Evolutionary Intelligence (2018) 11:31–52
https://doi.org/10.1007/s12065-018-0168-y

SPECIAL ISSUE

Optimization using lion algorithm: a biological inspiration from lion's social behavior

Rajakumar Boothalingam[1]

Received: 30 April 2018 / Revised: 17 August 2018 / Accepted: 28 August 2018 / Published online: 7 September 2018

Abstract
Nature-inspired optimization algorithms, especially evolutionary computation-based and swarm intelligence-based algorithms are being used to solve a variety of optimization problems. Motivated by the obligation of having optimization algorithms, a novel optimization algorithm based on a lion's unique social behavior had been presented in our previous work. Territorial defense and territorial takeover were the two most popular lion's social behaviors. This paper takes the algorithm forward on rigorous and diverse performance tests to demonstrate the versatility of the algorithm. Four different test suites are presented in this paper. The first two test suites are benchmark optimization problems. The first suite had comparison with published results of evolutionary and few renowned optimization algorithms, while the second suite leads to a comparative study with state-of-the-art optimization algorithms. The test suite 3 takes the large-scale optimization problems, whereas test suite 4 considers benchmark engineering problems. The performance statistics demonstrate that the lion algorithm is equivalent to certain optimization algorithms, while outperforming majority of the optimization algorithms. The results also demonstrate the trade-off maintainability of the lion algorithm over the traditional algorithms.

Keywords Lion algorithm · Optimization · Bio-inspired · Large-scale · Crossover · Mutation

1 Introduction

Nature-inspired computing plays a great role in solving uncertain, partially true, imprecise, highly conflicting and complex problems with the aid of nature's behavior and its inspiration [1, 2]. Bio-inspired computing can be said as a subset of natural computing that has emerged from the day of solving real-life problems with biological motivation [3, 4]. The nature computing has become popular among all the researchers, when it has broken barriers of classical computing [5], solving classification and prediction problems and more specifically optimization problems. Numerous optimization algorithms or optimal search algorithms have been developed based on natural inspiration since 1970 [6]. Since then, bio-inspired optimization algorithms such as genetic algorithm, particle swarm optimization algorithm, and many more find applications in almost all the emerging fields [5, 7–9].

In the family of bio-inspired optimization algorithms, a new optimization algorithm called as lion algorithm is introduced in this paper. Previously, a basic model of the algorithm (was termed as lion's algorithm) was proposed in [10]. Further, we extended the algorithm (named as lion algorithm) and solved large scale bilinear system identification [11]. Since the development, numerous researchers have adopted our algorithm for various applications [12–14]. However, it has not been well-studied for its versatility. Hence, this paper investigates the performance of the lion algorithm on different test suites (downloadable at https://sites.google.com/view/lionalgorithm/test-suite) and engineering optimization problems.

2 Proposed lion algorithm

2.1 Biological inspiration

Distinct from other cat species, lions survive with an interesting social system/behavior, termed as pride, to strengthen their own species at every generation. Generally, a pride is comprised of 1–3 lion pairs in which the resident females

✉ Rajakumar Boothalingam
rajakumar.br01@gmail.com; rajakumar.br@ieee.org

[1] Resbee Info Technologies, Nagercoil, Tamil Nadu, India

attend males to give birth to offspring. They share an area called a territory with peaceful interactions. The dominating lion of a territory (often called as territorial lion) rules the territory by fighting against other attacking animals including nomadic lions. The territorial defense continues till the cubs attain sexual maturity, probably for 2–4 years. During these 2–4 years, nomadic lions try to invade the pride. Frequent survival fights take place between the nomadic lions and the pride (for which territorial lion is responsible) for the territorial defense. A coalition among the lions, which are in pride, helps to defeat the nomadic lion. However, if the territorial lion is defeated, it will be killed or driven out by the pride of the nomadic lion. The nomadic lion becomes the territorial lion by taking charge of the pride. It kills the cubs of a lost lion and drives the lioness to estrus. Copulation begins between the lioness and the new territorial lion to give birth to offspring of a new lion. The behavior continues until cubs of a territory get matured.

Once the cubs become adult and if proved as they are stronger than the territorial lion, territorial takeover happens. Like territorial defense, the territorial laggard lion will be either killed or driven out from the pride of completely grown cubs. The new stronger pride lion drives/kills the cubs or weak lions of the pride and copulate with pride lioness to give birth to their own cubs [15].

2.2 Algorithm structure

The basic model of the lion algorithm was introduced as a searching algorithm in the year of 2012 [10]. Subsequently, it has been restructured with upgrades and presented in [11]. It comprises of six processing stages namely, (1) Pride generation, (2) Fertility evaluation, (3) Mating, (4) Territorial defense, (5) Territorial takeover and (6) Termination.

Pride generation is the initiating process of lion algorithm, which is similar to the initialization steps of most of the evolution and swarm-based optimization algorithms. Mating is the most responsible process for deriving new lions (called as cubs) from the parent lions after subjecting to fertility evaluation, which is the second process. Territorial defense and territorial takeover are identified as the unique processes over other optimization algorithms, as they are the explicit inspiration of lion's social behavior. These two steps play the primary roles to guide the algorithm in determining the optimal solutions from a huge search space. The termination process of lion algorithm is problem dependent and hence that could be either based on a number of iterations/generations or based on the optimality of the obtained solutions. More details about individual processing stages are described in [11], while their concise description is given further. The mathematical background on the paper is presented in Appendix.

3 Processes and operators of lion algorithm

In this section, the search processes and operators are explained in detail with the reference of Fig. 1. Despite two variants of lion's algorithm are given in [10]. This paper presents only real-coded lion algorithm. However, binary-coded lion algorithm is presented here as a special case of real-coded lion algorithm when the dimension of the solution variable is one.

3.1 Problem formulation

Let us consider the objective function given in Eq. (1)

$$X^{optimal} = \underset{x_i \in (x_i^{\min}, x_i^{\max})}{\arg\min} \; f(x_1, x_2, \dots x_n); \quad n \geq 1. \tag{1}$$

In Eq. (1), $f(\bullet)$ is a continuous unimodal or multimodal function for which the solution space is of size $\mathfrak{R}^n$ where, $\mathfrak{R}$ represents real numbers, $x_i \, : \, i = 1, 2, \dots, n$ is the i^{th} solution variable and n is the dimension of the solution vector, $x_i^{\min}$ and $x_i^{\max}$ are the minimum and maximum limits of ith solution variable, respectively. $X^{optimal}$ is the optimal/target solution to be obtained from the given optimization algorithm and it can be represented as in Eq. (3). The size of the solution space of $f(\bullet)$ can be determined as follows

$$\mathfrak{R}^n = \prod_{i=1}^{n} \left(x_i^{\max} - x_i^{\min} \right) \tag{2}$$

$$X^{optimal} = X \, : \, f(X) < f\left(X' \middle| X' \neq X; x_i' \in \left(x_i^{\min}, x_i^{\max} \right) \right), \tag{3}$$

where, X is the solution vector with the representation $X = \left[x_1, x_2 \dots x_n \right]$. Equation (1) presents the objective function to be solved as a minimization function. In some cases, this could be a maximization function and hence appropriate selection process has to be used in the optimization algorithm.

3.2 Pride generation

According to the pride definition [10] and Eq. (1), pride is initialized with a territorial lion X^{male}, its lioness X^{female} and a nomadic lion X^{normal}. The nomadic lion is not a member of pride, despite its generation is discussed in pride generation process. The lions representation is as similar to the solution vector representation. The vector elements of X^{male}, X^{female} and X^{nomad}, i.e., x_l^{male}, x_l^{female} and x_l^{nomad} are arbitrary integers within the minimum and maximum limits when $n > 1$ (search with real encoding), where, $l = 1, 2, \dots, L$. Here, L represents the length of the lion that can be determined as follows

$$L = \begin{cases} n; & n > 1 \, (general\, case) \\ m; & otherwise \, (special\, case), \end{cases} \tag{4}$$

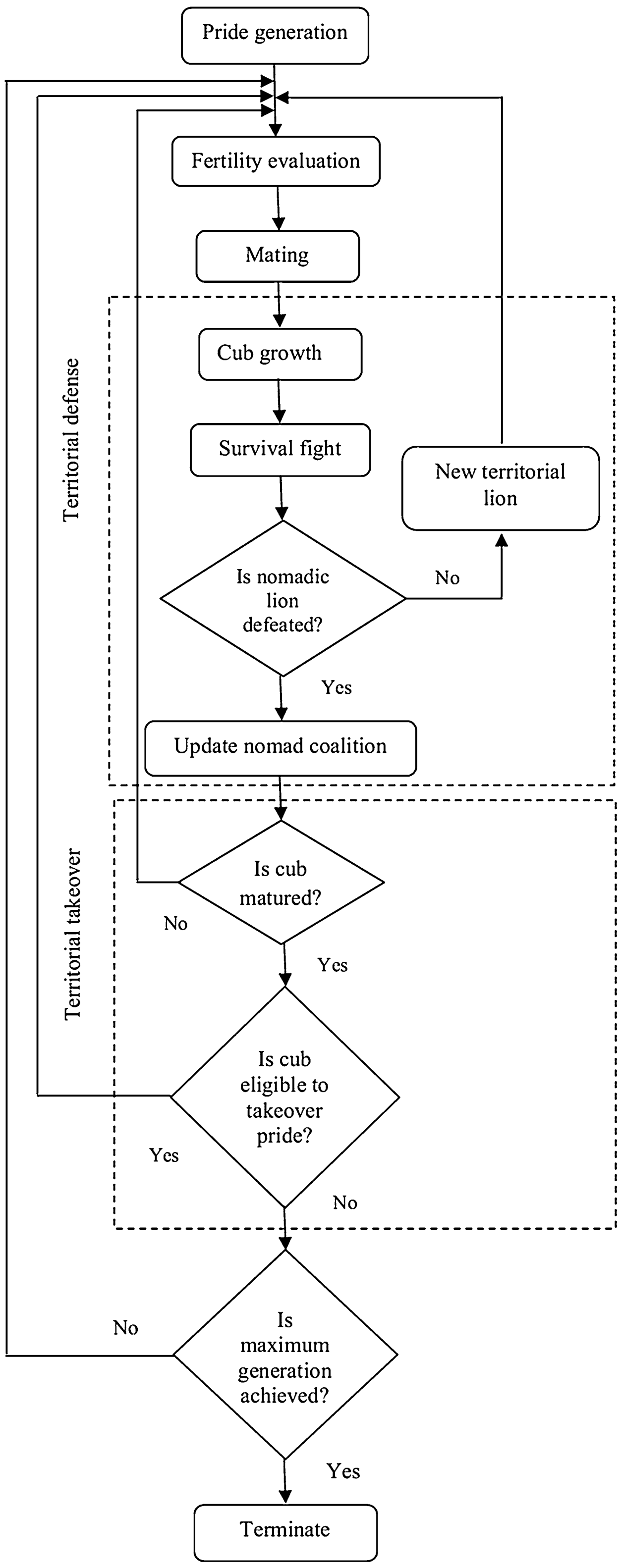

Fig. 1 Flowchart to depict the structure of lion algorithm

where, m and n are integers to decide the length of lions. When $n = 1$, the algorithm has to search with binary encoded lion and so the vector elements are either be generated as 1 or 0, provided the constraints are given in Eqs. (5) and (6) have to be satisfied.

$$g(x_l) \in \left(x_l^{\min}, x_l^{\max}\right) \tag{5}$$

$$m\%2 = 0, \tag{6}$$

where,

$$g(x_l) = \sum_{l=1}^{L} x_l 2^{\left(\frac{L}{2}-l\right)}. \tag{7}$$

Equations (5) and (7) ensure that the generated binary lion is within the solution space and Eq. (6) ensures that the numbers of binary bits before and after the decimal point are equal. As we experiment only with real-coded lion algorithm, we do not further discussed about the binary-encoded lion and henceforth all the discussions explicitly represent only real-coded lion algorithm.

The generated X^{nomad} fills one of the two nomadic lion positions as we assume that there are two nomadic lions try to invade territory. The other nomadic lion will be initialized only at the time of territorial defense. Hence, for the time being the position remains null and X^{nomad} will be represented as X_1^{nomad}.

3.3 Fertility evaluation

In the sequential process of lion algorithm, every territorial lion and lioness begin to age or sometimes infertile. This makes the lion as laggard either at survival fights or territorial takeover. If X^{male} and X^{female} become saturated by their fitness, then either they would have reached global optima or local optima from which they could not take us to better solutions. Fertility evaluation can help to skip from local optimal solutions. In this process, the X^{male} is found as becoming laggard and its laggardness rate L_r is increased by one, if $f\left(X^{male}\right)$ is greater than f^{ref}, which is reference fitness. When L_r exceeds its maximum limit $L_r^{\max}$ then territorial defense takes place. The fertility of X^{female} is ensured by sterility rate S_r, which is increased by one after crossover. If S_r exceeds the tolerance $S_r^{\max}$, then X^{female} undergoes update as given in Eq. (8). When the updated female $X^{female+}$ is taken as X^{female} due to its improvement, the mating process can be performed. On contrary, the updating continues till the female generation count g_c reaches $g_c^{\max}$. If there is no $X^{female+}$ to replace X^{female} throughout the updating process, it can be decided that the X^{female} is still fertile enough to produce better cubs [11].

$$x_l^{female+} = \begin{cases} x_k^{female+}; & if\ l = k \\ x_l^{female}; & otherwise \end{cases} \tag{8}$$

$$x_k^{female+} = \min\left[x_k^{\max}, \max\left(x_k^{\min}, \nabla_k\right)\right] \tag{9}$$

$$\nabla_k = \left[x_k^{female} + (0.1r_2 - 0.05)\left(x_k^{male} - r_1 x_k^{female}\right)\right], \tag{10}$$

where, $x_l^{female+}$ and $x_k^{female+}$ are the l^{th} and k^{th} vector elements of $X^{female+}$, respectively, k is a random integer generated within the interval $[1, L]$, ∇ is the female update function, r_1 and r_2 are random integers generated within the interval $[0, 1]$.

3.4 Mating

In lion algorithm, mating involves two primary steps and one supplementary step. Crossover and mutation are considered as the primary steps and gender clustering is called here as the supplementary step. In the literature, numerous works have been discussed about the crossover and mutation operations and their need for evolutionary algorithms [16–19]. These operators highly motivate us and hence we embed them in our algorithm. By performing crossover and mutation, X^{male} and X^{female} give birth to cubs, which are solutions derived from both the elements of X^{male} and X^{female}. We follow the maximum natural littering rate, i.e., four cubs (mostly) in a lioness pregnancy [20] and so our crossover process gives four cubs.

The proposed crossover operation for generating one cub is portrayed in Fig. 2, while the definition is given in Definition 1 of Appendix. The crossover mask B is varied to generate each cub, i.e. p^{th} mask B_p is used to obtain $X^{cubs}(p)$. These four cubs are further subjected to mutation to produce new four cubs. Henceforth, we use the terms 'X^{cubs}' to represent cubs that are obtained from crossover and 'X^{new}' to represent cubs that are obtained from mutation. These eight cubs occupy cub pool and are subjected to gender clustering to finalize X^{m_cub} and X^{f_cub}. The resultant X^{m_cub} and X^{f_cub} follows cub growth function to get self-update [11].

3.5 Lion operators

Territorial defense [10, 11] not only facilitates wide searching of solution space, but also guides to algorithm to evade from local optimal point as well as identifying diverse solutions with similar fitness. The territorial defense can be sequenced here as generating nomad coalition [21, 22], survival fight [23] and then pride and nomad coalition updates. The nomad coalition process is simplified by applying winner take-all approach [24] to identify X^{e_nomad}. Subsequently, X^{e_nomad} is selected if the criteria given in Eq. (11)–(13) are met.

$$f(X^{e\_nomad}) < f(X^{male}) \tag{11}$$

$$f\left(X^{e\_nomad}\right) < f\left(X^{m\_cub}\right) \tag{12}$$

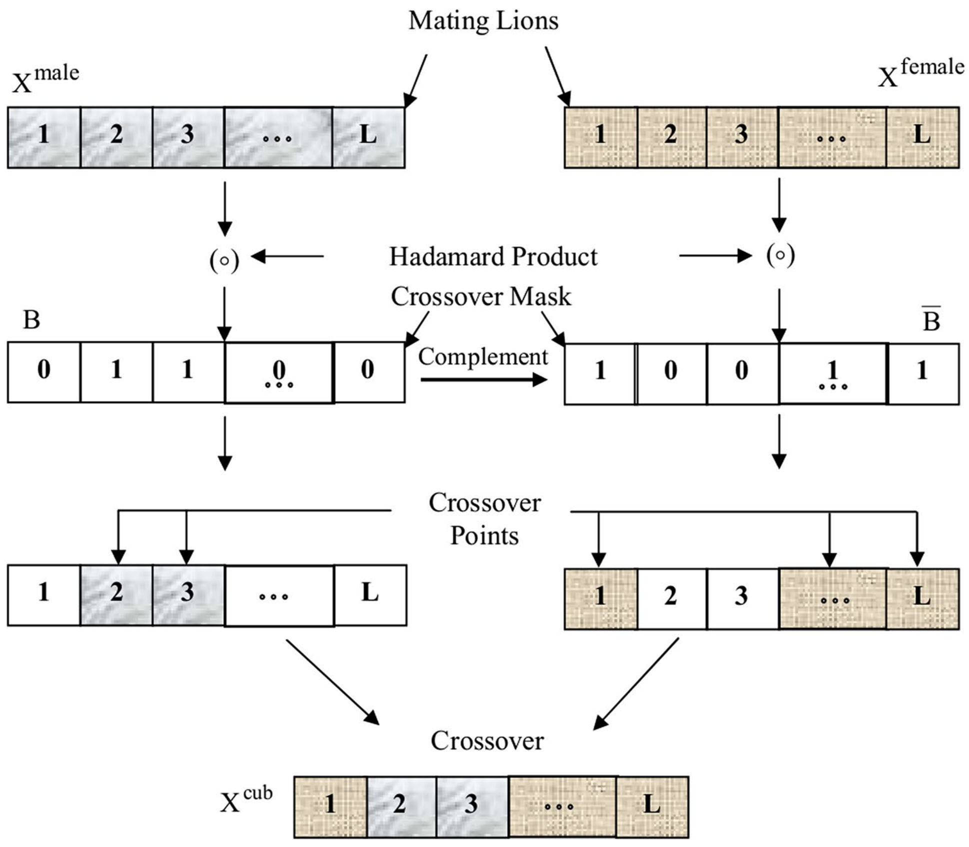

Fig. 2 Crossover operation for lion algorithm

$$f\left(X^{e\_nomad}\right) < f\left(X^{f\_cub}\right). \tag{13}$$

Pride is updated only when X^{male} is defeated, whereas nomad coalition is updated only when X^{e_nomad} is defeated. The pride updating is a process of replacing X^{male} by X^{e_nomad}, whereas updating a nomad coalition means selection of only one X^{nomad}, which has E^{nomad} greater than or equal to the exponential of unity (please refer theorem 2 in Appendix) and the other position will be filled only at the time of next territorial defense [11].

Territorial takeover [10] drives the algorithm to update X^{male} and X^{female} if X^{m_cub} and X^{f_cub} are matured, i.e. when the age of cubs exceeds the maximum age for cub maturity $A_{\max}$ [11].

3.6 Termination

The algorithm execution is terminated when atleast one of the following two termination criteria is met

$$N_g > N_g^{\max} \tag{14}$$

$$\left|f\left(X^{male}\right) - f\left(X^{optimal}\right)\right| \le e_T, \tag{15}$$

where, N_g is the number of generations, which is initialized as zero and incremented by one when a territorial takeover takes place, $N_g^{\max}$ and e_T are the maximum number of generations and error threshold, respectively and $|\bullet|$ is the absolute difference. Please note that second criterion, given in Eq. (15) can be considered only when the target minimum $f\left(X^{optimal}\right)$ (or maximum) is known and $f\left(X^{optimal}\right)$ does not mean that $X^{optimal}$ is known.

4 Experimental setup

4.1 Implementation

The steps to be followed to simulate lion algorithm are as follows

Step 1: Initialize X^{male}, X^{female} and X_1^{nomad}

Step 2: Calculate $f\left(X^{male}\right)$, $f\left(X^{female}\right)$ and $f\left(X_1^{nomad}\right)$

Step 3: Set $f^{ref} = f\left(X^{male}\right)$ and $N_g = 0$

Step 4: Store X^{male} and $f\left(X^{male}\right)$

Step 5: Perform fertility evaluation

Step 6: Perform mating and obtain cubpool

Step 7: Perform gender clustering and obtain X^{m_cub} and X^{f_cub}

Step 8: Initialize A_{cub} as zero

Step 9: Execute cub growth function

Step 10: Perform territorial defense; if defense result 0, go to step 4

Step 11: If $A_{cub} < A_{\max}$, go to step 9

Step 12: Perform territorial takeover and obtain updated X^{male} and X^{female}

Step 13: Increase N_g by one

Step 14: If the termination criteria are not met, go to step 4, otherwise terminate the process.

5 Results and discussions

5.1 Test suite 1

The test suite 1 has 23 benchmark functions of unimodal and multimodal in nature. These benchmark functions have been widely used by various researchers to assess the performance of optimization algorithms [25–27]. The performance of the lion algorithm on test suite 1 is compared in two phases. In phase 1, traditional evolutionary computation methods are used, whereas phase 2 considers renowned optimization algorithms.

5.1.1 Phase 1 of test suite 1

The traditional evolutionary computation methods include classical evolutionary programming (CEP) [28, 29], fast evolutionary programming (FEP) [27], canonical evolutionary strategies (CES) [30], fast evolutionary strategies (FES) [31], Covariance Matrix Adaptation-Evolution Strategy (CMA-ES) [32, 33], evolutionary strategies learned with automatic termination criteria (ESLAT) [34]. To ensure fair comparison, the published results of CEP and FEP are collected from [27], CES and FES from [31], CMA-ES and ESLAT from [34]. As unimodal functions and multimodal functions with many local minima are having higher dimension than multimodal functions with few local minima, at least 1000 executions are required to ensure the reliability of the obtained results [25]. Hence, 1000 executions of lion algorithm are made and the statistical measures are determined and tabulated in Tables 1 and 2. Table 3 depicts the statistical results for 50 runs on multimodal benchmark functions with few local minima.

From Table 1, it can be seen that the lion algorithm outperforms four algorithms when solving schwefel 2.21 and step functions. When solving schwefel 2.22, it dominates over three algorithms (FEP, CES and FES) while it dominates over two algorithms (CEP and CES) when solving schwefel 1.2 function. Despite the lion algorithm is incompetent to solve sphere and rosenbrock functions, it is the most competent algorithm to solve quartic function.

Table 2 shows that among all the other multimodal functions with many local minima, the penalized function is the

Table 1 Performance comparison between lion algorithm and evolutionary computing methods on unimodal benchmark functions

Function	Lion algorithm		CEP		FEP		CES		FES		CMA-ES		ESLAT	
	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R
Sphere	$5.45\times10^{-3}(1.08\times10^{-3})$	7	$2.2\times10^{-4}(5.9\times10^{-4})$	6	5.7×10^{-4} (1.3×10^{-4})	4	$3.4\times10^{-5}(8.6\times10^{-6})$	3	2.5×10^{-4} (6.8×10^{-4})	5	9.7×10^{-23} (3.8×10^{-23})	1	2.0×10^{-17} (2.9×10^{-17})	2
Schwefel 2.22	3.7×10^{-3} (3.1×10^{-3})	4	2.6×10^{-3} (1.7×10^{-4})	3	8.1×10^{-3} (7.7×10^{-4})	5	2.1×10^{-2} (2.2×10^{-3})	6	6.0×10^{-2} (9.6×10^{-3})	7	4.2×10^{-11} (7.1×10^{-23})	1	$3.8\times10^{-5}(1.6\times10^{-5})$	2
Schwefel 1.2	2.1×10^{-2} (8.3×10^{-3})	5	$5.0\times10^{-2}(6.6\times10^{-2})$	6	1.6×10^{-2} (1.4×10^{-2})	4	1.3×10^{-4} (8.5×10^{-5})	7	1.4×10^{-3} (5.3×10^{-4})	3	$7.1\times10^{-23}(2.9\times10^{-23})$	1	$6.1\times10^{-6}(7.5\times10^{-6})$	2
Schwefel 2.21	$4.21\times10^{-2}(4.48\times10^{-2})$	3	2.0 (1.2)	7	0.3 (0.5)	4	0.35 (0.42)	5	$5.5\times10^{-3}(6.5\times10^{-4})$	2	$5.4\times10^{-12}(1.5\times10^{-12})$	1	0.78 (1.64)	6
Rosenbrock	56.77(45.28)	7	6.17 (13.61)	4	5.06 (5.87)	3	6.69 (14.45)	5	33.28 (43.13)	6	0.4 (1.2)	1	1.93 (3.35)	2
Step	4.64×10^{-3}y(1.20×10^{-3})	3	577.76 (1125.76)	7	0(0)	1	411.16 (695.35)	6	0(0)	1	1.44 (1.77)	5	2.0×10^{-2} (0.14)	4
Quartic	6.73×10^{-11} (8.50×10^{-12})	1	1.8×10^{-2} (6.4×10^{-3})	4	7.6×10^{-3} (2.6×10^{-3})	2	3.0×10^{-2} (1.5×10^{-2})	5	1.2×10^{-2} (5.8×10^{-3})	3	0.23 (8.7×10^{-2})	6	0.39 (0.22)	7
Average rank	4.29		5.29		3.29		5.29		3.86		2.29		3.57	
Final rank	5		6		2		7		4		1		3	

SD standard deviation, *R* rank

Table 2 Performance comparison between lion algorithm and evolutionary computing methods on multimodal benchmark functions with many local minima

Function	Lion algorithm		CEP		FEP		CES		FES		CMA-ES		ESLAT	
	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R
Schwefel	−12559.52(34.02)	2	−7917.1(634.5)	5	−12554.5(52.6)	4	−7549.9(631.39)	7	−12556.4(32.53)	3	−7637.14(895.6)	6	$-2.3\times10^{15}(5.70\times10^{15})$	1
Rastrigin	$1.85\times10^{-6}(4.07\times10^{-6})$	1	89.0(23.1)	7	$4.6\times10^{-2}(1.2\times10^{-2})$	3	70.82(21.49)	6	0.16(0.33)	2	51.78(13.56)	5	4.65(5.67)	4
Ackley	$3.72\times10^{-3}(4.43\times10^{-3})$	1	9.2(2.8)	7	$1.8\times10^{-2}(2.1\times10^{-2})$	5	9.07(2.84)	6	$1.2\times10^{-2}(1.8\times10^{-3})$	4	$6.9\times10^{-12}(1.3\times10^{-12})$	2	$1.8\times10^{-8}(5.4\times10^{-9})$	3
Griewank	$7.51\times10^{-2}(5.98\times10^{-2})$	5	$8.6\times10^{-2}(0.12)$	6	$1.6\times10^{-2}(2.2\times10^{-2})$	3	0.38(0.77)	7	$3.7\times10^{-2}(5.0\times10^{-2})$	4	$7.4\times10^{-4}(2.7\times10^{-3})$	1	$1.4\times10^{-3}(4.7\times10^{-3})$	2
Penalized	2.3(2.89)	7	1.76(2.4)	6	$9.2\times10^{-6}(6.1395\times10^{-5})$	2	1.18(1.87)	5	$2.8\times10^{-2}(8.1\times10^{-11})$	4	$1.2\times10^{-4}(3.4\times10^{-2})$	3	$1.5\times10^{-12}(2.0\times10^{-12})$	1
Penalized 2	$2.65\times10^{-5}(6.41\times10^{-5})$	1	1.4(3.7)	7	$1.6\times10^{-4}(7.3\times10^{-5})$	3	1.39(3.33)	6	$4.7\times10^{-5}(1.5\times10^{-5})$	2	$1.7\times10^{-3}(4.5\times10^{-3})$	4	$6.4\times10^{-3}(8.9\times10^{-3})$	5
Average rank	2.83		6.33		3.33		6.17		3.17		3.5		2.67	
Final rank	2		7		4		6		3		5		1	

SD standard deviation, *R* rank

most challenging function for lion algorithm as it gives the least performance over other algorithms. However, from the same Table, lion algorithm is found to be the best of the six other evolutionary computation algorithms to solve rastrigin, Ackley and penalized 2 functions. It dominates over five algorithms (CEP, FEP, CES, FES and CMA-ES) while solving schwefel function and two algorithms (CEP and CES), while solving griewangk function.

From Table 3, it can be seen that all the seven algorithms found global minima of sixhump, branin and Hartman 3 functions and hence they grab the rank 1. However, lion algorithm provides zero standard deviation for the first two functions and second lesser standard deviation for Hartman 3 function. Similarly, lion algorithm dominates over other algorithms by zero standard deviation when solving goldstein-price function, despite CEP, CES, FES and ESLAT share rank 1 position. Lion algorithm is found as the best algorithm to solve foxholes function when compared to other algorithms. Even though lion algorithm is not dominating for the remaining multimodal functions with few local minima, it outperforms two algorithms while solving Hartman 6 and shekel 7 functions, three algorithms while solving shekel 10 function and four algorithms while solving kowalik function.

Based on final ranks of lion algorithm, it can be known that lion algorithm is far better than two, five and four evolutionary computation algorithms when solving unimodal, multimodal with many local minima and multimodal with few local minima functions, respectively.

5.1.2 Phase 2 of test suite 1

In this phase, four popular optimization algorithms such as genetic algorithm (GA) [35], particle swarm optimization (PSO) [36], artificial bee colony algorithm (ABC) [37] and group search optimizer (GSO) [25] are compared with lion algorithm. Like comparison with evolutionary computation algorithms, we have adopted the published results of GA, PSO and GSO from [25] and ABC from [26]. The comparison results for unimodal functions, multimodal functions with many local minima and multimodal functions with few local minima are tabulated in Tables 4, 5 and 6.

Table 4 shows that the lion algorithm is better than GA while solving sphere, schwefel 2.22 and rosenbrock functions. It outperforms GA and GSO while solving schwefel 1.2 function, while it is better than GA, PSO and GSO for solving step functions. The results of schwefel 2.21 and quartic functions showed that lion algorithm is the best among all the five algorithms.

From Table 5, it can be said as lion algorithm dominates PSO and GA while solving schwefel and Ackley functions respectively, whereas it is better than both PSO and GA for solving griewank function. In spite of lion algorithm's

Table 3 Performance comparison between lion algorithm and evolutionary computing methods on multimodal benchmark functions with few local minima

Function	Lion algorithm		CEP		FEP		CES		FES		CMA-ES		ESLAT	
	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R
Foxholes	$1(1.12\times10^{-15})$	1	1.66(1.19)	4	1.22(0.56)	3	2.16(1.82)	6	1.20(0.63)	2	10.44(6.87)	7	1.77(1.37)	5
Kowalik	$6.09\times10^{-4}(2.17\times10^{-4})$	3	$4.7\times10^{-4}(3.0\times10^{-4})$	1	$5.0\times10^{-4}(3.2\times10^{-4})$	2	$1.2\times10^{-3}(1.6\times10^{-5})$	6	$9.7\times10^{-4}(4.2\times10^{-4})$	5	$1.5\times10^{-3}(4.2\times10^{-3})$	7	$8.1\times10^{-4}(4.1\times10^{-4})$	4
SixHump	− 1.03(0)	1	$-1.03(4.9\times10^{-4})$	1	$-1.03(4.9\times10^{-4})$	1	$-1.03(6.0\times10^{-7})$	1	$-1.03(6.0\times10^{-7})$	1	$-1.03(7.7\times10^{-16})$	1	$-1.03(9.7\times10^{-14})$	1
Branin	$0.4(4.49\times10^{-16})$	1	$0.4(1.5\times10^{-7})$	1	$0.4(1.5\times10^{-7})$	1	$0.4(6.0\times10^{-8})$	1	$0.4(6.0\times10^{-8})$	1	$0.4(1.4\times10^{-15})$	1	$0.4(1.0\times10^{-13})$	1
Goldstein-Price	3(0)	1	3.0(0)	1	3.02(0.11)	6	3.0(0)	1	3.0(0)	1	14.34(25.05)	7	$3(5.8\times10^{-14})$	1
Hartman 3	$-3.86(4.49\times10^{-15})$	1	$-3.86(1.4\times10^{-2})$	1	$-3.86(1.4\times10^{-5})$	1	$-3.86(1.4\times10^{-5})$	1	$-3.86(4.0\times10^{-3})$	1	$-3.86(4.8\times10^{-16})$	1	$-3.86(2.9\times10^{-13})$	1
Hartman 6	$-3.26(5.55\times10^{-2})$	5	$-3.28(5.8\times10^{-2})$	2	$-3.27(5.9\times10^{-2})$	4	$-3.24(5.7\times10^{-2})$	6	− 3.23(0.12)	7	$-3.28(5.8\times10^{-2})$	2	$-3.31(3.3\times10^{-2})$	1
Shekel 5	− 6.47(2.36)	4	− 6.86(2.67)	3	− 5.52(1.59)	7	−6.96(3.10)	2	− 5.54(1.82)	6	− 5.86(3.60)	5	− 8.49(2.76)	1
Shekel 7	− 6.65(2.29)	5	− 8.27(2.95)	3	− 5.52(2.12)	7	−8.31(3.10)	2	− 6.76(3.01)	4	− 6.58(3.74)	6	− 8.79(2.64)	1
Shekel 10	− 7.82(2.49)	4	− 9.10 (2.92)	2	− 6.57(3.14)	7	−8.50(1.25)	3	− 7.63(3.27)	5	− 7.03(3.74)	6	− 9.65(2.06)	1
Average rank	2.6		1.9		3.9		2.9		3.3		4.3		1.7	
Final rank	3		2		6		4		5		7		1	

SD standard deviation, *R* rank

Table 4 Performance comparison between lion algorithm, GA, PSO, ABC and GSO on unimodal benchmark functions

Function	Lion algorithm		GA		PSO		ABC		GSO	
	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R
Sphere	$5.45\times10^{-3}(1.09\times10^{-3})$	4	3.17(1.66)	5	$3.69\times10^{-37}(2.46\times10^{-36})$	1	$7.57\times10^{-4}(2.48\times10^{-4})$	3	$1.95\times10^{-8}(1.16\times10^{-8})$	2
Schwefel 2.22	$3.74\times10^{-3}(3.12\times10^{-3})$	4	0.58(0.13)	5	$2.92\times10^{-24}(1.14\times10^{-23})$	1	$8.95\times10^{-4}(1.27\times10^{-4})$	3	$3.70\times10^{-5}(8.62\times10^{-5})$	2
Schwefel 1.2	$2.16\times10^{-2}(8.38\times10^{-3})$	3	9749.91(2594.96)	5	$1.20\times10^{-3}(2.11\times10^{-3})$	2	$7.01\times10^{-4}(2.78\times10^{-4})$	1	5.7829(3.6813)	4
Schwefel 2.21	$4.21\times10^{-2}(4.48\times10^{-2})$	1	7.96(1.51)	5	0.41(0.25)	3	2.72(1.18)	4	$0.1078(3.99\times10^{-2})$	2
Rosenbrock	56.77(45.28)	4	338.56(361.50)	5	37.36(32.14)	2	$9.36\times10^{-1}(1.76)$	1	49.8359(30.1771)	3
Step	$4.64\times10^{-3}(1.21\times10^{-3})$	2	3.70(1.95)	5	0.15(0.42)	4	0(0)	1	$1.60\times10^{-2}(0.1333)$	3
Quartic	$6.73\times10^{-11}(8.50\times10^{-12})$	1	$0.10(3.62\times10^{-2})$	5	$9.90\times10^{-3}(3.53\times10^{-2})$	2	$9.06\times10^{-2}(1.89\times10^{-2})$	4	$7.38\times10^{-2}(9.26\times10^{-2})$	3
Average rank	2.71		5		2.14		2.43		2.71	
Final rank	3		5		1		2		3	

SD standard deviation, *R* rank

Table 5 Performance comparison between lion algorithm, GA, PSO, ABC and GSO on multimodal benchmark functions with many local minima

Function	Lion algorithm		GA		PSO		ABC		GSO	
	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R
Schwefel	− 12559.52(34.02)	4	− 12566.1(2.11)	2	− 9659.61(463.79)	5	$-12563.67(2.36\times10^{1})$	3	$-12569.49(2.21\times10^{-2})$	1
Rastrigin	$1.85\times10^{-6}(4.07\times10^{-6})$	1	0.66(0.36)	3	20.79(5.94)	5	$4.66\times10^{-4}(3.44\times10^{-4})$	2	1.02(0.95)	4
Ackley	$3.72\times10^{-3}(4.43\times10^{-3})$	4	0.87(0.29)	5	$1.34\times10^{-3}(4.24\times10^{-2})$	3	$7.81\times10^{-4}(1.83\times10^{-4})$	2	$2.66\times10^{-5}(3.08\times10^{-5})$	1
Griewank	$7.51\times10^{-2}(5.98\times10^{-2})$	3	$1(6.76\times10^{-2})$	5	0.23(0.44)	4	$8.37\times10^{-4}(1.38\times10^{-3})$	1	$3.13\times10^{-2}(2.88\times10^{-2})$	2
Penalized	2.3(2.89)	5	$4.36\times10^{-2}(5.06\times10^{-2})$	4	$3.95\times10^{-2}(9.1424\times10^{-2})$	3	$6.98\times10^{-4}(2.78\times10^{-4})$	2	$2.76\times10^{-11}(9.17\times10^{-11})$	1
Penalized 2	$2.65\times10^{-5}(6.41\times10^{-5})$	1	$0.17(7.07\times10^{-2})$	5	$5.06\times10^{-2}(0.57)$	5	$7.98\times10^{-4}(2.13\times10^{-4})$	3	$4.69\times10^{-5}(7.001\times10^{-4})$	2
Average rank	3		4		4.17		2.17		1.83	
Final rank	3		4		5		2		1	

SD standard deviation, *R* rank

Table 6 Performance comparison between lion algorithm, GA, PSO, ABC and GSO on multimodal benchmark functions with few local minima

Function	Lion algorithm		GA		PSO		ABC		GSO	
	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R	Mean(SD)	R
Foxholes	$1(1.12\times10^{-15})$	3	$1(4.43\times10^{-3})$	3	1.02(0.15)	5	$9.98\times10^{-1}(3.21\times10^{-4})$	1	$9.98\times10^{-1}(0)$	1
Kowalik	$6.09\times10^{-4}(2.166\times10^{-4})$	3	$7.09\times10^{-3}(7.85\times10^{-3})$	5	$3.81\times10^{-4}(2.51\times10^{-4})$	1	$1.18\times10^{-3}(1.45\times10^{-4})$	4	$4.17\times10^{-4}(3.12\times10^{-4})$	2
SixHump	− 1.03(0)	1	$-1.03(3.13\times10^{-3})$	1	$-1.02(1.28\times10^{-2})$	5	$-1.03(3.04\times10^{-4})$	1	−1.03(0)	1
Branin	$0.4(4.49\times10^{-16})$	1	$0.40(1.04\times10^{-2})$	1	$0.40(6.88\times10^{-2})$	1	$0.40(3.27\times10^{-4})$	1	0.40(0)	1
Goldstein-Price	3(0)	1	7.50(10.4)	5	$3.01(1.21\times10^{-3})$	4	$3(3.09\times10^{-4})$	1	3.0(0)	1
Hartman 3	$-3.86\ (4.49\times10^{-15})$	1	$-3.86(6.28\times10^{-4})$	1	$-3.86(3.21\times10^{-3})$	1	$-3.86(2.77\times10^{-4})$	1	$-3.86(3.8430\times10^{-6})$	1
Hartman 6	$-3.26(5.55\times10^{-2})$	3	$-3.26(6.04\times10^{-2})$	3	$-3.18(6.11\times10^{-2})$	5	$-3.32(1.35\times10^{-4})$	1	$-3.27(5.96\times10^{-2})$	2
Shekel 5	− 6.47(2.36)	3	− 5.17(2.93)	5	− 7.54(3.03)	2	$-10.15(1.17\times10^{-2})$	1	− 6.09(3.46)	4
Shekel 7	− 6.65(2.29)	3	− 5.44(3.28)	5	− 8.36(2.02)	2	$-10.40(3.11\times10^{-4})$	1	− 6.55(3.24)	4
Shekel 10	− 7.82(2.49)	3	− 4.91(3.49)	5	− 8.94(1.63)	2	$-10.54(2.02\times10^{-3})$	1	− 7.4(3.21)	4
Average rank	2.2		3.4		2.8		1.3		2.1	
Final rank	3		5		4		1		2	

SD standard deviation, *R* rank

failure to perform in solving penalized function, it is proved to be the best algorithm for rastrigin and penalized 2 functions (source: Table 5).

From Table 6, the performance of lion algorithm over other algorithms for solving multimodal functions with few local minima can be seen. Lion algorithm outperforms PSO when solving foxholes and hartman6 functions. While solving these functions, both GA and LA shares third position, however performance deviation (standard deviation) is lesser than GA. For shekel 5, shekel 7 and shekel 10 functions, lion algorithm is found to be better than GA and GSO, while it dominates GA and ABC for solving kowalik functions. Lion algorithm finds the global solution of sixhump and gold-stein price function at every run (because there is zero standard deviation) and remains in the top position among all the algorithms. It again remains in the top position while solving Hartman 3 function with minimum standard deviation. For branin function, though lion algorithm found global minima, it exhibits a second minimum standard deviation function when compared to other algorithms. Despite the lion algorithm is not securing first ranks, it finds equivalent to GSO and ESLAT as per no free lunch theorem [38], and outperforming over other algorithms.

5.2 Test suite 2

The second test suite of this paper consists of 42 benchmark functions. The performance of lion algorithm on this test suite 2 is compared with the swarm algorithms such as Artificial Bee colony (ABC) [37], Bacterial Foraging Optimization Algorithm (BFO) [39], Cuckoo Search (CS) [40], Firefly (FF) [41], Group Search Optimization (GSO) [25], Moth-flame Optimization (MFO) [42], particle swarm optimization [36], Dragon Fly algorithm [43], Grey Wolf Optimization(GWO) [44], Whale Optimization Algorithm (WOA) [45] and Crow Search Algorithm (CrS) [46]. The results are presented in Tables 7 and 8. Subsequently, the lion algorithm is compared with three renowned evolutionary algorithms such as biogeography-based optimization (BBO) [47], differential evolution (DE) [48] and genetic algorithm (GA) [35] as well as three human/physics inspired algorithms such as gravitational search algorithm (GSA) [49], harmony search algorithm (HSA) [50] and simulated annealing (SA) [51] and presented in Tables 9 and 10.

Each algorithm attains diverse ranks on finding the optimal solution for the 42 benchmark functions. When compared with the swarm intelligence, the mean performance of the lion algorithm positions it in the second position (as per Table 7), but the performance meets low deviation (as per Table 7). This shows that the lion algorithm dominates all the swarm intelligence, except GSO. However, the lion algorithm finds stable searching of solution at any instant.

Table 7 Comparison on mean performance among the lion and swarm algorithms while solving test suite 2

Functions	ABC	BFO	CS	FF	GSO	MFO	PSO	GWO	WOA	CrS	DA	Lion
1	7e−6(8)	2.05(11)	0(1)	5e−3(10)	5e−12(6)	13.03(12)	0(1)	0(1)	0(1)	7e−6(8)	9e−7(7)	0(1)
2	1.3e−4(9)	2.21e−7(6)	0.02(11)	6.67e−10(3)	8.4e−6(7)	0.68(12)	0(1)	3.63e−8(4)	6.19e−10(2)	1.3e−4(9)	1.6e−5(8)	6.98e−8(5)
3(−)	106.76(1)	106.76(1)	106.65(10)	106.76(1)	106.76(1)	37.58(12)	106.76(1)	102.87(11)	106.76(1)	106.76(1)	106.76(1)	106.76(1)
4	2.32e−4(9)	1.23e−7(7)	0(1)	2.73e−9(5)	1.15e−12(4)	4.52(12)	0(1)	4.81e−7(8)	9.31e−4(11)	2.32e−4(9)	1.72e−8(6)	0(1)
5	1.67e−14(4)	3.35e−3(9)	6.78e−61(2)	8.68e−8(6)	8.75e−18(3)	1.77e−1(10)	3.05(11)	1.33e−5(7)	3.2e−5(8)	1.67e−14(4)	3.05(11)	0(1)
6	0.34813(7)	0.207205(6)	0.033416(2)	0.062809(4)	0.042762(3)	22.18795(10)	35.03475(11)	0.393046(9)	0.064273(5)	0.34813(7)	35.03475(11)	0(1)
7(−)	24.16(1)	24.16(1)	24.16(1)	24.16(1)	24.16(1)	20.33(11)	22.25(10)	24.1568(1)	24.15(9)	24.1568(1)	1.42(12)	24.16(1)
8(−)	42.94(1)	42.68(10)	42.93(4)	42.67(10)	42.92(5)	31.38(12)	42.85(6)	42.77(8)	42.76(9)	42.94(1)	42.85(6)	42.94(1)
9	4.85e−5(1)	4.85e−5(1)	4.85e−5(1)	4.85e−5(1)	4.85e−5(1)	5.21e−5(12)	4.85e−5(1)	4.85e−5(1)	4.85e−5(1)	4.85e−5(1)	4.85e−5(1)	4.85e−5(1)
10(−)	2.06(1)	2.06(1)	2.06(1)	2.06(1)	2.06(1)	1.99(12)	2.06(1)	2.06(1)	2.06(1)	2.06(1)	2.06(1)	2.06(1)
11(−)	8.4e−4(7)	4.9e−4(11)	1.83e−2(5)	6.9e−4(9)	3.51e−2(4)	1.3e−4(12)	0.03.88e−2(3)	0.20(2)	5.4e−4(10)	8.4e−4(7)	1.20e−2(6)	1(1)
12	1.4e−1(6)	1.97e−1(9)	2.9e−2(5)	1.90e−1(8)	5.9e−3(2)	9.4e−1(12)	1.7e−2(3)	3.3e−1(11)	1.9e−1(10)	1.4e−1(6)	2.4e−2(4)	1e−4(1)
13	137.9(8)	6e9(11)	65.70(5)	118.4(7)	49.49(2)	3e11(12)	3e5(10)	70.22(6)	7.27(1)	137.953(8)	64.25(4)	59.92(3)
14(−)	1(1)	0.6(9)	1(1)	0(10)	1(1)	0(10)	0(10)	1(1)	1(1)	1(1)	1(1)	0.8(8)
15(−)	5.61e3(5)	3.3e3(11)	5.9e3(3)	5.49e3(7)	6.7e3(2)	2.4e3(12)	4.7e3(8)	4.5e3(9)	5.67e3(4)	5.61e3(5)	3.6e3(10)	6.8e3(1)
16	0.06447(1)	0.06447(1)	0.06447(1)	0.06447(1)	0.06447(1)	0.069798(12)	0.06447(1)	0.06447(1)	0.064471(11)	0.06447(1)	0.06447(1)	0.06447(1)
17	3.001603(9)	3.000017(7)	3(1)	3(1)	3(1)	70.07004(12)	3(1)	3.000042(8)	3.00001(6)	3.001603(9)	8.4(11)	3(1)
18	7.37e−6(3)	0.232986(10)	0.029124(5)	0	0(1)	5.08483(11)	0.029124(5)	0.029124(5)	0.029124(5)	7.37e−06(3)	0.058247(9)	0(1)
19	0.15(2)	50.95(11)	4.29(10)	5e−5(1)	0.98(5)	1e4(12)	0.25(4)	1.26(7)	3.48(9)	0.15(2)	1.57(8)	1.23(6)
20	1e−7(7)	2e−6(9)	1e−16(4)	5e−9(6)	7e−19(3)	10(12)	0(1)	7e−6(10)	2e−4(11)	1e−7(7)	7e−12(5)	0(1)
21(−)	19.2085(1)	19.2085(1)	19.2085(1)	18.3948(10)	19.2085(1)	16.9112(12)	17.6822(11)	19.2085(1)	19.2085(1)	19.2085(1)	19.2085(1)	19.2085(1)
22	3.32e−3(9)	1.3e−5(5)	0(1)	4.33e−10(3)	3.42e−4(7)	1.75(12)	0.00157(8)	6.66e−7(4)	4.79e−5(6)	3.32e−3(9)	9.64e−3(11)	0(1)
23	5.4e−14(4)	8.79e−2(11)	1.35e−31(1)	8.98e−09(7)	3.84e−22(3)	2.259259(12)	4.16e−05(10)	8.13e−08(8)	9.09e−06(9)	5.4e−14(4)	4.36e−10(6)	1.35e−31(1)
24	7.84e−05(10)	3.48e−09(8)	5.61e−50(5)	1.26e−10(7)	3.92e−14(6)	0.277179(12)	1.09e−80(4)	6e−105(3)	1.6e−291(2)	7.84e−05(10)	1.11e−07(9)	0(1)
25(−)	1.913 (1)	1.913 (1)	1.913(1)	1.913(1)	1.913(1)	1.022(12)	1.910(10)	1.913(1)	1.913(1)	1.913(1)	1.910(10)	1.913(1)
26	15.22e−7(7)	0.025815(11)	48.43e−4(10)	6.92e−11(6)	0(1)	0.49(12)	0(1)	0(1)	0(1)	15.22e−4(7)	18.76e−4(9)	0(1)
27	7.97e−5(5)	0.04 (11)	7.59e−4(10)	1.67e−10(4)	0(1)	0.31(12)	0(1)	0(1)	1.92e−4(8)	7.97e−5(5)	3.64e−4(9)	1.52e−4(7)
28(e−4)	22.09(6)	465(11)	23.52(8)	15.67(1)	17.24(4)	4077(12)	54.82(9)	15.7(2)	15.83(3)	22.09(6)	19.11(5)	114.66(10)
29	0.292845(5)	0.307338(11)	0.293249(8)	0.292579(1)	0.292579(1)	0.441306(12)	0.295736(9)	0.292579(1)	0.292615(4)	0.292845(5)	0.293019(7)	0.295745(10)
30(−)	0.96(1)	0.96(1)	0.96(1)	0.96(1)	0.96(1)	0.84(12)	0.96(1)	0.96(1)	0.96(1)	0.96(1)	0.94(11)	0.96(1)
31	127.3516(7)	16495.09(11)	246.9543(10)	2.67e−5(3)	0.022836(5)	212383.7(12)	3.53096(6)	3.02e−8(1)	2.44e−5(2)	127.3516(7)	0.001474(4)	144.740(9)
32	4e−6(8)	8e−1(10)	0(1)	6e−8(7)	0(1)	1e1(12)	0(1)	0(1)	0(1)	4e−6(8)	2.3(11)	0(1)
33	1e3(8)	3e7(11)	99(2)	2e2(6)	1e2(3)	1e9(12)	4e5(10)	8(1)	4e2(7)	1e3(8)	1.5e2(4)	1.8e2(5)
34(−)	837.966(1)	816.26(10)	837.966(1)	837.966(1)	837.966(1)	500.739(12)	837.966(1)	766.901(11)	837.949(9)	837.966(1)	837.966(1)	837.966(1)
35	1.67063(7)	3.315345(11)	1.645522(6)	2.674383(9)	0.457445(2)	4.12898(12)	3.05646(10)	0.497562(3)	0.695547(4)	1.67063(7)	0.735808(5)	0.410484(1)
36(−)	1.03(1)	1.03(1)	1.02(11)	1.03(1)	1.03(1)	−0.4(12)	1.03(1)	1.03(1)	1.03(1)	1.03(1)	1.03(1)	1.03(1)
37(−)	391.319(3)	354.904(8)	383.18(5)	357.734(7)	391.661(2)	198.482(12)	348.118(9)	306.56(11)	361.951(6)	391.319(3)	332.287(10)	391.662(1)
38(−)	10.8723(1)	10.8433(9)	10.842(10)	10.8683(4)	10.8723(1)	10.4807(11)	10.8644(5)	10.8644(5)	10.8644(5)	10.8723(1)	10.4704(12)	10.8591(8)
39	2e−12(7)	5e−8(10)	6e−58(5)	3.19e−10(9)	4.62e−23(6)	1.25(12)	2e−70(4)	7e−194(2)	4e−94(3)	2e−12(7)	0.17(11)	0(1)
40	3e−1(8)	5−4(3)	8e−2(6)	1e−1(7)	2e−3(4)	3.4(10)	4.0(11)	32(12)	3e−3(5)	3e−1(8)	8e−5(1)	2e−4(2)
41	12(7)	8e3(10)	0.96(3)	0.64(2)	1.7(4)	7e7(12)	5e4(11)	3.21(6)	2.82(5)	12(7)	0.57(1)	25.5(9)

Table 7 (continued)

Functions	ABC	BFO	CS	FF	GSO	MFO	PSO	GWO	WOA	CrS	DA	Lion
42(−)	3.7e−3(1)	3.7e−3(1)	0(11)	3.7e−3(1)	3.7e−3(1)	−0.73(12)	3.7e−3(1)	3.7e−3(1)	3.7e−3(1)	3.7e−3(1)	3.7e−3(1)	3.7e−3(1)
Mean rank	4.73	7.33	4.54	4.52	2.64	11.73	5.33	4.5	4.78	4.73	6.26	2.66
Final rank	6	11	5	4	1	12	9	3	8	6	10	2

(−) symbol in the "Functions" column represents the values given in the respective row are negative

According to Tables 8, 9 and 10, the lion algorithm manages to secure first rank over the evolutionary algorithms and human/physics inspired algorithms.

5.3 Test suite 3

Here, the lion algorithm is tested on 17 benchmark functions, which have been widely used by various researchers to assess the performance of optimization algorithms [25–27]. However, they are not large scale. This test suite considers these functions in large scale, i.e., the dimension of the solution as 1000, because of its high significance [25]. Similar to test suite 2, the comparative study is conducted with swarm algorithms and evolutionary, human/physics inspired algorithms separately.

According to Tables 11, 12, 13 and 14, the lion algorithm finds first position in handling large scale optimization problems. The multiple updating stages in the lion algorithm, updating based on self-improvement, exploration and exploitation leads the solution update in lion algorithm in all possible dimensions. Moreover, the provision to find the diverse solution with similar fitness function makes it highly suitable for multimodal functions, which are in multiple in large scale problems.

5.4 Test suite 4

Test suite 4 has five widely exploited engineering problems such as welded beam design, pressure vessel design, gear train design, tension/compression spring design and three-bar truss design. The problem models are referred from [46]. Similar to test suite 2 and 3, the comparative study is conducted against swarm algorithms, evolutionary algorithms and physics/human inspired algorithms. Each algorithm is subjected to 20 test runs and the resultant statistics are presented in Tables 15 and 16.

Tables 15 and 16 reveals interesting outcomes about the performance of the lion algorithm over the conventional algorithms. The lion algorithm outperforms swarm algorithms on solving welded beam, pressure vessel design and gear train design, while it secures second position on solving compression/tension spring design and three-bar truss design. On contrary, the lion algorithm outperforms evolutionary algorithms and human/physics inspired algorithms on solving three-bar truss design, gear train design and welded beam design. The deviation from mean performance (standard deviation) gets fluctuated in the lion algorithm. However, the lion algorithm maintains maximum trade-off between the mean performance and standard deviation to outperform both the group of algorithms and hence remain in the first position of all the algorithms. The obtained best solution by the lion algorithm is given in Table 17.

Table 8 Comparison on performance deviation among the lion algorithm and swarm algorithms while solving test suite 2

Functions	ABC	BFO	CS	FF	GSO	MFO	PSO	GWO	WOA	CrS	DA	Lion
1	1e−5(9)	2.8(11)	0(1)	2.8e−4(10)	6e−12(6)	5.3(12)	0(1)	0(1)	0(1)	2e−8(7)	2e−6(8)	0(1)
2	1.4e−4(10)	2e−7(6)	5e−2(11)	6e−10(3)	1.3e−5(7)	0.9(12)	0(1)	3.3e−8(4)	3.3e−10(2)	7.18e−5(9)	3.4e−5(8)	1.6e−7(5)
3	6.7e−7(6)	3.3e−6(7)	0.2(8)	1.1e−7(5)	4e−9(4)	21(12)	2.3e−14(1)	11.7(10)	10.7(9)	14(11)	10e−12(2)	6.1e−10(3)
4	1.7e−4(9)	1.6e−7(7)	0(1)	3e−9(5)	2.5e−12(4)	4.2(12)	0(1)	3.3e−7(8)	2.2e−4(10)	3.4e−4(11)	3.2e−8(6)	0(1)
5	3e−14(7)	4e−3(11)	1.2e−60(4)	7.1e−8(8)	2e−17(5)	2.1(12)	0(1)	5e−6(10)	3.8e−6(9)	6.1e−15(6)	0(1)	0(1)
6	0.2(9)	0.067(8)	0.02(5)	0.03(6)	0.01(4)	13.3(12)	0(1)	0.3(10)	0.063(7)	0.4(11)	0(1)	0(1)
7	3e−9(5)	9.7e−7(7)	8e−15(1)	6.2e−8(6)	2.9e−9(4)	2.2(9)	1.7(8)	22(11)	21.2(10)	27.5(12)	4.5e−14(2)	9e−14(3)
8	4.3e−13(3)	0.244948(8)	0.02(4)	0.244949(9)	0.05(5)	0(1)	0.2(6)	42.9439(11)	42.9008(10)	42.9444(12)	0.2(6)	3.6e−15(2)
9	2e−15(7)	1.8e−13(8)	1.4e−16(5)	1.8e−15(6)	1.5e−17(4)	2.7e−6(9)	9.4e−18(3)	4.85e−5(10)	4.85e−5(10)	4.85e−5(10)	3e−20(2)	4.8e−21(1)
10	1.8e−13(4)	3.6e−9(8)	1.3e−12(6)	8.6e−11(7)	3.4e−13(5)	0.03(9)	0(1)	2.06261(10)	2.06261(10)	2.0626(10)	6e−16(3)	0(1)
11	0.000427(7)	0.000131(3)	0.037166(11)	0.000266(4)	0.026721(10)	1.38e−05(2)	0.042(12)	0.00041(6)	0.00037(5)	0.00085(8)	0.015473(9)	0(1)
12	0.058907(8)	0.028045(7)	0.026561(6)	0.116696(10)	0.00346(3)	0(1)	0.018118(4)	0.388848(12)	0.208012(11)	0.11649(9)	0.022341(5)	0(1)
13	70.36718(7)	5.84e+09(12)	130.9267(10)	82.1021(8)	44.4926(4)	0(1)	829384.3(11)	6.690757(2)	7.489887(3)	112.1352(9)	56.56213(6)	44.7719(5)
14	5.04e−06(8)	0.5477(9)	0(1)	0(1)	0(1)	0(1)	0(1)	1.9890(12)	1.4657(11)	1.4563(10)	3.74e−07(7)	0(1)
15	327.6038(2)	258.4003(1)	609.5496(4)	928.2179(8)	1269.438(9)	900.3274(7)	572.9415(3)	4648.7(10)	5789.74(12)	5634.781(11)	832.8938(5)	863.7747(6)
16	0(1)	3.49e−08(8)	0(1)	5.35e−12(7)	0(1)	0.000216(9)	0(1)	0.06447(10)	0.06447(10)	0.064471(12)	9.93e−17(5)	1.45e−12(6)
17	0.001779(8)	1.59e−05(7)	1.02e−15(3)	1.85e−08(6)	3.43e−15(5)	0(1)	1.2e−15(4)	3.000024(10)	3.000008(9)	3.0017241(11)	12.07477(12)	0(1)
18	1.54e−05(7)	0.19536(11)	0.065123(8)	4.41e−09(6)	0(1)	0.573513(12)	0.065123(8)	0(1)	0(1)	6.41e−071(5)	0.079759(10)	0(1)
19	0.11864(3)	107.6277(11)	4.095571(9)	5.94e−05(1)	1.647446(7)	4022.857(12)	0.578055(5)	0.003118(2)	4.640429(10)	0.1689671(4)	1.299141(6)	2.767411(8)
20	1.78e−07(8)	2.1e−06(9)	3.62e−16(5)	5.93e−09(7)	1.63e−18(4)	14.03499(12)	0(1)	4.35e−06(10)	8.02e−05(11)	9.56e−091(3)	1.15e−11(6)	0(1)
21	7.08e−09(4)	1.24e−06(5)	6.94e−10(3)	1.81939(8)	2.31e−06(6)	1.969413(9)	1.51121(7)	17.2985(10)	18.1075(11)	19.20871(12)	2.18e−11(1)	6.78e−11(2)
22	0.003004(9)	2.37e−05(6)	0(1)	5.5e−10(3)	0.000598(7)	1.712858(12)	0.002148(8)	5.19e−07(4)	8.24e−06(5)	0.004554(10)	0.015625(11)	0(1)
23	1.2e−13(6)	0.049138(12)	0(1)	7.22e−09(8)	8.31e−22(4)	0(1)	9.3e−05(11)	5.03e−08(9)	9.53e−07(10)	1.54e−16(5)	9.73e−10(7)	0(1)
24	6.75e−05(11)	2.68e−09(8)	1.25e−49(5)	1.04e−10(7)	3.26e−14(6)	0.140788(12)	2.44e−80(4)	2.2e−106(3)	0(1)	6.35e−05(10)	2.38e−07(9)	0(1)
25	1.85e−06(9)	1.13e−07(8)	1.06e−11(5)	1.78e−10(7)	2.31e−15(3)	0(1)	0(1)	1.95322(12)	1.87322(11)	1.561322(10)	3.36e−12(4)	1.96e−11(6)
26	0.001514(9)	0.031229(12)	0.01083(11)	5.25e−11(7)	0(1)	0(1)	0(1)	0(1)	0(1)	0.000692(8)	0.001713(10)	0(1)
27	7.26e−05(6)	0.042427(11)	0.001039(10)	2.2e−10(5)	0(1)	0.164585(12)	0(1)	0(1)	0(1)	7.61e−05(7)	0.00048(9)	0.000339(8)
28	0.000959(4)	0.042357(11)	0.001755(8)	3.57e−08(1)	0.000351(2)	0.055595(12)	0.005484(9)	0.001569(5)	0.001569(5)	0.001683(7)	0.00037(3)	0.01794(10)
29	0.000317(3)	0.012742(8)	0.001499(5)	3.19e−09(2)	2.78e−17(1)	0.068991(9)	0.005335(6)	0.292579(10)	0.292592(11)	0.292767(12)	0.000512(4)	0.005382(7)
30	3.4e−13(4)	1.61e−09(8)	2.85e−12(5)	4.45e−11(7)	1.24e−16(2)	0(1)	1.24e−16(2)	0.998353(12)	0.76353(11)	0.46453(10)	0.029823(9)	6.73e−12(6)
31	124.546(9)	21557.48(12)	330.8466(11)	2.52e−05(4)	0.046194(6)	0(1)	7.891553(7)	2.15e−08(2)	3.64e−06(3)	75.2373(8)	0.001084(5)	323.6487(10)
32	7.61e−06(9)	1.089916(10)	0(1)	6.68e−08(7)	0(1)	8.874322(12)	0(1)	0(1)	0(1)	1.43e−07(8)	3.696095(11)	0(1)
33	863.8596(10)	33436062(12)	27.50695(3)	182.0598(8)	66.66746(4)	0(1)	521347.4(11)	2.975421(2)	67.71243(5)	849.3706(9)	70.92023(6)	112.9015(7)
34	1.25e−05(8)	26.4373(9)	0(1)	3.15e−07(6)	0(1)	0(1)	0(1)	719.527(10)	887.935(12)	837.966(11)	7.31e−12(5)	4.75e−07(7)
35	0.223436(3)	0.243802(4)	0.959855(11)	0.425906(8)	0.295666(5)	0.085516(1)	0.479492(9)	0.354273(6)	0.087685(2)	1.673007(12)	0.38608(7)	0.514455(10)
36	2.23e−13(3)	2.4e−07(7)	0.014145(8)	2.28e−09(6)	0(1)	0.580276(9)	0(1)	1.23163(10)	2.0163(11)	3.04227(12)	5.02e−13(4)	1.13e−11(5)
37	0.257898(4)	12.64435(7)	7.743(5)	16.11834(8)	0.002168(3)	0(1)	9.825015(6)	320.977(10)	361.477(11)	391.285(12)	20.96815(9)	3.11e−07(2)
38	5.22e−08(2)	0.020557(5)	0.033404(7)	0.008858(3)	0(1)	0.341338(9)	0.010849(4)	18.8713(12)	17.833(11)	11.8523(10)	0.213589(8)	0.029416(6)
39	3.31e−12(8)	4.72e−08(10)	1.53e−57(5)	4.64e−10(9)	7.33e−23(6)	0.48832(12)	3.21e−70(4)	4.4e−207(2)	7.1e−109(3)	2.34e−16(7)	0.163571(11)	0(1)
40	0.153288(7)	0.000487(3)	0.059476(6)	0.164493(8)	0.005252(5)	0.332116(10)	0.361793(11)	14.25412(12)	0.00049(4)	0.261029(9)	4.83e−05(1)	0.000355(2)
41	6.979019(7)	8409.505(10)	0.842756(2)	1.424354(4)	2.42161(6)	69800534(12)	41081.79(11)	1.786854(5)	1.095842(3)	14.02005(8)	0.518085(1)	36.14519(9)

Table 8 (continued)

Functions	ABC	BFO	CS	FF	GSO	MFO	PSO	GWO	WOA	CrS	DA	Lion
42	5.75e−11(6)	2.37e−08(8)	0(1)	9.32e−11(7)	3.07e−19(3)	0.466914(12)	3.07e−19(3)	0.00379(9)	0.00379(9)	0.00379(9)	4.37e−16(5)	0(1)
Average rank	6.404762	8.214286	5.214286	6.095238	4	7.357143	4.595238	7.333333	7.214286	9.214286	5.952381	3.666667
Final rank	7	11	4	6	2	10	3	9	8	12	5	1

(−) symbol in the "Functions" column represents the values given in the respective row are negative

6 Conclusion and future work

In the family of nature-inspired algorithms, a new optimization algorithm called as lion algorithm had been proposed from the inspiration of lion's unique social behavior, which keeps the animal stronger than other mammals. A detailed study on the performance of the lion algorithm using four diverse test suites has been conducted in this paper. First two test suites are benchmark optimization problems with unimodal and multimodal characteristics. The results from first test suite are compared against the published results of evolutionary computation methods such as CES, FES, CEP, FEP, CMA-ES, ESLAT, and other popular optimization algorithms such as GA, PSO, ABC and GSO. In test suite 2, a wide comparison has been performed with state-of-the-art optimization algorithms. Test suite 3 has been presented as large-scale optimization problems and test suite 4 has engineering optimization problems. The comparative analysis has disclosed interesting outcomes about the lion algorithm. It was found to be competing over majority of the evolutionary computation methods when solving multimodal functions. In addition, it has better-averaged results than GA and PSO. For unimodal functions, lion algorithm has outperformed over two other evolutionary computation methods, whereas equal to GSO and better than GA. Trade-off has also been maintained between the mean performance and standard deviation and so the stability of the achieved performance has been guaranteed. The algorithm was also found to be competing over swarm algorithms on solving welded beam design problem, pressure vessel design problem and gear train design problem. While solving three bar truss design and tension/compression spring design problem, the lion algorithm outperforms evolutionary and human/physics inspired algorithms. Despite the lion algorithm underperforms certain algorithms on solving these test suites, the final rank discloses remarkable performance of the lion algorithm.

As the obtained results are encouraging, we have planned to apply lion algorithm on various engineering problems and applications to study its performance. Perhaps, lion algorithm requires certain amendments according to the problems. However, the presented algorithm structure can play as a cornerstone for all the future enhancements.

Table 9 Comparison on mean performance among the lion, evolutionary and human/physics inspired algorithms while solving test suite 2

Functions	BBO	DE	GA	GSA	HAS	SA	Lion
1	9.08e−07(2)	9.367753(5)	12.81959(7)	12.07019(6)	0.000177(3)	0.030306(4)	0(1)
2	1.66e−05(3)	0.171017(6)	0.000494(4)	2.634131(7)	1.07e−06(2)	0.003964(5)	6.98e−08(1)
3(−)	106.765(1)	87.2947(6)	106.74(5)	12.3598(7)	106.765(1)	106.764(4)	106.765(1)
4	1.72e−08(3)	9.070845(7)	0.000444(4)	7.472382(6)	1.09e−08(2)	0.062209(5)	0(1)
5	3.053732(6)	3.043247(5)	0.015376(3)	1.251493(4)	3.053732(6)	0.001562(2)	0(1)
6	35.03475(5)	13.97583(4)	1.619303(3)	35.03475(5)	35.03475(5)	0.189471(2)	0(1)
7(−)	1.42781(7)	3.59099(6)	9.16902(5)	20.8517(4)	24.1568(1)	24.1568(1)	24.1568(1)
8(−)	42.8549(4)	32.1808(7)	38.4256(6)	39.527(5)	42.9444(1)	42.9375(3)	42.9444(1)
9	4.85e−05(1)	5.04e−05(5)	5.11e−05(6)	5.9e−05(7)	4.85e−05(1)	4.85e−05(1)	4.85e−05(1)
10(−)	2.06261(1)	2.01386(6)	2.0626(5)	1.97379(7)	2.06261(1)	2.06261(1)	2.06261(1)
11(−)	0.01203(4)	0.0001(7)	0.00018(6)	0.00061(5)	0.01736(3)	0.02743(2)	1(1)
12	0.02429(2)	0.634307(6)	0.558454(5)	0.838567(7)	0.133322(4)	0.12931(3)	0.0001(1)
13	64.25729(2)	6.21e+10(6)	1.23e+11(7)	457.8695(4)	149.3677(3)	5.45e+08(5)	59.9248(1)
14(−)	1(1)	8.2e−93(7)	0.19992(5)	0.1973(6)	1(1)	0.4921(4)	0.80002(3)
15(−)	3623.93(4)	2447.51(5)	2030.8(6)	1676.03(7)	6989.35(1)	6144.49(3)	6877.68(2)
16	0.06447(1)	0.065211(6)	0.064547(5)	0.069895(7)	0.06447(1)	0.064471(4)	0.06447(1)
17	8.4(4)	36.15145(6)	8.457085(5)	70.07004(7)	3(1)	3.165408(3)	3(1)
18	0.058247(5)	2.605944(6)	4.768995(7)	0.008596(2)	0.029124(4)	0.020782(3)	0(1)
19	1.573239(3)	4089.703(7)	2319.044(6)	0.77763(1)	3.207501(4)	11.24056(5)	1.237624(2)
20	7.63e−12(2)	5.23598(6)	0.001857(4)	32.86475(7)	1.42e−08(3)	0.003321(5)	0(1)
21(−)	19.2085(1)	13.2169(7)	19.2072(5)	17.8899(6)	19.2085(1)	19.2085(1)	19.2085(1)
22	0.009648(4)	0.240421(6)	0.001916(3)	3.801337(7)	0.032138(5)	0.000617(2)	0(1)
23	4.36e−10(2)	1.724852(6)	0.024257(5)	2.259259(7)	6.73e−08(3)	0.001033(4)	1.35e−31(1)
24	1.11e−07(3)	0.377263(6)	0.000382(4)	0.422321(7)	5.38e−08(2)	0.003399(5)	0(1)
25(−)	1.91051(4)	0.10808(7)	1.91318(3)	1.05925(6)	1.91051(4)	1.91322(1)	1.91322(1)
26	0.001876(2)	0.010046(5)	0.159705(6)	0.189236(7)	0.003127(3)	0.005055(4)	0(1)
27	0.000364(2)	0.287496(7)	0.232103(6)	0.130523(5)	0.001659(3)	0.005151(4)	0.000152(1)
28	0.001911(1)	0.2133(7)	0.091343(5)	0.14227(6)	0.002894(2)	0.005115(3)	0.011466(4)
29	0.293019(2)	0.296375(5)	0.334012(6)	0.356616(7)	0.292795(1)	0.293301(3)	0.295745(4)
30 (−)	0.94382(5)	0.94623(4)	0.94039(6)	0.85892(7)	0.96353(1)	0.96353(1)	0.96353(1)
31	0.001474(2)	58243.37(6)	54404.35(5)	22083.49(4)	0.00119(1)	14671.54(3)	0
32	2.387896(4)	10.72624(6)	8.777119(5)	0	5.52e−08(2)	0.114777(3)	0(1)
33	151.1211(1)	2.62e+08(4)	3.92e+08(5)	0	267.9483(2)	415604.7(3)	0
34(−)	837.966(1)	561.572(5)	561.45(6)	500.739(7)	837.966(1)	837.963(4)	837.966(1)
35	0.735808(3)	3.856049(7)	3.753272(5)	3.789925(6)	0.552869(2)	1.965483(4)	0.410484(1)
36	− 1.03163(1)	− 0.23691(6)	− 0.70381(5)	0.697168(7)	− 1.03163(1)	− 1.0315(4)	− 1.03163(1)
37(−)	332.287(5)	246.396(7)	344.262(4)	250.308(6)	391.662(1)	368.045(3)	391.662(1)
38(−)	10.4704(5)	10.5961(4)	10.3745(6)	10.0504(7)	10.8723(1)	10.8656(2)	10.8591(3)
39	0.179183(5)	0.011526(4)	0.17936(6)	1.595538(7)	6.49e−07(2)	0.000538(3)	0(1)
40	8.14e−05(2)	356.0894(7)	0.849241(4)	2.211149(6)	3.89e−05(1)	1.361192(5)	0.000175(3)
41	0.576521(1)	472861.3(6)	23831636(7)	79261.05(5)	38.72378(3)	2434.018(4)	25.57872(2)
42	− 0.00379(1)	0.501655(6)	− 0.00361(4)	1.231383(7)	− 0.00379(1)	− 0.00273(5)	− 0.00379(1)
Average rank	2.809524	5.880952	5.119048	5.666667	2.166667	3.238095	1.309524
Final rank	3	7	5	6	2	4	1

(−) symbol in the "Functions" column represents the values given in the respective row are negative

Table 10 Comparison on performance deviation among the lion, evolutionary and human/physics inspired algorithms while solving test suite 2

Functions	BBO	DE	GA	GSA	HAS	SA	Lion
1	1.58e−06(2)	0.833183(5)	5.115353(6)	7.39228(7)	0.000186(3)	0.026977(4)	0(1)
2	3.44e−05(3)	0.03589(6)	0.000297(4)	1.135101(7)	1.85e−06(2)	0.004443(5)	1.56e−07(1)
3	9.87e−12(2)	3.921394(7)	0.019734(6)	0(1)	1.42e−07(4)	6.87e−05(5)	6.1e−10(3)
4	3.2e−08(3)	0.142784(6)	0.000766(4)	4.177189(7)	1.05e−08(2)	0.045378(5)	0(1)
5	0(1)	3.42e−05(4)	0.019667(6)	1.645212(7)	0(1)	0.001793(5)	0(1)
6	0(1)	1.99e−15(5)	1.04194(7)	0(1)	0(1)	0.219(6)	0(1)
7	4.5e−14(1)	1.032548(6)	8.688083(7)	0.192287(5)	4.21e−08(3)	7.81e−06(4)	8.56e−09(2)
8	0.2(4)	0.320822(5)	5.575198(7)	4.85804(6)	2.05e−06(2)	0.011949(3)	3.55e−15(1)
9	3.01e−20(3)	2.61e−07(6)	2.43e−06(7)	0(1)	2.62e−15(4)	2.88e−11(5)	4.79e−21(2)
10	5.87e−16(3)	0.006036(7)	6.47e−06(6)	0(1)	5.12e−10(4)	5.71e−07(5)	0(1)
11	0.015473(5)	1.15e−05(2)	5.8e−05(3)	0.001117(4)	0.037689(7)	0.03188(6)	0(1)
12	0.022341(2)	0.054636(4)	0.04(3)	0.111799(7)	0.074926(6)	0.066478(5)	0(1)
13	56.56213(1)	1.1e+10(6)	7.2e+10(7)	596.5479(4)	86.56948(2)	6.52e+08(5)	123.7719(3)
14	3.74e−07(3)	1.83e−92(1)	0.447036(5)	0.441186(4)	7.01e−08(2)	0.500293(7)	0.447177(6)
15	832.8938(5)	251.1806(2)	930.4567(7)	220.4297(1)	548.327(3)	687.1904(4)	863.7747(6)
16	9.93e−17(2)	0.0002(7)	0.000108(6)	0(1)	4.46e−11(4)	1.51e−07(5)	1.45e−12(3)
17	12.07477(7)	0.669572(5)	12.05432(6)	0(1)	2.65e−08(3)	0.203371(4)	0(1)
18	0.079759(5)	1.135114(6)	5.250773(7)	0.018857(3)	0.065123(4)	0.015953(2)	0(1)
19	1.299141(2)	324.685(6)	3329.536(7)	0.98032(1)	2.237334(3)	9.463523(5)	2.767411(4)
20	1.15e−11(2)	0.75275(6)	0.0023(4)	2.918562(7)	9.11e−09(3)	0.004794(5)	0(1)
21	2.18e−11(1)	0.854161(7)	0.001738(5)	0.205887(6)	7.39e−09(3)	2.58e−05(4)	6.78e−11(2)
22	0.015625(4)	0.021648(5)	0	0(1)	0.045098(6)	0.000554(3)	0(1)
23	9.73e−10(3)	0.020334(6)	0.053611(7)	0(1)	8.69e−08(4)	0.001307(5)	0(1)
24	2.38e−07(4)	0.006303(7)	0.000629(5)	0(1)	4.82e−08(3)	0.003016(6)	0(1)
25	3.36e−12(1)	1.252654(7)	5.75e−05(5)	0.08224(6)	1.2e−11(2)	2.82e−06(4)	1.96e−11(3)
26	0.001713(4)	0.001392(3)	0.152669(7)	0.131521(6)	1.97e−07(2)	0.003647(5)	0(1)
27	0.00048(2)	0.001551(3)	0.137574(7)	0.09034(6)	0.002942(4)	0.00605(5)	0.000339(1)
28	0.00037(1)	0.001541(3)	0.17074(7)	0.149512(6)	0.001225(2)	0.004008(4)	0.01794(5)
29	0.000512(3)	5.64e−05(1)	0.036056(6)	0.074383(7)	0.000146(2)	0.001235(4)	0.005382(5)
30	0.029823(5)	0.005477(4)	0.036811(6)	0.040622(7)	7.26e−11(2)	1.81e−07(3)	6.73e−12(1)
31	0.001084(2)	8657.613(4)	26646.29(6)	28015.5(7)	0.000986(1)	14616.07(5)	323.6487(3)
32	3.696095(6)	2.668684(5)	5.976367(7)	0(1)	5.12e−08(3)	0.229356(4)	0(1)
33	70.92023(1)	1.01e+08(6)	2.86e+08(7)	241640.3(4)	80.66749(2)	463357.5(5)	112.9015(3)
34	7.31e−12(2)	11.96794(6)	33.90606(7)	0(1)	5.14e−07(4)	0.004928(5)	4.75e−07(3)
35	0.38608(5)	0.04282(1)	0.183888(3)	0.088996(2)	0.354223(4)	0.585937(7)	0.514455(6)
36	5.02e−13(2)	0.038982(6)	0.447906(7)	0(1)	3.06e−09(4)	0.000201(5)	1.13e−11(3)
37	20.96815(6)	4.234592(3)	6.51993(4)	71.14095(7)	2.88e−07(1)	15.82329(5)	3.11e−07(2)
38	0.213589(7)	0.065441(6)	0.000517(3)	0(1)	5.81e−09(2)	0.009396(4)	0.029416(5)
39	0.163571(6)	0.0018(5)	0.163707(7)	0(1)	1.45e−06(3)	0.000899(4)	0(1)
40	4.83e−05(2)	64.7332(7)	0.919062(4)	0.948491(5)	2.84e−05(1)	0.959454(6)	0.000355(3)
41	0.518085(1)	56033.37(5)	21001882(7)	93518.7(6)	84.12407(3)	2672.475(4)	36.14519(2)
42	4.37e−16(3)	0.001132(6)	0.000151(5)	0(1)	4.45e−10(4)	0.00154(7)	0(1)
Mean rank	3.047619	4.952381	5.642857	3.809524	2.97619	4.738095	2.261905
Final rank	3	6	7	4	2	5	1

Table 11 Comparison on mean performance among the lion and swarm algorithms while solving test suite 3

Functions	ABC	BFO	CS	FF	GSO	MFO	PSO	DA	**GWO**	**WOA**	**CrS**	Lion
1	37.04171(3)	1.450303(1)	37.04171(3)	42.76901(11)	31.31711(2)	38.73648(10)	46.58722(12)	37.04172(9)	37.04171(3)	37.04171(3)	37.04171(3)	37.04171(3)
2(−)	1.03163(1)	1.03163(1)	1.03163(1)	1.03163(1)	1.03163(1)	0.01974(12)	1.03163(1)	1.03163(1)	1.03163(1)	1.03163(1)	1.03162(11)	1.03163(1)
3(−)	3.86278(1)	3.86276(8)	3.86278(1)	3.86278(1)	3.86278(1)	2.52383(12)	3.86278(1)	3.86278(1)	3.86265(9)	3.84916(10)	3.83024(11)	3.86278(1)
4(−)	3.32235(4)	3.29816(9)	3.22663(10)	3.32237(1)	3.29853(8)	0.8703(12)	3.18651(11)	3.32237(1)	3.32235(4)	3.31646(7)	3.31902(6)	3.32237(1)
5	0.001404(7)	0.001254(6)	0.00116(5)	0.00862(9)	0.000511(2)	2.246918(12)	0.022553(11)	0.000638(4)	0.012425(10)	0.000554(3)	0.00775(8)	0.000491(1)
6	0.130425(6)	4020902(11)	0.190622(7)	7.36e−06(1)	0.077091(5)	14577547(12)	2.643526(10)	2.336095(9)	0.05503(2)	0.066371(3)	2.054727(8)	0.076858(4)
7	0.035808(5)	16257282(11)	0.146915(8)	8.38e−06(2)	0.004398(3)	2.74e+08(12)	2.680918(10)	0.019091(4)	0.097379(6)	0.136622(7)	2.370464(9)	9.71e−13(1)
8	0.084147(9)	0.128748(10)	4.71e−05(1)	0.047667(7)	0.221932(11)	29.9168(12)	0.057206(8)	0.008384(4)	0.00041(2)	0.002272(3)	0.023834(6)	0.011084(5)
9(−)	623.836(4)	361.991(11)	636.35(1)	549.859(7)	636.35(1)	347.457(12)	581.163(5)	459.92(10)	482.319(9)	566.18(6)	505.846(8)	632.325(3)
10	39267.54(9)	213949.9(11)	2424.769(6)	0.005321(3)	148.0449(5)	710222.8(12)	135126.7(10)	12.00965(4)	5.71e−29(1)	2558.401(7)	11343.84(8)	0.000315(2)
11	3.001582(7)	39.91814(11)	8.2(9)	0.005293(2)	0.354507(4)	84.36489(12)	10.69884(10)	0.071153(3)	3.9e−21(1)	0.914357(6)	6.764746(8)	0.600265(5)
12	0.775156(5)	360064.1(11)	2.113405(6)	60.23957(7)	0.020624(4)	1.74e+13(12)	147.3059(9)	74.45345(8)	1.02e−37(2)	4.71e−82(1)	267.4916(10)	2e−05(3)
13(−)	10.1494(2)	4.63952(8)	5.63956(6)	8.13223(4)	2.63047(11)	0.60234(12)	5.63938(7)	6.43258(5)	8.6584(3)	4.26976(9)	3.61141(10)	10.1532(1)
14(−)	10.3931(4)	6.32647(6)	3.81351(11)	10.4029(1)	6.0124(8)	0.75384(12)	7.62245(5)	6.0124(8)	10.4024(3)	5.6805(10)	6.18244(7)	10.4029(1)
15(−)	10.4994(4)	6.77013(7)	4.4467(11)	10.5364(1)	5.17565(10)	0.71706(12)	7.31983(6)	6.30939(8)	10.5346(3)	8.27992(5)	6.24568(9)	10.5364(1)
16	− 1.1e+10(6)	− 1166.99(8)	65535(10)	− 1.3e+17(2)	− 1.4e+13(4)	119.998(9)	65535(10)	65535(10)	− 1.4e+15(3)	− 3e+07(7)	− 2.5e+12(5)	− 3.4e+65(1)
17	0.082363(6)	6027.808(11)	4.924331(8)	7.95e−05(4)	6.79e−05(3)	13762.9(12)	167.0186(10)	4.13e−05(2)	0.050152(5)	0.296237(7)	36.41853(9)	3.34e−13(1)
Mean rank	4.882353	8.294118	6.117647	3.764706	4.882353	11.70588	8	5.352941	3.941176	5.588235	8	2.058824
Final rank	4	11	8	2	4	12	9	6	3	7	9	1

(−) symbol in the “Functions” column represents the values given in the respective row are negative

Table 12 Comparison on performance deviation among the lion algorithm and swarm algorithms while solving test suite 3

Functions	ABC	BFO	CS	FF	GSO	MFO	PSO	DA	**GWO**	**WOA**	**CrS**	Lion
1	0(1)	1.450197(9)	0(1)	5.228291(11)	12.8006(12)	3.78962(10)	0(1)	5.97e−06(8)	0(1)	0(1)	0(1)	0(1)
2	6.47e−16(5)	2.81e−07(10)	1.2e−11(6)	1.4e−10(7)	0(1)	0.510013(12)	0(1)	0(1)	3.7e−08(9)	1.35e−09(8)	2.01e−05(11)	0(1)
3	2.58e−12(4)	1.64e−05(9)	3.77e−10(6)	4.91e−10(7)	5.33e−12(5)	0(1)	4.94e−08(8)	3.04e−15(3)	0.000124(10)	0.02066(11)	0.071166(12)	0(1)
4	2.84e−05(4)	0.053331(9)	0.053521(10)	3.87e−09(2)	0.053311(8)	0.083108(11)	0.207357(12)	1.51e−08(3)	2.85e−05(5)	0.007888(7)	0.004054(6)	1.54e−14(1)
5	0.000341(5)	0.000402(6)	0.000695(9)	0.010951(12)	0.000146(3)	0(1)	3e-18(2)	0.00055(8)	0.01087(11)	0.000156(4)	0.006144(10)	0.00041(7)
6	0.114999(4)	2706886(11)	0.20614(7)	4.9e−06(1)	0.172371(6)	5727253(12)	2.893648(10)	1.975863(9)	0.05028(3)	0.042341(2)	1.448823(8)	0.17186(5)
7	0.032898(4)	11332755(11)	0.159762(8)	8.34e−06(2)	0.006016(3)	71208827(12)	2.088504(10)	0.036522(5)	0.096684(6)	0.108512(7)	1.839531(9)	1.16e−12(1)
8	0.034019(8)	0.10361(10)	4.82e−05(1)	0.029148(7)	0.257633(12)	0.25039(11)	0.053224(9)	0.007009(5)	0.00022(2)	0.001742(3)	0.012851(6)	0.004716(4)
9	9.362974(4)	33.25774(6)	1.44e−06(1)	47.15534(9)	0.000113(2)	98.82074(12)	56.41822(11)	48.79238(10)	39.20673(7)	39.49668(8)	31.70461(5)	5.290938(3)
10	9354.768(8)	71964.28(10)	1487.434(6)	0.003328(3)	183.1153(5)	192927.4(12)	168632.3(11)	11.73189(4)	6.6e−29(1)	3141.42(7)	10102.2(9)	0.00056(2)
11	1.314255(8)	12.85846(12)	4.086563(9)	0.001489(3)	0.392729(5)	0(1)	7.743804(11)	0.052806(4)	7.78e−21(2)	0.954365(7)	5.579822(10)	0.552014(6)
12	0.392249(5)	524209.4(11)	1.259758(6)	107.5737(9)	0.022389(4)	9.65e+12(12)	144.3612(10)	55.01026(7)	1.15e−37(2)	6.39e−82(1)	86.84834(8)	1.97e−05(3)
13	0.007321(3)	3.260141(7)	4.120368(11)	2.767328(6)	1.47e−11(2)	0.05171(4)	4.120537(12)	3.762467(10)	3.34049(8)	3.608648(9)	1.359653(5)	1.26e−15(1)
14	0.010179(5)	3.82687(9)	0.844308(6)	9.04e−07(3)	4.027054(11)	0(1)	3.837661(10)	4.027054(11)	0.000349(4)	2.826982(8)	2.359016(7)	1.26e−15(2)
15	0.079681(6)	3.616367(11)	1.054766(7)	9.48e−7(4)	1.01e−9(3)	0(1)	2.936063(10)	3.99727(12)	0.001879(5)	2.882769(9)	2.39853(8)	1.56e−11(2)
16	2.37e10(7)	2877.79(5)	1.023(2)	1.61e17(11)	3.15e13(9)	0(1)	1.18(4)	1.123(3)	3.06e15(10)	67030955(6)	5.55e12(8)	7.51e65(12)
17	0.066014(6)	2972.22(12)	3.404334(9)	2.9e−05(3)	6.99e−05(5)	0(1)	246.4505(11)	6.11e−05(4)	0.112138(7)	0.207344(8)	31.78674(10)	5.45e−13(2)
Mean rank	5.117647	9.294118	6.176471	5.882353	5.647059	6.764706	8.411765	6.294118	5.470588	6.235294	7.823529	3.176471
Final Rank	2	12	6	5	4	9	11	8	3	7	10	1

Table 13 Comparison on mean performance among the lion, evolutionary and human/physics inspired algorithms while solving test suite 3

Functions	BBO	DE	GA	GSA	HSA	SA	Lion
1	44.67812(6)	43.19215(5)	46.58722(7)	37.04171(1)	37.04171(1)	37.04171(1)	37.04171(1)
2	− 0.70516(4)	− 0.11492(6)	− 0.54081(5)	0.846826(7)	− 1.03163(1)	− 1.03162(3)	− 1.03163(1)
3(−)	3.08976(6)	3.24674(5)	3.84944(4)	2.52383(7)	3.86278(1)	3.86276(3)	3.86278(1)
4(−)	3.32237(1)	2.4188(6)	3.17102(5)	0.81062(7)	3.32237(1)	3.31997(4)	3.32237(1)
5	0.004694(5)	0.00344(3)	0.033284(7)	0.013067(6)	0.003767(4)	0.001976(2)	0.000491(1)
6	1.27e−05(2)	1257998(6)	3633494(7)	0.704618(4)	1.4e−06(1)	35336.62(5)	0.076858(3)
7	1.79e−05(3)	80264189(6)	86406416(7)	17684847(5)	3.69e−06(2)	573898.9(4)	9.71e−13(1)
8	0.00434(1)	11.90561(6)	0.692018(5)	30.18762(7)	0.023386(3)	0.613645(4)	0.011084(2)
9(−)	507.837(4)	324.954(6)	233.914(7)	493.164(5)	636.35(1)	599.35(3)	632.325(2)
10	7.401581(2)	181527.2(6)	479541.7(7)	1931.715(3)	2530.188(4)	146876.8(5)	0.000315(1)
11	0.048745(2)	63.34128(7)	58.41237(6)	7.16e−16(1)	0.865555(4)	15.73728(5)	0.600265(3)
12	0.034092(3)	2.37e+10(7)	2.13e+10(6)	81.17991(4)	0.011233(2)	6975775(5)	2e−05(1)
13(−)	2.63047(4)	0.8263(6)	2.61849(5)	0.57922(7)	2.65143(3)	9.02516(2)	10.1532(1)
14(−)	3.14926(4)	2.16381(6)	2.74895(5)	0.75384(7)	7.73148(3)	9.33552(2)	10.4029(1)
15(−)	5.17565(4)	1.29693(6)	5.12449(5)	0.71706(7)	5.7869(3)	9.44529(2)	10.5364(1)
16	−7.3e+22(2)	119.998(5)	− 119.041(4)	65535(7)	− 9.6e+20(3)	119.998(5)	− 3.4e+65(1)
17	0.000198(4)	7320.788(6)	0	2.16e−32(1)	1.49e−05(3)	453.5853(5)	3.34e−13(2)
Mean rank	3.352941	5.764706	5.411765	5.058824	2.352941	3.529412	1.411765
Final rank	3	7	6	5	2	4	1

(−) symbol in the “Functions” column represents the values given in the respective row are negative

Table 14 Comparison on performance deviation among the lion, evolutionary and human/physics inspired algorithms while solving test suite 3

Functions	BBO	DE	GA	GSA	HSA	SA	Lion
1	4.268882(7)	0.478047(6)	1.04e−06(5)	0(1)	0(1)	0(1)	0(1)
2	0.447032(6)	0.040077(5)	0.447938(7)	1.24e−16(2)	1.69e−09(3)	6.18e−06(4)	0(1)
3	4.97e−16(3)	0.047366(7)	0.016929(6)	0(1)	2.17e−07(4)	3.19e−05(5)	0(1)
4	5.02e−11(3)	0.189149(7)	0.095325(6)	1.24e−16(1)	1.94e−09(4)	0.003023(5)	1.54e−14(2)
5	0.008773(5)	0.001558(3)	0.029481(7)	0.008901(6)	0.00611(4)	0.000693(2)	0.00041(1)
6	7.61e−6(2)	435877.5(6)	2452607(7)	0.671285(4)	1.08e−6(1)	50463.19(5)	0.1718(3)
7	1.91e−05(3)	23872623(5)	62132994(7)	39544520(6)	2.45e−06(2)	757992.5(4)	1.16e−12(1)
8	0.00235(1)	2.18592(7)	0.40867(6)	0.32888(5)	0.01636(3)	0.31206(4)	0.0047(2)
9	29.08413(4)	38.54154(5)	87.66384(7)	53.84716(6)	4.11e−6(1)	16.55112(3)	5.2909(2)
10	5.922497(2)	41581.53(5)	233465.7(7)	1549.069(3)	3576.703(4)	69957.13(6)	0.0005(1)
11	0.03361(2)	0.16820(3)	12.1076(7)	7.51e−16(1)	1.20058(5)	8.11455(6)	0.5520(4)
12	0.020556(3)	2.76e+10(7)	2.27e+10(6)	62.05451(4)	0.006559(2)	7170694(5)	1.97e−05(1)
13	1.49e−14(3)	0.052687(6)	0.007976(4)	0(1)	0.028694(5)	2.234464(7)	1.26e−15(2)
14	0.524939(5)	0.352888(4)	0.010214(3)	0(1)	3.658042(7)	2.378228(6)	1.26e−15(2)
15	9.15e−13(2)	0.01411(4)	0.031521(5)	0(1)	2.836387(7)	2.38718(6)	1.56e−11(3)
16	1.62e+23(6)	2.02e−08(2)	534.5072(4)	0(1)	1.96e21(5)	5.65e−07(3)	7.51e65(7)
17	0.000168(4)	795.8149(0)	4536.278(6)	1.73e−32(1)	9.22e−6(3)	283.1926(5)	5.45e−13(2)
Mean rank	3.588235	4.823529	5.882353	2.647059	3.588235	4.529412	2.117647
Final rank	3	6	7	2	3	5	1

Table 15 Comparison on mean performance and deviation among the lion and swarm algorithms while solving test suite 4

Engineering function	Statistics	ABC	BFO	CS	FF	GSO	MFO	PSO	DA	GWO	WOA	CrS	Lion
Welded beam	Mean	2.0307(6)	2.1895(7)	1.7534(3)	1.7459(2)	2.9437(8)	14.2957(11)	16.7133(12)	1.878(4)	3.3901(9)	4.6582(10)	1.8849(5)	1.7301(1)
	Std(e−2)	15.5411(5)	23.3597(7)	6.9867(2)	2.0315(1)	19.9593(6)	506.9932(12)	431.0398(11)	12.6503(4)	58.0295(8)	345.7692(10)	9.4464(3)	173.0384(9)
pressure vessel	Mean(e2)	91.9216(3)	168.5795(11)	100.3754(5)	101.2403(7)	91.6478(2)	265.6904(12)	100(4)	135.3904(9)	100.3756(6)	166.942(10)	101.4575(8)	86.8613(1)
	Std(e2)	14.7953(6)	42.6545(9)	0(1)	1.1681(4)	15.5443(7)	97.5748(12)	0(1)	56.0003(10)	0.0001(3)	57.6147(11)	1.2647(5)	24.0641(8)
Gear train	Mean	7.08e−12(11)	2.14e−15(8)	2.81e−15(9)	1.03e−16(6)	3.52e−25(4)	8.31e−07(12)	1.23e−15(7)	2.67e−24(5)	3.33e−12(10)	0(1)	1.28e−34(3)	0(1)
	Std	6e−12(12)	5e−15(10)	6e−15(11)	1e−16(7)	4e−25(5)	0(1)	2e−15(8)	5e−24(6)	3e−15(9)	0(1)	3e−34(4)	0(1)
Tension/compression spring	Mean	0.0107(1)	0.0178(6)	1.6763(8)	5.0061(9)	0.0151(5)	8.3488(11)	10.0003(12)	0.0135(4)	1.6753(7)	0.0118(3)	5.0088(10)	0.0111(2)
	Std	0.0001(1)	0.0018(5)	4.0789(9)	5.4705(11)	0.002(6)	4.062(8)	0.0001(1)	0.0029(7)	4.0794(10)	0.0013(3)	5.4741(12)	0.0017(4)
Three bar truss	Mean(e2)	2.6395(8)	2.055(1)	2.6389(2)	2.6389(2)	2.6395(8)	2.7662(12)	2.6389(2)	2.6853(11)	2.639(6)	2.6423(10)	2.6391(7)	2.6389(2)
	Std	0.0193(8)	101.1444(12)	4.02e−14(2)	0.0014(5)	0.0734(9)	0(1)	0.0002(4)	7.0321(11)	0.0113(6)	0.5322(10)	0.0154(7)	3.4e−08(3)
Mean rank		6.1	7.6	5.2	5.4	6	9.2	6.2	7.1	7.4	6.9	6.4	3.2
Final rank		5	11	2	3	4	12	6	9	10	8	7	1

Funding None.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Appendix

Definition 1 An X^{cub} of an X^{male} and an X^{female} is the sum of Hadamard product of crossover mask and X^{male} and Hadamard product of complement of the same crossover mask and X^{female}, provided the crossover mask is essentially to be a binary vector with $C_r \cdot L$ number of binary ones.

Assumption 1 Within a pride, male lions are always stronger than female lions, applicable to cubs also.

Lemma 1 *If the product of* C_r and L *is equal to zero, then* X^{cubs} *are equal to* X^{female}.

Proof of Lemma 1 According to Definition 1, B, which is a vector with L elements, has $C_r \cdot L$number of ones and$L(1 - C_r)$ number of zeros. If $C_r \cdot L = 0$, then B has no ones and L number of zeros, which means B is a vector of zeros. By applying this B in the mathematical representation of crossover operation given in [11], the first term of RHS becomes B and the second term of RHS becomes X^{female} as $\bar{B} = 1 - B$. Hence, X^{cubs} becomes X^{female} when $C_r \cdot L = 0$.

Lemma 2 *If the product of* C_r and L *is equal to one, then* X^{cubs} *are equal to* X^{male}.

Proof of Lemma 2 Similar to the proof of Lemma 1, B is a vector of ones, when $C_r \cdot L = L$ and $\bar{B}$ becomes a vector of zeros. This in turn, makes the first RHS term of Eq. (11) as X^{male} and the second RHS term as a vector of zeros. Hence, X^{cubs} become X^{male} when $C_r \cdot L = L$.

Lemma 3 E_1^{nomad} *is greater than* E_2^{nomad}, *if* d_1 *and* $f\left(X_1^{nomad}\right)$ *are greater than* d_2 *and* $f\left(X_2^{nomad}\right)$, *respectively.*

Proof of Lemma 3 According to the lemma,

$$\max\left(d_1, d_2\right) = d_1 \tag{16}$$

$$\max\left(f\left(X_1^{nomad}\right), f\left(X_2^{nomad}\right)\right) = f\left(X_1^{nomad}\right) \tag{17}$$

Hence, Eqs. (24) and (25) (from Theorem 2) takes the form,

Table 16 Comparison on mean and standard deviation performance among the lion, evolutionary and human/physics inspired algorithms while solving test suite 4

Engineering functions	Statistics	BBO	DE	GA	GSA	HAS	SA	Lion
Welded beam design	Mean	13.6534(5)	3.6578(3)	9.2342(4)	16.0827(7)	14.0993(6)	2.1308(2)	1.7301(1)
	Std	0.0361(1)	0.8252(4)	7.2407(6)	8.1954(7)	0.517(3)	0.2015(2)	1.7303(5)
Pressure vessel design	Mean(e2)	100(3)	497.2405(6)	1776.418(7)	134.6275(5)	100(3)	81.6229(1)	86.8612(2)
	Std(e2)	0(1)	14.6343(3)	250.2908(7)	56.6359(6)	0(1)	17.0338(4)	24.0641(5)
Gear train design	Mean	5.58e−26(3)	3.52e−12(5)	1.35e−10(7)	0(1)	2.42e−15(4)	8.88e−11(6)	0(1)
	Std	1.23e−25(3)	5e−12(5)	2.09e−10(7)	0(1)	3.78e−15(4)	1.19e−10(6)	0(1)
Compression/tension spring	Mean	10.003(4)	10.026(7)	0.01425(3)	10.002(6)	10.003(4)	0.01088(1)	0.01115(2)
	Std	0(1)	0.00037(4)	0.0040(7)	0(1)	0(1)	0.00047(5)	0.00173(6)
Three-bar truss	Mean(e2)	2.641(3)	2.7018(6)	2.6441(5)	2.7662(7)	2.6425(4)	2.6401(2)	2.6389(1)
	Std	0.1500(4)	0.3154(6)	0.4499(7)	0(1)	0.2541(5)	0.0556(3)	3.4e−8(2)
Mean rank		2.8	4.9	6	4.2	3.5	3.2	2.6
Final rank		2	6	7	5	4	3	1

Table 17 Optimized design parameters and constraints of the engineering problems obtained from lion algorithm

Design parameters/ constraint functions	Welded beam design	Pressure vessel design	Gear train design	Tension/compression spring design	Three-bar truss design
a_1	−2.155342	0.7908	−43.3016	0.05	0.78867
a_2	$-1.43504e^{2}$	0.3909	−12.7057	0.37443	0.4082481
a_3	$-3.50736e^{-4}$	40.9752	−14.4569	8.5467673	n/a
a_4	−3.42922	191.0709	−29.4012	n/a	n/a
a_5	−0.080392	n/a	n/a	n/a	n/a
a_6	−0.235643	n/a	n/a	n/a	n/a
h_1	−0.20539	$-1.6397422e^{-5}$	n/a	$-6.12125e^{-6}$	$-3.03024e^{-12}$
h_2	−3.470225	$-2.513248e^{-6}$	n/a	$-5.4141832e^{-6}$	−1.46410
h_3	−9.058026	−3.56100	n/a	−4.86066	−0.53589
h_4	−0.2057427	n/a	n/a	−0.7170461	n/a
f	1.728084	$5.90745e^{3}$	0	0.0098725	263.895

$$E_1^{nomad} = \exp(1) = 2.71 \tag{18}$$

$$E_2^{nomad} = \exp\left(\frac{d_2}{d_1}\right)\frac{f\left(X_1^{nomad}\right)}{f\left(X_2^{nomad}\right)}. \tag{19}$$

E_2^{nomad} can be greater than $\exp(1)$only if $f\left(X_1^{nomad}\right) >> f\left(X_2^{nomad}\right)$ as the first term produces exponential decay from $\exp(1)$, when $d_1 > d_2$. But, if $f\left(X_1^{nomad}\right) >> f\left(X_2^{nomad}\right)$, then probably $d_1 >> d_2$due to its linear relationship with $f\left(X_1^{nomad}\right)$, which leads $\exp(\bullet)$ towards zero, and hence E_2^{nomad} becomes lesser than $\exp(1)$. Thus it is proved that $E_1^{nomad} > E_2^{nomad}$, when $d_1 > d_2$ and$f\left(X_1^{nomad}\right) > f\left(X_2^{nomad}\right)$.

Lemma 4 *E_1^{nomad} is greater than E_2^{nomad}, if d_1 and $f\left(X_2^{nomad}\right)$ are greater than d_2 and $f\left(X_1^{nomad}\right)$, respectively.*

Proof of Lemma 4 According to the lemma,

$$\max\left(d_1, d_2\right) = d_1 \tag{20}$$

$$\max\left(f\left(X_1^{nomad}\right), f\left(X_2^{nomad}\right)\right) = f\left(X_2^{nomad}\right) \tag{21}$$

As $\frac{f\left(X_2^{nomad}\right)}{f\left(X_1^{nomad}\right)} > 1$ from Eq. (21)

$$E_1^{nomad} = \exp(1)\frac{f\left(X_2^{nomad}\right)}{f\left(X_1^{nomad}\right)} > \exp(1) \tag{22}$$

$$E_2^{nomad} = \exp\left(\frac{d_2}{d_1}\right) < \exp(1) \tag{23}$$

Hence, it is proved that $E_1^{nomad} > E_2^{nomad}$, when $d_1 > d_2$ and $f\left(X_2^{nomad}\right) > f\left(X_1^{nomad}\right)$.

Lemma 5 *E_2^{nomad} is greater than E_1^{nomad}, if d_2 and $f\left(X_1^{nomad}\right)$ are greater than d_1 and $f\left(X_2^{nomad}\right)$, respectively.*

Lemma 6 *E_2^{nomad} is greater than E_1^{nomad}, if d_2 and $f\left(X_2^{nomad}\right)$ are greater than d_1 and $f\left(X_1^{nomad}\right)$, respectively.*

Proof of Lemma 5 and 6 Lemma 5 is the vice versa of lemma 3 as $E_2^{nomad} > \exp(1)$ and $E_1^{nomad} < \exp(1)$. Lemma 6 is the vice versa of lemma 4 as $E_2^{nomad} = \exp(1)$ and probably $E_1^{nomad} < \exp(1)$ proved through Axiom 3.

Axiom 1 $C_r \cdot L = L$ is an integer possibly between 1 and $L - 1$, i.e.,$C_r \cdot L \in (1, L - 1)$.

Axiom 2 X_1^{nomad} and X_2^{nomad} are essentially different and hence d_1 is not always equal to d_2.

Axiom 3 $f\left(X_1^{nomad}\right)$ and $f\left(X_2^{nomad}\right)$ exhibit linear variation with respect to d_1 and d_2, respectively.

Theorem 1 *X^{cubs} can be a subset of both X^{male} and X^{female} only if C_r is selected in such a way that $C_r \cdot L = L$.*

Proof of Theorem 1 It is known that B has $C_r \cdot L$ number of ones in arbitrary vector positions and zeros in the remaining positions. Hence, $X^{male} \circ B$ have elements of X^{male} and zeros from the positions where B has ones and zeros, respectively. In contrast, $X^{female} \circ \bar{B}$ has elements of X^{female} and zeros from the positions where B has zeros and ones, respectively, as $\bar{B}$ is the one's complement of B. Hence, it can be said that $X^{male} \circ B \subset X^{male}$ and $X^{female} \circ \bar{B} \subset X^{female}$. As '+' operator in the mathematical representation of crossover operation given in [11] is equivalent to set union operation, the resultant X^{cubs} are subset of both and only X^{male} and X^{female}, i.e., $X^{cubs} \subset X^{male}, X^{female}$.

Theorem 2 *In a nomad coalition of only two lions, the evaluation score E_1^{nomad} will be always greater than E_2^{nomad} when E_1^{nomad} is greater than or equal to exponential function of unity and vice versa.*

Proof of Theorem 2 Let E_1^{nomad} and E_2^{nomad} be the evaluation scores of X_1^{nomad} and X_2^{nomad}, respectively. The evaluation scores can be calculated as

$$E_1^{nomad} = \exp\left(\frac{d_1}{\max\left(d_1, d_2\right)}\right) \frac{\max\left(f\left(X_1^{nomad}\right), f\left(X_2^{nomad}\right)\right)}{f\left(X_1^{nomad}\right)} \tag{24}$$

$$E_2^{nomad} = \exp\left(\frac{d_2}{\max\left(d_1, d_2\right)}\right) \frac{\max\left(f\left(X_1^{nomad}\right), f\left(X_2^{nomad}\right)\right)}{f\left(X_2^{nomad}\right)} \tag{25}$$

where d_1 is the Euclidean distance between X_1^{nomad} and X^{male}, d_2 is the Euclidean distance between X_2^{nomad} and X^{male}.

From lemmas 3 and 4, it can be said that if $E_1^{nomad} = \exp(1)$ (according to lemma 3) and $E_1^{nomad} > \exp(1)$ (according to lemma 4), then $E_2^{nomad} < \exp(1)$. Similarly, lemmas 5 and 6 asserts $E_2^{nomad} > E_1^{nomad}$, if $E_2^{nomad} \geq \exp(1)$. Hence, the theorem states that it is not necessary to calculate both E_1^{nomad} and E_2^{nomad} to evaluate X_1^{nomad} and X_2^{nomad}, respectively. It is sufficient to calculate either of them and can be concluded by comparing it with $\exp(1)$.

References

1. Rozenberg G, Bck T, Kok JN (2011) Handbook of natural computing, 1st edn. Springer Publishing Company, New York
2. Yang XS, Deb S, Fong S, He X, Zhao YX (2016) From swarm intelligence to metaheuristics: nature-inspired optimization algorithms. Computer 49(9):52–59
3. Shadbolt N (2004) Nature-inspired computing. IEEE J Intell Syst 19(1):2–3
4. Neumann F, Witt C (2010) Bioinspired computation in combinatorial optimization algorithms and their computational complexity. Natural computing series, XII, p 216
5. Corne D, Deb K, Knowles J, Yao X (2010) Selected applications of natural computing. In: Rozenberg G, Back T, Kok JN (eds) Handbook of natural computing. Springer, Berlin
6. Holland JH (1975) Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor
7. Bongard J (2009) Biologically inspired computing. IEEE Comput J 42(4):95–98
8. Forbes N (2000) Biologically inspired computing. Comput Sci Eng 2(6):83–87
9. Mahdiani HR, Ahmadi A, Fakhraie SM, Lucas C (2010) Bio-inspired imprecise computational blocks for efficient VLSI implementation of soft-computing applications. IEEE Trans Circuits Syst I Regul Pap 57(4):1549–8328
10. Rajakumar BR (2012) The lion's algorithm: a new nature-inspired search algorithm. In: Second international conference on communication, computing and security, vol 6, pp 126–135. https://doi.org/10.1016/j.protcy.2012.10.016
11. Rajakumar BR (2014) Lion algorithm for standard and large scale bilinear system identification: a global optimization based on lion's social behavior. In: 2014 IEEE congress on evolutionary computation (CEC), pp 2116–2123
12. Chander S, Vijaya P, Dhyani P (2016) ADOFL: multi-kernel-based adaptive directive operative fractional lion optimisation algorithm for data clustering. J Intell Syst 27:317
13. Babers R, Hassanien AE, Ghali NI (2015) A nature-inspired metaheuristic lion optimization algorithm for community detection. In: 2015 11th IEEE international computer engineering conference (ICENCO)
14. Chander S, Vijaya P, Dhyani P (2017) Multi kernel and dynamic fractional lion optimization algorithm for data clustering. Alex Eng J 57:267

15. Bauer H, Iongh de HH, Silvestre I (2003) Lion social behaviour in the West and Central African Savanna belt. Mamm Biol 68(1):239–243
16. Fogel LJ, Owens AJ, Walsh MJ (1966) Artificial intelligence through simulated evolution. Wiley Publishing, New York
17. Doerr B, Happ E, Klein C (2012) Crossover can probably be useful in evolutionary computation. Theor Comput Sci 425:17–33
18. Back T, Hoffmeister F, Schwefel HP (1993) An overview of evolutionary algorithms for parameter optimization. J Evol Comput 1(1):1–24 **(De Jong K (ed), Cambridge: MIT Press)**
19. Jong De KA (1975) An analysis of the behavior of a class of genetic adaptive systems. Doctoral thesis, Dept. Computer and Communication Sciences, University of Michigan, Ann Arbor
20. Packer C, Pusey AE (2016) Divided we fall: cooperation among lions. Sci Am 276:52–59
21. Packer C, Pusey AE (1982) Cooperation and competition within coalitions of male lions: Kin selection or game theory? Nature 296(5859):740–742
22. Grinnell J, Packer C, Pusey AE (1995) Cooperation in male lions: Kinship, reciprocity or mutualism? Anim Behav 49(1):95–105
23. Packer C, Pusey AE (1982) Cooperation and competition within coalition of male lions: Kin selection or game theory. Macmillan J 296(5859):740–742
24. Lotfi E, Akbarzadeh-T MR (2016) A winner-take-all approach to emotional neural networks with universal approximation property. Inform Sci 346:369–388
25. He S, Wu QH, Saunders JR (2009) Group search optimizer: an optimization algorithm inspired by animal searching behavior. IEEE Trans Evol Comput 13(5):973–990
26. Karaboga D, Akay B (2009) A comparative study of artificial bee colony algorithm. Appl Math Comput 214(1):108–132
27. Yao X, Liu Y, Lin G (1999) Evolutionary programming made faster. IEEE Trans Evol Comput 3(2):82–102
28. Fogel DB (1995) Evolutionary computation: toward a new philosophy of machine intelligence. IEEE Press, New York
29. Fogel LJ, Owens AJ, Walsh MJ (1965) Artificial intelligence through a simulation of evolution. In: Proc. 2nd cybern. sci. symp. biophysics cybern. syst., Washington: Spartan Books, pp 131–155
30. Schwefel HP (1995) Evolution and optimum seeking. Wiley, New York
31. Yao X, Liu Y (1997) Fast evolution strategies. Control Cybern 26(3):467–496
32. Hansen N, Ostermeier A (1996) Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In: IEEE int. conf. evolution. comput. (ICEC) proc, pp 312–317
33. Hansen N (2006) The CMA evolution strategy: a comparing review. In: Lozano JA, Larraaga P, Inza I, Bengoetxea E (eds) Towards a new evolutionary computation. Springer, Berlin
34. Hedar A, Fukushima M (2006) Evolution strategies learned with automatic termination criteria. In: Proceedings of SCIS-ISIS 2006, Tokyo, Japan
35. Goldberg DE (1989) Genetic algorithms in search optimization and machine learning. Addison Wesley, Reading, p 41
36. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, IV, pp 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
37. Karaboga D (2005) An idea based on honey bee swarm for numerical optimization. Technical Report-TR06, Erciyes University, Engineering Faculty, Computer Engineering Department
38. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
39. Cheng YP, Li Y, Wang G, Zheng Y-F, Cui XT (2017) A novel bacterial foraging optimization algorithm for feature selection. Expert Syst Appl 83:1–17
40. Yang X-S, Deb S (2009) Cuckoo search via Lévy flights. In: World congress on nature and biologically inspired computing (NaBIC 2009), IEEE Publications, pp 210–214
41. Yang X-S (2010) Nature inspired metaheuristic algorithms, 2nd edn. Luniver Press, London
42. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowl Based Syst 89:228–249
43. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. J Neural Comput Appl 27(4):1053–1073
44. Mirjalili SM, Mirjalili A, Lewis (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
45. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
46. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. Comput Struct 169:1–12
47. Li LL, Yang YF, Wang C-H, Lin K-P (2018) Biogeography-based optimization based on population competition strategy for solving the substation location problem. Expert Syst Appl 97:290–302
48. Price VK, Storn MR (1997) Differential evolution: a simple evolution strategy for fast optimization. Dr Dobb's J 22:18–24
49. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
50. Geem ZW, Kim JH, Loganathan GV (2001) A new heuristic optimization algorithm: harmony search. Simulation 76:60–68
51. Ingber L (1993) Simulated annealing: practice versus theory. Math Comput Model 18(11):29–57