Review of Nature Inspired Spotted Hyena Optimization Algorithm for Combinatorial t-Way Testing

Sami Rashid
Department of CSE
Independent University Bangladesh
Dhaka, Bangladesh
2120326@iub.edu.bd

Md Fahim Shahriar
Department of CSE
Independent University Bangladesh
Dhaka, Bangladesh
2120552@iub.edu.bd

Faria Anjum
Department of CSE
Independent University Bangladesh
Dhaka, Bangladesh
2120266@iub.edu.bd

***Abstract*—Combinatorial testing is an efficient approach for software testing that aims to minimize the number of test cases required to cover all possible combinations of input parameters, while maintaining high fault detection capability. However, generating optimal test suites for t-way combinatorial testing, where t represents the interaction strength or the number of parameters considered together, is an NP-hard problem. In this paper, we propose the application of a nature-inspired meta-heuristic algorithm called the Spotted Hyena Optimization (SHO) algorithm to solve the t-way combinatorial testing problem. The SHO algorithm mimics the hunting behavior of spotted hyenas, encircling, hunting, and attacking the prey, which represents the optimal solution in the search space. We integrate the SHO algorithm with combinatorial testing techniques and develop a fitness function tailored to the t-way testing problem. The proposed approach efficiently selects a minimized set of test cases that covers all possible combinations of parameter values at most once. Experimental results for a 2-way testing scenario involving four parameters with two valid values each demonstrate the efficacy of our approach. The SHO algorithm generated a compact test suite of six test cases, achieving a reduction of 62.5% compared to exhaustive testing. This study contributes to the ongoing research efforts in combinatorial testing by presenting a robust and efficient method for generating optimized test suites using a nature-inspired optimization algorithm.**

***Index Terms*—T-way testing, combinatorial testing, software testing, meta-heuristic, optimization algorithm, Spotted Hyena Optimization, SHO**

I. Introduction

Software development is a complex process with innumerable factors attempting to provide its users with quality, accuracy and usability, and testing is an integral part of ensuring all three. Software testing includes configuration of different input parameters or values. Often the scale of all possible combinations of these input parameters is huge, and testing all of them can be painstakingly difficult and time-consuming. This leads researchers to search for ways to minimize the volume of test cases using cost-effective methods and tools. Combinatorial testing is one such method, which minimizes number of test cases by creating subsets of interaction between parameters instead of considering all parameters at once, while keeping high fault detection ability [1]. This mechanism is called t-way testing, where the 't' refers to interaction strength, or how many parameters are being tested together at a time [4].

Metaheuristic algorithms are generally used to aid t-way testing, as they can find the best or optimal solution for complex optimization problems [3]. T-way combinatorial testing problems are NP-hard, meaning there is no one algorithm that can be always effective for all interactions in all test cases. Thus, researchers are always trying to adopt new and varied algorithms in order to facilitate generating optimal test suites for unique problems with specific interaction strengths.

Our objective in this paper is to solve the t-way combinatorial testing problem by utilizing a nature inspired metaheuristic swarm algorithm called Spotted Hyena Optimization (SHO) in order to generate a minimal number of test cases that covers all interactions of our input parameters.

II. Algorithm Description

The inspiration of this algorithm stems from the social interaction between the animals and how they construct the overall network analysis of animal behavior [2]. The spotted hyenas are dog-like carnivores who are very skillful hunters and are the largest among three other hyena species. Spotted hyenas hunt in a group and the female hyenas hold more of a dominant figure than male hyenas. This specific batch of carnivores hunt in a very socially trusted manner in which they only group with hyenas that are linked in some way through kinship. They recognize their kin through their calls such as postures and signals. In this section, the hunting techniques of the hyenas are mathematically modeled and illustrated.

The first step of this algorithm is to encircle the prey. In algorithmic terms, it means finding the proper best candidate solution. This best candidate solution is the target or the prey the hyenas are trying to hunt. After defining the best candidate solution the search agents will update their position. Since, the whole search space is not known beforehand, the rule of thumb is that the best candidate solution is as close to the optimum solution as possible.

The mathematical formulas to define this nature is as follows:

$$\bar{D}_h = \left|\bar{B} \cdot \bar{P}_p(x) - \bar{P}(x)\right| \tag{1}$$

$$\bar{P}(x+1) = \bar{P}_p(x) - \bar{E} \cdot \bar{D}_h \tag{2}$$

Here, D_h is defined as the distance between the prey and the hyena. x indicates the current iteration, B and E are coefficient vectors. P_p indicates the position vector of the prey and P is the position vector of the hyena. B and H are calculated using are random vector between 0 and 1 to retain the randomness of this algorithm.

The following step of this algorithm is to hunt. Since, the hunting nature of the hyenas are defined, this step starts with finding the best search agent. In this case the best search agent is whoever is closest to the best candidate solution. The other search agents move towards the best search agents by updating their position with respect to the best solution that they have obtained so far.

The equations illustrating this form of mechanism are:

$$\bar{D}_h = \left|\bar{B} \cdot \bar{P}_h - \bar{P}_k\right| \tag{3}$$

$$\bar{P}_k = \bar{P}_h - \bar{E} \cdot \bar{D}_h \tag{4}$$

$$\bar{C}_h = \bar{P}_k + \bar{P}_{k+1} + \ldots + \bar{P}_{k+N} \tag{5}$$

P_h is defined as the position of the first best spotted hyena and P_k tells the position of the other spotted hyenas. N is the number of hyenas computed so far.

The third step of spotted hyena optimization is to attack the prey. In order to model this step we decrease the value of h vector which is used to properly balance out the exploration and exploitation and defined as:

$$\bar{h} = 5 - \left(\text{Iteration} \times \frac{5}{\text{MaxIteration}}\right) \tag{6}$$

The variation in vector E is also decreased in order to properly integrate the value of h. The formula for attacking the prey is:

$$\bar{P}(x+1) = \frac{\bar{C}_h}{\bar{N}} \tag{7}$$

Here, the $P(x+1)$ saves the best solution and updates the other search agents accordingly.

The search for prey or in this case the exploration depends on the position of the group, which is defined by the C_h. The search agents tend to move away from each other in order to find the best global solution rather than a local solution. This is why E is used with a random value in between 0 to 1.

An overall overview of this algorithm can be boiled down to:

- Save the best possible solution so far over the iteration.
- Circle the best candidate solution which can be extended to higher dimensions.
- Random vectors such as B and E assist the candidate solution with random variables.
- The exploration and exploitation is adjusted with tweaking the values of E and h vectors respectively.

III. Experimental Setup

This study was conducted to implement a innovative approach of optimizing test case generation utilizing the idea of t-tuple coverage in combinatorial testing, by integrating the swarm based SHO algorithm. The underlying idea of the approach is to efficiently select a minimized set of test cases that covers all the possible combinations of input parameter at most once, which is essential to ensure consistent efficiency of the algorithm throughout the entire testing process.

A. Generating t-tuple table

The main component of this approach is a t-tuple table, which contains all possible combinations of parameter pairs. This table allows the implemented SHO algorithm to keep track of the combinations that has been covered by the generated test cases. Entries of this table indicates an unique pair of parameter values, which is used to ensure that the generated suite of test cases collectively cover every entry of the table at least once.

B. Implementing SHO algorithm

The swarm based SHO algorithm is implemented in this study that simulates the foraging behavior of hyenas. This algorithm is chosen for its balance of exploration and exploitation capabilities, making it suitable for navigating through the search space of potential test cases. The algorithm initiates with a population of random test cases, which are referred to as hyenas, each representing a potential solution to the test case generation problem.

C. Implementing Fitness function and Updating Hyenas position

In each iteration of the algorithm, the effectiveness of each hyena in covering the entries of the t-tuple table is evaluated. The effectiveness or 'fitness' of a hyena is determined by the number of previously uncovered entries it can cover.

The best-performing hyenas influence the rest of the population, guiding them towards more promising areas of the search space. This is achieved through a dynamic adjustment of each hyena's position, influenced by the best solution found in the current iteration. The position adjustment is governed by a set of mathematical equations mentioned in [2], that ensure a balance between diversifying the search (exploring new areas) and intensifying the search around promising solutions (exploiting known good areas).

D. Updating t-tuple table by removing entries

As the algorithm progresses, the t-tuple table is updated by marking the entries covered by the best hyena of each iteration. This process continues iteratively until the iteration stops when all entries in the t-tuple table are covered, indicating that a set of test cases has been found that collectively cover all possible parameter pair combinations.

AB	AC	AD	BC	BD	CD
00XX	0X0X	0XX0	X00X	X0X0	XX00
01XX	0X1X	0XX1	X01X	X0X1	XX01
10XX	1X0X	1XX0	X10X	X1X0	XX10
11XX	1X1X	1XX1	X11X	X1X1	XX11

(a) 1100 as test solution

AB	AC	AD	BC	BD	CD
00XX	0X0X	0XX0	X00X	X0X0	
01XX	0X1X	0XX1	X01X	X0X1	XX01
10XX					XX10
	1X1X	1XX1	X11X	X1X1	XX11

(b) 1011 as test solution

AB	AC	AD	BC	BD	CD
00XX	0X0X	0XX0	X00X	X0X0	
01XX	0X1X	0XX1			XX01
					XX10
			X11X	X1X1	

(c) 0000 as test solution

AB	AC	AD	BC	BD	CD
01XX	0X1X	0XX1			XX01
					XX10
			X11X	X1X1	

(d) 0111 as test solution

AB	AC	AD	BC	BD	CD
					XX01
					XX10

(e) 0010 as test solution

AB	AC	AD	BC	BD	CD
					XX01

(f) 1010 as test solution

Fig. 1. (a) Entries covered by 1100 as test solution (n=6); (b) Entries covered by 1011 as test solution (n=6); (c) Entries covered by 0000 as test solution (n=5); (d) Entries covered by 0111 as test solution (n=5); (e) Entries covered by 0010 as test solution (n=1); and (f) Entries covered by 1010 as test solution (n=1)

E. Final test suite generation

The result of the processes so far mentioned is a suite of test cases that ensures comprehensive coverage of the parameter space with minimal redundancy. This suite represents an optimized solution to the problem of test case generation in combinatorial testing, ensuring thoroughness and efficiency in testing processes. The approach detailed in this study demonstrates the effectiveness of integrating swarm based SHO algorithm with combinatorial testing techniques, offering a robust method for generating test cases in software testing.

The source code used for implementing SHO algortihm for t-way testing problem can be found at this GitHub repository.

IV. Results

The results obtained in this study indicates that the implemented SHO optimization algorithm is capable of generating a minimized test suite that covers all possible combinations of parameter pairs efficiently.

The results obtained and presented in this study has been conducted for 2-way testing using 4 parameters, while each parameter can have 2 valid values [0 or 1]. The process of test suite generation is demonstrated in Fig. 1. The SHO algorithm at each iteration selects the best hyena based on how many entries it can cover from the t-tuple table. For a particular run of the algorithm, the best hyena (test solution) selected by the SHO algorithm during first iteration was '1100'. As seen in Fig. 1-a, the test solution ('1100') covers a total of six entries which are marked in yellow. The covered entries are then removed from the t-tuple table, to ensure that same entries is not covered more than once by the test solutions in the next iteration.

Test Solutions	Weights
1100	6
1011	6
0000	5
0111	5
0010	1
0101	1

Fig. 2. Final optimized test suite

The SHO algorithm runs the second iteration, where '1011' is selected as the best hyena or test solution. The entries covered (n=6) by this test solution are demonstrated in Fig. 1-b. Similarly, in the third iteration, '0000' is selected as the best test solution, covering entries (n=5) shown in Fig. 1-c, while in the fourth iteration, '0111' covers the entries (n=5) shown in Fig. 1-d when selected as the best test solution by the

SHO algorithm. Iteration five and six are the last two iterations of the algorithm, where '0010' and '1010' is selected as the best test solution, and both of them covers only one entry from the t-tuple table which in shown in Fig. 1-e and Fig. 1-f respectively.

The final test suite generated, along with their weights indicating how many entries are covered by each test solution of the suite is shown in Fig. 2. The SHO algorithm therefore generates a total of six test cases that can collectively cover all possible combination of parameter values ($2^4 = 16$, parameter = 4, values for each parameter = 2), reducing the test suite from 16 (for exhaustive method) to mere 6 (for t=2), a saving of 62.5 percent.

V. Conclusion

This study demonstrates the successful application of the potted Hyena Optimization (SHO) algorithm to address the t-way combinatorial testing problem. Taking inspiration from the hunting behavior of spotted hyenas, the SHO algorithm proved effective in generating optimized test suites that can cover all possible combinations of parameter values for a given interaction strength (t).

The key contributions of this study includes the integration of the swarm-based SHO optimization algorithm with the combinatorial testing techniques, and the development of a fitness function tailored to the t-way testing problem. By utilizing the ability of SHO algorithm's to balance exploration and exploitation, the proposed approach efficiently navigated the search space of potential test cases, ultimately identifying a minimized set of test cases that collectively covered all required parameter combinations.

The experimental results demonstrated the efficacy of the approach, showcasing its ability to generate a compact test suite for a 2-way testing scenario involving four parameters with each consisting of two valid values. Compared to the exhaustive testing method, the SHO algorithm achieved a significant reduction of 62.5% in the number of required test cases, signifying the its potential for enhancing the efficiency and cost-effectiveness of software testing processes.

Overall, this study contributes to the ongoing efforts aimed at advancing the combinatorial testing by presenting a novel application of the nature-inspired SHO optimization algorithm. Future work could explore the extension of this approach to higher interaction strengths, more complex parameter configurations, and integration with other testing methodologies.

VI. Contributions

A. Research and Primal Code Implementation

In this phase all the group members participated in active research to find the nitty gritty of the t-way problem and its implementation. The major setback the group faced during the research is the confusion regarding how to approach, in order to solve this np hard problem. Group members were confused between choosing a specific method such as Uniform strength, Variable strength and Input output relation based. This problem, however, was mitigated once the actual code was studied of an already implemented algorithm.

The skeleton code of the overall problem was designed by Fahim Shahriar and Faria Anjum, in which all the exhaustive interaction was implemented from which the repetition was removed. From this step the group moved on to the next phase.

B. Algorithm generation and Defining Functions

The algorithm was generated by Sami Rashid and Fahim Shahriar by the help of chat gpt and the details of SHO algorithm given in [2]

The needed functions such as fitness function and the according t-table generation was implemented by Faria Anjum.

C. SHO implementation and Report

The main implementation of the code was done by Sami Rashid and on standby were the other two group members pointing out any shortcomings or any other efficient way regarding the algorithm

Lastly, all three of the group members contributed equally in the report writing.

References

[1] Xiang Chen, Qing Gu, Ang Li, and Daoxu Chen. Variable strength interaction testing with an ant colony system approach. In *2009 16th Asia-Pacific Software Engineering Conference*, pages 160–167. IEEE, 2009.

[2] Gaurav Dhiman and Vijay Kumar. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software*, 114:48–70, 2017.

[3] Aminu Aminu Muazu, Ahmad Sobri Hashim, and Aliza Sarlan. Review of nature inspired metaheuristic algorithm selection for combinatorial t-way testing. *IEEE Access*, 10:27404–27431, 2022.

[4] Kamal Zamli. T-way strategies and its applications for combinatorial testing. *International Journal of New Computer Architectures and their Applications (IJNCAA)*, 2:459–473, 01 2011.