PAIRWISE TEST DATA GENERATION BASED ON FLOWER POLLINATION ALGORITHM

Abdullah B. Nasser[1], AbdulRahman A. Alsewari[2], **Nasser M. Tairan[3]** *and Kamal Z. Zamli[4]*

[1,2,4]IBM Centre of Excellence, Faculty of Computer Systems and Software Engineering,
Universiti Malaysia Pahang, 26300 Kuantan, Pahang, Malaysia

[3]College of Computer Science, Department of Computer Science,
King Khalid University, Abha, Kingdom of Saudi Arabia

Email: pcc14003@stdmail.ump.edu.my[1], alsewari@ump.edu.my[2], nmtairan@kku.edu.sa[3],
kamalz@ump.edu.my[4]

ABSTRACT

Owing to an exponential increase in computational time associated with increasing number of system components, exhaustive testing is increasingly become impractical. Here, many researchers opt to adopt pairwise testing to minimize the overall number of tests. Recently, many existing works are focusing on the use of Search-Based algorithms as the basis of the implementation algorithm for pairwise test suite generation; however, there is no single strategy that can be the best for all cases. Currently, researches on Flower Pollination Algorithm (FPA) are very active and its applications have been proven successful to solve many optimization problems. This paper proposes a new search-based strategy for generating the pairwise test suite, called Pairwise Flower Strategy (PairFS). The main feature of PairFS is that it is the first pairwise strategy that adopts FPA as its core implementation. To evaluate and benchmark our proposed strategy, several existing comparative experiments are adopted from the literature. The results of the experiment show that PairFS in many cases are more efficient than the existing strategies in terms of the generated pairwise test suite size.

Keywords: ***Pairwise Testing, Flower Pollination Algorithm, Software Testing, Combinatorial Problem, Search Based Software Engineering.***

1.0 INTRODUCTION

Search Based Software Engineering (SBSE) is one of attractive fields in the past five years [1]. To be specific, SBSE has been successfully applied to address a wide range of software engineering problems on design, testing, software engineering management, requirements engineering and refactoring. SBSE involves applying optimization algorithms to solve software engineering problems [2, 3]. In the field of interaction testing, much recent works are focusing on searching (and minimizing) of test cases from a large potential of values based on defined interaction strength. Any software systems that have many of customizable options (e.g. Microsoft Word) can have an enormous number of input configurations need to be tested. Combinatorial Interaction Testing (CIT) is sampling technique that has been used extensively to test high-configuration software systems such that every t-combinations (where t indicates the interaction strength) of input values is covered by test case at least once [4].

As a special case of CIT is the pairwise testing. Pairwise testing is an effective technique that is based on the most software failures are often caused by interaction of two parameters [5]. In the literature, many studies have reported that pairwise testing is very effective to detect 2-way interaction faults. Kuhn, Wallace et al reports that on several systems [6], 76% of bugs can be detected by pairwise testing. A study conducted by Brownlie,

Prowse, and Phadke, researchers proposed a new method called Robust Testing™ [7]. The study uses AT&T's PMX/StarMAIL system as a case study. In this system, there are 1500 test cases need to be tested. However, due to time and resource constraints, only 1000 test cases can be performed. Robust Testing™ is applied to generate pairwise test cases resulting into reduction from 1000 to 422 test cases (i.e. potentially reducing more than halve of the testing efforts).

Finding a minimal pairwise test suite from potential large possible test parameter values is NP-hard (Non-deterministic Polynomial-time hard) problem. Many SBSE pairwise strategies have been proposed in an attempt to generate the smallest test suite size including Hill Climbing (HC) [8], Simulated Annealing (SA) [8], Genetic Algorithm (GA) [9], Ant Colony Optimization algorithm (ACO) [9], Particle Swarm Optimization (PSO) [10, 11], and Harmony Search (HS) [5]. Although useful, most of the previous studies are not without limitation. Strategies based on TS and SA are often given optimal results for small test configurations, but they are prone to get stuck in local minimum solution [4]. Strategies based on GA, ACA, PSO, and HS often require frequent interaction with the environments during computation. For instance, GA exploits the crossover and mutation operators with historical information to explore regions of better solutions. ACA requires indirect communication of a colony via pheromone trails. In a similar manner, PSO interacts with individual particles through velocity updates in the given swarm until the solution is reached. HSS requires the use probabilistic value from Pitch Adjustment Rate (PAR) and Harmony Memory Considering Rate (HMCR) to select the solution from Harmony Memory (HM) or regenerate newly random solution.

Compared to existing strategies adopting SBSE based approach, FPA mimics more straightforward nature abstraction of the (local and global) flower pollination process. In this manner, the learning curve for FPA is rather low. Unlike existing SBSE, FPA has two search capabilities: global pollination and flower consistency. Global pollination using Lévy flight allows FPA to jump for a long distance out of local optimum and thus can explore a larger search space very efficiently. Flower consistency reproduces a new solution from similarity of two flowers, thus maximizing the reproduction of the same flower species and guarantees the convergence speed more quickly. Global pollination via Lévy flight combined with Flower consistency make FPA explores the search space more efficiently. Moreover, currently researches on FPA are very active and its applications have been proven successes to solve many problems such as, image classification problem [12], image compression problem [13], and Wireless Sensor Network [14], to name a few. Many existing studies show that FPA is very efficient and outperform most of existing algorithms.

Arguably, no single strategies are capable to obtain the most minimal test suite in all cases at hand. Complementing existing works and focusing on the SBSE based approach; this paper proposes a new pairwise strategy, called Pairwise Flower Pollination Strategy (PairFS). The main contribution of PairFS is that it is the first pairwise strategy that adopts FPA as its core implementation.

The rest of this paper is organized as follows. Section 2 gives an overview of pairwise testing. Related works are stated in Section 3. Detailed reviews of SBSE pairwise strategies are provided in Section 4. Section 5 demonstrates Flower Pollination Algorithm. Section 6 introduces the proposed strategy, PairFS. Section 7 highlights the experimental results and discussion. Section 8 gives threats to validity for the experiments. Lastly, Section 9 gives the conclusion and future work.

2.0 BACKGROUND

In general, any system consists of number of components, which interact with each other (termed parameters with their associated values). Practically, testing all combinations is expensive. CIT minimizes tests by generating on the desirable interaction for every *t*-combination. In many classes of systems, faults are often related to 2-way interaction, thus, it is worthwhile to focus on pairwise testing.

Definition (Pairwise testing): Given a set of N parameters $P_1, P2, P3, ..., P_N$, each parameter having V_i possible values $\{v_1,v_2,...v_m\}$. A set of test data values Tc, contains N test values, is selected for each of the parameter values such that the test cases in Tc cover all pairs of input parameter values once. To illustrate the concept of pairwise testing, consider a Simple Find Dialog example as given in the following Fig. 1:

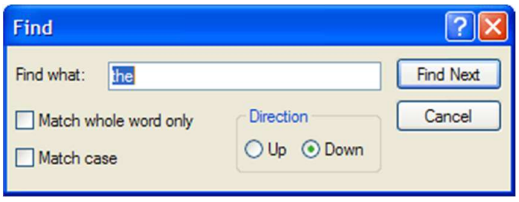

Fig. 1: Simple find dialog example

The dialog box treats an input text string and returns all records that match that string. This dialog box consists of four inputs (i.e. Text string, Match whole word, case sensitivity and direction of the search). By ignoring illegal values for simplicity; the values for each parameter can be summarized as the following: Text string: uppercase/lowercase/mixed, Match whole word: match/do not match, Case: match/do not match and Direction: Up/Down. Therefore, in order to test this system exhaustively, there are 24 test cases need to be considered. By using pairwise testing, all possible combinations of each pair of input parameters need to be tested at least one time. To generate a pairwise test suite, in our example, there are 30 pairs need to be covered, called interaction pairs. By using greedy algorithm, the first test case can cover 6 interaction pairs. Prudently selecting the test cases, all the 24 interaction pairs can minimize into 6 test cases. From optimization perspective, the pairwise test suite generation problem can be stated mathematically as follows:

$$\text{Maximize } f(x) = \sum_{1}^{N} x_i \qquad (1)$$

$$\text{subject to } x \in x_i \ , i = 1,2, \ldots . , N \qquad (2)$$

where, $f(x)$ is an objective function to be optimized that capture the weight of the test case in terms of the number of covered pairwise interactions, x_i are the decision variables that is, $x_i = \{x_i(1), x_i(2), \ldots, x_i(K)\}$ for discrete decision variables ($x_i(1) < x_i(2) < \ldots < x_i(K)$); N is the number of decision parameters; and K is the number of possible values for the discrete variables.

The interesting property of pairwise test cases, as Fig. 2 demonstrates, is that any two columns contain all possible pairs of these two columns, occur somewhere in any order. For example, as we see in columns A, and B contain all possible pairs of AB combination (i.e. 00, 01, 10, 11, 20, and 21), as well, C and D columns contain all possible pairs of CD combination (i.e. 00, 01, 10, 11).

A	B	C	D
0	0	1	1
0	1	0	0
1	0	1	0
1	1	0	1
2	1	1	0
2	0	0	1

Fig. 2: Pairwise test suite generation for find dialog example
only pairs of AB, AC, and CD are highlighted

3.0 RELATED WORKS

This section provides an overview of the existing works for constructing a pairwise test suite. Based on [10], the existing approaches can be classified into two main categories: algebraic construction, computational construction. Considering algebraic approach, test data sets are constructed without enumerating any combinations. Thus, this approach requires lightweight computations. There are two types of algebraic approach. The computation in the first type of algebraic approach is based on mathematical functions [15-17]. The second type of algebraic approach employs a recursive process to construct test sets by construct a large test sets from small test sets [18]. Strategies adopting this approach (such as CA, MCA and TConfig) are often

restricted to small configurations. Computational approach uses a greedy algorithm to construct the test cases. Each step tries to cover as many combinations as possible uncovered combinations. In Computational Approach, generating test set is accomplished by either using one-test-at-a-time strategy (OTAT) or one-parameter-at-a-time strategy (OPAT).

OTAT strategies start to build one complete test case per iteration and checks if this test case is the best test case to cover the most uncovered interaction or not, this procedure is repeated until all the combinations are covered. In the literature, there are many strategies and tools have developed uses OTAT techniques such as AETG [19], TConfig [20], Jenny [21], and WHITCH [22]. One-parameter-at-a-time (OPAT) strategy starts by building a completed test suite for the first two parameters, or the smallest number of components, then extends horizontal by adding one parameter per iteration, and sometimes, extends vertically until all the parameters is covered, such as an IPO [23] and its improvement (i.e. IPOG [24], IPOG-D [25], IPOF and IPAD2 [26]). Much recent works on test suite generation is adoption search based algorithms in software testing, which called Search Based Software Engineering Testing (SBSE).

4.0 SEARCH BASED SOFTWARE ENGINEERING TESTING

This section explores an overview of Search Based Software Engineering Testing (SBST) strategies. SBST strategies are one of the most emerging technologies for the last 20 years. SBST can be classified into two categories namely heuristic search algorithms and meta-heuristic search algorithms. Heuristic search is a way to find an approximate and reasonable solution but without guarantee to give an optimal solution such as Hill Climbing (HC). Meta-heuristic as term is first introduced in Tabu Search algorithm by Glover [27]. Meta-heuristic algorithms are a top level of heuristic search algorithms characterized by attempt escape from local optima. Meta-heuristic algorithms have been used successfully in software testing problems such as Simulated Annealing, Genetic algorithm, Ant Colony Optimization algorithm, Particle Swarm Optimization, and Harmony Search, to name a few [5, 8, 10, 11, 19, 28].

4.1 Hill Climbing

Cohen (2009) adopts a Hill Climbing (HC) algorithm to generate a pairwise test suite [8]. HC algorithm is a heuristic search algorithm which can be considered as the fundamental pillar of local search. The algorithm begins from a random feasible test case x and then generate transformation test case x' from the x, if x' is better than current test case, x' is accepted as the new solution, if it is not better than current solution, we check another transformation neighbor. This process is repeated until all the pair combinations are covered. A study conducted by Cohen shows that HC often fails to find optimal test suite size. This study suggests to repeat the algorithm many times, each time with different initial configurations to increase the chance of finding a good solution [8]. The key advantage of hill climbing is only need a limited amount of memory [29].

4.2 Simulated Annealing

Simulated Annealing (SA) has been implemented to construct the pairwise test suite by Cohen, 2004, and Patil and Nikumbh, 2012 [8, 30]. SA-based is a similar to HC, but SA allows wrongs movements to poor solution or test case, with an acceptable probability to avoid getting stuck in a local optimum solution. The acceptance probability function $P(e, e', T)$ decides to move to the new test case x′ or staying in current test case x (where $e= E(x)$ is the energy of current test case x, $e'= E(x')$ is new candidate test case x, and T controlling temperature) [31]. The algorithm used randomized searches to generating a pairwise test suite by finding the best test case per iteration until cover all interaction pairs. HC and SA can be considered as single-solution based strategies, where finding a solution starts from one position in the search space and then move to new solution [8].

4.3 Genetic Algorithm

Another meta-heuristic search technique that has been used to generate pairwise test data is Genetic algorithm (GA) (i.e. Shiba, Tsuchiya et al (2004) and Qi, Wang et al. (2015)). GA is based on AETG strategy [19]. GA-based strategy is considered the early works in adopting population-based search algorithms to generate a

pairwise testing strategy [9, 32, 33]. GA exploits the idea of genetic evolution by selecting and combining the best two chromosomes to ensure that only the best solutions are carried to next generations. For constructing test suite using GA, randomly a set of feasible chromosomes (or test cases) are generated. Each test case represents as a chromosome and the fitness for chromosome x is the number of pair combinations that are covered by x. In evaluation loop, the algorithm employs three processes: (1) Selection: GA selects two chromosomes randomly and then a copy of the winner is deposited mating pool. (2) Crossover: is taking two chromosomes and produce a child by exchanges between the two chromosomes values with probability. (3) Mutation process replaces the value of chromosomes randomly. Hereby, based on the fitness function the best chromosomes are selected at each generation and survive to the next generation until exceeds the maximum number generation, and then the best chromosomes or test case added to the final test suite. Recent work on GA for pairwise testing suite generation explore the uses of parallelize GA with Spark [33].

4.4 Ant Colony algorithm

Ant Colony Optimization algorithm (ACO) mimics the behavior of colonies of ants for finding food paths. The places of food represent the parameter and the food represents the value of the parameter, and each test case represents the quality of the paths to the food. The paths to the food are evaluated based on the quantity of pheromones which is reinforced by the ants. Over time, the density gets higher for some paths which are chosen by many ants. By comparison, the best path is selected to be added to the final test suite. The search process in ACO allows each ant to build a part of the solution or establishes a complete solution [9, 32]. Another important issue, among SBSE strategies, both of GA and ACO include some expensive computations such as crossover and mutation operations in GA and ant search process in ACO. Thereby both of GA and ACO address only small configurations [34].

4.5 Particle Swarm Optimization

Particle Swarm Optimization (PSO) algorithm has been implemented for pairwise test suite generation using two different two different based approaches OTAT and OPAT [10, 11]. The discrete version of PSO is adopted in Particle Swarm-based Test Generator (PSTG) strategy [11]. Each test case represents Particle. The algorithm starts by initializing a set of particle swarms, and interaction elements which presents the search space, and initializes the velocity of each particle Vi. During the procedure of test suite generation, the velocities are updated according to the best test case when the particle moves around the search space, and then make a move from current test case to the new test case based on the velocity. The particle continues its motion until termination criteria is met the test case is selected to be added to final test suite.

4.6 Harmony Search

Much recent work undertaken in this field, As the name suggests, harmony search has been adopted for harmony search algorithm-based strategy (PHSS) to implement and generate pairwise tests suite. PHSS is pairwise test data generation. Using PHSS, the test data generation process is mimicking the improvisation process by a skilled musician [5]. PHSS passes through many phases.

The core part of the PHSS is updating current solution process to cover more interaction elements. To this end, PHSS uses harmony search algorithm. Updating process, (also known as improvise new harmony) updates the new test case based on the value of harmony memory considering rate (HMCR). Based on HMCR chose the new value either from Harmony memory (HM) or randomly. If the new value is coming from HM, Based on pitch adjusting rate (PAR), HSS can choose to make a small change or not. Unlike PSTG, number of parameters in PHSS is less PSTG. PHSS requires harmony memory size, Iteration, pitch adjustment rate and harmony memory consideration rate as its parameter [4].

Summing up, our related work section highlights and analysis the strengths and the limitations for each SBSE-based strategy. At the end of each subsection of related work, our analysis focused on the search procedure of each strategy and parameter setting requirements in each strategy, as well as the tuning issues and how each algorithm balances between local intensification and global diversification. Table 1 gives a summary of the description and comparison of existing pairwise strategies. From the table, it can be seen that most of existing

SBSE-based strategies do not sufficiently address high configuration systems due to its high computational complexity. Based on some studies [2, 5, 34], strategies based on GA, ACO and PSO consider as expensive computation strategies due to the frequent communication needs with the environments during computation. Thereby, both of GA and ACO address only small configuration systems, whereas, other algorithms such HC, SA and HS based only on transformations. Most of existing pairwise testing requires the tuning of many parameters which add extra effort except HC. Here, it is worth mentioning HC has low computation and free parameters, however, it often fails to find optimal test suite size because it is single-based algorithm.

Table 1: The Summary and Limitation of the Existing Strategies

Year	Strategy	Non-deterministic	Complexity of Computation	Support High configurations	Population	Control parameters
2004	HC	√	Low	×	Single-Population	Free parameters
2001	SA	√	Low	×	Single-Population	3 Parameters
1995	GA	√	High	×	Population-based	2 Parameters
2008	ACA	√	High	×	Population-based	4 Parameters
2010	PSTG	√	High	×	Population-based	3 Parameters
2012	PHSS	√	Low	√	Population-based	3 Parameters

5.0 FLOWER POLLINATION ALGORITHM

Flower pollination algorithm (FPA) is a new met heuristic algorithm developed by Xin-She Yang [35]. FPA inspired by the pollination behavior of flowering plants. Flower pollination is the process of transferring pollen grains from male part of flower to female part via pollinators such as birds, bees, butterflies, bats and other animals. There are two mechanisms of pollen transfer: biotic and abiotic. Biotic pollination refers to transfer pollen via pollinators such insects or animals, while abiotic does not require any pollinators to transfer pollen. As well, pollen can be transferred from male parts to female parts in the same flower (self-pollination) or transfer to another flower (cross-pollination) as Figure 3 shows. Another point of interest in flowering plants, Some flowers help and restrict a specific pollinators to pollinate them [36, 37].

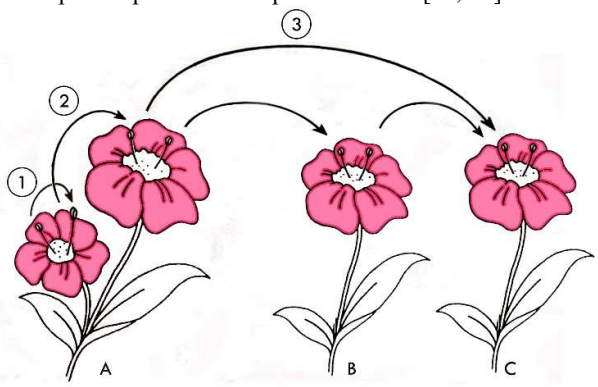

Fig. 3: Flower Pollination Methods, (1) Self Pollination in the same flower, (2) Self -Pollination from same plant but different flower, and (3) Cross-Pollination from different plant.

Based on the pollination characteristics of flowering plants described above, a new meta-heuristic algorithm, called Flower Pollination Algorithm (FPA), is proposed. For simplicity purposes, in FPA the above description is idealized as the following four rules [35]:

Rule 1. Biotic and cross-pollination can be considered as a process of global pollination process, and pollen-carrying pollinators move in a way that obeys Lévy flights.
Rule 2. For local pollination, a biotic and self-pollination are used.

Rule 3. Pollinators such as insects can develop flower constancy, which is equivalent to a reproduction probability that is proportional to the similarity of two flowers involved.
Rule 4. The interaction or switching of local pollination and global pollination can be controlled by a switch probability p ϵ [0, 1], with a slight bias toward local pollination.

Based on the above rules, FPA can be represented mathematically as two core parts: Global Pollination step and Local Pollination step. In global pollination, the flower pollens are transferred by pollinators such as insects, over a long distance. This guarantee the fittest pollens with high quality will carry over to the next reproduction. Hence, Rule 1 and flower constancy can be represented as followings:

$$x_i^{(t+1)} = x_i^{(t)} + \gamma L\left(x_i^{(t)} - gbest\right) \quad (3)$$

where $x_i^{(t)}$ the current pollen, *gbest* is the current best, γ>0 is the step size and *L* is Lévy flight. In FPA, Lévy flight mimics the characteristic of long-distance movement of the pollinators. The Lévy flight essentially is a random walks interspersed by long jumps which are distributed according to a power law to different regions.

The second core part of FPA, Local pollination is represented in a biotic and self-pollination, can formulated by following equation:

$$x_i^{(t+1)} = x_i^{(t)} + \epsilon(x_j^{(t)} - x_k^{(t)}) \quad (4)$$

Equation 4 mimics the characteristic of self-pollination and abiotic pollination where $x_j^{(t)}$ and $x_k^{(t)}$ two selected pollens from different flowers, ϵ is random number obey to uniform distribution in [0, 1].

In real-world, plants can employ self-pollination where pollen's flower can successfully pollinate the same flower or another flower in the same plant, or pollinate another flower in different plants; therefore, a switch condition *pa* can be used to alternate between common global pollination and intensive local pollination. Based on those four rules plus, FPA can be summarized as shown in Fig. 4.

Objective function f(x), x = (x_1, ..., x_d)
Initialize a population of n flowers/pollen gametes with random solutions
Find the best solution gbest in the initial population
Define a switch probability $pa \in [0, 1]$
While ($t < MaxGeneration$)
  For i = 1 : n (all n flowers in the population)
    If (rand < pa)
      Draw a (d-dimensional) step vector L which obeys Lévy distribution
      Global pollination via $x_t^{i+1} = x_t^i + L(x_t^i - g^{best})$
    Else
      Draw ϱ from a uniform distribution in [0,1]
      Randomly choose *j* and *k* among all the solutions
      Do local pollination via $x_t^{i+1} = x_t^i + \varrho(x_t^j - x_t^k)$
    End if
    Evaluate new solutions
    If new solutions are better, update them in the population
  End for
  Find the current best solution $gbest$
  End while
End-procedure

Fig. 4: Pseudo code of Flower Pollination Algorithm [35]

6.0 PAIRWISE FLOWER STRATEGY

In this research, a new strategy, called Pairwise Flower Strategy (PairFS), is proposed for pairwise test suite generation. FPA is adopted in the proposed strategy to generate an optimal pairwise test suite. In PairFS, each pollen or flower represents one test case, and flower's fitness is the number of interaction pairs that can be cove

by the test case. The proposed strategy, as Fig. 5 shows, is a composition of three main steps: (A) Input Analysis step, (B) Generating Interaction Pairs step, and (C) Test Suite Generation step by performing a search using FPA to find a set of optimal test cases.

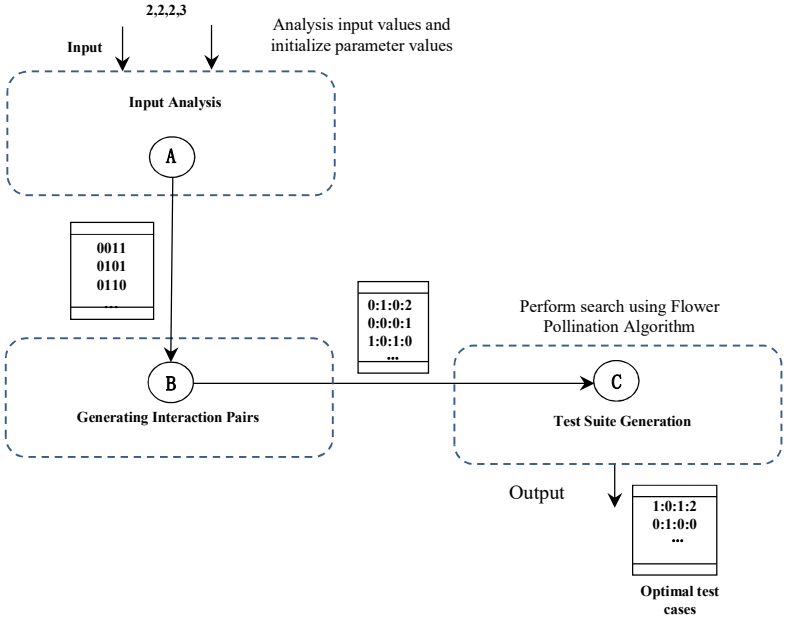

Fig. 5: Graphical representation of PairFS steps

In Input Analysis step, PairFS receives the inputs values (i.e. a set of parameters *P* and parameters values *V*) and initialize the values of population size *pollen_size*, switch probability *pa,* and stopping criteria. Next, PairFS generates all possible binary combinations of P-digit that contain only two 1's. For our running example (i.e. Find Dialog Example stated in Section 1.1), all possible binary combinations for a given set of parameters *P* (i.e. P1, P2, P3, and P4) are 1100, 1010, 1001, 0110, and 0101. Based on binary combinations of the parameters and its values, all Interaction Pairs are generated and added to Interaction Pairs List (PairList). For our example, P1, P2, and P3 having two values (i.e., 0 and 1), and P4 has three values (0, 1, and 2). For instance, for the binary combination 1100 (i.e. refers to P1, P2) has 2×2 possible interaction elements (i.e., 0:0, 0:1, 1:0, and 1:1), while 1001 (i.e. refers to P1, P4) has 2×3 possible interaction elements (i.e., 0:0, 0:1, 1:0, 1:1, 2:0, and 2:1).

The third step of PairFS is performing search using FPA to generate the pairwise test suite that cover all interaction pairs. In this step population of flower is generated randomly. Because of our strategy follows OTAT technics, which is more suitable for treatment as an optimization problem, at each cycle of iteration of the strategy, we find one optimal flower (or test case) that cover the maximum number of interaction pairs as Equation 1 illustrates. In general, there are two core operations performed on the population of pollen. The first core part of the strategy is generating new pollen using global pollination or Lévy flight Equation 3 showed. The second part is generating new pollen using local pollination. In local pollination two pollens are selected randomly from different flowers to generate new pollen as Equation 4 showed. Finally, the optimal pollen or test case is added to final test suite until all interaction pairs are covered. Fig. 6 describes the pseudo code of Pairwise Flower Strategy.

```
Input:
        P: Features number n, and
        V:  set of values for each feature V = [v0 ..vj];
Output:
        Final test suite list FTS;
Begin
 Let FTS be a set of candidate tests;
 Generate all possible Interactions Pairs List (PairList) based on P and V
 Generate initial population of pollens randomly
 while PairList is not empty do
   while t <MaxGeneration  or  stop criterion do
      for i = 1 : n (all n pollens in the population)
           if  ( rand < pa )
              Draw a (d-dimensional) step vector L which obeys Lévy distribution
              Global pollination via x_t^{i+1} = x_t^i + L(x_t^i − gbest)
           Else
              Draw ϱ from a uniform distribution in [0,1]
              Randomly choose j and k among all the solutions
              Do local pollination via x_t^{i+1} = x_t^i + ϱ(x_t^j − x_t^k)
           End if
           Evaluate new solutions
           If new solutions are better, update them in the population
      End for
      Find the current best solution gbest
   End while
   Add the best test case, gbest, into FTS.
   Remove covered interactions elements from PairList.
 End while
End-Procedure
```

Fig. 6 : Pairwise flower strategy pseudo code

7.0 EVALUATION OF PAIRFS

The main purpose of this paper is to propose an effective pairwise testing strategy to generate optimum test suite by minimizing the test cases. To this end, this section aims to evaluate the efficiency of the proposed strategy to generate the smallest test suite size and assess the statistical significance of the generated test suite size. Thus, this section is divided into two subsections. The first subsection is to evaluate the efficiency of the proposed strategy giants other exiting strategies by comparing the proposed strategy with other existing pairwise strategies. The second subsection deals with the statistical analysis based on the Wilcoxon signed-rank test of all experimental results conducted in this research.

7.1 Comparison of PairFS with other Strategies

In order to evaluate the efficiency of proposed strategy in term of test suite size, which is a main criterion for test suite generation, comparisons of proposed strategy against existing computational strategies and search-based strategies are made.

Preparing for the experiments, a detailed parametric study for PairFS's parameters is conducted by using different values for switch probability *pa* (0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9), iteration size (10, 20, 50, 100, 300, 500 and 1000) and population size *pollen_size* (10, 20, 50, 100, 500 and 1000), carried out that the optimal results are obtained when *Pa* = 0.8, iteration size = 500 iterations, and population size 30 pollens. Therefore, in our experiments, we have used these values as initial values for PairFS's parameters. Several existing comparative experiments [5, 10, 11] are adopted in our experiments. All experiments were employed

within the environment consists of a desktop PC with Windows 7 professional and 32-bit Operating System, Intel (R) core™ i7-3770 CPU @ 3.40 GHz, and 4GB of RAM.

The results of the experiments are reported in Tables 2-4. Each column represents the results of one strategy and each cell in these tables represents the smallest test suite size obtained by corresponding strategy. Regarding table rows, each table rows are varied from table to table according to the objective of each experiment. Here, we have performed three different experiments.

- First experiment, PairFS is compared with published results, which involve 11 different well-known system configurations. C1 through C11 indicate different system configurations as stated in the second column Table 1. For example, C1 = 3^3 represent a system with 3 parameters each parameter with 3 values, while C2 = 3^{13} refer to a system with 13 parameters each parameter with 3 values.
- Second experiment, we adopt a system configuration with 2-valued parameters and P varied From 3 to 12.
- Third experiment takes with 10 V-valued parameters, where V varied from 3 to 10.

We run each system configuration for 20 times, and report only the best test size. In all tables, one cell in each row is marked with * to show the best results obtained for the test configurations. The cell marked as "NA" indicates that the results are not available in the literature.

Table 2 : Comparison with existing strategies using 11 different systems configurations

Configurations		Computational Strategies												Search-based Strategies					
		PICT	TVG	AETG	mAETG	TConfig	CTE-XL	AllPairs	Jenny	IPO	IRPS	IPOG	G2Way	SA	GA	ACA	PPSTG	PHSS	PairFS
C1	3^3	10	11	NA	NA	10	10	9*	10	NA	9*	11	10	NA	NA	NA	9*	9*	9*
C2	3^4	13	12	9*	11	10	10	13	10	9	9*	12	10	9*	9*	9*	9*	9*	9*
C3	3^{13}	20	20	15*	17	20	21	20	22	17	17	20	19	16	17	17	17	18	18
C4	10^{10}	170	189	NA	NA	170	192	157	177	169	149*	176	160	NA	157	159	170	155	150
C5	15^{10}	NA	473	NA	NA	NA	NA	336	390	361	321*	373	343	NA	NA	NA	NA	341	344
C6	10^{20}	NA	NA	180	198	NA	NA	NA	230	212	210	NA	200	183*	227	225	NA	224	205
C7	5^{10}	47	50	NA	NA	48	50	45	49	47	45	50	46	NA	NA	NA	45	43	42*
C8	$5^1 3^8 2^1$	21	23	19	20	22	21	41	21	NA	17	19	23	15*	15*	16	21	20	20
C9	$6^1 5^1 4^6 3^8 2^3$	38	41	34	35	33	39	31*	NA	NA	NA	36	NA	NA	33	32	39	39	37
C10	$7^1 6^1 5^1 4^6 3^8 2^3$	46	52	45	44	49	53	51	NA	NA	NA	44	NA	NA	42*	42*	49	48	48
C11	$10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1$	101	100	NA	NA	92	102	98	NA	NA	NA	91*	NA	NA	NA	NA	97	95	92

y^x means: means that task take x parameters, each parameter with y values.

Table 3 : Comparison with existing strategies, with v = 2 and p varied from 3 to 12

P	Parameter value (V) = 2								
	TVG	CTE_XL	PICT	TConfig	Jenny	IPOG	PPSTG	PHSS	PairFS
3	4*	6	4*	4*	5	4*	4*	4*	4*
4	6	6	5*	6	6	6	6	6	6
5	6*	6*	7	6*	7	6*	6*	6*	6*
6	6*	8	6*	7	8	8	7	7	7
7	8	8	7	9	8	8	7*	7*	7*
8	8	7	8	9	8*	8*	8*	8*	8*
9	8	9	9	9	8*	8*	8*	8*	8*
10	9	9	9	9	10	10	8*	8*	8*
11	9	10	9	10	9	10	9	8*	8*
12	10	10	9*	*9	10	10	9*	9*	9*

Table 4 : Comparison with existing strategies, with p = 10 and v varied from 3 to 10

V	Parameter Size (P) = 10								
	TVG	CTE_XL	PICT	TConfig	Jenny	IPOG	PPSTG	PHSS	PairFS
3	18	18	18	17	19	20	17	17	16*
4	33	33	31	31	30	31	29	28*	28*
5	50	50	47	48	45	50	45	43	42*
6	72	71	66	64	62	68	62	60*	60*
7	98	97	88	85	83	90	81	79*	79*
8	124	125	112	114	104	117	109	105	101*
9	152	161	139	139	129	142	139	127	126*
10	189	192	170	170	157	176	170	155*	155*

At first glance, as Tables 2, 3, and 4 shows that most of SB strategies perform similarly. We also can observe that computational strategies perform better than search-based strategies in many cases but the scale configurations considered has been small. Taking a closer look at existing strategies, we observe that mAETG, AETG, ACA, SA and GA perform better that other strategies due to their randomization.

Referring to Table 2, most of existing strategies including the proposed strategy produces the most minimum test size for C1, and C2. In the case of C3, AETG outperforms all other strategies while IRPS outperforms all other strategies in case of C4 and C5. In the cases of C7 the proposed strategy outperforms all other strategies, while, in the cases of C4, C5 and C11 the proposed strategy produces the second best test size among the existing strategies. In the other cases, proposed strategy gives competitive results as compared to other existing strategies. Putting computational strategies aside, SA appears to be better than other search-based strategies but limited to small configurations. The performance of GA and ACA is almost the same in most of cases, which still better than PairFS, however GA, and ACA, as well PSO, require expensive computation owing to the need for the frequent communication with the environments during the searching process [2]. Comparing with PPSTG and PHS strategies, PairFS produces better test suite size.

Tables 3 and 4 show that PairFS obtains the smallest test suite in most case, only PICT has managed to outperform PairFS , where the test suite size produces by PairFS is equal to 7 test cases while the test suite size produces by PICT is 6 test cases, that is, in the case of p = 6. PairFS appears to generate the most optimum results in most of the configurations as marked with (*) owing to the good balance between global search and local search through the adoption of lévy flight.

7.2 Statistical Analysis

Further analysis of the experimental results, Wilcoxon Signed Rank Test technique is performed. The Wilcoxon test is a non-parametric analysis technique that can be used to compare two sets of ordinal data that are subject to different conditions. In this statistic analysis, the PairFS strategy will be compared to each individual strategy separately; to test if there is a significant difference between the proposed strategy results and other strategies results. Here, we have two hypotheses null hypothesis (H_0) and alternative hypothesis (H_1) as following:

$$H_0: \mu 1 - \mu 2 = 0 \qquad (there\ is\ no\ difference\ between\ two\ strategies' results)$$
$$H_1: \mu 1 - \mu 2 \neq 0 \qquad (there\ is\ difference\ between\ two\ strategies' results)$$

From the experiments results, Wilcoxon test statistic is calculated. Then the test statistic is converted into a conditional probability called a P-value. A small P-value means that it is strong evidence to reject the null hypothesis H_0 (i.e. there is no difference between two strategies' results) in favor the alternative hypothesis. Decision making is based on a probability threshold called Alpha (α) or significance level.

Table 5: Wilcoxon signed rank test using SPSS for experiments results from table 2

Categories	Pairs	Ranks				Asymp. Sig. (2-tailed)	Conclusion
		Negative Ranks	Positive Ranks	Ties	Total		
Computational Strategies	PICT- PairFS	1	7	0	8	0.029	*Reject the null hypothesis* H_0
	TVG- PairFS	2	3	1	6	0.446	Retain the null hypothesis H_0
	AETG- PairFS	5	0	1	6	0.020	*Reject the null hypothesis* H_0
	mAETG- PairFS	4	1	1	6	0.088	Retain the null hypothesis H_0
	TConfig- PairFS	1	7	0	8	0.045	*Reject the null hypothesis* H_0
	CTE-XL- PairFS	0	8	0	8	0.006	*Reject the null hypothesis* H_0
	AllPairs- PairFS	2	6	1	9	0.241	Retain the null hypothesis H_0
	Jenny- PairFS	0	8	0	8	0.006	*Reject the null hypothesis* H_0
	IPO- PairFS	1	3	2	6	0.072	Retain the null hypothesis H_0
	IRPS- PairFS	5	1	2	8	0.071	Retain the null hypothesis H_0
	IPOG- PairFS	3	6	0	9	0.055	Retain the null hypothesis H_0
	G2Way- PairFS	2	6	0	8	0.144	Retain the null hypothesis H_0
Search-Based Strategies	SA- PairFS	3	0	2	5	0.054	Retain the null hypothesis H_0
	GA- PairFS	4	2	1	7	0.300	Retain the null hypothesis H_0
	ACA- PairFS	0	0	7	7	0.500	Retain the null hypothesis H_0
	PPSTG- PairFS	0	0	8	8	0.500	Retain the null hypothesis H_0
	PHSS- PairFS	2	3	5	10	0.393	Retain the null hypothesis H_0

Table 6: Wilcoxon signed rank test using SPSS for experiments results from table 3 and 4

Categories	Pairs	Ranks				Asymp. Sig. (2-tailed)	Conclusion
		Negative Ranks	Positive Ranks	Ties	Total		
Computational and Search-Based Strategies	TVG - PairFS	1	12	5	18	.003	*Reject the null hypothesis* H_0
	CTE_XL - PairFS	1	15	2	18	.001	*Reject the null hypothesis* H_0
	PICT - PairFS	2	12	4	18	.004	*Reject the null hypothesis* H_0
	TConfig - PairFS	0	13	5	18	.001	*Reject the null hypothesis* H_0
	Jenny - PairFS	0	15	3	18	.001	*Reject the null hypothesis* H_0
	IPOG - PairFS	0	13	5	18	.001	*Reject the null hypothesis* H_0
	PPSTG - PairFS	0	9	9	18	.003	*Reject the null hypothesis* H_0
	PHSS - PairFS	0	4	14	18	.059	Retain the null hypothesis H_0

The statistics in Tables 5 and 6 gives the values of the Wilcoxon Signed Rank Test for PairFS compared with each strategy of our experiments. As the tables show, the Wilcoxon signed-rank test gives negative ranks (i.e.

number of cases that PairFS unable to outperform another strategy), and positive ranks (i.e. number of cases that PairFS is better than another strategy), along with ties, which is the total number of observations. The column labeled Asymp. Sig. (2-tailed) shows p-value probability; if p-value less than 0.005, as recommended in [38], there is no significant difference between the compared results.

Referring to Table 5 (i.e. showing Wilcoxon signed-rank test using SPSS for experimental results of Table 2), the p-value show that there is no significant differences between the results obtained by PiarFS and, TVG, mAETG, AllPairs, Jenny, IPO, IRPS,IPOG, G2Way, SA, GA, ACA, PPSTG, and PHSS. Statistically significant improvement was seen only when compare PairFS with PICT, AETG TConfig, and CTE-XL. However, even though the statistical analysis shows no significant improvement in the test suite size in most of cases, ranks column can figure out the rank of the proposed strategy. Ranks column show that the positive ranks of PairFS are higher than negative ranks.

Considering Table 5 I.e. the statistical analysis for the performance of PairFS for experiments results of Table 3 and 4), the null hypothesis H_0 is rejected in most of cases with the exceptions when compared PairFS with PHSS, which proves that PairFS gives favourable results and better test suite size than other strategies in table 3 and 4.

After having considered above results, we can summarize that although PairFS proved its efficiency in generating better test suites sizes in most cases, there are no single strategy can overcome all other strategies and there are some positive and some negative capabilities for each strategy. Therefore, finding a general strategy for the construction of optimum pairwise test suite generation is difficult and still an open problem. Secondly, we observe that the computational strategies tend to outperform search-based strategies but search-based strategies are more effective in dealing with high configuration systems.

8.0 THREATS TO VALIDITY

In this study, empirical experiments face different threats to validity. Here, we identified a number of threats to validity that may our experiments suffer from. First, the code we have used may not be representative of other strategies' code as we may have used language specific functions for our data structure implementation (e.g. array lists). Specifically, we have adopted the Java programming language on the Netbean platform. The way to address this threat would be to repeat the experiments using other code implementation and adopting more than one implementation platform. Second, the adopted benchmark is considered a set of toy example. For further evaluation and determining the strengths and weaknesses of PairFS, we need more experiments that are representative of real world case studies. Finally, all of non-deterministic strategies is often affected by chances. Therefore, iterations size may affect the findings of the strategy as higher iteration gives higher chance to get the optimum results. In such a case, taking an average values instead of the best values can be more representative of the performance of each strategy. In the absence of the average values, our work takes the best values for comparison.

9.0 CONCLUSION AND FURTHER WORK

Pairwise testing is an indispensable technique to reduce the size of test cases for saving time and effort. A number of pairwise strategies based on search algorithms have been proposed such as Hill Climbing, Simulated Annealing, Genetic algorithm, Ant Colony Algorithm, Particle Swarm Optimization, and Harmony Search. This paper proposes a new pairwise strategy based on Flower Pollination Algorithm (FPA), called Pairwise Flower Strategy (PairFS). Our results are encouraging. As far as the scope for future work, there are potentially two directions considering the effectiveness of SBSE for pairwise test generation. The first direction is to adopt hybrid search algorithm as an ensemble of two or more search-based algorithms. The second direction relates to the adoption of hyper-heuristic algorithms [39, 40] to choose a particular heuristic for execution adaptively during run-time.

ACKNOWLEDGMENT

The work undertaken in this paper is partially supported by the Fundamental Research Grant: Reinforcement Learning Sine Cosine based Strategy for Combinatorial Test Suite Generation (grant no: RDU170103) from the Ministry of Higher Education Malaysia and the research grant: Efficient Hybrid Metaheuristic Algorithms for Solving Optimization Problems. (grant no: R.G.P. 2/7/38) from the King Khaled University, Saudi Arabia. We thank both of our grants sponsors.

REFERENCES

[1] M. Harman, Y. Jia, J. Krinke, W. Langdon, J. Petke, and Y. Zhang, "Search based software engineering for software product line engineering: A survey and directions for future work," in *18th International Software Product Line Conference*, 2014, pp. 5-18.

[2] M. Harman, S. A. Mansouri, and Y. Zhang, "Search based software engineering: a comprehensive analysis and review of trends techniques and applications," King's College London TR-09-03, 2009.

[3] M. Harman, Y. Jia, and Y. Zhang, "Achievements, open problems and challenges for search based software testing," in *Software Testing, Verification and Validation (ICST), 2015 IEEE 8th International Conference on*, 2015, pp. 1-12.

[4] 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, pp. 553-568, Jun 2012.

[5] A. R. A. Alsewari and K. Z. Zamli, "A harmony search based pairwise sampling strategy for combinatorial testing," *International Journal of the Physical Sciences,* vol. 7, pp. 1062-1072, 2012.

[6] D. R. Kuhn, D. R. Wallace, and A. M. Gallo Jr, "Software Fault Interactions and Implications For Software Testing," *IEEE Transactions on Software Engineering,* vol. 30, pp. 418-421, 2004.

[7] R. Brownlie, J. Prowse, and M. S. Phadke, "Robust Testing of AT&T PMX/StarMAIL Using OATS," *AT&T Technical Journal,* vol. 71, pp. 41-47, 1992.

[8] C. J. Colbourn, M. B. Cohen, and R. Turban, "A Deterministic density algorithm for pairwise interaction coverage," in *IASTED Conf. on Software Engineering*, 2004, pp. 345-352.

[9] T. Shiba, T. Tsuchiya, and T. Kikuno, "Using artificial life techniques to generate test cases for combinatorial testing," in *International Computer Software and Applications Conference,*, 2004, pp. 72-77.

[10] X. Chen, Q. Gu, J. Qi, and D. Chen, "Applying particle swarm optimization to pairwise testing," in *34th IEEE Annual Computer Software and Applications Conference (COMPSAC),* , 2010, pp. 107-116.

[11] B. S. Ahmed, K. Z. Zamli, and C. Lim, "The development of a particle swarm based optimization strategy for pairwise testing," *Journal of Artificial Intelligence,* vol. 4, pp. 156-165, 2011.

[12] N. K. Johal, S. Singh, and H. Kundra, "A hybrid FPAB/BBO Algorithm For Satellite Image Classification," *International Journal of Computer Applications (0975–8887),* vol. 6, 2010.

[13] G. Kaur, D. Singh, and M. Kaur, "Robust and Efficient 'rgb'based Fractal Image Compression: Flower Pollination Based Optimization," *International Journal of Computer Applications,* vol. 78, pp. 11-15, 2013.

[14] M. Sharawi, E. Emary, I. A. Saroit, and H. El-Mahdy, "Flower Pollination Optimization Algorithm For Wireless Sensor Network Lifetime Global Optimization," *International Journal of Soft Computing and Engineering,* vol. 4, pp. 54-59, 2014.

[15] A. Hartman and L. Raskin, "Problems and Algorithms For Covering Arrays," *Discrete Mathematics,* vol. 284, pp. 149-156, 2004.

[16] R. Mandl, "Orthogonal latin squares: an application of experiment design to compiler testing," *Communications of the ACM,* vol. 28, pp. 1054-1058, 1985.

[17] K. A. Bush, "Orthogonal Arrays of Index Unity," *The Annals of Mathematical Statistics,* vol. 23, pp. 426-434, 1952.

[18] A. W. Williams, "Determination of Test Configurations For Pair-wise Interaction Coverage-old," in *Testing of Communicating Systems*, ed: Springer, 2000, pp. 59-74.

[19] 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, pp. 437-444, 1997.

[20] A. Williams, "TConfig download page [Online]," p. University of Ottawa. Available: http://www.site.uottawa.ca/~awilliam/[Accessed 23 Dec 2014]. 2008.

[21] B. Jenkins. (2003). *Jenny tool*. Available: http://www.burtleburtle.net/bob/math

[22] A. Hartman, T. Klinger, and L. Raskin, "IBM intelligent test case handler," *Discrete Mathematics,* vol. 284, pp. 149-156, 2010.

[23] Y. Lei and K.-C. Tai, "In-parameter-order: A test generation strategy for pairwise testing," in *3rd IEEE International Symposium on HighAssurance Systems Engineering*, 1998, pp. 254-261.

[24] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: a general strategy for t-way software testing," in *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems*, 2007, pp. 549-556.

[25] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG/IPOG-D: Efficient test generation for multi-way combinatorial testing," *Software Testing, Verification and Reliability,* vol. 18, pp. 125-148, 2008.

[26] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology,* vol. 113, p. 287, 2008.

[27] F. Glover, "Tabu search-part I," *ORSA Journal on computing,* vol. 1, pp. 190-206, 1989.

[28] A. B. NASSER, Y. A. SARIERA, A. R. A. ALSEWARI, and K. Z. ZAMLI, "A cuckoo search based pairwise strategy for combinatorial testing problem," *Journal of Theoretical & Applied Information Technology,* vol. 82, 2015.

[29] R.G. Raj and S. Abdul-Kareem. “A Pattern Based Approach for The Derivation Of Base Forms Of Verbs From Participles And Tenses For Flexible NLP”. *Malaysian Journal of Computer Science*, Vol. 24(2): Jun. 2011. pp 63-72.

[30] M. Patil and P. Nikumbh, "Pair-wise testing using simulated annealing," *Procedia Technology,* vol. 4, pp. 778-782, 2012.

[31] AARTS/KORST., *Simulated annealing and boltzmann machines. a stochastic approach to combinatorial optimization and neural computing*: John Wiley., 1990.

[32] C. Nie and H. Leung, "A Survey of Combinatorial Testing," *ACM Computing Surveys (CSUR),* vol. 43, p. 11, 2011.

[33] R. Qi, Z. Wang, and S. Li, "Pairwise Test Generation Based On Parallel Genetic Algorithm With Spark," in *International Conference on Computer Information Systems and Industrial Applications*, 2015.

[34] M. Harman and B. F. Jones, "Search-based software engineering," *Information and Software Technology,* vol. 43, pp. 833-839, 2001.

[35] X.-S. Yang and S. Deb, "Two-stage eagle strategy with differential evolution," *International Journal of Bio-Inspired Computation,* vol. 4, pp. 1-5, 2012.

[36] P. Leins and C. Erbar, *Flower and fruit: Morphology, ontogeny, phylogeny, function and ecology*: Schweizerbart Stuttgart, 2010.

[37] C. Grüter and F. L. Ratnieks, "Flower constancy in insect pollinators," *Communicative & Integrative Biology,* vol. 4, 2011.

[38] R.G. Raj and A.N. Zainab. “Relative Measure Index: A Metric to Measure the Quality of Journals”, *Scientometrics*, vol. 93, no. 2, 2012, pp. 305-317. doi: 10.1007/s11192-012-0675-z.

[39] K. Z. Zamli, F. Din, G. Kendall, and B. S. Ahmed, "An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation," *Information Sciences,* vol. 399, pp. 121-153, 2017.

[40] K. Z. Zamli, F. Din, S. Baharom, and B. S. Ahmed, "Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites," *Engineering Applications of Artificial Intelligence,* vol. 59, pp. 35-50, 2017.