Optimizing the Design of Airfoil and Optical Buffer Problems Using Spotted Hyena Optimizer

Gaurav Dhiman *† and Amandeep Kaur †

Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, Patiala, Punjab 147004, India; kaur.amandeep@thapar.edu
* Correspondence: gaurav.dhiman@thapar.edu; Tel.: +91-828-884-1068
† These authors contributed equally to this work.

Received: 27 April 2018; Accepted: 27 July 2018; Published: 1 August 2018

Abstract: This paper presents the contemporary metaheuristic optimization algorithm named the Spotted Hyena Optimizer (SHO). The proposed technique is based on the law of gravitation and simulates the social behavior of spotted hyenas. The three basic steps of SHO, namely, searching for prey, encircling, and attacking prey, are mathematically modelled and discussed. The main concept of this work is to apply the recently developed SHO algorithm to two real-life design problems, namely optical buffer design and airfoil design. Experimental results reveal the supremacy of the SHO algorithm for solving the engineering design problems as compared to other competitor algorithms.

Keywords: metaheuristics; constrained optimization; engineering design problems; SHO

1. Introduction

Optimization is the process of determining the decision variables of a function to minimize or maximize its values. Most of the real world problems [1–3] include nonlinear constraints, non-convex, complicated, and a large number of solution spaces. Therefore, solving such problems with a large number of variables and constraints is very tedious and complex. There are many local optimum solutions that do not guarantee the best overall solution using classical numerical methods.

To overcome such problems, metaheuristic optimization algorithms have been introduced which are capable of solving such complex problems during the course of iterations. Recently, immense interest has been focused on the development of metaheuristic algorithms owing to their flexibility and simplicity by nature.

Metaheuristics are broadly classified into two categories [4] such as single solution and population based algorithms. Single solution based algorithms are those in which a solution is randomly generated and improved until the optimum result is obtained, whereas population based algorithms are those in which a set of solutions are randomly generated in a given search space and solution values are updated during iterations until the best solution is generated.

However, single solution based algorithms may trap into local optima which may prevent us to find global optimum as it reforms only one solution, which is randomly generated for a given problem. On the other hand, population based algorithms have an inherent ability to escape local optima [5]. Due to this, nowadays, population based algorithms have gained the attention of multitudinous researchers.

The categorization of population based algorithms is done on the basis of theory of evolutionary algorithms, physics laws based algorithms, swarm intelligence of particles, and biological behavior of bio-inspired algorithms (see Figure 1). Evolutionary algorithms are inspired by the evolutionary processes such as reproduction, mutation, recombination, and selection. These algorithms are based on the survival fitness of candidate in a population (i.e., a set of solutions) for a given environment. The physics law based algorithms are inspired by physical processes according to some physics rules.
such as gravitational force, electromagnetic force, inertia force, heating and cooling of materials. Swarm intelligence based algorithms are inspired by the collective intelligence of swarms.

Some of the most popular evolutionary algorithms are Genetic Algorithms (GA) [6], Evolution Strategy (ES) [7], Differential Evolution (DE) [8], and Biogeography-Based Optimizer (BBO) [9].

A well-known algorithm of swarm intelligence technique is Particle Swarm Optimization (PSO) [10,11]. PSO is inspired by the social behavior of fish schooling or bird flocking. Each particle can move around the search space and update its current position with respect to the global best solution. However, Table 1 shows the other popular optimization based techniques.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simulated Annealing [12]</td>
<td>SA</td>
</tr>
<tr>
<td>Gravitational Search Algorithm [13]</td>
<td>GSA</td>
</tr>
<tr>
<td>Charged System Search [14]</td>
<td>CSS</td>
</tr>
<tr>
<td>Black Hole Algorithm [15]</td>
<td>BH</td>
</tr>
<tr>
<td>Emperor Penguin Optimizer [16]</td>
<td>EPO</td>
</tr>
<tr>
<td>Artificial Chemical Reaction Optimization Algorithm [17]</td>
<td>ACROA</td>
</tr>
<tr>
<td>Ray Optimization Algorithm [18]</td>
<td>RO</td>
</tr>
<tr>
<td>Galaxy-Based Search Algorithm [19]</td>
<td>GbSA</td>
</tr>
<tr>
<td>Ant Colony Optimization [20]</td>
<td>ACO</td>
</tr>
<tr>
<td>Cuckoo Search [21]</td>
<td>CS</td>
</tr>
<tr>
<td>Bat-Inspired Algorithm [22]</td>
<td>BA</td>
</tr>
<tr>
<td>Firefly Algorithm [23]</td>
<td>FA</td>
</tr>
<tr>
<td>Spotted Hyena Optimizer [24]</td>
<td>SHO</td>
</tr>
<tr>
<td>Exchange Market Algorithm [25]</td>
<td>EMA</td>
</tr>
<tr>
<td>Social-Based Algorithm [26]</td>
<td>SBA</td>
</tr>
<tr>
<td>Harmony Search [27]</td>
<td>HS</td>
</tr>
<tr>
<td>Grey Wolf Optimizer [28]</td>
<td>GWO</td>
</tr>
<tr>
<td>Mine Blast Algorithm [29]</td>
<td>MBA</td>
</tr>
</tbody>
</table>

Every optimization algorithm needs to address the exploration and exploitation of a search space [30] and maintains a good balance between exploration and exploitation. The exploration phase investigates the different promising regions in a search space, whereas exploitation searches the close global optimal solutions around the promising regions [31,32]. Therefore, to acquire the close optimal solutions, fine-tuning of these two phases is required. Despite the significant number of recently developed optimization algorithms, the question that arises is why do we need to develop more optimization techniques. The answer lies in a No Free Lunch (NFL) theorem [33]. According to this theorem, the performance of one optimization algorithm for a specific set of problems does not guarantee solving other optimization problems because of their different nature. The NFL theorem allows researchers to propose some novel optimization algorithms for solving the problems in various fields [34–36].

This paper presents the recently developed bio-inspired metaheuristic algorithm named the Spotted Hyena Optimizer (SHO) for optimizing constrained problems. As its name implies, SHO mimics the social behaviors of spotted hyenas in nature. The performance of the SHO algorithm is evaluated on designs of optical buffer and airfoil problems. The results reveal that the performance of SHO is more competitive than the existing algorithms.

The rest of this paper is structured as follows: Section 2 presents the fundamental concepts of a recently developed optimization algorithm. Section 3 presents the constrained handling approach. The two real-life constrained industrial optimization problems and their comparison are presented in Sections 4 and 5, respectively. Finally, the conclusions are discussed in Section 6.
2. Spotted Hyena Optimizer (SHO)

Spotted Hyena Optimizer is a metaheuristic bio-inspired optimization algorithm developed by Dhiman et al. [24,37,38]. The fundamental concept of this algorithm is to simulate the social behaviors of spotted hyenas. There are four main steps of the SHO algorithm that are inspired by encircling, hunting, attacking and searching behaviors of spotted hyenas.

2.1. Encircling Prey

The target prey or objective is considered as the best solution and the other search agents can update their positions with respect to the best solution obtained. The mathematical model of this behavior is represented by Equations (1) and (2):

\[ \vec{D}_h = | \vec{B} \times \vec{P}_p(x) - \vec{P}(x) |, \quad (1) \]
\[ \vec{P}(x+1) = \vec{P}_p(x) - \vec{E} \times \vec{D}_h, \quad (2) \]

where \( \vec{D}_h \) represents the distance between the prey and spotted hyena, \( x \) indicates the current iteration, \( \vec{B} \) and \( \vec{E} \) are co-efficients vectors, \( \vec{P}_p \) indicates the position vector of prey, and \( \vec{P} \) is the position vector of the spotted hyena.

However, the vectors \( \vec{B} \) and \( \vec{E} \) are calculated by Equations (3)–(5), respectively:

\[ \vec{B} = 2 \times r \vec{d}_1, \quad (3) \]
\[ \vec{E} = 2 \vec{s} \times r \vec{d}_2 - \vec{h}, \quad (4) \]
\[ \vec{s} = 5 - \left( \text{Iteration} \times \frac{5}{\text{MaxIteration}} \right), \quad (5) \]

where, \( \text{Iteration} = 0, 1, 2, \ldots, \text{MaxIteration} \).

The \( \vec{s} \) is linearly decreased from 5 to 0 and \( r \vec{d}_1, r \vec{d}_2 \) are random vectors in range \([0, 1]\).

2.2. Hunting

The hunting strategy of the SHO algorithm is described by Equations (6)–(8):

\[ \vec{D}_h = | \vec{B} \times \vec{P}_h - \vec{P}_k |, \quad (6) \]
\vec{P}_k = \vec{P}_h - \vec{E} \times \vec{D}_h, \quad (7)
\vec{C}_h = \vec{P}_k + \vec{P}_{k+1} + \ldots + \vec{P}_{k+N}, \quad (8)

where \( \vec{P}_h \) defines the position of first best obtained spotted hyena, and \( \vec{P}_k \) represents the position of other spotted hyenas. However, variable \( N \) indicates the total number of spotted hyenas, which is calculated by Equation (9):

\[ N = \text{count}_{\text{nos}}(\vec{P}_h, \vec{P}_{h+1}, \vec{P}_{h+2}, \ldots, (\vec{P}_h + \vec{M})), \quad (9) \]

where \( \vec{M} \) is a random vector in range \([0.5, 1]\), \( \text{nos} \) represents the number of solutions and count all candidate solutions, and \( \vec{C}_h \) is a group of \( N \) number of optimal solutions.

2.3. Attacking Prey

The mathematical formulation for attacking the prey is defined by Equation (10):

\[ \vec{P}(x + 1) = \frac{\vec{C}_h}{N}, \quad (10) \]

where \( \vec{P}(x + 1) \) saves the best solution and updates the positions of other search agents with respect to the position of the best search agent.

2.4. Search for Prey

For searching the suitable solution, \( \vec{E} \) is responsible which is greater than 1 or less than 1 using Equation (4). The \( \vec{B} \) is an another important constituent of SHO algorithm for exploration purposes. It contains random values that provide the random weights of prey as shown in Equation (3). To show the more random behavior of the SHO algorithm, assume vector \( \vec{B} > 1 \) precedence than \( \vec{B} < 1 \) to demonstrate the effect in the distance.

The SHO algorithm solves various high dimensional problems with low computational efforts and avoids the local optimum problem. The pseudo code of the SHO algorithm is described in the Algorithm.

---

**Algorithm**  Spotted Hyena Optimizer

**Input:** the spotted hyenas population \( P_i \) \((i = 1, 2, \ldots, n)\)

**Output:** the best search agent

1: procedure SHO
2: Initialize the parameters \( h, B, E, \) and \( N \)
3: Calculate the fitness of each search agent
4: \( \vec{P}_h \)= the best search agent
5: \( \vec{C}_h \)= the group or cluster of all far optimal solutions
6: while \((x < \text{Max iteration})\) do
7: for each search agent do
8: Update the position of current agent by Equation (10)
9: end for
10: Update \( h, B, E, \) and \( N \)
11: Check if any search agent goes beyond the given search space and then adjust it
12: Calculate the fitness of each search agent
13: Update \( \vec{P}_h \) if there is a better solution than previous optimal solution
14: Update the group \( \vec{C}_h \) with respect to \( \vec{P}_h \)
15: \( x = x + 1 \)
16: end while
17: return \( \vec{P}_h \)
18: end procedure
2.5. Analysis of the SHO Algorithm

The convergence analysis of metaheuristic algorithm is another feature for better understanding of explorative and exploitative mechanisms. In order to demonstrate the convergence analysis of SHO algorithm, three metrics are employed in the 2D environment that are shown in Figure 2. The employed metrics are discussed as follows:

- **Search history** shows the location history of spotted hyenas during optimization.
- **Trajectory** shows the value of the first variable in each iteration. The trajectory curves show that the spotted hyenas exhibit large and abrupt changes in the initial steps of optimization. According to Berg et al. [39], this behavior can guarantee that a Swarm-based method eventually converges to a point in the search space.
- **Average fitness** indicates the average objective value of all spotted hyenas in each iteration. The curves show descending behavior on all of the test functions. This proves that the SHO algorithm improves the accuracy of the approximated optimum during simulation runs.

Therefore, the success rate of SHO algorithm is computationally high for solving optimization problems.

![Figure 2. Search history, trajectory, and average fitness of the spotted hyena optimizer (SHO) algorithm on 2D benchmark test problems.](image)

3. Constraint Handling

Constraint handling is one of the biggest challenges in solving optimization problems using metaheuristic techniques. There are five constraint handling techniques [40]: penalty functions, hybrid methods, separation of objective functions and constraints, repair algorithms, and special operators. Among these techniques, the penalty functions are simple and easy to implement. There are numerous penalty functions such as static, annealing, adaptive, co-evolutionary, and death penalty. These approaches convert constraint problems into unconstraint problems by adding some penalty values. In this paper, a static penalty approach is employed to handle constraints in optimization problems:

$$\zeta(z) = f(z) \pm \left[ \sum_{i=1}^{m} l_i \times max(0, t_i(z))^a + \sum_{j=1}^{n} o_j \times |U_j(z) |^\beta \right],$$  \hspace{1cm} (11)
where $\zeta(z)$ is the modified objective function, $l_i$ and $o_j$ are positive penalty values, $t_i(z)$ and $U_j(z)$ are constraint functions, and $l_i$ and $o_j$ are positive constants. The values of $\alpha$ and $\beta$ are 1 and 2, respectively. This approach assigns the penalty value for each infeasible solution. In the death penalty approach, a large value is assigned to the objective function of infeasible solution. Therefore, the static penalty function is employed which helps the search agents to move towards the feasible search space of the problem.

**Experimental Setup**

The parameter settings of metaheuristic algorithms are tabulated in Table 2. The parameter values of these algorithms are set as they are recommended in their original papers. The experimentation has been done with the Matlab R2014a (8.3.0.532) version in the environment of Microsoft Windows 8.1 using 64 bit Core i-5 processor with 2.40 GHz and 4 GB main memory.

<table>
<thead>
<tr>
<th>#</th>
<th>Algorithms</th>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Spotted Hyena Optimizer (SHO)</td>
<td>Search Agents</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control Parameter ($h_i$)</td>
<td>[5, 0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$M$ Constant</td>
<td>[0.5, 1]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of Generations</td>
<td>1000</td>
</tr>
<tr>
<td>2</td>
<td>Grey Wolf Optimizer (GWO)</td>
<td>Search Agents</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Control Parameter ($a_i$)</td>
<td>[2, 0]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of Generations</td>
<td>1000</td>
</tr>
<tr>
<td>3</td>
<td>Particle Swarm Optimization (PSO)</td>
<td>Number of Particles</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Inertia Coefficient</td>
<td>0.75</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cognitive and Social Coeff</td>
<td>1.8, 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of Generations</td>
<td>1000</td>
</tr>
<tr>
<td>4</td>
<td>Genetic Algorithm (GA)</td>
<td>Population Size</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Crossover</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Mutation</td>
<td>0.05</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Number of Generations</td>
<td>1000</td>
</tr>
<tr>
<td>5</td>
<td>Differential Evolution (DE)</td>
<td>Population Size</td>
<td>80</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Crossover</td>
<td>0.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Scale Factor ($F$)</td>
<td>0.5</td>
</tr>
</tbody>
</table>

**4. Optical Buffer Design Problem**

The optical buffer permits the optical CPUs to measure different optical packets by slowing down the group velocity of light. This whole process is executed using the most popular device known as the Photonic Crystal Waveguide (PCW). Generally, PCWs have a lattice-shaped structure with a line defect and holes with different radii that yield the characteristics of slow light. In this subsection, the structure of PCW called a Bragg Slot Photonic Crystal Waveguide (BSPCW) is optimized to achieve these characteristics by the SHO algorithm.

The performance of slow light devices is compared using Delay Bandwidth Product (DBP) and Normalized DBP (NDBP) metrics that are formulated by Equation (12) [41]:

$$DBP = \Delta d \times \Delta b,$$

where $\Delta d$ and $\Delta b$ indicate the delay and bandwidth of slow light device, respectively,

$$NDBP = \overline{m_g} \times \Delta \omega / \omega_0,$$

where $\overline{m_g}$ is the average of group index, $\Delta \omega$ is the bandwidth, and $\omega_0$ is the central frequency of light wave. However, NDBP has a relation with group index ($m_g$) as:
Designs 2018, 2, 28

\[ m_g = \frac{V}{\varepsilon_g} = C \frac{dv}{d\omega} \]  

(14)

where \( \omega \) is the dispersion, \( v \) defines the wave vector, \( C \) indicates the velocity of light, and \( m_g \) is responsible for changing in the bandwidth range. The average of \( m_g \) is calculated as follows:

\[ m_g = \int_{\omega_L}^{\omega_H} m_g(\omega) \frac{d\omega}{\Delta \omega} \]  

(15)

since \( m_g \) has a constant value with maximum fluctuation of \( \pm 10\% \) [42]. The detailed information about PCWs can be found in [43]. The mathematical formulation of this problem is described in Equation (16):

\[ \mathbf{z} = [z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8] = \left[ \begin{array}{cccccccc} R_1 & R_2 & R_3 & R_4 & R_5 & l & w_L & w_H \end{array} \right], \]

Maximize:  \( f(\mathbf{z}) = \text{NDBP} \),

subject to:

\[ \text{max}(|\beta_2(\omega)|) < 10^6 a / 2\pi c^2, \]

\[ \omega_H < \min(\omega_{\text{upband}}), \]

\[ \omega_L > \max(\omega_{\text{lowband}}), \]

\[ k_n > k_n^{\text{guidedmode}} > \omega_H, \]

\[ k_n < k_n^{\text{low}} = \omega^{\text{guidedmode}} < \omega_L, \]

where,

\[ \omega_H = \omega(k_n^{\text{high}}) = \omega(1.1m_0), \]

\[ v\omega_L = \omega(k_n^{\text{low}}) = \omega(0.9m_0), \]

\[ vk_n = \frac{k a}{2\pi} \Delta \omega = \omega_H - \omega_L, a = \omega_0 \times 1550\text{nm}, \]

\[ 0 \leq z_1 \leq 0.5, \quad 0 \leq z_6 \leq 1, \quad 0 \leq z_7, z_8 \leq 1. \]

(16)

There are five constraints defined in this problem for the SHO algorithm. The algorithm is iterated for 30 times and the obtained results are tabulated in Table 3. The results reveal the substantial improvements of 99% and 10% in bandwidth using the SHO approach in comparison to the results reported by Wu et al. [44] and GWO [28], respectively. The similar behavior has been observed in the NDBP. The improvements achieved in NDBP are 90% and 14% as compared with Wu et al. [44] and GWO [28] approaches, respectively. The optimized super cell is shown in Figure 3. It shows that the optimized structure has a very good bandwidth without band mixing. The results demonstrate that the SHO algorithm proved its merit for solving the optical buffer optimization problem.

![Figure 3. Optimized super cell using the SHO algorithm.](image-url)
Table 3. Comparison results for the optical buffer design problem.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>SHO</th>
<th>GWO [28]</th>
<th>Wu et al. [44]</th>
</tr>
</thead>
<tbody>
<tr>
<td>R_1</td>
<td>0.31349a</td>
<td>0.33235a</td>
<td>-</td>
</tr>
<tr>
<td>R_2</td>
<td>0.22378a</td>
<td>0.24952a</td>
<td>-</td>
</tr>
<tr>
<td>R_3</td>
<td>0.23400a</td>
<td>0.26837a</td>
<td>-</td>
</tr>
<tr>
<td>R_4</td>
<td>0.27681a</td>
<td>0.29498a</td>
<td>-</td>
</tr>
<tr>
<td>R_5</td>
<td>0.30217a</td>
<td>0.34992a</td>
<td>-</td>
</tr>
<tr>
<td>l</td>
<td>0.7263a</td>
<td>0.7437a</td>
<td>-</td>
</tr>
<tr>
<td>w_b</td>
<td>0.2052a</td>
<td>0.2014a</td>
<td>-</td>
</tr>
<tr>
<td>w_l</td>
<td>0.60027a</td>
<td>0.60073a</td>
<td>-</td>
</tr>
<tr>
<td>a (nm)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bandwidth</td>
<td>328</td>
<td>343</td>
<td>430</td>
</tr>
<tr>
<td>( \beta_2 ) ((a/2\pi c^2))</td>
<td>10^3</td>
<td>10^3</td>
<td>10^3</td>
</tr>
<tr>
<td>NDBP</td>
<td>0.49</td>
<td>0.43</td>
<td>0.26</td>
</tr>
</tbody>
</table>

5. Airfoil Design Problem

There are two objectives in the airfoil design problem: lift and drag. It has been observed that lifting causes a plane to fly, whereas drag decreases the speed of a plane. Both of these objectives are very important on different occasions. In this section, only the drag is considered to minimize the force and consequently defines the best shape of the wing. The B-spline is utilized to define the shape of an airfoil as shown in Figure 4. There are eight controlling parameters along with x-axis and y-axis directions. The problem formulation of airfoil design is defined by Equation (17):

Minimize: \[ F(x, y) = E_d(x, y), \]
Subject to: \[-1 \leq x, y \leq 1, \text{ and set of } SC, \]

where \( x = \{x_1, x_2, \ldots, x_7\} \), \( y = \{y_1, y_2, \ldots, y_7\} \), \( E_d \) is the drag, and set of many constraints \( SC \) which includes maximum thickness, minimum thickness, and so on. The penalty function is utilized, which is proportional to the level of violation.

\[ F(x, y) = F(x, y) + pe \sum_{j=1}^{3} P_j, \]

where \( pe \) is a constant and \( P_j \) defines the violation size on the \( j \)th constraint in the set \( SC \). After performing 1000 iterations and 30 independent runs, the best results and convergence curves are obtained as shown in Figures 5 and 6, respectively. Figure 6 demonstrates that the SHO algorithm performs better than other competitor approaches and improves the airfoil shape to minimize the drag. The standard deviation of this problem obtained by various approaches is shown in Figure 7. Table 4 shows the parameters value for the airfoil design problem. The optimum drag force value of SHO as well as other approaches is shown in Table 5. The results show that the SHO algorithm is able to minimize the drag force value using low computational efforts.

Table 4. Parameter values of the airfoil design problem.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Foil co-ordinates</td>
<td>NACA0012</td>
</tr>
<tr>
<td>Reynolds number</td>
<td>10^6</td>
</tr>
<tr>
<td>Mach number</td>
<td>0.2</td>
</tr>
<tr>
<td>Co-efficient of Lift</td>
<td>0</td>
</tr>
</tbody>
</table>
Table 5. Optimum drag force values using SHO and other competitor approaches.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Force</th>
</tr>
</thead>
<tbody>
<tr>
<td>SHO</td>
<td>0.0068</td>
</tr>
<tr>
<td>PSO</td>
<td>0.0021</td>
</tr>
<tr>
<td>GA</td>
<td>0.0067</td>
</tr>
<tr>
<td>DE</td>
<td>0.0020</td>
</tr>
</tbody>
</table>

Figure 4. B-spline for the airfoil design problem.

Figure 5. Optimized airfoil design using (a) SHO algorithm; (b) PSO algorithm.

Figure 6. Convergence curves of airfoil design using SHO and other competitive approaches.
Figure 7. The standard deviation of the proposed as well as competitor algorithms for the airfoil design problem.

6. Conclusions

This paper discussed the contemporary Spotted Hyena Optimizer (SHO) bio-inspired optimization algorithm. The main concept of the SHO algorithm is to analyze the social hierarchy and hunting behaviors of spotted hyenas. Furthermore, the SHO algorithm is employed on two real-life constrained engineering design problems such as optical buffer design and airfoil design problems. The obtained results are compared with other competitor algorithms. The results of engineering design problems reveal that the SHO algorithm is an efficient optimizer to solve these problems and generate the near optimal designs.

Author Contributions: G.D. has performed the whole experimentation and justified the research problem. A.K. surveyed the updated state-of-the-art of the area and wrote the paper.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare that they have no conflict of interest.

Appendix A. Unimodal, Multimodal, and Fixed-Dimension Multimodal Benchmark Test Functions

Appendix A.1. Unimodal Benchmark Test Functions

Appendix A.1.1. Sphere Model

\[ F_1(z) = \sum_{i=1}^{30} z_i^2 \]

\[-100 \leq z_i \leq 100, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.1.2. Schwefel’s Problem 2.22

\[ F_2(z) = \sum_{i=1}^{30} |z_i| + \prod_{i=1}^{30} |z_i| \]

\[-10 \leq z_i \leq 10, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.1.3. Schwefel’s Problem 1.2

\[ F_3(z) = \sum_{i=1}^{30} \left( \sum_{j=1}^{i} z_j \right)^2 \]

\[-100 \leq z_i \leq 100, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]
Appendix A.1.4. Schwefel’s Problem 2.21

\[ F_4(z) = \max_i \{|z_i|, 1 \leq i \leq 30\}, \]

\(-100 \leq z_i \leq 100, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.1.5. Generalized Rosenbrock’s Function

\[ F_5(z) = \sum_{i=1}^{30} \left[100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2\right] \]

\(-30 \leq z_i \leq 30, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.1.6. Step Function

\[ F_6(z) = \sum_{i=1}^{30} \left(\lfloor z_i + 0.5 \rfloor\right)^2 \]

\(-100 \leq z_i \leq 100, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.1.7. Quartic Function

\[ F_7(z) = \sum_{i=1}^{30} iz_i^4 + \text{random}[0, 1] \]

\(-1.28 \leq z_i \leq 1.28, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.2. Multimodal Benchmark Test Functions

Appendix A.2.1. Generalized Schwefel’s Problem 2.26

\[ F_8(z) = \sum_{i=1}^{30} -z_i \sin(\sqrt{|z_i|}) \]

\(-500 \leq z_i \leq 500, \quad f_{\text{min}} = -12569.5, \quad \text{Dim} = 30,\]

Appendix A.2.2. Generalized Rastrigin’s Function

\[ F_9(z) = \sum_{i=1}^{30} [z_i^2 - 10\cos(2\pi z_i) + 10] \]

\(-5.12 \leq z_i \leq 5.12, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.2.3. Ackley’s Function

\[ F_{10}(z) = -20\exp\left(-0.2\sqrt{\frac{1}{30} \sum_{i=1}^{30} z_i^2}\right) - \exp\left(\frac{1}{30} \sum_{i=1}^{30} \cos(2\pi z_i)\right) + 20 + e \]

\(-32 \leq z_i \leq 32, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]

Appendix A.2.4. Generalized Griewank Function

\[ F_{11}(z) = \frac{1}{4000} \sum_{i=1}^{30} z_i^2 - \prod_{i=1}^{30} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 \]

\(-600 \leq z_i \leq 600, \quad f_{\text{min}} = 0, \quad \text{Dim} = 30,\]
Appendix A.2.5. Generalized Penalized Functions

\[ F_{12}(z) = \frac{\pi}{30} \left( 10 \sin(\pi x_1) + \sum_{i=1}^{29} (x_i - 1)^2 \right) + \left( x_n - 1 \right)^2 + \sum_{i=1}^{30} u(z_i, 10, 100, 4) \]
\[-50 \leq z_i \leq 50, \quad f_{\min} = 0, \quad \text{Dim} = 30, \]

\[ F_{13}(z) = 0.1 \left( \sin^2(3\pi z_1) + \sum_{i=1}^{29} (z_i - 1)^2 \right) + \left( z_n - 1 \right)^2 + \sum_{i=1}^{N} u(z_i, 5, 100, 4) \]
\[-50 \leq z_i \leq 50, \quad f_{\min} = 0, \quad \text{Dim} = 30, \]

where \( x_i = 1 + \frac{z_i + 1}{4} \),

\[ u(z_i, a, k, m) = \begin{cases} 
    k(z_i - a)^m, & z_i > a, \\
    0, & -a < z_i < a, \\
    k(-z_i - a)^m, & z_i < -a. 
\end{cases} \]

Appendix A.3. Fixed-Dimension Multimodal Benchmark Test Functions

Appendix A.3.1. Shekel’s Foxholes’ Function

\[ F_{14}(z) = \left( \frac{1}{500} + \frac{25}{j} \sum_{i=1}^{2} \left( \frac{1}{2} \sum_{i=1}^{m} (z_i - a_{ij})^6 \right) \right)^{-1} \]
\[-65.536 \leq z_i \leq 65.536, \quad f_{\min} \approx 1, \quad \text{Dim} = 2. \]

| \( (a_{ij}, i = 1, 2 \text{ and } j = 1, 2, \ldots, 25) \) |
|-----|-----|-----|-----|-----|-----|-----|-----|
| \( i \backslash j \) | 1   | 2   | 3   | 4   | 5   | 6   | 25  |
| 1   | -32 | -16 | 0   | 16  | 32  | -32 | ... | 32  |
| 2   | -32 | -32 | -32 | -32 | -32 | -16 | ... | 32  |

Appendix A.3.2. Kowalik’s Function

\[ F_{15}(z) = \sum_{i=1}^{11} \left[ a_i - \frac{z_1 (b_i^2 + b_i z_2)}{b_i^2 + b_i z_3 + z_4} \right]^2 \]
\[-5 \leq z_i \leq 5, \quad f_{\min} \approx 0.0003075, \quad \text{Dim} = 4, \]

Appendix A.3.3. Six-Hump Camel-Back Function

\[ F_{16}(z) = 4z_1^2 - 2.1z_1^4 + \frac{1}{3}z_1^6 + z_1 z_2 - 4z_2^2 + 4z_2^4 \]
\[-5 \leq z_i \leq 5, \quad f_{\min} = -1.0316285, \quad \text{Dim} = 2, \]

Appendix A.3.4. Branin Function

\[ F_{17}(z) = \left( z_2 - \frac{5.1}{4\pi^2}z_1^2 + \frac{5}{\pi}z_1 - 6 \right)^2 + 10 \left( 1 - \frac{1}{8\pi} \right) \cos z_1 + 10 \]
\[-5 \leq z_1 \leq 10, \quad 0 \leq z_2 \leq 15, \quad f_{\min} = 0.398, \quad \text{Dim} = 2, \]
Appendix A.3.5. Goldstein–Price Function

\[ F_{18}(z) = [1 + (z_1 + z_2 + 1)^2(19 - 14z_1 + 3z_1^2 - 14z_2 + 6z_1z_2 + 3z_2^2)] \times [30 + (2z_1 - 3z_2)^2 \times (18 - 32z_1 + 12z_1^2 + 48z_2 - 36z_1z_2 + 27z_2^2)] \times \]

\[ -2 \leq z_i \leq 2, \quad f_{\text{min}} = 3, \quad \text{Dim} = 2, \]

Appendix A.3.6. Hartman’s Family

- \( F_{19}(z) = -\frac{4}{i=1} c_i \exp \left( -\sum_{j=1}^{3} a_{ij}(z_j - p_{ij})^2 \right) \)

\[ 0 \leq z_j \leq 1, \quad f_{\text{min}} = -3.86, \quad \text{Dim} = 3, \]

- \( F_{20}(z) = -\frac{4}{i=1} c_i \exp \left( -\sum_{j=1}^{6} a_{ij}(z_j - p_{ij})^2 \right) \)

\[ 0 \leq z_j \leq 1, \quad f_{\text{min}} = -3.32, \quad \text{Dim} = 6, \]

Table A2. Hartman Function \( F_{19} \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( (a_{ij}, j = 1, 2, 3) )</th>
<th>( c_i )</th>
<th>( (p_{ij}, j = 1, 2, 3) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3 10 30 1</td>
<td>0.3689</td>
<td>0.1170 0.2673</td>
</tr>
<tr>
<td>2</td>
<td>0.1 10 35 1.2</td>
<td>0.4699</td>
<td>0.4387 0.7470</td>
</tr>
<tr>
<td>3</td>
<td>3 10 30 3</td>
<td>0.1091</td>
<td>0.8732 0.5547</td>
</tr>
<tr>
<td>4</td>
<td>0.1 10 35 3.2</td>
<td>0.038150</td>
<td>0.5743 0.8828</td>
</tr>
</tbody>
</table>

Appendix A.3.7. Shekel’s Foxholes Function

- \( F_{21}(z) = -\frac{5}{i=1} \left[ (X - a_i)(X - a_i)^T + c_i \right]^{-1} \)

\[ 0 \leq z_i \leq 10, \quad f_{\text{min}} = -10.1532, \quad \text{Dim} = 4, \]

- \( F_{22}(z) = -\frac{7}{i=1} \left[ (X - a_i)(X - a_i)^T + c_i \right]^{-1} \)

\[ 0 \leq z_i \leq 10, \quad f_{\text{min}} = -10.4028, \quad \text{Dim} = 4, \]

- \( F_{23}(z) = -\frac{10}{i=1} \left[ (X - a_i)(X - a_i)^T + c_i \right]^{-1} \)

\[ 0 \leq z_i \leq 10, \quad f_{\text{min}} = -10.536, \quad \text{Dim} = 4. \]

Table A3. Shekel Foxholes’ Functions \( F_{21}, F_{22}, F_{23} \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>( (a_{ij}, j = 1, 2, 3, 4) )</th>
<th>( c_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4 4 4 4</td>
<td>0.1</td>
</tr>
<tr>
<td>2</td>
<td>1 1 1 1</td>
<td>0.2</td>
</tr>
<tr>
<td>3</td>
<td>8 8 8 8</td>
<td>0.2</td>
</tr>
<tr>
<td>4</td>
<td>6 6 6 6</td>
<td>0.4</td>
</tr>
<tr>
<td>5</td>
<td>3 7 3 7</td>
<td>0.4</td>
</tr>
<tr>
<td>6</td>
<td>2 9 2 9</td>
<td>0.6</td>
</tr>
<tr>
<td>7</td>
<td>5 5 3 3</td>
<td>0.3</td>
</tr>
<tr>
<td>8</td>
<td>8 1 8 1</td>
<td>0.7</td>
</tr>
<tr>
<td>9</td>
<td>6 2 6 2</td>
<td>0.5</td>
</tr>
<tr>
<td>10</td>
<td>7 3.6 7 3.6</td>
<td>0.5</td>
</tr>
</tbody>
</table>
Table A4. Hartman Function $F_{20}$.

<table>
<thead>
<tr>
<th>$i$</th>
<th>$(a_{ij}, j = 1, 2, \ldots, 6)$</th>
<th>$c_i$</th>
<th>$(p_{ij}, j = 1, 2, \ldots, 6)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10 3 17 3.5 1.7 8</td>
<td>1</td>
<td>0.1312 0.1696 0.5569 0.0124 0.8283 0.5886</td>
</tr>
<tr>
<td>2</td>
<td>0.05 10 17 0.1 8 14</td>
<td>1.2</td>
<td>0.2329 0.4135 0.8307 0.3736 0.1004 0.9991</td>
</tr>
<tr>
<td>3</td>
<td>3 3.5 1.7 10 17 8</td>
<td>3</td>
<td>0.2348 0.1415 0.3522 0.2883 0.3047 0.6650</td>
</tr>
<tr>
<td>4</td>
<td>17 8 0.05 10 0.1 14</td>
<td>3.2</td>
<td>0.4047 0.8828 0.8732 0.5743 0.1091 0.0381</td>
</tr>
</tbody>
</table>

References


42. Zhai, Y.; Tian, H.; Ji, Y. Slow Light Property Improvement and Optical Buffer Capability in Ring-Shape-Hole Photonic Crystal Waveguide. J. Lightwave Technol. 2011, 29, 3083–3090. [CrossRef]