International Journal of Engineering Trends and Technology
ISSN: 2231 – 5381 / https://doi.org/10.14445/22315381/IJETT-V71I4P203

Volume 71 Issue 4, 21-28, April 2023

Original Article

Combinatorial Black Hole Algorithm: A Metaheuristic Approach for Combinatorial Testing

Izrulfizal Saufihamizal Ibrahim[1], Rosziati Ibrahim[2], Mazidah Mat Rejab[3]

[1,2,3]*Department of Software Engineering, Universiti Tun Hussein Onn Malaysia, Johor, Malaysia*.

[1]*Corresponding Author : rosziati@uthm.edu.my*

Received: 26 December 2022 Revised: 02 March 2023 Accepted: 01 April 2023 Published: 25 April 2023

Abstract *- Combinatorial Testing (CT) is a software testing technique that aims to identify defects in complex systems by covering as many combinations of input parameters as possible within a given time and resource constraint. The black hole algorithm (BHA) is a metaheuristic approach that has been used in multiple problems involving optimization. In this paper, a new approach called the Combinatorial Black Hole Algorithm (CBHA) is presented for CT that combines the strengths of CT and BHA. The effectiveness of this approach is demonstrated through experiments on a series of real-world computer programs. The findings indicate that the method is feasible in identifying defects with fewer test cases and in less time needed compared to the current technology in CT techniques. The approach can also handle larger and more complex systems more effectively. This study contributes to the software testing field with a way of providing a new and efficient approach for CT that practitioners and researchers can use*.

Keywords *- Black hole algorithm, CT, Metaheuristics, Test cases.*

1. Introduction

CT, a technique for identifying defects in software systems by covering a huge number of combinations of input values and parameters, is widely used. It has been shown to be particularly effective for testing complex systems with multiple interacting components, such as web applications, mobile apps, and embedded systems. After 30 years of effort, CT has been adopted as an important black box testing method in the latest software testing standards [1]. However, the high cost of CT can be a major barrier to its widespread adoption, especially for large and complex systems. Bugs are found and fixed by programmers for around 50% of their time [2].

Moreover, as the complexity and size of software systems increase, the number of possible input combinations can become astronomically large. This can make it difficult to achieve complete coverage using traditional CT techniques. As a result, researchers and practitioners are exploring ways to reduce the cost of CT, such as using machine learning algorithms to automatically generate test cases or prioritizing test cases based on their likelihood of finding defects. Despite these challenges, CT remains a valuable tool for ensuring software quality and reducing the risk of defects in complex systems.

To address this challenge, various heuristic search approaches have been proposed to guide the selection of test cases and reduce the cost of CT. A covering array is employed by CT as the test suite to systematically cover combinations in an effort to achieve a good trade-off between test cases and the effectiveness of revealing failures [2]. BHA has received significant attention among these algorithms for its ability to balance the trade-off between test coverage and cost. BHA is a metaheuristic optimization technique[6] influenced by the behaviour of black holes in space and has been utilized in multiple optimization problems.

In this paper, an innovative approach for CT using BHA is put forward. The strengths of CT and BHA are combined in this approach to achieve high test coverage while minimizing the test cases. The proposed approach is called the Combinatorial Black Hole Algorithm (CBHA). The following research question is aimed to be answered: Can the BHA be used to effectively guide CT and achieve a higher level of test coverage at a lower cost?

To answer this question, a series of experiments were conducted using various software systems and test case combinations. The experimental setup included a range of evaluation metrics, including test coverage, cost, and execution time, to allow for a comprehensive analysis of the execution of the proposed approach.

The contributions of this paper are threefold. First, a comprehensive overview of CT techniques and BHA, including their strengths and limitations, is provided. Second,

a detailed description of the proposed approach for combining CT and BHA is presented, highlighting this approach's key features and advantages. Third, the performance of the execution of the advanced approach is measured through a tier of experiments and compared with the standard CT techniques, providing insights into the competence of the suggested approach under different conditions.

The remainder of this document is organized in a subsequent manner. The next section explains the metaheuristics concept to provide an in-depth understanding of the current issue. The section after that explains BHA to bridge the understanding of metaheuristics with the combinatorial problem. Then, the related work on CT and BHA is reviewed, highlighting previous approaches' key contributions and limitations. In Section 5, the experimental findings are presented and being evaluated against the state-of-the-art techniques, and the implications of the findings are discussed. Section 6 provides the paper's conclusion and outlines potential areas for future research.

2. Metaheuristics

Metaheuristics are high-level strategies that can be utilized for a range of multiple optimization problems to find good solutions that are very close to being optimal for problems that may be too complex or too large to be solved exactly using traditional optimization algorithms. These strategies are designed to improve a set of solutions to the problem iteratively rather than trying to find the exact optimal solution. Metaheuristics have made significant progress in solving complex optimization problems and have been widely applied in various fields. A survey of 14 recent and effective metaheuristics introduced between 2000 and 2020 is presented, along with a discussion of research trends, hybridization of metaheuristics, advances in parallel metaheuristics, open problems, and new research circumstances [3].

Metaheuristics can be categorized into several categories, including local search, population-based, and swarm intelligence. Local search metaheuristics begin with an initial solution and then iteratively improve it by making small changes to the solution. On the other hand, population-based metaheuristics maintain a population of solutions and use techniques such as selection, crossover, and mutation to generate new solutions. Swarm intelligence metaheuristics, sparked by the behaviour of natural systems, use techniques such as swarm intelligence to generate solutions.

Metaheuristics are often utilized in situations where the optimization problem at hand is too complex or too large to be solved exactly or when the problem is not well understood, and there is no efficient algorithm available to solve it. However, stagnant development processes of new metaheuristic approaches have led to a high concentration and frequency in the field of stochastic search. It is important to strive to strike a suitable equilibrium between exploring new options and exploiting existing opportunities to produce superb performance [4]. They are also useful for solving problems with a high degree of uncertainty or when the objective function is not fully known.

Overall, metaheuristics are a powerful tool for solving optimization problems that are too difficult or too large to be solved exactly and can be applied to a range of optimization issues. Software testing is an important IT field involving various testing tactics, strategies, and methodologies, including metaheuristics [5]. They offer a flexible and efficient approach to finding good, near-optimal solutions in situations where traditional optimization algorithms may not be practical or effective.

3. Black Hole Algorithm

The black hole algorithm (BHA) is a metaheuristic approach used to find a valid solution to a given set of input values. It is a search-based optimization technique useful for finding solutions to problems with a considerable amount of input variables and complex interactions between them. BHA is a new bio-inspired metaheuristic algorithm based on the phenomenon of black holes and is an approach based on a population like other bio-inspired computation algorithms [24].

To use the BHA, the input values must first be defined and arranged into an appropriate data structure, such as an array. The BHA process begins by iterating through the input values using a for loop, allowing for processing each element in the array one at a time. At the start of each iteration, a random starting point is initialized using the BHA, representing the current position or solution in the search for a valid solution.

The current solution is then evaluated to determine if it meets the criteria for a valid solution, as defined by the requirements or constraints set for the problem. If the solution is deemed valid, the BHA process is terminated, and the solution is returned. If the solution is invalid, it is discarded, and a new solution is selected using the BHA. This process continues for each iteration of the BHA, with a new, random solution being generated each time. Figure 1 shows the pseudocode for BHA.

Based on Figure 1, one of the key features of the BHA is that it allows for the exploration of a large number of input values and complex interactions between them, as the distance between the black hole (representing the unknown or untested combination of inputs) and the current solution is calculated. The black hole is moved to the new solution if the distance between the two is less than the current distance. BHA has also been found to be faster than other algorithms in testing [7]. This allows for a more thorough search for a valid solution.

Input
Number of stars(N), number of iteration
Output
Black hole
The fitness value of black hole
Begin
Initialize a population of stars
For j = 1 to number of stars
Calculate the objective function of the star(j) and save in fitness array(f)
Next j
The star with the most remarkable fitness value is chosen as the black hole
While (max iteration or convergence criteria is not met) do
For a = 1 to number of stars

$$X_a^{new} = X_a^{old} + rand \times (X_{BH} - X_a^{old})$$

Evaluate fitness value of the star(X_a)
If fitness of (X_a) > fitness of (X_{BH})**Then**

$$X_{BH} = X_a$$

End if
Replace the new fitness value of the star (X_a) with the previous value
Update the fitness array(f) and calculate:

$$R = \frac{f_{BH}}{\sum_{i=1}^{N} f_i}$$

If $\sqrt{(X_{BH} - X_a)^2} < R$ **Then**
Replace X_a with a new star in an optional location in the search scope
End if
next a
end while
End

Fig. 1 BHA Pseudocode [24]

BHA is a metaheuristic that has been demonstrated to be competent in various applications and has been modified and extended in various ways in the literature [8]. On top of that, a study shows that BHA can be used to find the optimal values for the vector of the feature of a Support Vector Machine for emotion classification based on EEG signals to achieve results similar to manual noise elimination methods [9].

However, it is important to note that the BHA may not always find a valid solution, as it can reach the maximum number of iterations without finding one, potentially due to the lack of valid solutions within the defined input values or the inability to find a valid solution due to the random nature of the algorithm. In general, BHA is a useful tool for finding valid solutions to problems with many input variables and complex interactions between them due to its ability to explore a wide range of possibilities and its lack of preconceived notions or biases.

4. Related Work

In the literature, various heuristic search algorithms have been proposed to guide the selection of test cases and reduce the cost of CT. These algorithms have the potential to significantly improve the efficiency of CT by focusing on the most relevant combinations of input values and parameters. BHA, a metaheuristic optimization technique stimulated by the behaviour of black holes in space and applied to various optimization problems, has received significant attention for its ability to balance the trade-off between test coverage and cost. It has been shown to be effective in finding near-optimal solutions to a multitude of optimization problems and has the potential to improve the efficiency of CT significantly. The following section presents a review of the related work on CT and BHA, focusing on previous efforts to combine these two approaches. These approaches' key contributions and limitations are discussed, and their relevance to the proposed approach is highlighted. An overview of the current standard in CT and BHA is also provided, and the main research gaps the work aims to address are outlined.

In 2020, a research investigation was carried out using BHA to solve the knapsack problem. The findings from this research show that the black hole approach can find better solutions regarding the quality and in lesser time compared to other metaheuristics approaches [10]. Other study includes the application of BHA to the t-way testing approach. The study was done to see if BHA can be improved for t-way testing. The improved algorithm is called BBH (Binary Black Hole). The research outcome shows that BBH achieved the desired improvement in generating small covering arrays compared with BPSO [25]. The hybridization of two metaheuristics algorithms was also conducted in 2020. A hybrid BHA and genetic algorithm conducted a study to solve an optimization problem. The result of this study is that the proposed algorithm works better and faster than the original BHA and genetic algorithm [11,12]. The study uses benchmark functions to test the proposed algorithm. Another application of BHA in research was conducted recently in the field of software testing, where the algorithm was used in solving the problem regarding CT's test case explosion, and the improved algorithm is called BH-AllStar. For the study, BH-AllStar is being compared with BBH, and the result shows that it outperforms BBH in terms of generating test cases which achieve a 43% increase in condition coverage [13].

Combining CT and heuristic search algorithms has received significant attention in the literature to improve the efficiency of CT. Various heuristic search algorithms, including ant colony optimization, genetic algorithms, and

particle swarm optimization, have been applied to CT with promising results in terms of test coverage and cost. BHA, a metaheuristic optimization technique encouraged by the behaviour of black holes in space, has also been explored to guide CT. Previous work combining CT and BHA has focused on various approaches, including using BHA to select test cases and optimize the CT process. These approaches have achieved promising results, but there is still room for improvement and further studies are needed to fully understand the potential of combining CT and heuristic search algorithms. A study was done in 2020 using Ant Colony Optimization (ACO) with Fuzzy Logic called ACOF for CT [14]. Another study solving fortifying CT used Whale Optimization Algorithm (WOA) to find the most optimum test cases for a transport system [15]. Other metaheuristics approaches include BMBH (Multiple Black Hole) [16], GSA (Gravitational Search Algorithm) [17], HSCA (Hybrid Sine Cosine Algorithm) [18], ABC (Artificial Bee Colony) [19], FPA-HC (Hybrid Flower Pollination and Hill Climbing Algorithm) [20], CS (Cuckoo Search) [21] and FA (Firefly Algorithm). The summary of the studies is shown in Table 1.

Table 1. Recent Study Summary

Study	Baseline Method	Dataset/Project	Evaluation Criteria
[14]	ACOF	Test suite for Scholarship Application	Test suite size
[15]	WOA	Transport system parameters	Test suite size performance
[16]	BMBH	Pizza ordering system and Smart Mobile System	Benchmark functions
[17]	GSA	Foodpanda Delivery Service Search System	Test suite size
[18]	HSCA	Web configurable software system	Test suite size
[19]	ABC	Find dialogue box	Test suite size
[20]	FPA-HC	Car Ordering System	Test suite size
[21]	CS	Ms Word Paragraph Dialogue box	Test suite size
[22]	FA	Matrices	Test Cases
[23]	GA	Surveys	Test case generation

5. Proposed Approach - CBHA

A valid solution to a given set of input values can be found using BHA. The criterion for a valid solution is defined as meeting certain requirements or constraints that have been set for the problem.

To begin the BHA process, the input values used in the problem are first defined. These inputs, which can be variables such as numbers, strings, or objects, are typically arranged into arrays or other data structures. Then, through the use of a for loop, the input arrays are iterated through, allowing for the processing of each element of the array one at a time. At the start of each iteration, a random starting point is initialized using BHA. This starting point represents the current position or solution in the search for a valid solution. The current solution is evaluated to determine if it meets the criteria for a valid solution. If it does, the BHA process is terminated, and the solution is returned. If the solution is invalid, it is discarded, and a new solution is selected using BHA. For each iteration of the BHA process, a new, random solution is generated.

The distance between the black hole (representing the unknown or untested combination of inputs) and the current solution is then calculated. The black hole is moved to the new solution if the distance between the two is less than the current distance. The new solution is then evaluated to see if it is a valid solution. If it is, the BHA process is terminated, and the solution is returned.

If the BHA process reaches the maximum number of iterations without finding a valid solution, an error message is returned indicating that no solution was found. This can occur if there are no valid solutions within the defined input values or if the BHA process cannot find a valid solution due to the random nature of the algorithm. BHA is a useful tool for finding valid solutions to problems with many input variables and complex interactions between them. The steps for the Combinatorial Black Hole Algorithm (CBHA) are shown in Figure 2, and the implementation in Python is shown in Figure 3.

1) Define the problem and the criteria for a valid solution.
2) Define the inputs to use in the problem.
3) Iterates through the input arrays using a *for* loop.
4) Initialize a random starting point in each iteration using BHA.
5) Evaluates the current solution.
   i. If valid, terminate the and return the solution.
   ii. If invalid, discard and selects a new solution using BHA.
6) For each BHA iteration,
   i. Generate a random solution.
   ii. Measure the distance between the black hole and the current solution.
   iii. Move the black hole to the new solution if the distance is less than the current distance.
   iv. If the solution is valid, terminate and return the solution.
7) If max iteration is reached with no valid solution, return an error message.

Fig. 2 Steps for Combinatorial Black Hole Algorithm (CBHA)

```
for input in inputs:
  current_solution = random.sample(input, 5)
  if is_valid_solution(current_solution):
    print("Valid solution found for input ", input, ": ", current_solution)
  else:
    black_hole = current_solution
    max_iterations = 100
    for i in range(max_iterations):
      solution = random.sample(input, 4)
      distance = sum([abs(a-b) for a, b in zip(solution, black_hole)])
      if distance < sum([abs(a-b) for a, b in zip(black_hole, solution)]):
        black_hole = solution
      if is_valid_solution(solution):
        print("Valid solution found for input ", input, ": ", solution)
        break
    else:
      print("No valid solution found for input ", input)
```

Fig. 3 Code Snippet for CBHA in Python

The algorithm works by randomly generating a set of five elements from the input and checking whether this set is a valid solution using the is_valid_solution() function. If the randomly generated set is a valid solution, it is printed to the console. If it is invalid, the algorithm initializes a "black hole" variable to this set of five elements and proceeds to iterate up to 100 times through a loop. The algorithm randomly generates a new set of five elements from the input within each loop iteration. It calculates the distance between this new set and the "black hole" using the sum([abs(a-b) for a, b in zip(solution, black_hole)]) expression. If the distance is less than the distance between the "black hole" and the current solution, the "black hole" is updated to this new solution. This is intended to gradually move the algorithm towards a solution closer to the optimal solution. If a valid solution is found within the loop, it is printed to the console and the loop is terminated using the break statement. If no valid solution is found within the loop, the algorithm prints a message to the console indicating no valid solution.

Considering the MS Excel software dialogue in Figure 4, here are 5 parameters or options (i.e., page orientation, scaling adjustments, scaling fit to, scaling fit width, first-page number). The possible parameters for the first input are 2, 6 possible parameters for the second input, and so on. For the input type number, the possible parameters that can be input would be a negative number, a positive number, zero value, null, and NaN (not-a-number), which are summarised in Table 2.

Table 2. Possible Parameter Values

Orientation (A)	Scaling			First-page number (E)
	Adjust to (B)	Fit to		
		Pages (C)	Width (D)	
Portrait	-10	-10	-10	-10
Landscape	10	10	10	10
	0	0	0	0
	null	*null*	*null*	*null*
	NaN	NaN	NaN	NaN
				Auto

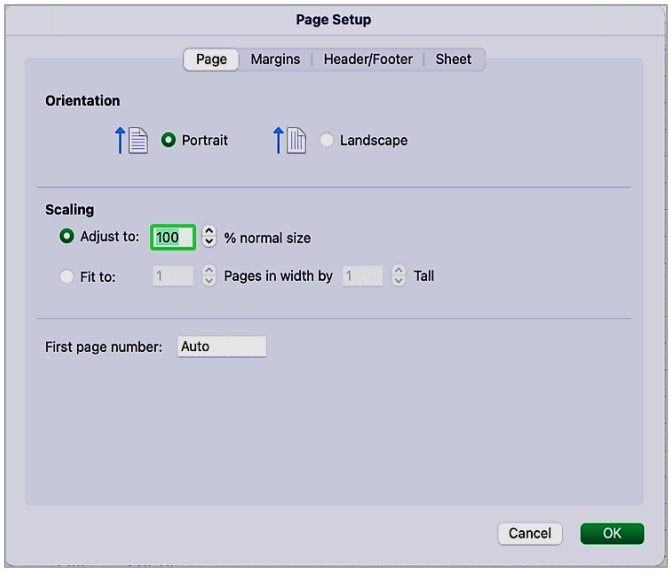

Fig. 4 MS Excel Dialogue Box

6. Results and Discussion

Before applying the CBHA, the total test combinations are totally up to 3000 combinations, as shown in the calculation:

$$A = \{\text{Portrait}, Landscape\}$$
$$B = \{Adjust, Fit\}$$
$$C = \{-10, 10, 0, null, NaN\}$$
$$D = \{-10, 10, 0, null, NaN\}$$
$$E = \{-10, 10, 0, null, NaN\}$$
$$F = \{-10, 10, 0, null, NaN\}$$

$$\therefore |test\ cases| = |A| \times |B| \times |C| \times |D| \times |E| \times |F|$$

The Cartesian product is used in this case to generate all possible combinations of the input values. This can be useful for testing purposes, as it allows the quick and easily create of a large quantity of test scenarios that cover all possible combinations of input values. The Cartesian product A x B x C x D x E x F can then be represented as the set of all ordered 6-tuples (a, b, c, d, e, f) such that a belongs to A, b belongs to B, c belongs to C, d belongs to D, e belongs to E, and f belongs to F.

Table 3. Test Suite Parameters

Base Values	Input variables				
	(A)	(B)	(C)	(D)	(E)
	{Portrait, Landscape}	{null, NaN, 0}	{null, -10, NaN}	{null, -10, NaN}	{null, -10, NaN}

Table 4. Test Case Combinations Comparison

Test Case Combinations	**Exhaustive Combinatorial**	**CBHA**
6 tuples (A, B, C, D, E, F)	3000 possible combinations	432 faulty combinations
3 tuples (A, B, C)	20 possible combinations	12 faulty combinations

This can be written more formally as:

$$A \times B \times C \times D \times E \times F = \{(a,b,c,d,e,f) | a \in A, b \in B, c \in C, d \in D, e \in E, f \in F\}$$

In this case, the test cases variable is defined as the set of all ordered 6-tuples in the Cartesian product A x B x C x D x E x F. The length of this set is equal to the number of tuples in the set. The test cases set would contain a total of |A| * |B| * |C| * |D| * |E| * |F| tuples, which is equal to 2 * 2 * 5 * 5 * 5 * 6 tuples. This means that the length of the test case set would be a total of 3000 combinations.

Applying CBHA to the case study greatly reduces the test suite of the example in Figure 3, as shown in Table 3. The total test suite size is reduced to a total of 2 * 2 * 3 * 3 * 3 * 5 tuples which is 432 instead of 3000 possible combinations by using the same calculations. The comparison of test case combinations is summarised in Table 4.

Based on the calculation, it is clear that by applying CBHA, the amount of test suites is greatly reduced by eliminating the not-needed value as inputs. As a result, the number of test cases will also be reduced to 85.6% of the total combinations. The calculation for the two case studies is shown as follows:

- Without CBHA,

$$Test\ cases = A \times B \times C \times D \times E \times F$$
$$Test\ cases = 2 \times 2 \times 5 \times 5 \times 5 \times 6$$
$$Test\ cases = 3000$$

- With CBHA

$$Test\ cases = A \times B \times C \times D \times E \times F$$
$$Test\ cases = 2 \times 2 \times 3 \times 3 \times 3 \times 5$$
$$Test\ cases = 432$$

- Percentage of reduction = 85.6%

To explain the reduction, the original set of 3000 test cases may include many test cases that are similar to each other or that test the same functionality in different ways. In such cases, it may be possible to lessen the numerical count of tests that need to be executed by identifying and eliminating redundant or unnecessary test cases. This can be beneficial for several reasons, such as reducing the time and resources required to run the test cases and making it effortless to analyse and interpret the results of the tests.

One way to identify and remove similar or redundant test cases is by using an algorithm like the CBHA. The CBHA is a technique designed specifically for identifying and removing test cases that are similar to each other. It works by starting with a set of test cases and iteratively removing test cases that are not "useful" in testing the system's functionality under test.

For example, let us say that the original set of 3000 test cases includes 1000 test cases that are identical to each other and 1000 test cases that are similar to each other but have minor differences. In this case, CBHA might be able to identify and remove the 1000 identical test cases, leaving 2000 test cases remaining. It could then identify and remove the 1000 test cases that are similar to each other but have minor differences, leaving 1000 test cases remaining.

In this example, the utilization of CBHA would result in a reduction of 3000 test cases to 432 test cases. Suppose the algorithm is able to reduce further the total amount of tests that are required to be performed to assess the functionality, reliability, and performance of a system or software by identifying and eliminating additional redundant or unnecessary test cases. In that case, it is possible that the final test cases could be reduced to fewer than 432.

Overall, CBHA can be a very effective approach for cutting down the amount of test cases in a set while still ensuring that the set is comprehensive and covers all necessary functionality. By iteratively identifying and removing test cases similar to each other, CBHA can significantly drop the number of test cases without sacrificing the quality or completeness of the tests.

7. Conclusion and Future Work

Much recent research show that CT can be improved using metaheuristics approaches and is very efficient in reducing test suite optimally. However, based on the recent study, it is clear that there will be some test cases that will be “missed” among the needed ones in the test suite.

This research proposes a new approach which is the Combinatorial Black Hole Algorithm (CBHA), a new application of metaheuristics to the standard CT. The study results clearly show that the CBHA can generate results that can be improved. In the future, the application of BHA to T-way Testing is still currently being worked on to be evaluated and improved for other case studies. For future research, other methods that may be used in conjunction with CBHA, including statistical sampling, domain partitioning, and functional coverage analysis, can be explored. By combining these techniques and carefully selecting the most appropriate approach for a given situation, it is possible to significantly reduce the number of test cases while still ensuring that the system under test is thoroughly tested.

Acknowledgements

The authors express their gratitude to Universiti Tun Hussein Onn Malaysia (UTHM) for assisting with their research. They were awarded financial support for this investigation through the REGG Grant under Grant Vote No Q051. Their research is disseminated through financial aid from Universiti Tun Hussein Onn Malaysia and the UTHM Publisher's Office through Publication Fund E15216.

References

[1] Tzoref-Brill, R, "Advances in Combinatorial Testing," *Advances in Computers*, vol. 112, pp. 79-134, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[2] Huayao Wu et al., "Combinatorial Testing of RESTful APIs," *ACM/IEEE International Conference on Software Engineering (ICSE)*, 2020. [Google Scholar] [Publisher Link]

[3] Tansel Dokeroglu et al., "A Survey on New Generation Metaheuristic Algorithms," *Computers & Industrial Engineering*, vol. 137, p. 106040, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[4] Bernardo Morales-Castañeda et al., "A Better Balance in Metaheuristic Algorithms: Does It Exist?," *Swarm and Evolutionary Computation*, vol. 54, p. 100671, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[5] Madhavi, D, "A White Box Testing Technique in Software Testing: Basis Path Testing," *Journal for Research*, vol. 2, no. 4, 2016. [Google Scholar] [Publisher Link]

[6] Ramanathan.L, and Ulaganathan.K, "Nature-Inspired Metaheuristic Optimization Technique-Migrating Bird'S Optimization in Industrial Scheduling Problem," *SSRG International Journal of Industrial Engineering*, vol. 1, no. 2, pp. 12-17, 2014. [CrossRef] [Publisher Link]

[7] Laith Abualigah et al., "Black Hole Algorithm: A Comprehensive Survey," *Applied Intelligence*, pp. 1-24, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[8] Abdolreza Hatamlou, "Solving Travelling Salesman Problem Using Black Hole Algorithm," *Soft Computing*, vol. 22, no. 24, pp. 8167-8175, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[9] Rodrigo Olivares et al., "Using Black Hole Algorithm to Improve EEG-Based Emotion Recognition," *Computational Intelligence and Neuroscience*, vol. 2018, p. 21, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[10] Abdolreza Hatamlou, "Application of Black Hole Algorithm for Solving Knapsack Problems," *Computer and Knowledge Engineering*, vol. 3, no. 1, pp. 117-122, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[11] Seeven Amic, K. M. Sunjiv Soyjaudah, and Gianeshwar Ramsawock, "Fitness Landscape Analysis of Block Ciphers for Cryptanalysis Using Metaheuristics," *International Journal of Engineering Trends and Technology*, vol. 70, no. 6, pp. 257-271, 2022. [CrossRef] [Publisher Link]

[12] Omar Sabah Mohammed, Adel Abo Al-Maged Sewisy, and Ahmed Ibrahim Taloba, "Solving Optimization Problems Using Hybrid Metaheuristics: Genetic Algorithm and Black Hole Algorithm," *2020 2nd International Conference on Computer and Information Sciences (ICCIS)*, pp. 1-5, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[13] Derya Yeliz Ulutaş, and Ayşe Tosun, "A Condition Coverage-Based Black Hole Inspired Meta-Heuristic for Test Data Generation," *CEUR Workshop Proceedings*, pp. 70-78, 2021. [Google Scholar]

[14] Mohd Zamri Zahir Ahmad et al., "A Self-Adapting Ant Colony Optimization Algorithm Using Fuzzy Logic (ACOF) for Combinatorial Test Suite Generation," *IOP Conference Series: Materials Science and Engineering*, vol. 767, no. 1, 2020.[CrossRef] [Google Scholar] [Publisher Link]

[15] Ali Abdullah Hassan et al., "Combinatorial Test Suites Generation Strategy Utilizing the Whale Optimization Algorithm," *IEEE Access*, vol. 8, pp. 192288-192303, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[16] Hamsa Naji Nsaif Al-Sammarraie, and Dayang N. A. Jawawi, "Multiple Black Hole Inspired Meta-Heuristic Searching Optimization for CT," *IEEE Access*, vol. 8, pp. 33406-33418, 2020. [CrossRef] [Google Scholar] [Publisher Link]

[17] Khin Maung Htay et al., "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, 2022. [CrossRef] [Google Scholar] [Publisher Link]

[18] J M Altmemi et al., "Implementation of Hybrid Sine Cosine Algorithm for Input-Output CT," *International Conference on Applied Computing*, 2021. [Google Scholar]

[19] Ammar K Alazzawi, Helmi Md Rais, and Shuib Basri "Artificial Bee Colony Algorithm for T-Way Test Suite Generation," *4th International Conference on Computer and Information Sciences (ICCOINS)*, pp. 1-6, 2018. [CrossRef] [Google Scholar] [Publisher Link]

[20] Abdullah B. Nasser et al., "T-Way Test Suite Generation Based on Hybrid Flower Pollination Algorithm and Hill Climbing," *10th International Conference on Software and Computer Applications*, pp. 244-250 2021. [CrossRef] [Google Scholar] [Publisher Link]

[21] Abdullah B. Nasser, and Kamal Z. Zamli "A New Variable Strength T-Way Strategy Based on the Cuckoo Search Algorithm," *Intelligent and Interactive Computing*, pp. 193-203, 2019. [CrossRef] [Google Scholar] [Publisher Link]

[22] Muhammad Afiq Ariffin et al., "Test Cases Prioritization Using Ant Colony Optimization and Firefly Algorithm," *International Journal of Engineering Trends and Technology*, vol. 70, no. 3, pp. 22–28, 2022. [CrossRef] [Publisher Link]

[23] Deepak kumar, and Manu Phogat "Genetic Algorithm Approach for Test Case Generation Randomly: A Review," *International Journal of Computer Trends and Technology*, vol. 49, no. 4, pp. 213-216, 2017. [CrossRef] [Google Scholar] [Publisher Link]

[24] Santosh Kumar, Deepanwita Datta, and Sanjay Kumar Singh, "Black Hole Algorithm and Its Applications," *Computational Intelligence Applications in Modelling and Control*, pp. 147-170, 2015. [CrossRef] [Google Scholar] [Publisher Link]

[25] Hamsa N Nsaif, and Dayang Norhayati Abang Jawawi, "Binary Black Hole-Based Optimization for T-Way Testing," *IOP Conference Series: Materials Science and Engineering*, vol. 864, no. 1, p. 012073. [CrossRef] [Google Scholar] [Publisher Link]