] S. Esfandyari and V. Rafe, 2018"A tuned version of genetic algorithm for efficient test suite generation in interactive t-way testing strategy," *Inf Softw Technol*, vol. 94, doi: 10.1016/j.infsof.2017.10.007.

[23] A. B. Nasser, A. R. A. Alsewari, N. M. Tairan, and K. Z. Zamli, 2017"Pairwise test data generation based on flower pollination algorithm," *Malaysian Journal of Computer Science*, vol. 30, no. 3, pp. 242–257, doi: 10.22452/mjcs.vol30no3.5.

[24] A. K. Alazzawi, H. M. Rais, and S. Basri, 2019"ABCVS: An artificial bee colony for generating variable t-way test sets," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 4, doi: 10.14569/ijacsa.2019.0100431.

[25] A. A. B. Homaid, A. A. Alsewari, A. K. Alazzawi, and K. Z. Zamli, 2018"A Kidney Algorithm for Pairwise Test Suite Generation," *Adv Sci Lett*, vol. 24, no. 10, pp. 7284–7289, doi: 10.1166/asl.2018.12929.

[26] K. Z. Zamli, F. Din, B. S. Ahmed, and M. Bures, 2018"A hybrid Q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem," *PLoS One*, vol. 13, no. 5, pp. 1–29, doi: 10.1371/journal.pone.0195675.

[27] J. Torres-Jimenez and J. C. Perez-Torres, 2019"A greedy algorithm to construct covering arrays using a graph representation," *Inf Sci (N Y)*, vol. 477, doi: 10.1016/j.ins.2018.10.048.

[28] M. I. Younis, A. R. A. Alsewari, N. Y. Khang, and K. Z. Zamli, 2020"CTJ: Input-output based relation combinatorial testing strategy using jaya algorithm," *Baghdad Science Journal*, vol. 17, no. 3, pp. 1002–1009, doi: 10.21123/BSJ.2020.17.3(SUPPL.).1002.

[29] J. M. Altmemi, R. R. Othman, and R. Ahmad, 2020"SCAVS: Implement Sine Cosine Algorithm for generating Variable t-way test suite," *IOP Conf Ser Mater Sci Eng*, vol. 917, no. 1, doi: 10.1088/1757-899X/917/1/012011.

[30] A. K. Alazzawi, H. M. Rais, and S. Basri, 2019"Parameters tuning of hybrid artificial bee colony search based strategy for t-way testing," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 5s.

[31] A. A. Muazu, A. S. Hashim, and A. Sarlan, 2022"Review of Nature Inspired Metaheuristic Algorithm Selection for Combinatorial t-Way Testing," *IEEE Access*, vol. 10, doi: 10.1109/ACCESS.2022.3157400.

[32] K. Z. Zamli, Y. A. Alsariera, A. B. Nasser, and A. Alsewari, 2015"On adopting parameter free optimization algorithms for combinatorial interaction testing," *ARPN Journal of Engineering and Applied Sciences*, vol. 10, no. 19.

[33] Z. Zhang, J. Yan, Y. Zhao, and J. Zhang, 2014"Generating combinatorial test suite using combinatorial optimization," *Journal of Systems and Software*, vol. 98, doi: 10.1016/j.jss.2014.09.001.

[34] N. Covic and B. Lacevic, 2020"Wingsuit Flying Search-A Novel Global Optimization Algorithm," *IEEE Access*, vol. 8, pp. 53883–53900, doi: 10.1109/ACCESS.2020.2981196.

[35] N. Covic and B. Lacevic, 2020"Wingsuit Flying Search-A Novel Global Optimization Algorithm," *IEEE Access*, vol. 8, pp. 53883–53900, doi: 10.1109/ACCESS.2020.2981196.

[36] A. R. A. Alsewari, H. C. Har, A. A. B. Homaid, A. B. Nasser, K. Z. Zamli, and N. M. Tairan, "Test cases minimization strategy based on flower pollination algorithm," in *Lecture Notes on Data Engineering and Communications Technologies*, vol. 5, 2018. doi: 10.1007/978-3-319-59427-9_53.

[37] A. B. Nasser, Y. A. Sariera, A. R. A. Alsewari, and K. Z. Zamli, 2015"Cuckoo search based pairwise strategy for combinatorial testing problem," *J Theor Appl Inf Technol*, vol. 82, no. 1.

[38] A. A. Hassan, S. Abdullah, K. Z. Zamli, and R. Razali, 2022"Whale Optimization Algorithm Strategies for Higher Interaction Strength T-Way Testing," *Computers, Materials and Continua*, vol. 73, no. 1, pp. 2057–2077, doi: 10.32604/cmc.2022.026310.

[39] K. Z. Zamli, F. Din, S. Baharom, and B. S. Ahmed, 2017"Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites," *Eng Appl Artif Intell*, vol. 59, no. September 2016, pp. 35–50, doi: 10.1016/j.engappai.2016.12.014.

[40] B. Ahmed, 2017"Generating Pairwise Combinatorial Interaction Test Suites Using Single Objective Dragonfly Optimisation Algorithm," *Journal of Zankoy Sulaimani - Part A,* vol. 19, no. 1, doi: 10.17656/jzs.10586.

[41] A. B. Nasser, A. Alsewari, and K. Z. Zamli, *Learning Cuckoo Search Strategy for t-way Test Generation*. Springer Singapore, 2018. doi: 10.1007/978-981-13-0755-3.

[42] Abdul Rahman A. Alsewari, 2012"A harmony search based pairwise sampling strategy for combinatorial testing," *International Journal of the Physical Sciences*, vol. 7, no. 7, pp. 1062–1072, doi: 10.5897/ijps11.1633.

Notes on contributors

Nurol Husna Che Rose is a PhD student in Computer Engineering at University Malaysia Perlis (UniMAP). Her research interests include software testing, optimization algorithms, and artificial intelligence. She is also a lecturer at UniMAP with nearly 5 years of teaching experience in software engineering field

Assoc. Prof. Dr. Rozmie Razif Bin Othman obtained his bachelor's degree in Electronics Engineering (Computer) from Multimedia University, Malaysia in 2006. Later in 2009, he finished his master's degree in Telecommunication Engineering from University of Malaya. Then, he completed his doctoral degree in Software Engineering in 2012. His doctoral thesis is on the design and development of combinatorial test suite generator. His areas of interest are software testing, optimization algorithm and machine learning. He has produces more than 50 articles in his field of interest and graduated more than 5 post graduate students. He also a certified professional for Requirement Engineering (CPRE), Certified Tester for both Foundation Level (CTFL) and Advance Level (CTAL-TM). Currently, he is an Associate Professor of Computer Engineering in Universiti Malaysia Perlis (UniMAP).

Ts. Dr. Hasneeza Liza Zakaria earned her Ph.D. in Software Engineering from Universiti Malaysia Pahang (UMP). She also holds a Master's degree in Computer Science from Universiti Teknologi Malaysia (UTM) and a Bachelor's degree in Information Technology from Universiti Utara Malaysia (UUM). Her doctoral research centered on the development of a Hybrid Optimization Algorithm using the Etilist method. Her research interests span software testing, optimization algorithms, and other optimization problems. Additionally, she is a certified professional technologist with the Malaysia Board of Technologists (MBOT). Currently, she serves as a lecturer at UniMAP.