Journal of Engineering Science and Technology
Vol. 15, No. 2 (2020) 746 - 767

HABC: HYBRID ARTIFICIAL BEE COLONY FOR GENERATING VARIABLE T-WAY TEST SETS

AMMAR K. ALAZZAWI*, HELMI MD RAIS, SHUIB BASRI

Department of Computer and Information Sciences, Universiti Teknologi PETRONAS, Bandar Seri Iskandar 32610, Perak, Malaysia
*Corresponding Author: ammar_16000020@utp.edu.my

Abstract

Exhaustive testing of occurred interaction amongst components (i.e., parameters and values) of a software system is usually impossible due to some factors such as the restriction of budget and time. One of the effective software testing techniques used for detecting faults of interactions between components is combinatorial testing (CT). CT is a black box testing technique, used to find the mistakes among components of a software system in a systematic and effective way. However, CT is highly complex (NP-hard). The input variables for a real-world software may diverge in how they strongly influence variable strength (VS) interaction can achieve that effectively. This paper proposed a hybrid artificial bee colony (HABC) strategy based on the hybrid artificial bee colony algorithm and practical swarm optimization to generate optimal test suite of variable strength interaction. PSO was integrated as the exploitation agent for the ABC hence the hybrid nature. The information sharing ability of PSO via the Weight Factor is used to enhance the performance of ABC. The output of the hybrid HABC is a set of promising optimal test set combinations. Through several benchmark experiments, HABC proved the effectiveness of the proposed strategy. The HABC has achieved 76.31 % better result than most of the compared strategies.

Keywords: An optimization problem, Combinatorial testing, Hybrid artificial bee colony algorithm, Software testing, *t*-way testing, Variable strength interaction.

1. Introduction

Software system failures have been linked to the interactions of its numerous components parameters (i.e., inputs). These interactions amongst the parameters have been considered as one of the primary sources of defects in software systems. Conventional software testing techniques such as boundary value and Equivalence Partitioning have been useful in fault detection. These techniques can address the single input values in detecting the fault from time to time during software testing. However, the conventional techniques may not be efficient enough to address faults that generates among several input variables [1].

Combinatorial Testing (CT) or named *t*-way testing can give a viable solution to the interaction of components for two parameters or more and producing a test set that can assist in fault detection early in software development life cycle (SDLC) (*t* indicates the interaction strength between combinations) [2, 3]. Since not all components of a software system usually leads to faults in such system, this makes CT a feasible approach by generating minimal number of interactions amongst software component. In other words, CT can perform exhaustive testing by reducing the number of test cases [3].

Pairwise or 2-way testing which is a form of CT has been used effectively in several practical software testing [4]. However, many studies have shown software faults maybe as a result of more than two inputs (i.e., parameters and values) [3, 5, 6]. Software faults that happen by more than two interactions and are uniform (i.e., where the parameter values are equal) are known as a *t*-way and but there are many cases that showed the interaction between components are mostly non-uniform or have same input values [1, 7, 8]. Therefore, variable-strength (VS) was introduced as a type of combinatorial testing to handle cases where there are more than two non-uniform interaction input values [9, 10].

Variable-strength combinatorial testing is an NP-hard computational problem as in the case of combinatorial testing. In the case of CT strategies, these strategies are based on the artificial intelligence (AI) and heuristic algorithms and have shown to be efficient and effective in producing minimum test suite of interaction. As a result, many CT strategies based on the Al and heuristic algorithms have been proposed but only a few of these strategies can be deployed as variable-strength to generate optimal test suite [11]. Simulated Annealing (SA) and Particle Swarm Optimization (PSO), which are flexible, practical and one of the most popular heuristic algorithms support the generation of test suites based on variable-strength interaction [7, 12, 13].

As an extension of our previous works [14, 15], hybrid artificial bee colony algorithm (HABC) is developed and employed to support variable strength combinatorial testing. HABC has been enhanced to support uniform and non-uniform covering array (CA) and for generating optimal test suite size for variable strength-covering array (VSCA). The remainder of this paper is structured as follows: Section 2 presents combinatorial interaction testing. Section 3 presents a comprehensive review on VSCA strategies. An overview of ABC algorithm is discussed in Section 4. Section 5 addresses the HABC strategy. Sections 6 and 7 presents the experimental and statistical test results conducted. Section 8 highlights threats to validity of the study and Section 9 addresses the conclusion.

2. Combinatorial Interaction Testing

2.1. Theoretical background

The major motive of t-way testing to produce a test set size (i.e., number of test cases); where this test set indicates an array of (n x m). This n x m signifies the number of rows (i.e., the generated number test cases) and the combination value of each test case respectively. All input parameters and associated value are covered by a test set, where each pair of values can be covered by one test case at least. However, generating the effective test set (i.e., test set with a few test cases) still the main weakness of t-way testing.

In general, each test set contains a specific number of parameters ($P_1, P_2, ..., P_n$) with associated values ($V_1, V_2, ..., V_i$) for every single parameter. In addition, each test set has interaction strength t as N x m, whilst each column involved one value and the sub-array N x t includes all combination minimums once a time.

Covering Array (CA) is a well-established mathematical theme which combinatorial testing theoretically depends on to produce test suite [16]. For statistical experiment's purpose, CA is a preferred alternative in comparison to the old mathematical theme such as Orthogonal Array (OA) [17]. Generally, all systems under test (SUT) include various items such as parameters and their values in addition to t to signify as the interaction strength level.

Definition 1: The earlier description mentioned that each SUT has various parameters (P) linked with their values (V), the CA termed uniform interaction strength CA (N, t, v^P) if all V equates to each other for all P. For example, it is presumed that a system with 4-parameters are linked to 2-values each, it can be identified as CA (6; 2, 2^4). The system includes six test cases (rows) that are created in reference to four parameters (columns).

Definition 2: MCA (N, t, v_1^{p1} v_2^{p2} v_3^{p3} . . . · · v_i^{pj}) is representing the mixed covering array as the number of values are not matched to each parameter, which is different from uniform interaction strength. As an example, a system with four parameters, e.g., 3-parameters with 2-values and 1-parameter with 3-values is expressed as MCA (12, 3, 2^3, 3^1). The system includes 12 test cases (rows) that are created in reference to four parameters (columns).

Definition 3: Additional to CA and MCA notation in the real-world systems, due to the complexity of the systems resulting in the advancement of technology, there is a big difference in interaction of parameters. Parameters may be stronger or may not interact with each other, and this leads to a variable strength covering array (VSCA). The VSCA is proposed to solve this problem and can be represented as VSCA (N; t, v^p, (CA_1 … CA_i)), where both CA_1 and CA_i represent a subset of main-set with different interaction strength.

To illustrate VSCA, a system with 5-parameters have 3-values each, where the VSCA concepts consider t=3 for main-set and t=2 for the sub-set as shown in Fig. 1. The CA can be represented as VSCA (N, 3, 3^5, (CA (N, 2, 3^3)). The system involves 27 test cases (rows) that are produced based on five parameters (columns).

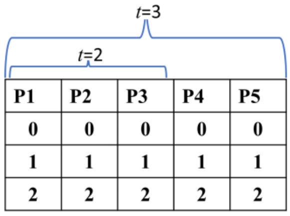

Fig. 1. Variable strength interaction system.

2.2. T-way test generation problem

The *t*-way testing technique is vital for generating test cases; however, these techniques are dependent on the element's behavior interaction within the system. A web configurable software system is utilized to justify the concept of *t*-way testing techniques in generating the optimal test set as seen in Fig. 2. This system is composed of five components, namely; a device, a processor, the operating system (OS), a browser, and a Screen with Fig. 2 illustrating their relationships. This system has been employed as a simple illustration of the main idea in *t*-way testing regarding variable strength, and uniform strength. Each component of this system is known as a separate parameter, and each parameter has one or more values with Table 1 displaying the parameters of the systems and their respective values. In this example, the number of parameters is five, consisting of three parameters with two values and two parameters with three values. The adopted display tab interaction strength is t=2; the CA is represented as MCA (N, 2, 3^2 2^3).

Table 1. A Web configurable software system.

Parameters				
Device	**Processor**	**OS**	**Browser**	**Display**
PC	Dual core	Android	Firefox	720 x 1280
Phone	Multi core	IOS	Chrome	1024 x 768
		Windows	Safari	

The exhaustive test case to test this system completely is (i.e., 2 × 2 × 3 × 3 × 2= 72) test cases. Assuming the t that denotes the interaction strength is 2, the test set created is 9 test cases only. Consequently, approximately 80% of the resources will be saved. The interactions between parameters are directly proportional to the number of test cases. In general, the minimum test case for each combination value is one test case [18, 19]. The study of NASA application showed if we tested parameter (t=1), (t=2) and (t=3), it can detect 67%, 93% and 98% failures respectively. It will eventually achieve 100% failure detection with (t = 4-6) interaction of elements [20, 21]. Therefore, increasing the interaction strength of testing is not important. However, it often comes that certain components have complex interactions while others may have few or none. For this reason, it is not important to go for higher interaction strength of testing.

To achieve better testing, identify some components (i.e., subsets) with strong interactions, and increase the interaction strength higher than the interaction strength of the main components to detect the faults by their interactions. This is called the variable strength-covering array (VSCA). A web configurable software system example is next shown to understand VSCA. In the web, configurable software assumes that all parameter combinations require 2-way and a device, processor and OS that requires 3-way interaction. In this case, the configuration system is indicated as VSCA (N; 2, $2^3 3^2$, CA (3, $2^2 3^1$)). Therefore, a web configurable software system required 12 test cases for VSCA. We can notice that VSCA achieving higher coverage of the fault detection.

3. Related Work

Many researchers have suggested several *t*-way strategies for the last two decades. Most of these suggested strategies such as Bat-Testing Strategy [22-28], Jenny [29], etc. have been focusing on the uniform *t*-way and pairwise testing strategies (where the interaction strength *t*=2). Therefore, most of the available strategies depends on orthogonal array like (OA) [30, 31], MOA [32]. However, still, there are restricted in application to small configurations [33, 34] only, although with fast execution time.

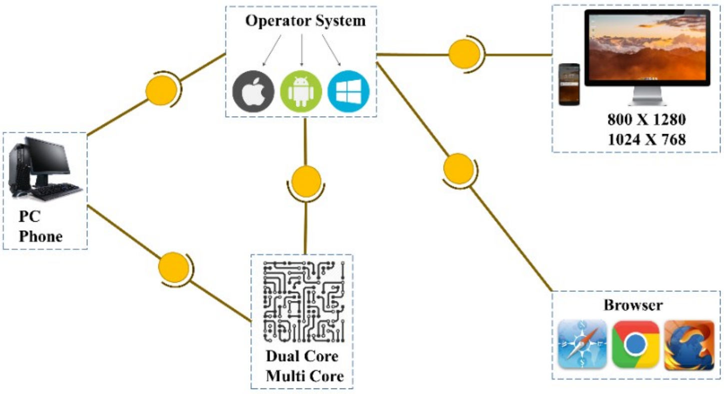

Fig. 2. An illustration of the web configurable software system.

Given the mentioned constraints above, focus has shifted to *t*-way uniform strategies such as GTWay [35], Jenny [29], TCG [36], AETG [37], TConfig [38] and MIPOG [39] that support the uniform interaction strength. However, due to the progress in software domain, it is hard to find uniformity as interaction between parameters became hard. Therefore, several strategies have been suggested to support variable strength interaction such as VS-PSTG [7], ACS [40], IPOG [33, 41], ITTDG [42], Density [43], and PICT [44, 45]. Consequently, one of the goals of this paper is to focus on uniform interaction strength and variable interaction strength for *t*-way strategies that previously were ignored. These strategies are categorized based on the general computational based strategies or Meta-heuristic based strategies as follows:

3.1. General computational based strategies

ITTDG strategy is categorized as a one-test-at-a-time strategy that supports the variable strength interaction. It generates one test case iteratively to achieve the maximum coverage for all interactions. At the end, ITTDG selects the best test cases (test set) of several generated test case candidates that can achieve the maximum coverage of the uncovered components. The test case selection of ITTDG strategy adopts random heuristics and iterative as AETG. However, AETG the contrary of the ITTDG strategy, where to produce new test case for every single iteration; ITTDG generates new candidates test case when there is a tie situation (i.e., when more than one value can cover the most uncovered tuples).

PICT [45] is proposed by Czerwonka in 2006 to generate a complete test set, which has been used extensively by Microsoft for software testing purposes. Based on the SUT configuration, PICT generates all possible combinations and marks every combination as an uncovered combination. Therefore, the test engineer will announce if there is any constraint. Based on these constraints, the combination will be marked as excluded. After that, one of the uncovered combinations will be selected and extended until completion using a greedy heuristic. This is done to achieve the maximum coverage as possible as without excluded any combination. The selected test case will be stored as the final test set; this procedure will be repeated until all uncovered combinations are covered.

Another strategy that can be characterized as one-test-at-a-time is Density. It works exactly like PICT and ITTDG, however, the generated test case for this strategy depends completely on a mathematical formula obtained from density properties. Bryce introduced Density [46, 47] to generate the final *t*-way test set, and to generate the uniform *t*-way. However, density concepts are extended by Wang et al [43] to support the variable strength interaction by introducing new formulae for local and global density [43, 48].

In contrast to one-test-at-a-time strategies, IPOG is one-parameter-at-a-time strategy. The IPOG generation process begins with the creation of a full test set for the initial *t* of parameters (the highest *t* will be chosen for the VSCA (*t*) case) as an initial test set. IPOG generation process is dependent on two processes namely horizontal and vertical extension as one of one parameter at time strategies. The horizontal extension will begin to add one parameter to the test set continuously until the test set covered all parameters. Vertical extension is responsible to ensure all combinations are covered but if there is a case of the test set that cannot cover some combinations, the test set will be expanded vertically by addition of several new test cases on the initial set.

3.2. Meta-heuristic based strategies

All strategies mentioned earlier are a part of the computational approach; however, several strategies have been proposed that depends on artificial intelligence (AI) approach such as VS-PSTG [7, 49] and ACS [40] for generating a *t*-way test set. VS-PSTG and ACS are *t*-way strategies based on particle swarm optimization and ant colony optimization respectively as the key feature. The searching process of VS-PSTG for the best test case is inspired by flocks of birds. In order to detect the best test case, VS-PSTG iteratively merges local and global search to cover the interaction combinations greedily. For ACS, the ant colony searches for the best test case on some paths, the path's qualities are evaluated in terms of the pheromones. Consequently, the best test case that corresponds with the optimum path is included in the final test set. VS-PSTG and ACS are exactly similar to PICT, Density and ITTDG as one-test-at-a-time strategies. Alsewari [18] proposed Harmony Search Strategy (HSS) based on Harmony Search algorithm for test set generation. HSS is one of the existing strategies that addressed the support for constraints by mimicking musician behaviour to create good music. Iteratively, HSS executes the global search by inserting randomized values to the Harmonic memory whereby the local best value can be selected considering rate probability. The best value at each iteration will be added to the final test set until all required interactions are covered.

4. An Overview of Artificial Bee Colony Algorithm

Artificial Bee Colony (ABC) is one of the meta-heuristics algorithms inspired by the foraging behaviour of honeybee colony as proposed by Karoboge in 2005 [50]. This algorithm is performed by three kinds of bees: employed bees, onlooker bees, and scout bees. The onlooker bees and scout bees are also identified as unemployed bees. These bees are assigned to increase the food source from the hive (nectar) by dividing the bee population and organize them with a specific task. Half of the colony is presented by an employed bee, while the remaining half is represented by onlooker bee. Employed bees are given the task of exploring the higher nectar potential food source and convey the information to standby onlooker bees at the hive. The information includes the navigation, location, and potential of the food source. The selection process of food source made by onlooker bees is in reference to the information disseminated by the employed bees. Scout bees are derived from employed bees and assigned to search the environment randomly for new or better food source discovery. In general, employed bees and onlookers perform exploitation process (where exploitation refers to finding good solutions), while scout bees perform exploration process (where exploration refers to the avoiding being trapped in local optima by widening the search to new areas). The behavioural manner of an artificial bee colony can be divided into four phases:

i. Initial phase: - using Eq. (1), the algorithm initiates the food source discovery by randomly searching the environment, provided it is within the algorithm's parameters border; producing the initial food sources.

$$x_{ij} = x_{min,j} + \text{rand }(0,1)(\, x_{max,j} - x_{min,j}) \tag{1}$$

ii. Employed bee phase: -Upon identified, the food source information will be detected by the employed bees. The employed bee numbers equate to a number of a food source at the ratio 1:1. The employed bees will gather the nectar and return to the hive with all the standby bees to convey the information by dancing in the dance area. The employed bee then transformed into a scout bee if the nectar of source is running out so it can begin to search for a better food source. The discovery for a new food source is expressed as below Eq (2).

$$V_{i,j} = X_{i,j} + rand(-1, 1)(\, x_{i,j} - x_{k,j}) \tag{2}$$

Upon detection of the food source, the selection food source probability is expressed using Eq (3).

$$fitness_i = \begin{cases} \frac{1}{1+f_i}\,, & if\ f_{i\,\geq 0} \\ 1 + |f_i|, & if f_{i\,<0} \end{cases} \tag{3}$$

iii. Onlooker bee phase: -The amount of nectar is the main criterion for onlooker bee food source selection activity. The information on the profitability of the food nectar is transmitted by employed bee on the dance area. The food source selection probability is expressed using Eq. (4).

$$Pi = \frac{fit_i}{\sum_{n=1}^{sn} fit_n} \tag{4}$$

iv. Scout bee and Limit phase: Upon completion of their tasks, both employed and onlooker bees, the search continues for any remaining source that is not yet explored. The credibility of the food source is calculated using limit Eq. (5). The limit is a control parameter where the counter value of the food source is compared to the algorithm, if it is superior to the limit, the food source will be disregard. A better food source discovered by scout bee will replace the disregard food source. The second phase to the fourth phase is repeated until all solutions are found. The flowchart on Fig. 3 represents all the bee's interaction and activities.

$$limit = c.ne\ .D \tag{5}$$

where *ne* signify the number of unemployed bees, *c is a* constant coefficient with a recommended value of 0.5 or 1. ABC minimum application requirement is one scout bee implementation. Scout-type operations hypothetical searches in the whole D-dimensional space. As they provide exceptional effectiveness to the ABC method in searching the best global solution. Scout bees are independent when it comes to global optimum solution discovery in comparison to other bee types. Both (employed/onlooker) concurrently check on their local candidate solutions for the global best. Thus, it is impossible for ABC to be trapped in local optima [50].

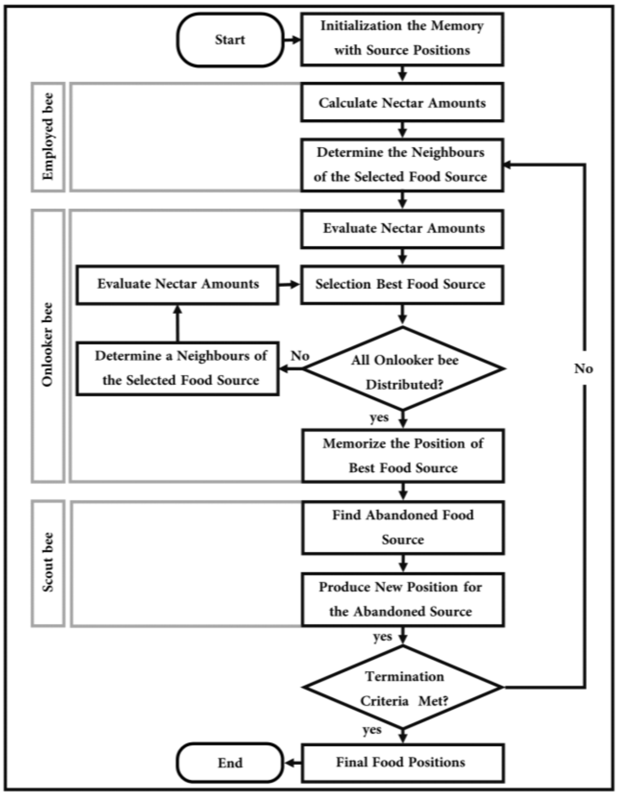

Fig. 3. Flowchart of the artificial bee colony algorithm.

5. Hybrid Artificial Bee Colony Algorithm

The swarm intelligence (SI) algorithms are the most utilized algorithm for the past twenty years. Its establishment was inspired from foraging behaviour of bird, fish, bee and insect colonies such as termites and ants. Its derives such as the Artificial Bee Colony (ABC) algorithm [50], Bacterial Foraging Optimization (BFO) [51], Particle Swarm Optimization (PSO) [52], etc. are applied to real world optimization problem. Bee colony algorithm is one of the derivatives of SI algorithm that are used to solve optimization problems. This algorithm has various advantages which helps to implement intelligent search such as "queen bee, task selection, bee foraging, navigation systems, nest site selection, mating, collective decision making, bee dance (communication), floral/pheromone laying" [53].

ABC algorithm has its downfall as well where its convenient operation is very dependent on the solution development process. Suppose it has insufficiencies, it will increase the convergence speed of the algorithm [53]. The occurrence of rapid convergence in this algorithm can lead to obstructions for several complications in the local optimum. Besides that, the information sharing activity is being completed in one dimension with random neighbour to one solution improvement, thus the speed of convergence is inversely proportional to problem dimension. [54]. Hence, the weak performance as a result of information sharing activity of the ABC algorithm [55]. The chance to search optimization algorithm is very low, thus the failure to get global optimum for most optimization problem. These limitations can be surpassed by tuning or modifying different characteristics of ABC algorithm such as convergence speed, exploration and exploitation ability, and missed being trapped at the local optimum [53, 54]. However, the focus is mainly on information conveying activity or solution improvements, which are the dual most vital parameter of exploitation and exploration ability of the algorithm.

From this research, we believe there still need for ABC algorithm improvement by adopting some characteristics from particles in the PSO algorithm operation. The mechanism of PSO is one of a kind in terms of information sharing and solution improvement processes. It requires a crucial and unique parameter termed as Weight Factor (w). Referring to the previous solution, the velocity parameter is vital for required improvement degree control. Besides velocity, there are dual factors (C_1 and C_2) which are required for the relative influence of cognitive (self-confidence) and social (swarm-confidence) components determination, respectively. (Eq. (6)).

$$V_{i,d}^{t+1} = W^t * V_{i,d}^t + C_1^t * r_1 * (pbest_{i,d}^t - X_{i,d}^t) + C_2^t * r_2 * (gbest_{i,d}^t - X_{i,d}^t) \tag{6}$$

The particles in PSO depends on three categories for the movement operation or the local search. The following are the categories:

i. The variation of velocity will affect every subsequent particle movement. It can be assumed that particles movement is not random or arbitrary.
ii. The local best solution variable generates the best local information which later links with the chosen particle's next move value.
iii. The particle's next move best dependent on global best solution variable.

The absence of these properties is identified in the ABC algorithm. As shown in Eq. (2), the selected neighbour solution was randomly searched to determine the size that is a parameter for solution improvement or local search that bees depend on, which represents the exploitation process. This has proven that the used value for experimental solutions in every single loop is random. On the contrary, the velocity value of PSO varies. The local search of ABC algorithm lacks an information-based procedure, where the discovered solutions (the best) usually not held in the population. Therefore, it will be exchanged with other generated solutions randomly by scout bee; it may not clearly contribute in generated experimental solutions. The chosen solution will be selected at the improvement part (local search) of employed bee and onlooker bee stage. Nevertheless, onlooker bees create experimental solutions using a higher fitness value of the solutions. The onlooker bee is highly dependent on the probability of the solution value. Scout bees have the ability to constraint problematic search. For example, premature convergences providing a global mechanism search for ABC. PSO has this limitation in its search procedures. PSO keeps the best solution despite this shortcoming and the new velocities are utilized to generate new solutions. With a shorter time, the regions illustrated considered as search space is ransacked in detail. This is a good enough justification why the algorithm will be trapped in the local minima and cause limitation for ABS as suggested by the proposed HABC. This ABC limit parameter intercepts the algorithm from trapping in the local minima with the act of inserting the random selected solution into search space occasionally. The limit parameter represents the exploration process. The strengths of the PSO and the strengths of ABC are combined to minimize optimization search troubles as proposed in HABC algorithm. However, there are still several issues among researchers about ABC and PSO that are taken into account. There are idea clashes in terms of execution time as PSO is faster than ABC, but ABC is more accurate solution than PSO [56].

HABC algorithm proposed that the bee colony size is comprised mainly into three types of bees known as employed, onlookers and scouts bee. The food source thus equates to the colony size. The employed bee transformed into a scout bee once the food source is exhausting. The scout bee then searches for a better food source to replace the old one. The flowchart on Fig. 4 represents all the bee's interaction and activities of HABC algorithm.

The discovery activity of HABC is listed as the following:

- The algorithm initiates the food source discovery and identifies the amount of nectar source.
- The employed bee conveys the information of the selected food source to the standby onlooker bee within the dance area (in the hive). The food source with higher nectar amount is selected by the onlooker bee.
- The selected food source is examined if it needs a new replacement food source using Eq. (6).
- If the food source failed, an employed bee transformed into a scout bee. The scout bee is assigned to find a replacement for the old food source using Eq. (1).

The main steps of the HABC algorithm are shown as follows in Fig. 5:

The HABC algorithm search cycle is enclosed within these three steps;

i. Employed bees evaluate the profitability of the food source in the search space, choose the food source with higher random discovery probability, and selected by the onlooker bees.
ii. PSO algorithm as in Eq. (6) is an alternative equation for the local discovery of the employed bee phase replacing Eq. (2).

Scout bees use Eq. (1) search for better food source if the parameter is out limit (similar to the original ABC).

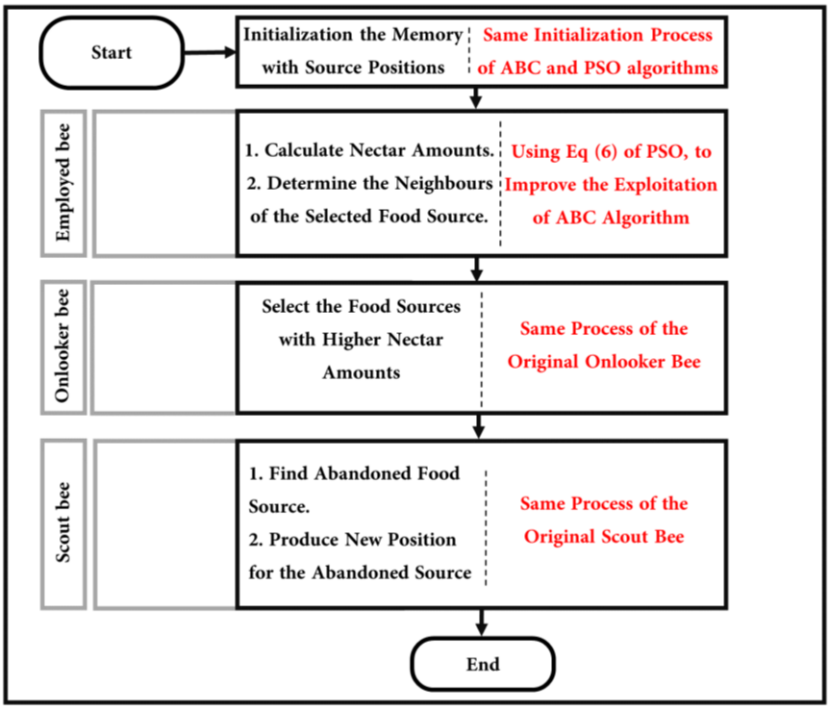

Fig. 4. Flowchart of the hybrid artificial bee colony algorithm.

1: *Initialization step: The same process as the original ABC and PSO algorithms.*
2: *REPEAT*
3: *Move the employed bees onto their food sources and determine their nectar amounts.*
4: *Calculate the probability value of the sources with which they are preferred by the onlooker's bee.*
5: *Move the Onlookers onto the food sources and determine their nectar amounts.*
6: *Move the scouts to search for new food sources replacing the abandoned ones.*
7: *Memorize the best food source found so far.*
8: *UNTIL (requirements are met).*

Fig. 5. Hybrid artificial bee colony algorithm.

6. Experiments

6.1. Experimental Setup

The main goal of this section is to evaluate the proposed HABC strategy compared to the existing tools and strategies such as Jenny, TConfig, PICT, TVG, IPOG, IPOG-D, VS-PSTG, ACS, SA, HSS, and VS-MGS. The benchmarking data of the tools and strategies were adopted from the published results in [14, 15, 18, 41, 57-61]. The HABC strategy parameters were set at *Nbees* = 5, *maxCycle*= 1000, limit = 100, *C1* & *C2*=2.0 and *W*=0.9. All experiments are implemented on a Windows 7 (OS) desktop computer with 3.40 GHz Xeon (R) CPU E3 and 8GB RAM. The Java language JDK 1.8. It was used to code and implement the HABC. Figure 6 shows the programming code during the implementation of the actual input.

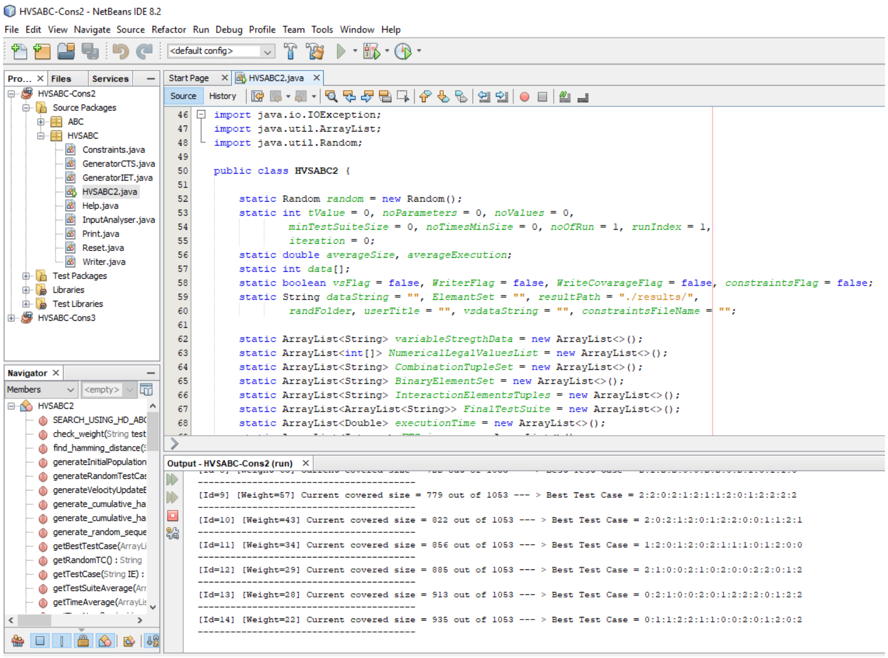

Fig. 6. The HABC strategy implementation.

6.2. Experimental evaluation

Due to the randomization characteristic of the proposed HABC strategy, the experiment runs twenty independent times for each configuration system to get the best result. Tables 2 to 7 present the experimental result, and each table presents the optimal test set size for each configuration. The dark cell with (*) represents the optimal test set size and the cell with NA or NS represents (not available) and (not supported) respectively. Tables 2, 3 and 4 mainly focus on the *t*-way, while Tables 5, 6 and 7 focus on VSCA.

The proposed HABC strategy showed a good result compared to other existing strategies as depicted in Table 2. HABC produced the optimal test set size for both configurations No. 3 and No. 5 when the interaction strength equals 4 and 6. However, HABC strategy generated similar results to CS and HSS for configuration No.1 and No. 4. ITCH produced the optimal test set size when the interaction strength equals 3. Jenny, TConfig, PICT, TVG, CTE-xl, IPOG-D, IPOG, PSTG and ABC-TG produced worst results most of the time compared to HABC, HSS and CS. In Table 3, the HABC strategy showed a good result better than HSS, CS, PSTG and ABC-TG. HABC strategy produced the most minimum test set size for configuration No. 1 like both of ITCH, IPOG-D, and PSTG. On the other hand, HABC strategy produced the optimal test set size when the number of the parameters was 9 and 10 for Configuration No. 6 and No. 7. ITCH and CS produced the optimal test set size for configuration No. 4, No. 5 and No. 2 respectively. Jenny, TConfig, PICT, TVG, CTE-Xl, IPOG-D, IPOG, PSTG, HSS and ABC-TG produced worst results most of the time compared to HABC. From Table 4, it is evident that HABC strategy outcomes excelled for both of the configuration No. 1, No. 3, No. 4 and No. 5 and regarding configuration No. 2 produced a competitive test set close to the optimal test set.

Tables 5, 6 and 7 highlights the VSCA experimental results with Tables 5 and 6 fairly focusing on variable strength interaction with the uniform covering array, and Table 7 on variable strength interaction with a mixed covering array (Non-uniform). In overall, HABC strategy produced four of the optimal test set size for the sub-configurations No. 9 (CA (5, 3^7) and No. 11 (CA (6, 3^7) in Table 5, the sub-configuration No. 8 (CA (4, 3^7)) in Table 6, and the sub configuration No. 12 (MCA (5, 4^3 5^3)) in Table 7 compared to other existing strategy's results. On the other hand, some of HABC strategy results match the best results of other strategies for most of the sub configurations. The shaded and bold cells presented the best test set size that matched with other results reported by existing strategies. HABC and HSS had the best results followed by PSTG, SA, ACS, and GS respectively. Overall, WHITCH, ParaOrder, Density, and PICT produced the worst results.

Table 2. CA (N; 3, 3^P), P is variable from 4 to 10.

		Pure Computational Strategies								AI-Strategies					
Number	**Parameter**	**Jenny** *Best*	**TConfig** *Best*	**ITCH** *Best*	**PICT** *Best*	**TVG** *Best*	**CTE-XL** *Best*	**IPOG-D** *Best*	**IPOG** *Best*	**PSTG** *Best*	**ABC-TG** *Best*	**CS** *Best*	**HSS** *Best*	**HABC** *Best*	**HABC** *Avg.*
1	**4**	34	32	**27**	34	34	34	**27**	39	**27**	33	28	30	**27**	29.75
2	**5**	40	40	45	43	41	43	49	43	39	40	**38***	39	39	43.00
3	**6**	51	48	45	48	49	52	49	53	45	**43**	**43**	45	**43**	45.55
4	**7**	51	55	**45***	51	55	54	63	57	50	50	48	50	47	50.90
5	**8**	58	58	**45***	59	60	63	63	63	54	54	53	54	50	54.30
6	**9**	62	64	75	63	64	66	71	65	58	58	58	59	**56***	58.85
7	**10**	65	68	75	65	68	71	71	68	62	62	62	62	**61***	63.50

Table 3. CA (N; 3, V^7), V is variable from 2 to 6.

		Pure Computational Strategies								AI-Strategies					
Number	**Value**	**Jenny** *Best*	**TConfig** *Best*	**ITCH** *Best*	**PICT** *Best*	**TVG** *Best*	**CTE-XL** *Best*	**IPOG-D** *Best*	**IPOG** *Best*	**PSTG** *Best*	**ABC-TG** *Best*	**CS** *Best*	**HSS** *Best*	**HABC** *Best*	**HABC** *Avg.*
1	**2**	14	16	13	15	15	15	14	19	13	**12**	**12**	**12**	**12**	14.700
2	**3**	51	55	**45***	51	55	54	63	57	50	49	49	50	47	50.900
3	**4**	124	112	112	124	134	136	112	208	116	116	117	121	**110***	117.85
4	**5**	236	239	225	241	260	267	292	275	225	228	223	223	**222***	224.90
5	**6**	400	423	1177	413	464	467	532	455	425	391	NA	NA	**376***	384.80

Table 4. Test set size for VSCA (N, 2, 3^{15}, {C}).

		Pure Computation Strategies						AI- Strategies								
Number	**{C}**	**WHITCH** *Best*	**IPOG** *Best*	**ParaOrder** *Best*	**Density** *Best*	**TVG** *Best*	**PICT** *Best*	**SA** *Best*	**ACS** *Best*	**PSTG** *Best*	**HSS** *Best*	**PwiseGen** *Best*	**GS** *Best*	**VS-MGS** *Best*	**HABC** *Best*	**HABC** *Avg.*
1	Ø	31	21	33	21	22	35	**16**	19	19	20	**16**	19	20	19	21.300
2	CA (3, 3^3)	48	**27**	**27**	28	**27**	81	**27**	**27**	**27**	**27**	**27**	28	**27**	**27**	27.850
3	CA (3, 3^4)	59	39	**27**	32	35	105	**27**	**27**	30	**27**	**27**	29	31	30	34.550
4	CA (3, 3^5)	62	39	45	40	41	131	**33**	38	38	38	**33**	38	39	39	42.900
5	CA (4, 3^4)	103	**81**	NA	NA	**81**	245	NA	NA	**81**	**81**	**81**	**81**	**81**	**81**	81.200
6	CA (4, 3^5)	118	122	NA	NA	103	301	NA	NA	97	94	**91***	92	96	93	103.40
7	CA (4, 3^7)	189	181	NA	NA	168	505	NA	NA	158	159	158	155	**153***	154	160.20
8	CA (5, 3^5)	261	**243**	NA	NA	**243**	730	NA	NA	**243**	**243**	**243**	**243**	NA	**243**	243.10
9	CA (5, 3^7)	481	581	NA	NA	462	1356	NA	NA	441	441	441	441	NA	**439***	446.15
10	CA (6, 3^6)	745	**729**	NA	NA	**729**	2187	NA	NA	**729**	**729**	**729**	**729**	**729**	**729**	729.00
11	CA (6, 3^7)	1050	967	NA	NA	1028	3045	NA	NA	966	902	NA	960	NA	**830***	961.10
12	CA (3, 3^4) CA (3, 3^5) CA (3, 3^6)	114	51	44	46	53	1376	**34***	40	45	45	NA	NA	NA	82	85.100
13	CA (3, 3^6)	61	53	49	46	48	146	**34***	45	45	45	40	46	44	45	46.700
14	CA (3, 3^7)	68	58	54	53	54	154	**41***	48	49	51	47	50	48	50	51.850
15	CA (3, 3^9)	94	65	62	60	62	177	**50**	57	57	62	57	57	57	**50**	60.100
16	CA (3, 3^{15})	132	NS	82	70	81	83	**67***	76	74	77	74	75	81	81	83.200

Table 5. Test set size for VSCA (N; 3, 3^{15}, {C}).

		Pure Computation Strategies						AI- Strategies								
Number	**{C}**	**WHITCH** *Best*	**IPOG** *Best*	**ParaOrder** *Best*	**Density** *Best*	**TVG** *Best*	**PICT** *Best*	**SA** *Best*	**ACS** *Best*	**PSTG** *Best*	**HSS** *Best*	**PwiseGen-VSCA** *Best*	**GS** *Best*	**VS-MGS** *Best*	**HABC** *Best*	**HABC** *Avg.*
1	Ø	75	82	NA	NA	84	83	NS	NS	75	75	NA	**74***	NA	85	86.50
2	CA(4, 3^4)	129	**87**	NA	NA	93	1507	NS	NS	91	**87**	NA	88	NA	94	98.55
3	CA(5, 3^5)	273	**243**	NA	NA	244	5366	NS	NS	**243**	**243**	NA	**243**	NA	**243**	247.10
4	CA(6, 3^6)	759	**729**	NA	NA	**729**	12,609	NS	NS	**729**	**729**	NA	**729**	NA	**729**	729.60
5	CA(4, 3^5)	151	119	NA	NA	118	1793	NS	NS	114	112	NA	**111***	NA	115	120.25
6	CA(5, 3^6)	387	337	NA	NA	323	5387	NS	NS	314	309	NA	**308***	NA	325	331.25
7	CA(6, 3^7)	1441	1215	NA	NA	1018	16,792	NS	NS	1002	**888***	NA	959	NA	964	994.85
8	CA(4, 3^7)	219	183	NA	NA	168	2781	NS	NS	159	159	NA	158	NA	**157***	163.40
9	CA(4, 3^9)	289	227	NA	NA	214	3095	NS	NS	195	199	NA	**194***	NA	199	202.65
10	CA(4, 3^{11})	354	259	NA	NA	256	2824	NS	NS	226	242	NA	**226***	NA	238	242.60
11	CA(4, 3^{15})	498	498	NA	NA	327	NA	NS	NS	284	323	NA	**282***	NA	315	324.25
12	CA(5, 3^7)	481	713	NA	NA	471	7475	NS	NS	**437**	448	NA	**437**	NA	442	445.75
13	CA(5, 3^8)	620	714	NA	NA	556	8690	NS	NS	**516**	**516**	NA	**516**	NA	531	537.55
14	CA(6, 3^8)	1513	2108	NA	NA	1479	22,833	NS	NS	**1396***	1430	NA	1397	NA	1414	1425.6
15	CA(6, 3^9)	1964	2124	NA	NA	1840	26,729	NS	NS	1690	1739	NA	**1687**	NA	1737	1743.45

Table 6. Test set size for VSCA (N, 2, 4^3 5^3 6^2, {C}).

		Pure Computation Strategies						AI- Strategies								
Number	**{C}**	**WHITCH** *Best*	**IPOG** *Best*	**ParaOrder** *Best*	**Density** *Best*	**TVG** *Best*	**PICT** *Best*	**SA** *Best*	**ACS** *Best*	**PSTG** *Best*	**HSS** *Best*	**PwiseGen-VSCA** *Best*	**GS** *Best*	**VS-MGS** *Best*	**HABC** *Best*	**HABC** *Avg.*
1	Ø	48	43	49	41	44	43	**36**	41	42	42	37	NA	41	42	46.050
2	CA (3, 4^3)	97	83	**64**	**64**	67	384	**64**	**64**	**64**	**64**	**64**	NA	**64**	**64**	64.000
3	MCA (3, 4^3 5^2)	164	147	141	131	132	781	**100***	104	124	116	120	NA	120	121	130.90
4	CA (3, 5^3)	145	136	126	**125**	**125**	750	**125**	**125**	**125**	**125**	**125**	NA	**125**	**125**	125.00
5	MCA (4, 4^3 5^1)	354	329	NA	NA	**320**	1920	NS	NS	**320**	**320**	**320**	NA	**320**	**320**	320.00
6	MCA (5, 4^3 5^2)	1639	1602	NA	NA	**1600**	9600	NS	NS	**1600**	**1600**	**1600**	NA	NA	**1600**	1600.0
7	CA (3, 4^3) CA (3, 5^3)	194	136	129	**125**	**125**	8000	**125**	**125**	**125**	**125**	**125**	NA	NA	**125**	125.00
8	MCA (4, 4^3 5^1) MCA (4, 5^2 6^2)	1220	**900**	NA	NA	**900**	288,000	NS	NS	**900**	**900**	**900**	NA	NA	**900**	900.00
9	CA (3, 4^3) MCA (4, 5^3 6^1)	819	**750**	NA	NA	**750**	48,000	NS	NS	**750**	**750**	**750**	NA	NA	**750**	750.00
10	CA (3, 4^3) MCA (5, 5^3 6^2)	4569	**4500**	NA	NA	**4500**	288,000	NS	NS	**4500**	**4500**	**4500**	NA	NA	**4500**	4500.0
11	MCA (4, 4^3 5^2)	510	512	NA	NA	496	2874	NS	NS	472	**453***	454	NA	461	461	476.75
12	MCA (5, 4^3 5^3)	2520	2763	NA	NA	2592	15,048	NS	NS	2430	2430	NA	NA	NA	**2411***	2437.0
13	MCA (3, 4^3 5^3 6^1)	254	215	247	207	237	1266	**171***	201	206	212	204	NA	203	212	219.20
14	MCA (3, 5^1 6^2)	188	**180**	**180**	**180**	**180**	900	**180**	**180**	**180**	**180**	**180**	NA	**180**	**180**	180.00
15	MCA (3, 4^3 5^3 6^2)	312	NS	307	256	302	261	**214***	255	260	263	260	NA	258	269	276.45

Table 7. Wilcoxon signed rank sum test for Table 2.

Pairs	Ranks			Test statistics			Conclusion
	ABCS <	ABCS >	ABCS =	Z	Asymp. Sig. (2-tailed)	α holm	
HABC-HSS	6	0	1	2.225996	0.026014	0.0041	Reject the null hypothesis
HABC- PICT	7	0	0	2.387848	0.016947	0.0045	Reject the null hypothesis
HABC-TConfig	7	0	0	2.387848	0.016947	0.0050	Reject the null hypothesis
HABC- CTE-XL	7	0	0	2.374929	0.017552	0.0055	Reject the null hypothesis
HABC- TVG	7	0	0	2.374929	0.017552	0.0062	Reject the null hypothesis
HABC- Jenny	7	0	0	2.374929	0.017552	0.0071	Reject the null hypothesis
HABC-IPOG	7	0	0	2.370669	0.017756	0.0083	Reject the null hypothesis
HABC-ABC-TG	6	0	1	2.207471	0.027281	0.0100	Reject the null hypothesis
HABC-IPOG-D	6	0	1	2.207471	0.027281	0.0125	Reject the null hypothesis
HABC-PSTG	5	0	2	2.031856	0.042168	0.0166	Reject the null hypothesis
HABC-CS	5	1	1	1.725324	0.084469	0.0250	Retain null hypothesis
HABC- ITCH	4	2	1	1.261412	0.207160	0.0500	Retain null hypothesis

Table 8. Wilcoxon signed rank sum test for Table 3.

Pairs	Ranks			Test statistics			Conclusion
	ABCS <	ABCS >	ABCS =	Z	Asymp. Sig. (2-tailed)	α holm	
HABC-PSTG	5	0	0	2.031856	0.042168	0.0050	Reject the null hypothesis
HABC-IPOG-D	5	0	0	2.031856	0.042168	0.0055	Reject the null hypothesis
HABC- Jenny	5	0	0	2.031856	0.042168	0.0062	Reject the null hypothesis
HABC-IPOG	5	0	0	2.022600	0.043114	0.0071	Reject the null hypothesis
HABC- CTE-XL	5	0	0	2.022600	0.043114	0.0083	Reject the null hypothesis
HABC- TVG	5	0	0	2.022600	0.043114	0.0100	Reject the null hypothesis
HABC- PICT	5	0	0	2.022600	0.043114	0.0125	Reject the null hypothesis
HABC-TConfig	5	0	0	2.022600	0.043114	0.0166	Reject the null hypothesis
HABC-ABC-TG	4	0	1	1.841149	0.065600	0.0250	Retain null hypothesis
HABC- ITCH	4	1	0	1.354571	0.175554	0.0500	Retain null hypothesis

Table 9. Wilcoxon signed rank sum test for Table 4.

Pairs	Ranks			Test statistics			Conclusion
	ABCS <	ABCS >	ABCS =	Z	Asymp. Sig. (2-tailed)	α holm	
HABC- TVG	10	1	5	2.045961	0.040760	0.0100	Reject the null hypothesis
HABC-WHITCH	16	0	0	3.517372	0.000436	0.0125	Reject the null hypothesis
HABC- PICT	16	0	0	3.516196	0.000438	0.0166	Reject the null hypothesis
HABC-HSS	7	4	5	0.670151	0.502762	0.0250	Retain null hypothesis
HABC-PSTG	5	4	7	0.593914	0.552570	0.0500	Retain null hypothesis

Table 10. Wilcoxon signed rank sum test for Table 5.

Pairs	Ranks			Test statistics			Conclusion
	ABCS <	ABCS >	ABCS =	Z	Asymp. Sig. (2-tailed)	α holm	

HABC-GS	1	12	2	3.114669	0.001842	0.0083	Reject the null hypothesis
HABC-WHITCH	14	1	0	3.352327	0.000801	0.0100	Reject the null hypothesis
HABC-IPOG	11	2	2	2.901140	0.003718	0.0125	Reject the null hypothesis
HABC- TVG	11	3	1	2.796310	0.005169	0.0166	Reject the null hypothesis
HABC-PSTG	2	11	2	2.201398	0.027708	0.0250	Reject the null hypothesis
HABC-HSS	6	6	3	0.745815	0.455779	0.050	Retain null hypothesis

Table 11. Wilcoxon signed rank sum test for Table 6.

Pairs	Ranks			Test statistics			Conclusion
	ABCS <	ABCS >	ABCS =	Z	Asymp. Sig. (2-tailed)	α holm	
HABC- TVG	7	0	8	2.366432	0.017960	0.0100	Reject the null hypothesis
HABC-WHITCH	15	0	0	3.411211	0.000647	0.0125	Reject the null hypothesis
HABC- PICT	14	1	0	3.294179	0.000987	0.0166	Reject the null hypothesis
HABC-PSTG	3	2	10	0.674200	0.500184	0.0250	Retain null hypothesis
HABC-HSS	1	3	11	0.365148	0.715001	0.0500	Retain null hypothesis

7.Statistical Evaluation

Statistical analysis is another method to evaluate the proposed strategy in terms of effectiveness to assess the significance of strategy. The Wilcoxon signed-rank test is used to evaluate HABC strategy with existing strategies in the experimental (from Tables 2 to 6) with 95% confidence level (i.e., α=0.05). The main reason for adopting a Wilcoxon signed-rank test is to determine whether there is a statistical difference between the proposed strategy and the remaining strategies in the comparison. This test is ideal to measure the difference of the two sets.

In order to control the error rate due to multiple comparisons, Bonferroni-Holm correction was adopted for adjusting α value (i.e., based on Holm's sequentially rejective step-down procedure [62]). Depending on the first stored p-value (Asymp. Sig. (2-tailed)) in scaling in ascending order. Therefore, α Holm is adjusted based on:

$$\alpha\ Holm = \frac{\alpha}{M-i+1} \tag{7}$$

where M indicates the overall number of paired comparison and i indicate the test number.

There are three values to evaluate HABC; Ranks HABC>, HABC<, and HABC= are used. In other words, the results of the proposed strategy are greater, smaller or equal to the other existing strategies. Two values have a Statistical Test part; Asymp. Sig. (2-tailed) and Z. The value of Asymp. Sig. (2-tailed) indicates the significant difference between the two sets and that the value does not exceed α Holm. Regarding the Z value, it is out of the scope of this paper (i.e., not considered). The corresponding hypothesis is rejected if the value of the Asymp. Sig. (2tailed) alternatively, called P-value is less than α Holm. The second hypothesis if rejected, the test proceeds with the third and so on. As soon as a certain null hypothesis cannot be rejected, all the remaining hypotheses are retained as well. The strategies with N/A and N/S results are considered incomplete and ignored samples, as there is no available result for the specified test configuration. The complete statistical analyses are shown in Tables 7 to 11.

Statistical results based on Wilcoxon test for Tables 2 to 6 are presented in Tables 7 to 11. From Table 7, HABC shows there is a significant difference with other strategies in column Asymp. Sig. (2-tailed), except for CS and ITCH, which has a significant difference from HABC. From Table 8, HSS and CS results are considered "missing" due to unavailability of results when *t* value is 6. Although HABC performed better than PSTG, IPOG-D, Jenny, IPOG, CTE-XL, TVG, PICT and TConfig. However, ABC-TG and ITCH showed better significant difference from HABC. From Table 9, HABC showed there is a significant difference compared to WHITCH, TVG and PICT strategies except for the HSS and PSTG that showed a better significant difference from HABC. For the other existing strategies, the results are considered "missing" due to unavailable results or not supported to the certain configuration. Table 10 presents the test results of Table 5. HSS is shown to have a significant difference compared to HABC. However, HABC excelled with the WHITCH, GS, IPOG, TVG and PSTG strategies. Table 11 presents the test results of Table 6. PSTG and HSS are shown to have a significant difference compared to HABC. However, HABC excelled with the WHITCH, TVG and PICT strategies.

8. Threats to Validity

There are many threats that confront the validity of any study during the research. Therefore, it is necessary to reduce and validate these threats as soon as possible. Reducing these threats is not an impossible task, by designing and development experiments can achieve the mitigation of threats. There are many *t*-way testing strategies that can be compared with the proposed HABC strategy. However, a few strategies support the VSCA. These strategies have been selected in order to compare the performance of HABC results.

The effectiveness of the proposed HABC strategy is one of the most important threats to validity. As such, we cannot use statistical hypothesis tests method such as the t-test in order to evaluate and compare HABC strategy with other existing strategies for producing the VSCA because we do not have the source code of any of them. Based on the previously mentioned reason, we cannot make a comparison in terms of the efficiency of generating time with existing strategies. All VSCA strategies have been implemented in a different environment. Therefore, our benchmarking result has been based on the published results.

9. Conclusion

This paper proposed a hybridization of ABC algorithm and PSO algorithm. This hybridization is aimed to generate the perfect test set size for *t*-way and VSCA. The HABC strategy exploits the advantage of both ABC and PSO algorithm. The experiments revealed that HABC strategy delivers phenomenal performance in comparison to the old strategies for both *t*-way and VSCA except GS and SA due to sub-configuration regarding VSCA when performed on various problems. Whereas the generated result was competitive and close with SA and GS in terms of VSCA. As part of our future work, we plan to improve the performance of the HABC strategy for generating the better result and to support the constraints within variable strength interaction.

Nomenclatures

$C1\&C2$	Learning Vector
fit	Fitness Value
gbest	Dimension of best particle
n_e	Number Of Unemployed Honeybees
pbest	Personal best of particle
r1&r2	independent random numbers
rand	Scaling factor
V	Velocity of particle
X_i	Test case
$X_{i,d}^t$	Position of particle
X_{max}	Upper Boundary Parameter
X_{min}	Lower Boundary Parameter

Abbreviations

ABC	Artificial Bee Colony
ACS	Ant Colony System
AETG	Automatic Efficient Test Generator
AI	Artificial Intelligent
CA	Covering Array
CS	Cuckoo search
CT	Combinatorial Testing
CTE-XL	Classification-Tree Editor (eXtended Logics
HABC	Hybrid Artificial Bee Colony
HSS	Harmony Search Strategy
IPOG	In-Parameter-Order Generator
ITCH	Intelligent Test Case Handler
ITTDG	Integrated *t*-way Test Data Generation
MCA	Mixed Covering Array
OA	Orthogonal Array
PICT	Pairwise Independent Combinatorial Testing
PSO	Particle Swarm Optimization
PSTG	Particle Swarm Test Generator
SA	Simulated Annealing
SI	swarm intelligence
TCG	Test Case Generation Tool
TVG	Test Vector Generator
VS	Variable Strength
VSCA	Variable Strength Covering Array

References

1. Cohen, M.B.; Gibbons, P.B.; Mugridge, W.B.; Colbourn, C.J.; and Collofello, J.S. (2003). A variable strength interaction testing of components. *Proceedings of the 27th Annual International Computer Software and Applications Conference COMPSAC* Dallas, Texas, USA, 413-418.

2. Yilmaz, C.; Fouche, S.; Cohen, M.B.; Porter, A.; Demiroz, G.; and Koc, U. (2014). Moving forward with combinatorial interaction testing. *Computer,* 47(2), 37-45.
3. Nie, C.; and Leung, H. (2011). A survey of combinatorial testing. *ACM Computing Surveys (CSUR),* 43(2), 11.
4. Hervieu, A.; Marijan, D.; Gotlieb, A.; and Baudry, B. (2016). Practical minimization of pairwise-covering test configurations using constraint programming. *Information and Software Technology,* 71(1), 129-146.
5. Kacker, R.N.; Kuhn, D.R.; Lei, Y.; and Lawrence, J.F. (2013). Combinatorial testing for software: An adaptation of design of experiments. *Measurement,* 46(9), 3745-3752.
6. Kuhn, D.R.; Kacker, R.N.; and Lei, Y. (2013). *Introduction to combinatorial testing*. CRC press.
7. Ahmed, B.S.; and Zamli, K.Z. (2011). A variable strength interaction test suites generation strategy using particle swarm optimization. *Journal of Systems and Software,* 84(12), 2171-2185.
8. Zamli, K.Z.; Din, F.; Kendall, G.; and Ahmed, B.S. (2017). An experimental study of hyper-heuristic selection and acceptance mechanism for combinatorial t-way test suite generation. *Information Sciences,* 399(C), 121-153.
9. Huang, R.; Chen, J.; Zhang, T.; Wang, R.; and Lu, Y. (2013). Prioritizing variable-strength covering array. *Proceedings of the 37th Annual Computer Software and Applications Conference (COMPSAC),* Kyoto, Japan, 502-511.
10. Cai, L.; Zhang, Y.; and Ji, W. (2018). Variable Strength Combinatorial Test Data Generation Using Enhanced Bird Swarm Algorithm. *Proceedings of the 19th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD),* Busan, South Korea, 391-398.
11. Cohen, M.B.; Colbourn, C.J.; and Ling, A.C. (2003). Augmenting simulated annealing to build interaction test suites. *Proceedings of the 14th International Symposium on Software Reliability Engineering ISSRE,* Denver, CO, USA, 394-405.
12. Ahmed, B.S.; Gambardella, L.M.; Afzal, W.; and Zamli, K.Z. (2017). Handling constraints in combinatorial interaction testing in the presence of multi objective particle swarm and multithreading. *Information and Software Technology,* 86(C), 20-36.
13. Ahmed, B.S.; Zamli, K.Z.; and Lim, C.P. (2012). Application of particle swarm optimization to uniform and variable strength covering array construction. *Applied Soft Computing,* 12(4), 1330-1347.
14. Alsewari, A.A.; Alazzawi, A.K.; Rassem, T.H.; Kabir, M.N.; Homaid, A.A.B.; Alsariera, Y.A.; Tairan, N.M.; and Zamli, K.Z. (2017). ABC Algorithm for Combinatorial Testing Problem. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC),* 9(3-3), 85-88.
15. Alazzawi, A.K.; Homaid, A.A.B.; Alomoush, A.A.; and Alsewari, A.A. (2017). Artificial Bee Colony Algorithm for Pairwise Test Generation. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC),* 9(1-2), 103-108.
16. Yilmaz, C.; Cohen, M.B.; and Porter, A.A. (2006). Covering arrays for efficient fault characterization in complex configuration spaces. *IEEE Transactions on Software Engineering,* 32(1), 20-34.

17. Cohen, M.B.; Gibbons, P.B.; Mugridge, W.B.; and Colbourn, C.J. (2003). Constructing test suites for interaction testing. *Proceedings of the 25th International Conference on Software Engineering,* Portland, OR, USA, 38-48.
18. Alsewari, A.R.A.; and Zamli, K.Z. (2012). Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support. *Information and Software Technology,* 54(6), 553-568.
19. Cohen, D.M.; Dalal, S.R.; Parelius, J.; and Patton, G.C. (1996). The combinatorial design approach to automatic test generation. *IEEE software,* 13(5), 83-88.
20. Williams, A.W. (2002). *Software component interaction testing: Coverage measurement and generation of configurations*. University of Ottawa (Canada).
21. Bell, K.Z.; and Vouk, M.A. (2005). On effectiveness of pairwise methodology for testing network-centric software. *Proceedings of the 3rd International Conference on Information and Communications Technology,* Cairo, Egypt, 221-235.
22. Alsariera, Y.A.; and Zamli, K.Z. (2017). A real-world test suite generation using the bat-inspired t-way strategy. *Proceedings of the 10th Asia Software Testing Conference (SOFTEC2017),* Kuala Lumpur, Malaysia,
23. Alsariera, Y.A.; Alamri, H.S.; and Zamli, K.Z. (2017). A bat-inspired testing strategy for generating constraints pairwise test suite. *Proceedings of the The 5th International Conference on Software Engineering & Computer Systems (ICSECS),* Pahang, Malaysia,
24. Alsariera, Y.A.; Nasser, A.; and Zamli, K.Z. (2016). Benchmarking of Bat-inspired interaction testing strategy. *International Journal of Computer Science and Information Engineering (IJCSIE),* 7(71-79.
25. Alsariera, Y.A.; and Zamli, K.Z. (2015). A bat-inspired strategy for t-way interaction testing. *Advanced Science Letters,* 21(7), 2281-2284.
26. Alsariera, Y.A.; Majid, M.A.; and Zamli, K.Z. (2015). Adopting the bat-inspired algorithm for interaction testing. *Proceedings of the The 8th edition of annual conference for software testing,* Kuala Lumpur, Malaysia, 14.
27. Alsariera, Y.A.; Majid, M.A.; and Zamli, K.Z. (2015). SPLBA: An interaction strategy for testing software product lines using the Bat-inspired algorithm. *Proceedings of the 4th International Conference on Software Engineering and Computer Systems (ICSECS),* Kuantan, Malaysia, 148-153.
28. Alsariera, Y.A.; Majid, M.A.; and Zamli, K.Z. (2015). A bat-inspired Strategy for Pairwise Testing. *ARPN Journal of Engineering and Applied Sciences,* 10(8500-8506.
29. Jenkins (2003). Jenny. *http://www.burtleburtle.net/bob/math/.*
30. Hartman, A.; and Raskin, L. (2004). Problems and algorithms for covering arrays. *Discrete Mathematics,* 284(1), 149-156.
31. Hedayat, A.S.; Sloane, N.J.A.; and Stufken, J. *Orthogonal arrays: theory and applications*. Springer Science & Business Media, 1999.
32. Mandl, R. (1985). Orthogonal Latin squares: an application of experiment design to compiler testing. *Communications of the ACM,* 28(10), 1054-1058.
33. Lei, Y.; Kacker, R.; Kuhn, D.R.; Okun, V.; and Lawrence, J. (2007). IPOG: A general strategy for t-way software testing. *Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems,* Tucson, AZ, USA, 549-556.

34. Yan, J.; and Zhang, J. (2006). Backtracking algorithms and search heuristics to generate test suites for combinatorial testing. *Proceedings of the 30th Annual International Computer Software and Applications Conference, COMPSAC'06,* Chicago, IL, USA, 385-394.
35. Zamli, K. Z.; Klaib, M.F.; Younis, M.I.; Isa, N.A.M.; and Abdullah, R. (2011). Design and implementation of a t-way test data generation strategy with automated execution tool support. *Information Sciences,* 181(9), 1741-1758.
36. Tung, Y.-W.; and Aldiwan, W.S., (2000). Automating test case generation for the new generation mission software system. *Proceedings of the 2000 IEEE Aerospace Conference,* Big Sky, MT, USA, 431-437.
37. Cohen, D.M.; Dalal, S.R.; Fredman, M.L.; and Patton, G.C. (1997). The AETG system: An approach to testing based on combinatorial design. *Software Engineering, IEEE Transactions on,* 23(7), 437-444.
38. Williams, A. TConfig. download page [Online]. Available: http://www.site.uottawa.ca/~awilliam/ [Accessed 23 Dec 2014]
39. Younis, M. (2010). M*IPOG: A parallel t-way minimization strategy for combinatorial testing*. PhD. Thesis, School of Electrical And Electronics, Universiti Sains Malaysia.
40. Chen, X.; Gu, Q.; Li, A.; and Chen, D. (2009). Variable strength interaction testing with an ant colony system approach. *Proceedings of the 16th Asia-Pacific Software Engineering Conference, APSEC'09,* Penang, Malaysia, 160-167.
41. Lei, Y.; Kacker, R.; Kuhn, D.R.; Okun, V.; and Lawrence, J. (2008). IPOG/IPOG - D: efficient test generation for multi - way combinatorial testing. *Software Testing, Verification and Reliability,* 18(3), 125-148.
42. Othman, R.R.; and Zamli, K.Z. (2011). ITTDG: Integrated T-way test data generation strategy for interaction testing. *Scientific Research and Essays,* 6(17), 3638-3648.
43. Wang, Z.; Xu, B.; and Nie, C. (2008). Greedy heuristic algorithms to generate variable strength combinatorial test suite. *Proceedings of the Eighth International Conference on Quality Software,* Oxford, UK, 155-160.
44. Keith & Doug, H. (2006). PICT. http://testmuse.wordpress.com/2006/04/05/pict-tool-available/2006/
45. Czerwonka, J. (2006). Pairwise testing in real world. *Proceedings of the 24th Pacific Northwest Software Quality Conference,*
46. Bryce, R.C.; and Colbourn, C.J. (2009). A density - based greedy algorithm for higher strength covering arrays. *Software Testing, Verification and Reliability,* 19(1), 37-53.
47. Bryce, R.C.; and Colbourn, C.J. (2007). The density algorithm for pairwise interaction testing. *Software Testing, Verification and Reliability,* 17(3), 159-182.
48. Ziyuan, W.; Changhai, N.; and Baowen, X. (2007). Generating combinatorial test suite for interaction relationship. *Proceedings of the Fourth international workshop on Software quality assurance: in conjunction with the 6th ESEC/FSE joint meeting,* Dubrovnik, Croatia, 55-61.
49. Ahmed, B.S.; Zamli, K.Z.; and Lim, C.P. (2012). Constructing a t-way interaction test suite using the particle swarm optimization approach. *International Journal of Innovative Computing, Information and Control,* 8(1 A), 431-451.

50. Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical report-tr06*, Erciyes university, engineering faculty, computer engineering department, Vol. 200.
51. Passino, K.M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE control systems,* 22(3), 52-67.
52. Eberhart, R.C.; and Kennedy, J. (1995). A new optimizer using particle swarm theory. *Proceedings of the sixth international symposium on micro machine and human science,* Nagoya, Japan, 39-43.
53. Karaboga, D.; and Akay, B. (2009). A survey: algorithms simulating bee swarm intelligence. *Artificial Intelligence Review,* 31(1-4), 61-85.
54. Yan, X.; Zhu, Y.; and Zou, W. (2011). A hybrid artificial bee colony algorithm for numerical function optimization. *Proceedings of the 11th International Conference on Hybrid Intelligent Systems (HIS),* Melacca, Malaysia, 127-132.
55. Kıran, M.S.; and Gündüz, M. (2012). A novel artificial bee colony-based algorithm for solving the numerical optimization problems. *International Journal of Innovative Computing, Information & Control,* 8(9), 6107-6121.
56. Alqattan, Z.N.; and Abdullah, R. (2013). A comparison between artificial bee colony and particle swarm optimization algorithms for protein structure prediction problem. *Proceedings of the 11th International Conference on Neural Information Processing,* Daegu, Korea, 331-340.
57. Esfandyari, S.; and Rafe, V. (2018). A tuned version of genetic algorithm for efficient test suite generation in interactive t-way testing strategy. *Information and Software Technology,* 94(165-185.
58. Alazzawi, A.K.; Rais, H.M.; and Basri, S. (2018). Artificial Bee Colony Algorithm for t-Way Test Suite Generation. *Proceedings of the 4th International Conference on Computer and Information Sciences (ICCOINS),* Kuala Lumpur, Malaysia, 1-6.
59. Arshem, J. (2010). TVG. *http://sourceforge.net/projects/tvg,*.
60. Ahmed, B.S.; and Zamli, K.Z. (2010). PSTG: a t-way strategy adopting particle swarm optimization. *Proceedings of the the Fourth Asia International on Mathematical/Analytical Modelling and Computer Simulation (AMS),* Bornea, Malaysia, 1-5.
61. HOMAID, A.B.; ALSWEARI, A.; ZAMLI, K.; and Alsariera, Y. (2018). Adapting the Elitism on the Greedy Algorithm for Variable Strength Combinatorial Test Cases Generation. *IET Software*.
62. Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian journal of statistics,* 65-70.