electronics

MDPI

Article

TWGH: A Tripartite Whale–Gray Wolf–Harmony Algorithm to Minimize Combinatorial Test Suite Problem

Heba Mohammed Fadhil [1,2,*], Mohammed Najm Abdullah [1] and Mohammed Issam Younis [3]

[1] Department of Computer Engineering, University of Technology, Al Wihda 10075, Iraq
[2] Department of Information and Communication, Al-Khwarizmi College of Engineering, University of Baghdad, Al Jadriya 10070, Iraq
[3] Department of Computer Engineering, College of Engineering, University of Baghdad, Al Jadriya 10070, Iraq
* Correspondence: ce.19.15@grad.uotechnology.edu.iq or heba@kecbu.uobaghdad.edu.iq

Abstract: Today's academics have a major hurdle in solving combinatorial problems in the actual world. It is nevertheless possible to use optimization techniques to find, design, and solve a genuine optimal solution to a particular problem, despite the limitations of the applied approach. A surge in interest in population-based optimization methodologies has spawned a plethora of new and improved approaches to a wide range of engineering problems. Optimizing test suites is a combinatorial testing challenge that has been demonstrated to be an extremely difficult combinatorial optimization limitation of the research. The authors have proposed an almost infallible method for selecting combinatorial test cases. It uses a hybrid whale–gray wolf optimization algorithm in conjunction with harmony search techniques. Test suite size was significantly reduced using the proposed approach, as shown by the analysis of the results. In order to assess the quality, speed, and scalability of TWGH, experiments were carried out on a set of well-known benchmarks. It was shown in tests that the proposed strategy has a good overall strong reputation test reduction size and could be used to improve performance. Compared with well-known optimization-based strategies, TWGH gives competitive results and supports high combinations ($2 \leq t \leq 12$).

Keywords: combinatorial testing; whale optimization algorithm; harmony search algorithm; grey wolf optimization; evolutionary algorithms; metaheuristics algorithms; test suite optimization; software testing

Citation: Fadhil, H.M.; Abdullah, M.N.; Younis, M.I. TWGH: A Tripartite Whale–Gray Wolf–Harmony Algorithm to Minimize Combinatorial Test Suite Problem. *Electronics* **2022**, *11*, 2885. https://doi.org/10.3390/electronics11182885

Academic Editors: Laith Alzubaidi, Jinglan Zhang, Ye Duan and Jose Santamaria

Received: 14 July 2022
Accepted: 3 September 2022
Published: 12 September 2022

1. Introduction

In order to avoid customer complaints, software testing ensures that products meet the user's needs. For example, failure to conduct enough testing at any stage of the software development lifecycle may result in the loss of critical data [1]. An optimal solution to a discrete, well-defined problem is sought using combinatorial methods. Combinatorial optimization problems are difficult to solve analytically because of their NP-completeness, and an accurate search may degenerate into complete enumeration. Given the constraints of time and resources, a thorough enumeration is practically impossible [2].

For interaction test generation (or t-way), the combinational optimization problem is addressed in this study (where t indicates the combination degree). In order to determine the most optimal test cases for the needed interaction in vast lists of values, numerous research methodologies (in the form of t-way strategies) have been created in recent years. As a result, there are numerous PC-based (pure computation) solutions. Current techniques have some drawbacks; however, they are prone to local minimums using simulated annealing and hill climbing strategies. Recent empirical findings in [3] suggest that higher combination degrees of $t > 6$ are appropriately required to assess highly interconnected and interacting computer programs. As a result, a combined degree is considered to be limited [4,5].

In such cases, conventional optimization methods cannot achieve the requirements for speed and accuracy. When compared to conventional approaches, metaheuristic algorithms are often capable of providing the best possible solution. In addition to being easy to implement, these algorithms are also capable of avoiding local optima. This set of algorithms was designed to tackle high-dimensional and complicated problems and produce high-quality results quickly. For example, they use concepts from the study of neurological systems, statistical mechanics, evolutionary biology, and intelligent problem solving to support their theories. Metaheuristic algorithms can be separated into groups based on the usual wonders they follow, such as evolutionary, physics-based, swarm-insight-related, and human-conduct-related. As a result of this, experts around the world are constantly looking for new ways to improve the existing methods [6].

Metaheuristics algorithms can be divided into two groups based on these criteria: population-based (e.g., swarm intelligence, evolutionary algorithms) and single-solution-based (e.g., local search and simulated annealing). It is important to strike a healthy balance between these two objectives. In order to attain this equilibrium, a hybrid model can be used, combining at least two techniques to improve each technique's effectiveness. It is the goal of this research to leverage the recently proposed whale optimization algorithm (WOA), gray wolf optimization (GWO), and harmony search algorithm (HSA) to create a novel hybrid approach to advance the performance of wide-ranging classification tasks [7]. In order to address the issues of weak diversity and premature convergence in the WOA, this study's primary contributions are as follows:

(1) A creative strategy to combine the three strategies in a hybrid approach;
(2) An HSA algorithm incorporated into WOA and GWO by utilizing a modified mechanism;
(3) Additionally, an asynchronous approach, employed to improve accuracy.

A few more sections will follow: Section 2 summarizes prior studies in this field. Section 3 concisely explains the concept of a covering array, while Section 4 summarizes the metaheuristic algorithms used. Section 5 describes the suggested methodology. Detailed descriptions of the experimental strategy and the data analysis may be found in Section 6. Section 7 focuses on statistical evaluation. Finally, Section 8 summarizes the findings and outlines the development's future path.

2. Related Work

Several well-known solutions use this approach, such as Test Vector Generator (TVG), Jenny, and Intelligent Test Case Handler (WHITCH) [4], to put that into perspective. The sequential creation of the test case takes place one parameter at a time when using the strategic approach, which uses horizontal extension and step-by-step construction. IPOG and MIPOG are two recent variation strategy implementations that use this approach to build a t-way test suite. The strategy performs a coverage check and finds the optimal value for each parameter component [3].

In recent years, many scholars have resorted to hybrid metaheuristics [8,9]. Plentiful practical and academic challenges can be tackled more efficiently by utilizing hybrid algorithms. In the subject of combinatorial testing, many effective hybrid metaheuristic algorithms have been proposed. Alazzawi et al. suggested a new algorithm that incorporates two cutting-edge methods. PhABC is a new paired test suite generation strategy that combines an artificial bee colony (ABC) method with the particle swarm optimization (PSO) algorithm. When building the final test suite, the PhABC strategy sometimes outperforms other approaches and produces results similar to the competition. Arram et al. [10] proposed five metaheuristics for a hybrid bird mating optimizer (BMO): hill climbing, late-acceptance hill climbing, simulating annealing, iterative greedy heuristics, and variable iterative greedy heuristics. By combining these methods, researchers could better explore the BMO population's search space and develop new solutions. Regression test case selection was developed by Agrawal et al. [11]. The authors utilized a hybrid whale optimization algorithm, which used bat search (BS)- and ant colony optimization (ACO)-based regression test case-selection procedures to extract subject programs from

the software artifact infrastructure repository to evaluate the algorithm. A study of the findings shows that the proposed strategy reduces the size of the test suite by a considerable margin of error. Alazzawi et al.'s [12] primary objective of this investigation was to propose an alternative method for reducing the number of test instances. An ABC and PSO method are combined to create a hybrid artificial bee colony (HABC) approach. High-interaction-strength combinatorial test suites up to t = 6 are needed. Experiments show that the HABC strategy outperforms other strategies when creating the best test case. Alazzawi et al.'s [13] study proposed a novel metaheuristic-based t-way strategy known as the hybrid artificial bee colony (HABCSm) strategy, which combines the advantages of the ABC algorithm with the advantages of the PSO. The t-way strategy HABCSm is the first algorithm to utilize Hamming distance to create a final test set and to use it as a final selection criterion for discovering innovative solutions. It is possible to compensate for the weaknesses of one algorithm with the strengths of others by combining two or more algorithms. Nasser et al. [14] suggested that hybrids of the flower pollination algorithm (FPA) combine with other algorithmic components to create four hybrid variations. FPA hybrids outperform existing t-way techniques in test suite size, as shown by the results of the experiments, by overcoming the sluggish convergence issues of the original FPA. Alsewari et al. [4] introduced a new combinatorial test list-generating technique based on the harmony search algorithm and its design and implementation. The general t-way harmony search-based strategy (GTHS) is a new approach to generating designed and implemented combinatorial test lists. HS was chosen as the principal engine for test generation because of its capacity to balance intensification and diversification. In [15], Alazzawi et al. presented a T-way-generating approach for uniform- and variable-strength test suites (ABCVS) illustrated by applying the ABC technique to lower the total size of a test suite while simultaneously increasing the interaction between tests in the suite.

3. Covering Array

T-way testing can be mathematically based on orthogonal array notation (OA). Each subarray of the orthogonal array contains all ordered subsets of size t from the level of interdependence that is represented by N, the number of generated test cases, the parameter count (k), the number of values (v), and the degree of interdependence (t). However, OA is typically viewed as restrictive because it requires all component values to be the same, which is a limitation. In order to address this issue, the covering Array (CA) notation was added. In the CA notation CA (N; t, k, v), each subarray contains all prearranged subsets of the same size from v levels. To denote the combination degree of the covering array and the number of parameters (or factors) and values (or levels), the symbols t, k, and v are used here. Once in the CA is usually adequate to cover all t-interactions between the components. In this case, the notation is CA (N; 2, 4, 3), which is an example of a system with two-way interaction with four three-valued parameters. The mixed covering array (MCA) can handle situations where one or more component values (v) change. In order to make matters more confusing, N and t have the same meaning as in CA and MCA but with the addition of a new symbol, C, which represents the parameters and values of each configuration in the following format: ($v1^{k1}$, $v2^{k2}$,... vn^{kn}) where there are k1 parameters with v1 values and k2 parameters with v2 values. Consider MCA (9, 2, 36, 24) as an example. The interaction strength degree is 2, and the number of parameters is six, linked with three values each, and four are linked with two values each [4,16,17].

4. Metaheuristic Algorithms

Metaheuristic and evolutionary optimization algorithms are unusual in that they are widely considered useful in tackling NP-hard problems, as they can find near-optimal solutions to the given optimization issues in an acceptable period. Many metaheuristic techniques for solving combinatorial problems, particularly those using covering arrays, have been developed in the literature. Optimizing a fitness function is the goal of these opti-

mization methods, which are used to find the best possible solution to a given problem [18]. The proposed system's metaheuristic algorithms are detailed in this section.

4.1. Whale Optimization Algorithm

In terms of metaheuristic optimization algorithms, the whale optimization algorithm (shown in Algorithm 1) is one of the most recent. It is based on swarm intelligence. It closely mimics the natural behavior of humpback whales when hunting using bubble nets. This approach has been used successfully in many fields to deal with complex optimization problems. However, it performs poorly in large-scale settings because it requires substantial computational work. When dealing with large-scale problems, distributed computing is an excellent way to increase WOA's scalability [19,20].

Algorithm 1 Whale Optimization Algorithm

Input population size N, halt criteria
Output the best solution X^*
Generate initial population Xi (i = 1, 2,...,N)
Calculate the fitness for each solution in the population
Find the best solution X^* with the best fitness
while halt criteria not satisfied **do**
Update a
for i=1 to N **do**
Update WOA parameters (A, C, l, and p)
for j=1 to m **do**
if p < 0.5 **then**
if $|A| < 1$ **then**
Update the current search agent by
$\vec{D} = \left|\vec{C}\vec{X}^*(t) - \vec{X}(t)\right|$
$\vec{X}^*(t+1) = \vec{X}(t) - \vec{A}\vec{D}$
else if $|A| \geq 1$ **then**
Select a random search agent (Xrnd)
Update the current search agent by
$\vec{D} = \left|\vec{C}\vec{X}^*(t) - \vec{X}(t)\right|$
$\vec{X}^*(t+1) = \vec{X}(t) - \vec{A}\vec{D}$
end if
else if $p > 0.5$ **then**
Update the current search agent by
$\vec{X}(i+1) = e^{bk}\cos(2\pi k)\vec{D}^* + \vec{X}^*(i)$
$\vec{D}^* = \left|\vec{X}^*(i) - \vec{X}(i)\right|$
end if
end for
Evaluate the search agent X^*
end for
Find the best solution X^*
end while

4.2. Gray Wolf Optimizer

As a population-based metaheuristic algorithm, GWO is a novel addition to the family of algorithms; like all other metaheuristic algorithms, it begins with evenly distributed random positions of the gray wolves. At the end of each cycle, the positions are recalculated. Wolf packs refer to a large group of wolves living nearby. In order to hunt, the gray wolf group has a social hierarchy in which the leader of the pack is the most powerful. Alpha, Beta, and Delta wolves are members of the gray wolf pack's leadership hierarchy. The three most powerful wolves in the pack are responsible for making important decisions

and updating the wolves' position while hunting. In gray wolf society, the hierarchy is carefully maintained, and decisions are made by the best-hunting agents, Alpha, Beta, and Delta. In every iteration, the current best agents are re-evaluated, and their positions are revised, as shown in Algorithm 2. Omega wolves are the rest of the wolves that follow the leaders' orders. Using information from the Alpha, Beta, and Delta wolves, Omega wolves constantly recalculate their positions and values [21,22].

Algorithm 2 Grey Wolf Optimization Algorithm

Input population size of wolves' pop, MaxIter
Output optimal grey wolf position X_α
Initialize the grey wolf population X_i Randomly
initialize a, A and C
Determine the fitness of each wolf X_i
X_α = the best solution
X_β = the second best solution
X_δ = the third best solution
while i $\leq$ MaxIter **do**
for each wolf X_i **do**
update the position

$$\vec{D_\alpha} = \left|\vec{C_1} \cdot \vec{X_\alpha}\right|$$
$$\vec{D_\beta} = \left|\vec{C_2} \cdot \vec{X_\beta}\right|$$
$$\vec{D_\delta} = \left|\vec{C_3} \cdot \vec{X_\delta}\right|$$
$$\vec{X_1} = \vec{X_\alpha} - \vec{A_1}.(\vec{D_\alpha})$$
$$\vec{X_2} = \vec{X_\beta} - \vec{A_2}.(\vec{D_\beta})$$
$$\vec{X_3} = \vec{X_\delta} - \vec{A_3}.(\vec{D_\delta})$$
$$\vec{X}\ (t+1) = \left(\vec{X_1} + \vec{X_2} + \vec{X_3}\right)/3$$

end for
update a, A and C
determine the fitness of each wolf X_i
update X_α, X_β and X_δ
$i = i + 1$
end while
return $X\alpha$

4.3. Harmony Search Algorithm

The harmony search Algorithm is an evolutionary algorithm that mimics the process of musical improvisation in order to achieve a beautiful harmony. Optimization issues in a variety of fields were effectively addressed using this approach. As a result, the algorithm has a serious flaw in solving difficult tasks. Pitch modification and random consideration are part of the algorithm's improvisational functionality. There are three ways in which a musician can improve their pitch:

(1) An old pitch from the musician's recollection is played back;
(2) The adjacent stored pitch is played;
(3) Any pitch that falls within the normal range is played.

There are three steps to selecting a value for each variable in the method:

(1) A memory value is chosen from among those already stored there;
(2) Some adjacent values to the recorded data are chosen;
(3) An arbitrary number that falls within the standard range is chosen.

Both of the HSA's primary pillars are responsible for enforcing the activities outlined above. These are known as the harmony memory considering rate, or HMCR, as well as the pitch adjusting rate (PAR) [23,24].

In the same way, as artists iteratively enhance their music (see Algorithm 3), the harmony search algorithm (HSA) imitates this process. To obtain the best possible harmony, each musician takes turns playing an instrument during this procedure. In order to discover a global answer, the algorithm generates a value for each choice variable that represents a musician [25].

Algorithm 3 Harmony Search Algorithm

```
Input Generate the initial harmonics randomly
Output Optimal solution with its fitness value
Initialize the parameters of the HS HMCR, PAR and etc
Initialize harmony memory (HM)
Evaluate all solutions using the fitness function
  while Termination criteria do
    new solution = 0
    if rand HMCR then
      Memory consideration
      if rand < PAR then
        Pitch adjustment
      end if
    else
      Random consideration
    end if
    Evaluate the fitness function of the new solution
    Replaces the worst solution in HM by the new solution
  end while
```

5. Proposed Work

Failures caused by system interactions can be found via combinatorial testing, which is a powerful technique. Swarm intelligence and population-based algorithms have been successfully applied to the resolution of a wide variety of optimization issues. There are some challenging issues for which the conventional methods cannot identify the most appropriate solution at a given time. Metaheuristic search algorithms provide the best answer in existing t-way procedures regarding test suite size. The most current examples of metaheuristic optimization algorithms are known as WOA and GWO. These algorithms are population-based and based on swarm intelligence, and they are what are known as swarm-based. The whale and gray wolf optimization algorithms, considering innovative approaches to the optimization problem, have been combined by us in the construction of the hybrid scheme.

A TWGH algorithm that generates a set of optimal initial solutions in TWGH using the one-test-at-a-time technique is proposed as a new and extensively used optimization technique in portfolio design. To speed up the convergence, the new algorithm constantly adjusts the values of HMCR and PAR to prevent the program from sliding into local optima by using the HAS algorithm. In addition, the performance of WOA and GWO algorithms in exploration and exploitation has been the focus of this research. Two variations, either heterogeneous or homogeneous, can be hybridized using low-level or high-level strategies, such as relay or coevolution, respectively, as stated by [1]. TWGH uses a low-level mixed hybrid to combine the WOA and GWO algorithms. The hybrid is low since it is a coevolutionary process that does not use each variant one after the other. What this means is that they operate simultaneously or in parallel. The benchmark test and real-world problems are optimally solved using a hybrid method that combines two different methodologies. This update aims to strengthen the capability of exploitation in the GWO by combining it with the capability of exploration in the WOA. This enables us to demonstrate the benefits

of both techniques. At first, TWGH begins with establishing a baseline population size for the search agents (which include both whales and wolves). The population will go through a process to adjust the agents if they travel beyond the search space. Thus, it is necessary to compute the fitness function. If the fitness score is lower than the Alpha score, also known as the Best Score, then the Alpha score is the same as the fitness score. After then, the following variables are given fresh values: a, A, C, L, and p. Secondly, it checks if the random number is smaller than 0.5, then it moves on to check if (A < 1) condition. In this case, the new coordinates can be determined with the help of the first equation in the algorithm (1). When comparing the two, if the new one is superior, the previous one is changed to reflect that.

To begin, we modified WOA's hunting process by inserting a new condition into the exploitation stage. In order to prevent WOA from reaching local optima when p more than or equal to 1, a new condition has been added to the usual exploitation phase. The change is made to Equations (1) and (2), then they are employed within the new $\vec{D_\alpha}, \vec{D_\beta}$ and $\vec{D_\delta}$ exploitation phase condition. Finally, a new condition is introduced during exploration to push the present answer closer to the optimal one. Furthermore, this prevents the whale from moving to a worse location than its current one.

After all that has been done, the new position needs to be checked to see if it lies outside the search space. Therefore, if they lie outside of the region where human activity is possible, as a means of hastening convergence, the new algorithm dynamically modifies HMCR and PAR values to forestall the program from settling into a local optimum. When constraints are known, the HAS algorithm can adjust the position accordingly. A revised fitness value is computed, and the highest-scoring value is ultimately returned.

In order to arrive at the best possible solution, the hybrid TWGH combines the most advantageous characteristics of the GWO during the exploitation phase with those of the WOA during the exploration phase. The position of the gray wolf, who is responsible for discovering the global optimum solution to the problem, is swapped out for the position of the whale, which is equivalent to that of the gray wolf but is much more effective at nudging the solution in the direction of the optimum. The computational time is decreased, while the whale optimizer algorithm guides an ideal value. The gray wolf optimizer is well-known as a method for extracting the optimal solution from a black box of possibilities. Therefore, combining the greatest features promises to find the best feasible global optimal solution for real-world and standard issues while avoiding local stagnation and optima. Utilizing the condition during exploration boosts search efficiency by enhancing the quality of the answer, assuming one exists.

The following outlines the mathematical model for TWGH: Improving the method's convergence performance is the goal of the TWGH variation, which updates the position of Alpha, Beta, and Delta using the spiral updating equation of the whale optimizer algorithm. The GWO method and the WOA algorithm share the remaining operations performed. With this in mind, we derive the following equations for updating the spiral and hunting position.

$$ps = e^{bk} cos(2\pi k)\vec{D^l} + \vec{X^*}(i) \tag{1}$$

$$\begin{aligned} \vec{D_\alpha} &= \left| \vec{C_1} \cdot \vec{X_\alpha} - ps \right| \\ \vec{D_\beta} &= \left| \vec{C_2} \cdot \vec{X_\beta} - ps \right| \\ \vec{D_\delta} &= \left| \vec{C_3} \cdot \vec{X_\delta} - ps \right| \end{aligned} \tag{2}$$

where $\vec{D^l}\left|\vec{X^*} - \vec{X}\right|$ shows how far away a whale is from its meal, and b is a constant used to describe the logarithmic spiral's overall form. $\vec{D_\alpha}, \vec{D_\beta}$ and $\vec{D_\delta}$ are the positions of the top three search agents, the mean, and a random integer l between $[-1, 1]$. Algorithm 4

presents the pseudocode form, and Figure 1 is a flowchart detailing the TWGH algorithm's implementation.

Algorithm 4 Pseudo Code of TWGH

Initialize the population for GWO and WAO
Calculate the fitness of each search agent
X* is the best search member
While (it < MaxIter)
For every search agent **do**
Calculate a, A, C, L, and p.
if (p < 0.5) then
if (A < 1)then
update the position of the current search member by

$$\vec{D} = \left| \vec{C}\vec{X^*}(t) - \vec{X}(t) \right|$$

else if (A $\geq$ 1) then
Select a random search agent (X_{rnd}).
Update the current search agent by

$$\vec{X^*}(t+1) = \vec{X}(t) - \vec{A}\vec{D}$$

end if
else if (p $\geq$ 0.5) then
update the position of the present search member by using Equations (1) and (2).
end if
end for
Find the fitness of all search members
If (Objective function of current agent > Objective function of previous position) then
Initialize the parameters of the HS HMCR, PAR and etc
Initialize harmony memory (HM)
Evaluate all solutions using the fitness function
while Termination criteria **do**
new solution = 0
if rand HMCR **then**
Memory consideration
if rand < PAR **then**
Pitch adjustment
end if
else
Random consideration
end if
Evaluate the fitness function of the new solution
Replaces the worst solution in HM by the new solution
end while
return to while loop
else
Update $\vec{X^*}$, $\vec{D_\alpha}$, $\vec{D_\beta}$ and $\vec{D_\delta}$
it = it+1
end while
return $\vec{X^*}$

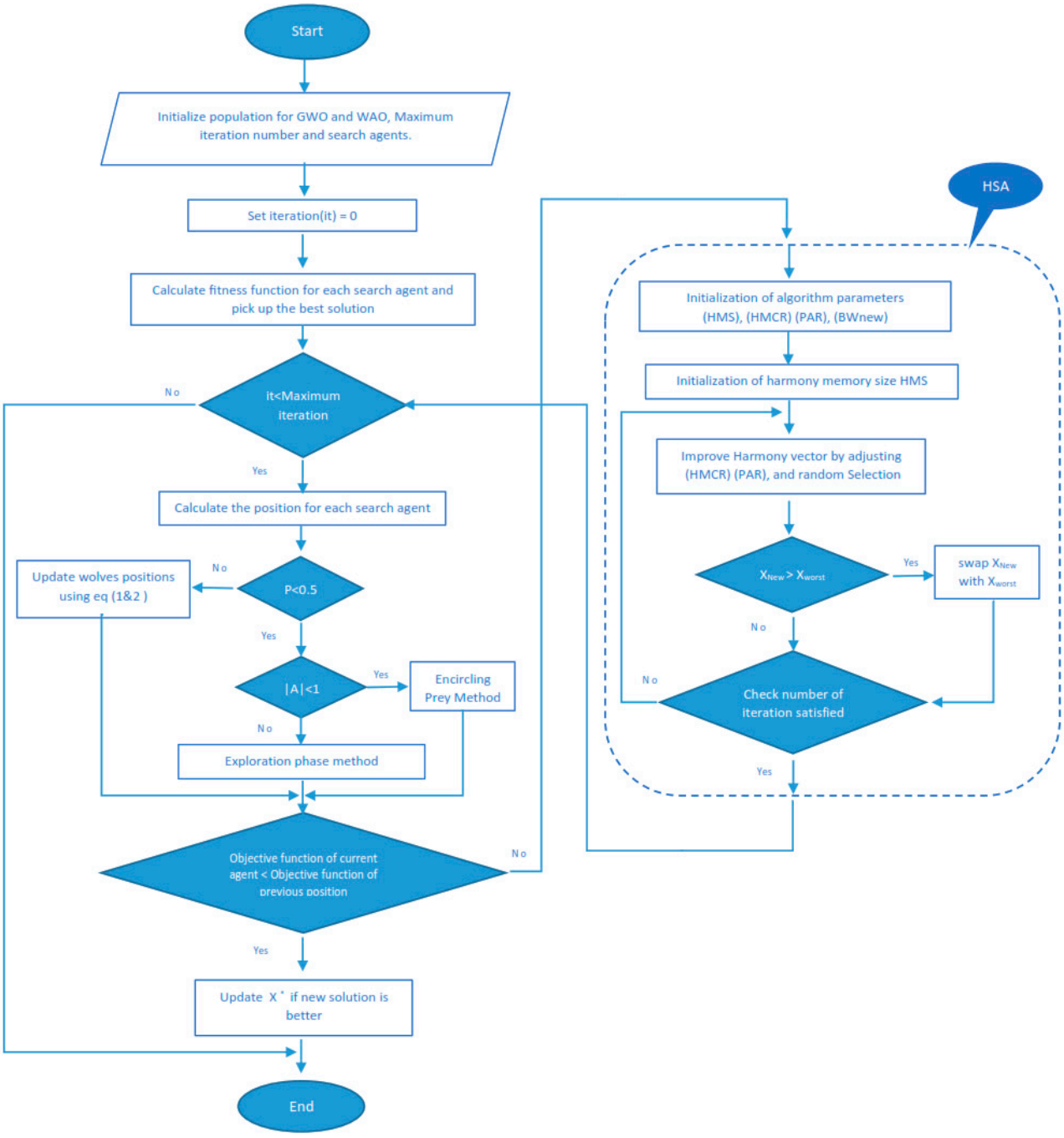

Figure 1. The proposed TWGH.

6. Results and Discussion

This hybrid heuristic algorithm was developed over two decades of research in the development of combinatorial search algorithms. During this time, researchers tested the algorithm not just against the most recent CIT approaches but also against clearly stated results. As a way to evaluate TWGH to other strategies, TWGH is compared to other available techniques in the manner of comparing the size of the TWGH-produced test suite to that of other well-known benchmark configurations. Thus, TWGH results can be directly compared to published results for techniques in [4,12,13,15,26]. A random selection of parameter values may have varied test suite sizes on each experiment.

Researchers calculated each segment's average and ideal test suite size by repeating the TWGH run 30 times for comparative purposes. The computer had an Intel(R) Core

(TM) i7-10750H CPU running at 2.60 GHz, 2.59 GHz, and 24 GB of RAM. The operating system used was Windows 11 Home. TWGH was implemented in PYTHON3. Tables are used to present the collected data (see Tables 1–6). In addition, Figures 2–6 represent results for CA (N; t, v^7) with $2 \leq t \leq 6$ and $2 \leq v \leq 5$.

Table 1. CA (N; t,10,2) with t between 2 and 10.

t	Jenny	TConfig	WHITCH	IPOG	MIPOG	TVG	GTHS	ABCVS	HABC	HABCm	HGHC	Proposed TWGH
2	10	9	6	10	NA	10	7	NA	NA	NA	7	6
3	18	20	18	19	NA	17	16	NA	NA	NA	17	16
4	39	45	58	49	NA	41	37	NA	NA	NA	38	36
5	87	95	NS	128	NA	84	81	NA	NA	NA	80	80
6	169	183	NS	352	NA	168	158	NA	NA	NA	175	155
7	311	NS	NS	NS	NA	302	298	NA	NA	NA	300	300
8	521	NS	NS	NS	NA	514	498	NA	NA	NA	505	500
9	788	NS	NS	NS	NA	651	512	NA	NA	NA	NA	510
10	1024	NS	NS	NS	NA	NS	1024	NA	NA	NA	NA	1022

Table 2. CA (N; t, 10, 5) with t between 2 and 10.

t	Jenny	TConfig	WHITCH	IPOG	MIPOG	TVG	GTHS	ABCVS	HABC	HABCm	HGHC	Proposed TWGH
2	45	48	45	50	45	50	43	NA	NA	NA	43	45
3	225	312	225	313	281	342	276	NA	NA	NA	236	225
4	1719	1878	1750	1965	1643	1971	1624	NA	NA	NA	1770	1625
5	9437	NA	NS	11,009	8169	NA	8866	NA	NA	NA	9933	8199
6	NA	NA	NS	57,290	45,168	NA	51,001	NA	NA	NA	45,339	45,244
7	NA	NS	NS	NS	NA	NA	225,924	NA	NA	NA	299,336	225,578
8	NA	NS	NS	NS	NA	NA	990,966	NA	NA	NA	994,339	990,023
9	NA	NS	NS	NS	NA	NA	2,971,150	NA	NA	NA	NA	2,971,022
10	NA	NS	NS	NS	NA	NS	9,765,624	NA	NA	NA	NA	9,765,145

Table 3. CA (N; t, 7, 3) with t between 2 and 7.

t	Jenny	TConfig	WHITCH	IPOG	MIPOG	TVG	GTHS	ABCVS	HABC	HABCm	HGHC	Proposed TWGH
2	16	15	15	17	NA	16	14	15	15	14	14	14
3	51	55	55	57	NA	54	50	49	47	46	50	48
4	169	166	216	185	NA	167	157	157	155	149	150	147
5	458	477	NS	561	NA	463	437	442	438	437	440	435
6	1089	921	NS	1281	NA	1049	916	944	836	729	778	725
7	2187	NA	NS	NS	NA	NS	2187	NA	NA	NA	2202	2184

Table 4. CA (N; 4, k, 5) with k varied from 5 to 12.

k	Jenny	TConfig	WHITCH	IPOG	MIPOG	TVG	GTHS	ABCVS	HABC	HABCm	HGHC	Proposed TWGH
5	837	773	625	908	625	849	751	NA	759	750	730	625
6	1074	1092	625	1239	625	1128	990	NA	1000	996	956	625
7	1248	1320	1750	1349	1125	1384	1186	NA	1189	1179	1230	1132
8	1424	1532	1750	1792	1384	1595	1358	NA	1386	1354	1395	1351
9	1578	1724	1750	1793	1543	1795	1530	NA	1591	1526	1530	1523
10	1791	1878	1750	1965	1643	1971	1624	NA	1798	1718	1697	1624
11	1839	2038	1750	2091	1722	2122	1860	NA	NA	NA	1730	1734
12	1964	NA	1750	2285	1837	2268	2022	NA	NA	NA	1834	1755

Table 5. CA (N; 4, 10, v) with v varied from 2 to 7.

v	Jenny	TConfig	WHITCH	IPOG	MIPOG	TVG	GTHS	ABCVS	HABS	HABSm	HGHC	Proposed TWGH
2	39	45	58	49	43	40	39	NA	NA	NA	43	38
3	221	235	336	241	217	228	211	NA	NA	NA	218	211
4	703	718	704	707	637	782	691	NA	NA	NA	664	639
5	1719	1878	1750	1965	1643	1917	1624	NA	NA	NA	1625	1622
6	3519	NA	NA	3935	3657	4159	3475	NA	NA	NA	3464	3464
7	6462	NA	NA	7061	5927	7854	6399	NA	NA	NA	6125	5929

Table 6. Five system configurations with mixed variables.

Configurations	Jenny	TConfig	WHITCH	IPOG	MIPOG	TVG	GTHS	HGHC	Proposed TWGH
MCA (N; 4, 3^4 4^5)	457	499	704	463	NA	487	436	445	438
MCA (N; 4, 5^1 3^8 2^2)	303	302	1683	324	NA	313	286	300	285
MCA (N; 4, 8^2 7^2 6^2 5^2)	4580	4317	4085	4776	NA	5124	4395	4090	4085
MCA (N; 4, 6^5 5^4 3^2)	3033	NA	NA	3273	NA	2881	2520	2656	2520
MCA (N; 4, 10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1)	6138	5495	5922	5492	NA	6698	5915	5955	5485

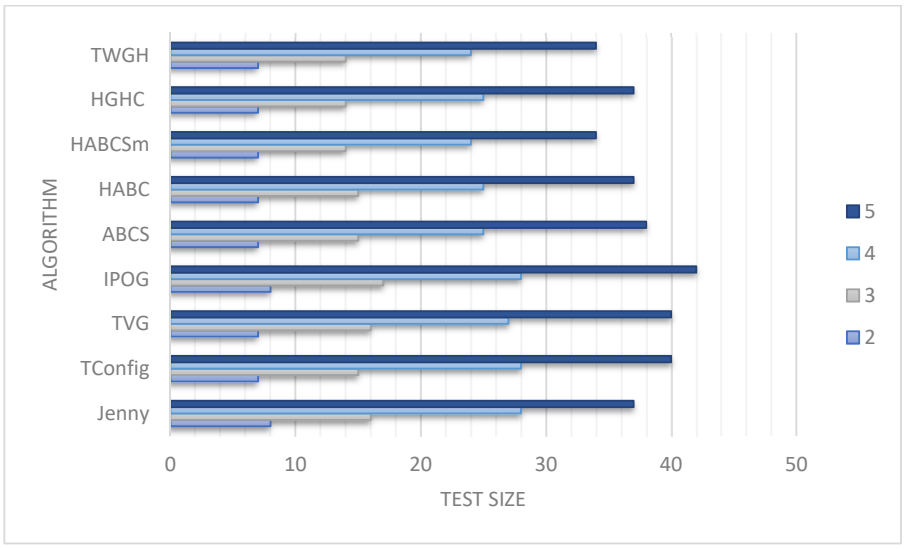

Figure 2. Test set size for CA (N; t, v^7) with t = 2 and $2 \leq v \leq 5$.

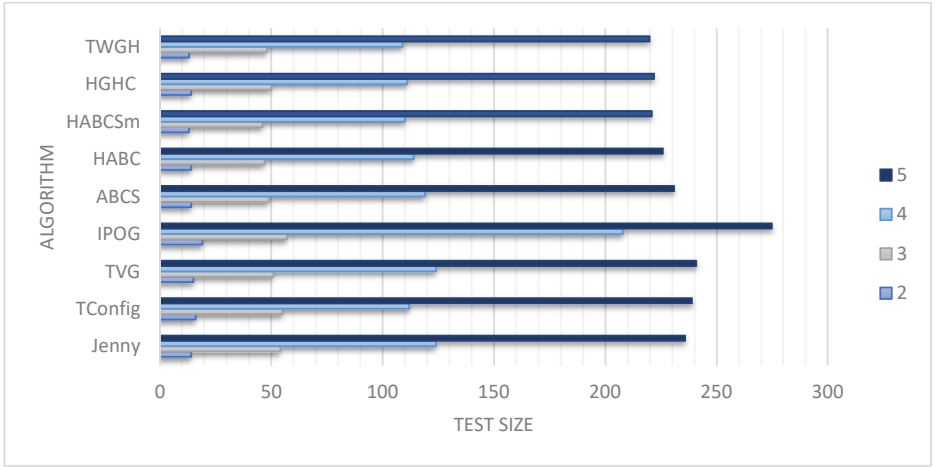

Figure 3. Test set size for CA (N; t, v^7) with t = 3 and $2 \leq v \leq 5$.

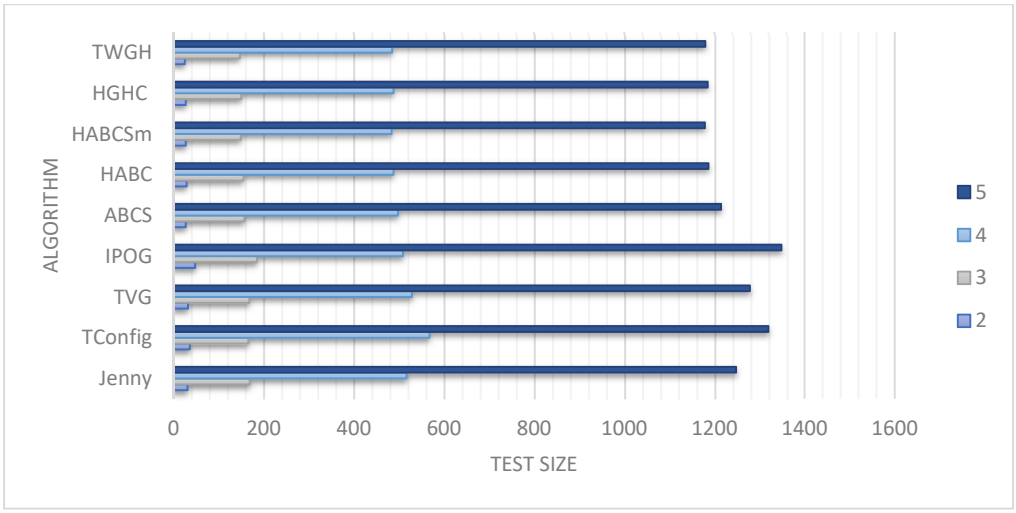

Figure 4. Test set size for CA (N; t, v^7) with t = 4 and $2 \leq v \leq 5$.

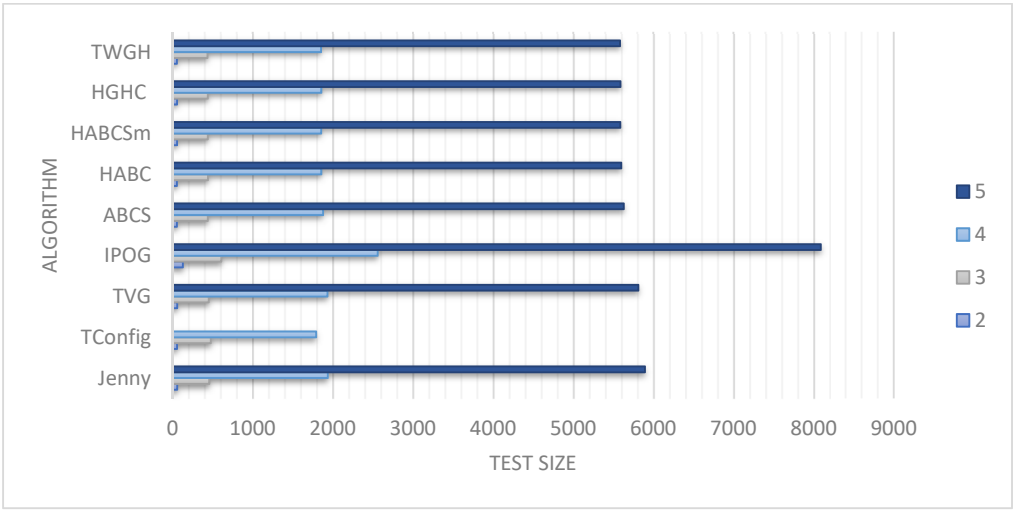

Figure 5. Test set size for CA (N; t, v^7) with t = 5 and $2 \leq v \leq 5$.

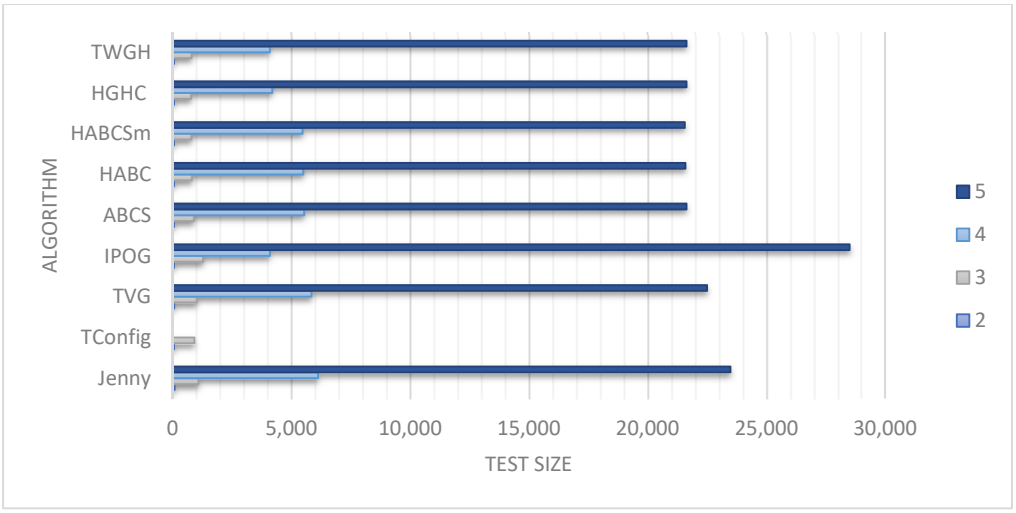

Figure 6. Test set size for CA (N; t, v^7) with t = 6 and $2 \leq v \leq 5$.

The darker cells epitomize the unsurpassed feasible configuration of interests, as denoted by the shaded sections of the cells inside. Noncompliant combinations are denoted

by cells labeled "NS" (not supported). No results were made public for any cells with the designation of NA, which stands for "not available.".

Table 1 demonstrates that the TWGH recommendation for the ideal test suite size applies to the majority of t values, except t = 7. The better test size is produced by GTHS whenever t equals 7 or 8. No recorded findings have been found for MIPOG, ABCVS, HABC, or HABCm. As shown in Table 2, TWGH appears to produce the optimum sample size for the test suite in most cases. MIPOG and GTHS contribute to the generation of the second-best size for the test suite. However, even though they are not the best possible outcomes, TWGH's results in these scenarios are still competitive. Since none of the other approaches, such as ABCVS, HABC, or HABCm, have reported or supported it, this is the conclusion that must be drawn. The results of Table 3 demonstrate that TWGH provides the smallest possible test suite consistently. Only HABCm has surpassed TWGH in the scenario where t equals three. This is the case at time t equal to 2; this indicates that GTHS, HABCm, HGHC, and TWGH all produce the same number of tests, which is the optimal number. MIPOG has not produced any verified results. According to the findings, Table 4 demonstrates that TWGH is superior to all other methods for the values of k = 5,6,8,9, and 10. As for the rest of the scenarios, MIPOG and WHITCH have the upper hand (i.e., k = 7 and 12, respectively). In Table 5, it would appear that TWGH, MIPOG, and WHITCH are performing the best overall. When k is equal to 10, only then does GTHS achieve the same sample size as TWGH. TWGH produces the optimal sample size whenever v values are set to 2, 3, 5, or 6. According to MIPOG, an ideal test suite would include these many components, except v = 4 or 7. The TWGH test suite is the smallest of all the other test suites. On the other hand, the test size produced by GTHS and HGHC is the same as that produced by TWGH when v equals 2 and 6, respectively. According to Table 6, TWGH generates the shortest test suite size in all four possible configurations, which are as follows: MCA (N; 4, 5^1 3^8 2^2), MCA (N; 4, 8^2 7^2 6^2 5^2), MCA (N; 4, 6^5 5^4 3^2), and MCA (N; 4, 10^1 9^1 8^1 7^1 6^1 5^1 4^1 3^1 2^1). GTHS helps keep the number of tests in the MCA (N; 4, 3^4 4^5) test suite to a minimum. MIPOG, ABCVS, HABC, and HABCm yielded no recorded results for the MCA's ideal test suite size.

7. Statistical Evaluation

Statistical analysis can also be used to evaluate the effectiveness of the proposed plan and identify its relevance. The Wilcoxon signed-rank test is applied, with a level of confidence equal to 95 percent (that is, a value of 0.05), to analyze the TWGH strategy about other existing techniques (from Tables 3, 4 and 6). The Wilcoxon signed-rank test will be used to see if the proposed strategy differs statistically from the other options under consideration. Because it compares the two sets side by side, this test is great for determining the discrepancy between them and using the Bonferroni–Holm technique to adjust value when multiple comparisons are involved. It uses asymptotic significance (two-tailed) to scale the data. Therefore, Holm is recalculated based on the following factors in Equation (3) [27]:

$$\propto Holm = \frac{\propto}{M - i + 1} \tag{3}$$

The total number of paired comparisons is $\boldsymbol{M}$, and the number of tests is $\boldsymbol{i}$. When evaluating TWGH, its three ranks—TWGH >, TWGH, and TWGH =—are used. The results of other strategies are higher, smaller, or the same as those of the proposed plan based on the first p-value that was recorded (asymp. Sig. (2-tailed)) in the sequence of scaling. If the value surpasses Holm, there is a significant difference between the two sets, as shown by the asymp. sig. (2-tailed) statistic. This study does not address the Z value (i.e., not considered). The hypothesis is rejected if the asymp. sig. (2-tailed) value is less than Holm. If one null hypothesis cannot be ruled out, the other hypotheses are also preserved. N/A results are regarded as incomplete and ignored samples since there is no test configuration for which a result has been provided. Tables 7–9 show the statistical results of the Wilcoxon test for Tables 3–6, which may be seen below. Table 7 shows that

TWGH performed BETTER than IPOG, TVG, or WHITCH, while HABCm did better. On the contrary, as shown in Table 8, TWGH is superior to all other methods except MIPOG and WHITCH. While TWGH outperformed all other states in Table 9, it fell short of GTHS. Since the results of the other methodologies are unavailable or do not support a certain setup, they are labeled "missing".

Table 7. Analysis of data from Table 3 using the Wilcoxon signed-rank sum test.

Pairs	Ranks			Test Statistics			Conclusion
	TWGH <	TWGH >	TWGH =	Z	Asymp. Sig. (2- tailed)	α Holm	
TWGH-GTHS	15	11	11	2.02260	0.0591	0.0025	Retain the null hypothesis
TWGH-HGHC	6	3	0	2.0226	0.0101	0.0125	Retain the null hypothesis
TWGH-IPOG	3	0	0	2.2014	0.0361	0.0171	Reject the null hypothesis
TWGH-Jenny	7	4	4	2.2014	0.0546	0.0173	Reject the null hypothesis
TWGH-HABCm	6	3	3	1.5724	0.0141	0.0014	Retain the null hypothesis
TWGH-TVG	4	0	1	2.2014	0.0363	0.2301	Reject the null hypothesis
TWGH-WHITCH	7	4	4	2.2014	0.0546	0.0163	Reject the null hypothesis

Table 8. Analysis of data from Table 4 using the Wilcoxon signed-rank sum test.

Pairs	Ranks			Test Statistics			Conclusion
	TWGH <	TWGH >	TWGH =	Z	Asymp. Sig. (2- tailed)	α Holm	
TWGH-GTHS	3	2	2	1.0954	0.0363	0.0641	Retain the null hypothesis
TWGH-HGHC	11	7	7	1.0226	0.0591	0.0573	Reject the null hypothesis
TWGH-IPOG	5	3	3	2.0226	0.0691	0.0472	Reject the null hypothesis
TWGH-Jenny	5	1	1	2.0432	0.0786	0.0435	Reject the null hypothesis
TWGH-TVG	6	3	0	2.678	0.0978	0.0349	Reject the null hypothesis
TWGH-WHITCH	4	2	1	2.9226	0.0991	0.0322	Reject the null hypothesis

Table 9. Analysis of data from Table 6 using the Wilcoxon signed-rank sum test.

Pairs	Ranks			Test Statistics			Conclusion
	TWGH <	TWGH >	TWGH =	Z	Asymp. Sig. (2- tailed)	α Holm	
TWGH-GTHS	13	11	11	2.3664	0.0220	0.0213	Reject the null hypothesis
TWGH-HGHC	3	1	1	2.3805	0.0209	0.017	Reject the null hypothesis
TWGH-IPOG	15	11	11	2.5205	0.0143	0.0127	Reject the null hypothesis
TWGH-Jenny	6	0	2	2.5205	0.0183	0.0142	Reject the null hypothesis
TWGH-MIPOG	7	3	3	1.5724	0.1422	0.0085	Retain the null hypothesis
TWGH-TVG	5	0	2	2.5205	0.0143	0.0107	Reject the null hypothesis
TWGH-WHITCH	3	1	1	1.9917	0.0592	0.00125	Retain the null hypothesis

8. Conclusions

Based on the results of comparative investigations, the suggested strategy beats current techniques in terms of CA/MCA generation quality and the number of generations required to get there. When comparing the size of CA and MCA, the newly developed strategy performs far better than the conventional approaches in the vast majority of instances. Hybrid approaches are essential for achieving the greatest results, and utilizing a cutting-edge hybrid whale–gray wolf–harmony metaheuristic approach, encompassing orders of coverage arrays $2 \leq v \leq 7, 2 \leq k \leq 12$ and strengths $2 \leq t \leq 12$ were produced. This is a highly competitive technology for the manufacturing of such covering arrays, as seen by this success. It was presented as an example of how uniform cover arrays of degree 4 can be used to evaluate compost composition.

To assess the effect on the ultimate array size of each decision, TWGH experiments were meticulously planned and carried out. It is possible to conclude that (1) the configuration of a covering array significantly impacts its performance based on data, and in some cases, the optimal configuration is even better than proven approaches such as IPOG, MIPOG, and Jenny compared to the established methods; (2) consequently, the suggested TWGH algorithm may be more efficient for producing test data because it assures enough coverage, optimality, and minimal complexity. In the future, we will study whether the metaheuristic technique can generate CAs and MCAs with higher strength and consider the seed scenarios and constraints in manufacturing a covering array.

Author Contributions: Conceptualization, H.M.F., M.N.A. and M.I.Y.; methodology, H.M.F.; software, H.M.F.; validation, H.M.F., M.N.A. and M.I.Y.; formal analysis, H.M.F.; investigation, H.M.F.; resources, H.M.F.; data curation, H.M.F.; writing—original draft preparation, H.M.F.; writing—review and editing, H.M.F., M.N.A. and M.I.Y.; visualization, H.M.F.; supervision, M.N.A. and M.I.Y.; project administration, M.N.A. and M.I.Y.; funding acquisition, H.M.F. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alsewari, A.A.; Mu'aza, A.A.; Rassem, T.H.; Tairan, N.M.; Shah, H.; Zamli, K.Z. One-Parameter-at-a-Time Combinatorial Testing Strategy Based on Harmony Search Algorithm OPAT-HS. *Adv. Sci. Lett.* **2018**, *24*, 7273–7277. [CrossRef]
2. Fadhil, H.M.; Abdullah, M.N.; Younis, M.I. Combinatorial Testing Approaches: A Systematic Review. *IRAQI J. Comput. Commun. Control. Syst. Eng. (IJCCCE)* **2023**, *24*. *accepted*.
3. Younis, M.I.; Zamli, K.Z. MIPOG—An efficient t-way minimization strategy for combinatorial testing. *Int. J. Comput. Theory Eng.* **2011**, *3*, 388–397. [CrossRef]
4. Alsewari, A.R.A.; Poston, R.; Zamli, K.Z.; Balfaqih, M.; Aloufi, K.S. Combinatorial test list generation based on Harmony Search Algorithm. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *13*, 3361–3377. [CrossRef]
5. Younis, M.I.; Alsewari, A.R.A.; Khang, N.Y.; Zamli, K.Z. CTJ: Input-output based relation combinatorial testing strategy using jaya algorithm. *Baghdad Sci. J.* **2020**, *17*, 1002–1009. [CrossRef]
6. Chakraborty, S.; Saha, A.K.; Chakraborty, R.; Saha, M.; Nama, S. HSWOA: An ensemble of hunger games search and whale optimization algorithm for global optimization. *Int. J. Intell. Syst.* **2022**, *13*, 3361–3377. [CrossRef]
7. Mafarja, M.M.; Mirjalili, S. Hybrid whale optimization algorithm with simulated annealing for feature selection. *Neurocomputing* **2017**, *260*, 302–312. [CrossRef]
8. AbdulJabbar, I.A.; Abdullah, S.M. Hybrid metaheuristic technique based tabu searchand simulated annealing. *Eng. Technol. J.* **2017**, *35*, 154–160.
9. Abdulsalam, W.H.; Alhamdani, R.S.; Abdullah, M.N. Emotion recognition system based on hybrid techniques. *Int. J. Mach. Learn. Comput.* **2019**, *9*, 490–495. [CrossRef]
10. Arram, A.; Ayob, M.; Sulaiman, A. Hybrid bird mating optimizer with single-based algorithms for combinatorial optimization problems. *IEEE Access* **2021**, *9*, 115972–115989. [CrossRef]
11. Agrawal, A.P.; Choudhary, A.; Kaur, A. An effective regression test case selection using hybrid whale optimization algorithm. *Int. J. Distrib. Syst. Technol.* **2020**, *11*, 53–67. [CrossRef]
12. Alazzawi, A.K.; Rais, H.M.; Basri, S. HABC: Hybrid artificial bee colony for generating variable T-way test sets. *J. Eng. Sci. Technol.* **2020**, *15*, 746–767.
13. Alazzawi, A.K.; Rais, H.M.; Basri, S.; Alsariera, Y.A.; Capretz, L.C.; Balogun, A.O.; Imam, A.A. HABCSm: A hamming based t-way strategy based on hybrid artificial bee colony for variable strength test sets generation. *Int. J. Comput. Commun. Control* **2021**, *16*, 1–18. [CrossRef]
14. Nasser, A.B.; Zamli, K.Z.; Alsewari, A.R.A.; Ahmed, B.S. Hybrid flower pollination algorithm strategies for t-way test suite generation. *PLoS ONE* **2018**, *13*, 1–24. [CrossRef] [PubMed]
15. Alazzawi, A.K.; Rais, H.M.; Basri, S. ABCVS: An artificial bee colony for generating variable t-way test sets. *Int. J. Adv. Comput. Sci. Appl.* **2019**, *10*, 259–274. [CrossRef]
16. Younis, M.I. MVSCA: Multi-valued sequence covering array. *J. Eng.* **2019**, *25*, 82–91. [CrossRef]
17. Younis, M.I. DEO: A dynamic event order strategy for T-way sequence covering array test data generation. *Baghdad Sci. J.* **2020**, *17*, 575–582. [CrossRef]
18. Abualigah, L.; Gandomi, A.H.; Elaziz, M.A.; Hamad, H.A.; Omari, M.; Alshinwan, M.; Khasawneh, A.M. Advances in meta-heuristic optimization algorithms in big data text clustering. *Electronics* **2021**, *10*, 101. [CrossRef]

19. Khalil, Y.; Alshayeji, M.; Ahmad, I. Distributed whale optimization algorithm based on mapreduce. *Concurr. Comput. Pract. Exp.* **2019**, *31*, 2019. [CrossRef]
20. Nadimi-Shahraki, M.H.; Taghian, S.; Mirjalili, S.; Abualigah, L.; Elaziz, M.A.; Oliva, D. Ewoa-opf: Effective whale optimization algorithm to solve optimal power flow problem. *Electronics* **2021**, *10*, 2975. [CrossRef]
21. Sopto, D.S.; Ayon, S.I.; Akhand, M.A.H.; Siddique, N. Modified grey wolf optimization to solve traveling salesman problem. In Proceedings of the 2018 International Conference on Innovation in Engineering and Technology (ICIET), Dhaka, Bangladesh, 27–28 December 2018. [CrossRef]
22. Fatima, A.; Javaid, N.; Anjum Butt, A.; Sultana, T.; Hussain, W.; Bilal, M.; Hashmi, M.A.U.R.; Akbar, M.; Ilahi, M. An enhanced multi-objective gray wolf optimization for virtualmachine placement in cloud data centers. *Electronics* **2019**, *8*, 218. [CrossRef]
23. Dubey, M.; Kumar, V.; Kaur, M.; Dao, T.P. A Systematic review on harmony search algorithm: Theory, literature, and applications. *Math. Probl. Eng.* **2021**, *2021*, 5594267. [CrossRef]
24. Muazu, A.A.; Maiwada, U.D. PWiseHA: Application of harmony search algorithm for test suites generation using pairwise techniques. *Int. J. Comput. Inf. Technol.* **2020**, *9*, 91–98. [CrossRef]
25. Doush, I.A.; Santos, E. Best polynomial harmony search with best β-hill climbing algorithm. *J. Intell. Syst.* **2020**, *30*, 1–17. [CrossRef]
26. Fadhil, H.M.; Abdullah, M.N.; Younis, M.I. Innovations in T-way test creation based on a hybrid hill climbing-greedy algorithm. *IAES Int. J. Artif. Intell. (IJ-AI)* **2022**. *Accepted*.
27. Holm, S. A simple sequentially rejective multiple test procedure. *Scand. J. Stat.* **1979**, *6*, 1979.