IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 1, 2020, accepted February 9, 2020, date of publication February 13, 2020, date of current version February 26, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2973696

Multiple Black Hole Inspired Meta-Heuristic Searching Optimization for Combinatorial Testing

HAMSA NAJI NSAIF AL-SAMMARRAIE AND DAYANG N. A. JAWAWI
Faculty of Engineering, School of Computing, Universiti Teknologi Malaysia (UTM), Johor Bahru 81310, Malaysia

Corresponding author: Hamsa Naji Nsaif Al-Sammarraie (omeehams@gmail.com)

This work was supported in part by the Ministry of Higher Education Malaysia (MOHE) for Fundamental Research Grant Scheme (FRGS) under Grant 5F117, and in part by the Universiti Teknologi Malaysia (UTM) for UTM-Transdiciplinary Research Grant (TDR) under Grant 06G23.

ABSTRACT Combinatorial searching-based software testing (CSST) is a challenging optimization procedure. The achievement of optimal solutions involves a careful formulation of the optimization problem and the selection of an appropriate approach. Meta-heuristic searching procedures have proven to be effective for solving CSST issues. Black hole (BH) optimization is among the more recently developed meta-heuristic searching algorithms. While this approach has been observed to be an effective alternative to particle swarm optimization, its operation is based on only one swarm. To date, no efforts have been made to modify this approach to accommodate multiple swarms. This study proposes a new variant of BH that involves a combination of multiple swarms. The BH optimizer is modified from continuous searching to binary searching and subsequently applied for solving CSST. The evaluation is based on a modified-benchmarking mathematical function and well-known CSST problems. This modified BH method is superior to the original BH and the established particle swarm optimization (PSO) approach. In terms of CSST problems, binary multiple black hole (BMBH) optimizations generate reduction rates between 50% and more than 60% for $t = 4$ according to the problem.

INDEX TERMS Black hole optimization, combinatorial searching-based software testing, meta-heuristic searching, multiple black hole optimization, swarm meta-heuristic.

I. INTRODUCTION

Currently, the emergence of a variety of complicated systems in software products is increasing. Consequently, the development of an effective process for assessing the quality of these systems has proven to be an arduous task. A typical way of testing any system is to consider the possible values of input interactions to generate test suites that are independent of each other and can be associated with different faults. According to our observations, the software developed over the last two decades has had an emphasis on customizability to user needs. This makes the configuration of such software an important issue [1]. Software testing entails the coverage of all possible values of input or configurations and their interactions. However, given the current speed of hardware, the coverage of all possible variable interactions is not practical. In a system with 9 input variables, if these inputs are indicated as parameters and each parameter has 5 values, the coverage of all the possible combinations of input variables in the traditional way requires testing of $5^9 = 1,953,125$ combinations overall. This would require an exceedingly high level of hardware power [2].

Combinatorial testing (CT) is a subtopic in software engineering for testing purposes. The goal of CT is the conversion of an original space of software variables into a reduced space. This reduced space is considered the input for the testing process to detect faults. This method facilitates the coverage of variable interactions to generate possible faults while excluding superfluous operations [3]. In the context of software generation, automatic generation of a reduced space is crucial in regard to cost-effectiveness, time savings, and quality. In a review [4], CT based on searching approaches was termed search-based software testing (SBST). In addressing the various challenges associated with SBST, it was orated that “There exists a structured parallel approach for test data generation, but an idea of using search together with parallel

The associate editor coordinating the review of this manuscript and approving it for publication was Mostafa M. Fouda.

islands has not been explored with branch selection'' [4]. Existing search algorithms need to be upgraded before these methods can be applied to parallel-based searching.

The relevant literature documented numerous approaches related to meta-heuristic searching for optimization. Each approach is based on a certain concept or metaphor. Some approaches were inspired by genetic evolution and their suitability with regard to the environment, as in [5] and [6], while others were inspired by social aspects [7]. The literature related to heuristic searching reported a wide range of applicable models. The mechanism and searching process vary from one approach to another, which is reflected in the outcome. In the research of artificial intelligence (AI), opinions vary on the superiority of one searching model over another. While some models generate superior results in certain applications, these methods perform less effectively in others [8]. It is essential that the evaluation of meta-heuristic searching be application-dependent. Some models are applicable for evolutionary searching, while others are more suitable for parallel searching. Researchers have recommended several models with a parallel search forte to overcome the problems associated with SBST. This study emphasized the upgrade of an existing meta-heuristic searching algorithm termed black hole (BH)-based optimization into a parallel-based searching algorithm. In this study, the upgraded meta-heuristic algorithm is named the multiple black hole (MBH) optimization method.

The current article is separated into the following sections: Section II provides a theoretical background. Section III presents the t-way test suite generation problem. Next, section IV presents a literature survey. Then, section V gives the background and terms. Next, section VI presents the proposed algorithm and contributions followed by the proposed approach in section VII. Section VIII and section IX present the evaluation and experimental results and the corresponding analysis, respectively. Finally, section X presents the conclusion and future work.

II. THEORETICAL BACKGROUND

Several important preliminaries are necessary to explain the data and theory presented in this article, with the following definitions:

A. DEFINITION 1

A software under test (SUT) is a software or application that is tested based on the possible values of its parameters by assuming that the SUT has n parameters $c_i (i = 1, 2 \dots n)$; the parameters can represent a possible input, certain event (internal or external), or configuration variable, as illustrated in Example 1 in section III. The SUT has n variables $V_1, V_2, \dots V_n$ that take any possible value such that $v_1 \in V_1, v_2 \in V_2 \dots v_n \in V_n$. Based on this approach, the application of combinatorial searching-based software testing (CSST) is appropriate for solving the unlimited array of problems associated with the software engineering industry.

B. DEFINITION 2

The test case TC is an n-tuple $(\mathrm{v}_1, \mathrm{v}_2, \dots \mathrm{v}_n)$ where $\mathrm{v}_1 \in \mathrm{V}_1$, $\mathrm{v}_2 \in \mathrm{V}_2, \dots \mathrm{v}_n \in \mathrm{V}_n$. If a fault is generated when testing the system based on TC, then the test case has served in the t-testing of the system. The examples provided in section III, such as the pizza ordering system, which includes vegetarian cheese, extra thick, ground beef, large, and take away, are regarded TCs.

C. DEFINITION 3

All the possible test cases are denoted TC_{all}, $TC_{all} = \{(v_1, v_2, \dots v_n) | v_1 \in V_1, v_2 \in V_2, \dots v_n \in V_n\}$. In the examples provided in section III, the encoded possible values with integer levels = 0, 1, 2… are portrayed in Table 2. Table 3 displays TC_{all}.

D. DEFINITION 4

The covering array (CA) is adopted as a mathematical object for the description of the generated t-way test suit. Generally, an SUT is composed of multiple parameters that cross-interact with their associated values. In this research, n denotes the number of parameters. The associated levels are denoted by p, while t is the interaction strength. When all the parameters n have the same number of values (v), the CA is represented as CA $(\mathrm{N}; \mathrm{t}, \mathrm{v}^{\mathrm{n}})$. Otherwise, when the numbers of values are not similar for all the parameters, CA can be represented as MCA $(\mathrm{N}, \mathrm{t}, \mathrm{v}_1^{\mathrm{n1}}, \mathrm{v}_2^{\mathrm{n2}}, \mathrm{v}_3^{\mathrm{n3}} \dots, \mathrm{v}_{\mathrm{j}}^{\mathrm{nj}})$.

E. DEFINITION 4

An orthogonal array (OA) with strength t is defined as $OA(N; t; n; (a_1, a_2, \dots a_n))$, where the size of the array is $N \times n$ and the array fulfills the two following conditions:

- Each column of OA with an index i contains all possible elements from the set V_i, where $a_i = |V_i|$.
- Any subset of t columns covers all possible t-tuples exactly $\frac{N}{a_1 \times a_2 \times, \dots \times a_n}$.

III. T-WAY TEST SUITE GENERATION PROBLEM

There are three examples from different types of applications: the first example is the service application known as the pizza ordering system case study, the second example is the smart mobile system case study, and the third example is a heart disease case study.

A. EXAMPLE 1

This example is found in article [9] as well as in other relevant studies and is summarized in Table 1. The pizza ordering system comes with five variables: pizza type, crust, toppings, size, and delivery. Three variables take one of two values (pizza type, crust, and delivery), while two variables take one of three values (toppings and size). Therefore, the covering array combined with the number of cases is equal to $3^2 \times 2^3$. Table 2 shows encoding for the covering array for the pizza ordering system case study. Additionally, Table 3 shows the dataset for the pizza ordering system case study.

TABLE 1. The pizza ordering system case study.

Pizza options (parameters)	Pizza type	Crust	Toppings	Size	Delivery
Configurations	Vegetarian Cheese	Thin Crust	Roasted Chicken	Large	Eat In
	Meat Lover	Extra Thick	Ground Beef	Medium	Takeaway
			Mushroom	Small	

TABLE 2. Encoding of possible values of input to the pizza ordering system case study.

No	v_1	v_2	v_3	v_4	v_5
1	$v_1=0$	$v_2=0$	$v_3=0$	$v_4=0$	$v_5=0$
2	$v_1=1$	$v_2=1$	$v_3=1$	$v_4=1$	$v_5=1$
3			$v_3=2$	$v_4=2$	

TABLE 3. The dataset extracted from the Pizza ordering system case study.

No	Pizza type	Crust	Toppings	Size	Delivery
1	0	0	0	0	0
2	0	0	0	0	1
3	0	0	0	1	0
4	0	0	0	1	1
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
72	1	1	2	2	1

B. EXAMPLE 2

The smart mobile system, this system, is also utilized as a case study where the selection is based on its capacity for offering a wide range of factors and levels [10], as explained in Fig. 1. This renders the smart mobile system favorable for software testing. The factors are correlated such that every factor has a relationship with several other factors. The smart mobile system consists of 18 features that are divided into two categories: one-valued and two-valued parameters. There are ten features in the two-valued parameter category (video call, voice messages, video messages, basic colors, high resolution, camera, video player, music player, radio and voice recorder) and eight features in the one-valued parameter category (smart mobile system, calls, messages, GPS, screen, media, voice call, and text messages).

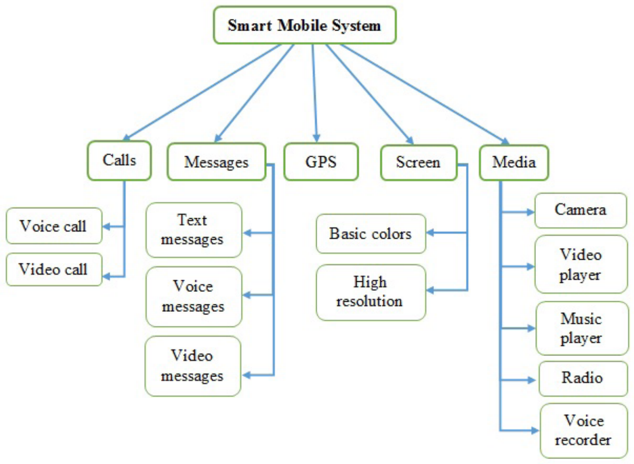

FIGURE 1. Smart mobile system case study.

TABLE 4. Heart disease predictions case study.

No	v_1	v_2	v_3	v_4	v_5	v_6	v_7
1	0	0	0	1	1	1	1
2	1	1	1	1	1	1	1
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
1296	1	1	3	2	2	2	2

From these parameters, the possible configurations that need to be tested are $2^{10} \times 1^8 = 1024$.

C. EXAMPLE 3

The heart disease case study [11] for predictions comprises seven variables: gender (2 levels: male/female), age (four levels: younger than 20, between 20 and 40, between 40 and 60, and older than 60), morning, evening, and sleeping heartbeat rates (three levels: lower than 60, between 60 and 100, and more than 100), and blood test (three levels: lower than 10, between 10 and 15, and more than 15). The full extent of the problem is $2^2 \times 4 \times 3^4 = 1296$. The states can be represented as in Table 4.

IV. LITERATURE SURVEY

The literature survey is separated into two subsections, the literature of CSST and a review of meta-heuristic-based approaches focusing on the parallel nature of some methods.

A. CSST TESTING

The literature on CSST is relatively extensive, embracing numerous approaches. However, all these approaches share a common factor: these approaches exploit the power of random combinatorial searching when they are integrated with heuristics for determining the t-strength covering array. The formulation of the optimization problem calls for an emphasis on two aspects: the devising of the objective function and the selection of the approach employing a pure-based approach [12] or hybrid-based approach [13]. In terms of the objective function formulation, some models use a minimization formulation, while others use a maximization formulation. The fitness function is the total number of different pairs covered from all the test cases by an individual. If an individual covers a greater number of different pairs than others, that individual is superior to the others [14]. Thus, the formulation of the problem is a maximization problem, where an individual becomes a solution when it covers all pairs. In other work, the number of covered d-tuples was used as a fitness value for the candidate solution. On the other hand, the objective function was investigated using the number of non-covered interaction tuples covered by the candidate solution [15]. In another situation, the number of uncovered tuples was taken as the cost of the candidate solution, which required minimization. The cardinality of the set and the objective value were represented by the number of non-covered interaction tuples covered in [16].

In terms of the type of meta-heuristic algorithm applied, the study in [17] proposed a new variant of the teaching learning-based optimization (TLBO) algorithm. Their meta-heuristic algorithm comes with a Mamdani fuzzy inference system, which not only provides a solution to trapping in local optima but also provides greater diversity. In the undertaking of [18], the firefly meta-heuristic algorithm was applied as a test suite generator. The computational comparison between the firefly meta-heuristic algorithm, genetic algorithm, and ant colony optimization revealed that the firefly algorithm has a shorter optimization time. Other models developed for solving combinatorial software testing using meta-heuristic searching include the memetic algorithm (MA) in [19], the particle swarm optimization (PSO) in [20], the artificial bee colony (ABC) and corresponding developed variants in [13], [21]–[26], the harmony search algorithm (HSA) in [27], the bat algorithm (BA) in [12], [28]–[30], and the flower pollination algorithm (FPA) in [31] and [32]. All these models are considered pure models because they utilize a common pure meta-heuristic approach. Each of the meta-heuristic searching-based approaches uses a representation related to the metaphor of the approach. For example, in the work of [21], ABC was used for t-way generation. The representation these researchers used presents the number of food sources by a test case and the fitness value by the coverage. The food source is optimized by the employed bees and presented to the onlooker bees. The food sources are selected based on the probability calculated using their fitness, while the onlooker bees investigate the solutions and select the best. On the other hand, analyzing the literature of meta-heuristic-based optimization reveals that there are differences in the approaches with respect to the parameters. This aspect has been discussed in the work of [25], where various approach parameters were presented. For example, PSO uses the maximum iteration, swarm size, learning factors and inertia factor. ABC uses the maximum iteration, the number of colony sizes, the number of food source limits, and the number of cycles.

In terms of hybrid-based approaches, several researchers integrated meta-heuristic algorithms with other models. For example, [33] integrated the particle swarm optimization approach with fuzzy logic for the tuning of w,c_1, and c_2. The authors built three fuzzy inference systems to monitor the PSO performance and to adjust w, c_1, and c_2 for optimization improvement efficiency. The role of PSO is as an exploitation agent for ABC. ABC takes the information from PSO via the weight factor. This algorithm is an extended variant of [21].

B. PARALLEL META-HEURISTIC SEARCHING

Parallel meta-heuristic searching refers to methods that come with a parallel searching quality. Researchers have made efforts to improve the exploration qualities of existing meta-heuristic searching algorithms by converting them into parallel searching algorithms. The study in [34] developed the parallel comprehensive learning particle swarm optimizer (PCLPSO). This optimizer, which comes with multiple swarms based on the master-slave paradigm, operates both cooperatively and concurrently. The simulations in [34] fashioned a multi-swarm optimizer for multi-objective optimization. The authors used a hybrid strategy decomposition and dominance (MSMO/2D) to improve convergence and diversity by splitting the primary swarm into several sub-swarms. Similarly, [29] devised a multi-swarm algorithm, called the multi-swarm bat algorithm (MBA), for global optimization. This algorithm recommends the exchange of information between different swarms. Rule hiding is included in several applications using the MBA [35]. The study in [13] states that the strength t was increased from $t = 2$ to $t = 6$ through the development of a hybrid artificial bee colony (HABC) strategy by hybridizing an ABC algorithm and a PSO algorithm. The observed performance of this family of parallel searching approaches and the good performance of black hole optimization motivate us to develop a multi-variant of this approach for solving the problem of t-way testing.

V. BACKGROUND AND TERMS

Scientists define a black hole (BH) as a part of the universe where a great gravitational force attracts and subsequently swallows anything (including light). The attraction of anything moving around the black hole occurs in a region with a certain radius. This region is known as the event horizon (EH). Light that crosses the EH inevitably disappears into the black hole for good. This has inspired researchers, such as in [36], to apply this concept as a metaphor for a point in the solution space that has a better objective value than its surroundings. This serves to remove all nearby solutions within a given radius. Researchers associate this point in the solution space with a BH and its surrounding region with the EH.

The pseudocode of the BH algorithm is provided in Fig. 2. The sequence for this algorithm is as follows: (a) random initialization of the solutions in the solution space, (b) evaluation of each solution using the objective function, (c) selection of the solution corresponding to the minimum cost value to

```
Input: Definition of the solution space
       Iteration              // the maximum number of iterations
Output: ymin                  // the minimum value of y=f(x)
        Sol                   // the best solution
Loop :
P =init ()                    // create array that represents
                                 population of Stars Distributed
                                 randomly in the solution space
For t=1 until Iteration do    // the maximum iteration
    [EP, BH]=update Fitness (P)     // update the fitness value
                                       and the black hole
    P=moveStars (P, BH)             // move the stars towards the
                                       black hole
    [EP, BH]=update Fitness (P)
    R=update Radius (EP, BH)        // update the radius of black
                                       hole
    For each S inside P do
        If (dis (BH, S) <R)
            Replace (P, S)
        End
    End
    [EP, BH]=update Fitness (P)
    If (Stop criteria is met) then
        Exit
End
End
ymin =min (P)
Sol=argmin (p)
End
Function y=f(x)               //definition of objective function
Function [EP, BH] =update Fitness (P)
        EP= {}                // initialize the evaluation array of
                                 solutions as an empty array
        For each S inside P do
            y=f(S)
            EP.Add(y)         // add y to the solution
                                 evaluation set array EP
        End
        BH=P (index of min (EP))   // the optimization performs
                                      a minimization
End
Function P=moveStars (P, BH)
        For each S inside P do
            S=S+rand (0, 1)*(xBH-S)    // move the solutions
                                          toward the BH
        End
        For each dimension in S do
            If (S (dimension) <Bmin (dimension))
                S (dimension) =Bmin (dimension).
            Else if (S (dimension)>Bmax (dimension))
                S (dimension) =Bmax (dimension)
            End
        End
End
Function R=update Radius (P, EP, BH)   // update the
                                          radius of the black hole
        Sum=0
        For each S in P do
            Sum=sum+EP(S)
        End
        R=EP (BH)/sum
End
```

FIGURE 2. The pseudocode of the black hole algorithm.

represent the BH, (d) shifting of all the solutions towards the BH, (e) a fresh evaluation of the population to update the BH with the new solutions corresponding to the minimum cost value, (f) calculation of the radius of the BH and replacement of the solutions that cross the radius with new solutions, (g) a fresh evaluation to update the BH with the new solution corresponding to the minimum cost value, and last, (h) a check of the stopping criterion (the number of iterations or the convergence of the fitness value). The algorithm is finished when the stopping criterion is met. The terms, notations, and solutions used in the methodology are provided in Table 5. The solution space is defined as dimension n. Any solution is presented as star x_s, while the black hole is presented as x_{BHi}. Prior to operating the algorithm, it is essential that N_{BH}, which represents the number of black holes, be defined. The radius of the EH for each BH is denoted as R (i).

TABLE 5. The symbols of the MBH algorithm and their definitions.

No	Symbol	Meaning
1	$y = f(x)$	Objective function for optimization
2	$x_s \in R^n$	Star in the MBH algorithm
3	N_s	Number of solutions
4	x_{BHi}	Black hole
5	$N_{smin,BHi}$	Minimum allowed number of solutions inside black hole i
6	N_{BH}	Number of black holes
7	N_{maxBH}	Maximum number of black holes
8	R(i)	Radius for each black hole
9	p(i)	Population of stars for each black hole
10	EP(i)	Evaluated population for each black hole
11	Ei	Energy inside the black hole
12	$E_{threshold}$	Minimum energy for the black hole before it dies
13	E_{max}	Maximum energy of the black hole when it is born
14	B_{min}	Lower boundary of the solution space
15	B_{max}	Upper boundary of the solution space
16	BHn	Black hole new
17	BHo	Black hole old
18	It	Number of iterations

VI. PROPOSED ALGORITHM AND CONTRIBUTION

Among the recently developed meta-heuristic searching algorithms is the black hole (BH) algorithm [37]. This algorithm is based on the black hole phenomenon and the behavior of stars during their interaction with the BH. A star that gets too close to the black hole is swallowed by the BH. In the context of the BH algorithm, a new star is randomly generated to represent a new solution. This new star is included in the search. This study in [36] compared the black hole algorithm to other meta-heuristic optimization algorithms. Among them, the performance of PSO was observed to be exceptional. As presented in section III, our literature survey identified CSST as one of the computational problems that can be addressed through meta-heuristic searching. The computational difficulties associated with CSST call for the application of a powerful meta-heuristic optimization. Considering the substantial power of the multi-swarm-based optimization algorithm, the modification of the black hole optimization into a multi-black hole with multiple swarms of stars delivers improved optimization results. Combinatorial t-way testing is one of the most well-known optimization problems solvable through meta-heuristic optimization. A multi-swarm variant of the BH searching optimizer is successfully developed, called the multiple black hole (MBH) optimization, and is applied to the problem of CSST.

Based on the problem statement stated, the following contributions are presented.

- A novel variant of the black hole optimization approach based on the multi-swarm concept is proposed that can be characterized as a variant multiple black hole or MBH optimization. This is supported by introducing the concept of the black hole energy to facilitate the elimination of certain black hole swarms and to generate fresh swarms.
- A binary variant of the MBH optimization algorithm is presented and given the name of the binary multiple black hole (BMBH) optimization algorithm. The purpose of this variant is to solve the combinatory problems of CSST.
- A benchmarking mathematical function is proposed to provide a comprehensive comparison between PSO, the classical BH, and the BH variant MBH.
- A well-known benchmarking problem of CSST documented in the relevant literature is applied to evaluate BMBH in conjunction with the binary original variant BH and PSO.

VII. THE PROPOSED APPROACH

The purpose of this section is to describe the methodology employed to realize the objectives of this investigation, which is a new variant of the BH algorithm named the multiple black hole (MBH) algorithm. Another algorithm, a binary variant of the MBH algorithm named the binary multiple black hole (BMBH) algorithm, is also employed. The performance of the BMBH algorithm in solving CSST is demonstrated.

A. MBH ALGORITHM

In this section, the focus is on an explanation with regard to the multiple black hole optimization algorithm. This algorithm begins with the creation of N_{BH} populations and the selection of a BH for each population. The black hole x_{BHi} of population Pi is the solution that achieves the minimum objective value within the stars of the population. The movement of the stars of each BH towards the BH leads to an update of the BH by the star that achieves the lowest value for the objective function. Each update of the BH corresponds to an increase in the energy of the BH by one.

The higher the energy of a particular BH is, the greater the potential for the future generation of stars by that BH. In this study, the radius of the BH and the elimination of the solutions that crossed that radius are developed. The eliminated stars or solutions are then replaced at a probability according to the existing energy within the black hole. A BH with less energy has fewer stars. Such a BH meets the omitting conditions or is removed.

1) PSEUDOCODE OF MBH

The pseudocode for MBH is illustrated in Fig. 3. As portrayed, the difference between BH and MBH has to do with the latter's acceptance of a predefined number of BHs, rather than a solitary BH. The algorithm performs the search in parallel for NBH equal to the number of black holes. Another concept introduced is the energy of the BH at the beginning of the search, or at the birth of the BH, when the BH has the maximum energy. With each occurrence of non-improvement in the best solution in the black hole, the energy is decreased by one until the energy of the BH is negative. At this point, the BH is replaced by a fresh BH. This concept facilitates simultaneous searching in multiple BHs to enhance the probability of attaining better solutions. The last difference is that

```
1.  Input: Definition of the solution space
2.        Iteration                        // the maximum number of iterations
3.  Output: ymin                           // the minimum value of y=f(x)
4.        Sol                              // the best solution
5.  Loop :
6.  P =init ()                             // create array that represents
                                             population of Stars Distributed
                                             randomly in the solution space
7.  For t=1 until Iteration do             // the maximum iteration
8.     [EP, BH]=update Fitness (P)         // update the fitness value
                                              and the black hole
9.     P=moveStars (P, BH)                 // move the stars towards the
                                              black hole
10.    [EP, BH]=update Fitness (P)
11.    R=update Radius (EP, BH)            // update the radius of black
                                              hole
12.    For each S inside P do
13.       If (dis (BH, S) <R)
14.          Replace (P, S)
15.       End
16.    End
17.  [EP, BH]=update Fitness (P)
18.  If (Stop criteria is met) then
19.     Exit
20.  End
21.  End
22.  ymin =min (P)
23.  Sol=argmin (p)
24.  End
25.  Function y=f(x)                       //definition of objective function
26.  Function [EP, BH] =update Fitness (P)
27.        EP= {}                          // initialize the evaluation array of
                                             solutions as an empty array
28.        For each S inside P do
29.           y=f(S)
30.           EP.Add(y)                    // add y to the solution
                                              evaluation set array EP
31.        End
32.        BH=P (index of min (EP))        // the optimization performs
                                              a minimization
33.  End
34.  Function P=moveStars (P, BH)
35.        For each S inside P do
36.           S=S+rand (0, 1)*(xBH-S)      // move the solutions
                                              toward the BH
37.        End
38.        For each dimension in S do
39.           If (S (dimension) <Bmin (dimension))
40.              S (dimension) =Bmin (dimension).
41.           Else if (S (dimension)>Bmax (dimension))
42.              S (dimension) =Bmax (dimension)
43.           End
44.        End
45.  End
46.  Function R=update Radius (P, EP, BH)  // update the
                                              radius of the black hole
47.      Sum=0
48.      For each S in P do
49.         Sum=sum+EP(S)
50.      End
51.      R=EP (BH)/sum
52. End
```

FIGURE 3. The pseudocode of the MBH algorithm.

MBH enables smarter and more dynamic searching than BH, which is dependent on the initialization of its parameters because of the static nature due to the lack of elimination and regeneration of swarms.

2) COMPARISON BETWEEN MBH AND BH

There are several differences between the BH optimization and the newly developed MBH optimization. While the BH optimization involves the use of a single BH during its search in the solution space, the MBH optimization involves the use of the N_{BH} black hole for the same purpose. Additionally, the MBH optimization algorithm includes the concept of the energy of the BH. This concept serves to eliminate non-active BHs, as well as BHs that do not show improvement with time. Improvement means the capacity of the algorithm to search for better fitness values for the solution, representing the BH. In comparison to the BH optimization algorithm, the MBH optimization algorithm is more effective for searches in the solution space. Additionally, the MBH optimization algorithm can be relied on to achieve better results with fewer iterations than required by the BH optimization algorithm. In both the MBH and BH optimizations, the movement of stars beyond the permitted region of search (violation of the constraint) results in their clipping. This renders their searching prowess less effective. The MBH optimization algorithm overcomes this dilemma through the addition of random components after being subjected to clipping.

Assuming N_s is the number of solutions, It is the number of iterations, and N_{maxBH} is the maximum number of black holes, then the complexity of BMBH for the worst case scenario is $O(It.N_sN_{maxBH})$, while the complexity of the BBH optimization is $O(It.N_s)$.

B. BINARY VARIANT OF BOTH THE BH AND MBH ALGORITHMS

The equation defining the movement of stars towards the black hole, rand, is a random number between 0 and 1, and the movement equation for the stars is as follows:

$$x_i(t+1) = x_i(t) + rand \times (x_{BH} - x_i(t)), \quad i = 1, 2 \ldots N \qquad (1)$$

The setback accompanying (1) is its ineffectiveness with regard to binary values because of the addition of a second term to a binary value, causing a breaching of the binary constraint, which is 0 or 1. To overcome this problem, a parameter named the pulling rate pr is introduced. This approach generates a random number r_d between 0 and 1, where d denotes the index of the dimension. The value of rd is compared to that of pr, and the value of $x_i(t+1)^d$ is defined in compliance with (2) below:

$$x_i(t+1)^d = \begin{cases} BH(t)^d & if\ r < pr \\ x_i(t)^d & otherwise \end{cases} \qquad (2)$$

A small value for the pulling rate pr is preferred for a diligent search in the solution space. The following example

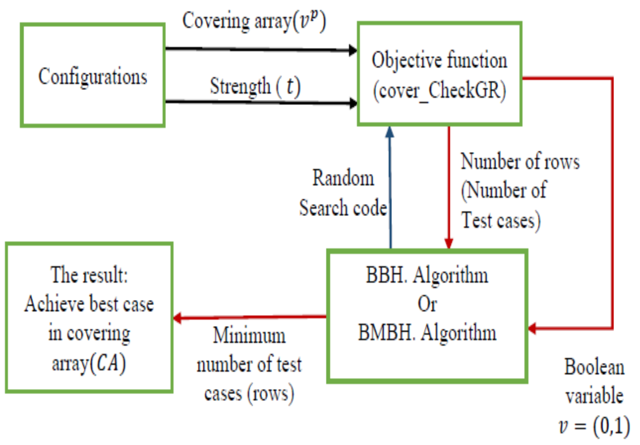
Configurations
Covering array(v^p)
Strength (t)
Objective function (cover_CheckGR)
Random Search code
Number of rows (Number of Test cases)
The result: Achieve best case in covering array(CA)
Minimum number of test cases (rows)
BBH. Algorithm Or BMBH. Algorithm
Boolean variable $v = (0,1)$

FIGURE 4. **The t-way optimization framework using the black hole algorithm.**

provides an explanation of the performance of (2). Assuming one star has the value $x_t = [1\ 0\ 0\ 1]$, while the black hole $BH_t = [0\ 0\ 0\ 0]$ and $pr = 0.2$, we generate one random number rd for each element of x_t. Assuming that $r = [r_1 r_2 r_3 r_4] = [0.1\ 0.2\ 0.04\ 0.6]$, then based on (2), the value of $x_i(t+1) = [0\ 0\ 0\ 1]$. While the initial distance between the star and the black hole is observed to be ($\sqrt{2}$), the movement of the star in compliance with (2) alters this distance to 1. The similarity of the BMBH and BBH concepts allows for the use of the same equation to move the stars. The addition of another subscript to the equation serves to indicate the index of the BH. Subsequently, this generates a random number r_d between 0 and 1, where d denotes the index of the dimension. Then, we identify its value and alter the value of $x_i^j(t)^d$ to $x_i^j(t+1)^d$ in compliance with (3) below:

$$x_i^j(t+1)^d = \begin{cases} BH^j(t) & if\ r_d < pr \\ x_i^j(t)^d & otherwise \end{cases} \qquad (3)$$

where j indicates the index of the black hole and

$$j = 1, 2, \ldots N_{BH}$$

C. CSST BASED ON BMBH

To verify the covering array of strength t, with the entire number of test cases, a function designed for this purpose is employed. This function, which verifies the validity of the covering array, returns a Boolean variable with a value of 0 if the covering condition for a certain strength is not met and with a value of 1 if the covering condition for a certain strength is met. Known as the cover CheckGR, the integration of this function with the BBH and BMBH algorithms is portrayed in Fig. 4. As can be observed, the searching algorithm generates a random candidate solution and delivers the solution to the objective function. The objective function employs cover CheckGR, which determines if the array has a covering property with strength t. Then, if the covering property strength is met, the number of rows of the covering array

TABLE 6. The ranges of the parameters and the effect of each parameter.

Parameter	Meaning	Range	Effect
NBH	number of black holes	{1,2, ...10}	Increasing this number implies more exploration; however, it increases the computational time according to the big O notation
IterationsMax	maximum number of iterations	[100,300]	Increasing this number implies more exploration; however, it increases the computational time according to the big O notation
Ethreshold	minimum energy of the black hole when solution does not change	0.1 * Emax	When the energy reaches 10% of the energy, the black hole is assumed to be non-useful and is dead
Emax	maximum energy of the black hole when it is born	IterationsMax	the initial energy is considered to be the same as the maximum number of iterations IterationsMax
Bmin and Bmax	the lower and upper boundaries of the solution space	Depends on the problem	Narrowing this range provides faster results; however, it is important to ensure that the range contains the optimal solution

represents a reflection of the fitness values. The objective of this exercise is the minimization of the number of rows. In a circumstance where the covering property strength is not met, the fitness value is given as infinity. This implies that the solution is no longer included in iterations for the generation of new solutions.

D. PARAMETER TUNING

The goal of parameter tuning is to determine the best values of parameters for the algorithm. To determine the best values for the parameters, it is necessary to provide the effect of each of the parameters. This information is given in Table 6. The values that are used for the problem of t-way testing are selected based on a tuning process that balances exploration and execution time. The algorithm selects an average value of 3 for NBH and 250 for *IterationsMax* for adequate results.

VIII. EVALUATION AND EXPERIMENTAL RESULTS

To evaluate our proposed MBH and BMBH, we use MathWorks ® MATLAB 2019b and code the PSO, BPSO, BH, MBH, BBH, and BMBH approaches with the evaluating objective functions. The experiments are conducted on a 1.4 GH Intel i5 processor. The evaluation process is separated into two sections. First, the MBH optimization is assessed, and its performance is compared with that of the original BH optimization and PSO. Benchmarking mathematical functions are employed for this first stage. In the second section, the BMBH optimization is assessed, and its performance is compared with that of BBH and BPSO. Several CSST problems reported in the relevant literature are referred to for this second stage.

A. BENCHMARKING MATHEMATICAL FUNCTIONS

To gauge the performance of the MBH algorithm, the algorithm is compared with the BH and PSO algorithms with respect to the capacity for determining the optimal point of the provided benchmark optimization functions. The formulas of the mathematical functions, which cover the true optimal, dimension, searching range, and setting of the algorithms in terms of the number of solutions and number of iterations, are presented in Table 7. The MBH optimization converges closer to the optimal value than either BH or PSO. Furthermore, the BH optimization is better than PSO for most functions and similar for some. This comparison emphasizes the superiority of the MBH optimization over both the BH optimization and PSO.

B. BENCHMARKING CASE STUDIES BASED ON CSST

Three case studies (i.e., the pizza ordering system, heart disease system, and smart mobile system) are used to evaluate the performance of the proposed BMBH and BBH and compare them with that of the BPSO. Each case study has different features and aspects of industry. The pizza ordering system considers the user as a client of restaurants. The heart disease system considers the health care center to be the user. The smart mobile system offers a wide range of factors and levels, which makes this system suitable for software testing. From a t-way testing point of view, all the case studies have different sizes of covering array, numbers of variables and levels. The results for the three case studies are generated based on CSST. Table 8 shows the features of each case study.

The results of comparisons between the BMBH optimization, BBH optimization and BPSO for the three literature case studies in Examples 1, 2, and 3 are presented. The algorithms are tested with 250 solutions and iterations. The same values that resulted from the parameter tuning phase are used. The resulting numbers are used for expediting experimental operations, as well as for increasing the number of solutions and iterations to generate better results.

For the pizza ordering system problem, the full size of the problem is CA ($2^3 \times 3^2 = 72$) cases. In a comparison experiment involving BPSO, BBH optimization, and BMBH optimization, for the three values of $t = 2$, $t = 3$, and $t = 4$, both BBH and BMBH are superior to BPSO in this

TABLE 7. The benchmark mathematical optimization functions and their optimal values.

No	Function Name	Function Formula	Searching Range	Dimension of Solution	Number of Solutions	Number of Iterations	Optimal Cost	PSO Results	BH Results	MBH Results
1	Parabolic	$y = x_1^2 + x_2^2$	$[-10\ \ 10]^2$	2	50	50	0	2.4865×10^{-15}	2.7188×10^{-31}	0
2	Ackley	f(x) = $-a.\exp\left(-b\sqrt{\frac{1}{n}\sum_{i=1}^{n} xi^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(cxi)\right) + a + \exp(1)$	$[-10\ \ 10]^2$	2	50	50	0	1.4283×10^{-6}	1.2893×10^{-10}	8.8818×10^{-16}
3	Rastrigin	f(x,y)= $10n + \sum_{i=1}^{n}(xi^2 - 10\cos(2\pi xi))$	$[-10\ \ 10]^2$	2	50	50	0	3.2391×10^{-5}	3.7162×10^{-7}	0
4	Alpine	f(x) = $\prod_{i=1}^{n}\sqrt{xi}\sin(xi)$	$[5\ \ 10]^2$	2	50	50	7.8849	6.0213	7.8856	0

TABLE 8. Features of case studies.

Case study	Size of Full covering array	Number of variables	Levels	Industry
Pizza ordering system	CA($2^3 \times 3^2$ = 72)	5	Mixed Covering array (MCA)	Services
Smart mobile system	CA($2^{10} \times 1^8$ = 1024)	18	Mixed Covering array (MCA)	Communication
Heart disease system	CA($2^2 \times 4 \times 3^4$ = 1296)	7	Mixed Covering array (MCA)	Health care

area. BMBH achieves a lower cost value for t = 2 with a reduction in the full size from 72 to 22 and for t = 3 with a reduction rate in the full size from 72 to 35. Furthermore, as illustrated in Table 9, the greatest reduction rate of 68% for t = 2 is recorded by BMBH because this algorithm reduced the original size of 72 to 22.

In Fig. 5, for the pizza ordering system problem, a boxplot diagram based on 20 experiments performed for t = 3 shows that the BMBH optimization is the only method that can effectively reduce the full size of the case study. This is portrayed in Fig. 5. Based on the figure, the BBH and BMBH results have similar minimum cost tabulations, but the BMBH results are better in terms of the means. However, cross-referencing Fig. 5 with Table 9 shows that the results and the standard deviations of both algorithms are similar. This is due to the multi-swarm nature of BMBH. Another factor is the BMBH performance; the energy variable of the swarm enables the algorithm to replace BHs that do not produce improvement with other black holes.

For smart mobile systems, the full size of the problem is CA($2^{10} \times 1^8 = 1024$) cases. The BMBH optimization reduces the full problem size from 1024 to 400 for t = 2 and t = 3, while the BBH optimization reduces the full problem size from 1024 to 485 for t = 2 and t = 3. However, although BPSO reduces the full problem size from 1024 to 1002, this is only for t = 2 and t = 3. Table 10 displays the reduction rate of each algorithm with respect to the full size of the problem. This table makes clear that the highest reduction rate is achieved by BMBH optimization, which is 61% for t = 2 and t = 3.

Additionally, according to the boxplot diagram for the smart mobile system displayed in Fig. 6, the BMBH optimization is better than the rest with regard to reducing the problem size. This optimization exhibited a lower standard deviation, as well as fewer rows, compared to BPSO and the

TABLE 9. Pizza ordering system problem, BMBH vs. BBH vs. BPSO.

t - strength	t = 2				t = 3				t = 4				
Algorithm	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Full Size
BPSO	72	0	0	0	72	0	0	0	72	0	0	0	72
BBH	28	26.05	2.1879	63%	39	38.15	2.059	48%	72	0	0	0	72
BMBH	22	22.7	0.9787	68%	35	35.9	1.0208	51%	53	55.05	2.5849	23%	72

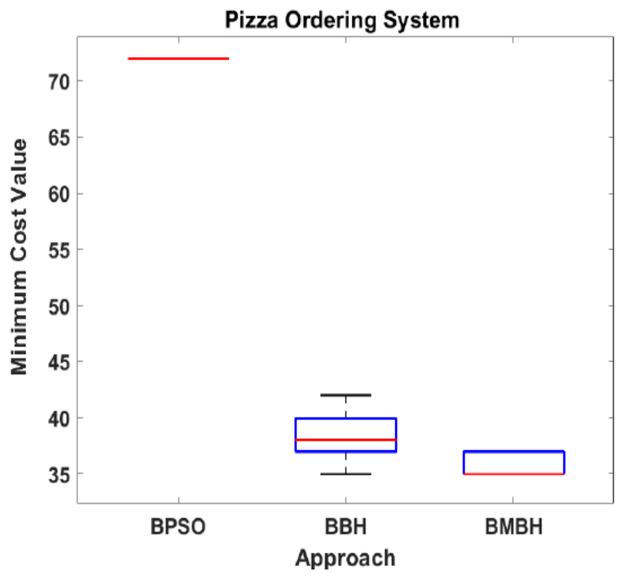

FIGURE 5. Pizza ordering system problem boxplot diagram for t = 3.

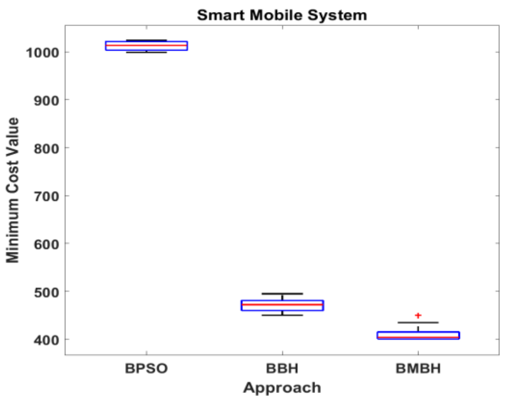

FIGURE 6. Smart mobile system problem boxplot diagram for t = 3.

BBH optimization. Based on the figure, the BBH and BMBH results have similar minimum cost tabulations, but the BMBH results have better mean and standard deviation results. Additionally, cross-referencing Fig. 6 with Table 10 shows that BMBH has superior results. Again, this is due to the multi-swarm nature of BMBH. Another factor is the BMBH performance; the energy variable of the swarm provides high exploration power in the searching space.

For the heart disease prediction problem, the full size of the problem is CA ($2^2 \times 4 \times 3^4 = 1296$). The BMBH optimization reduces the full problem size from 1296 to 587 for $t = 2$ and $t = 3$, while the BBH optimization reduces the full problem size from 1296 to 610 for $t = 2$ and $t = 3$. However, while BPSO reduces the full problem size from 1296 to 883, this is only for $t = 2$ and $t = 3$. Table 10 displays the reduction rate of each algorithm with respect to the full size of the problem. This table clearly shows that the highest reduction is associated with the BMBH optimization, with 54% for $t = 2$, $t = 3$ and $t = 4$.

Additionally, the boxplot diagram for the heart disease problem shown in Fig. 7 reveals the BMBH optimization to be superior to the rest in terms of problem size reduction. This optimization exhibited a lower standard deviation, as well as fewer rows, in comparison to BPSO and the BBH optimization. Additionally, cross-referencing Fig. 7 with Table 11 shows that BMBH has a better mean and standard deviation. Again, the superiority of this algorithm comes from the multi-swarm nature of BMBH. Another factor is BMBH performance; the energy variable of the swarm provides powerful searching.

IX. RESULT ANALYSIS

The results reveal that BMBH is able to provide more optimal and consistent performance than either BBH or BPSO. The combinatorial nature of the problem and the large number

TABLE 10. Smart mobile system problem - BMBH vs. BBH vs. BPSO.

t – strength	t = 2				t = 3				t = 4				
Algorithm	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Full size
BPSO	1002	1002.3	4.5318	2%	1002	1002.3	4.5318	2%	1024	0	0	0	1024
BBH	485	485.00	4.1422	53%	485	485.00	4.1422	53%	496	496.00	3.5689	52%	1024
BMBH	400	400.8	4.021	61%	400	400.8	4.021	61%	410	410.1	4.1409	60%	1024

TABLE 11. Heart disease problem - BMBH vs. BBH vs. BPSO.

t – strength	t = 2				t = 3				t = 4				
Algorithm	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Minimum cost Value	Mean	Standard Deviation	Reduction Rate	Full size
BPSO	883	889.8	12.5514	31%	883	889.8	12.5514	31%	1296	0	0	0	1296
BBH	610	605.25	6.4144	53%	610	605.25	6.4144	53%	613	606.9	5.6186	53%	1296
BMBH	587	594.85	6.0981	54%	587	594.85	6.0981	54%	595	597.25	3.4163	54%	1296

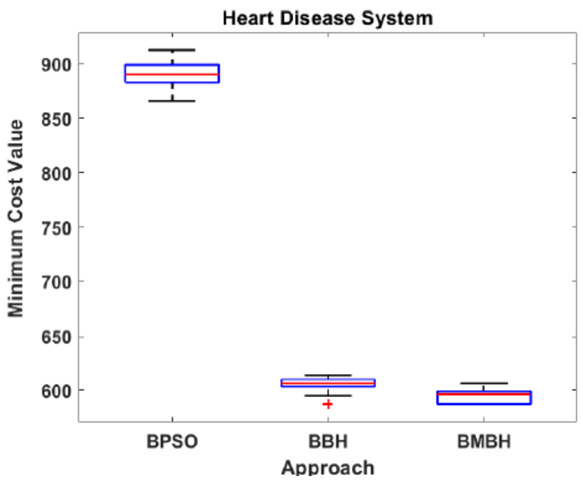

FIGURE 7. Heart disease system problem boxplot diagram for t = 3.

of possible solutions implies a very large solution space. This makes searching difficult. In both BPSO and BBH, the candidate solutions are selected randomly and change their position in the space according to the best solutions, represented by the leader in BPSO and the black hole in BBH. However, it is very possible that the quality of the whole swarm in both approaches is poor, including the leader in BPSO and the black hole in BBH. Hence, the search will not provide good results at the end. However, the incorporation of multi-swarm searching under multi-black holes in BMBH with the concept of the energy of the black holes provides the algorithm with the capability of replacing poor quality black holes with good ones. The indicator of good quality is the improvement in the black hole value over the progress of searching. Another observation is the consistency in the experimental results of BMBH compared with those of both BBH and BPSO. The boxplot standard deviation is lower in BMBH, which means that this algorithm has more consistency. This is facilitated by the capability of the algorithm to determine the quality of the black holes based on the concept of energy compared with BBH and BPSO, which suffer from sensitivity to the seed that provides the solution.

The differences between the performances of BMBH, BBH and BPSO occur when the t value increases, which indicates a wider solution space and more complex optimization.

X. CONCLUSION AND FUTURE WORK

CSST represents a challenging optimization problem. This challenge was treated with the proposal of the application of the multiple swarm concept to existing swarm methods. To achieve this, the existing meta-heuristic searching optimization method known as the BH optimization as upgraded to a multiple swarm-based BH optimization method named the MBH optimization. In this paper, the application of MBH to CSST by the conversion of the MBH optimization into a binary variant called the BMBH optimization was employed. BMBH was able to generate more optimal solutions to CSST problems compared with BBH and BPSO. This was exhibited by the exploration power on CSST; BMBH is able to generate swarm distributions in various regions in the solution space, while BBH is not. Another aspect of the powerful searching performance of BMBH is the energy variable, which increases or decreases according to the success of the black hole in finding more optimal solutions and updating its value. This factor does not exist in classical BH or the binary variants of BH.

The approach suffers from various limitations. First, the current variant of BH and BMBH has no mutation operations, which makes the algorithm subject to local minima by some percentage. Incorporating mutation operations might increase the performance of the algorithm and make it immune to such local optima. Second, the algorithms BBH and BMBH have a randomly blind initialization of the first population. Developing an application-oriented initialization might improve the performance of these algorithms. Finally, the process of pulling the stars towards the black hole currently does not focus on the application part. Moving the stars in an application-dependent operation will generate better performance.

In the future, the optimization will be extended to accept various constraints on the variables, which will be more complicated than having only boundary constraints. Another future direction is to incorporate reinforcement learning, which makes the searching subject to training and, hence, more fruitful.

ACKNOWLEDGMENT

The authors would like to express their profound gratitude to the lab members of the Software Engineering Research Group, Universiti Teknologi Malaysia (UTM), for their continuous support and constructive feedback towards the completion of this paper.

REFERENCES

[1] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Softw. Eng.*, vol. 16, no. 1, pp. 61–102, Feb. 2011.

[2] J. Lin, C. Luo, S. Cai, K. Su, D. Hao, and L. Zhang, "TCA: An efficient two-mode meta-heuristic algorithm for combinatorial test generation (T)," in *Proc. 30th IEEE/ACM Int. Conf. Automat. Softw. Eng. (ASE)*, Nov. 2015, pp. 494–505.

[3] L. Ma, F. Zhang, M. Xue, B. Li, Y. Liu, J. Zhao, and Y. Wang, "Combinatorial testing for deep learning systems," 2018, *arXiv:1806.07723*. [Online]. Available: http://arxiv.org/abs/1806.07723

[4] M. Khari and P. Kumar, "An extensive evaluation of search-based software testing: A review," *Soft Comput.*, vol. 23, no. 6, pp. 1933–1946, Mar. 2019.

[5] R.-Z. Qi, Z.-J. Wang, and S.-Y. Li, "A parallel genetic algorithm based on spark for pairwise test suite generation," *J. Comput. Sci. Technol.*, vol. 31, no. 2, pp. 417–427, Mar. 2016.

[6] X. Li, W. E. Wong, R. Gao, L. Hu, and S. Hosono, "Genetic algorithm-based test generation for software product line with the integration of fault localization techniques," *Empirical Softw. Eng.*, vol. 23, no. 1, pp. 1–51, Feb. 2018.

[7] D. V. C. Prakash, S. Tatale, V. Kondhalkar, and L. Bewoor, "A critical review on automated test case generation for conducting combinatorial testing using particle swarm optimization," *Int. J. Eng. Technol.*, vol. 7, no. 3.8, p. 22, Jul. 2018.

[8] R. R. Sahoo and M. Ray, "Metaheuristic techniques for test case generation: A review," *J. Inf. Technol. Res.*, vol. 11, no. 1, pp. 158–171, Jan. 2018.

[9] A. R. A. Alsewari and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support," *Inf. Softw. Technol.*, vol. 54, no. 6, pp. 553–568, Jun. 2012.

[10] Y. A. Alsariera, M. A. Majid, and K. Z. Zamli, "SPLBA: An interaction strategy for testing software product lines using the Bat-inspired algorithm," in *Proc. 4th Int. Conf. Softw. Eng. Comput. Syst. (ICSECS)*, Aug. 2015, pp. 148–153.

[11] C. S. Dangare and S. S. Apte, "Improved study of heart disease prediction system using data mining classification techniques," *Int. J. Control Automat.*, vol. 47, no. 10, pp. 44–48, Jul. 2012.

[12] Y. A. Alsariera and K. Z. Zamli, "A bat-inspired strategy for t-way interaction testing," *Adv. Sci. Lett.*, vol. 21, no. 7, pp. 2281–2284, Jul. 2015.

[13] A. K. Alazzawi, H. M. Rais, and S. Basri, "HABC: Hybrid artificial bee colony for generating variable t-way test sets," *Interaction*, vol. 7, no. 12, p. 13, 2019.

[14] B. S. Ahmed, "Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing," *Eng. Sci. Technol., Int. J.*, vol. 19, no. 2, pp. 737–753, Jun. 2016.

[15] K. Z. Zamli, B. Y. Alkazemi, and G. Kendall, "A tabu search hyper-heuristic strategy for t-way test suite generation," *Appl. Soft Comput.*, vol. 44, pp. 57–74, Jul. 2016.

[16] K. Z. Zamli, F. Din, B. S. Ahmed, and M. Bures, "A hybrid Q-learning sine-cosine-based strategy for addressing the combinatorial test suite minimization problem," *PLoS ONE*, vol. 13, no. 5, May 2018, Art. no. e0195675.

[17] K. Z. Zamli, F. Din, S. Baharom, and B. S. Ahmed, "Fuzzy adaptive teaching learning-based optimization strategy for the problem of generating mixed strength t-way test suites," *Eng. Appl. Artif. Intell.*, vol. 59, pp. 35–50, Mar. 2017.

[18] A. A. Alsewari, L. M. Xuan, and K. Z. Zamli, "Firefly combinatorial testing strategy," in *Proc. Sci. Inf. Conf.* Cham, Switzerland: Springer, 2018, pp. 936–944.

[19] G. Fraser, A. Arcuri, and P. Mcminn, "A memetic algorithm for whole test suite generation," *J. Syst. Softw.*, vol. 103, pp. 311–327, May 2015.

[20] B. S. Ahmed and K. Z. Zamli, "A variable strength interaction test suites generation strategy using particle swarm optimization," *J. Syst. Softw.*, vol. 84, no. 12, pp. 2171–2185, Dec. 2011.

[21] A. K. Alazzawi, H. M. Rais, and S. Basri, "ABCVS: An artificial bee colony for generating variable t-way test sets," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 4, pp. 259–274, 2019.

[22] A. K. Alazzawi, A. A. B. Homaid, A. A. Alomoush, and A. A. Alsewari, "Artificial bee colony algorithm for pairwise test generation," *J. Telecommun., Electron. Comput. Eng.*, vol. 9, nos. 1–2, pp. 103–108, 2017.

[23] A. A. Alsewari, A. K. Alazzawi, T. H. Rassem, M. N. Kabir, A. A. B. Homaid, Y. A. Alsariera, and N. M. Tairan, "ABC algorithm for combinatorial testing problem," *J. Telecommun., Electron. Comput. Eng.*, vol. 9, nos. 3–3, pp. 85–88, 2017.

[24] A. K. Alazzawi, H. M. Rais, and S. Basri, "Artificial bee colony algorithm for t-way test suite generation," in *Proc. 4th Int. Conf. Comput. Inf. Sci. (ICCOINS)*, Aug. 2018, pp. 1–6.

[25] A. K. Alazzawi, H. M. Rais, and S. Basri, "Parameters tuning of hybrid artificial bee colony search based strategy for t-way testing," *Int. J. Innov. Technol. Exploring Eng.*, vol. 8, no. 5S, pp. 204–212, 2019.

[26] A. K. Alazzawi, H. M. Rais, S. Basri, and Y. A. Alsariera, "PhABC: A hybrid artificial bee colony strategy for pairwise test suite generation with constraints support," in *Proc. IEEE Student Conf. Res. Develop. (SCOReD)*, Oct. 2019, pp. 106–111.

[27] A. R. A. Alsewari and K. Z. Zamli, "A harmony search based pairwise sampling strategy for combinatorial testing," *Int. J. Phys. Sci.*, vol. 7, no. 7, pp. 1062–1072, 2012.

[28] Y. A. Alsariera, M. A. Majid, and K. Z. Zamli, "A bat-inspired strategy for pairwise testing," *ARPN J. Eng. Appl. Sci.*, vol. 10, pp. 8500–8506, Oct. 2015.

[29] G.-G. Wang, B. Chang, and Z. Zhang, "A multi-swarm bat algorithm for global optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, May 2015, pp. 480–485.
[30] Y. A. Alsariera, M. A. Majid, and K. Z. Zamli, "Adopting the bat-inspired algorithm for interaction testing," in *Proc. 8th Ed. Annu. Conf. Softw. Test.*, 2015, p. 14.
[31] A. B. Nasser, Y. A. Sariera, A. A. Alsewari, and K. Z. Zamli, "Assessing optimization based strategies for t-way test suite generation: The case for flower-based strategy," in *Proc. IEEE Int. Conf. Control Syst., Comput. Eng. (ICCSCE)*, Nov. 2015, pp. 150–155.
[32] A. B. Nasser, K. Z. Zamli, and B. S. Ahmed, "Dynamic solution probability acceptance within the flower pollination algorithm for combinatorial t-way test suite generation," in *Intelligent and Interactive Computing*. Singapore: Springer, 2019, pp. 3–11.
[33] T. Mahmoud and B. S. Ahmed, "An efficient strategy for covering array construction with fuzzy logic-based adaptive swarm optimization for software testing use," *Expert Syst. Appl.*, vol. 42, no. 22, pp. 8753–8765, Dec. 2015.
[34] Ş. Gülcü and H. Kodaz, "A novel parallel multi-swarm algorithm based on comprehensive learning particle swarm optimization," *Eng. Appl. Artif. Intell.*, vol. 45, pp. 33–45, Oct. 2015.
[35] K. E. Heraguemi, N. Kamel, and H. Drias, "Multi-swarm bat algorithm for association rule mining using multiple cooperative strategies," *Appl. Intell.*, vol. 45, no. 4, pp. 1021–1033, Dec. 2016.
[36] A. Hatamlou, "Black hole: A new heuristic optimization approach for data clustering," *Inf. Sci.*, vol. 222, pp. 175–184, Feb. 2013.
[37] M. Farahmandian and A. Hatamlou, "Solving optimization problems using black hole algorithm," *J. Adv. Comput. Sci. Technol.*, vol. 4, no. 1, p. 68, Feb. 2015.

HAMSA NAJI NSAIF AL-SAMMARRAIE was born in Baghdad, Iraq, in 1982. She received the B.S. degree in computer science from the University of Baghdad, Baghdad, in 2004. She is currently pursuing the master's degree in computer science from Universiti Teknologi Malaysia (UTM), Johor Bahru, Malaysia. Her main research interests include software engineering, search-based software testing, soft computing, and artificial intelligence.

DAYANG N. A. JAWAWI received the bachelor's degree in software engineering from Sheffield Hallam University, U.K., and the master's degree in computer science and the Ph.D. degree in software engineering from Universiti Teknologi Malaysia (UTM), Malaysia. She is currently an Associate Professor with the Faculty of Engineering, School of Computing, UTM. Her main research interests include software engineering, software reuse, software quality, software testing, requirement engineering, and computing education. A major part of her research projects focuses on rehabilitation and mobile robotics, real-time embedded systems, and precision farming applications.