Innovations in t-way test creation based on a hybrid hill climbing-greedy algorithm

Heba Mohammed Fadhil[1,2], Mohammed Najm Abdullah[1], Mohammed Issam Younis[3]
[1]Department of Computer Engineering, University of Technology, Baghdad, Iraq
[2]Department of Information and Communication, Al-Khwarizmi College of Engineering, University of Baghdad, Baghdad, Iraq
[3]Department of Computer Engineering, College of Engineering, University of Baghdad, Baghdad, Iraq

Article Info

Article history:

Received Mar 18, 2022
Revised Nov 15, 2022
Accepted Dec 12, 2022

Keywords:

Combinatorial testing
Computational intelligence
Covering array
Greedy methods
Hill climbing
Mixed covering arrays

ABSTRACT

In combinatorial testing development, the fabrication of covering arrays is the key challenge by the multiple aspects that influence it. A wide range of combinatorial problems can be solved using metaheuristic and greedy techniques. Combining the greedy technique utilizing a metaheuristic search technique like hill climbing (HC), can produce feasible results for combinatorial tests. Methods based on metaheuristics are used to deal with tuples that may be left after redundancy using greedy strategies; then the result utilization is assured to be near-optimal using a metaheuristic algorithm. As a result, the use of both greedy and HC algorithms in a single test generation system is a good candidate if constructed correctly. This study presents a hybrid greedy hill climbing algorithm (HGHC) that ensures both effectiveness and near-optimal results for generating a small number of test data. To make certain that the suggested HGHC outperforms the most used techniques in terms of test size. It is compared to others in order to determine its effectiveness. In contrast to recent practices utilized for the production of covering arrays (CAs) and mixed covering arrays (MCAs), this hybrid strategy is superior since allowing it to provide the utmost outcome while reducing the size and limit the loss of unique pairings in the CA/MCA generation.

Corresponding Author:

Heba Mohammed Fadhil
Department of Computer Engineering, University of Technology
Baghdad, Iraq
Email: ce.19.15@grad.uotechnology.edu.iq or heba@kecbu.uobaghdad.edu.iq

1. INTRODUCTION

Testing all possible combinations of configuration parameters using a sample of all possible configurations is called combinatorial interaction testing (CIT), which is an alternative to exhaustive testing. An exponential increase of test cases is seen during exhaustive testing; however, the number of configuration options grows at a maximum logarithmic rate [1]–[3]. Testing the interactions between multiple configuration options is critical to reducing the likelihood of interacting problems in software that is extremely flexible. A system having m configuration options, for example, would require an exhaustive test set to comprise m^n test cases in order to cover entirely probable permutations of the configuration parameters in use. When configuration choices are offered, the number of test cases grows at an exponential rate. Due to lack of resources or time, it may be hard to thoroughly test a highly flexible system in its entirety. Over than 70% of computing system failures are caused by the interplay of configuration settings in two directions at the same time [4].

In software testing, test cases are represented as combinatorial objects known as covering arrays (CAs) and mixed covering arrays (MCAs). Constructing a CA/MCA that is both effective and efficient is necessary in order to get the most out of pair-wise testing. This is an intractable problem. Thus, researchers' key goal is to develop an effective strategy for building an optimal CA/MCA. As a consequence of this, it is more practical to make use of approximate approaches that are able to produce (almost) optimal solutions in a reasonable amount of time when the problem at hand is extremely complex. Heuristics and metaheuristics are two types of approximation strategies that can be used to solve problems [5], [6]. Whenever dealing with CA, it is advisable to use metaheuristics like: Hill climbing (HC) [7], harmony search algorithm (HSA) [8], particle swarm optimization (PSO) [9], tabu search (TS) [10], ant colony optimization (ACO) [11], and simulated annealing (SA) [12].

For the advancement and improvement of CA, a variety of meta-heuristic methodologies are available. This study anticipates one-test-at-a-time technique to build a valid CA with N rows intended for a specific CIT problem instance in several procedures. Meta-heuristic search algorithms are used to condense the number of arrays used in the initial CA repeatedly. This process is repetitive until all arrays are eliminated. Once a predetermined stopping criterion, including the amount of retries or the allotted time constraint, is met, the procedure is repeated. Greedy algorithm is a straightforward and fast method since it only selects solutions that satisfy greedy requirements. Numerous papers combined greedy with their hybrid algorithm like [13], [14] in the aim that the greedy solution will assist the hybrid algorithm in getting closer to the nearest solution. To address these concerns, this article offers a new greedy technique for array generating limitations based on the HC algorithm, named hybrid greedy hill climbing algorithm (HGHC). As with rival meta-heuristic-based methods, HGHC produces results that are sufficiently optimum in comparison to general computational-based and meta-heuristic strategies [15], [16].

Although a greedy strategy provides good coverage and run speed, there are some trials in which it fails to provide the test cases that are required [17]. It is proposed in this study that the HGHC method be used to tackle this problem by integrating the HC and greedy algorithms into a single solution. In this technique, the HGHC algorithm takes use of the iterative nature of the HC algorithm, which always results in feasible test cases, while the greedy section is added later to boost the optimality of the solutions produced by the HC algorithm, as Figure 1 found in section 5.

This paper has seven more sections: section 2 reflects the work that is related with this. Section 3 presents combinatorial testing methods; section 4 describes meta-heuristic algorithms. In section 5, the specifics of the hybridization strategy that has been presented. Section 6 the experimental data is summarized and analyzed. Section 7 assesses the statistical analysis. After that, section 8 contains the conclusion and specifics on future works.

2. RELATED WORK

Numerous algorithms are used to construct near optimal CAs and MCAs, including those that use algebraic, greedy, metaheuristic, and random techniques for construction. It is not uncommon for mathematicians to use algebraic methods. Because separately parameter must have the same number of values, algebraic approaches despite their speed are rarely used in CIT.

Greedy methods are preferred by the software testing community when it comes to producing CAs. There are two methods for building CAs with a greedy approach: one parameter per test and one-test per attempt. The CA is built up row by row, and the manner in which each row is built can vary depending on the approach used. When it comes to one test at a time (OTAT) methodologies, a comprehensive test case is constructed for each iteration that incorporates the interface components that have been the most recently uncovered. The same method is used to cover all areas of interaction with the system. Several other tools and tactics are presented in the literature, all of which are derived from the OTAT method. such as automatic efficient test generator (AETG) [18], pairwise independent combinatorial testing (PICT) [19], classification-tree editor extended logics (CTE-XL) [20], deterministic density algorithm (DDA) [21], in parameter order general (IPOG) [22], GTWay [23], in parameter order d-construction (IPOD) [24], and genetic multi-parameter-order-algorithm (MIPOG)/(GMIPOG) [25] strategies are examples of techniques that have adopted this approach.

In recent years, researchers have looked into metaheuristic techniques such as SA [12], [26] and HSA [8], [27]. Heuristic techniques were used by Cohen *et al.* [26] to generate CAs and MCAs of strength t=3, as well as the experimental outcomes indicated that heuristic strategies outclassed greedy approaches for strength-2 CAs but not for higher strength CAs, notably at t=3. Regarding the number of iterations required to reach an acceptable solution, HC surpassed SA in producing equal lower bounds.

Many approaches have been developed by Cohen *et al.* [28]–[30] which produce uniform covering arrays, and variable coverage arrays using a mixture of varied methods (for example, algebraic and computational techniques). Constrained systems complicate CIT because the resulting CA may contain certain parameter values that are incompatible with the restrictions. As a result, it is best to exercise caution when dealing with such limits. Garvin *et al.* [6] developed an extension of the SA technique to create CAs for limited interaction testing. Calvagna and Gargantini [31] make usage of satisfiability modulo theory (SMT) solvers, which they developed to produce pair-wise test coverage for CA.

Alazzawi *et al.* proposed several studies in the field of t-way testing, some of them employed hybrid techniques such a hybrid artificial bee colony (HABC) approach [32] constructed on the HABC algorithm and PSO to build optimal test suite with variable strength interaction. The hybrid nature of PSO is due to the fact that it was integrated into the artificial bee colony (ABC) as an exploitation agent. ABC's performance is improved by PSO's information-sharing via the weight factor. A T-way generating approach for both a uniform and variable strength test suite called (ABCVS) is [33] by utilizing the ABC technique to reduce the overall size of a test suite while simultaneously improving the interaction between tests in the suite. Alazzawi *et al*. [34] proposed a new meta-heuristic-based t-way approach called hybrid artificial bee colony (HABCSm). It combines the advantages of the ABC algorithm and PSO. HABCSm is the first t-way strategy to use the HABC algorithm with hamming distance as its fundamental approach for creating a final test set and final selection criterion for boosting the discovery of new solutions.

Gravitational search test generator is the name given to the novel t-way method that was developed by Htay *et al.* [35] and is based on the gravitational search algorithm (GSA). The most significant contribution of this research is the adaptation of GSA to the production of t-way test data for the first time. Recently, Guo *et al.* [36] provides a synergistic solution for the constrained covering array generation (CCAG) problems that is initially based on quantum particle swarm optimisation (QPSO). Three auxiliary procedures are presented to increase QPSO's performance: contraction-expansion coefficient adaptive modification, differential evolution, and discretization.

3. COMBINATORIAL TESTING

An insight to combinatorial testing is provided in this section. Combinatorial testing can be used for a plethora of ways including drug screening and data compression as well as graphical user interface (GUI) testing and web application testing. More than one area is covered by this umbrella, including drug screening and data compression. There is at least one CA/MCA for every t-way combination of parameter value. Since CAs and MCAs have proven to be useful in numerous industries, researchers are looking at the best approaches to develop optimal CAs and MCAs [37], [38].

3.1. Covering arrays

It's called a covering array, and its notation is CA (N; t, k, v). A two-dimensional array with K signifies how many parameters there are in S; N indicates how many columns there are; and *v* represents how many possible values each parameter might have. *t* denotes how strong an interaction there is. Ideally, a CA should have a minutest number of rows in order to mollify all of the criteria of the full covering array. An abbreviation for the covering array number is covering array number (CAN); it stands for (t, k, v). An input parameter is represented by a column and the values in that column indicate its respective input parameter's range [1], [39].

3.2. Mixed covering arrays

In this case, the cardinality vectors v1v2...vk correspond to the values for each column in the mixed covering array, resulting in an N-by-k two-dimensional array. MCAs have the following two features, both of which are present: At least once, the rows of each N t sub-array contain all t-tuples of values from each of the N t columns, with the exception of those in the set Si where |Si|=vi. This is true for all N t sub-arrays. It is denoted by the symbol MCA (N (t, k, (v1 v2...vk))), and it represents the smallest number of variables for which an MCA exists, which is also known as the mixed covering array number. It is possible to represent MCAs in a shorthand notation by merging equal elements in (vi: 1... k) by merging equal values [40].

4. METAHEURISTIC APPROCH

Optimizing techniques that start with the best possible answer and enhance it over time are known as metaheuristics. This work employs greedy and HC metaheuristic search approaches to solve optimization problems. A metaheuristic is a way of improving a problem by iteratively improving a prospective solution's quality. Metaheuristics can seek large spaces of possible solutions with little or no prior knowledge about the problem. In the absence of a perfect solution, metaheuristics assure a workable one and prevent the issue from being stalled. Optimization concerns three operators: i) refining the optimal solution by either reducing it or increasing it, ii) the objective function is controlled by moves, and iii) a conventional of constraints that allows the moves to exclude some data while keeping others [41], [42].

It's possible to solve difficult issues using metaheuristic algorithms, even though it's impossible to ensure that they'll find the global best solution. Metaheuristic-based algorithms are used to search the issue space in an effort to find a better solution. This type of navigation is guided by an understanding of the issue and the hope of locating a global optimum. The majority of metaheuristic implementations are based on local search algorithms as well as population search algorithms.

The initial step in population-based approaches is to generate a collection of solutions chosen at random. A subsequent step is to combine characteristics from many solutions in order to create better ones. It is possible to simultaneously scan large areas of the search space with an algorithm that uses a large population. These algorithms may miss the local optimum in each section. Because of this, it's possible that the algorithm won't produce the best outcome. A few examples of population-based algorithms in use are the genetic algorithm, bee colony optimization, and particle swarm optimization. Local search-based strategies begin with a single solution, which is then iterated upon by a neighborhood-based strategy. Initially when it finds a locally optimal solution, the algorithm comes to a halt. It is possible to break down the search area into smaller parts. In contrast to population-based search methods, local search approaches focus on a smaller search area in order to determine the most suitable way to perform the search. In spite of this, these strategies do not cover a large portion of the possible search areas. HC is a local search method that is regarded to be the most basic. Population-based approaches can be boosted in their ability to find local optima by using this technique [12], [43].

5. THE PROPOSED APPROACH

This section provides in-depth information regarding the HGHC algorithm, which was developed for the first time. Demonstrate how it incorporates the benefits of both the greedy and the HC algorithms when they are combined to form a hybrid algorithm, and then explain why HGHC outperforms both the greedy algorithm and the HC algorithm when they are deployed separately. In addition, the whole procedure for developing HGHC test data is provided in order to achieve branch coverage with the fewest possible test cases by applying HGHC. Using both greedy and HC algorithms, this is the first study of its kind to create a hybrid solution to solve a wide variety of issues, making it unique in the area. Beginning with its intrinsic limitations, the HC is prone to becoming caught in a local optimal, rendering it useless for determining discrete issues. Nevertheless, the greedy method assumes a much simpler premise, making it easier to incorporate, and is more effective; but it does not ensure that it will give a globally optimal solution. Due to the complementing nature of the two procedures, it appears as though they might be utilized in conjunction to tackle a wide variety of optimization issues. Our hybrid approach, which accepts as input a preliminary covering array generated previously using the greedy method and then adds additional rows to it. When adding a new row to the existing covering array, the initial stage is to use a greedy method to allocate distinct element of the new row to the current covering array. If an element has an unallocated value, the greedy algorithm iteratively tests separately possible value assignment to the unassigned element and formerly chooses the assignment that results in the fewest missing tuples. Tuples that were not covered in the first stage of the algorithm are covered in the second stage, which is when the HC method is used as explained in Algorithm 1. If the HC algorithm fails to cover the entire array, repeat the HC algorithm by adding a new row to the existing array and running it as stated in Figure 1. A covering array is shaped in two steps: first, build the covering perfect hash families (CPHF). A CPHF (n;k,v,t) is an array of size n * k over $\boldsymbol{F}_{v}^{t}$ such that every sub array of t columns cotains at least one row with a covering tuple as Figure 2. Let's take an exmple of CPHF (2;16,5,3); here the elements of $\boldsymbol{F}_{5}^{3}$ are written as $c_0\, c_1\, c_2$ instead of $(c_0, c_1, c_2)^T$:

This array, which is composed of three columns, has at minimum one covering tuple present in each and every one of its rows [44]. As explained in Algorithm 2, CPHFs is utilized in the first step to quickly construct huge covering arrays, which is followed by the addition of rows to the resulting array while maintaining the same number of columns throughout both processes. Non-redundant members from one row can be copied to redundant elements in subsequent rows using this operation if it uses the greedy method. The next stage of the metaheuristic method is to fill in the missing tuples that were introduced by the row deletion. A new round of optimization takes place if the algorithm completes the array.

Algorithm 1: Hill climbing algorithm

```
Step 1: Evaluate the initial state. If it is a goal state then stop and return success.
Otherwise, make intial state as current state.
Step 2: Loop until the solution state is found or there are no new operators present whick
can be applied to the cirrent state.
  a) Select state that has not been yet applied to the cureent state and apply it to
     produce a new state
```

```
  b)  Perform these to evaluate new state
     i.   If the current state is a goal state, then stop and return success.
    ii.   If it is better than the current state, then make it cureent state and proceed
          further.
   iii.   If it is not better than the current state, then continue in the loop until a
          solution is found.
Step 3: Exit
```

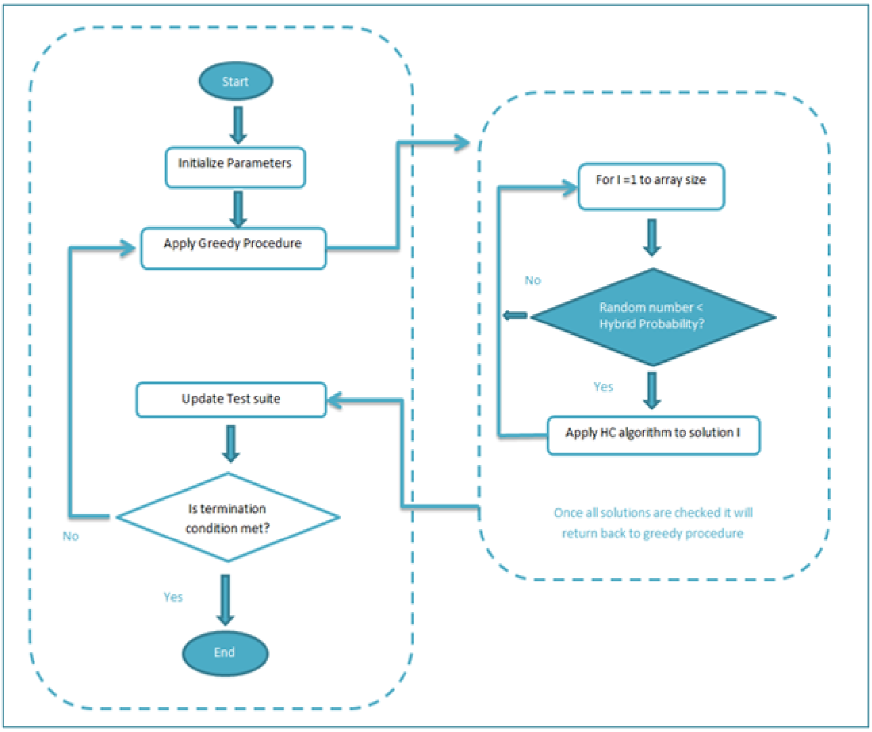

Figure 1. The proposed HGHC approach

Algorithm 2: CPHF algorithm

```
Input: n: target number of rows, t: strength, k: number of factors, q:v symbols;
Output: A; a CPHF(M;k, q, t) with m ≤ n upon termination
Construct an n × k array A with each entry chosen independently;
Repeat
   Set covered:= true;
   Set M:=0;
   for each t-set T of columns, in the same fixed order do
     if T is covered in A then
       Let R be the index of the first row covering T;
       Set M:=max(M,R)
else
       Set covered:= false;
       Set missing set;= T;
       Break;
If not covered then
  Choose all the entires in the t columns of missing-set independently and uniformly at
  random;
Until covered = true;
Output the first M rows of A;
```

$$\begin{pmatrix} 312 & 331 & 340 & 310 & 230 & 010 & 402 & 143 & 003 & 332 & 104 & 132 & 204 & 442 & 042 & 043 \\ 423 & 223 & 032 & 113 & 102 & 003 & 231 & 134 & 030 & 443 & 200 & 241 & 203 & 043 & 124 & 140 \end{pmatrix}$$

Figure 2. Covering perfect hash families (CPHF) array

For an improved covering array, the final step of the HGHC technique involves optimizing the covering array called A1. Using a greedy approach, the optimization algorithm first searches A1 for redundant items. The optimization method then uses a greedy approach to remove a row from A1 and then an HC algorithm to fill in any missing tuples that may have appeared as a result of removing the row. Once a row has been cleared, the procedure repeats again until all of the empty tuples have been filled. In Algorithm 3, the algorithm is depicted in pseudocode.

Algorithm 3: Optimization pseudocode

```
Fungtion optimization (A 1)(
     Do A 2 A 1;
     Find redundant elements in A 1 using a greedy algorithm;
     A row from A 1 using a greedy algorithm is deleted;
        If a missing tuple is in A 1
              Then cover missing tuples in A 1 using a HC
algorithm;
        Until the HC algorithm cannot cover the missing tuples in A 1;
     return A 2;
```

In order to concealment a relatively insignificant number of uncovered combinations, a technique that constructs complete CPHFs may necessitate the insertion of an entire row to the derivative covering array. To wrap up, consider how a greedy algorithm works in practice. In this case, the goal is to discover a solution as quickly as possible by selecting a choice that appears to be local optimal at the time. Using the greedy method, each iteration generates a random sample from an unknown distribution. The greedy method has an effect on the distribution's mean and variance. If it's limited to a single component, the iterative solution will be the same. The distribution's mean equals the greedy solution's value, and its variance is zero. A search that is conducted in this manner repeatedly is referred to as an iterative search. Prior decisions are relevant, but the option is independent of those made in the future or of those inherent in the sub-problem. In other words, the greedy still commonly employed as a backup technique or to generate accurate estimations of the optimal for particular instance scenarios. Meanwhile, the greedy strategy works well for problems involving optimal substructures, where the globally optimal solution embraces local solutions to subproblems.

6. RESULTS AND DISCUSSION

The results reported in Tables 1 to 5 was obtained using the hybrid approach outlined in this section. The hybrid approach uses CPHFs massively reduce the time it takes to build covering arrays when using a metaheuristic technique. An array with as many rows as possible is a good place to start when developing the procedure. The greedy method is exceptionally fast when only t-combinations that may contain missing tuples are considered. Column vectors of length v can be substituted for the CPHF's elements to create an array subarray arrays that cover the entire t-tuple domain or have only a few missing tuples. This can be seen when the CPHF's row count exceeds v and only a few more rows are required to complete the coverage. A comparison of current state-of-the-art approaches is presented using Python to code the suggested algorithm on a computer with a Core i7-7th Generation Intel processor, 8 Giga of random-access memory, and Windows 10.

Table 1. The array size of the proposed approach vs other approaches at t = 2.

	Jenny	TConfig	PICT	IPOG	CPSO	DSPO	GS	GALP	ABCVS	HABS	HABCSm	Proposed HGHC
CA (N; $2,2^7$)	8	7	7	7	7	7	6	6	NA	7	7	6
CA (N; $2,3^3$)	9	10	10	9	9	9	9	9	9	9	9	9
CA (N; $2,3^4$)	13	10	13	9	9	9	9	9	9	9	9	11
CA (N; $2,3^5$)	14	14	13	15	11	11	11	11	11	12	11	13
CA (N; $2,3^6$)	15	15	14	15	14	14	13	13	13	13	13	14
CA (N; $2,3^7$)	16	15	16	15	15	15	14	14	15	15	14	15
CA (N; $2,3^8$)	17	17	16	15	15	15	15	15	15	15	15	16
CA (N; $2,3^9$)	18	17	17	15	16	15	15	15	16	16	15	15
CA (N; $2,3^{10}$)	19	17	18	15	16	16	16	16	17	17	16	17
CA (N; $2,3^{11}$)	17	20	18	17	16	17	16	16	17	18	17	18
CA (N; $2,3^{12}$)	19	20	19	21	17	16	16	16	18	18	18	18
CA (N; $2,4^7$)	28	28	27	29	25	24	24	24	NA	25	24	24
CA (N; $2,5^7$)	37	40	40	45	36	34	36	35	NA	37	34	34

Table 2. The array size of the proposed approach vs other approaches at t=3.

	Jenny	TConfig	PICT	IPOG	CPSO	DSPO	GS	GALP	ABCVS	HABS	HABCSm	Proposed HGHC
CA (N; 3,2^7)	14	16	15	16	12	15	12	12	NA	14	13	12
CA (N; 3,2^8)	14	18	17	18	16	16	14	12	NA	NA	NA	13
CA (N; 3,2^9)	17	20	17	20	16	16	16	16	NA	NA	NA	16
CA (N; 3,2^{10})	18	20	18	20	16	16	16	16	NA	NA	NA	18
CA (N; 3,3^4)	34	32	34	32	38	41	38	37	27	27	27	32
CA (N; 3,3^5)	40	40	43	41	30	28	27	27	38	39	39	28
CA (N; 3,3^6)	51	48	48	46	42	33	43	40	44	43	43	32
CA (N; 3,3^7)	51	55	51	55	49	48	49	48	49	47	46	48
CA (N; 3,3^8)	58	58	59	56	53	52	54	52	54	53	45	55
CA (N; 3,3^9)	62	64	63	63	58	56	58	56	58	56	56	61
CA (N; 3,3^{10})	65	68	65	66	61	59	61	59	62	61	61	64
CA (N; 3,3^{11})	65	72	70	70	63	63	63	62	66	68	65	62
CA (N; 3,3^{12})	68	77	72	73	68	65	67	65	70	72	68	64
CA (N; 3,4^7)	124	122	124	112	115	112	116	112	NA	114	110	110

Table 3. The array size of the proposed approach vs other approaches at t=4

	Jenny	TConfig	PICT	IPOG	CPSO	DSPO	GS	GALP	ABCVS	HABS	HABCSm	Proposed HGHC
CA (N; 4,2^7)	31	36	32	35	24	31	27	24	NA	29	27	28
CA (N; 4,2^8)	37	38	35	39	32	32	30	29	NA	NA	NA	32
CA (N; 4,2^9)	37	41	41	41	33	34	33	25	NA	NA	NA	35
CA (N; 4,2^{10})	39	45	43	46	37	34	25	26	NA	NA	NA	38
CA (N; 4,3^5)	109	97	100	97	94	81	88	88	98	81	81	81
CA (N; 4,3^6)	140	141	142	141	132	131	129	129	135	134	132	131
CA (N; 4,3^7)	169	166	168	167	153	150	152	152	157	155	149	148
CA (N; 4,3^8)	187	190	189	192	174	171	171	171	179	177	159	171
CA (N; 4,3^9)	206	213	211	210	191	187	187	189	197	196	185	185
CA (N; 4,3^{10})	221	235	231	233	211	206	206	206	215	217	212	209
CA (N; 4,3^{11})	236	258	249	251	226	221	223	221	234	237	229	220
CA (N; 4,3^{12})	252	272	269	272	242	237	236	237	251	257	246	236

Table 4. The array size of the proposed approach vs other approaches at t>4

	Jenny	TConfig	PICT	IPOG	CPSO	DSPO	GS	GALP	Proposed HGHC
CA (N; 5,3^7)	458	477	452	466	441	428	431	432	429
CA (N; 6,3^8)	1,466	1,515	1,455	1409	1,397	1,402	1,398	1,392	1,396
CA (N; 7,3^9)	4,746	>day	4,618	NS	4,422	4,427	4,437	4,425	4,422
CA (N; 8,3^{10})	14,999	>day	14,599	NS	13,925	13,933	13,907	13,903	13,909
CA (N; 9,3^{11})	47,009	>day	45,521	NS	43,587	>day	43,808	43,543	45,520
CA (N; 10,3^{12})	147004	>day	141,990	NS	135,498	>day	136,096	135,381	135,391
CA (N; 11,3^{12})	3,057,977	>day	278,993	NS	268,173	>day	267,630	267,803	267,630
CA (N; 12,2^{14})	9,422	>day	9,112	NS	8,882	8,972	8,890	8,904	8,890
CA (N; 13,2^{14})	13,251	>day	12,441	NS	11,588	>day	10,251	11,051	10,250
CA (N; 14,2^{15})	26,579	>day	25,036	NS	23,889	>day	23,377	22,642	23,360
CA (N; 15,2^{16})	53,977	>day	51,127	NS	45,838	>day	46,575	41,820	42,990

Table 5. For various MCA configurations, a comparison of existing techniques

	Jenny	TConfig	PICT	IPOG	CPSO	DSPO	GS	GALP	Proposed HGHC
MCA(N; 2, 5^1 3^8 2^2)	23	22	15	16	15	NA	21	20	15
MCA(N; 2, 7^1 6^1 5^1 4^6 3^8 2^3)	50	51	42	42	42	48	51	48	44
MCA(N; 3, 5^2 4^2 3^2)	131	136	100	106	108	NA	NA	113	100
MCA(N; 3, 10^1 6^2 4^3 3^1)	399	414	360	361	361	385	393	365	360

First, the CPHF is constructed using the first of two algorithms; the second used algorithm is to fill in the missing tuples afterwards removing a row from the covering array. Range of 2 ≤v ≤12 and 2 ≤t ≤10 was used to evaluate the suggested system's performance. A combination of greedy and metaheuristic algorithms, as well as the partition of the process into three stages, has resulted in a significant number of improvements for our technique. It is necessary to compare HGHC's effectiveness in decreasing the size of the test suite with that of other existing approaches as deliberated in [34]–[36]. A total of five sets of comparisons are made in the experiment:

– HGHC is compared to the results of techniques for various setups involving t=2.

- HGHC is compared to the results of techniques for various setups involving t=3.
- HGHC is compared to the results of techniques for various setups involving t=4.
- HGHC is compared to the results of techniques for various setups involving for CA, t varied from 2 to 10.
- HGHC is compared to the results of techniques for various setups involving for MCA different configurations involving: MCA (N; 2, 5^1 3^8 2^2), MCA (N; 2, 7^1 6^1 5^1 4^6 3^8 2^3), MCA (N; 3, 5^2 4^2 3^2), and MCA (N; 3, 10^1 6^2 4^3 3^1).

Based on a comparison of the findings, it is clear that the HGHC strategy surpasses the original existing techniques (hill climbing and greedy) in terms of covering array size. As shown in Figure 3, that the original two algorithms (hill climbing and greedy) are somewhat close, while the HGHC algorithm produces less CA size. Figure 4 shows that the two original algorithms, hill climbing and greedy, are comparable in certain tests, but the HGHC approach results in a smaller CA size from the fifth case and beyond. Regarding the value of t=4, we can observe that the outcomes are rather near to one another in terms of the validity of the HGHC algorithm. as illustrated in Figure 5.

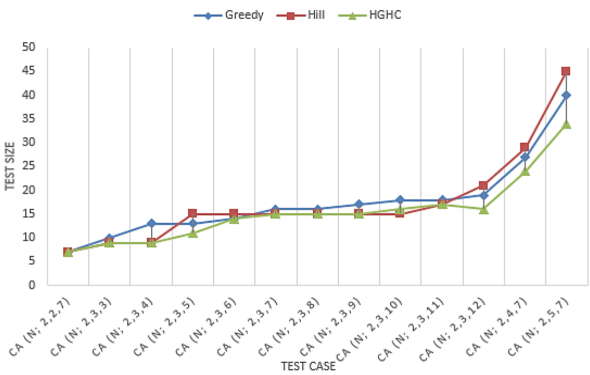

Figure 3. New algorithm vs original performance for t=2

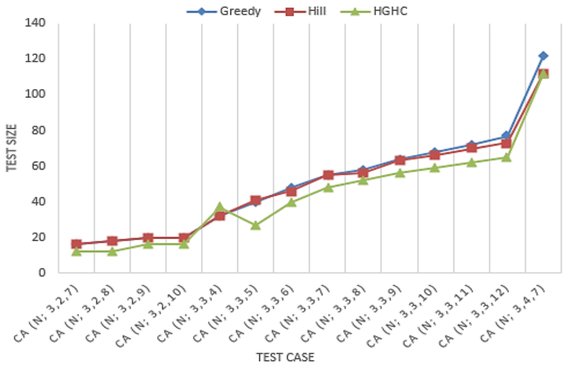

Figure 4. New algorithm vs original performance for t=3

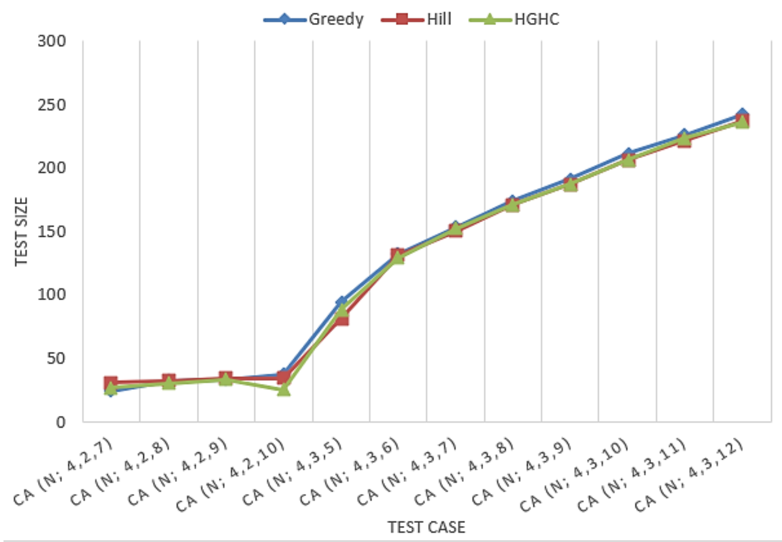

Figure 5. New algorithm vs original performance for t=4

According to results, meta-heuristic-based techniques surpass those that are based on computation in terms of performance. Compared to existing techniques, the suggested HGHC strategy performed well, as shown in Table 1 Similar findings were achieved by HGHC, GALP, and HABCSm in configurations no. (2, 8, and 12) when the interaction was 2. In terms of configurations no. (3, 4, and 7) conventional PSO (CPSO) and discrete particle swarm optimization (DSPO) came up with the ideal test set size. The worst results were obtained by using Jenny, TConfig, PICT, IPOG and harmful algal blooms (HABS). For setups no. (7, 12, and 13) When the interaction is equal to 3, HGHC discovered the appropriate size for the test set. As with genetic strategy (GS) and GALP, the HABC method yielded the smallest possible test set for configuration No. (1 and 3). As for configuration no. (1, 3 and 4) the CSPO method provided the ideal test set size, whereas the HABCSm method produced the most effective size of the test set with the other three configurations (9,10 and 14). When compared to HABC, Jenny, TConfig, PICT, IPOG, ABCVS, and HABCSm consistently produced the lowest outcomes as Table 2. Constructed on the results obtainable in Table 3, it is clear that the HGHC method achieved excellent results for the configurations no. (7, 11 and 12); as configuration no. 9 produced a competitive test set that was somewhat near to the ideal test set. In Table 4, when the interaction value is equal to 4, HGHC provided the ideal test set size for configurations no. (3, 5, 7, and 9), whereas GALP created the optimal test set size for configurations no. (2, 4,6,10, and 11) correspondingly. HABC approach yielded a set of tests that was competitive with the optimal set for configurations 1, 3, and 4, as can be Table 5.

7. STATISTICAL EVALUATION

The use of statistical analysis is yet another approach that can be taken in order to evaluate the proposed strategy in terms of its efficacy and determine the significance of the strategy. With a confidence level of 95 percent (i.e., α=0.05), the Wilcoxon signed-rank test is utilized in order to evaluate the HABC strategy in comparison to other existing strategies from Tables 1 to 4. The Wilcoxon signed-rank test will be used to determine if there is a statistically significant difference between the suggested approach and the other strategies being examined for this comparison. This test is ideal for measuring the difference between the two sets because it compares them side by side. When multiple comparisons are involved, Bonferroni-Holm correction (i.e., Holm's sequentially rejective step-down process) was used to adjust value. the asymptotic significance (2-tailed) of the first value is used to scale the data [45]. As a result, Holm is recalculated using the following factors,

$$\propto Holm = \frac{\propto}{M-i+1}$$

where M is the total number of paired comparisons, and $\boldsymbol{i}$ is the number of tests. HGHC has three ranks: HGHC>, HGHC<, and HGHC= are used to evaluate it. Other existing tactics are either greater, smaller or equal to the suggested strategy's results. Asymptotic sig. (2-tailed) and Z are the two values that have a statistical test component asymp. sig. (2-tailed) shows a significant difference between the two sets, and the corresponding hypothesis will be retained if the value exceeds Holm. The Z value is not addressed in this study (i.e., not considered). If the asymp. sig. (2tailed) value is less than Holm, the associated hypothesis is rejected. Once a certain null hypothesis cannot be ruled out, the rest of the hypotheses are also kept. As there is no test configuration for which a result is provided, the strategies with N/A results are regarded as incomplete and ignored samples. The statistical findings from the wilcoxon test for Tables 1 to 4 are presented in Tables 6 to 9, which may be seen. A considerable difference may be seen in asymp between HABS and HABCSm alone. From Table 6, HGHC is clearly better to all other approaches, with the exception of HABC and HABCSm. A look at Table 7 reveals that even though HGHC outperformed Jenny, IPOG, CSPO, DSPO, GS, GALP, PICT, HABC, and HABCSm, it was inferior to TConfig. In Table 8, HGHC did better than TConfig, IPOG, CSPO, DSPO, GS, GALP, HABC, and HABCSm, but not as well as Jenny and PICT. The findings of the tests presented in Table 9. shows HGHC is significantly different from those of CSPO, GS, JENNY, and PICT. The GALP strategy, on the other hand, outperformed the performers of the HABC strategy. The other approaches' conclusions are labeled "missing" because they are either unavailable or do not support a certain set up.

Table 6. Analysis of data from Table 1 using the wilcoxon signed rank sum test

Pairs	Ranks			Test statistics			Conclusion
	HGHC<	HGHC>	HGHC=	Z	Asymp. sig. (2-tailed)	α Holm	
HGHC-CPSO	9	3	3	0.8664	0.0707	0.05	Reject the null hypothesis
HGHC-DSPO	7	3	3	1.9439	0.0564	0.025	Reject the null hypothesis
HGHC-GALP	11	7	7	2.3102	0.0207	0.0167	Reject the null hypothesis
HGHC-GS	38	18	18	1.8363	0.0679	0.0125	Reject the null hypothesis
HGHC-HABC	9	4	4	0.07	0.11	0.0100	Retain the null hypothesis
HGHC-HABCSm	29	17	17	2.0304	0.0401	0.0083	Retain the null hypothesis
HGHC-IPOG	10	6	6	1.0703	0.0036	0.0021	Reject the null hypothesis
HGHC-JENNY	4	0	0	0.9435	0.0067	0.0060	Reject the null hypothesis
HGHC-PICT	6	4	4	2.6656	0.0079	0.0050	Reject the null hypothesis
HGHC-TCONFIG	9	3	3	2.6229	0.0085	0.0045	Reject the null hypothesis

Table 7. Analysis of data from Table 2 using the wilcoxon signed rank sum test

Pairs	Ranks			Test statistics			Conclusion
	HGHC<	HGHC>	HGHC=	Z	Asymp. sig. (2-tailed)	α Holm	
HGHC-CPSO	15	11	11	0.9478	0.0038	0.4258	Reject the null hypothesis
HGHC-DSPO	21	13	13	0.169	0.0167	0.932	Reject the null hypothesis
HGHC-GALP	20	14	14	0.0845	0.0125	0.100	Reject the null hypothesis
HGHC-GS	17	17	17	0.6516	0.0100	0.5703	Reject the null hypothesis
HGHC-HABC	15	13	13	0.8885	0.0083	0.4258	Reject the null hypothesis
HGHC-HABCSm	15	11	11	0.8885	0.0071	0.4258	Reject the null hypothesis
HGHC-IPOG	5	0	2	2.5205	0.0063	0.0141	Reject the null hypothesis
HGHC-JENNY	6	3	0	2.6656	0.0050	0.0039	Reject the null hypothesis
HGHC-PICT	4	2	1	2.6661	0.0045	0.0039	Reject the null hypothesis
HGHC-TCONFIG	3	0	0	2.5205	0.0321	0.0014	Retain the null hypothesis

Table 8. Analysis of data from Table 3 using the wilcoxon signed rank sum test

Pairs	Ranks			Test statistics			Conclusion
	HGHC<	HGHC>	HGHC=	Z	Asymp. sig. (2-tailed)	α Holm	
HGHC-CPSO	3	0	0	2.5205	0.025	0.0141	Reject the null hypothesis
HGHC-DSPO	7	4	4	0.9439	0.0167	0.4164	Reject the null hypothesis
HGHC-GALP	7	3	5	1.5213	0.0125	0.1501	Reject the null hypothesis
HGHC-GS	7	5	5	1.0215	0.0100	0.1493	Reject the null hypothesis
HGHC-HABC	7	5	0	2.3664	0.0083	0.0225	Reject the null hypothesis
HGHC-HABCSm	7	5	2	1.1531	0.0071	0.2945	Reject the null hypothesis
HGHC-IPOG	4	0	1	2.0205	0.0167	0.0143	Reject the null hypothesis
HGHC-JENNY	15	11	11	2.5115	0.0125	0.0441	Retain the null hypothesis
HGHC-PICT	5	0	2	2.0005	0.0100	0.0514	Retain the null hypothesis
HGHC-TCONFIG	6	3	0	2.1105	0.0083	0.0742	Reject the null hypothesis

Table 9. Analysis of data from Table 4 using the wilcoxon signed rank sum test

Pairs	Ranks			Test statistics			Conclusion
	HGHC<	HGHC>	HGHC=	Z	Asymp. sig. (2-tailed)	α Holm	
HGHC-CPSO	13	11	11	1.3624	0.0925	0.0100	Reject the null hypothesis
HGHC-GALP	25	25	25	0.2548	0.8384	0.0083	Retain the null hypothesis
HGHC-GS	15	11	11	0.9802	0.0604	0.0171	Reject the null hypothesis
HGHC-JENNY	6	0	2	2.0361	0.0079	0.0063	Reject the null hypothesis
HGHC-PICT	3	1	0	2.8031	0.0059	0.0050	Reject the null hypothesis

8. CONCLUSION

Findings from comparative studies shows that the proposed strategy outperforms existing techniques when it comes to CA/MCA generation quality and the number of generations it takes to get there. Most of the time, when comparing CA/MCA size; The new method outperforms conventional methods. encompassing orders of coverage arrays $2 \leq v \leq 5$ and strengths $2 \leq t \leq 10$ were constructed using an innovative hybrid greedy-metaheuristic technique. This proves that it is a highly competitive technology for the production of such arrays of coverings. In order to achieve the best outcomes, one needs to use both greedy as well as metaheuristic algorithms. When it comes to evaluating the composition of compost, uniform cover arrays of degree four are used, and it was offered as an illustration of how they can be used. The HGHC experimentations were premeditated and carried out appropriate to appraise the influence of each decision on the resultant array size. Observations based on the data allow us to say that i) the framework's configurations have a substantial impression on the performance of the covering array. When compared to established methods such as IPOG, PICT, and DSPO, ii) find that the optimal configuration has apparent advantages, and in some systems, it is even improved than the existing methods. As a result, the proposed HGHC algorithm may prove to be a more efficient tool for autonomously producing test data, particularly because it ensures adequate coverage, optimality, and minimal complexity. In the future, investigate to see if the greedy strategy is capable of being utilized to generate CAs and MCAs with higher strength, and consider the scenarios of seeds and limitations in the production of a covering array.

REFERENCES

[1] S. Sabharwal, P. Bansa, and N. Mittal, "Construction of strength two mixed covering arrays using greedy mutation in genetic algorithm," *International Journal of Information Technology and Computer Science*, vol. 7, no. 10, pp. 23–34, Sep. 2015, doi: 10.5815/ijitcs.2015.10.04.

[2] M. I. Younis, "DEO: a dynamic event order strategy for T-way sequence covering array test data generation," *Baghdad Science Journal*, vol. 17, no. 2, p. 575, May 2020, doi: 10.21123/bsj.2020.17.2.0575.

[3] M. I. Younis, A. R. A. Alsewari, N. Y. Khang, and K. Z. Zamli, "CTJ: input-output based relation combinatorial testing strategy using

jaya algorithm," *Baghdad Science Journal*, vol. 17, no. 3(Suppl.), p. 1002, Sep. 2020, doi: 10.21123/bsj.2020.17.3(Suppl.).1002.

[4] D. R. Kuhn, D. R. Wallace, and A. M. Gallo, "Software fault interactions and implications for software testing," *IEEE Transactions on Software Engineering*, vol. 30, no. 6, pp. 418–421, Jun. 2004, doi: 10.1109/TSE.2004.24.

[5] A. Arram, M. Ayob, and A. Sulaiman, "Hybrid bird mating optimizer with single-based algorithms for combinatorial optimization problems," *IEEE Access*, vol. 9, pp. 115972–115989, 2021, doi: 10.1109/ACCESS.2021.3102154.

[6] B. J. Garvin, M. B. Cohen, and M. B. Dwyer, "Evaluating improvements to a meta-heuristic search for constrained interaction testing," *Empirical Software Engineering*, vol. 16, no. 1, pp. 61–102, Feb. 2011, doi: 10.1007/s10664-010-9135-7.

[7] C. Nie and H. Leung, "A survey of combinatorial testing," *ACM Computing Surveys*, vol. 43, no. 2, pp. 1–29, Jan. 2011, doi: 10.1145/1883612.1883618.

[8] A. R. A. Alsewari and K. Z. Zamli, "Design and implementation of a harmony-search-based variable-strength t-way testing strategy with constraints support," *Information and Software Technology*, vol. 54, no. 6, pp. 553–568, Jun. 2012, doi: 10.1016/j.infsof.2012.01.002.

[9] B. S. Ahmed and K. Z. Zamli, "PSTG: a t-way strategy adopting particle swarm optimization," in *2010 Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, 2010, pp. 1–5, doi: 10.1109/AMS.2010.14.

[10] K. J. Nurmela, "Upper bounds for covering arrays by tabu search," *Discrete Applied Mathematics*, vol. 138, no. 1–2, pp. 143–152, Mar. 2004, doi: 10.1016/S0166-218X(03)00291-9.

[11] M. F. Zeng, S. Y. Chen, W. Q. Zhang, and C. H. Nie, "Generating covering arrays using ant colony optimization: exploration and mining," *Ruan Jian Xue Bao/Journal of Software*, vol. 27, no. 4, pp. 855–878, 2016, doi: 10.13328/j.cnki.jos.004974.

[12] J. Torres-Jimenez and I. Izquierdo-Marquez, "A simulated annealing algorithm to construct covering perfect hash families," *Mathematical Problems in Engineering*, vol. 2018, pp. 1–14, Jul. 2018, doi: 10.1155/2018/1860673.

[13] I. Izquierdo-Marquez, J. Torres-Jimenez, B. Acevedo-Juárez, and H. Avila-George, "A greedy-metaheuristic 3-stage approach to construct covering arrays," *Information Sciences*, vol. 460–461, pp. 172–189, Sep. 2018, doi: 10.1016/j.ins.2018.05.047.

[14] M. A. Basmassi, L. Benameur, and J. A. Chentoufi, "A novel greedy genetic algorithm to solve combinatorial optimization problem," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIV-4/W3-, pp. 117–120, Nov. 2020, doi: 10.5194/isprs-archives-XLIV-4-W3-2020-117-2020.

[15] R. C. Bryce and C. J. Colbourn, "The density algorithm for pairwise interaction testing," *Software Testing, Verification and Reliability*, vol. 17, no. 3, pp. 159–182, Sep. 2007, doi: 10.1002/stvr.365.

[16] J. Xin, J. Zhong, S. Li, J. Sheng, and Y. Cui, "Greedy mechanism based particle swarm optimization for path planning problem of an unmanned surface vehicle," *Sensors*, vol. 19, no. 21, p. 4620, Oct. 2019, doi: 10.3390/s19214620.

[17] F. I. Telchy and S. Rafaat, "Intelligent neural network with greedy alignment for job-shop scheduling," *Iraqi Journal of Computers, Communication, Control & Systems Engineering*, vol. 15, no. 3, 2015.

[18] D. M. Cohen, S. R. Dalal, M. L. Fredman, and G. C. Patton, "The AETG system: an approach to testing based on combinatorial design," *IEEE Transactions on Software Engineering*, vol. 23, no. 7, pp. 437–444, Jul. 1997, doi: 10.1109/32.605761.

[19] J. Czerwonka, "Pairwise testing in the real world: practical extensions to test-case scenarios," in *in Proceedings of 24th Pacific Northwest Software Quality Conference*, 2008, pp. 419–430.

[20] E. Lehmann and J. Wegener, "Test case design by means of the CTE XL," *Proceedings of the 8th European International Conference on Software Testing, Analysis & Review (EuroSTAR 2000), Kopenhagen, Denmark*. 2000.

[21] C. J. Colbourn, M. B. Cohen, and R. Turban, "A deterministic density algorithm for pairwise interaction coverage," in *Proceeding of the IASTED Conference on Software Engineering*, 2004, vol. 41, no. 10, pp. 242–252.

[22] Y. Lei, R. Kacker, D. R. Kuhn, V. Okun, and J. Lawrence, "IPOG: a general strategy for T-way software testing," in *14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'07)*, Mar. 2007, pp. 549–556, doi: 10.1109/ECBS.2007.47.

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

[24] M. Forbes, J. Lawrence, Y. Lei, R. N. Kacker, and D. R. Kuhn, "Refining the in-parameter-order strategy for constructing covering arrays," *Journal of Research of the National Institute of Standards and Technology*, vol. 113, no. 5, p. 287, Sep. 2008, doi: 10.6028/jres.113.022.

[25] M. I. Younis, K. Z. Zamli, and N. A. M. Isa, "A strategy for grid based t-way test data generation," in *2008 First International Conference on Distributed Framework and Applications*, Oct. 2008, pp. 73–78, doi: 10.1109/ICDFMA.2008.4784416.

[26] M. B. Cohen, C. J. Colbourn, and A. C. H. Ling, "Augmenting simulated annealing to build interaction test suites," in *14th International Symposium on Software Reliability Engineering, 2003. ISSRE 2003.*, 2003, pp. 394–405, doi: 10.1109/ISSRE.2003.1251061.

[27] A. R. A. Alsewari and K. Z. Zamli, "A harmony search based pairwise sampling strategy for combinatorial testing," *International Journal of the Physical Sciences*, vol. 7, no. 7, Feb. 2012, doi: 10.5897/ijps11.1633.

[28] M. B. Cohen, C. J. Colbourn, and A. C. H. Ling, "Constructing strength three covering arrays with augmented annealing," *Discrete Mathematics*, vol. 308, no. 13, pp. 2709–2722, Jul. 2008, doi: 10.1016/j.disc.2006.06.036.

[29] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, and C. J. Colbourn, "Constructing test suites for interaction testing," in *25th International Conference on Software Engineering, 2003. Proceedings.*, 2003, pp. 38–48, doi: 10.1109/ICSE.2003.1201186.

[30] M. B. Cohen, P. B. Gibbons, W. B. Mugridge, C. J. Colbourn, and J. S. Collofello, "A variable strength interaction testing of components," in *Proceedings 27th Annual International Computer Software and Applications Conference. COMPAC 2003*, 2003, pp. 413–418, doi: 10.1109/CMPSAC.2003.1245373.

[31] A. Calvagna and A. Gargantini, "T-wise combinatorial interaction test suites construction based on coverage inheritance," *Software Testing, Verification and Reliability*, vol. 22, no. 7, pp. 507–526, Nov. 2012, doi: 10.1002/stvr.466.

[32] A. K. Alazzawi, H. M. Rais, and S. Basri, "HABC: hybrid artificial bee colony for generating variable t-way test sets," *Journal of Engineering Science and Technology*, vol. 15, no. 2, pp. 746–767, 2020.

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

[34] A. K. Alazzawi *et al.*, "HABCSm: a hamming based t -way strategy based on hybrid artificial bee colony for variable strength test sets generation," *International Journal of Computer Communications & Control*, vol. 16, no. 5, Oct. 2021, doi: 10.15837/ijccc.2021.5.4308.

[35] K. M. Htay, R. R. Othman, A. Amir, and M. H. A. Jalal, "Gravitational search algorithm based strategy for combinatorial t-way test suite generation," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 8, pp. 4860–4873, Sep. 2022, doi: 10.1016/j.jksuci.2021.06.020.

[36] X. Guo, X. Song, and J. Zhou, "A synergic quantum particle swarm optimisation for constrained combinatorial test generation,"

IET Software, vol. 16, no. 3, pp. 279–300, Jun. 2022, doi: 10.1049/sfw2.12054.
[37] K. Z. Zamli, 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 Computer Systems & Software Engineering*, vol. 1, no. 1, pp. 15–27, Feb. 2015, doi: 10.15282/ijsecs.1.2015.2.0002.
[38] M. I. Younis, "MVSCA: multi-valued sequence covering array," *Journal of Engineering*, vol. 25, no. 11, pp. 82–91, Oct. 2019, doi: 10.31026/j.eng.2019.11.07.
[39] H. M. Fadhil, M. N. Abdullah, and M. I. Younis, "Combinatorial testing approaches: a systematic review," *Iraqi Journal of Computers, Communications, Control, and Systems Engineering (IJCCCE)*, vol. 24, 2023.
[40] I. A. AbdulJabbar and S. M. Abdullah, "Hybrid metaheuristic technique based tabu searchand simulated annealing," *Engineering and Technology Journal*, vol. 35, no. 2, pp. 154–160, 2017.
[41] H. S. Abdullah, "Comparative study of swarm intelligence behavior to solve optimization problems," *Engineering and Technology Journal*, vol. 29, no. 14, 2011.
[42] N. Ramli, R. R. Othman, Z. I. Abdul Khalib, and M. Jusoh, "A review on recent t-way combinatorial testing strategy," *MATEC Web of Conferences*, vol. 140, p. 1016, Dec. 2017, doi: 10.1051/matecconf/201714001016.
[43] J. Torres-Jimenez and I. Izquierdo-Marquez, "Improved covering arrays using covering perfect hash families with groups of restricted entries," *Applied Mathematics and Computation*, vol. 369, p. 124826, Mar. 2020, doi: 10.1016/j.amc.2019.124826.
[44] A. B. Nasser, K. Z. Zamli, A. A. Alsewari, and B. S. Ahmed, "Hybrid flower pollination algorithm strategies for t-way test suite generation," *PLOS ONE*, vol. 13, no. 5, p. e0195187, May 2018, doi: 10.1371/journal.pone.0195187.
[45] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian journal of statistics*, vol. 16, no. 2, 1979.

BIOGRAPHIES OF AUTHORS

Heba Mohammed Fadhil graduated from University of Baghdad with a Master of Computer Engineering in 2014. In 2006, she graduated from University of Al-Mustansiriyah with a Bachelor of Computer Engineering. Among her scientific interests include parallel processing, encryption, algorithms, object-oriented technology, artificial intelligence, image processing, cloud computing and the internet of things. She is a member of several committees, including ACM, the International Association for the Engineers, Cisco Networking Academy, and Oracle Academy, among others. At the moment, she works as an instructor for both Cisco and Oracle. In addition to being a member of the teaching staff at the Department of Information and Communication in the Al-Khwarizmi College of Engineering at the University of Baghdad. She is a reviewer for the International Journal of Computer Science, Peer-to-Peer Networking and Applications, BEEI, International Journal of New Computer Architectures and their Applications, Circulation in Computer Science Journal. She can be contacted at email: ce.19.15@grad.uotechnology.edu.iq or heba@kecbu.uobaghdad.edu.iq.

Dr. Mohammed Najm Abdullah is currently an assistant professor at the Department of Computer Engineering, University of Technology, Baghdad, Iraq. He received his B.Sc. degree in 1983 in Electrical Engineering from the College of Engineering, University of Baghdad. He received his M.Sc. degree in Electronics and Communication Engineering from the same college in 1989 and his Ph.D. degree in 2002 in Electronics and Communication Engineering/Airbrone Computer from the University of Technology. He published many research papers in national and international journals and conferences, as well as books. He can be contacted at email: mohammed.n.abdullah@uotechnology.edu.iq.

Dr. Mohammed Issam Younis obtained his Doctorate in Computer Engineering from Universiti Sains Malaysia in 2011. He had done the M.Sc. and B.Sc. in Computer Engineering from University of Baghdad in 2001 and 1997 respectively. His research interests are: distributed system, information security and cryptography, parallel processing, algorithms, computer networking, software engineering, RFID, and IoT. He has various publications as books, thesis, journals, Invited IEEE Tutorials. He is associated with various committee like: Iraqi Union of Engineers, Cisco Networking Academy, Software Engineering Research Groups, AIDL Research Groups. He honored by different awards, medals, patents, and grants. Prof. Dr. Younis is currently a faculty member and Cisco Instructor at the Computer Engineering Department, College of Engineering, University of Baghdad. He can be contacted at email: younismi@coeng.uobaghdad.edu.iq.