Spotted Hyena Optimization

Yogesh Shinde

Indian Institute of Information Technology(IIIT), Dharwad
20bds065@iiitdwd.ac.in

Abstract

A Spotted hyena Optimization (SHO) is nature inspired optimization algorithm which try to mimic the hunting pattern of spotted hyenas. Spotted hyena Optimization (SHO) is a metaheuristic algorithm, which is inspired by social hierarchy and collaborative hunting behavior of spotted hyenas. It can be categorized as a swarm-based algorithm. So searching for prey, encircling the prey, and attack on prey are the 3 basic steps of SHO. It is simple to implement and efficient, and it has been shown to be effective in solving a variety of engineering optimization problems. It is particularly good at finding solutions to problems with a large number of variables. This paper carries out a comprehensive review of SHO, in order to apply it to real life engineering problems. However, it also motivates new researchers and algorithm creators to better understand and utilize this straightforward yet highly effective method for solving problems.

Keywords : Spotted Hyena Optimization Algorithm, Optimization techniques, Meta heuristics.

1. Introduction

In recent times, the increasing complexity of real-world problems has led to a growing demand for enhanced metaheuristic techniques. These methods are invaluable for obtaining optimal solutions in the context of engineering design challenges, gaining popularity due to their efficiency and versatility when compared to conventional classical approaches. Metaheuristic optimization algorithms are a class of population-based stochastic algorithms that are designed to solve complex optimization problems. They have been successfully applied to a wide variety of problems in engineering, science, and business. In recent years, there has been a growing interest in the development of new metaheuristic algorithms inspired by nature. One such algorithm is the Spotted Hyena Optimizer (SHO) (Dhiman et al. 2017).

Metaheuristic approaches can be broadly categorized into three main groups: evolutionary-based, physical-based, and swarm-based methods. Evolutionary-based methods, like Genetic Algorithms (GA) (Mirjalili et al. 2019) and Differential Evolutionary , simulate the process of biological evolution, evolving populations of solutions over generations, commonly applied to optimization problems such as parameter tuning in machine learning. Physical-based methods, such as Simulated Annealing (SA) , model natural phenomena or physical processes, with particles moving through a search space and adjusting their positions based on

experiences, used in optimization and control tasks. Other famous Physical-based methods include Harmony Search. Swarm-based methods, like Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO), draw inspiration from collective behaviors in social or biological systems, where groups of agents collaborate towards a common goal, often employed in solving routing or scheduling problems. Other popular swarm intelligence techniques are, Cuckoo Search (CS) (Yang et al. 2010), Firefly Algorithm (FA) (Yang 2010)etc.

Using a single optimizer to tackle various optimization problems with different characteristics doesn't ensure success. This idea has inspired to develop a novel metaheuristic algorithm called Spotted Hyena Optimization (SHO) [1], which takes its inspiration from the world of biology. The SHO is a new bio-inspired technique that draws its inspiration from the teamwork and hunting strategies observed in spotted hyenas. Similar to how spotted hyenas work together efficiently in cohesive groups, it uses clustering to enhance cooperation among its components. The SHO algorithm's key steps are influenced by the way these animals hunt. In a rigorous evaluation, it's effectiveness is tested on a set of challenges. The outcomes highlight SHO's superior performance when compared to other competing algorithms in the proposed paper. So in this paper we will review the same algorithm and analyze its performance over standard problems to optimize it.

In this paper, we present a review of the Spotted Hyena Optimizer (SHO) algorithm. We discuss the algorithm's motivation, design, implementation, and evaluation. We also discuss the algorithm's limitations and future research directions. We believe that the SHO algorithm is a promising new metaheuristic algorithm with the potential to be applied to a wide variety of optimization problems. We hope that this paper will provide a useful overview of the algorithm and encourage further research in this area. The structure of this paper can be outlined as follows: In Section 2 is dedicated to literature search. Coduct a comparative analysis with existing literature to establish the effectiveness of the SHO approach. Moving on to Section 3, it briefly presents the SHO algorithm and methodology employed. In Section 4, we comprehensively discuss the simulation and experimental results achieved through the implementation of the SHO method. Section 5 concludes the insights and important findings from the analysis.

2. Literature Review

The original work on the SHO algorithm was published in 2017 by Gaurav Dhiman and Vijay Kumar (Dhiman et al. 2017). The paper proposes the algorithm and evaluates its performance on a set of benchmark test functions and five real-life engineering design problems. The results show that the SHO algorithm is competitive with other existing metaheuristic algorithms. Further advancement and use of SHO was explored in the next paper by the same author (Dhiman et al. 2017, 2018) Since the original paper was published, there have been a number of studies that have proposed improvements and extensions to the SHO algorithm. These studies have focused on improving the algorithm's performance, robustness, and scalability.

In the original SHO algorithm, the learning rate is a constant value. However, a number of studies have proposed using an adaptive learning rate, which can improve the algorithm's

performance. For example, the adaptive learning rate that is based on the number of iterations (Ghafori et al. 2021) The SHO algorithm is a population-based algorithm, and the diversity of the population is important for its performance. A number of studies have proposed techniques for improving the diversity of the SHO algorithm. technique proposed in (2020) for introducing diversity into the population by randomly swapping solutions between different hyenas. (Kumar et al. 2020; Dhiman et al. 2019) The study by by Dhiman (Dhiman et al. 2017, 2018), proposed a multi-objective SHO algorithm that uses a Pareto-based selection mechanism. SHO algorithm can be hybridized with other metaheuristic algorithms like Particle Swarm Optimization algorithm to improve its performance. The SHO algorithm has been applied to a variety of optimization problems, including engineering design, scheduling, and financial optimization. The results of these studies have shown that the SHO algorithm is a promising new algorithm for solving a wide range of optimization problems. The SHO algorithm has been applied to machine learning problems, such as the training of neural networks (Ghafori et al. 2021; Ewees et al. 2021) and the tuning of hyperparameters and feature selection (Kumar et al. 2020; Luo et al. 2021). It is used in financial optimization problemslike portfolio optimization problem. The neural network and image processing problems uses SHO for solving problems like image segmentation and image denoising. The SHO algorithm is a relatively new algorithm, and its applications are still being explored. However, the studies that have been conducted so far have shown that the SHO algorithm has the potential to be applied to a wide range of optimization problems.

There is still room for improvement in the performance of the SHO algorithm. Researchers are working on developing new techniques to improve the algorithm's convergence rate and its ability to find the global optimum. The SHO algorithm is still sensitive to the parameter settings. It can be computationally expensive for large-scale problems. The theoretical properties of the SHO algorithm are still not well understood. Researchers are working on developing a better understanding of the algorithm's behavior. The literature survey shows that there is a growing interest in the algorithm, and there is a lot of ongoing research on improving its performance and applications.

3. Methodology

Spotted hyenas are big meat-eating animals that live in dry, open places. They eat animals like zebras, impalas, and wildebeests, which come in different sizes. Spotted hyenas are smart and social creatures. They can recognize their family members using their senses, and they have a social hierarchy. When they hunt together as a group, they are quite successful in catching prey in the wild. Spotted Hyena Optimization (SHO) works a bit like how a group of spotted hyenas hunt together in the wild. Just like hyenas work as a team, SHO uses teamwork to solve problems. It has different steps, just like how hyenas have steps in their hunting. Figure 1 shows the different phases of spotted-hyena hunting behavior, first of all the searching and tracking prey then chasing, troublesome and encircling and lastly immobilizing and attacking the prey. So in implementation first, it looks for possible solutions to the problem, like how hyenas look for prey. Then, it isolates the best solution, similar to how hyenas isolate their prey. Finally, it organizes things neatly, like how hyenas share their food in a specific order. So, SHO is a clever way to solve tough problems by learning from how hyenas hunt in nature.

Spotted hyenas hunt together in groups. They work as a team to find and catch their food. Each hyena keeps track of where the food is and how far away it is. They do this because they need to get closer to the food before they can catch it. At first, they don't know where the food is, so they have to figure it out. Hyenas in the cluster update their distance according to the position of the prey. This distance needs to be reduced before attacking the prey. So for mathematically modelling the above process, it starts with a population of randomly generated solutions. In each iteration, the algorithm (Dhiman et al. 2017, 2018) performs the following steps:

Searching: Each hyena randomly searches for a new solution in the search space.

Encircling: The hyenas form a cluster around the best solution found so far.

Attacking: The hyenas attack the best solution found so far and try to improve it.

So the method they used is the solution to the optimization problem that can be mathematically modeled as

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

$$\overrightarrow{P}(x+1) = \overrightarrow{P_p}(x) \overrightarrow{E} \overrightarrow{Dh} \tag{2}$$

In the above equations, there are few things that help us explore new possibilities and prevent us from getting stuck in one solution. We use two values called B and E for this. Also, there are other values called Pp, P, and Dh, which tell us where the hyena is and how far it is from its prey. We figure out these values using a specific method as follows

$$\overrightarrow{B} = 2.r \overrightarrow{d} \tag{3}$$

$$\overrightarrow{E} = 2. \overrightarrow{h} .r. \overrightarrow{d_2} - \overrightarrow{h} \tag{4}$$

$$\overrightarrow{h} = 5 - (Iteration * (5/Max_i)) \tag{5}$$

To make sure we balance between trying new things and sticking with what works, we gradually decrease the 'k' value from 5 to 0 as we keep trying different things. This happens until we reach the maximum number of times we're allowed to try. We use random numbers, rd1 and rd2, which can be any number between 0 and 1. We use equations (4) and (5) in a two-dimensional way, as shown in the picture below. In this picture, think of the spotted hyena at point (A, B) trying to move closer to where the prey is at point (A*, B*).

We count how many perfect spots there are in the bunch when they go hunting. This count depends on something called the coefficient vector E and is given by

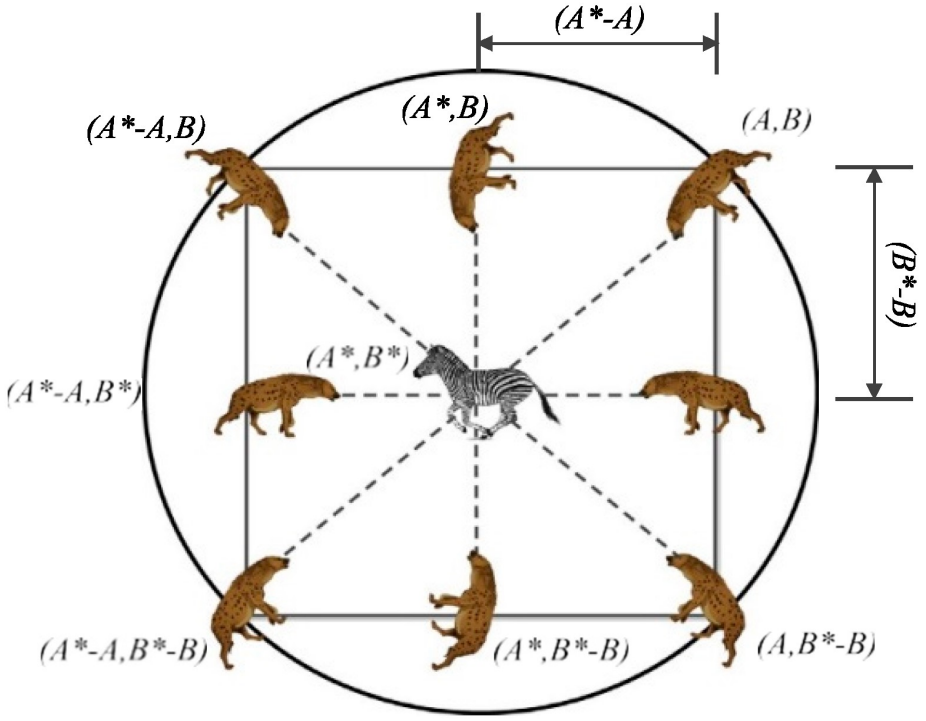

Figure 1: Position vectors of spotted hyena in 2D before attacking the prey

$$\overrightarrow{D}_h = \overrightarrow{B} \bullet \overrightarrow{P}_h - \overrightarrow{P_k} \tag{6}$$

$$\overrightarrow{P}_k = \overrightarrow{P_h} \bullet \overrightarrow{E}.\overrightarrow{D_h} \tag{7}$$

$$\overrightarrow{C}_h = \overrightarrow{P}_k + \overrightarrow{P}_{k+1} + ... + \overrightarrow{P}_{k+N} \tag{8}$$

Here Pk shows where the spotted hyenas are, and "Ph" points to the very best spot where a spotted hyena can be found. The letter "N" tells us how many spotted hyenas there are, and we can count them as follows:

$$N = count_{nos}(\overrightarrow{P}_h + \overrightarrow{P}_{h+1} + ... + (\overrightarrow{P}_h + \overrightarrow{M})) \tag{9}$$

In this situation, we have a vector called 'M' that's randomly set between 0.5 and 1. Equation 8 represents a bunch of the best solutions, like a team of hunters. When these hunters get really close to their target, we use a special number called 'E' to decide if they should attack. If 'E' is less than 1, it means they go in for the kill. This attack on the target is shown in math as follows:

$$\overrightarrow{\overrightarrow{P_{x+1}}} = \overrightarrow{C_h}/N \tag{10}$$

In the formula, P_{x+1} is to figure out where the best solution is and how the other search entities should move based on where the best one is. In the SHO algorithm, we keep track of the best solution and update the positions of other search entities based on where this best solution is. We stop the algorithm when a specific condition is met. So whole process can be understood by following the steps:

1) Initialize Population: Begin by creating a population of spotted hyenas (search agents) labeled as Pi, where i ranges from 1 to n. ested on five constrained and one unconstrained engineering design problems: welded beam, tension/comp

2) Set Initial Parameters: Choose the initial parameters for the SHO algorithm, including h, B, E, and N. also mention no of iterations.

3) Evaluate Fitness: Calculate the fitness value for each search agent to assess their performance.

4) Explore Best Agent: Identify the best-performing search agent within the search space.

5) Define Optimal Group: Define a group of optimal solutions or clusters using Eq 8,9 until good results.

6) Update Agent Positions: Update the positions of the search agents using Equation (10).

7) Boundary Checking: Ensure that no search agent goes beyond the defined boundaries in the search space. Adjust their positions if needed.

8) Update Fitness and Optimal Solution: Calculate and update the fitness values of the search agents. If a better solution than the previous optimal one is found, update the vector Ph.

9) Update Spotted Hyena Group: Update the group of spotted hyenas, Ch, with the updated fitness values of the search agents.

10) Check Stopping Criteria: Check if the algorithm's stopping criterion has been met. If satisfied, stop the algorithm; otherwise, return to Step 5 i.e. Define Optimal Group.

11) Optimal Solution: As after stopping criteria is satisfied, return the best optimal solution whichever obtained so far.

To solve an optimization problem using the Spotted Hyena Optimizer (SHO) algorithm (Dhiman et al. 2017) :

The overall time complexity of the SHO algorithm is O(k × n × Maxiter × dim × N) and the total space complexity is O(n × dim). 'n' determines many times random group is created. This group is based on how many search agents there are and the lowest and highest values allowed in a test problem. 'dim' tells us how many aspects we're checking and fixing in the solutions if they go outside the allowed space. Maxiter is maximum iterations and N is the count of spotted hyenas.

Algorithm 1 Spotted Hyena Optimizer

Input: Population Pi the spotted hyenas (i = 1, 2, . . ., n)
Output: Best hyena

procedure SHO
Initialize parameters h, B, E, and N
Calculate the fitness of each hyena.
Set the P_h = best hyena with lowest fitness value
Initialize the cluster C_h = cluster of far-optimal solutions to be empty.
  while ($x < MaxIterations$) **do**
    for each hyena :
      By using Eq. (10) update position of hyena
    end for
    Update the paramters of SHO
    Boundry check for search Hyenas and adjust it
    Calculate the fiteness for each hyena
    Update Ph for better solution
    Update the cluster C_h by using Ph
    x = x+1
  end while
return Ph
end procedure

4. Results

The SHO algorithm is effective in solving a variety of optimization problems, including engineering design problems, scheduling problems, and financial optimization problems. It is robust to noise and outliers. It is easy to implement and tune. The SHO algorithm is competitive with other metaheuristic algorithms, such as the Particle Swarm Optimization algorithm and the Genetic Algorithm. This section explains how we tested the new algorithm. Proposed paper used 29 standard test problems to see how well SHO algorithm works. These test problems are explained in Section 3. Then, we looked at the results and compared them to eight other commonly used methods to see how good our algorithm is.

4.1 Benchmark test comparison and analysis

The given algorithm was evaluated on total of 29 benchmark functions, which were divided into four categories: unimodal, multimodal, fixed-dimension multimodal, and composite functions. The unimodal functions have one global optimum, while the multimodal functions have multiple local optima. The fixed-dimension multimodal functions are a subset of the multimodal functions that have a fixed number of dimensions. The composite functions are a combination of classical functions.

The proposed algorithm was compared to eight other well-known algorithms: Particle Swarm Optimization (PSO) (Kennedy et al. 1995), Grey Wolf Optimizer (GWO) (Mirjalili et al. 2014),

Sine Cosine Algorithm (SCA) (Mirjalili 2016), etc. So total search agents for each algorithm was given a value 30. The outcomes for functions F1 to F29 indicate that the SHO algorithm performs exceptionally well when compared to other optimization methods. To dig deeper into how good this new algorithm is, we're going to take a closer look at how it tackles six classic engineering design problems in the upcoming sections. Further Convergence analysis and Scalability studies are performed and results was favourable. Statistical testing proves that This algorithm is significant statistically when compared with other algorithms.

4.2 SHO for engineering problems

SHO is a robust, easy to implement algorithm, hence if you are facing an engineering problem that needs to be solved, SHO is a potential algorithm to consider. In this section, the SHO algorithm was tested on five constrained and one unconstrained engineering design problems: welded beam,speed reducer,pressure vessel etc. So different penalty functions to handle constraint problems. Various standard engineering problems like Tension/compression spring design problem, Pressure vessel design, Speed reducer design problem, Rolling element bearing design problem, Displacement of loaded structure are optimized with SHO and mostly it outperformed other algorithms. The results showed that the proposed algorithm outperformed the other algorithms on all four categories of benchmark functions. The proposed algorithm was able to find the global optimum of the unimodal functions in fewer iterations than the other algorithms. The proposed algorithm was also able to find the global optimum of the multimodal functions, even when the functions had multiple local optima. (Dhiman et al. 2017, 2018, 2019) We have implementated the SHO for optimizing two types of problems, the first one is minimizing the value of the given function and second one is to parameter tunning for machine learning classification models.

4.2.1 Minimizing the cost of the objective function:

We have taken a following function f(x) and applied SHO algorithm to find the smalled value and position of x in the search space.

$$f(x) = cos(3x) + 3.sin(3x)) + x^2 \tag{11}$$

The SHO algorithm worked well and converges to find the optimal solution. We also tried to find the optimized answer with the help of Particle Swarm Optimization Algorithm (PSO). The figure 2 and figure 3 shows the performaces of both the algorithms. Figure 2 showcases the fiteness of the standard objective function named F292017 (Nguyen 2020) to be minimize whereas Figure 3 shows the results of optimization of Eq. (11). As plot from figure 3 suggest that for easy linear problem SHO is as good as PSO as both converges quickly. But Figure 2 shows that when complexity of problem increases SHO converges faster than PSO to the best optimal solution. So SHO works well in this type of problems and outperforms other algorithms.

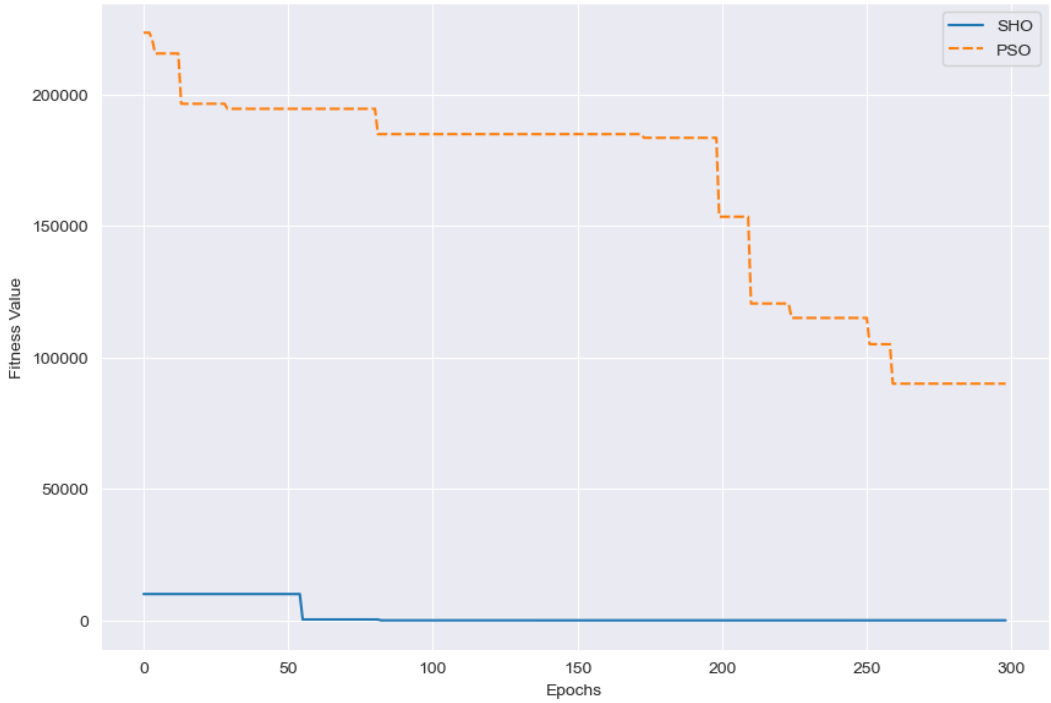

Figure 2: Results of optimization of F292017 data instance

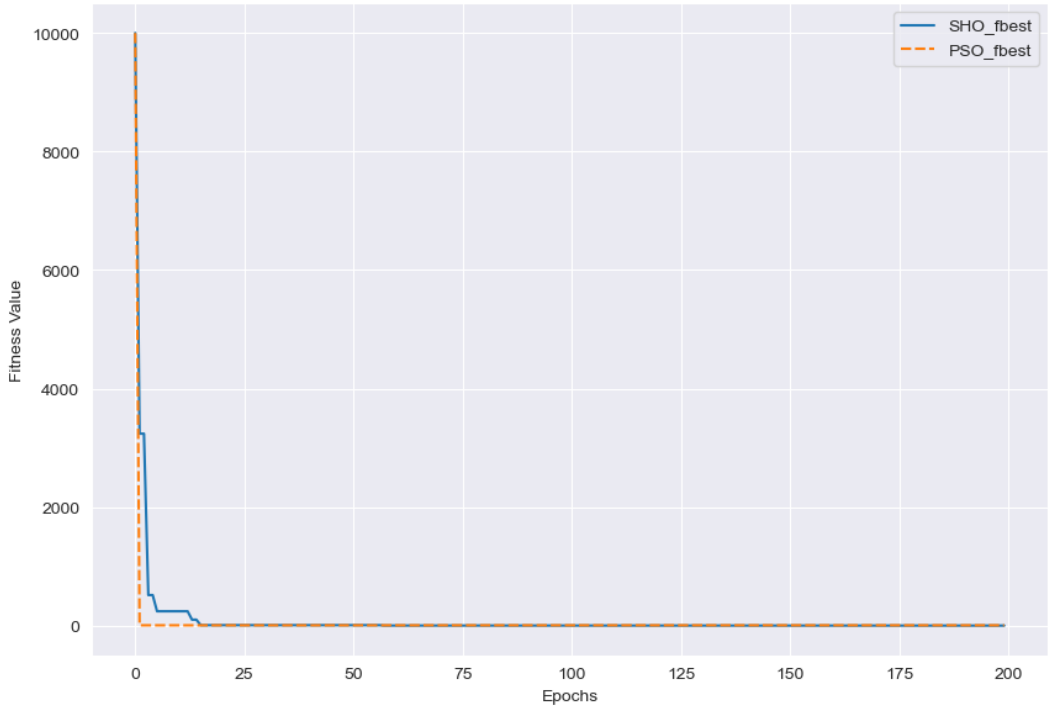

Figure 3: Results of optimization of self define function

4.2.2 Hyperparameter tunning with SHO:

We used SHO algorithm to find the best combination of the hyperparameter of the classifiers. For that sake we take many datasets from small size (less than 200 data points) to higher size (More than 1 lakh datapoints). We used two different classification models as follows: KNN and LGBM CLassifier. We have analyse the performance of the alogorithms given by the log_loss of the classifier. In this case PSO seems to be more optimal than the SHO algorithm. We tried to increase the complexity of the problem for which PSO works far well than SHO. Hence we can say that for smaller datasets SHO can work well for finding the right combination of the hyperparametrs, but as the complexity increases its getting hard towards getting optimized. This may change in respect of the data dependency of classification model, data size, number of iterations etc. . So results suggest that hyperparameter parameter tunning can done by SHO but other algorithm can outperform SHO for the same task for complex problems.

5. Conclusion

This paper presented a Spotted Hyena Optimizer (SHO) in a simple terms. The Spotted Hyena Optimizer (SHO) algorithm is a novel metaheuristic algorithm that is inspired by the hunting behavior of spotted hyenas. The SHO algorithm has been shown to be effective in solving a variety of optimization problems, including minimizing the value of a given function and parameter tuning for machine learning classification models. The results of the study showed that the SHO algorithm was able to find the optimal solution to the function minimization problem in a relatively short amount of time. The SHO algorithm also outperformed the Particle Swarm Optimization (PSO) algorithm on this problem. The results of the study also showed that the SHO algorithm was able to find the optimal hyperparameters for machine learning classification models. However, the PSO algorithm was able to find better hyperparameters for some datasets. This may be because the PSO algorithm is more sensitive to the hyperparameters of the classification model. Overall, the results of the study suggest that the SHO algorithm is a promising new metaheuristic algorithm that can be used to solve a variety of optimization problems. The SHO algorithm is more efficient than other metaheuristic algorithms such as the PSO algorithm for function minimization problems. However, the PSO algorithm may be more effective for parameter tuning for machine learning classification models. We believe that the SHO algorithm is a significant contribution to the field of metaheuristic optimization. We hope that this paper will help to promote the use of the algorithm and encourage further research in this area.

References

Dhiman, Gaurav, and Vijay Kumar. 2017. "Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications." *Advances in Engineering Software* 114:48–70. https://www.sciencedirect.com/science/article/abs/pii/S0965997816305567.

Dhiman, Gaurav, and Vijay Kumar. 2018. "Multi-objective spotted hyena optimizer: a multi-objective optimization algorithm for engineering problems." *Knowledge-Based Systems* 150:175–197.

Dhiman, Gaurav, and Vijay Kumar. 2019. "Spotted hyena optimizer for solving complex and non-linear constrained engineering problems." In *Harmony Search and Nature Inspired Optimization Algorithms: Theory and Applications, ICHSA 2018,* 857–867. Springer.

Ewees, Ahmed A, Mohamed Abd Elaziz, and Diego Oliva. 2021. "A new multi-objective optimization algorithm combined with opposition-based learning." *Expert Systems with Applications* 165:113844.

Ghafori, Shafih, and Farhad Soleimanian Gharehchopogh. 2021. "Advances in spotted hyena optimizer: a comprehensive survey." *Archives of computational methods in engineering,* 1–22.

Kennedy, James, and Russell Eberhart. 1995. "Particle swarm optimization." In *Proceedings of ICNN'95-international conference on neural networks,* 4:1942–1948. IEEE.

Kumar, Vijay, and Avneet Kaur. 2020. "Binary spotted hyena optimizer and its application to feature selection." *Journal of Ambient Intelligence and Humanized Computing* 11:2625–2645.

Luo, Qifang, Jie Li, Yongquan Zhou, and Ling Liao. 2021. "Using spotted hyena optimizer for training feedforward neural networks." *Cognitive Systems Research* 65:1–16.

Mirjalili, Seyedali. 2016. "SCA: a sine cosine algorithm for solving optimization problems." *Knowledge-based systems* 96:120–133.

Mirjalili, Seyedali, Seyed Mohammad Mirjalili, and Andrew Lewis. 2014. "Grey wolf optimizer." *Advances in engineering software* 69:46–61.

Mirjalili, Seyedali, and Seyedali Mirjalili. 2019. "Genetic algorithm." *Evolutionary Algorithms and Neural Networks: Theory and Applications,* 43–55.

Nguyen, Thieu. 2020. "A framework of Optimization Functions using Numpy (OpFuNu) for optimization problems," https://doi.org/10.5281/zenodo.3620960. https://doi.org/10.5281/zenodo.3620960..

Yang, Xin-She. 2010. "Firefly algorithm, stochastic test functions and design optimisation." *International journal of bio-inspired computation* 2 (2): 78–84.

Yang, Xin-She, and Suash Deb. 2010. “Engineering optimisation by cuckoo search.” *International Journal of Mathematical Modelling and Numerical Optimisation* 1 (4): 330–343.