designs

MDPI

Article

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 1. Optimization approaches.

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

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.

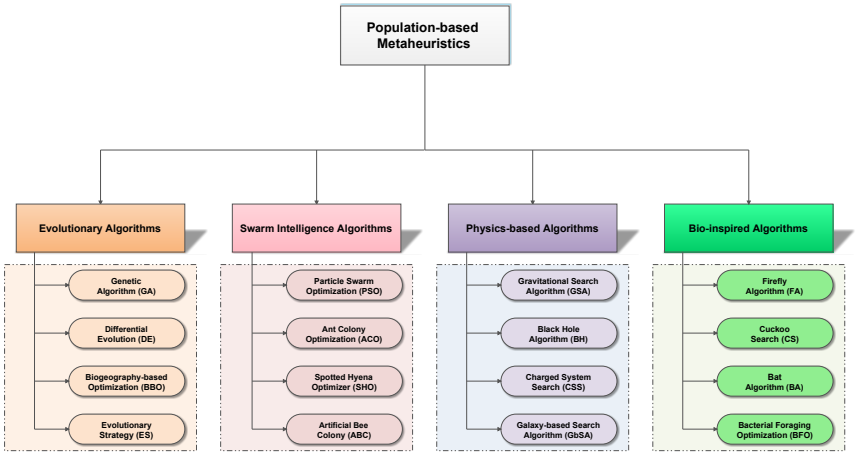


Figure 1. Classification of metaheuristic algorithms [24].

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):

$$\vec{D}_h = \mid \vec{B} \times \vec{P}_p(x) - \vec{P}(x) \mid, \tag{1}$$

$$\vec{P}(x+1) = \vec{P}_p(x) - \vec{E} \times \vec{D}_h, \tag{2}$$

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:

$$\vec{B} = 2 \times r\vec{d}_1, \tag{3}$$

$$\vec{E} = 2\vec{s} \times r\vec{d}_2 - \vec{h}, \tag{4}$$

$$\vec{s} = 5 - \left(Iteration \times \frac{5}{Max_{Iteration}} \right), \tag{5}$$

where, $Iteration = 0, 1, 2, \ldots, Max_{Iteration}$.

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):

$$\vec{D}_h = \mid \vec{B} \times \vec{P}_h - \vec{P}_k \mid, \tag{6}$$

$$\vec{P}_k = \vec{P}_h - \vec{E} \times \vec{D}_h, \tag{7}$$

$$\vec{C}_h = \vec{P}_k + \vec{P}_{k+1} + \ldots + \vec{P}_{k+N}, \tag{8}$$

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):

$$N = count_{nos}(\vec{P}_h, \vec{P}_{h+1}, \vec{P}_{h+2}, \ldots, (\vec{P}_h + \vec{M})), \tag{9}$$

where $\vec{M}$ is a random vector in range $[0.5, 1]$, *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):

$$\vec{P}(x+1) = \frac{\vec{C}_h}{N}, \tag{10}$$

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
procedure SHO
Initialize the parameters h, B, E, and N
Calculate the fitness of each search agent
P_h= the best search agent
C_h= the group or cluster of all far optimal solutions
    while $(x < Max_{Iteration})$ **do**
        for each search agent **do**
            Update the position of current agent by Equation (10)
        end for
        Update h, B, E, and N
        Check if any search agent goes beyond the given search space and then adjust it
        Calculate the fitness of each search agent
        Update P_h if there is a better solution than previous optimal solution
        Update the group C_h with respect to P_h
        $x = x + 1$
    end while
return P_h
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.

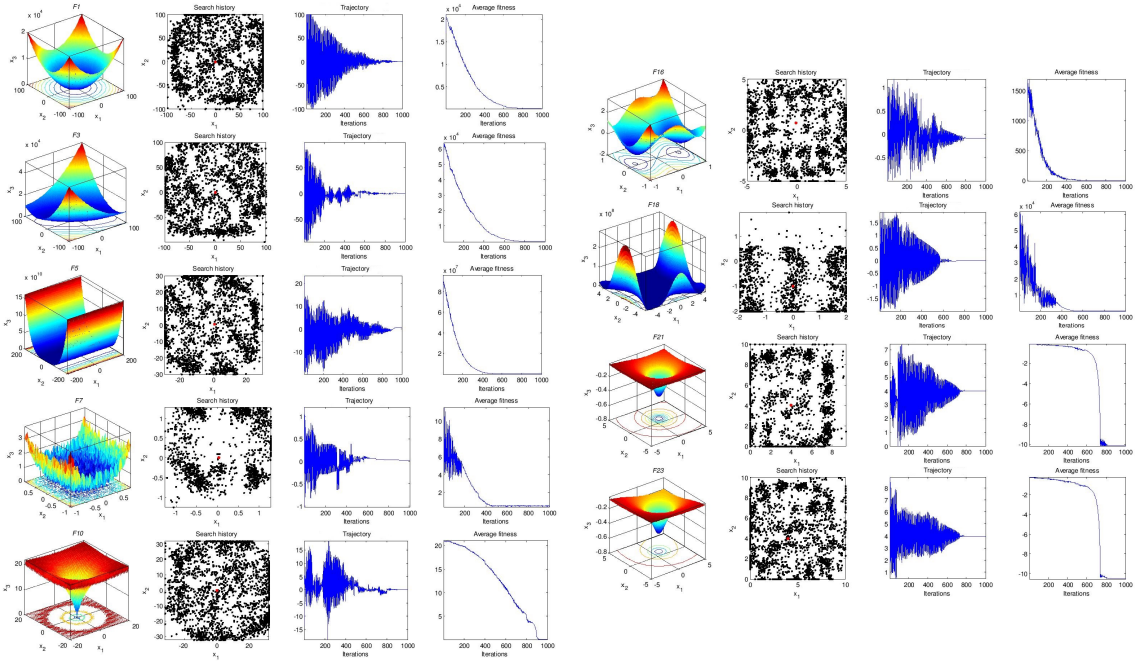


Figure 2. Search history, trajectory, and average fitness of the spotted hyena optimizer (SHO) algorithm on 2D benchmark test problems.

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:

$$\zeta(z) = f(z) \pm \left[\sum_{i=1}^{m} l_i \times max(0, t_i(z))^{\alpha} + \sum_{j=1}^{n} o_j \times \mid U_j(z) \mid^{\beta} \right], \tag{11}$$

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 α and β 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 2. Parameter settings for algorithms.

#	Algorithms	Parameters	Values
1	Spotted Hyena Optimizer (SHO)	Search Agents	80
		Control Parameter ($\vec{h}$)	[5, 0]
		$\vec{M}$ Constant	[0.5, 1]
		Number of Generations	1000
2	Grey Wolf Optimizer (GWO)	Search Agents	80
		Control Parameter ($\vec{a}$)	[2, 0]
		Number of Generations	1000
3	Particle Swarm Optimization (PSO)	Number of Particles	80
		Inertia Coefficient	0.75
		Cognitive and Social Coeff	1.8, 2
		Number of Generations	1000
4	Genetic Algorithm (GA)	Population Size	80
		Crossover	0.9
		Mutation	0.05
		Number of Generations	1000
5	Differential Evolution (DE)	Population Size	80
		Crossover	0.9
		Scale Factor (F)	0.5

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 Wave guide (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]:

$$DBP = \Delta d \times \Delta b, \tag{12}$$

where Δd and Δb indicate the delay and bandwidth of slow light device, respectively,

$$NDBP = \overline{m_g} \times \Delta\omega/\omega_0, \tag{13}$$

where $\overline{m_g}$ is the average of group index, $\Delta\omega$ is the bandwidth, and ω_0 is the central frequency of light wave. However, NDBP has a relation with group index (m_g) as:

$$m_g = \frac{V}{v_g} = C\frac{dv}{d\omega}, \tag{14}$$

where ω 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:

$$\overline{m_g} = \int_{\omega_L}^{\omega_H} m_g(\omega)\frac{d\omega}{\Delta\omega} \tag{15}$$

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):

$$\begin{aligned}
&\vec{z} = [z_1 z_2 z_3 z_4 z_5 z_6 z_7 z_8] = \left[\frac{R_1}{a}\frac{R_2}{a}\frac{R_3}{a}\frac{R_4}{a}\frac{R_5}{a}\frac{l}{a}\frac{w_h}{a}\frac{w_l}{a}\right],\\
&\text{Maximize:} \quad f(\vec{z}) = NDBP,\\
&\text{subject to:}\\
&max(\mid \beta_2(\omega) \mid) < 10^6 a/2\pi c^2,\\
&\omega_H < min(\omega_{upband}),\\
&\omega_L > max(\omega_{lowband}),\\
&k_n > k_{nH} = \omega_{guidedmode} > \omega_H,\\
&k_n < k_{nL} = \omega_{guidedmode} < \omega_L,\\
&\text{where,}\\
&\omega_H = \omega(k_{nH}) = \omega(1.1 m_{g0}), v\omega_L = \omega(k_{nL}) = \omega(0.9 m_{g0}),\\
&vk_n = \frac{ka}{2\pi}, \Delta\omega = \omega_H - \omega_L, a = \omega_0 \times 1550nm,\\
&0 \le z_{1-5} \le 0.5, \quad 0 \le z_6 \le 1, \quad 0 \le z_{7,8} \le 1.
\end{aligned} \tag{16}$$

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.

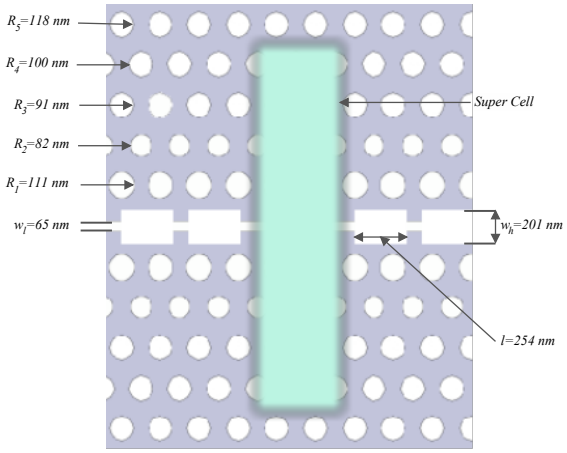


Figure 3. Optimized super cell using the SHO algorithm.

Table 3. Comparison results for the optical buffer design problem.

Parameters	SHO	GWO [28]	Wu et al. [44]
R_1	0.31349a	0.33235a	-
R_2	0.22378a	0.24952a	-
R_3	0.23400a	0.26837a	-
R_4	0.27681a	0.29498a	-
R_5	0.30217a	0.34992a	-
l	0.7263a	0.7437a	-
w_h	0.2052a	0.2014a	-
w_l	0.60027a	0.60073a	-
$a(nm)$	328	343	430
$\overline{m_g}$	17.4	19.6	23
$Bandwidth$	37.2	33.9	17.6
$\beta_2(a/2\pi c^2)$	10^3	10^3	10^3
$NDBP$	0.49	0.43	0.26

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):

$$\begin{aligned} &\text{Minimize:} \quad F(x,y) = E_d(x,y), \\ &\text{Subject to:} -1 \leq x,y \leq 1, \text{and set of } SC, \end{aligned} \tag{17}$$

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.

$$F(x,y) = F(x,y) + pe \sum_{j=1}^{3} P_j, \tag{18}$$

where pe is a constant and P_j defines the violation size on the jth 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.

Parameters	Values
Foil co-ordinates	NACA0012
Reynolds number	10^6
Mach number	0.2
Co-efficient of Lift	0

Table 5. Optimum drag force values using SHO and other competitor approaches.

Algorithms	Force
SHO	0.0068
PSO	0.0021
GA	0.0067
DE	0.0020

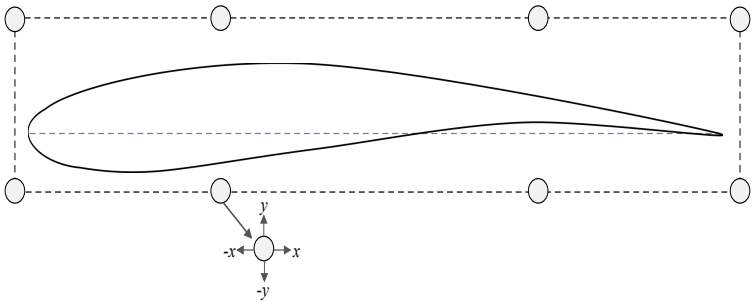
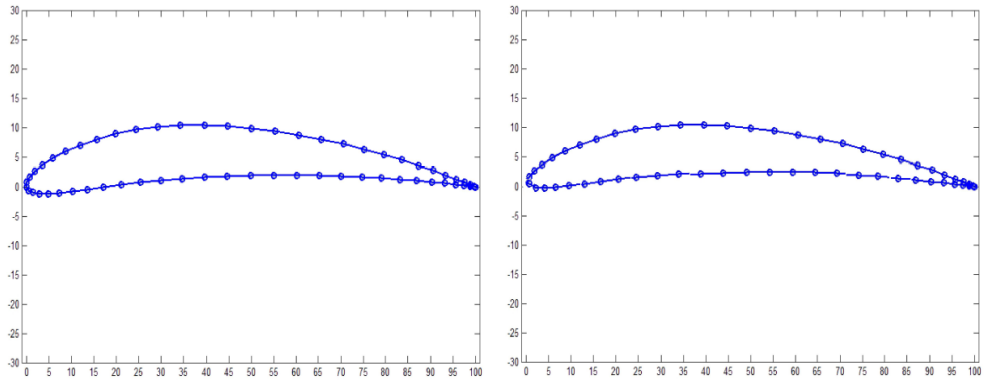


Figure 4. B-spline for the airfoil design problem.



(**a**) (**b**)

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

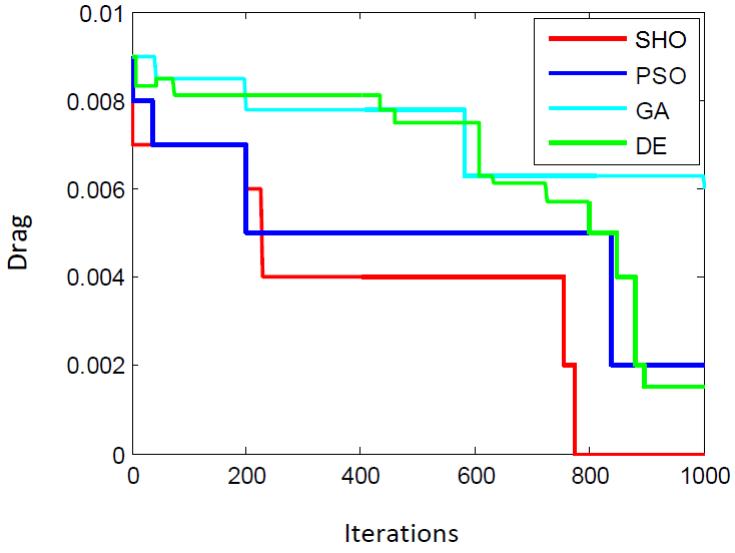


Figure 6. Convergence curves of airfoil design using SHO and other competitive approaches.

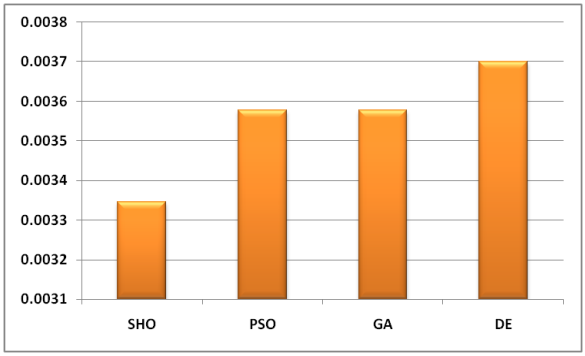


Figure 7. The standard deviation of the proposed as well as competitor algorithms for the airfoil design problem.

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

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

$$-100 \leq z_i \leq 100, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.1.2. Schwefel's Problem 2.22

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

$$-10 \leq z_i \leq 10, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.1.3. Schwefel's Problem 1.2

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

$$-100 \leq z_i \leq 100, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.1.4. Schwefel's Problem 2.21

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

$$-100 \leq z_i \leq 100, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.1.5. Generalized Rosenbrock's Function

$$F_5(z) = \sum_{i=1}^{29}[100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2]$$

$$-30 \leq z_i \leq 30, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.1.6. Step Function

$$F_6(z) = \sum_{i=1}^{30}(\lfloor z_i + 0.5 \rfloor)^2$$

$$-100 \leq z_i \leq 100, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.1.7. Quartic Function

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

$$-1.28 \leq z_i \leq 1.28, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.2. Multimodal Benchmark Test Functions

Appendix A.2.1. Generalized Schwefel's Problem 2.26

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

$$-500 \leq z_i \leq 500, \quad f_{min} = -12569.5, \quad Dim = 30,$$

Appendix A.2.2. Generalized Rastrigin's Function

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

$$-5.12 \leq z_i \leq 5.12, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.2.3. Ackley's Function

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

$$-32 \leq z_i \leq 32, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.2.4. Generalized Griewank Function

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

$$-600 \leq z_i \leq 600, \quad f_{min} = 0, \quad Dim = 30,$$

Appendix A.2.5. Generalized Penalized Functions

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

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

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

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

Table A1. Shekel's Foxholes Function F_{14}.

$(a_{ij}, i = 1,2$ 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

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

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

$$F_{16}(z) = 4z_1^2 - 2.1z_1^4 + \frac{1}{3}z_1^6 + z_1z_2 - 4z_2^2 + 4z_2^4$$
$$-5 \le z_i \le 5, \quad f_{min} = -1.0316285, \quad Dim = 2,$$

Appendix A.3.4. Branin Function

$$F_{17}(z) = \left(z_2 - \frac{5.1}{4\pi^2}z_1^2 + \frac{5}{\pi}z_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)cosz_1 + 10$$
$$-5 \le z_1 \le 10, \quad 0 \le z_2 \le 15, \quad f_{min} = 0.398, \quad Dim = 2,$$

Appendix A.3.5. Goldstein–Price Function

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

$$-2 \leq z_i \leq 2, \quad f_{min} = 3, \quad Dim = 2,$$

Appendix A.3.6. Hartman's Family

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

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

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

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

Table A2. Hartman Function F_{19}.

i	$(a_{ij}, j = 1,2,3)$			c_i	$(p_{ij}, j = 1,2,3)$		
1	3	10	30	1	0.3689	0.1170	0.2673
2	0.1	10	35	1.2	0.4699	0.4387	0.7470
3	3	10	30	3	0.1091	0.8732	0.5547
4	0.1	10	35	3.2	0.038150	0.5743	0.8828

Appendix A.3.7. Shekel's Foxholes Function

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

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

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

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

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

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

Table A3. Shekel Foxholes' Functions F_{21}, F_{22}, F_{23}.

i	$(a_{ij}, j = 1,2,3,4)$				c_i
1	4	4	4	4	0.1
2	1	1	1	1	0.2
3	8	8	8	8	0.2
4	6	6	6	6	0.4
5	3	7	3	7	0.4
6	2	9	2	9	0.6
7	5	5	3	3	0.3
8	8	1	8	1	0.7
9	6	2	6	2	0.5
10	7	3.6	7	3.6	0.5

Table A4. Hartman Function F_{20}.

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

References

1. Marzband, M.; Azarinejadian, F.; Savaghebi, M.; Pouresmaeil, E.; Guerrero, J.M.; Lightbody, G. Smart transactive energy framework in grid-connected multiple home microgrids under independent and coalition operations. *Renew. Energy* **2018**, *126*, 95–106. [CrossRef]
2. Kaur, A.; Dhiman, G. A Review on Search Based Tools and Techniques to Identify Bad Code Smells in Object Oriented Systems. In *Advances in Intelligent Systems and Computing*; Springer: New York, NY, USA, 2018; in press.
3. Singh, P.; Dhiman, G. Uncertainty Representation using Fuzzy-Entropy Approach: Special Application in Remotely Sensed High Resolution Satellite Images (RSHRSIs). *Appl. Soft Comput.* **2018**, in press.
4. Dhiman, G.; Kaur, A. A Hybrid Algorithm based on Particle Swarm and Spotted Hyena Optimizer for Global Optimization. In *Advances in Intelligent Systems and Computing*; Springer: New York, NY, USA, 2018; in press.
5. Dhiman, G.; Kumar, V. Multi-objective spotted hyena optimizer: A Multi-objective optimization algorithm for engineering problems. *Knowl.-Based Syst.* **2018**, *150*, 175–197. [CrossRef]
6. Bonabeau, E.; Dorigo, M.; Theraulaz, G. *Swarm Intelligence: From Natural to Artificial Systems*; Oxford University Press, Inc.: Oxford, UK, 1999.
7. Beyer, H.G.; Schwefel, H.P. Evolution strategies—A comprehensive introduction. *Nat. Comput.* **2002**, *1*, 3–52. [CrossRef]
8. Storn, R.; Price, K. Differential Evolution—A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
9. Simon, D. Biogeography-Based Optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
10. Selvakumar, A.I.; Thanushkodi, K. A New Particle Swarm Optimization Solution to Nonconvex Economic Dispatch Problems. *IEEE Trans. Power Syst.* **2007**, *22*, 42–51. [CrossRef]
11. Nobile, M.S.; Cazzaniga, P.; Besozzi, D.; Colombo, R.; Mauri, G.; Pasi, G. Fuzzy Self-Tuning PSO: A settings-free algorithm for global optimization. *Swarm Evol. Comput.* **2018**, *39*, 70–85. [CrossRef]
12. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
13. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
14. Kaveh, A.; Talatahari, S. A novel heuristic optimization method: Charged system search. *Acta Mech.* **2010**, *213*, 267–289. [CrossRef]
15. Hatamlou, A. Black hole: A new heuristic optimization approach for data clustering. *Inf. Sci.* **2013**, *222*, 175–184. [CrossRef]
16. Dhiman, G.; Kumar, V. Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. *Knowl.-Based Syst.* **2018**, doi:10.1016/j.knosys.2018.06.001. [CrossRef]
17. Alatas, B. ACROA: Artificial Chemical Reaction Optimization Algorithm for global optimization. *Expert Syst. Appl.* **2011**, *38*, 13170–13180. [CrossRef]
18. Kaveh, A.; Khayatazad, M. A new meta-heuristic method: Ray Optimization. *Comput. Struct.* **2012**, *112–113*, 283–294. [CrossRef]
19. Shah Hosseini, H. Principal components analysis by the galaxy-based search algorithm: A novel metaheuristic for continuous optimisation. *Int. J. Comput. Sci. Eng.* **2011**, *6*, 132–140. [CrossRef]

20. Dorigo, M.; Birattari, M.; Stutzle, T. Ant Colony Optimization - Artificial Ants as a Computational Intelligence Technique. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
21. Yang, X.S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
22. Yang, X.S. *A New Metaheuristic Bat-Inspired Algorithm*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 65–74.
23. Yang, X. Firefly Algorithm, Stochastic Test Functions and Design Optimisation. *Int. J. Bio-Inspired Comput.* **2010**, *2*, 78–84. [CrossRef]
24. Dhiman, G.; Kumar, V. Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Adv. Eng. Softw.* **2017**, *114*, 48–70. [CrossRef]
25. Ghorbani, N.; Babaei, E. Exchange market algorithm. *Appl. Soft Comput.* **2014**, *19*, 177–187. [CrossRef]
26. Ramezani, F.; Lotfi, S. Social-Based Algorithm. *Appl. Soft Comput.* **2013**, *13*, 2837–2856. [CrossRef]
27. Geem, Z.W.; Kim, J.H.; Loganathan, G.V. A New Heuristic Optimization Algorithm: Harmony Search. *Simulation* **2001**, *76*, 60–68. [CrossRef]
28. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
29. Sadollah, A.; Bahreininejad, A.; Eskandar, H.; Hamdi, M. Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Appl. Soft Comput.* **2013**, *13*, 2592–2612. [CrossRef]
30. Alba, E.; Dorronsoro, B. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. *IEEE Trans. Evol. Comput.* **2005**, *9*, 126–142. [CrossRef]
31. Lozano, M.; Garcia-Martinez, C. Hybrid Metaheuristics with Evolutionary Algorithms Specializing in Intensification and Diversification: Overview and Progress Report. *Comput. Oper. Res.* **2010**, *37*, 481–497. [CrossRef]
32. Singh, P.; Dhiman, G. A hybrid fuzzy time series forecasting model based on granular computing and bio-inspired optimization approaches. *J. Comput. Sci.* **2018**. [CrossRef]
33. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
34. Singh, P.; Dhiman, G. *A Fuzzy-LP Approach in Time Series Forecasting*; Shankar, B.U., Ghosh, K., Mandal, D.P., Ray, S.S., Zhang, D., Pal, S.K., Eds.; Pattern Recognition and Machine Intelligence; Springer International Publishing: Cham, Switzerland, 2017; pp. 243–253.
35. Chandrawat, R.K.; Kumar, R.; Garg, B.P.; Dhiman, G.; Kumar, S. An Analysis of Modeling and Optimization Production Cost Through Fuzzy Linear Programming Problem with Symmetric and Right Angle Triangular Fuzzy Number. In *Proceedings of Sixth International Conference on Soft Computing for Problem Solving: SocProS 2016, Volume 1*; Deep, K., Bansal, J.C., Das, K.N., Lal, A.K., Garg, H., Nagar, A.K., Pant, M., Eds.; Springer: Singapore, 2017; pp. 197–211. [CrossRef]
36. Pritpal Singh, K.R.; Dhiman, G. A Four-Way Decision-Making System for the Indian Summer Monsoon Rainfall. *Mod. Phys. Lett. B* **2018**, in press.
37. Dhiman, G.; Kaur, A. Spotted Hyena Optimizer for Solving Engineering Design Problems. In Proceedings of the 2017 International Conference on Machine Learning and Data Science (MLDS), Noida, India, 14–15 December 2017; pp. 114–119. [CrossRef]
38. Dhiman, G.; Kumar, V. Spotted Hyena Optimizer for Solving Complex and Non-linear Constrained Engineering Problems. In *Advances in Intelligent Systems and Computing*; Springer: New York, NY, USA, 2018; in press.
39. van den Bergh, F.; Engelbrecht, A. A study of particle swarm optimization particle trajectories. *Inf. Sci.* **2006**, *176*, 937–971. [CrossRef]
40. Coello, C.A.C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Comput. Methods Appl. Mech. Eng.* **2002**, *191*, 1245–1287. [CrossRef]
41. Baba, T. Slow light in photonic crystals. *Nat. Photonics* **2008**, *2*, 465–473. [CrossRef]
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]

43. Mirjalili, S.M.; Abedi, K.; Mirjalili, S. Optical buffer performance enhancement using Particle Swarm Optimization in Ring-Shape-Hole Photonic Crystal Waveguide. *Opt.-Int. J. Light Electron Opt.* **2013**, *124*, 5989–5993. [CrossRef]
44. Wu, J.; Li, Y.; Peng, C.; Wang, Z. Wideband and low dispersion slow light in slotted photonic crystal waveguide. *Opt. Commun.* **2010**, *283*, 2815–2819. [CrossRef]