Advances in Engineering Software 114 (2017) 48–70

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Research paper

Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications

Gaurav Dhiman, Vijay Kumar*

Computer Science and Engineering Department, Thapar University, Patiala, India

ARTICLE INFO

Article history:
Received 27 October 2016
Revised 11 March 2017
Accepted 21 May 2017
Available online 27 May 2017

Keywords:
Optimization
Optimization techniques
Metaheuristics
Constrained optimization
Unconstrained optimization
Benchmark test functions

ABSTRACT

This paper presents a novel metaheuristic algorithm named as Spotted Hyena Optimizer (SHO) inspired by the behavior of spotted hyenas. The main concept behind this algorithm is the social relationship between spotted hyenas and their collaborative behavior. The three basic steps of SHO are searching for prey, encircling, and attacking prey and all three are mathematically modeled and implemented. The proposed algorithm is compared with eight recently developed metaheuristic algorithms on 29 well-known benchmark test functions. The convergence and computational complexity is also analyzed. The proposed algorithm is applied to five real-life constraint and one unconstrained engineering design problems to demonstrate their applicability. The experimental results reveal that the proposed algorithm performs better than the other competitive metaheuristic algorithms.

1. Introduction

In the last few decades, the increase in complexity of real life problems has given risen the need of better metaheuristic techniques. These have been used for obtaining the optimal possible solutions for real-life engineering design problems. These become more popular due to their efficiency and complexity as compared to other existing classical techniques [35].

Metaheuristics are broadly classified into three categories such as evolutionary-based, physical-based, and swarm-based methods. The first technique is generic population-based metaheuristic which is inspired from biological evolution such as reproduction, mutation, recombination, and selection. The evolutionary algorithms are inspired by theory of natural selection in which a population (i.e., a set of solutions) tries to survive based on the fitness evaluation in a given environment (defined as fitness evaluation). Evolutionary algorithms often perform well near optimal solutions to all types of problems because these methods ideally do not make any assumption about the basic fitness or adaptive landscape. Some of the popular evolutionary-based techniques are Genetic Algorithms (GA) [7], Genetic Programming (GP) [34], Evolution Strategy (ES) [6], and Biogeography-Based Optimizer (BBO) [56].

Some of well-known techniques such as Genetic Algorithms(GA) [7], Ant Colony Optimization (ACO) [12], Particle Swarm Optimization (PSO) [32] and Differential Evolution (DE) [57] are popular among different fields. Due to easy implementation, metaheuristic optimization algorithms are more popular in engineering applications [4,9,50](Fig. 1). The second category is physical-based algorithms. In these algorithms, search agents communicate and move throughout the search space according to physics rules such as gravitational force, electromagnetic force, inertia force, and so on. The name of few algorithms are Simulated Annealing (SA) [33], Gravitational Search Algorithm (GSA) [52], Big-Bang Big-Crunch (BBBC) [14], Charged System Search (CSS) [31], Black Hole (BH) [23] algorithm, Central Force Optimization (CFO) [16], Small-World Optimization Algorithm (SWOA) [13], Artificial Chemical Reaction Optimization Algorithm (ACROA) [1], Ray Optimization (RO) algorithm [29], Galaxy-based Search Algorithm (GbSA) [54], and Curved Space Optimization (CSO) [45].

The last one is swarm-based algorithms which are based on the collective behavior of social creatures. The collective intelligence is inspired by the interaction of swarm with each other and their environment. The well-known algorithm of SI technique is Particle Swarm Optimization (PSO). Another popular swarm-intelligence technique is Ant Colony Optimization [12], Monkey Search [47], Wolf pack search algorithm [61], Bee Collecting Pollen Algorithm (BCPA) [39], Cuckoo Search (CS) [64], Dolphin Partner Optimization (DPO) [55], Bat-inspired Algorithm (BA) [63], Firefly Algorithm (FA) [62], Hunting Search (HUS) [48]. Generally, swarm-based algorithms are easier to implement than evolutionary-based

* Corresponding author.
E-mail addresses: gdhiman0001@gmail.com (G. Dhiman), vijaykumarchahar@gmail.com (V. Kumar).

http://dx.doi.org/10.1016/j.advengsoft.2017.05.014
0965-9978/

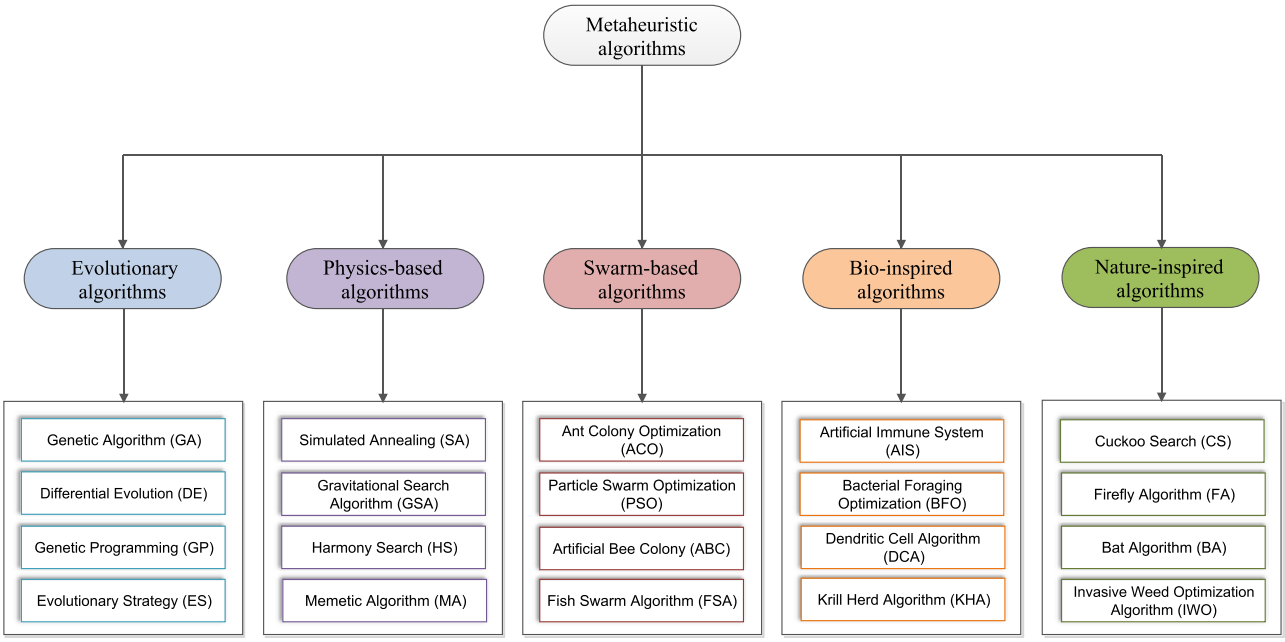

Fig. 1. Classification of metaheuristic algorithms.

algorithms due to include fewer operators (i.e., selection, crossover, mutation). Apart from these, there are other metaheuristic techniques inspired by human behaviors. Some of the popular algorithms are Harmony Search (HS) [19], Parameter Adaptive Harmony Search (PAHS) [35], Variance-Based Harmony Search [36], Harmony Search-Based Remodularization Algorithm (HSBRA) [3], Tabu (Taboo) Search (TS) [15,21,22], Group Search Optimizer (GSO) [24,25], Imperialist Competitive Algorithm (ICA) [5], League Championship Algorithm (LCA) [28], Firework Algorithm [59], Colliding Bodies Optimization (CBO) [30], Interior Search Algorithm (ISA)[17], Mine Blast Algorithm (MBA) [53], Soccer League Competition (SLC) algorithm [46], Seeker Optimization Algorithm (SOA) [10], Social-Based Algorithm (SBA) [51], and Exchange Market Algorithm (EMA) [20].

The main components of metaheuristic algorithms are exploration and exploitation [2,49]. Exploration ensures the algorithm to reach different promising regions of the search space, whereas exploitation ensures the searching of optimal solutions within the given region [38]. The fine tuning of these components is required to achieve the optimal solution for a given problem. It is difficult to balance between these components due to stochastic nature of optimization problem. This fact motivates us to develop a novel metaheuristic algorithm for solving real-life engineering design problem.

The performance of one optimizer to solve the set of problem does not guarantee to solve all optimization problems with different nature [60]. It is also the motivation of our work and describes a new metaheuristic based algorithm.

This paper introduces a novel metaheuristic algorithm for optimizing constraint and unconstrained design problems. The main objective of this paper is to develop a novel metaheuristic algorithm named as Spotted hyena Optimization (SHO), which is inspired by social hierarchy and hunting behavior of spotted hyenas. Cohesive clusters can help for efficient co-operation between spotted hyenas. The main steps of SHO are inspired by hunting behavior of spotted hyenas. The performance of the SHO algorithm is evaluated on twenty-nine benchmark test functions and six real structural optimization problems. The results demonstrate that the performance of SHO performs better than the other competitive algorithms.

The rest of this paper is structured as follows: Section 2 presents the concepts of the proposed SHO algorithm. The experimental results and discussion is presented in Section 3. In Section 4, the performance of SHO is tested on five constrained and one unconstrained engineering design problems and compared with other well-known algorithms. Finally, the conclusion and some future research directions are given in Section 5.

2. Spotted hyena optimizer (SHO)

In this section, the mathematical modeling of proposed algorithm is described in detail.

2.1. Inspiration

Social relationships are dynamic in nature. These are affected by the changes in the relationships among comprising the network and individuals leaving or joining the population. The social network analysis of animal behavior has been classified into three categories [26]:

- The first category includes environmental factors, such as resource availability and competition with other animal species.
- The second category focuses on social preferences based on individual behavior or quality.
- The third category has less attention from scientists which includes the social relations of species itself.

The social relation between the animals is the inspiration of our work and correlates this behavior to spotted hyena which is scientifically named as Crocuta.

Hyenas are large dog-like carnivores. They live in savannas, grasslands, sub-deserts and forests of both Africa and Asia. They live 10–12 years in the wild and up to 25 years in imprisonment. There are four known species of hyena these are, spotted hyena,

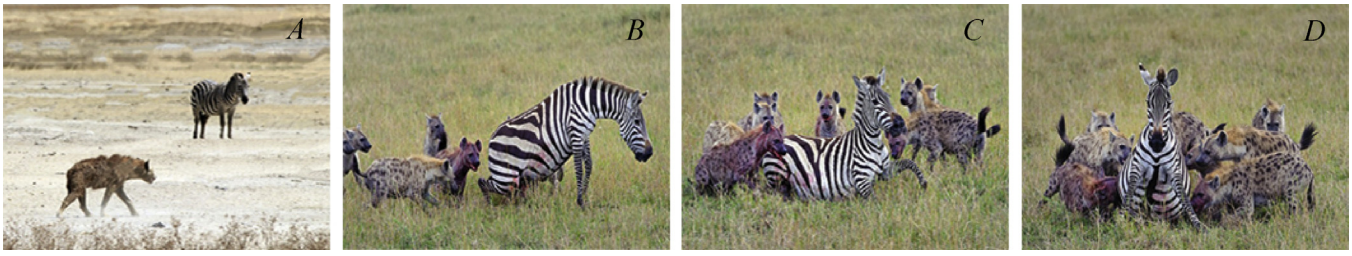

Fig. 2. Hunting behavior of spotted hyenas: (A) searching and tracking prey (B) chasing (C) troublesome and encircling (D) immobile situation and attack prey.

striped hyena, brown hyena and aardwolf that differ in size, behavior, and type of diet. All these species have a bear-like attitude as the front legs are longer than the back legs.

Spotted hyenas are skillful hunters and largest of three other hyena species (i.e., striped, brown, and aardwolf). The Spotted Hyena is also called Laughing Hyena because its sounds is much similar to a human laugh. They are so called because there are spots on their fur which is reddish brown in color with black spots. Spotted hyenas are complicated, intelligent, and highly social animals with really dreadful reputation. They have an ability to fight endlessly over territory and food.

In spotted hyenas family, female members are dominant and live in their clan. However, male members leave their clan when they are adults to search and join a new clan. In this new family, they are lowest ranking members to get their share of the meal. A male member who has joined the clan always stays with the same members (friends) for a long time. Whereas a female, is always assured of a stable place. An interesting fact about spotted hyenas is that they produce sound alert which is very similar to human laugh to communicate with each other when a new food source is found.

According to Ilany et al. [26] spotted hyenas usually live and hunt in groups, rely on a network of trusted friends have more than 100 members. And to increase their network, they usually tie up with another spotted hyena that is a friend of a friend or linked in some way through kinship rather than any unknown spotted hyena. Spotted hyenas are social animals that can communicate with each other through specialized calls such as postures and signals. They use multiple sensory procedures to recognize their kin and other individuals. They can also recognize third party kin and rank the relationships between their clan mates and use this knowledge during social decision making. The spotted hyena track prey by sight, hearing, and smell. Fig. 2 shows the tracking, chasing, encircling, and attacking mechanism of spotted hyenas. Cohesive clusters are helpful for efficient co-operation between spotted hyenas and also maximize the fitness. In this work, the hunting technique and the social relation of spotted hyenas are mathematically modeled to design SHO and perform optimization.

2.2. Mathematical model and optimization algorithm

In this subsection, the mathematical models of the searching, encircling, hunting, and attacking prey are provided. Then the SHO algorithm is outlined.

2.2.1. Encircling prey

Spotted hyenas can be familiar with the location of prey and encircle them. To mathematically model the social hierarchy of spotted hyenas, we consider the current best candidate solution is the target prey or objective which is close to the optimum because of search space not known a priori. The other search agents will try to update their positions, after the best search candidate solution is defined, about the best optimal candidate solution. The

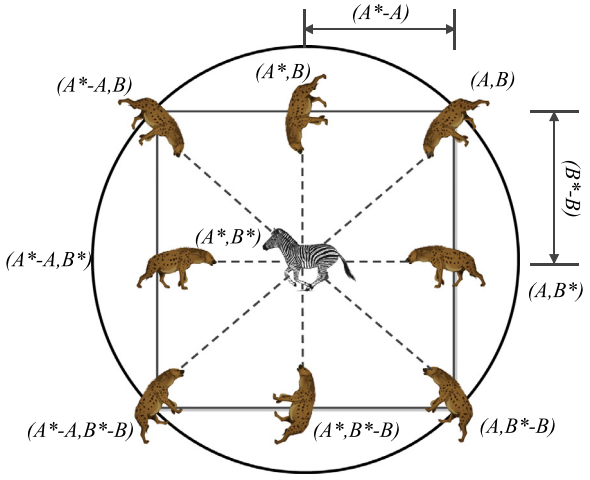

Fig. 3. 2D position vectors of spotted hyena.

mathematical model of this behavior is represented by the following equations:

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

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

where $\vec{D}_h$ define the distance between the prey and spotted hyena, x indicates the current iteration, $\vec{B}$ and $\vec{E}$ are co-efficient vectors, $\vec{P}_p$ indicates the position vector of prey, $\vec{P}$ is the position vector of spotted hyena. However, $||$ and $\cdot$ is the absolute value and multiplication with vectors respectively.

The vectors $\vec{B}$ and $\vec{E}$ are calculated as follows:

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

$$\vec{E} = 2\vec{h} \cdot r\vec{d}_2 - \vec{h} \tag{4}$$

$$\vec{h} = 5 - (Iteration * (5/Max_{Iteration}))$$
$$\text{where}, Iteration = 1, 2, 3, \ldots, Max_{Iteration} \tag{5}$$

For proper balancing the exploration and exploitation, $\vec{h}$ is linearly decreased from 5 to 0 over the course of maximum number of iterations ($Max_{Iteration}$). Further, this mechanism promotes more exploitation as the iteration value increases. However, $r\vec{d}_1$, $r\vec{d}_2$ are random vectors in [0, 1]. Fig. 3 shows the effects of Eqs. (1) and (2) in two-dimensional environment. In this figure, the spotted hyena *(A,B)* can update its position towards the position of prey *(A*,B*)*. By adjusting the value of vectors $\vec{B}$ and $\vec{E}$, there are a different number of places which can be reached about the current position. The probably updated positions of a spotted hyena in the 3D environment are shown in Fig. 4. By using Eqs. (1) and (2), a spotted hyena can update its position randomly

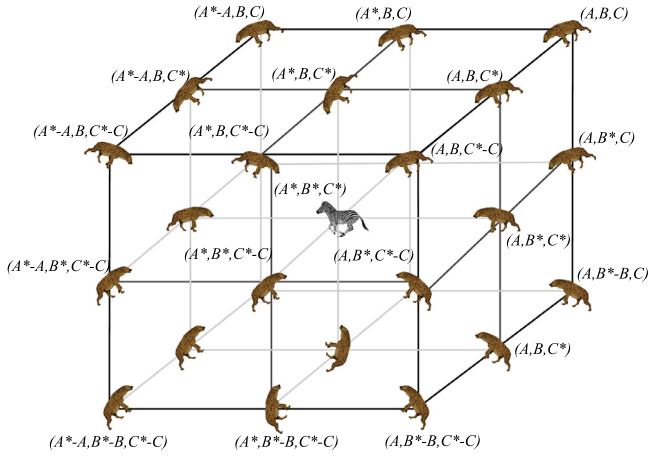

Fig. 4. 3D position vectors and possible next locations of spotted hyena.

around the prey. Therefore, the same concept can further extend with n-dimensional search space.

2.2.2. Hunting

Spotted hyenas usually live and hunts in groups and relies on a network of trusted friends and ability to recognize the location of prey. To define the behavior of spotted hyenas mathematically, we suppose that the best search agent, whichever is optimum, has knowledge the location of prey. The other search agents make a cluster, trusted friends group, towards the best search agent and saved the best solutions obtained so far to update their positions. The following equations are proposed in this mechanism:

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

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

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

where $\vec{P}_h$ defines the position of first best spotted hyena, $\vec{P}_k$ indicates the position of other spotted hyenas. Here, N indicates the number of spotted hyenas which is computed as follows:

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

where $\vec{M}$ is a random vector in [0.5, 1], *nos* defines the number of solutions and count all candidate solutions, after addition with $\vec{M}$, which are far similar to the best optimal solution in a given search space, and $\vec{C}_h$ is a group or cluster of N number of optimal solutions.

2.2.3. Attacking prey (exploitation)

In order to mathematically model for attacking the prey, we decrease the value of vector $\vec{h}$. The variation in vector $\vec{E}$ is also decreased to change the value in vector $\vec{h}$ which can decrease from 5 to 0 over the course of iterations. Fig. 6 shows that $|E| < 1$ forces the group of spotted hyenas to assault towards the prey. The mathematical formulation for attacking the prey is as follows:

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

where $\vec{P}(x+1)$ save the best solution and updates the positions of other search agents according to the position of the best search agent. The SHO algorithm allows its search agents to update their position and attack towards the prey.

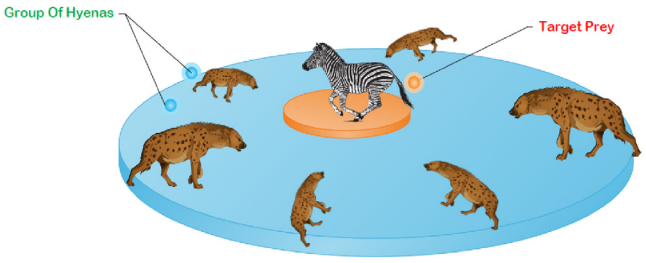

Fig. 5. Searching prey ($|E| > 1$).

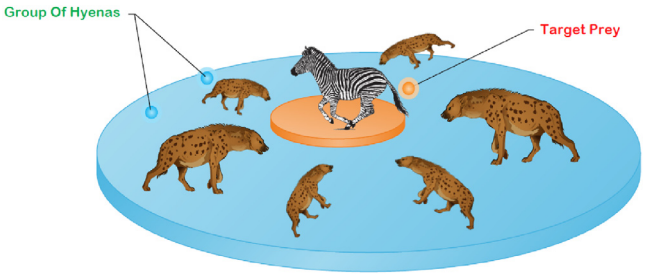

Fig. 6. Attacking prey ($|E| < 1$).

2.2.4. Search for prey(exploration)

Spotted hyenas mostly search the prey, according to the position of the group or cluster of spotted hyenas which reside in vector $\vec{C}_h$. They move away from each other to search and attack for prey. Therefore, we use $\vec{E}$ with random values which are greater than 1 or less than −1 to force the search agents to move far away from the prey. This mechanism allows the SHO algorithm to search globally. To find a suitable prey, Fig. 5 shows that $|E| > 1$ facilitates the spotted hyenas to move away from the prey. Another constituent of SHO algorithm which makes possible for exploration is $\vec{B}$. In Eq. (3), the $\vec{B}$ vector contains random values which provide the random weights of prey. To show the more random behavior of SHO algorithm, assume vector $\vec{B} > 1$ precedence than $\vec{B} < 1$ to demonstrate the effect in the distance as may be seen in Eq. (3). This will helpful for exploration and local optima avoidance. Depending on the position of a spotted hyenas, it can randomly decide a weight to the prey and possible makes it rigid or beyond to reach for spotted hyenas. We intentionally need vector $\vec{B}$ to provide random values for exploration not only during initial iterations but for also final iterations. This mechanism is very helpful to avoid local optima problem, more than ever in the final iterations. Finally, the SHO algorithm is terminated by satisfying termination criteria.

The pseudo code of the SHO algorithm shows that how SHO can solve optimization problems, some points may be noted as follows:

- The proposed algorithm saves the best solutions obtained so far over the course of the iteration.
- The proposed encircling mechanism defines a circle-shaped the neighborhood around the solutions which can be extended to higher dimensions as a hyper-sphere.
- Random vectors $\vec{B}$ and $\vec{E}$ assist candidate solutions to have hyper-spheres with different random positions.
- The proposed hunting method allows candidate solutions to locate the probable position of the prey.
- The possibility of exploration and exploitation by the adjusted values of the vectors $\vec{E}$ and $\vec{h}$ and allows SHO to easily changeover between exploration and exploitation.
- With vector $\vec{E}$, half of the iterations are dedicated to searching (exploration) ($|E| \geq 1$) and the other half are devoted to hunting (exploitation) ($|E| \leq 1$).

2.3. Steps and flowchart of SHO

The steps of SHO are summarized as follows:

Step 1: Initialize the spotted hyenas population P_i where $i = 1, 2, \ldots, n$.
Step 2: Choose the initial parameters of SHO: h, B, E, and N and define the maximum number of iterations.
Step 3: Calculate the fitness value of each search agent.
Step 4: The best search agent is explored in the given search space.
Step 5: Define the group of optimal solutions, i.e cluster using Eqs. (8) and (9) until the satisfactory result is found.
Step 6: Update the positions of search agents using Eq. (10).
Step 7: Check whether any search agent goes beyond the boundary in a given search space and adjust it.
Step 8: Calculate the update search agent fitness value and update the vector P_h if there is a better solution than previous optimal solution.
Step 9: Update the group of spotted hyenas C_h to updated search agent fitness value.
Step 10: If the stopping criterion is satisfied, the algorithm will be stopped. Otherwise, return to Step 5.
Step 11: Return the best optimal solution, after stopping criteria is satisfied, which is obtained so far.

Algorithm 1 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 $(x < Max\,number\,of\,iterations)$ **do**
    for each search agent **do**
      Update the position of current agent by Eq. (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 w.r.t P_h
    x=$x + 1$
  end while
return P_h
end procedure

2.4. Computational complexity

In this subsection, the computational complexity of proposed algorithm is discussed. Both the time and space complexities of proposed algorithm is given below (Fig. 7).

2.4.1. Time complexity

1. Initialization of SHO population needs $O(n \times dim)$ time where n indicates the number of iterations to generate random population which is based on the number of search agents, lowerbound, and upperbound of a test function. However, dim indicates the dimension of a test function to check and adjust the solutions which are goes beyond the search space.
2. In the next step, the fitness of each agent requires $O(Max_{iter} \times n \times dim)$ time where Max_{iter} is the maximum number of iterations to simulate the proposed algorithm.
3. It requires $O(Max_{iter} \times N)$ time to define the group of spotted hyenas where Max_{iter} is the maximum iteration of an algorithm and N is the counting value of spotted hyenas.

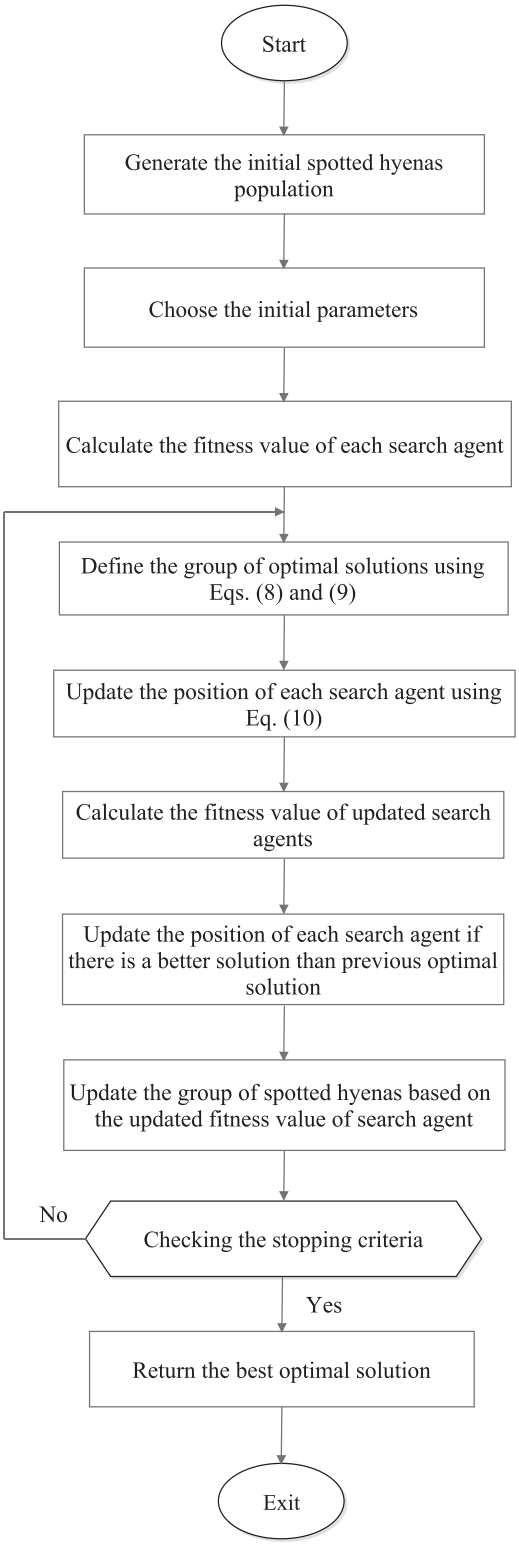

Fig. 7. Flowchart of the proposed SHO.

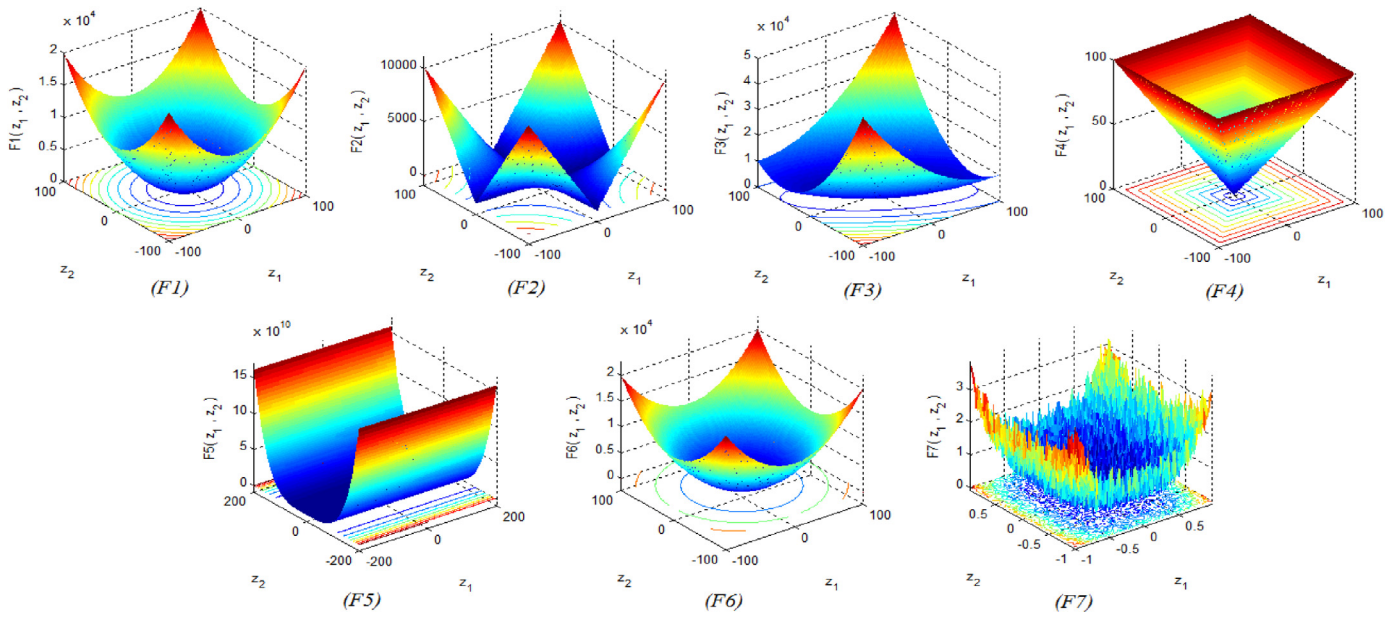

Fig. 8. 2-D versions of unimodal benchmark functions.

4. Steps 2 and 3 is repeated until the satisfactory results is found which needs $O(k)$ time.

Hence, the total complexity of Step 2 and 3 is $O(n \times Max_{iter} \times dim \times N)$. Therefore, the overall time complexity of SHO algorithm is $O(k \times n \times Max_{iter} \times dim \times N)$.

2.4.2. Space complexity

The space complexity of SHO algorithm is the maximum amount of space used at any one time which is considered during its initialization process. Thus, the total space complexity of SHO algorithm is $O(n \times dim)$.

3. Experimental results and discussion

This section describes the experimentation to evaluate the performance of proposed algorithm. Twenty-nine standard benchmark test functions are utilized for evaluating the performance of proposed algorithm. These benchmarks are described in Section 3.1. The results are evaluated and compared with eight well-known metaheuristic algorithms.

3.1. Benchmark test functions and algorithms used for comparisons

The twenty-nine benchmark functions are applied on proposed algorithm to demonstrate its efficiency. These functions are divided into four main categories such as unimodal [11], multimodal [62], fixed-dimension multimodal [11,62], and composite functions [37]. These are described in the Appendix A. In Appendix A Dim and Range indicate the dimension of the function and boundary of the search space respectively. Tables A.1 and A.2 show the characteristics of unimodal and multimodal benchmark functions respectively. The description of fixed-dimension modal and composite test functions are tabulated in Tables A.3 and A.4 respectively. The seven test functions ($F_1 - F_7$) are included in unimodal test functions. There is only one global optimum and has no local optima in the first group of test functions which make them highly appropriate for analyzing the convergence speed and exploitation ability of an algorithm. The second category consists of nine test functions ($F_8 - F_{16}$). The third category includes ten test functions ($F_{14} - F_{23}$). In the second and third category of test functions, there are multiple local solutions besides the global optimum which are useful for examining the local optima avoidance and an explorative ability of an algorithm. The fourth category comprises six composite test functions ($F_{24} - F_{29}$). These composite benchmark functions are the shifted, rotated, expanded, and combined version of classical functions [58]. All of these benchmark test functions are minimization problems. Figs. 8–11 show the 2D plots of the cost function for unimodal, multimodal, fixed-dimension multimodal, and composite functions test cases respectively.

To validate the performance of the proposed algorithm, the eight well-known algorithms are chosen for comparison. These are Grey Wolf Optimizer (GWO) [44], Particle Swarm Optimization (PSO) [32], Moth-Flame Optimization (MFO) [41], Multi-Verse Optimizer (MVO) [43], Sine Cosine Algorithm (SCA) [42], Gravitational Search Algorithm (GSA) [52], Genetic Algorithm (GA) [7], and Harmony Search (HS) [19]. The number of search agent for each algorithms are set to 30. It is observed that 30 is a reasonable number of search agents for solving the optimization problems because of larger the number of artificial search agents, the higher probability of determining the global optimum.

3.2. Experimental setup

The parameter setting of proposed SHO and other metaheuristic algorithms such as GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS are mentioned in the Table 1. All of these parameters are set according to the reported literature. We used the same initialization technique to compare between SHO and the above-mentioned metaheuristic algorithms. The experimentation and algorithms are implemented in Matlab R2014a (8.3.0.532) version and run it in the environment of Microsoft Windows 8.1 with 64 bits on Core i-5 processor with 2.40 GHz and 4GB memory. The average and standard deviation of the best obtained optimal solution till the last iteration is computed as the metrics of performance. To generate and report the results, for each benchmark function, the SHO algorithm utilizes 30 independent runs where each run employs 1000 times of iterations.

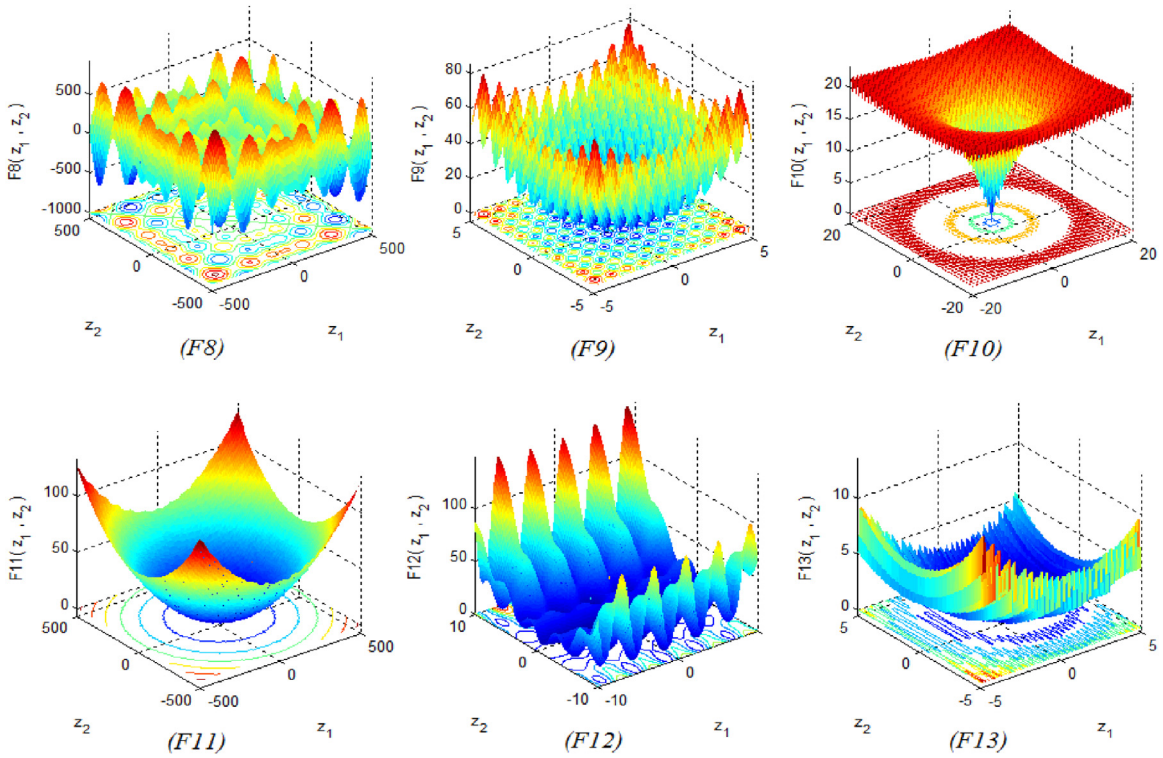

Fig. 9. 2-D versions of multimodal benchmark functions.

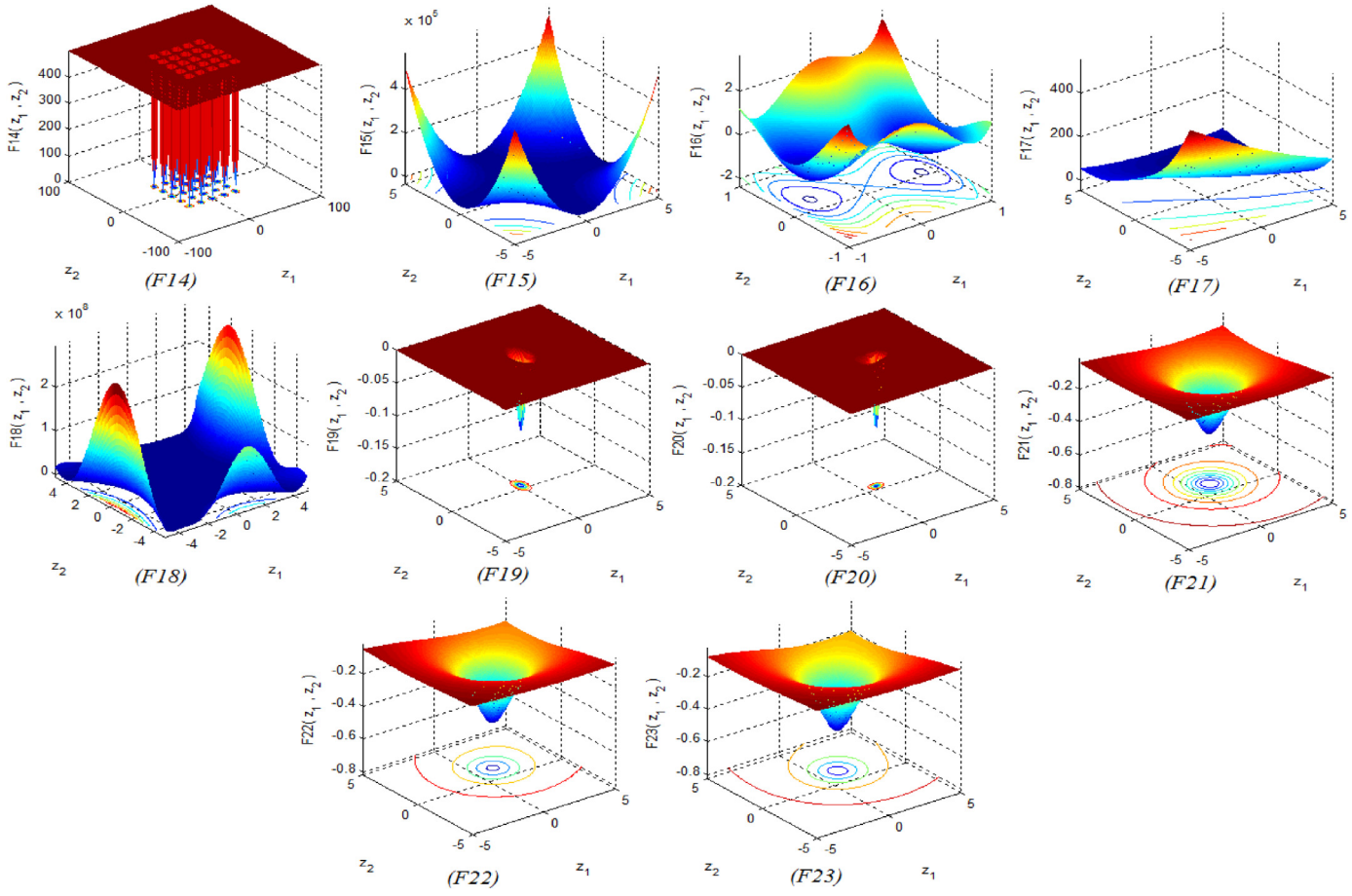

Fig. 10. 2-D versions of fixed-dimension multimodal benchmark functions.

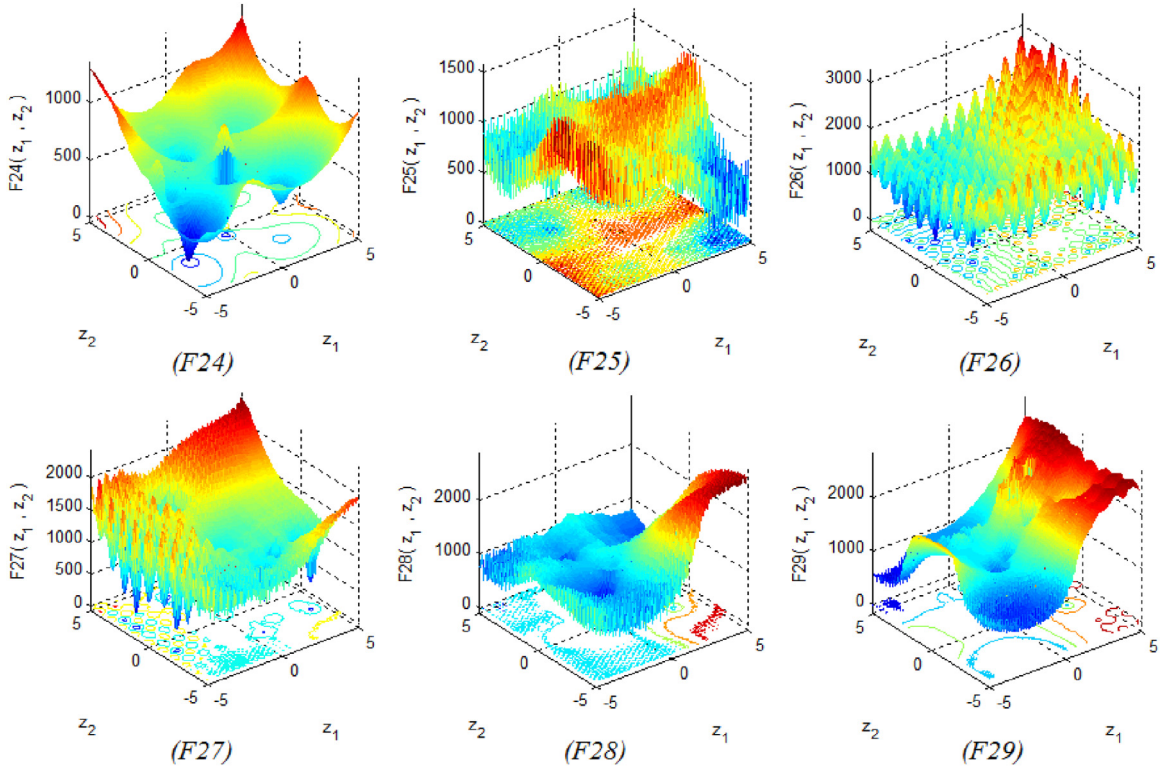

Fig. 11. 2-D versions of composite benchmark functions.

Table 1
Parameters values of competitor algorithms.

Algorithms	Parameters	Values
Spotted Hyena Optimizer (SHO)	Search Agents	30
	Control Parameter ($\vec{h}$)	[5, 0]
	$\vec{M}$ Constant	[0.5, 1]
	Number of Generations	1000
Grey Wolf Optimizer (GWO)	Search Agents	30
	Control Parameter ($\vec{a}$)	[2, 0]
	Number of Generations	1000
Particle Swarm Optimization (PSO)	Number of Particles	30
	Inertia Coefficient	0.75
	Cognitive and Social Coeff	1.8, 2
	Number of Generations	1000
Moth-Flame Optimization (MFO)	Search Agents	30
	Convergence Constant	[−1, −2]
	Logarithmic Spiral	0.75
	Number of Generations	1000
Multi-Verse Optimizer (MVO)	Search Agents	30
	Wormhole Existence Prob.	[0.2, 1]
	Travelling Distance Rate	[0.6, 1]
	Number of Generations	1000
Sine Cosine Algorithm (SCA)	Search Agents	30
	Number of Elites	2
	Number of Generations	1000
Gravitational Search Algorithm (GSA)	Search Agents	30
	Gravitational Constant	100
	Alpha Coefficient	20
	Number of Generations	1000
Genetic Algorithm (GA)	Crossover and Mutation	0.9, 0.05
	Population Size	30
	Number of Generations	1000
Harmony Search (HS)	Harmony Memory and Rate	30, 0.95
	Neighbouring Value Rate	0.30
	Discrete Set and Fret Width	17700, 1
	Number of Generations	1000

3.3. Performance comparison

In order to demonstrate the performance of proposed algorithm, its results are compared with eight well-known metaheuristic algorithms on benchmark test functions mentioned in the Section 3.1.

3.3.1. Evaluation of functions F1-F7 (exploitation)

The functions $F1 - F7$ are unimodal and allow to assess the exploitation capability of the metaheuristic based algorithms. Table 2 shows that SHO is very competitive as compared with other reported methods. In particular, SHO was the most efficient algorithm for functions $F1$, $F2$, $F3$, $F5$, and $F7$ and find the best optimal solution as evaluate to other metaheuristic based algorithms. The SHO algorithm can provide best exploitation.

3.3.2. Evaluation of functions F8-F23 (exploration)

Multimodal functions may include many local optima which can increase exponentially. These test problems have a capability to evaluate the exploration of an optimization algorithm.Tables 3 and 4 shows the results for functions $F8 - F23$ (multimodal and fixed-dimension multimodal functions) which can indicate that SHO has good exploration capability. According to these tables, SHO was most efficient in seven test problems (i.e., $F9$, $F11$, $F15$, $F16$, $F17$, $F18$, $F22$) and also competitive in other test problems. In fact, the SHO algorithm is the most efficient best algorithm in the majority of these test problems. These results show that the SHO algorithm has good worth regarding exploration.

3.3.3. Evaluation of functions F24-F29 (escape local minima)

Optimization in case of composite benchmark functions is very competitive task because of balance between exploration and exploitation which can keep away from the local optima problem. The local optima avoidance of an algorithm can be observed in such test functions due to the immense number of local optima.

Table 2
Results of unimodal benchmark functions.

F	SHO		GWO		PSO		MFO		MVO		SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F1	**0.00E+00**	**0.00E+00**	4.69E−59	7.30E−59	4.98E−09	1.40E−08	3.15E−04	5.99E−04	2.81E−01	1.11E−01	3.55E−02	1.06E−01	1.16E−16	6.10E−17	1.95E−12	2.01E−11	7.86E−10	8.11E−09
F2	**0.00E+00**	**0.00E+00**	1.20E−34	1.30E−34	7.29E−04	1.84E−03	3.71E+01	2.16E+01	3.96E−01	1.41E−01	3.23E−05	8.57E−05	1.70E−01	9.29E−01	6.53E−18	5.10E−17	5.99E−20	1.11E−17
F3	**0.00E+00**	**0.00E+00**	1.00E−14	4.10E−14	1.40E+01	7.13E+00	4.42E+03	3.71E+03	4.31E+01	8.97E+00	4.91E+03	3.89E+03	4.16E+02	1.56E+02	7.70E−10	7.36E−09	9.19E−05	6.16E−04
F4	7.78E−12	8.96E−12	**2.02E−14**	**2.43E−14**	6.00E−01	1.72E−01	6.70E+01	1.06E+01	8.80E−01	2.50E−01	1.87E+01	8.21E+00	1.12E+00	9.89E−01	9.17E+01	5.67E+01	8.73E−01	1.19E−01
F5	**8.59E+00**	**5.53E−01**	2.79E+01	1.84E+00	4.93E+01	3.89E+01	3.50E+03	3.98E+03	1.18E+02	1.43E+02	7.37E+02	1.98E+03	3.85E+01	3.47E+01	5.57E+02	4.16E+01	8.91E+02	2.97E+02
F6	2.46E−01	1.78E−01	6.58E−01	3.38E−01	9.23E−09	1.78E−08	1.66E−04	2.01E−04	3.15E−01	9.98E−02	4.88E+00	9.75E−01	1.08E−16	4.00E−17	3.15E−01	9.98E−02	**8.18E−17**	**1.70E−18**
F7	**3.29E−05**	**2.43E−05**	7.80E−04	3.85E−04	6.92E−02	2.87E−02	3.22E−01	2.93E−01	2.02E−02	7.43E−03	3.88E−02	5.79E−02	7.68E−01	2.77E+00	6.79E−04	3.29E−03	5.37E−01	1.89E−01

Table 3
Results of multimodal benchmark functions.

	SHO		GWO		PSO		MFO		MVO		SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F8	−1.16E+03	**2.72E+02**	−6.14E+03	9.32E+02	−6.01E+03	1.30E+03	**−8.04E+03**	8.80E+02	−6.92E+03	9.19E+02	−3.81E+03	2.83E+02	−2.75E+03	5.72E+02	−5.11E+03	4.37E+02	−4.69E+02	3.94E+02
F9	**0.00E+00**	**0.00E+00**	4.34E−01	1.66E+00	4.72E+01	1.03E+01	1.63E+02	3.74E+01	1.01E+02	1.89E+01	2.23E+01	3.25E+01	3.35E+01	1.19E+01	1.23E−01	4.11E+01	4.85E−02	3.91E+01
F10	2.48E+00	1.41E+00	**1.63E−14**	**3.14E−15**	3.86E−02	2.11E−01	1.60E+01	6.18E+00	1.15E+00	7.87E−01	1.55E+01	8.11E+00	8.25E−09	1.90E−09	5.31E−11	1.11E−10	2.83E−08	4.34E−07
F11	**0.00E+00**	**0.00E+00**	2.29E−03	5.24E−03	5.50E−03	7.39E−03	5.03E−02	1.74E−01	5.74E−01	1.12E−01	3.01E−01	2.89E−01	8.19E+00	3.70E+00	3.31E−06	4.23E−05	2.49E−05	1.34E−04
F12	3.68E−02	1.15E−02	3.93E−02	2.42E−02	**1.05E−10**	**2.06E−10**	1.26E+00	1.83E+00	1.27E+00	1.02E+00	5.21E+01	2.47E+02	2.65E−01	3.14E−01	9.16E−08	4.88E−07	1.34E−05	6.23E−04
F13	9.29E−01	9.52E−02	4.75E−01	2.38E−01	4.03E−03	5.39E−03	7.24E−01	1.48E+00	6.60E−02	4.33E−02	2.81E+02	8.63E+02	**5.73E−32**	**8.95E−32**	6.39E−02	4.49E−02	9.94E−08	2.61E−07

Table 4
Results of fixed-dimension multimodal benchmark functions.

F	SHO		GWO		PSO		MFO		MVO		SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F14	9.68E+00	3.29E+00	3.71E+00	3.86E+00	2.77E+00	2.32E+00	2.21E+00	1.80E+00	**9.98E−01**	**9.14E−12**	1.26E+00	6.86E−01	3.61E+00	2.96E+00	4.39E+00	4.41E−02	6.79E+00	1.12E+00
F15	**9.01E−04**	**1.06E−04**	3.66E−03	7.60E−03	9.09E−04	2.38E−04	1.58E−03	3.50E−03	7.15E−03	1.26E−02	1.01E−03	3.75E−04	6.84E−03	7.37E−03	7.36E−03	2.39E−04	5.15E−03	3.45E−04
F16	**−1.06E+01**	**2.86E−011**	−1.03E+00	7.02E−09	−1.03E+00	0.00E+00	−1.03E+00	0.00E+00	−1.03E+00	4.74E−08	−1.03E+00	3.23E−05	−1.03E+00	0.00E+00	−1.04E+00	4.19E−07	−1.03E+00	3.64E−08
F17	**3.97E−01**	2.46E−01	3.98E−01	7.00E−07	3.97E−01	9.03E−16	3.98E−01	1.13E−16	3.98E−01	1.15E−07	3.99E−01	7.61E−04	3.98E−01	1.13E−16	3.98E−01	**3.71E−17**	3.99E−01	9.45E−15
F18	**3.00E+00**	9.05E+00	3.00E+00	7.16E−06	3.00E+00	6.59E−05	3.00E+00	**4.25E−15**	5.70E+00	1.48E+01	3.00E+00	2.25E−05	3.01E+00	3.24E−02	3.01E+00	6.33E−07	3.00E+00	1.94E−10
F19	−3.75E+00	4.39E−01	−3.86E+00	1.57E−03	**3.90E+00**	3.37E−15	−3.86E+00	**3.16E−15**	−3.86E+00	3.53E−07	−3.86E+00	2.55E−03	−3.22E+00	4.15E−01	−3.30E+00	4.37E−10	−3.29E+00	9.69E−04
F20	−1.44E+00	5.47E−01	−3.27E+00	7.27E−02	**−3.32E+00**	2.66E−01	−3.23E+00	6.65E−02	−3.23E+00	**5.37E−02**	−2.84E+00	3.71E−01	−1.47E+00	5.32E−01	−2.39E+00	4.37E−01	−2.17E+00	1.64E−01
F21	−2.08E+00	**3.80E−01**	**−9.65E+00**	1.54E+00	−7.54E+00	2.77E+00	−6.20E+00	3.52E+00	−7.38E+00	2.91E+00	−2.28E+00	1.80E+00	−4.57E+00	1.30E+00	−5.19E+00	2.34E+00	−7.33E+00	1.29E+00
F22	**−1.61E+01**	**2.04E−04**	−1.04E+01	2.73E−04	−8.55E+00	3.08E+00	−7.95E+00	3.20E+00	−8.50E+00	3.02E+00	−3.99E+00	1.99E+00	−6.58E+00	2.64E+00	−2.97E+00	1.37E−02	−1.00E+00	2.89E−04
F23	−1.68E+00	2.64E−01	**−1.05E+01**	**1.81E−04**	−9.19E+00	2.52E+00	−7.50E+00	3.68E+00	−8.41E+00	3.13E+00	−4.49E+00	1.96E+00	−9.37E+00	2.75E+00	−3.10E+00	2.37E+00	−2.46E+00	1.19E+00

Table 5
Results of composite benchmark functions.

F	SHO		GWO		PSO		MFO		MVO		SCA		GSA		GA		HS	
	Ave	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*	*Ave*	*Std*
F24	2.30E+02	1.37E+02	8.39E+01	8.42E+01	6.00E+01	8.94E+01	1.18E+02	7.40E+01	1.40E+02	1.52E+02	1.20E+02	3.11E+01	**4.49E−17**	**2.56E−17**	5.97E+02	1.34E+02	4.37E+01	2.09E+01
F25	4.08E+02	9.36E+01	1.48E+02	3.78E+01	2.44E+02	1.73E+02	**9.20E+01**	1.36E+02	2.50E+02	1.44E+02	1.14E+02	**1.84E+00**	2.03E+02	4.47E+02	4.09E+02	2.10E+01	1.30E+02	3.34E+00
F26	**3.39E+02**	**3.14E+01**	3.53E+02	5.88E+01	3.39E+02	8.36E+01	4.19E+02	1.15E+02	4.05E+02	1.67E+02	3.89E+02	5.41E+01	3.67E+02	8.38E+01	9.30E+02	8.31E+01	5.07E+02	4.70E+01
F27	7.26E+02	1.21E+02	4.23E+02	1.14E+02	4.49E+02	1.42E+02	**3.31E+02**	**2.09E+01**	3.77E+02	1.28E+02	4.31E+02	2.94E+01	5.32E+02	1.01E+02	4.97E+02	3.24E+01	3.08E+02	2.07E+02
F28	**1.06E+02**	**1.38E+01**	1.36E+02	2.13E+02	2.40E+02	4.25E+02	1.13E+02	9.27E+01	2.45E+02	9.96E+01	1.56E+02	8.30E+01	1.44E+02	1.31E+02	1.90E+02	5.03E+01	8.43E+02	6.94E+02
F29	**5.97E+02**	**4.98E+00**	8.26E+02	1.74E+02	8.22E+02	1.80E+02	8.92E+02	2.41E+01	8.33E+02	1.68E+02	6.06E+02	1.66E+02	8.13E+02	1.13E+02	6.65E+02	3.37E+02	7.37E+02	2.76E+01

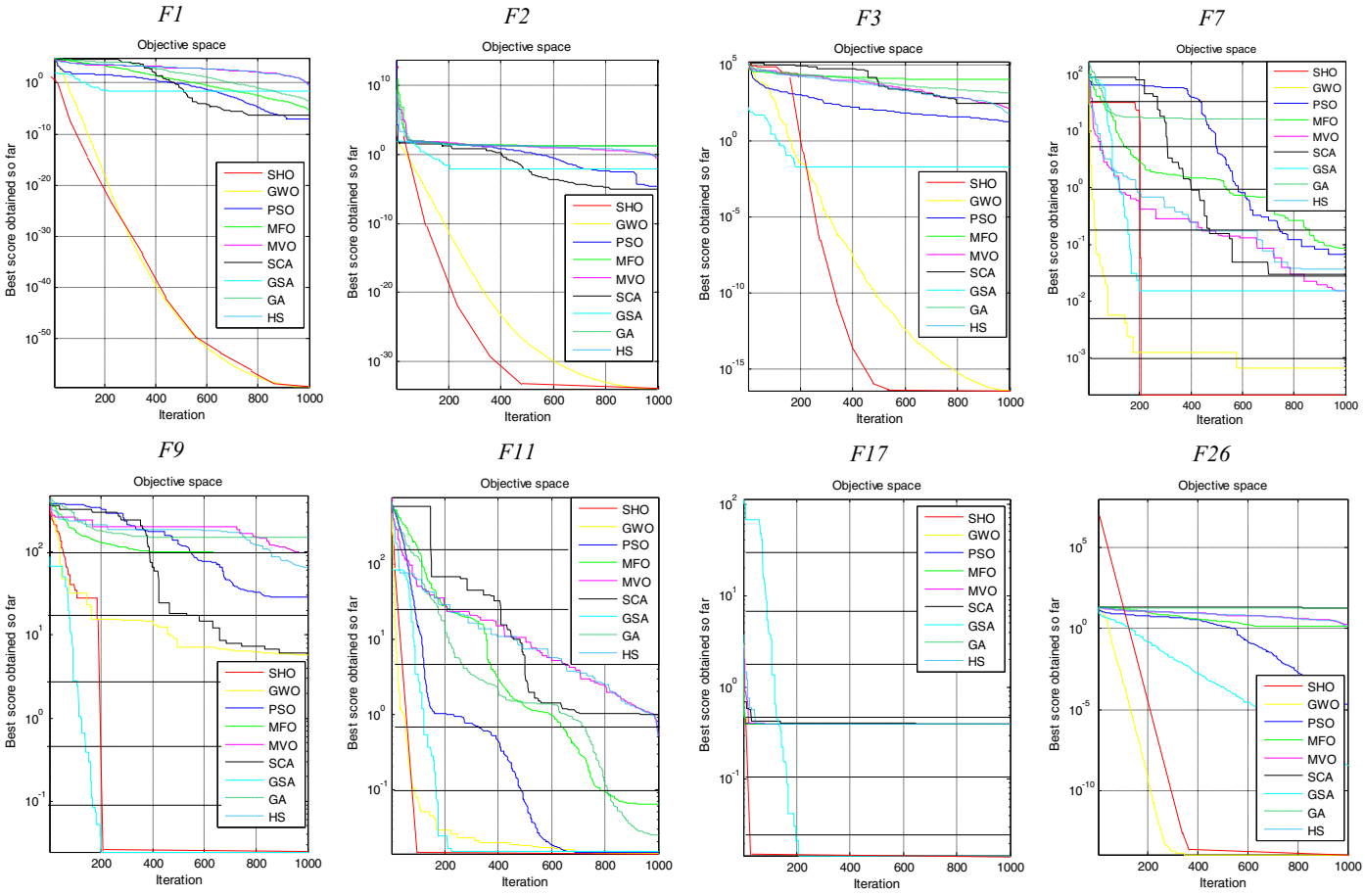

Fig. 12. Comparison of convergence curves of SHO algorithm and proposed algorithms obtained in some of the benchmark problems.

Table 5 shows that the SHO algorithm was the best optimizer for functions *F*26, *F*28, and *F*29 and very competitive in the other cases.

The results for functions $F1 - F29$ show that SHO is the best efficient algorithm as compared with other optimization methods. To further examine the performance of the proposed algorithm, six classical engineering design problems are discussed in the following sections.

3.4. Convergence analysis

The intention behind the convergence analysis is to increase the better understanding of different explorative and exploitative mechanisms in SHO algorithm. It is observed that SHO's search agents explore the promising regions of design space and exploit the best one. These search agents change rapidly in beginning stages of the optimization process and then increasingly converge. The convergence curves of SHO, GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS are provided and compared in Fig. 12. It can be observed that SHO is best competitive with other state-of-the-art metaheuristic algorithms. When optimizing the test functions, SHO shows three different convergence behaviors. In the initial steps of iteration, SHO converges more rapidly for the promising regions of search space due to its adaptive mechanism. This behavior is apparent in test functions *F*1, *F*3, *F*9, *F*11, and *F*26. In second step SHO converges towards the optimum only in final iterations which are experiential in *F*7 test function. The last step is the express convergence from the initial stage of iterations as seen in *F*2 and *F*17. These results show that the SHO algorithm maintains a balance of exploration and exploitation to find the global optima. Overall, the results of this section exposed different trait of the proposed SHO algorithm and seems that the success rate of the SHO algorithm is high in solving optimization problems. The re-positioning mechanism of spotted hyenas using Eq. (10) for high exploitation which allows the spotted hyenas move around the prey in the group towards the best solution which are obtained so far. Since SHO shows the local optima avoidance during iteration process. However, the other methods such as GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS has a low success rate compared to SHO algorithm as shown in Fig. 12.

3.5. Scalability study

The next experiment is performed to see the effect of scalability on various benchmark test functions. The dimensionality of the test functions are changed from 30 to 50, 50 to 80, and 80–100. As shown in Fig. 13, the results indicate that the performance of SHO algorithm provides different behaviors on various dimensionality. The performance of the proposed algorithm degraded as depicted in Fig. 13. It is also observed that the degradation in the performance is not too much. Therefore, it shows the applicability of proposed algorithm on the highly scalable environment.

3.6. Statistical testing

Besides the basic statistical analysis (i.e., mean and standard deviation), ANOVA test has been conducted for comparison of above mentioned algorithms. The ANOVA test has been used to test whether the results obtained from proposed algorithms differ from

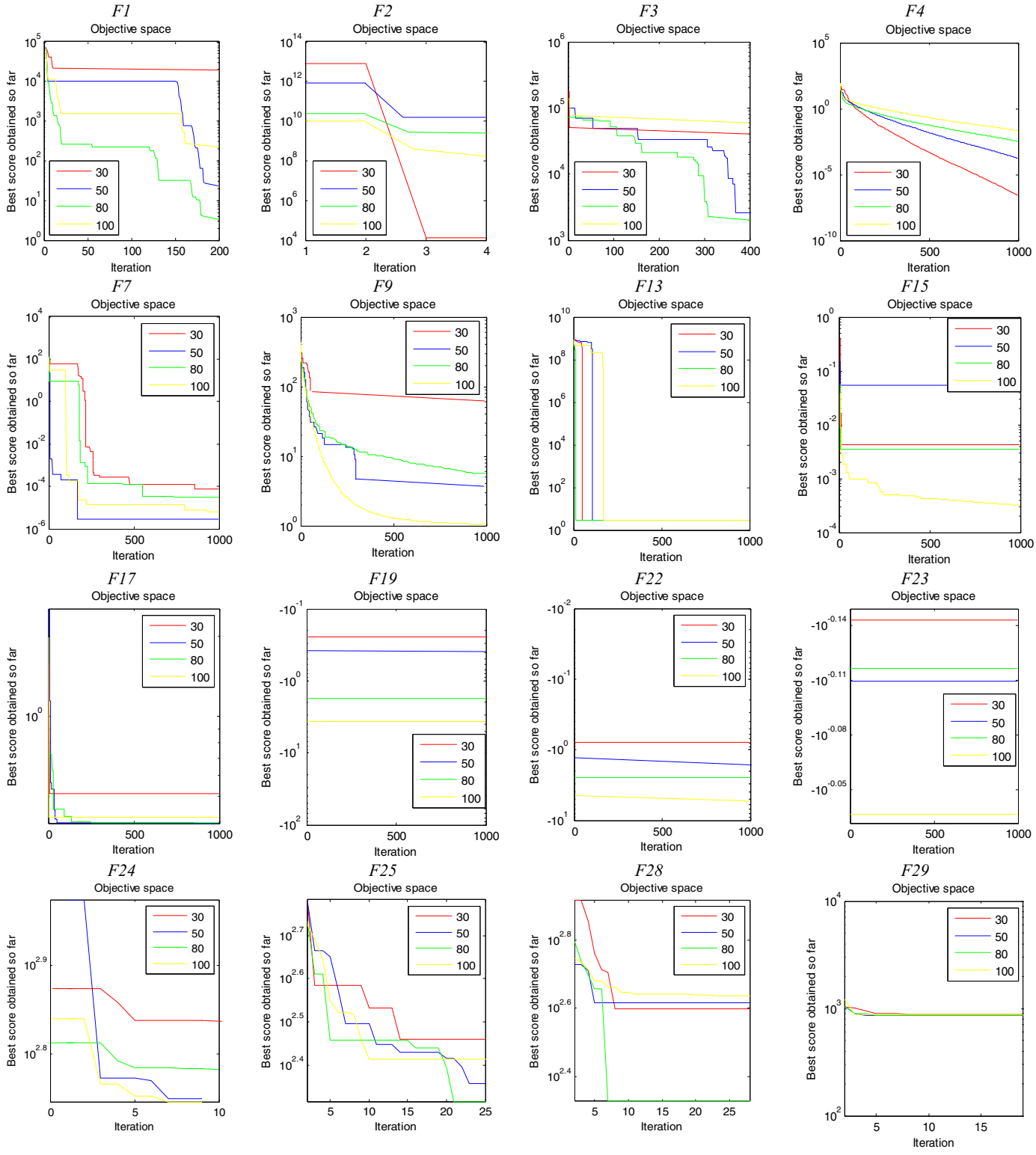

Fig. 13. Effect of scalability analysis of SHO algorithm.

the results of other competitive algorithms in a statistically significant way. We have taken 30 as the sample size for ANOVA test. We have used 95% confidence for ANOVA test. The result analysis of ANOVA test for benchmark functions is tabulated in Table 6. The results reveal that the proposed algorithm is statistically significant as compared to other competitor algorithms.

4. SHO for engineering problems

In this section, SHO was tested with five constrained and one unconstrained engineering design problems: welded beam, tension/compression spring, pressure vessel, speed reducer, rolling bearing element, and displacement of loaded structure. These

Table 6
ANOVA test results.

F	*P*−value	SHO	GWO	PSO	MFO	MVO	SCA	GSA	GA	HS
F1	1.92E−74	GWO, PSO, MFO, MVO, GSA, GA HS	SHO, PSO, MFO, MVO, GSA, GA HS	SHO, GWO, MFO, MVO, SCA, GSA GA, HS	SHO, GWO, PSO, MVO, SCA, GSA GA	SHO, GWO, PSO, MFO, SCA, GSA GA	SHO, GWO, PSO, MFO, MVO, GSA GA, HS	SHO, PSO, MFO, MVO, SCA, GA HS	SHO, GWO, PSO, MFO, MVO, SCA GSA	SHO, GWO, PSO, MFO, MVO, SCA GSA, GA
F2	3.61E−72	GWO, PSO, MFO,MVO, SCA, GSA,GA, HS	SHO, PSO, MFO, GSA, GA, HS	SHO, GWO, MFO, SCA, GSA, GA, HS	SHO, GWO, MVO, SCA, GSA, GA,HS	SHO, GWO, PSO, MFO, GSA, GA,HS	SHO, GWO, PSO, MFO, MVO, GSA,HS	SHO, GWO, PSO, MFO, MVO, SCA,HS	SHO, GWO, MFO,MVO, SCA, GSA,HS	SHO, GWO, PSO, MFO, MVO, SCA, GSA
F3	3.56E−38	GWO, PSO, MFO,MVO, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GSA	SHO, GWO, MFO, MVO, SCA, GSA	SHO, GWO, PSO, MVO, SCA,GSA,GA	SHO, GWO, PSO, MFO, SCA, GSA,HS	SHO, PSO, MFO, MVO, GSA, GA	SHO, GWO, PSO, MFO, MVO, SCA	SHO, PSO, MFO,MVO, SCA, GSA,HS	SHO, GWO, MVO, SCA, GSA, GA
F4	2.85E−174	GWO, PSO, MVO,SCA, GSA, GA,HS	SHO, PSO, MFO, MVO, SCA, GSA,GA, HS	SHO, GWO, MFO, MVO, SCA, HS	SHO, PSO, MVO, SCA, GSA, HS	SHO, GWO, PSO, MFO, SCA, GSA,GA, HS	SHO, PSO, MFO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO, MVO, GA,HS	SHO, GWO, PSO, MFO, MVO, SCA,HS	SHO, PSO, MFO, MVO, SCA, GSA,GA
F5	2.16E−22	GWO, PSO, MFO,MVO, SCA, GSA,GA, HS	SHO, PSO, MFO, MVO, GSA, GA,HS	SHO, GWO, MVO, GSA, GA, HS	SHO, GWO, PSO, SCA, GSA, HS	SHO, PSO, MFO, SCA, GSA, GA,HS	SHO, GWO, PSO, MFO, MVO, GSA,GA	SHO, GWO, PSO, MVO, SCA,GA,HS	SHO, GWO, PSO, MFO, MVO, GSA,HS	SHO, GWO, PSO, MFO, MVO, SCA
F6	1.97E−147	GWO, PSO, MFO,MVO, SCA, GSA,GA, HS	SHO, PSO, MFO, MVO, GSA, GA,HS	SHO, GWO, MFO, SCA, GSA, GA,HS	SHO, GWO,PSO, MVO, SCA, GSA,HS	SHO, GWO, PSO, MFO, GSA, GA, HS	SHO, GWO, PSO, MFO, MVO, GSA,HS	SHO, GWO, MFO, MVO, SCA,GA,HS	SHO, GWO, MFO, MVO, SCA, GSA,HS	SHO, GWO, PSO, MFO, MVO, SCA, GA
F7	1.53E−01	GWO, PSO, MFO,MVO, SCA, GSA, GA	SHO, MFO, MVO, SCA, GA, HS	SHO, MFO, MVO, SCA, GSA, GA,HS	SHO, GWO, PSO, MVO, SCA, GA,HS	SHO, GWO, MFO, SCA, GSA, GA,HS	SHO, GWO, MFO, MVO, GSA, GA,HS	SHO, GWO, PSO, MVO, SCA, GA,HS	SHO, PSO, MFO, MVO, SCA, GSA,HS	SHO, GWO, PSO,MFO, SCA, GSA,GA
F8	1.56E−122	GWO, PSO, MFO,MVO, SCA, HS	SHO, MFO, MVO, SCA, GSA, GA,HS	SHO, MVO, SCA, GSA, GA, HS	SHO, GWO, PSO,MVO, GSA, GA,HS	SHO, GWO, PSO, SCA, GSA, GA,HS	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO, MVO, SCA, GSA, HS	SHO, GWO, PSO, MFO, MVO, SCA, GSA	SHO, PSO, MFO, MVO, SCA, GSA, GA
F9	1.87E−112	GWO, PSO, MFO, GSA, GA, HS	SHO, PSO, MFO, MVO, GSA, GA,HS	SHO, GWO, MFO, SCA, GSA, GA	SHO, GWO, PSO,SCA, GSA, GA,HS	SHO, GWO, PSO, MFO, SCA, GSA,GA	SHO, PSO, MFO, MVO, GSA, GA,HS	SHO, PSO, MFO, MVO, SCA, GA,HS	SHO, GWO, PSO, MFO, MVO, SCA,GSA	SHO, GWO, PSO, SCA, GSA, GA
F10	1.28E−86	GWO, PSO, MFO, SCA, GSA, GA,HS	SHO, PSO, MFO, MVO, SCA, GSA,HS	SHO, GWO, MFO, MVO, SCA, GSA,GA, HS	SHO, MVO, SCA, GSA, GA, HS	SHO, GWO, PSO, MFO, SCA, GSA	SHO, GWO, PSO, MFO, MVO, GSA, GA	SHO, PSO, MFO, MVO, SCA, GA,HS	SHO, GWO, MFO, MVO, SCA, GSA,HS	SHO, GWO, PSO, MVO, SCA, GSA,GA
F11	1.86E−77	GWO, PSO, MFO, MVO, SCA, GSA, GA, HS	SHO, PSO, MFO, MVO, GSA, GA, HS	SHO, MFO, SCA, GSA, GA, HS	SHO,GWO, PSO, MVO, SCA, GSA,GA	SHO, GWO, PSO, MFO, SCA, GSA,HS	SHO, GWO, PSO, MFO, MVO, GSA	SHO, GWO, MFO, MVO, SCA,GA,HS	SHO, GWO, MVO, SCA, GSA, HS	SHO, PSO, MFO, MVO, SCA, GSA,GA
F12	2.39E−01	GWO, PSO, MVO, SCA, GSA, HS	SHO, PSO, SCA, GSA, GA, HS	SHO, GWO, MFO, MVO, SCA, GSA,HS	SHO, GWO, PSO, MVO, SCA, GA,HS	SHO, PSO, MFO, SCA, GSA, HS	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO,MVO, SCA,GA	SHO, GWO, PSO, MVO, SCA, GSA,HS	SHO, PSO, MFO, MVO, SCA, GSA,GA
F13	1.90E−03	PSO, MFO, SCA, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GSA	SHO, MFO, MVO, SCA, GSA, HS	SHO, GWO, MVO, SCA,GSA, GA,HS	SHO, GWO, MFO, SCA, GSA, GA, HS	SHO, GWO, PSO, MFO, MVO, GSA	SHO, GWO, PSO, SCA, GA, HS	SHO, GWO, PSO,MFO, MVO, SCA	SHO, GWO, MVO, SCA, GSA, GA
F14	6.37E−47	GWO, PSO, MFO,MVO, SCA, GSA,GA, HS	SHO, PSO, MFO, MVO, SCA, GSA, GA, HS	SHO, GWO, MFO, MVO, SCA, GSA,GA	SHO, PSO, SCA, GSA, GA, HS	SHO, GWO, PSO, MFO, SCA, GSA,HS	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, MFO, MVO, SCA,GA, HS	SHO, GWO, PSO, MFO, MVO, GSA,HS	SHO, GWO, MFO, MVO, SCA, GSA, GA
F15	1.15E−05	GWO, PSO, MVO,SCA, GSA, GA	SHO, PSO, MVO, SCA, GSA, HS	SHO, GWO, MFO, MVO, GSA, GA,HS	SHO, PSO, MVO, SCA, GSA, GA,HS	SHO, GWO, PSO, MFO, SCA, GSA,GA, HS	SHO, GWO, PSO, MFO, MVO, GSA,GA	SHO, GWO, PSO, MVO, SCA, GA, HS	SHO, GWO, MFO, MVO, SCA, GSA,HS	SHO, PSO, MFO, MVO, SCA, GSA, GA
F16	2.66E−40	GWO, PSO, MFO,MVO, SCA, GSA,GA, HS	SHO, PSO, MFO, MVO, SCA, GA,HS	SHO, GWO, MFO, MVO, SCA, GA,HS	SHO, GWO, PSO, MVO, SCA, GSA,GA	SHO, GWO, PSO, MFO, GSA, GA, HS	SHO, GWO, PSO, MFO, MVO, GSA,GA	SHO, GWO, MFO, MVO, SCA, GA,HS	SHO, PSO, MFO, MVO, SCA, GSA,HS	SHO, GWO, PSO, SCA, GSA, GA

(*continued on next page*)

Table 6 (*continued*)

F	*P*−value	SHO	GWO	PSO	MFO	MVO	SCA	GSA	GA	HS
F17	1.81E−12	GWO, PSO, MFO, MVO, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GSA,HS	SHO, GWO, MFO, SCA, GSA, GA, HS	SHO, GWO, PSO, SCA, GSA, GA,HS	SHO, GWO, PSO, SCA, GSA, GA,HS	SHO, GWO, PSO, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GA,HS	SHO, GWO, PSO, MVO, SCA, GSA,HS	SHO, GWO, PSO, MFO, MVO, SCA,GSA
F18	4.60E−02	MFO, MVO, SCA, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GSA,HS	SHO, GWO, MFO, MVO, SCA, GSA	SHO, GWO, PSO, MVO, SCA, GSA,GA	SHO, GWO, MFO, SCA, GSA, GA, HS	SHO, PSO, MFO, MVO, GSA, GA, HS	SHO, GWO, PSO, MFO, MVO, SCA,HS	SHO, GWO, PSO, MFO, SCA, GSA,HS	SHO, GWO, PSO, MFO, MVO, SCA, GA
F19	9.51E−45	GWO, PSO, MFO, MVO, SCA, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GSA, GA	SHO, GWO, MFO, SCA, GSA, GA,HS	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO, SCA, GSA,GA	SHO, GWO, MFO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO, MVO, SCA	SHO, GWO, PSO, MFO, MVO, SCA,GSA, HS	SHO, GWO, PSO, SCA, GSA, GA
F20	1.97E−98	GWO, MFO, MVO, SCA, GSA, GA,HS	SHO, MFO, MVO, SCA, GA, HS	SHO, GWO, MFO, MVO, SCA, GSA,HS	SHO, PSO, MVO, SCA, GSA, GA,HS	SHO, GWO, PSO, MFO, SCA, GA,HS	SHO, GWO, PSO, MFO, SCA, GA,HS	SHO, PSO, MFO, MVO, SCA, GA, HS	SHO, GWO, PSO, MFO, MVO, GSA, HS	SHO, GWO, MFO, MVO, SCA, GSA, GA
F21	2.55E−31	GWO, PSO, MFO, MVO, SCA, GA,HS	SHO, PSO, MVO, SCA, GSA, GA,HS	SHO, GWO, MFO, MVO, SCA, GSA	SHO, GWO, PSO, MVO, SCA, GSA,HS	SHO, PSO, SCA, GSA, GA, HS	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, GWO, MVO, SCA, GA, HS	SHO, GWO, PSO, MFO, MVO, SCA,GSA	SHO, GWO, PSO, MVO, SCA, GSA,GA
F22	8.10E−31	GWO, PSO, MFO, MVO, SCA, GSA,GA	SHO, PSO, MFO, SCA, GSA, GA,HS	SHO, MFO, MVO, GSA, GA, HS	SHO, GWO, PSO, GSA, GA, HS	SHO, PSO, MFO, SCA, GSA, GA, HS	SHO, GWO, PSO, MFO, MVO, GSA, GA	SHO, GWO, MFO, MVO, SCA, GA, HS	SHO, GWO, PSO, SCA, GSA, HS	SHO, MFO, MVO, SCA, GSA, GA
F23	2.79E−33	PSO, MFO, MVO, SCA, GSA, GA	SHO, PSO, MFO, GSA, GA, HS	SHO, MFO, MVO, SCA, GSA, GA,HS	SHO, GWO, PSO, SCA, GSA, GA,HS	SHO, GWO, PSO, MFO, SCA, GSA,GA	SHO, GWO, PSO, MFO, MVO, GA,HS	SHO, GWO, PSO, MFO, MVO, SCA, HS	SHO, GWO, PSO, MFO, MVO, SCA	SHO, GWO, PSO, MFO, MVO, SCA, GSA
F24	5.50E−03	GWO, PSO, MFO, MVO, SCA, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GSA, HS	SHO, GWO, MFO, MVO, SCA, GSA, HS	SHO, GWO, PSO, MVO, SCA, GSA, HS	SHO, GWO, PSO, MFO, SCA, GSA	SHO, GWO, PSO, MVO, GSA, GA, HS	SHO, GWO, MFO, MVO, SCA, GA, HS	SHO, GWO, PSO, MFO, MVO, SCA, HS	SHO, GWO, PSO, MFO, MVO, SCA
F25	1.20E−03	GWO, PSO, MFO, MVO, SCA, GSA,HS	SHO, PSO, MFO, MVO, SCA, GSA	SHO, GWO, MFO, MVO, SCA, GA,HS	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO, SCA, GA, HS	SHO, GWO, PSO, MFO, MVO, GSA,HS	SHO, PSO, MFO, MVO, SCA, GA, HS	SHO, GWO, PSO, MVO, SCA, GSA,HS	SHO, GWO, MFO, MVO, SCA, GSA,GA
F26	7.60E−03	GWO, PSO, SCA, GSA, GA, HS	SHO, PSO, SCA, GSA, GA, HS	SHO, GWO, MFO, SCA, GSA, GA,HS	SHO, GWO, PSO, MVO, SCA, GSA, HS	SHO, GWO, PSO, SCA, GSA, GA,HS	SHO, GWO, PSO, MFO, GSA, GA,HS	SHO, GWO, MFO, MVO, SCA, GA,HS	SHO, MFO, MVO, SCA, GSA, HS	SHO, GWO, PSO, MVO, SCA, GSA,GA
F27	2.48E−02	GWO, PSO, MFO, MVO, SCA, GSA, GA, HS	SHO, PSO, MFO, MVO, SCA, GSA, GA, HS	SHO, GWO, MFO, MVO, SCA, GSA,HS	SHO, GWO, PSO, SCA, GSA, GA,HS	SHO, PSO, MFO, SCA, GSA, GA,HS	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, MFO, MVO, SCA, GA, HS	SHO, GWO, PSO, MVO, SCA, GSA, HS	SHO, MVO, SCA,GSA, GA
F28	2.32E−01	GWO, MFO, MVO, SCA, GSA, HS	SHO, PSO, MVO, GSA, GA, HS	SHO, MFO, MVO, GSA, GA, HS	SHO, GWO, PSO, MVO, SCA, GSA,GA	SHO, GWO, MFO, SCA, GSA, GA,HS	SHO, PSO, MFO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO, MVO, SCA,GA	SHO, GWO, PSO, MFO, MVO, SCA,GSA	SHO, PSO, MFO, SCA, GA
F29	1.13E−01	PSO, MFO, MVO, SCA, GSA, GA,HS	SHO, PSO, MFO, MVO, SCA, GSA,GA	SHO, GWO, MFO, MVO, SCA, GSA,GA	SHO, GWO, PSO, MVO, GSA, GA,HS	SHO, GWO, PSO, MFO, SCA, GSA,GA	SHO, GWO, PSO, MFO, MVO, GSA,HS	SHO, GWO, PSO, MVO, SCA, GA,HS	SHO, GWO, PSO, MFO, MVO, GSA, HS	SHO, MFO, MVO, SCA, GSA

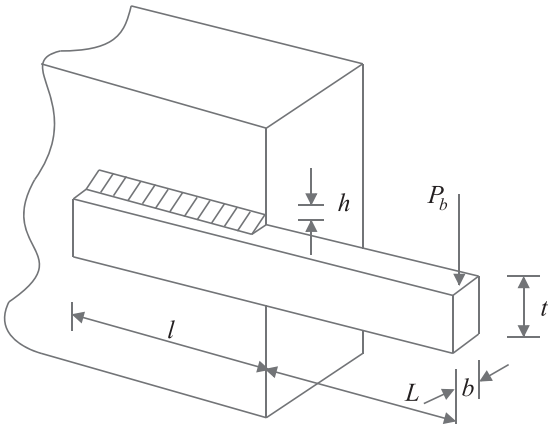

Fig. 14. Schematic view of welded beam problem.

problems have different constraints, so to optimize these problems we need to employ the constraint handling method. There are different types of penalty functions to handle constraint problems [8]:

- **Static penalty** does not depend on the current generation number and always remains constant during the entire computational process.
- **Dynamic penalty** in which the current generation is involved in the computation of the equivalent penalty factors (increases over time i.e., generations).
- **Annealing penalty** coefficients are changed only once in many courses of iterations. Only active constraints are considered, which are not trapped in local optima, at each iteration.
- **Adaptive penalty** takes the feedback from the previous search process and only changes if the feasible/infeasible solution is considered as a best in the population.
- **Co-evolutionary penalty** is split into two values (i.e., coeff and viol) to find out the constraints which are violated and also corresponding amounts of violation.
- **Death penalty** handles a solution which can violate the constraint and assigns the fitness value of zero. It can discard the infeasible solutions during optimization.
  However, this method does not employ the information of infeasible solutions which are further helpful to solve the problems with dominated infeasible regions. Due to its simplicity and low computational cost, the SHO algorithm is equipped with death penalty function in this section to handle constraints.

4.1. Welded beam design

The main objective of this problem is to minimize the fabrication cost of the welded beam as shown in Fig. 14. The optimization constraints are shear stress (τ) and bending stress (θ) in the beam, buckling load (P_c) on the bar, end deflection (δ) of the beam. There are four optimization variables of this problem which are as follows:

- Thickness of weld (h)
- Length of the clamped bar (l)
- Height of the bar (t)
- Thickness of the bar (b)

Table 7
Comparison results for welded beam design.

Algorithms	Optimum variables				Optimum cost
	h	t	l	b	
SHO	0.205563	3.474846	9.035799	0.205811	**1.725661**
GWO	0.205678	3.475403	9.036964	0.206229	1.726995
PSO	0.197411	3.315061	10.00000	0.201395	1.820395
MFO	0.203567	3.443025	9.230278	0.212359	1.732541
MVO	0.205611	3.472103	9.040931	0.205709	1.725472
SCA	0.204695	3.536291	9.004290	0.210025	1.759173
GSA	0.147098	5.490744	10.00000	0.217725	2.172858
GA	0.164171	4.032541	10.00000	0.223647	1.873971
HS	0.206487	3.635872	10.00000	0.203249	1.836250

Table 8
Comparison of SHO statistical for the welded beam design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
SHO	**1.725661**	**1.725828**	1.726064	**0.000287**	**1.725787**
GWO	1.726995	1.727128	1.727564	0.001157	1.727087
PSO	1.820395	2.230310	**3.048231**	0.324525	2.244663
MFO	1.732541	1.775231	1.802364	0.012397	1.812453
MVO	1.725472	1.729680	1.741651	0.004866	1.727420
SCA	1.759173	1.817657	1.873408	0.027543	1.820128
GSA	2.172858	2.544239	3.003657	0.255859	2.495114
GA	1.873971	2.119240	2.320125	0.034820	2.097048
HS	1.836250	1.363527	2.035247	0.139485	1.9357485

The mathematical formulation is described as follows:

$$
\begin{aligned}
&\text{Consider } \vec{z} = [z_1 z_2 z_3 z_4] = [hltb],\\
&\text{Minimize } f(\vec{z}) = 1.10471 z_1^2 z_2 + 0.04811 z_3 z_4 (14.0 + z_2),\\
&\text{Subject to}\\
&g_1(\vec{z}) = \tau(\vec{z}) - \tau_{max} \le 0,\\
&g_2(\vec{z}) = \sigma(\vec{z}) - \sigma_{max} \le 0,\\
&g_3(\vec{z}) = \delta(\vec{z}) - \delta_{max} \le 0,\\
&g_4(\vec{z}) = z_1 - z_4 \le 0,\\
&g_5(\vec{z}) = P - P_c(\vec{z}) \le 0,\\
&g_6(\vec{z}) = 0.125 - z_1 \le 0,\\
&g_7(\vec{z}) = 1.10471 z_1^2 + 0.04811 z_3 z_4 (14.0 + z_2) - 5.0 \le 0,
\end{aligned}
\tag{11}
$$

Variable range

$$
\begin{aligned}
0.05 \le z_1 \le 2.00,\\
0.25 \le z_2 \le 1.30,\\
2.00 \le z_3 \le 15.0,
\end{aligned}
$$

The optimization methods previously applied to this problem include GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS. The comparison for the best solution obtained by such algorithms is presented in Table 7.

The results indicate that SHO converged to the best design. The comparison of the statistical results is given in Table 8. The results show that SHO again performs better in mean (average), median, standard deviation and also requires less number of analyses to find the best optimal design. By observing Fig. 15, SHO achieve the near optimal solution in the initial steps of iterations. The success rate of SHO algorithm is high as compared to other methods of solving optimization problems.

4.2. Tension/compression spring design problem

The objective of this problem is to minimize the tension/ compression spring weight as shown in Fig. 16. The optimization constraints of this problem are:

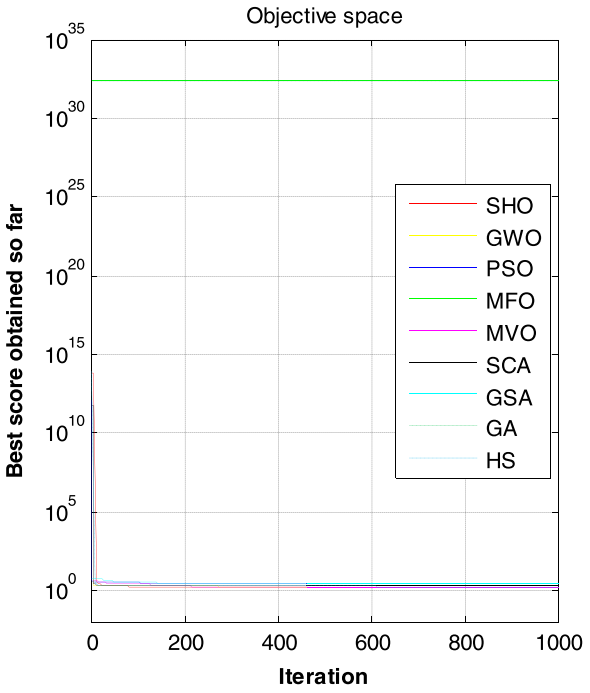

Fig. 15. Analysis of SHO with various techniques for the welded beam problem.

Fig. 16. Schematic view of tension/compression spring problem.

- Shear stress.
- Surge frequency.
- Minimum deflection.

There are three design variables of this problem: such as wire diameter (d), mean coil diameter (D), and the number of active coils (N). The mathematical formulation of this problem is described as follows:

$$\text{Consider } \vec{z} = [z_1 z_2 z_3] = [dDN],$$
$$\text{Minimize } f(\vec{z}) = (z_3 + 2) z_2 z_1^2,$$
$$\text{Subject to}$$
$$g_1(\vec{z}) = 1 - \frac{z_2^3 z_3}{71785 z_1^4} \leq 0,$$
$$g_2(\vec{z}) = \frac{4z_2^2 - z_1 z_2}{12566(z_2 z_1^3 - z_1^4)} + \frac{1}{5108 z_1^2} \leq 0,$$
$$g_3(\vec{z}) = 1 - \frac{140.45 z_1}{z_2^2 z_3} \leq 0,$$
$$g_4(\vec{z}) = \frac{z_1 + z_2}{1.5} - 1 \leq 0, \qquad (12)$$

The comparison of best optimal solution among several algorithms is given in Table 9. This problem has been tested with different optimization methods such as GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS. The comparison for the best solution obtained by such algorithms is presented in Table 9. The results indicate that SHO performs better than other optimization methods. The statistical results of tension/compression spring design is given in

Table 9
Comparison results for tension/compression spring design.

Algorithms	Optimum variables			Optimum cost
	d	D	N	
SHO	0.051144	0.343751	12.0955	**0.012674000**
GWO	0.050178	0.341541	12.07349	0.012678321
PSO	0.05000	0.310414	15.0000	0.013192580
MFO	0.05000	0.313501	14.03279	0.012753902
MVO	0.05000	0.315956	14.22623	0.012816930
SCA	0.050780	0.334779	12.72269	0.012709667
GSA	0.05000	0.317312	14.22867	0.012873881
GA	0.05010	0.310111	14.0000	0.013036251
HS	0.05025	0.316351	15.23960	0.012776352

Table 10
Comparison of SHO statistical for tension/compression spring design.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
SHO	**0.012674000**	**0.012684106**	0.012715185	**0.000027**	**0.012687293**
GWO	0.012678321	0.012697116	0.012720757	0.000041	0.012699686
PSO	0.013192580	0.014817181	**0.017862507**	0.002272	0.013192580
MFO	0.012753902	0.014023657	0.017236590	0.001390	0.013896512
MVO	0.012816930	0.014464372	0.017839737	0.001622	0.014021237
SCA	0.012709667	0.012839637	0.012998448	0.000078	0.012844664
GSA	0.012873881	0.013438871	0.014211731	0.000287	0.013367888
GA	0.013036251	0.014036254	0.016251423	0.002073	0.013002365
HS	0.012776352	0.013069872	0.015214230	0.000375	0.012952142

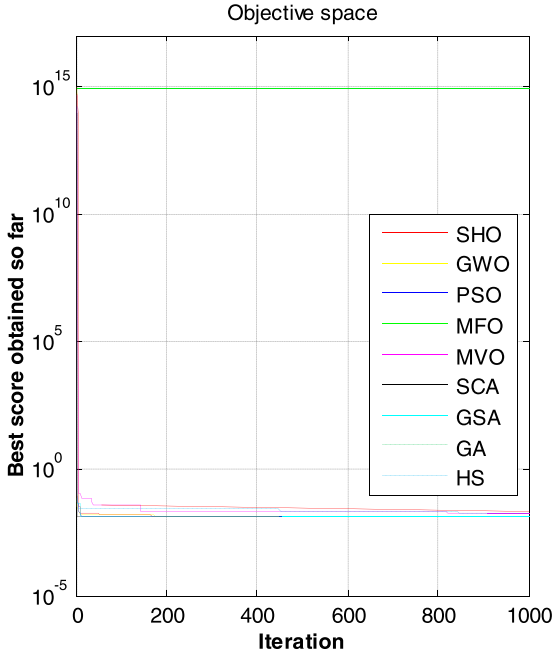

Fig. 17. Analysis of SHO with various techniques for the tension/compression spring problem.

Table 10. The results show that SHO again performs better and require less number of analyses to find the best optimal design.

Fig. 17shows that SHO algorithm achieve the near optimal solution in the initial steps of iterations for tension/compression spring problem and obtain better results than other optimization methods.

4.3. Pressure vessel design

This problem was proposed by Kannan and Kramer [27] to minimize the total cost, including cost of material, forming and welding of cylindrical vessel which are capped at both ends by

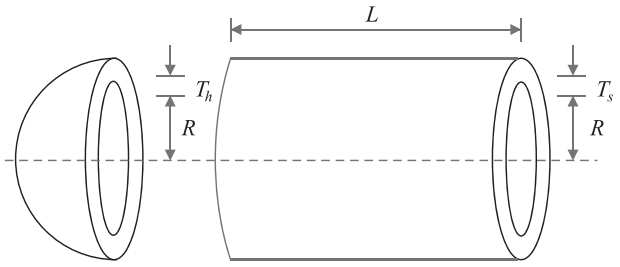

Fig. 18. Schematic view of pressure vessel problem.

Table 11
Comparison results for pressure vessel design problem.

Algorithms	Optimum variables				Optimum cost
	T_s	T_h	R	L	
SHO	0.778210	0.384889	40.315040	200.00000	**5885.5773**
GWO	0.779035	0.384660	40.327793	199.65029	5889.3689
PSO	0.778961	0.384683	40.320913	200.00000	5891.3879
MFO	0.835241	0.409854	43.578621	152.21520	6055.6378
MVO	0.845719	0.418564	43.816270	156.38164	6011.5148
SCA	0.817577	0.417932	41.74939	183.57270	6137.3724
GSA	1.085800	0.949614	49.345231	169.48741	11550.2976
GA	0.752362	0.399540	40.452514	198.00268	5890.3279
HS	1.099523	0.906579	44.456397	179.65887	6550.0230

hemispherical heads as shown in Fig. 18. There are four variables in this problem ($x_1 - x_4$):

- T_s (x_1, thickness of the shell).
- T_h (x_2, thickness of the head).
- R (x_3, inner radius).
- L (x_4, length of the cylindrical section without considering the head).

Among these four design variables R and L are continuous variables while T_s and T_h are integer values which are multiples of 0.0625 *in*. The mathematical formulation of this problem is formulated as follows:

$$\begin{aligned}
&\text{Consider } \vec{z} = [z_1 z_2 z_3 z_4] = [T_s T_h R L],\\
&\text{Minimize } f(\vec{z}) = 0.6224 z_1 z_3 z_4 + 1.7781 z_2 z_3^2 + 3.1661 z_1^2 z_4\\
&\quad +19.84 z_1^2 z_3,\\
&\text{Subject to}\\
&g_1(\vec{z}) = -z_1 + 0.0193 z_3 \le 0,\\
&g_2(\vec{z}) = -z_3 + 0.00954 z_3 \le 0,\\
&g_3(\vec{z}) = -\pi z_3^2 z_4 - \frac{4}{3}\pi z_3^3 + 1,296,000 \le 0,\\
&g_4(\vec{z}) = z_4 - 240 \le 0,
\end{aligned} \tag{13}$$

$$\begin{aligned}
&\text{Variable range}\\
&0 \le z_1 \le 99,\\
&0 \le z_2 \le 99,\\
&0 \le z_3 \le 200,\\
&0 \le z_4 \le 200,
\end{aligned}$$

This problem has been popular among researchers in various studies. Table 11 represents the comparisons of best optimal solution for SHO and other reported methods such as GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS. According to this table, SHO is able to find optimal design with minimum cost. Table 12 shows the comparison statistical results of pressure vessel design problem. The results show that SHO performs better than all other algorithms in terms of best optimal solution.

Table 12
Comparison of SHO statistical for pressure vessel design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
SHO	**5885.5773**	**5887.4441**	5892.3207	**002.893**	**5886.2282**
GWO	5889.3689	5891.5247	5894.6238	013.910	5890.6497
PSO	5891.3879	6531.5032	7394.5879	534.119	6416.1138
MFO	6055.6378	6360.6854	7023.8521	365.597	6302.2301
MVO	6011.5148	6477.3050	7250.9170	327.007	6397.4805
SCA	6137.3724	6326.7606	6512.3541	126.609	6318.3179
GSA	11550.2976	23342.2909	**33226.2526**	5790.625	24010.0415
GA	5890.3279	6264.0053	7005.7500	496.128	6112.6899
HS	6550.0230	6643.9870	8005.4397	657.523	7586.0085

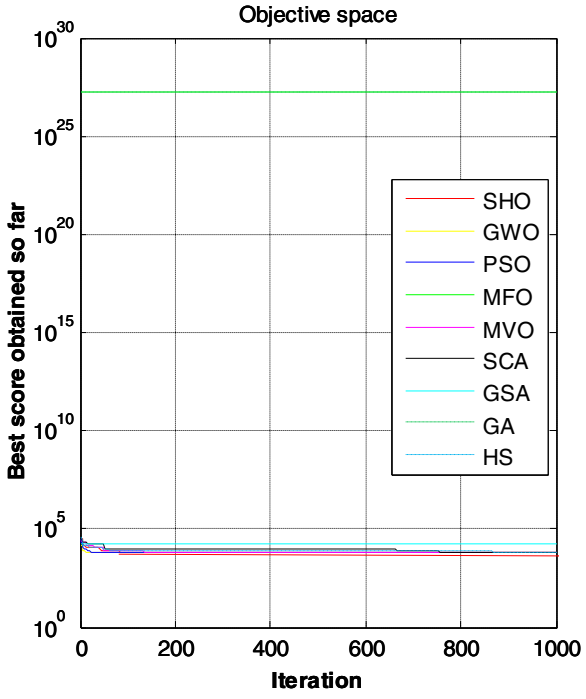

Fig. 19. Analysis of SHO with various techniques for the pressure vessel problem.

In Fig. 19, SHO algorithm obtained the near optimal solution in the initial steps of iterations and achieve better results than other optimization methods for pressure vessel problem.

4.4. Speed reducer design problem

The design of the speed reducer is a more challenging benchmark because it can be associated with seven design variables [18]. In this optimization problem (see Fig. 20) the weight of speed reducer is to be minimized with subject to constraints [40]:

- Bending stress of the gear teeth.
- Surface stress.
- Transverse deflections of the shafts.
- Stresses in the shafts.

There are seven optimization variables ($x_1 - x_7$) of this problem which can represent as the face width (b), module of teeth (m), number of teeth in the pinion (z), length of the first shaft between bearings (l_1), length of the second shaft between bearings (l_2), the diameter of first (d_1) shafts, and the diameter of second shafts (d_2).

This is an illustration of a mixed-integer programming problem. The third variable, number of teeth in the pinion (z), is of integer values. Therefore, all other variables (excluding x_3) are continuous.

Table 13
Comparison results for speed reducer design problem.

Algorithms	Optimum variables							Optimum cost
	b	m	z	l_1	l_2	d_1	d_2	
SHO	3.50159	0.7	17	7.3	7.8	3.35127	5.28874	**2998.5507**
GWO	3.506690	0.7	17	7.380933	7.815726	3.357847	5.286768	3001.288
PSO	3.500019	0.7	17	8.3	7.8	3.352412	5.286715	3005.763
MFO	3.507524	0.7	17	7.302397	7.802364	3.323541	5.287524	3009.571
MVO	3.508502	0.7	17	7.392843	7.816034	3.358073	5.286777	3002.928
SCA	3.508755	0.7	17	7.3	7.8	3.461020	5.289213	3030.563
GSA	3.600000	0.7	17	8.3	7.8	3.369658	5.289224	3051.120
GA	3.510253	0.7	17	8.35	7.8	3.362201	5.287723	3067.561
HS	3.520124	0.7	17	8.37	7.8	3.366970	5.288719	3029.002

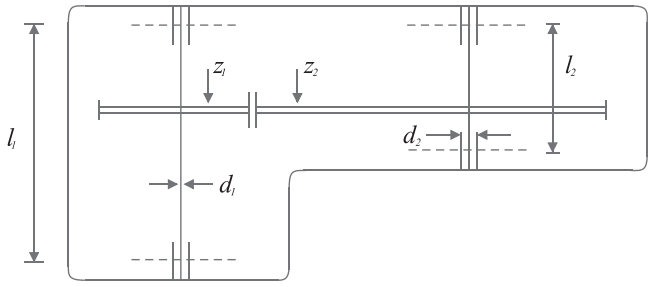

Fig. 20. Schematic view of speed reducer problem.

Table 14
Comparison of SHO statistical for speed reducer design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
SHO	**2998.5507**	**2999.640**	3003.889	**1.93193**	**2999.187**
GWO	3001.288	3005.845	3008.752	5.83794	3004.519
PSO	3005.763	3105.252	3211.174	79.6381	3105.252
MFO	3009.571	3021.256	3054.524	11.0235	3020.524
MVO	3002.928	3028.841	3060.958	13.0186	3027.031
SCA	3030.563	3065.917	3104.779	18.0742	3065.609
GSA	3051.120	3170.334	**3363.873**	92.5726	3156.752
GA	3067.561	3186.523	3313.199	17.1186	3198.187
HS	3029.002	3295.329	3619.465	57.0235	3288.657

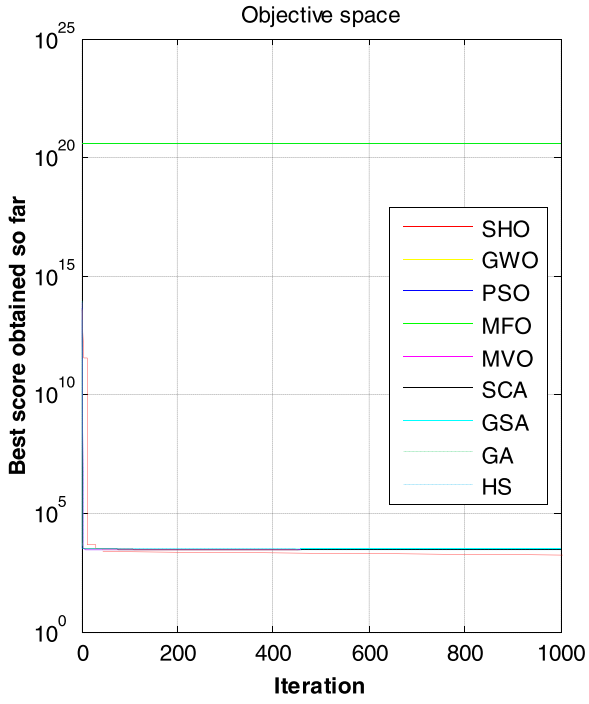

Fig. 21. Analysis of SHO with various techniques for the speed reducer problem.

The comparison of the best optimal solution with various optimization methods is given in Table 13. The statistical results of eight optimization methods including GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS are compared with the proposed method which is given in Table 14.

According to Table 13, among the compared optimization algorithms, SHO have found the best optimal solution so far and able to find a design with minimum cost. The mathematical formulation of this problem is formulated as follows:

$$\text{Minimize} f(\vec{z}) = 0.7854 z_1 z_2^2 (3.3333 z_3^2 + 14.9334 z_3 - 43.0934) - 1.508 z_1 (z_6^2 + z_7^2) + 7.4777 (z_6^3 + z_7^3) + 0.7854 (z_4 z_6^2 + z_5 z_7^2),$$

Subject to

$$g_1(\vec{z}) = \frac{27}{z_1 z_2^2 z_3} - 1 \le 0,$$

$$g_2(\vec{z}) = \frac{397.5}{z_1 z_2^2 z_3^2} - 1 \le 0,$$

$$g_3(\vec{z}) = \frac{1.93 z_4^3}{z_2 z_6^4 z_3} - 1 \le 0,$$

$$g_4(\vec{z}) = \frac{1.93 z_5^3}{z_2 z_7^4 z_3} - 1 \le 0,$$

$$g_5(\vec{z}) = \frac{[(745(z_4/z_2 z_3))^2 + 16.9 \times 10^6]^{1/2}}{110 z_6^3} - 1 \le 0,$$

$$g_6(\vec{z}) = \frac{[(745(z_5/z_2 z_3))^2 + 157.5 \times 10^6]^{1/2}}{85 z_7^3} - 1 \le 0,$$

$$g_7(\vec{z}) = \frac{z_2 z_3}{40} - 1 \le 0,$$

$$g_8(\vec{z}) = \frac{5 z_2}{z_1} - 1 \le 0,$$

$$g_9(\vec{z}) = \frac{z_1}{12 z_2} - 1 \le 0,$$

$$g_{10}(\vec{z}) = \frac{1.5 z_6 + 1.9}{z_4} - 1 \le 0,$$

$$g_{11}(\vec{z}) = \frac{1.1 z_7 + 1.9}{z_5} - 1 \le 0,$$

where

$$2.6 \le z_1 \le 3.6, 0.7 \le z_2 \le 0.8, 17 \le z_3 \le 28, 7.3 \le z_4 \le 8.3, 7.3 \le z_5 \le 8.3, 2.9 \le z_6 \le 3.9, 5.0 \le z_7 \le 5.5 \quad (14)$$

To analyze the Fig. 21, SHO algorithm obtains a best optimal solution for speed reducer problem during the course of iterations. In

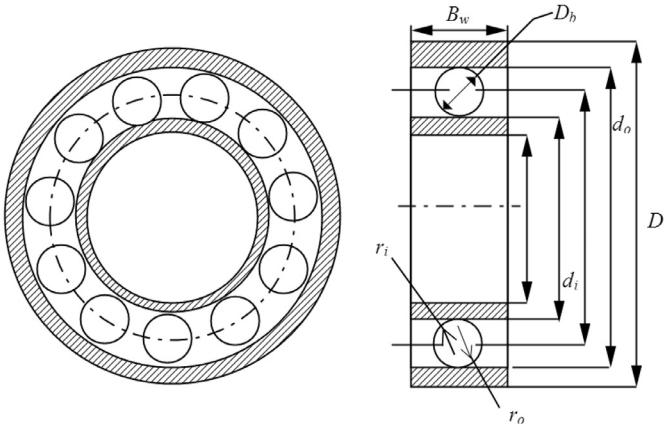

Fig. 22. Schematic view of rolling element bearing problem.

the initial steps of iterations, SHO algorithm achieves better results than other proposed optimization methods.

4.5. Rolling element bearing design problem

The main objective of this problem is to maximize the dynamic load carrying capacity of a rolling element bearing as shown in Fig. 22. There are 10 decision variables of this problem which are pitch diameter (D_m), ball diameter (D_b), number of balls (Z), inner (f_i) and outer (f_o) raceway curvature coefficients, K_{Dmin}, K_{Dmax}, ε, e, and ζ (see Fig. 22). The mathematical formulation is described as follows:

$$\text{Maximize}\, C_d = f_c Z^{2/3} D_b^{1.8} \qquad if D \le 25.4\,\text{mm}$$
$$C_d = 3.647 f_c Z^{2/3} D_b^{1.4} \qquad if D \succ 25.4\,\text{mm}$$
Subject to
$$g_1(\vec{z}) = \frac{\phi_0}{2sin^{-1}(D_b/D_m)} - Z + 1 \le 0,$$
$$g_2(\vec{z}) = 2D_b - K_{Dmin}(D-d) \ge 0,$$
$$g_3(\vec{z}) = K_{Dmax}(D-d) - 2D_b \ge 0,$$
$$g_4(\vec{z}) = \zeta B_w - D_b \le 0,$$
$$g_5(\vec{z}) = D_m - 0.5(D+d) \ge 0,$$
$$g_6(\vec{z}) = (0.5+e)(D+d) - D_m \ge 0,$$
$$g_7(\vec{z}) = 0.5(D - D_m - D_b) - \varepsilon D_b \ge 0,$$
$$g_8(\vec{z}) = f_i \ge 0.515,$$
$$g_9(\vec{z}) = f_o \ge 0.515,$$
where
$$f_c = 37.91\left[1 + \left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41}$$
$$x = [\{(D-d)/2 - 3(T/4)\}^2 + \{D/2 - T/4 - D_b\}^2 - \{d/2 + T/4\}^2]$$
$$y = 2\{(D-d)/2 - 3(T/4)\}\{D/2 - T/4 - D_b\} \qquad (15)$$

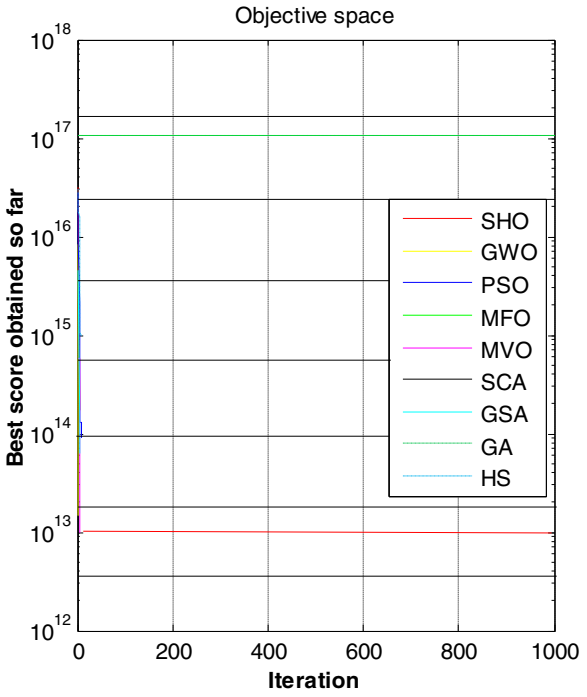

Fig. 23. Analysis of SHO with various techniques for the rolling element bearing problem.

$$\phi_o = 2\pi - 2cos^{-1}\left(\frac{x}{y}\right)$$
$$\gamma = \frac{D_b}{D_m}, \quad f_i = \frac{r_i}{D_b}, \quad f_o = \frac{r_o}{D_b}, \quad T = D - d - 2D_b$$
$$D = 160, \quad d = 90, \quad B_w = 30, \quad r_i = r_o = 11.033$$
$$0.5(D+d) \le D_m \le 0.6(D+d), \quad 0.15(D-d) \le D_b \le 0.45(D-d), \quad 4 \le Z \le 50, \quad 0.515 \le f_i and f_o \le 0.6,$$
$$0.4 \le K_{Dmin} \le 0.5, \quad 0.6 \le K_{Dmax} \le 0.7, \quad 0.3 \le e \le 0.4,$$
$$0.02 \le e \le 0.1, \quad 0.6 \le \zeta \le 0.85$$

Table 15 shows the comparison of best optimal solution with different optimization methods. The statistical optimization results for reported algorithms were compared in Table 16.

From Table 15, the proposed method detected the best optimal solution over other optimizers. Fig. 23 shows that SHO algorithm is capable to achieve a near optimal solution. Regarding statistical results, SHO offered better results against other considered algorithms.

4.6. Displacement of loaded structure

A displacement is a vector which defines the shortest distance between the initial and the final position of a given point. This problem is described the minimization of potential energy to reduce the excess load of structure and overcome the problem of structural failure. It consists of minimizing the potential energy ($f(z)$) of loaded structure. The problem can be stated as follows:

$$f(\vec{z}) = \text{Minimize}_{z_1,z_2}\pi$$
where
$$\pi = \frac{1}{2}K_1 u_1^2 + \frac{1}{2}K_2 u_2^2 - F_z z_1 - F_y z_2$$
$$K_1 = 8N/cm, K_2 = 1N/cm, F_y = 5N, F_z = 5N$$
$$u_1 = \sqrt{z_1^2 + (10 - z_2^2) - 10}, \quad u_2 = \sqrt{z_1^2 + (10 + z_2^2) - 10}$$

Table 17 presents the comparisons of best optimal solution for SHO and other competitor methods such as GWO, PSO, MFO, MVO,

Table 15
Comparison results for Rolling element bearing design problem.

Algorithms	Optimum variables										Optimum cost
	D_m	D_b	Z	f_i	f_o	K_{Dmin}	K_{Dmax}	ε	e	ζ	
SHO	125	21.40732	10.93268	0.515	0.515	0.4	0.7	0.3	0.02	0.6	**85054.532**
GWO	125.6199	21.35129	10.98781	0.515	0.515	0.5	0.68807	0.300151	0.03254	0.62701	84807.111
PSO	125	20.75388	11.17342	0.515	0.515000	0.5	0.61503	0.300000	0.05161	0.60000	81691.202
MFO	125	21.03287	10.96571	0.515	0.515000	0.5	0.67584	0.300214	0.02397	0.61001	84002.524
MVO	125.6002	21.32250	10.97338	0.515	0.515000	0.5	0.68782	0.301348	0.03617	0.61061	84491.266
SCA	125	21.14834	10.96928	0.515	0.515	0.5	0.7	0.3	0.02778	0.62912	83431.117
GSA	125	20.85417	11.14989	0.515	0.517746	0.5	0.61827	0.304068	0.02000	0.624638	82276.941
GA	125	20.77562	11.01247	0.515	0.515000	0.5	0.61397	0.300000	0.05004	0.610001	82773.982
HS	125	20.87123	11.16697	0.515	0.516000	0.5	0.61951	0.301128	0.05024	0.614531	81569.527

Table 16
Comparison of SHO statistical for Rolling element bearing design problem.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
SHO	**85054.532**	**85024.858**	85853.876	0186.68	**85040.241**
GWO	84807.111	84791.613	84517.923	0137.186	84960.147
PSO	81691.202	50435.017	**32761.546**	**13962.150**	42287.581
MFO	84002.524	82357.493	83979.254	1401.524	84497.397
MVO	84491.266	84353.685	84100.834	0392.431	84398.601
SCA	83431.117	81005.232	77992.482	1710.777	81035.109
GSA	82276.941	78002.107	71043.110	3119.904	78398.853
GA	82773.982	81198.753	80687.239	1679.367	8439.728
HS	81569.527	80397.998	79412.779	1756.902	8347.009

Table 17
Comparison results for displacement of loaded structure.

Algorithms	Optimum cost (π)
SHO	**168.8889**
GWO	170.3645
PSO	170.5960
MFO	173.3654
MVO	169.3023
SCA	169.0032
GSA	176.3697
GA	171.3674
HS	172.0324

Table 18
Comparison of SHO statistical for displacement of loaded structure.

Algorithms	Best	Mean	Worst	Std. Dev.	Median
SHO	**168.8889**	**170.3659**	173.6357	**023.697**	**169.6710**
GWO	170.3645	171.3694	174.3970	196.037	173.3694
PSO	170.5960	174.6354	175.3602	236.036	173.9634
MFO	173.3654	178.6972	**180.6357**	323.364	175.3670
MVO	169.3023	171.0034	174.3047	202.753	170.0032
SCA	169.0032	171.7530	174.4527	129.047	170.3647
GSA	176.3697	178.7521	179.5637	113.037	174.367
GA	171.3674	172.0374	174.0098	212.703	172.0097
HS	172.0324	173.4327	174.0399	080.111	171.3697

SCA, GSA, GA, and HS. According to this table, SHO is able to minimize the potential energy for displacement of loaded structure problem.

Table 18 shows the comparison statistical results which shows that SHO performs better than all other algorithms. In Fig. 25, SHO algorithm obtained the near optimal solution and achieve better results than other competitor algorithms for minimization of potential energy (Fig. 24).

In summary, the results on the six real-life engineering problems shows that SHO demonstrates high performance in solving various challenging problems. The optimization results point out that the SHO has the capability to handle various combinatorial optimization problems (COPs). Therefore, it can be concluded that the SHO is the best optimizer which can offer better optimum solutions under lower computational attempts and converge towards the optimum quickly.

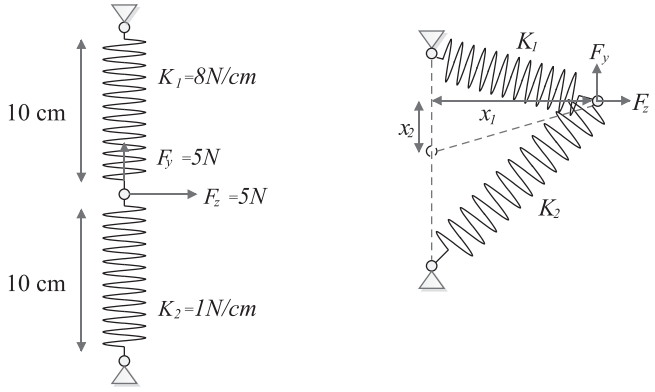

Fig. 24. Schematic view of displacement of loaded structure.

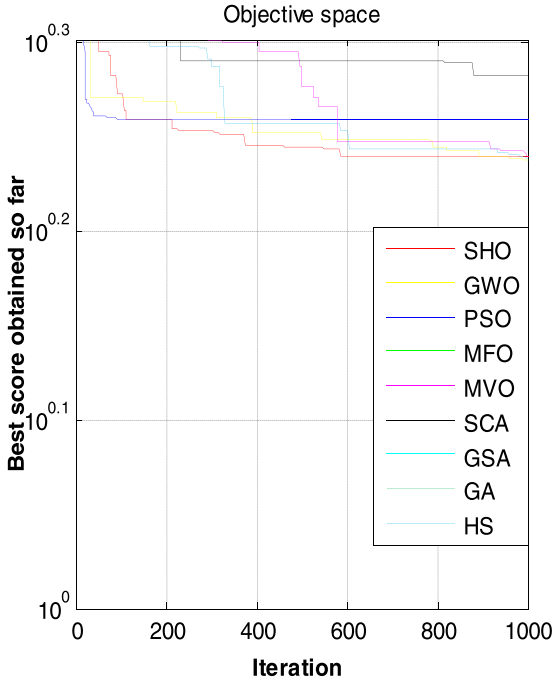

Fig. 25. Analysis of SHO with various techniques for the displacement of loaded structure problem.

5. Conclusion

This paper presented a new swarm-based optimization algorithm called the Spotted Hyena Optimizer (SHO). The fundamental concepts which motivate the method are inspired by social

hierarchy and hunting behavior of spotted hyenas. In this paper, the SHO algorithm is proposed for solving twenty-nine test functions to analyze the exploration, exploitation, local optima avoidance, and convergence behavior. Moreover, six real-life engineering design problems are employed to further examine the efficiency of SHO algorithm. The results demonstrate that the SHO provides extremely competitive results as compared to other well-known heuristics such as GWO, PSO, MFO, MVO, SCA, GSA, GA, and HS. The statistical results, which are based on the comparisons of the proposed SHO against other optimization methods, show that the proposed method can handle various types of constraints and offer better solutions than other optimizers. The proposed method used for solving the real-life optimization problems perhaps require less computational efforts to find the optimum solutions.

Several research directions can be recommended for future works. Solving different fields optimization problems can be done. The binary version of the SHO algorithm can be another motivating future work. Also, to extend this algorithm to solve multi-objective as well as many-objective problems can also be seen as a future contribution.

Appendix A

A.1. Unimodal benchmark functions

The detail description of seven well-known unimodal benchmark test functions ($F_1 - F_7$) are mentioned in Table A.1.

Table A.1
Unimodal benchmark functions.

Function	Dim	Range	f_{min}
$F_1(z) = \sum_{i=1}^{N} z_i^2$	30	[−100, 100]	0
$F_2(z) = \sum_{i=1}^{N} \|z_i\| + \prod_{i=1}^{N} \|z_i\|$	30	[−10, 10]	0
$F_3(z) = \sum_{i=1}^{N} (\sum_{j-1}^{i} z_j)^2$	30	[−100, 100]	0
$F_4(z) = max_i\{\|z_i\|, 1 \le i \le N\}$	30	[−100, 100]	0
$F_5(z) = \sum_{i=1}^{N-1} [100(z_{i+1} - z_i^2)^2 + (z_i - 1)^2]$	30	[−30, 30]	0
$F_6(z) = \sum_{i=1}^{N} (\|z_i + 0.5\|)^2$	30	[−100, 100]	0
$F_7(z) = \sum_{i=1}^{N} iz_i^4 + random[0, 1]$	30	[−1.28, 1.28]	0

A.2. Multimodal benchmark functions

The detail description of nine well-known multimodal benchmark test functions ($F_8 - F_{16}$) are mentioned in Table A.2.

A.3. Fixed-dimension multimodal benchmark functions

The detail description of ten well-known fixed-dimension multimodal benchmark test functions ($F_{14} - F_{23}$) are mentioned in Table A.3.

A.4. Composite benchmark functions

The detail description of six well-known composite benchmark test functions ($F_{24} - F_{29}$) are mentioned in Table A.4.

Table A.2
Multimodal benchmark functions.

Function	Dim	Range	f_{min}
$F_8(z) = \sum_{i=1}^{N} -z_i \sin(\sqrt{\|z_i\|})$	30	[−500, 500]	−418.982 × 5
$F_9(z) = \sum_{i=1}^{N} [z_i^2 - 10\cos(2\pi z_i) + 10]$	30	[−5.12, 5.12]	0
$F_{10}(z) = -20exp\left(-0.2\sqrt{\frac{1}{N}\sum_{i=1}^{N} z_i^2}\right) -$ $\exp\left(\frac{1}{N}\sum_{i=1}^{N} \cos(2\pi z_i)\right) + 20 + e$	30	[−32, 32]	0
$F_{11}(z) = \frac{1}{4000}\sum_{i=1}^{N} z_i^2 - \prod_{i=1}^{N} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$	30	[−600, 600]	0
$F_{12}(z) = \frac{\pi}{N}\{10sin(\pi x_1) + \sum_{i=1}^{N-1}(x_i - 1)^2[1 +$ $10sin^2(\pi x_{i+1})] + (x_n - 1)^2\} + \sum_{i=1}^{N} u(z_i, 10, 100, 4) x_i =$ $1 + \frac{z_i+1}{4} u(z_i, a, k, m) = \begin{cases} k(z_i - a)^m & z_i > a \\ 0 & -a < z_i < a \\ k(-z_i - a)^m & z_i < -a \end{cases}$	30	[−50, 50]	0
$F_{13}(z) = 0.1\{sin^2(3\pi z_1) + \sum_{i=1}^{N}(z_i - 1)^2[1 + \sin^2(3\pi z_i +$ $1)] + (z_n - 1)^2[1 + \sin^2(2\pi z_n)]\} + \sum_{i=1}^{N} u(z_i, 5, 100, 4)$	30	[−50, 50]	0
$F_{13}(z) = 0.1\{sin^2(3\pi z_1) + \sum_{i=1}^{N}(z_i - 1)^2[1 + \sin^2(3\pi z_i +$ $1)] + (z_n - 1)^2[1 + \sin^2(2\pi z_n)]\} + \sum_{i=1}^{N} u(z_i, 5, 100, 4)$	30	[−50, 50]	0
$F_{14}(z) = -\sum_{i=1}^{N} \sin(z_i) \cdot \left(sin\left(\frac{iz_i^2}{\pi}\right)\right)^{2m}, m = 10$	30	[0, π]	−4.687
$F_{15}(z) = \left[e^{-\sum_{i=1}^{N}(z_i/\beta)^{2m}} - 2e^{-\sum_{i=1}^{N} z_i^2}\right] \cdot \prod_{i=1}^{N} cos^2 z_i, m = 5$	30	[−20, 20]	−1
$F_{16}(z) =$ $\{[\sum_{i=1}^{N} sin^2(z_i)] - \exp(-\sum_{i=1}^{N} z_i^2)\} \cdot \exp[-\sum_{i=1}^{N} sin^2\sqrt{\|z_i\|}]$	30	[−10, 10]	−1

Table A.3
Fixed-dimension multimodal benchmark functions.

Function	Dim	Range	f_{min}
$F_{14}(z) = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}(z_i - a_{ij})^6}\right)^{-1}$	2	[−65, 65]	1
$F_{15}(z) = \sum_{i=1}^{11} \left[a_i - \frac{z_1(b_i^2 + b_i z_2)}{b_i^2 + b_i z_3 + z_4}\right]^2$	4	[−5, 5]	0.00030
$F_{16}(z) = 4z_1^2 - 2.1z_1^4 + \frac{1}{3}z_1^6 + z_1 z_2 - 4z_2^2 + 4z_2^4$	2	[−5, 5]	−1.0316
$F_{17}(z) = \left(z_2 - \frac{5.1}{4\pi^2} z_1^2 + \frac{5}{\pi} z_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right) cosz_1 + 10$	2	[−5, 5]	0.398
$F_{18}(z) = [1 + (z_1 + z_2 + 1)^2 (19 - 14z_1 + 3z_1^2 - 14z_2 + 6z_1 z_2 + 3z_2^2)] \times$ $[30 + (2z_1 - 3z_2)^2 \times (18 - 32z_1 + 12z_1^2 + 48z_2 - 36z_1 z_2 + 27z_2^2)]$	2	[−2, 2]	3
$F_{19}(z) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(z_j - p_{ij})^2)$	3	[1, 3]	−3.86
$F_{20}(z) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(z_j - p_{ij})^2)$	6	[0, 1]	−3.32
$F_{21}(z) = -\sum_{i=1}^{5} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	−10.1532
$F_{22}(z) = -\sum_{i=1}^{7} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	−10.4028
$F_{23}(z) = -\sum_{i=1}^{1} 0[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	−10.536

Table A.4
Composite benchmark functions.

Function	Dim	Range	f_{min}
$F_{24}(CF1)$:			
$f_1, f_2, f_3, \ldots, f_{10}$ = Sphere Function			
$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$			
$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$	10	[-5, 5]	0
$F_{25}(CF2)$:			
$f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's Function			
$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$			
$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [5/100, 5/100, 5/100, \ldots, 5/100]$	10	[-5, 5]	0
$F_{26}(CF3)$:			
$f_1, f_2, f_3, \ldots, f_{10}$ = Griewank's Function			
$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$			
$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1, 1, 1, \ldots, 1]$	10	[-5, 5]	0
$F_{27}(CF4)$:			
f_1, f_2 = Ackley's Function			
f_3, f_4 = Rastrigin's Function			
f_5, f_6 = Weierstras's Function			
f_7, f_8 = Griewank's Function			
f_9, f_{10} = Sphere Function			
$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$			
$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [5/32, 5/32, 1, 1, 5/0.5, 5/0.5, 5/100, 5/100, 5/100, 5/100]$	10	[-5, 5]	0
$F_{28}(CF5)$:			
f_1, f_2 = Rastrigin's Function			
f_3, f_4 = Weierstras's Function			
f_5, f_6 = Griewank's Function			
f_7, f_8 = Ackley's Function			
f_9, f_{10} = Sphere Function			
$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [1, 1, 1, \ldots, 1]$			
$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	10	[-5, 5]	0
$F_{29}(CF6)$:			
f_1, f_2 = Rastrigin's Function			
f_3, f_4 = Weierstras's Function			
f_5, f_6 = Griewank's Function			
f_7, f_8 = Ackley's Function			
f_9, f_{10} = Sphere Function			
$[\alpha_1, \alpha_2, \alpha_3, \ldots, \alpha_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$			
$[\beta_1, \beta_2, \beta_3, \ldots, \beta_{10}] =$			
[0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]ye	10	[-5, 5]	0

References

[1] Alatas B. Acroa: artificial chemical reaction optimization algorithm for global optimization. Expert Syst Appl 2011;38(10):13170–80. doi:10.1016/j.eswa.2011.04.126. http://www.sciencedirect.com/science/article/pii/S0957417411006531.

[2] Alba E, Dorronsoro B. The exploration/exploitation tradeoff in dynamic cellular genetic algorithms. IEEE Trans Evol Comput 2005;9(2):126–42. doi:10.1109/TEVC.2005.843751.

[3] Amarjeet, Chhabra JK. Harmony search based remodularization for object-oriented software systems. Comput Lang, Syst Struct 2017b;47, Part 2:153–69. doi:10.1016/j.cl.2016.09.003. http://www.sciencedirect.com/science/article/pii/S1477842416301245.

[4] Amarjeet, Chhabra JK. Improving modular structure of software system using structural and lexical dependency. Inf Softw Technol 2017a;82:96–120. doi:10.1016/j.infsof.2016.09.011. http://www.sciencedirect.com/science/article/pii/S0950584916301951.

[5] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: IEEE Congress on Evolutionary Computation; 2007. p. 4661–7. doi:10.1109/CEC.2007.4425083.

[6] Beyer H-G, Schwefel H-P. Evolution strategies – a comprehensive introduction. Nat Comput 2002;1(1):3–52. doi:10.1023/A:1015059928466.

[7] Bonabeau E, Dorigo M, Theraulaz G. Swarm intelligence: from natural to artificial systems. Oxford University Press, Inc; 1999.

[8] Coello CAC. 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(11-12):1245–87. doi:10.1016/S0045-7825(01)00323-1. http://www.sciencedirect.com/science/article/pii/S0045782501003231.

[9] Dahiya SS, Chhabra JK, Kumar S. Use of genetic algorithm for software maintainability metrics' conditioning. In: 15th International Conference on Advanced Computing and Communications (ADCOM 2007); 2007. p. 87–92. doi:10.1109/ADCOM.2007.69.

[10] Dai C, Zhu Y, Chen W. Seeker optimization algorithm. In: International Conference on Computational Intelligence and Security; 2007. p. 167–76. doi:10.1007/978-3-540-74377-4_18.

[11] Digalakis J, Margaritis K. On benchmarking functions for genetic algorithms. Int J Comput Math 2001;77(4):481–506. doi:10.1080/00207160108805080.

[12] Dorigo M, Birattari M, Stutzle T. Ant colony optimization - artificial ants as a computational intelligence technique. IEEE Comput Intell Mag 2006;1:28–39.

[13] Du H, Wu X, Zhuang J. Small-world optimization algorithm for function optimization. Springer Berlin Heidelberg; 2006. p. 264–73. doi:10.1007/11881223_33.

[14] Erol OK, Eksin I. A new optimization method: big bang-big crunch. Adv Eng Softw 2006;37(2):106–11. doi:10.1016/j.advengsoft.2005.04.005. http://www.sciencedirect.com/science/article/pii/S0965997805000827.

[15] Fogel DB. Artificial intelligence through simulated evolution. Wiley-IEEE Press; 1998. p. 227–96. doi:10.1109/9780470544600.ch7. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5311738.

[16] Formato RA. Central force optimization: a new deterministic gradient-like optimization metaheuristic. Opsearch 2009;46(1):25–51. doi:10.1007/s12597-009-0003-4.

[17] Gandomi AH. Interior search algorithm (isa): a novel approach for global optimization. ISA Trans 2014;53(4):1168–83. doi:10.1016/j.isatra.2014.03.018. http://www.sciencedirect.com/science/article/pii/S0019057814000597.

[18] Gandomi AH, Yang X-S. Benchmark problems in structural optimization. Springer Berlin Heidelberg; 2011. p. 259–81. doi:10.1007/978-3-642-20859-1_12.

[19] Geem ZW, Kim JH, Loganathan GV. A new heuristic optimization algorithm: harmony search. Simulation 2001;76(2):60–8. doi:10.1177/003754970107600201.

[20] Ghorbani N, Babaei E. Exchange market algorithm. Appl Soft Comput 2014;19:177–87. doi:10.1016/j.asoc.2014.02.006. http://www.sciencedirect.com/science/article/pii/S156849461400074X.

[21] Glover F. Tabu search-part i. ORSA J Comput 1989;1(3):190–206.

[22] Glover F. Tabu search-part ii. ORSA J Comput 1990;2(1):4–32.

[23] Hatamlou A. Black hole: a new heuristic optimization approach for data clustering. Inf Sci 2013;222:175–84. doi:10.1016/j.ins.2012.08.023. http://www.sciencedirect.com/science/article/pii/S0020025512005762.

[24] He S, Wu QH, Saunders JR. A novel group search optimizer inspired by animal behavioural ecology. In: IEEE International Conference on Evolutionary Computation; 2006. p. 1272–8. doi:10.1109/CEC.2006.1688455.

[25] He S, Wu QH, Saunders JR. Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 2009;13(5):973–90. doi:10.1109/TEVC.2009.2011992.

[26] Ilany A, Booms AS, Holekamp KE. Topological effects of network structure on long-term social network dynamics in a wild mammal. Ecol Lett 2015;18(7):687–95. doi:10.1111/ele.12447.
[27] Kannan B, Kramer SN. An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J Mech Des 1994;116(2):405–11.
[28] Kashan AH. League championship algorithm: a new algorithm for numerical function optimization. In: International Conference of Soft Computing and Pattern Recognition; 2009. p. 43–8. doi:10.1109/SoCPaR.2009.21.
[29] Kaveh A, Khayatazad M. A new meta-heuristic method: ray optimization. Comput Struct 2012;112-113:283–94. doi:10.1016/j.compstruc.2012.09.003. http://www.sciencedirect.com/science/article/pii/S0045794912002131.
[30] Kaveh A, Mahdavi V. Colliding bodies optimization: a novel meta-heuristic method. Comput Struct 2014;139:18–27. doi:10.1016/j.compstruc.2014.04.005. http://www.sciencedirect.com/science/article/pii/S0045794914000935.
[31] Kaveh A, Talatahari S. A novel heuristic optimization method: charged system search. Acta Mech 2010;213(3):267–89. doi:10.1007/s00707-009-0270-4.
[32] Kennedy J, Eberhart RC. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks; 1995. p. 1942–8.
[33] Kirkpatrick S, Gelatt CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80.
[34] Koza JR. Genetic programming: on the programming of computers by means of natural selection. MIT Press 1992.
[35] Kumar V, Chhabra JK, Kumar D. Parameter adaptive harmony search algorithm for unimodal and multimodal optimization problems. J Comput Sci 2014a;5(2):144–55. doi:10.1016/j.jocs.2013.12.001. http://www.sciencedirect.com/science/article/pii/S1877750313001403.
[36] Kumar V, Chhabra JK, Kumar D. Variance-based harmony search algorithm for unimodal and multimodal optimization problems with application to clustering. Cybern Syst 2014b;45(6):486–511. doi:10.1080/01969722.2014.929349.
[37] Liang JJ, Suganthan PN, Deb K. Novel composition test functions for numerical global optimization. In: Proceedings IEEE Swarm Intelligence Symposium; 2005. p. 68–75. doi:10.1109/SIS.2005.1501604.
[38] Lozano M, Garcia-Martinez C. Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report. Comput Oper Res 2010;37(3):481–97. doi:10.1016/j.cor.2009.02.010.
[39] Lu X, Zhou Y. A novel global convergence algorithm: bee collecting pollen algorithm. In: 4th International Conference on Intelligent Computing. Springer; 2008. p. 518–25. doi:10.1007/978-3-540-85984-0_62.
[40] Mezura-Montes E, Coello CAC. Useful infeasible solutions in engineering optimization with evolutionary algorithms. Springer Berlin Heidelberg; 2005. p. 652–62. doi:10.1007/11579427_66.
[41] Mirjalili S. Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 2015;89:228–49. doi:10.1016/j.knosys.2015.07.006. http://www.sciencedirect.com/science/article/pii/S0950705115002580.
[42] Mirjalili S. Sca: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 2016;96:120–33. doi:10.1016/j.knosys.2015.12.022. http://www.sciencedirect.com/science/article/pii/S0950705115005043.
[43] Mirjalili S, Mirjalili SM, Hatamlou A. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 2016;27(2):495–513. doi:10.1007/s00521-015-1870-7.
[44] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Software 2014;69:46–61. doi:10.1016/j.advengsoft.2013.12.007. http://www.sciencedirect.com/science/article/pii/S0965997813001853.
[45] Moghaddam FF, Moghaddam RF, Cheriet M. Curved space optimization: a random search based on general relativity theory. Neural Evol Comput 2012. abs/1208.2214.
[46] Moosavian N, Roodsari BK. Soccer league competition algorithm: a novel meta-heuristic algorithm for optimal design of water distribution networks. Swarm Evol Comput 2014;17:14–24. doi:10.1016/j.swevo.2014.02.002. http://www.sciencedirect.com/science/article/pii/S2210650214000224.
[47] Mucherino A, Seref O. Monkey search: a novel metaheuristic search for global optimization. AIP Conf Proc 2007;953(1).
[48] Oftadeh R, Mahjoob M, Shariatpanahi M. A novel meta-heuristic optimization algorithm inspired by group hunting of animals: hunting search. Comput Math Appl 2010;60(7):2087–98. doi:10.1016/j.camwa.2010.07.049. http://www.sciencedirect.com/science/article/pii/S0898122110005419.
[49] Olorunda O, Engelbrecht AP. Measuring exploration/exploitation in particle swarms using swarm diversity. In: IEEE Congress on Evolutionary Computation; 2008. p. 1128–34. doi:10.1109/CEC.2008.4630938.
[50] Praditwong K, Harman M, Yao X. Software module clustering as a multi-objective search problem. IEEE Trans Softw Eng 2011;37(2):264–82. doi:10.1109/TSE.2010.26.
[51] Ramezani F, Lotfi S. Social-based algorithm. Appl Soft Comput 2013;13(5):2837–56. doi:10.1016/j.asoc.2012.05.018. http://www.sciencedirect.com/science/article/pii/S1568494612002542
[52] Rashedi E, Nezamabadi-pour H, Saryazdi S. GSA: A gravitational search algorithm. Inf Sci 2009;179(13):2232–48. doi:10.1016/j.ins.2009.03.004. http://www.sciencedirect.com/science/article/pii/S0020025509001200.
[53] 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(5):2592–612. doi:10.1016/j.asoc.2012.11.026. http://www.sciencedirect.com/science/article/pii/S1568494612005108.
[54] Shah Hosseini H. Principal components analysis by the galaxy-based search algorithm: a novel metaheuristic for continuous optimisation. Int J Comput Sci Eng 2011;6:132–40. doi:10.1504/IJCSE.2011.041221.
[55] Shiqin Y, Jianjun J, Guangxing Y. A dolphin partner optimization. In: Proceedings of the WRI Global Congress on Intelligent Systems; 2009. p. 124–8. doi:10.1109/GCIS.2009.464.
[56] Simon D. Biogeography-based optimization. IEEE Trans Evol Comput 2008;12(6):702–13. doi:10.1109/TEVC.2008.919004.
[57] Storn R, Price K. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11(4):341–59. doi:10.1023/A:1008202821328.
[58] Suganthan PN, Hansen N, Liang JJ, Deb K, Chen Y-P, Auger A, et al. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization Technical Report. Singapore: Nanyang Technological University; 2005.
[59] Tan Y, Zhu Y. Fireworks algorithm for optimization. Springer Berlin Heidelberg; 2010. p. 355–64. doi:10.1007/978-3-642-13495-1_44.
[60] Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput 1997;1(1):67–82. doi:10.1109/4235.585893.
[61] Yang C, Tu X, Chen J. Algorithm of marriage in honey bees optimization based on the wolf pack search. In: International Conference on Intelligent Pervasive Computing; 2007. p. 462–7. doi:10.1109/IPC.2007.104.
[62] Yang X. Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2010b;2(2):78–84. doi:10.1504/IJBIC.2010.032124.
[63] Yang X-S. A new metaheuristic bat-inspired algorithm. Springer Berlin Heidelberg. p. 65–74. doi:10.1007/978-3-642-12538-6_6.
[64] Yang XS, Deb S. Cuckoo search via levy flights. In: World congress on nature biologically inspired computing; 2009. p. 210–14. doi:10.1109/NABIC.2009.5393690.