Hybridized BA & PSO *t*-way Algorithm for Test Case Generation

Yazan A. Alsariera[1†], Ahmed H. Al Omari[1], Mahmoud A. Albawaleez[2], Yousef K. Sanjalawe[1], Kamal Z. Zamli[3]

[1] Department of Computer Science, Collage of Science, Northern Border University, Arar, Saudi Arabia.
[2] Deanship of Information Technology, Northern Border University, Arar, Saudi Arabia.
[3] Faculty of Computer Systems and Software Engineering Universiti Malaysia Pahang, Kuantan, Malaysia

Summary
Software testing is an essential part of the software development life cycle. However, due to limited time and resources, extensive testing of highly configurable software is impractical. In addition, extensive testing can lead to combinatorial explosion problems, where test cases grow exponentially with the increase in software input. Because of their effectiveness in detecting errors, many researchers are turning to a sampling strategy based on input interactions, called t-way testing, where t represents the strength of the interaction. It is known to be an NP-complete problem (ie, non-deterministic polynomial time). Due to the potentially large search space generated when dealing with large input values, the process of minimizing t-way test cases is challenging. So far, many designs and strategies utilizing the t-way methods have been proposed in the literature. Recently, researchers have advocated the use of emerging of fields to call search-based software engineering (SBSE), which utilizing the metaheuristic-based search algorithm. Therefore, in this research, a new hybrid metaheuristic-based is proposed based on merging the advantages of BA and PSO. Mainly, the design and implementation of a new hybrid metaheuristic- based t-way strategy is presented called , BAPSO, based on the two well-known algorithms, namely, Bat-inspired algorithm and Particle Swarm Optimization algorithm. BAPSO strategy can contribute to the field of software testing, as it achieved comparable performance in terms of minimizing the number of test cases used for test execution.

Keywords:
t-way interaction testing, Bat-inspired algorithm, Particle Swarm Optimization algorithm, Software testing.

1. Introduction

Combinatorial interaction testing is being investigated dramatically in the last few decades. Interaction testing approaches are sharing quite a lot of similarities. In fact, Interaction testing has been established for a specific target, that is the process of minimizing a large and complex search space for a targeted domain problem [1]. Several interaction testing approaches and software tools been developed for a potential large and complex search space area including software testing. These approaches referred to as combinatorial interaction testing. its testing technique introduced to generate and evaluate the test data for system-under-test (SUT) [2]. combinatorial interaction testing considered a functional test generation method that uses a specification model to evaluate SUT. It uses the interaction among parameters and their values as test factors for generating the test of a particular system domain. combinatorial interaction testing effectiveness is observed based on the system failures that often happens due to interactions t among few system parameters [3, 4], t refers to as the number of interaction strength. This observation proved by NIST study throughout a several real-life systems, that is all system faults happened on the level of interacting no more than six system factors [5]. combinatorial interaction testing technique covers all t-way combinations of system parameters to evaluate and detect faults resulted by interactions of t or less components [6]. in practically, pairwise testing, or two-way testing is the most often used interaction strength. In case of pairwise testing, all pair of parameter values are covered at least once in test suite. For example, a system with three parameters; A, B and C, has three pairs; AB, AC, and BC. Although worth to mention that pairwise is balancing both the efficiency and time performance for most of the real-world case studies. However, in case of higher t number, referred to as higher interaction strength the efficiency and performance is traded-off normally, this trade of is worth to be taken in order to covers all the possible test interaction. In this article, we have motivated to design and implement a t-way strategy that support high interaction strength. The strategy supports high interaction strength t up to 6. Likewise, it capable of providing high effective and efficient test suite. We have adopted a search-based test generation using a hyper method of Bat-inspired algorithm (BA) and Particle Swarm Optimization algorithm (PSO) to explore the possible test data and find the applicable test suite for a given software configurations. BAPSO gives us the capability to randomly generate a set of test cases and improve the quality of the test cases in controlled process progressively. Also, discussing the design of Bat representations of test configurations and the definition of a Bat weight function or called fitness function. The strategy efficiency will be shown throughout a comparative experiment of interaction testing benchmarking configurations.

Manuscript received October 5, 2021
Manuscript revised October 20, 2021
https://doi.org/10.22937/IJCSNS.2021.21.10.48

The emerging field of search-based engineering [7] currently allows many researchers to adopt combinatorial interaction testing search-based strategies. The adoption of an optimization algorithms as backbone search engine for combinatorial interaction testing is successfully effective in interaction testing [8], Such as Late Acceptance Hill Climbing constraints based Strategy (LAHC) [9, 10], evolutionary genetic algorithm based strategy (GA) [1], Greedy [11], harmony search based Strategy [12, 13], Artificial Bee Colony [14-17]. Optimization algorithm-based strategy seems to be superior. The use of meta-heuristic optimization algorithms such as BA and PSO as a backbone search engine for combinatorial interaction testing strategies seems promising and its variant test generation strategies [18-23]. The BA is considered a superior optimization algorithm as BA produces significantly better results than most popular optimization algorithms, including the GA, particle swarm optimization (PSO), and SA. Moreover, the BA is easy to implement, and its parameters are highly adjustable to fit many engineering solutions. PSO will fit the exploration problem as it will enhance the converges toward optima. Thus, the use of BA as the main algorithm is suggested and PSO to enhance the exploration of BA.

As a result, this study proposes and evaluates a test suite generation strategy based on combinatorial interaction testing concepts using BA and PSO optimization algorithm for high interaction. The contributions of the BAPSO based strategy are summarized as follows:

- the analysis the advantage of BA and PSO over the currant implemented meta-heuristic algorithms for interaction testing research in practice.
- the provide a method of implementing a meta-heuristic optimization algorithm for interaction testing based that support a high interaction strength up to $t = 6$.
- We evaluate the strategy based on the test suite efficiency throughout a comparative experiment of interaction testing benchmarking configurations. Furthermore, we achieved a good quality solution in comparison with existing uniform interaction testing strategies and provide new level of benchmarking results in terms of test suite sizes.

The result of our benchmarking experiment based on a comparison with existing combinatorial interaction testing generation strategies, we show the usefulness of adopting the BA and PSO.

The remaining section of the paper includes Section 2 which explain the Combinatorial interaction testing notations and its processing as used in this study. It also contains to Section 3 which discusses Uniform interaction testing and its design and reveal the needed definitions. Section 4 reviews the BA and PSO algorithms. Section 5 implemented hybridized BAPSO and shows the optimization process and pseudocode respectively. Finally, Section 6 presents the experimental framework and the experimental setup for conducting the benchmarking comparative analysis results which include the obtained sizes (performances) of the proposed methods. Conclusively, section 7 provided the conclusion of the study and the future work.

2. Combinatorial interaction testing notations

Empirical research results indicate that software systems failure is mostly triggered by interactions among t parameters from $(2 \leq t \leq 6)$ [4, 5]. combinatorial interaction testing is a method capable of constructing the test suite that covers all system parameter-value at all t interacted parameters on the constructed test suite. To represent the combinatorial interaction testing, normally a covering array (CA) notations used as mathematical concept to construct test suite for the targeted system configurations. The notation CA has four main parameters, namely, N, t, p, and v (i.e., $CA\ (N, t, p, v)$). CA is a matrix of size $N \times p$. Here, the symbols t refer to the interaction strength, N represents the test cases (rows), p is known as number of parameters (columns) and v refer to the number of CA values for a specific p. For instance, $CA\ (N, 2, 4, 3\ 3\ 3\ 3)$ equivalents to $CA\ (N, t, v^p)$ that is indicated as $CA\ (N, 2, 3^4)$. Here, CA involving $N \times 4$ array that covers the test suite. In this case, the test suite covers pairwise interaction strength with three v values and four p parameters, N is $3 \times 3 = 9$ test cases. this case represents the most minimum covering array see Figure 1a, that is can be represented by optimal value of N as in eq. (1) based on the definition of CAN [24].

$$CAN\ (t, v^p) = \min\ [\exists | \lambda\ CA\ (N, t, p, v)] \quad (1)$$

$CA\ (9, 2, 2^4)$

P_1	P_2	P_3	P_4
v_2	v_2	v_3	v_2
v_1	v_2	v_1	v_3
v_3	v_1	v_3	v_3
v_1	v_3	v_3	v_1
v_2	v_3	v_2	v_3
v_3	v_2	v_2	v_1
v_2	v_1	v_1	v_1
v_3	v_3	v_1	v_2
v_1	v_1	v_2	v_2

(a)

$MCA\ (9, 2, 2^2\ 3^2)$

P_1	P_2	P_3	P_4
v_2	v_1	v_1	v_1
v_2	v_2	v_2	v_2
v_1	v_2	v_3	v_1
v_1	v_1	v_3	v_3
v_1	v_1	v_1	v_2
v_2	v_1	v_3	v_3
v_1	v_2	v_1	v_3
v_1	v_1	v_2	v_1
v_1	v_1	v_3	v_2

(b)

Figure 1. Illustration of $CA\ (9, 2, 3^4)$ and $MCA\ (N, 2, 2^2\ 3^2)$.

Nonetheless, systems usually have different values for each of its components. Therefore, mixed covering array (MCA) is introduced. To represent each component as individual test parameter has its own values. MCA indicated by $MCA\ (N, t, p, v_1 v_2, \ldots, v_n)$ is here the only deferent is that

v_n is a specified values for related p parameter. Whereas, For every single column $v_i \in (1 \leq i \leq p)$ contains elements of the set $|V_n| = v_n$. The raw of each submatrix are contains $N \times t$ interaction elements that cover all t-tuples matched from the related t columns at least once. Similarly to CA, MCA is indication of $MCA\ (N, t, p, v_1^{x1} v_1^{x2}, \dots, v_n^{xi})$, where p as in eq. (2).

$$p = \sum_{i=1}^{n} x_n \qquad (2)$$

In here, each x_i parameter has its own v_n value, for example, a test suite covers pairwise interaction strength with four p parameters, first two parameters have two values and the rest two has three values. MCA is formulated as $MCA\ (N, 2, 2^2\ 3^2)$. In case of the most optimal value of N, MCA is represented as in eq. (3), see Figure. 1b for the most optimal test suite.

$$MCAN(t, v_1^{p_1}\ v_2^{p_2}, \dots, v_n^{p_i}) = \min[\exists|\lambda\ MCA\ (N, t, p, v_1 v_2, \dots, v_n)] \quad (3)$$

As instance of $CCA(N, 2, 2^4, C)$ where $C = [(x, 0,0, x), (1,1, x, x)]$. In this case, the CCA indicates the test size of N for pairwise interaction of four parameters each has two values with constraints pair interaction elements of $(x, 0,0, x)$ and $(1,1, x, x)$, where "x" indicates the ''don't care'' values. Here, regardless of the ''don't care'' values, any test cases covering the interaction elements of $(x, 0,0, x)$ and $(1,1, x, x)$ are not allowed.

3. Uniform interaction testing

In net shell, combinatorial interaction testing is dealing with the interaction among the system features and their dependence values. Consider a simple setting menu for a connectivity in a smart device has three features each has two values as system features in Figure 2. The representation of pairwise interaction strength $(t = 2)$ using covering array notation is a $CA\ (N;\ 2, 2^3)$. In this case, there are three input features resulting of three pairs configurations: $[\ (Wi-Fi, Network)\ ,\ (Wi-Fi, Bluetooth), (Network, Bluetooth)]$. Referred to as interaction tuples. Here, in this example we $Network$ is related to network connectivity (internet connection).

	System features		
	Wi-Fi	***Network***	***Bluetooth***
values	On	On	On
	Off	Off	Off

Figure 2. Connectivity menu system features.

Here, the exhaustive combinations at $t = 3$ contains eight test cases, but at two interaction strength $t = 2$ a four test cases could be achieved as the most optimal test suite without considering constraints values, that is covers all the possible test configurations. Therefore, 50% of the exhaustive test suite is reduced by applying pairwise interaction.

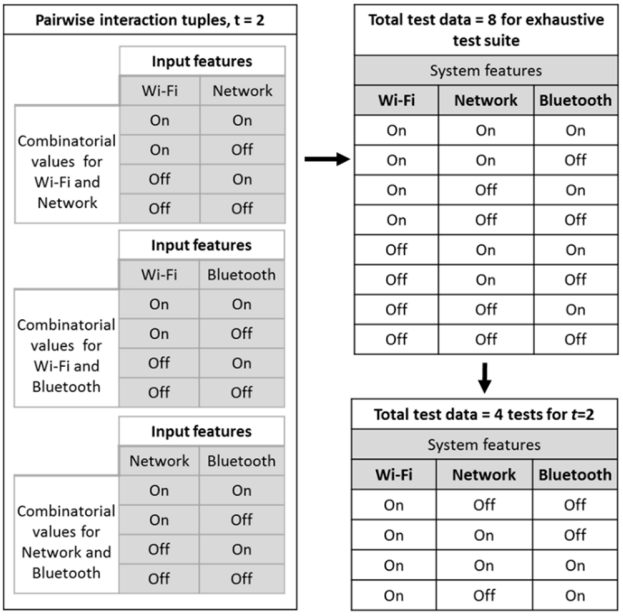


Figure 3. test generation for the connectivity system features.

Here, based on the above overview, there is a need to define the terminology for testing based on combinatorial interaction testing concepts.

Definition 1. An element list (EL) that contains the element model. Here, we use the term element as a synonym of feature. As in combinatorial interaction testing terminology. The EL is;

$$EL = [Wi-Fi, Network, Bluetooth].$$

Definition 2. Element set (es) that contains the t based pairing for the EL elements $\in [vel, \overline{vel}\ ]$, where $es.vel$ and $es.\overline{vel}$ are the selected and not selected elements respectively. Let EL be an element list, thus $vel, \overline{vel} \subseteq EL, vel \cap \overline{vel} = \emptyset$, and $vel \cup \overline{vel} = EL$.

Table 1. sample of valid element set for mobile game system.

***ie* sets**	***Wi − Fi***	***Network***	***Bluetooth***
es 1	✓	✓	
es 2	✓		✓
es 3		✓	✓

Definition 3. Combination's element list (CEL) that is a list contains all the valid element set. es is valid within an element list EL. Thus, CEL will contains all the valid combination elements. For example, the in case of pairwise interaction, [(Wi-Fi, Network), (Wi-Fi, Bluetooth), (Network, Bluetooth)], based on the interaction value using to generate all required interaction elements that covers all the test suite.

Definition 4. Interaction element list (IEL). A t based interaction covering array list has a t-tuples sets $[vel, \overline{vel}]$, representing all the elements configurations of element list CEL.

Definition 5. Valid Interaction element list. A covering array contains all the IEL list elements of the targeted model.

4. The BA and PSO

Yong [25] has developed a new nature inspired meta-heuristic optimization technique based on the observation of the hunting behavior of micro bats in nature, called bat algorithm (BA). In order to mimic the behavior of micro-bats in the simplest way, some approximations are necessary, Yong [25] has idealized three assumptions, as follows:

- For sensing the distance ahead of their flight paths, micro-bats use echolocation, and they have their unique instinct to distinguish a target of food/prey from the background barriers.
- During hunting, bats may travel in a random manner at a velocity v_i at position x_i with a combination of sensing frequency Q_{min}, with varying wavelength λ and loudness A_0 to hunt for prey. The frequency (or wavelength) of their emitted pulses can be automatically adjusted, and the pulse emission rate $r \in [0,1]$ can also be fine-tuned automatically, given the current proximity of their target.
- The loudness of echoes by a micro-bat can be assumed to decay over time from a large and positive amplitude A_0 to a small constant value A_{min}.

Algorithm 1: BA Algorithm

Input: Objective function $f(x_i)$, $x_i = (x_{i1}, \dots, x_{iD})^T$
Output: Best fitness x̂.

```
Initialize the bat population x_i and velocities v_i for i = 1 ... NP
Define pulse frequency Q_i ∈ [Q_min , Q_max]
Initialize pulse rates r_i and the loudness A_i
while (t < T_max) do
    for each bat n_i do
        Generate new solutions by adjusting frequency, and update
        velocities and location / solutions using motion equations.
        if (rand(0,1) > r_i) then
            Select the best solution in the current population.
            Generate a local solution around the best solution.
        end
        if (rand(0,1) < A_i and f(x_i) < f(x)) then
            Accept the new solutions.
            Increase r_i and reduce A_i.
        end
        Rank the bats and find the current best.
    end
end
process results and visualization
```

Figure 4. BA pseudo code.

In algorithem 1, the process of the BA presanted by Yong [25], the first step is to initial the algorithm variables randomly, for each bat n_i; initial location (solution) x_i, initial velocity v_i and initial frequency Q_i, at each time t. Define the maximum of number of generation T_{max} , referred to as iterations. virtual bats n_i movement are given by the bat motion Eqs. (4) – (6) that are corresponded for updating each bat velocity v_i and location x_i, based on a updating frequency Q_i by each cycle of iterations. The pace and range of the movement are basically controlled by Q_i , just like the movement of the swarming particles. As follows:

$$Q_i = Q_{min} + (Q_{max} + Q_{min})\, rnd, \tag{4}$$

$$v_i^{t+1} = v_i^t + (x_i^t - x_*)Q_i, \tag{5}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}, \tag{6}$$

In Eq. (4), rnd indicates a vector randomly generated within the interval $[0, 1]$, that control the speed and range of the new velocity at specific time step t in Eq. (5) for each bat. The new location of each bat is processed by the movement of the current location and the updated velocity of the specific bat n_i at the new time using Eq. (6). Here, the new solution is considered as a currant global best to be compared for improvement with the best solution. After comparing all the solutions among all the n bats at each iteration for t, a current best solution x_* that represents the global best solution can be obtained. This current best solution corresponds to the best value of the objective at the current iteration for t. In general, the velocities and the locations of bats are updated slightly like particle swarm optimization (PSO).

Yong [25] has employ a local search approach to improve the effectiveness and efficiency of the potential solutions, namely, random walks. a new solution is selected locally by using a random walk around the current best solution. In order to enhance the quality of the global best solution. the new solution tested to see if it is the best among all the solutions based on the random vector condition, the random vector at the time step t for the bat n_i at cycle i must be greater than pulse emission rate r_i for the associated bat. The random walk is defined as follows:

$$x_{new} = x_{best} + \epsilon\, A^t \tag{7}$$

in which, $A^t = < A_i^t >$ stand for the average loudness of all the bats at time step t. ϵ drawn from $[-1, 1]$ as a random vector control the direction and strength of the random walk. To certain extent, the bat algorithm is deemed to be a balanced combination of swarm optimization and the

intensive local search, that are governed by the frequency tuning ability and the variables of loudness and pulse rate. Thus, for each iteration of BA, the loudness A_i and the emission pulse rate r_i are updated, as follow in Eqs. (8) and (9);

$$A_i^{t+1} = \alpha A_i^t \quad (8)$$

$$r_i^t = r_i^0[1 - \exp(-\gamma t)] \quad (9)$$

In which, a and γ are BA constant parameters that are has a similar effect like cooling factor in a cooling schedule as in the simulated annealing algorithm, in the range of $0 < a < 1$ and $\gamma > 0$. Except the case in Eq. (10).

Actually, a has the effect of the cooling factor in a cooling schedule as in the simulated annealing algorithm. For any $0 < a < 1$ and $\gamma > 0$. generally, A_i^0 and r_i^0 are randomly chosen to drown from $A_i^0 \in [1, 2]$ and $r_i^0 \in [0, 1]$. the loudness A_i and the pulse emission rate r_i only updated when new solutions are improved, that means, the bats are moving towards the optimal solution. Therefore, BA shows good exploration and exploitation mechanism capabilities. however, every meta-heuristic algorithm applies a different strategy of exploration and exploitation for optimization problems. Emerging hybrid algorithm research on optimization problems has been attracting attention algorithm to overcome the problem of the exploration and exploitation or enhance their outcomes. Thus, we proposed a strategy based on merging the advantages of BA and PSO. Thus, this paper proposes a new algorithm called Hybridized BAPSO which utilizes the exploitation ability of PSO (i.e. local search process) and the exploration ability of BA (i.e. global search process). While the original PSO has considerable exploitation ability and a fast convergence speed [26], it has poor exploration ability. By contrast, the original BA, has great exploration ability but poor exploitation mechanism abilities strategies [18, 19].

5. Optimization process with BAPSO

The BA [25] efficiently solves several related engineering optimization problems. This algorithm can improve solution quality because of the global and local search behavior it implements. BA is considered the nature-inspired population-based meta-heuristic optimization algorithm that is based on the observation of the hunting behavior of micro bats in nature [27]. Micro bats use echolocation to find their prey in complete darkness. Nature interpretation is typically not perfect. Therefore, BA is based on swarm optimization with the path algorithm. For swarm behavior, several bats (known as populations) are considered. Each bat has its own position and velocity at predefined dimensions to find the best solution. This behavior is a replicate of that from PSO, in which a swarm population is used to find a solution. An exhaustive local search method is provided throughout; its a random walk behavior in each iteration cycle around the best solution found. The local search behavior around the appropriate solutions to find the best overall solution is based on path algorithms. The BA also adopts a homogeneous path approach similar to SA. This approach considers several variables, such as bat size and emission pulse rates, which function similar to the temperature and cooling factor in SA. The BA is conventionally developed on the assumption that a bat can locate its prey in complete darkness. Applying this

$$A_i^t \to 0, \qquad r_i^t \to r_i^0, \qquad as\ t \to \infty. \quad (10)$$

meta-heuristic algorithm for interaction testing, we assume the test cases as the bat position, which is a possible solution of the problem. BA searching provides an optimum test case (or global optimum), which indicates solution quality in terms of the best bat position from its prey. Bats avoid obstacles using echolocation. Thus, different frequencies are returned in each iteration.

In the past several years, many meta-heuristic algorithms such as Particle Swarm Optimization (PSO) [28], Differential Evolution (DE) [29, 30], Genetic Algorithm (GA) [31, 32], etc., have achieved significant success in solving optimization problems.

The local search (i.e. exploitation ability process) for PSO particles depends on three categories: velocity variable, local best solution and global best solution. The next move of each particle depends on the velocity variable, which means the particle's movements are not random. Which might help extend the ability of exploitation process of the BA. The local information comes from the local best solution variable that interacts with the particle's next move value. The best solution discovered is not permanently held inside the population, because it may be replaced with another one that is produced randomly by a Bats. The global best solution variable has a considerable effect on the next move of a particle. Which might help to avoid the early convergence that might happiness in some cases. The best solution discovered by PSO is permanently held inside the population and is used again to find a new solution, but with a new velocity that change the velocity of the main algorithm. The search space is searched again over a short period of time and in detail, accordingly. On the other hand, the PSO algorithm can be trapped in the local minima. For this reason, the limit parameter still exists in the proposed algorithm. The limit parameter can prevent the original BA trapped due to this process by inserting a randomly selected solution into the search space from time to time.

Strategy 1: optimization process with bat-inspired algorithm
Input: covering array notation.
Output: final test suite list (FTSL).

1 Define $CEL, IEL, BEL, FTSL, conEL$ lists in the memory.

```
2  Initialize BA and PSO attributes; nbats, nGeneration,
   Loudness (A_0), pulse emission r_0, Q_min, Q_max,
   tolerance, w and c.
3  Initialize generation of interaction element list generation
   algorithm.
4  Initialize BA population.
5      while (IEL is not empty) do
6  while (T < nGeneration) do
7  evaluate fitness for all E_i.
8  select maximum fitness E_i as best bat (best solution).
9      for all nbats do
10         calculate frequency Q_i.
11         get the v_{i+1} according to the old v_i ,x_i and Q_i.
12         move x_i to x_{i+1} according to the new v_{i+1}.
13             if (best x_{i+1} cover the maximum interaction
               elements) do
14             accept the new solutions Best x_i = Best
               x_{i+1}.
15             Increase r_i and reduce A_i.
16             else get new v_{i+1} based in PSO then check
               coverage if cover the maximum accept the
               new solutions Best x_i = Best x_{i+1}.
17             end
18            accept Best x_{i+1} test case as best case
19            End
20          evaluate fitness for all E_i.
21          end
22         if (best bat is constraints element) then
22         return step 9.
23     else
24     add best test case to FTSL.
25     remove the covered E_i from IEL.
26    end
27    end
28 Process results and visualization
```

Figure 5. Pseudo code of optimization process with bat-inspired algorithm

Based on the aforementioned algorithms in section 4, the BAPSO test suite generation strategy can be explained as follows (see also Figure 5):

- Stage 1. Initialize the strategy lists and BA and PSO algorithms attributes. in this stage, a combination elements list (CEL), interaction elements list (IEL), binary elements list (BEL), final test suite list ($FTSL$) and constraints elemants list ($conEL$) is defined in the memory. Then, Initialize BA algorithm attributes; Population size (number of bats), Iteration (number of generation), Minimum frequency (Q_i), Maximum frequency (Q_i), Loudness (A_0), Rate of pulse emission (r_0) the tolerance (t), weight value (w) and cyclic walk value (c). Then, Initialize the generation of interaction tuples, that covers all the possible elements sets in the targeted problem. The optimization problem can be specified as follows:

$$f(x) = |[ie \text{ in } IEL: x \text{ covers } ie]| \quad (6)$$

The $f(x)$ is fitness function subject to $x = x_1, x_2, \dots, x_n$ in $E_1, E_2, \dots, E_n$; $n = 1, 2, \dots, N$ uses to evaluate the fitness of the covering interaction by each generated solution. Here, x is a set of interaction elements to evaluate the coverage of ie in the non-covered interaction in IEL. the vertical bars $|\ |$ represent the cardinality of the set and the objective value is the number of non-covered interaction elements covered by x. E_i is the set of possible range of values for each result, N is the number of bats population.

- Stage 2: Initialize BA population based on the interaction elements list, BA population is Initialized randomly for the first iteration. The bats are considered as an interaction elements E_i Initialize and their values v_i are considered as bats location or known positions x_i. In this stage all the bats are having Initial velocity v_i based on the BA, the fitness and frequency of zero.
- Stage 3: the evaluation of the generated solutions based on the objective function until all the interaction elements in IEL are covered, that means IEL is got empty. Here, the maximum coverage elements selected as best solution to be improved by the evaluation of the solution on the motion movements of the bats.
- Stage 4: improve the solution based the motion movements of the using the Eqs. 6-8 mentioned in section 4, depending on the following; a frequency is calculated for the specified bat that is control it new velocity based on the best solation location and the old location of the targeted bat. This changes the bat (E) to a new location. This process goes for all the bat in the population. Then, all the bats are evaluated using the fitness function to see if a new best has achieved.
- Stage 5: the best solution is being locally improved by finding a new solution around the best. If only the new founded solution is covering more interaction elements in IEL is considered as new best. In this case we increase the rate of emission pulse and reduce the loudness. If not the PSO change the velocity to try enhancing the best solution, then, if only the new founded solution is covering more interaction elements the new best is considered. The best solution is accepted, then the best solution (test case) is added into the final test suite and the covered interactions are removed from the IEL.
- Stage 7: Check Exit Criteria for interaction coverage when all the interaction elements are covered (i.e., the interaction elements list is empty), the iteration stops. Otherwise, generation process is continuing from Step 2.

3. Results and Discussions

The evaluation stagy of the BAPSO combinatorial interaction testing strategy is shown in this section, The strategies used in the comparative analysis are available publicly as test generation tools that can be download and

installed. we consider some strategies that available from previous works for compression. also, we consider some of the results that are published for certain test profiles to cover as much as possible test generation strategies. In fact, we conduct the evaluation for strategies that are available for benchmarking within fair evaluation setting. we consider the test suite sizes published only. The efficiency of the test suite size is not affected by the hardware used. In other hand the performance can be effected by hardware such as speed of the processer, bus speed and ram capacity. Therefore, conducting all the test generation strategies in the same hardware is essential to ensure a fair comparison of performance as the generation time is affected by the specifications of the work environment hardware.

The main experimental results aim is to benchmark the test suite sizes of our strategy against existing combinatorial interaction testing strategies to validate the efficiency of the former strategy. The experiments are conducted on a laptop with Windows 10 professional and hardware specification; Intel Core ™ i7-8565U 2.00 GHz, 8 GB RAM. The strategy is developed in Java SE 64 bits (JDK 1.8). The results of the best test suite size (*N*) and the average size (*avg.N*) reported based on 10 number of executions for each configurations. The highlighted cells indicate the best obtained test suite size result for the configuration of interests. Cells marked NS (not supported) indicate that the strategy does not support the specified interaction degree. Cells marked NA (not available) indicate that the results are not available in the publications.

Table 2. Variables used for the optimization algorithm based existing strategies.

Optimization algorithm	Variables	Values
BA	Population size	100
	Loudness	0.9
	Rate of pulse emission	0.9
	Minimum frequency	0
	Maximum frequency	1
	tolerance	0.025
	Iteration	100
PSO	Weight Value	0.4
	Cyclic Walk Value	1.49

Table 3. Minimum test suite for uniform strength test profile $CA(N;\ 4, 5^P)$ with varied parameters from 5 to 10, each has with fixed 5 values

	GVS	GTWay	TVG	TConfig	Jenny	ITCH	IPOG	BAPSO	
P	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***Avg.N***
5	733	731	849	773	837	625	784	736	741.00
6	1012	1027	1128	1092	1074	625	1064	965	972.70
7	1215	1216	1384	1320	1248	1750	1290	1213	1222.4
8	1398	1443	1595	1532	1424	1750	1491	1381	1394.6
9	1556	1579	1795	1724	1578	1750	1677	1508	1510.3
10	2294	2332	NA	NA	NA	NA	2497	1746	1858.2

Table 4. Minimum test suite for BA uniform strength test profiles compared to AETG, mAETG, ACS, CS, GA, PSO and SA.

	AETG	mAETG	ACA	CS	GA	PSO	SA	BAPSO	
Test Profile	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***Avg.N***
$CA\ (N;\ 2, 3^4)$	9	9	9	9	9	9	9	9	9.000
$CA\ (N;\ 2, 3^3)$	15	17	17	20	17	17	16	18	18.80
$CA\ (N;\ 3, 3^6)$	47	38	33	43	33	42	33	33	37.20
$CA\ (N;\ 3, 4^6)$	105	77	64	105	64	102	64	64	70.00
$CA\ (N;\ 3, 5^7)$	229	218	218	233	218	229	201	217	219.4
$CA\ (N;\ 3, 6^6)$	343	330	330	350	331	338	300	322	328.0
$MCA\ (N;\ 2, 5^1 3^8 2^2)$	19	20	16	21	15	21	15	20	21.60
$MCA\ (N;\ 2, 6^1 5^1 4^6 3^8 2^2)$	34	35	32	43	33	39	30	39	40.80
$MCA\ (N;\ 2, 7^1 6^1 5^1 4^6 3^8 2^2)$	45	44	42	51	42	48	42	49	50.90
$MCA\ (N;\ 3, 10^1 6^2 4^3 3^1)$	NA	377	361	393	360	385	360	381	390.5

Table 5. Minimum test suite and exaction time for uniform strength test profile $CA\ (N;\ t, 3^7)$ with varied interaction straingthe up to t = 6.

	Jenny	TConfig	ITCH	PICT	TVG	CTE-XL	IPOG-D	IPOG	PSO	BAPSO	
t	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***N***	***Avg.N***
2	16	15	15	16	15	16	18	17	15	15	15.00
3	51	55	45	51	55	54	63	57	50	49	50.90
4	169	166	216	168	167	NS	NS	185	155	154	157.0
5	458	477	NS	452	464	NS	735	608	441	436	440.2
6	1087	921	NS	1015	1016	NS	1548	1281	977	920	935.9

Referring to experimental sets 1 (Table 4) the BAPSO strategy reveals good performance and efficiency compare to GVS, GTWay, TVG, TConfig, Jenny, ITCH, and IPOG. Here, BAPSO ware able to aciave minimum test suite in 4

parmaters configrations, where P are 7, 8, 9, and 10, respectively. In contrast for the low parameters ITCH still have the best suite. In experimental sets 2 (Table 5) have conducted with AETG, mAETG, ACA, CS, GA, PSO, and SA. Here, the BAPSO shows good result in the first, third and fourth configurations, AETG still have the edge for the second configuration, in other hand, SA have the most minimum test suite for all the configurations except the second configuration. Finally, referring to the experimental sets 3, BAPSO has been compared to Jenny, TConfig, ITCH, PICT, TVG, CTE-XL, IPOG-D, IPOG, and PSO. In the interaction strength t = 2, TConfig, ITCH, PSO and BAPSO have achieved the optimal test suite, ITCH is the only tool to achieved the most minimum for t = 3, Whereas, BAPSO achieved the most minimum for the t = 4, 5 , and 6, respectively. Overall, BAPSO has get (11 out of 21) configurations tested in our evaluation.

5. Conclusion

We proposed and evaluated a new design for combinatorial interaction testing strategy based on the hybridized BA and PSO for uniform interaction testing, within the objective of generating the minimal test suite that valid and cover all the possible test configuration up to $t = 6$. The strategy is based on covering array notation modelling for the domain targeted either traditional software system and combinatorial interaction testing. It also involves a modeling methodology that describes how to properly specify variability of traditional software system feature to CA or MCA for test suite generation. Here, the test specification is the only process that specifying by domain experts, the strategy processes the test specification requirement as notation to generate the minimum test suite automatically. In this study we evaluate the strategy with some of the available test generation strategies and present the efficiency of our strategy. BAPSO combinatorial interaction testing strategy gives the smallest test suite in most of the experiments conducted. Generally, existing combinatorial interaction testing strategies perform well. However, exploring new methods and the achievement of smaller test suite is necessity, as seen from an interaction testing perspective to reduce testing effort. Thus, the experimental results are encouraging. Although, this research proposes an effective strategy for test suite generation, with the BA as a backbone search engine in the reduction process.

As a future work, this would need to design more effective and effortless search-based test suite generation strategy with GUI web-application.

Acknowledgment

The authors gratefully acknowledge the approval and the support of this research study by the grant no. SCI-2018-3-9-F-6682 from the Deanship of Scientific Research at Northern Border University, Arar, K.S.A.

References

[1]. Lopez-Herrejon, R.E., et al., *Evolutionary computation for software product line testing: An overview and open challenges*. Computational Intelligence and Quantitative Software Engineering. 2016: Springer. 59-87.

[2]. Cohen, D.M., et al., *The AETG system: An approach to testing based on combinatorial design.* IEEE Transactions on Software Engineering, 1997. **23**(7): p. 437-444.

[3]. Grindal, M., J. Offutt, and S.F. Andler, *Combination testing strategies: a survey.* Software Testing, Verification and Reliability, 2005. **15**(3): p. 167-199.

[4]. Kuhn, D.R., R.W. Dolores, and A.M. Gallo, *Software fault interactions and implications for software testing.* IEEE Transactions on Software Engineering, 2004. **30**(6): p. 418-421.

[5]. Czarnecki, K., et al. *Cool features and tough decisions: a comparison of variability modeling approaches*. in *6th International Workshop on Variability Modeling of Software-Intensive Systems*. 2012. ACM.

[6]. Marijan, D., et al. *Practical pairwise testing for software product lines*. in *17th International Software Product Line Conference*. 2013. ACM.

[7]. Haslinger, E.N., R.E. Lopez-Herrejon, and A. Egyed. *Using feature model knowledge to speed up the generation of covering arrays*. in *7th International Workshop on Variability Modelling of Software-intensive Systems*. 2013. ACM.

[8]. Othman, R.R., K.Z. Zamli, and S.M. Syed Mohamad, *t-way testing strategies: a critical survey and analysis.* International Journal of Digital Content Technology and Its Applications, 2013. **7**(9): p. 222-235.

[9]. Zamli, K.Z., A.R. Alsewari, and B. Al-Kazemi, *Comparative benchmarking of constraints t-way test generation strategy based on late acceptance hill climbing algorithm.* International Journal of Software Engineering and Computer Systems, 2015. **1**(1): p. 15-27.

[10]. Nasser, A.M., et al. *Late acceptance hill climbing based strategy for addressing constraints within combinatorial test data generation*. in *7th Edition of Asia Software Testing Conference*. 2014. MSTB.

[11]. Homaid, A.A.B., et al., *Adapting the elitism on greedy algorithm for variable strength combinatorial test cases generation.* IET Software, 2019. **13**(4): p. 286-294.

[12]. Alsewari, A.A. and K.Z. Zamli, *A harmony search based pairwise sampling strategy for combinatorial testing.* International Journal of The Physical Sciences, 2012. **7**(7): p. 1062-1072.

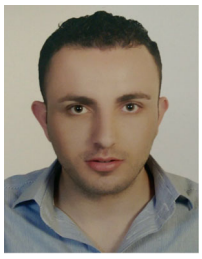
[13]. Alsewari, A.A. and K.Z. Zamli, *Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support.* Information and Software Technology, 2012. **54**(6): p. 553-568.

[14]. Alazzawi, A.K., et al. *A Hybrid Artificial Bee Colony Strategy for t-way Test Set Generation with Constraints Support*. in *Journal of Physics: Conference Series*. 2020. IOP Publishing.
[15]. Alazzawi, A.K., et al., *Pairwise Test Suite Generation Based on Hybrid Artificial Bee Colony Algorithm*, in *Advances in Electronics Engineering*. 2020, Springer. p. 137-145.
[16]. Alazzawi, A.K., et al. *PhABC: A hybrid artificial bee colony strategy for pairwise test suite generation with constraints support*. in *2019 IEEE Student Conference on Research and Development (SCOReD)*. 2019. IEEE.
[17]. Alazzawi, A.K., et al., *HABCSm: A Hamming Based t-way Strategy based on Hybrid Artificial Bee Colony for Variable Strength Test Sets Generation.* INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL, 2021. **16**(5): p. 18.
[18]. Alsariera, Y.A., et al., *A bat-inspired testing strategy for generating constraints pairwise test suite.* Advanced Science Letters, 2018. **24**(10): p. 7245-7250.
[19]. Alsariera, Y.A. and K. Zamli. *A real-world test suite generation using the bat-inspired t-way strategy*. in *The 10th Asia Software Testing Conference (SOFTEC2017)*. 2017.
[20]. Alsariera, Y.A.Z., Kamal Z., *TBat: A Novel Strategy for Minimization of T-Way Interaction Test Suite Based on the Particle Swarm Optimization and the Bat Algorithm*, MA, Editor. 2016.
[21]. Alsariera, Y.A. and K.Z. Zamli, *A bat-inspired strategy for t-way interaction testing.* Advanced Science Letters, 2015. **21**(7): p. 2281-2284.
[22]. Alsariera, Y.A., M.A. Majid, and K.Z. Zamli. *SPLBA: An interaction strategy for testing software product lines using the Bat-inspired algorithm*. in *4th International Conference on Software Engineering and Computer Systems (ICSECS'15)*. 2015. Kuantan, Pahang, Malaysia: IEEE.
[23]. Alsariera, Y.A., M.A. Majid, and K.Z. Zamli, *A bat-inspired Strategy for Pairwise Testing.* ARPN Journal of Engineering and Applied Sciences, 2015. **10**(18): p. 8500-8506.
[24]. Hartman, A. and L. Raskin, *Problems and algorithms for covering arrays.* Discrete Mathematics, 2004. **284**(1): p. 149-156.
[25]. Yang, X.-S., *A new metaheuristic bat-inspired algorithm*. Nature Inspired Cooperative Strategies for Optimization. 2010: Springer. 65-74.
[26]. Jia, D., et al., *A hybrid particle swarm optimization algorithm for high-dimensional problems.* Computers & Industrial Engineering, 2011. **61**(4): p. 1117-1122.
[27]. Meng, X., X. Gao, and Y. Liu, *A novel hybrid bat algorithm with differential evolution strategy for constrained optimization.* International Journal of Hybrid Information Technology, 2015. **8**(1): p. 383-396.
[28]. Eberhart, R. and J. Kennedy. *A new optimizer using particle swarm theory*. in *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*. 1995. Ieee.
[29]. Brest, J., et al., *Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems.* IEEE transactions on evolutionary computation, 2006. **10**(6): p. 646-657.
[30]. Price, K., R.M. Storn, and J.A. Lampinen, *Differential evolution: a practical approach to global optimization*. 2006: Springer Science & Business Media.
[31]. Chelouah, R. and P. Siarry, *A continuous genetic algorithm designed for the global optimization of multimodal functions.* Journal of Heuristics, 2000. **6**(2): p. 191-213.
[32]. Sánchez, A.M., et al., *Hybrid crossover operators with multiple descendents for real-coded genetic algorithms: Combining neighborhood-based crossover operators.* International Journal of Intelligent Systems, 2009. **24**(5): p. 540-567.

Yazan A. Alsariera (PhD) is Assistant Professor of Software Engineering in the department of computer sciences at Northern Border University. He obtained Bachelor of Computer Science from Mut'ah University, Jordan, in 2010. Master of Science in Computer Science (Minor in Software Engineering) from University of Putra Malaysia (UPM), Malaysia, in 2013. A Ph.D. degree in Science (Software Engineering) from University of Malaysia Pahang (UMP), Malaysia, in 2018. His main research interest includes artificial intelligence, computational intelligence, combinatorial optimization, cybersecurity, secure software development, and high-performance computing.

Ahmad H. Al-Omari (PhD) received his B. Sc. In Computer Science in 1985, M.Cs of Computer Science in 2001, and he received his Ph.D. in Computer Information Systems in 2004. He was the acting dean, dean, department head in the faculty of Information Technology FIT, Applied Science University, Amman-Jordan. He supervised many master students, he participated in many masters' examination and discussion committees. His main research interest includes MANET, Encryption, RTA, e-government, supply chain and the tendering process.

Mahmoud A.Albawaleez (PhD) is Assistant Professor of Network and Communication in the Deanship of IT at Northern Border University. He obtained Bachelor of Computer Science from Mut'ah University, Jordan, in 2006. Master of Information Technology from Universiti Utara Malaysia (UUM), Malaysia, in 2009. A Ph.D. degree in Science (Network and Communication) from University of / University Sciences Islamic Malaysia (USIM), Malaysia, in 2018.



Yousef K. Sanjalawe (PhD) is a senior lecturer of Artificial Intelligence/ Cloud Computing in the department of computer sciences at Northern Border University. He obtained Bachelor of Computer Information Systems Science from Jordan University of Science and Technology (JUST), Jordan, in 2007. Master of Computer Information Systems from Yarmouk University, Jordan, in 2014. A Ph.D. degree in Computer Science (Artificial Intelligence/ Cloud Computing) from Universiti Sains Malaysia (USM), Malaysia, in 2019. His main research interest includes cybersecurity, optimization, artificial intelligence, computational intelligence, deep learning, cloud computing.

Kamal Z. Zamli (PhD) is Professor of electrical engineering in Faculty of Computing College of Computing and Applied Sciences at Universiti Malaysia Pahang. He received the degree in electrical engineering from the Worcester Polytechnic Institute, USA, in 1992, the M.Sc. degree in real-time software engineering from Universiti Teknologi Malaysia, in 2000, and the Ph.D. degree in software engineering from the University of Newcastle upon Tyne, U.K., in 2003. His research interests include (combinatorial t-way) software testing and search-based software engineering.