Memories - Materials, Devices, Circuits and Systems 4 (2023) 100021

Contents lists available at ScienceDirect

Memories - Materials, Devices, Circuits and Systems

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

Eagle perching optimizer for the online solution of constrained optimization

Ameer Tamoor Khan [a], Shuai Li [b,*], Yinyan Zhang [c], Predrag S. Stanimirovic [d]

[a] Department of Computing, The Hong Kong Polytechnic University, Hong Kong
[b] Department of Electronic and Electrical Engineering, Swansea University, UK
[c] College of Cyber Security, Jinan University, Guangzhou, China
[d] Department of Computer, Faculty of Sciences and Mathematics, University of Niš, Niš, Serbia

ARTICLE INFO

Keywords:
Optimization
Benchmark
Particle swarm optimization
Swarm algorithm
Constrained optimization
Stochastic algorithm
Heuristic algorithm

ABSTRACT

The paper proposes a novel nature-inspired optimization technique called Eagle Perching Optimizer (EPO). It is an addition to the family of swarm-based meta-heuristic algorithms. It mimics eagles' perching nature to find prey (food). The EPO is based on the exploration and exploitation of an eagle when it descends from the height such that it formulates its trajectory in a way to get to the optimal solution (prey). The algorithm takes bigger chunks of search space and looks for the optimal solution. The optimal solution in that chunk becomes the search space for the next iteration, and this process is continuous until EPO converges to the optimal global solution. We performed the theoretical analysis of EPO, which shows that it converges to the optimal solution. The simulation includes three sets of problems, i.e., uni-model, multi-model, and constrained real-world problems. We employed EPO on the benchmark problems and compared the results with state-of-the-art meta-heuristic algorithms. For the real-world problems, we used a cantilever beam, three-bar truss, and gear train problems to test the robustness of EPO and later made the comparison. The comparison shows that EPO is comparable with other known meta-heuristic algorithms.

1. Introduction

There are a variety of social behaviors in nature that are used to accomplish a task. Creatures collaborate and interact in groups, herds, schools, colonies, and flocks for various reasons, including hunting, protecting, navigating, and foraging. At the same time, the ultimate goal of all individual and collective activities is survival. In particular, wolf packs have one of the most sophisticated social structures coordinating with others to accomplish a common goal during hunting. Generally speaking, wolves have a social direction during the many stages of hunting, including pursuing prey, pestering prey, attacking prey, and circling prey are all examples of this [1]. On the other hand, living things like Beetle [2–5] live on their own and search for food based on their smelling nature.

It is fascinating to see how different species figure out how to get along best and work together to get the job done. The evolution of such perfect and efficient habits over thousands of years is evident. In light of this, we would draw motivation from them to find answers to our issues. With this in mind, researchers Beni and Wang in 1989 [6]proposed a new field of study called swarm intelligence (SI). Collective and social intelligence in a group of living organisms in nature [7] is what SI attempts to emulate artificially. Social intelligence researchers seek to understand the contextual norms governing the interactions between people that produce this intelligence. Without a leader to direct them, the population can be "simulated" by discovering the basic principles that govern interactions among a subset of individuals.

For the past few years, meta-heuristic algorithms have been vital in determining optimal solutions to engineering optimization problems. They are based on a stochastic approach, different from the deterministic approach. The deterministic approach provides the exact solution repeatedly for the same initial conditions. In contrast, in the stochastic process, the algorithms search randomly over the search space and every time end up at a different solution, close to optimal [8]. Deterministic algorithms work efficiently in the case of uni-modal problems, which only have one global solution. However, there is an issue with multi-modal problems, i.e., with several local minima. It will create a local minima entrapment, making it problematic for the deterministic algorithms to search for the global optimum. It is known as local stagnation, in which an algorithm fails to find the global optimal point and gets stuck in a local solution. Real-world problems are mainly like this, containing several local optima.

Stochastic (meta-heuristic) algorithms are based on stochastic operators that ensure randomness [9]. Local stagnation or entrapment is avoided because of the random nature and will result in a different solution on each run despite the same initial conditions. These evolutionary algorithms initially made an initial guess by generating random

* Corresponding author.
E-mail addresses: ameer.khan@connect.polyu.hk (A.T. Khan), Shuaili@ieee.org (S. Li), yyzhang@jnu.edu.cn (Y. Zhang), peckois@ptt.rs (P.S. Stanimirovic).

https://doi.org/10.1016/j.memori.2022.100021
Received 27 July 2022; Received in revised form 16 November 2022; Accepted 12 December 2022

candidate solutions. The solutions are iteratively improved until the final condition meets, finding a globally optimal solution. The evolutionary algorithms bring many advantages: simplicity, derivation independency, problem independency, and local optima avoidance [10].

Evolutionary algorithms follow a specific pattern in nature that involves randomness. They treat an optimization problem as a black box and are only concerned with inputs and outputs. The primary process of the optimization remains the same [11]. To achieve the goal, they improve those candidate solutions iteratively until it reaches the optimal output.

The avoidance of local stagnation is another advantage of the evolutionary algorithm. Although there is no theoretical guarantee to avoid it altogether, its random nature suggests that the chances of such occurrence are narrow. The exact and accurate approximation of the global optimum solution is also not guaranteed. Running the algorithm several times and then taking an average out will improve the results [12]. Finally, the simplicity of evolutionary algorithms makes them appealing since they are based on some mechanisms presented in nature. Understanding those natural phenomena helps us formulate algorithms we can employ to solve real-world problems and achieve the desired solutions. They follow the same rules of the defined framework and treat every problem equally under the specified norms.

There are three optimization algorithms; basic, genetic, and swarm optimization. Basic algorithms generally involve the derivative and differential approaches to solve the optimization problem. Genetic algorithms are inspired by Charles Darwin's theory of evolution and are based on a natural selection process that mimics biological evolution. Swarm optimization is based on those species that work as a group in nature and provides each other assistance while searching for food or avoiding predator; Particle Swarm Optimization (PSO) [13], Ant-Lion Optimizer (ALO) [14], and Dragon Optimization (DA) [15] are some of those. There are some other popular algorithms which includes; Grey Wolf Optimizer (GWO) [16], Cuckoo Optimization Algorithm (COA) [17], Beetle Antennae Search [2,3,18–28], Magnetic Charged System Search [29], Cuckoo Search (CS) algorithm [30], Gravitational Search Algorithm (GSA) [31], Democratic Particle Swarm Optimization (DPSO) [32], and Chaotic Swarming Of Particles (CSP) [33].

In this paper, we propose a new algorithm called eagle perching optimizer (EPO), inspired by eagles and how they are wired by nature. It is another meta-heuristic algorithm that is iteratively for the optimal solution. The main contributions of this paper are listed as follows:

1. The inspiration, mathematical formulation of the algorithm, global convergence proof, and comparison between the designed algorithms.
2. Results and discussion based on the comparison between EPO and other meta-heuristic algorithms.
3. Analyze the algorithm by varying its different controlling variables and will discuss the results.
4. Engineering problems will be optimized using EPO algorithms, and then we will compare the results with the rest of the meta-heuristic algorithm.

The rest of this paper is organized as follows. Section 2 presents the inspiration behind the algorithm, its mathematical formulation, EPO algorithm, and its modified version, a comparison between the two, and mathematical proof for global convergence. Section 3 will discuss the benchmark testing results and performance comparisons. Section 4 will discuss the constrained problems, in which we will optimize some real-world engineering optimization problems. Section 5 will conclude the paper with final remarks.

2. Eagle perching optimizer

In this section, we will first discuss the inspiration behind the EPO algorithm. The algorithm and mathematical formulation are then discussed.

2.1. Inspiration

The term "eagle" is typically applied to a variety of huge predatory birds in the Accipitridae family. Their typical length is between 30 and 31 inches, while their wingspan is between 6 and 7 feet [34]. They normally live in the heavens, and even when it is time to reproduce, the male and female engage in a special rite of wooing. They soar upward to a great height. There, they lock their clays together, execute aerobic exercises as they fall, and shatter just before they hit the ground. Their life cycle primarily consists of five cycles: hatching, fledgling, juvenile stage, and adulthood [35]. Females often hatch 2 to 4 eggs.

They come under the category of predators. They typically eat fish, other aquatic life, and tiny animals for food. Their method of hunting is distinctive; they soar into the air to perhaps a great height before focusing on their prey [36]. Once it is located, they swoop down and seize the prey. They live higher up, as was previously indicated, typically on tall trees, cliffs, and mountains. They just descend from a high vantage point, look down at the ground, sample a few places, then identify the most elevated position among those samples to determine where to travel next using a natural process. As they get closer, they reassess a few areas and further solidify their opinion of the highest place. They carry out this work iteratively and fine-tune their search for the best location to live. The characteristics of their built-in perching algorithm are depicted in Fig. 1.

We will exploit this nature and will employ it in optimization to obtain optimal solutions. In our algorithm, we will make a folk of eagles search for the optimal elevation to reside. They all will look for the best solution individually. The algorithm will then pick the best solution out of all the eagles and compare it with the previously stored best solution. This process will run iteratively until the algorithm reaches its optimal solution, after which no further improvement is obtained.

In the next section, we will discuss this algorithm's mathematical representation that how we exploited nature and brings it into the mathematical formulation.

2.2. Mathematical formulation of EPO

The EPO algorithm mimics the eagle perching behavior. Like eagle this algorithm also finds the highest point of the solution i.e. optimal solution. In optimization, there is a unique relation between the minima and maxima of a function i.e., for function $f, min(f) = max(-f)$. The nature defines the working algorithm for all its inhabitants. Eagle has a very simple but unique way to explore its terrain. Flying high up in skies it looks around by sampling few points and move towards the highest point, reaching there it again glance around and repeating the same process, this recurrence allows an eagle to reach the highest point. At first that the eagle views the whole terrain from the skies, which is exploration, after repeating the same exercise for several times it reaches near the ground, which is exploitation. The transformation from exploration to exploitation is the key for stochastic optimization (meta-heuristic) algorithms. This is mathematically formulated in EPS algorithm as follows:

$$l_{scale} = l_{scale} * eta, \tag{1}$$

where l_{scale} is the scaling variable that will decrease recurrently and will move from exploration to exploitation, eta is a shrinking constant $0 <$ eta < 1. To achieve the optimality faster we will employ a group of i.e. eagles. They will look the search space cooperatively to make the task easier,

$$X = \begin{bmatrix} X_{1,1} & X_{1,2} & X_{1,3} & \dots & X_{1,m} \\ X_{2,1} & X_{2,2} & X_{2,3} & \dots & X_{2,m} \\ X_{3,1} & X_{3,2} & X_{3,3} & \dots & X_{3,m} \\ X_{4,1} & X_{4,2} & X_{4,3} & \dots & X_{4,m} \\ \vdots & \vdots & \vdots & & \vdots \\ X_{n,1} & X_{n,2} & X_{n,3} & \dots & X_{n,m} \end{bmatrix}, \tag{2}$$

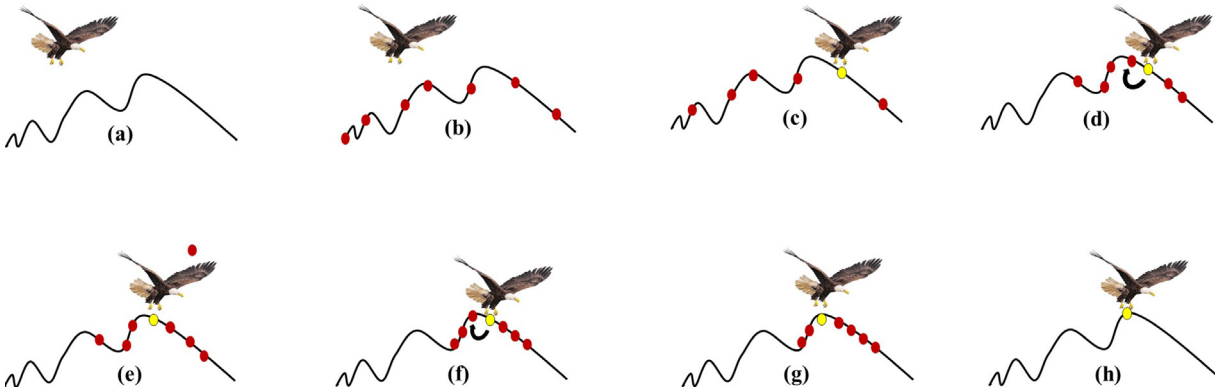

Fig. 1. It demonstrates the perching nature of the eagle: in (a), the eagle is over its search space; in (b), the eagle samples the search space and looks for the sample at the highest point; in (c), the eagle has reached over the sampled point; in (d), the eagle sampled the search space again, but since the space is now small, it will again look for the sample at the highest point; from (e) to (h).

where n represents the number of particles that we are employing in the search space and m are the number of dimensions of the search space.

To understand the strolling of particles (eagles) in search space, consider a particle at position x, to roam freely and randomly in all possible direction. A Δx (which is a random value) is added to its current position, i.e., $x + \Delta x$ at each iteration. As a result, we have,

$$X = X + \Delta X, \tag{3}$$

where,

$$\Delta X = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & \dots & R_{1,m} \\ R_{2,1} & R_{2,2} & R_{2,3} & \dots & R_{2,m} \\ R_{3,1} & R_{3,2} & R_{3,3} & \dots & R_{3,m} \\ R_{4,1} & R_{4,2} & R_{4,3} & \dots & R_{4,m} \\ \vdots & \vdots & \vdots & & \vdots \\ R_{n,1} & R_{n,2} & R_{n,3} & \dots & R_{n,m} \end{bmatrix}, \tag{4}$$

with $R \in (-1, 1)$ denoting the random values. For each element of X, we have,

$$X_{ij} = X_{ij} + l_{scale} \times \Delta X_{ij}, \tag{5}$$

where, i represents the i_{th} particle and j represents the j_{th} dimension of the corresponding position. Since the folk of eagles are high-up in air with some samples looking for the highest place on ground. They evaluate their respective set of samples and find out the highest place. We will replicate that result in our algorithm and will pass the X_{ij} to the function we want to minimize,

$$Y_{ij} = f(X_{ij}). \tag{6}$$

It shows that each particle position gets evaluated. Our goal is to minimize the given function so as to find the best solution out of all the eagle positions, which is denoted by Y_{min}. We will define two more variable Y_{Best} and X_{Best}. The evolution of Y_{Best} and X_{Best} is set as follows,

$$\text{if} : Y_{min} < Y_{Best} \tag{7}$$

$$Y_{Best} = f(X_{ij}) \tag{8}$$

$$X_{Best} = X_{ij}, \tag{9}$$

The recursion of this algorithm will ultimately find the optimal solution of a given function.

2.3. EPO algorithm

With the above explained mathematical formulation of the EPO, we are now able to discuss its algorithmic procedure in detail. The pseudo code of the EPO shown in Algorithm 1.

Algorithm 1 EPO algorithm

```
procedure
    initialize all the variables
    for < maximum number of iterations > do
        calculate ΔX using (4)
        calculate X using (3)
        for < total number of particles > do
            evaluate Y using (6)
        evaluate Y_min from using (6)
        compare Y_min with (7)
        if (7) satisfies then
            implement (8) and (9)
            reevalute l_scale using (1)
```

2.4. Modified EPO algorithm

To accelerate the convergence of the EPO, we introduce a modification. This modification is related to the calculation of *eta*. Specifically, we will modify the value of *eta* as well in every iteration as shown below:

$$eta = eta_{max} - t * \frac{eta_{max} - eta_{min}}{t_s} \tag{10}$$

where eta_{max} and eta_{min} represents maximum value (starting value of *eta*) and minimum value (ending value of *eta*) respectively. This will make the transformation more fast and efficient and the modified algorithm with"varying *eta*" is shown as Algorithm 2.

Algorithm 2 Modified EPO algorithm

```
procedure
    initialize all the variables
    for < maximum number of iterations > do
        calculate ΔX using (4)
        calculate X using (3)
        for < total number of particles > do
            evaluate Y using (6)
        evaluate Y_min from using (6)
        compare Y_min with (7)
        if (7) satisfies then
            implement (8) and (9)
            reevalute l_scale using (1)
            calculate eta using (10)
```

The algorithm of this modified version is same as that of "EPO Algorithm" except that in the current version we need to update the value of *eta* as well. In the next section we will compare both these

Table 1
Uni-modal functions.

Function	Dim	Range	f_{min}
$g_1(x) = \Sigma_{i=1}^{n}(x+2)_i^2 + 2$	30	[−10 10]	2
$g_2(x) = \Sigma_{i=1}^{n}\|x_i^2\| + \Pi_{i=1}^{n}\|x_i\|$	30	[−10 10]	0
$g_3(x) = \Sigma_{i=1}^{n}(\Sigma_{j-1}^{i}x_j)^2$	30	[−10 10]	0

Table 2
Results of uni-modal functions.

Function			f_x
$g_1(x)$	EPO	avg	2.0116
		std	0.00168
	EPO (mod.)	avg	2
		std	0
$g_2(x)$	EPO	avg	0.4388
		std	0.0604
	EPO (mod.)	avg	2.934E−22
		std	4.956E−38
$g_3(x)$	EPO	avg	0.0364
		std	0.00284
	EPO (mod.)	avg	2.189E−23
		std	0.00E+00

algorithms to find out which one is more efficient and accurate to find the optimality of test functions.

We introduced another improvement to the algorithm. After evaluating the values of all the particles using (6), we sort them from best solution to the worst (11). Out of the sorted solutions we select "n" number of best solutions and store their corresponding coordinates in an array (13). We average out the array as shown in (14) and the resultant is use to again calculate the value of function using (6). This procedure repeats recursively until an optimal solution is achieved. The purpose of this improvement is to further broaden the scope of algorithm so instead of relying on single X_{best} value now we are relying on an average value X_{avg}.

$$Y_{sort} = [Y_{best_1}\ Y_{best_2}\ Y_{best_3}\ Y_{best_4}\ \dots\ Y_{wrost}] \tag{11}$$

$$Y_{sort} = [Y_{best_1}\ Y_{best_2}\ Y_{best_3}\ Y_{best_4}\ \dots\ Y_{best_n}] \tag{12}$$

$$X_{sort} = [X_{best_1}\ X_{best_2}\ X_{best_3}\ X_{best_4}\ \dots\ X_{best_n}] \tag{13}$$

$$X_{avg} = \frac{X_{best_1} + X_{best_2} + X_{best_3} \dots + X_{best_n}}{n}. \tag{14}$$

2.5. Comparison between the two algorithms

Our main goal is to optimize the function and search for the best possible solution. Our algorithm's general framework is to employ all the particles in search space and then look for the one at the best position. This iterative process will finally converge the function to its optimal value. Both algorithms perform the same job, and we test some uni-modal and multi-modal functions to evaluate their performance and efficiency. For the test purposes, we ran both the algorithms 30 time with 500 iterations with $l_{scale} = 500$ and $res = 0.05$. For the modified EPO, the value of eta deteriorated from 0.9 and 0.8.

2.5.1. Comparison based on uni-modal functions

The functions mentioned in Table 1 are uni-modal since they have only one convergence point, all of which have 0 as an optimal solution. The corresponding results are shown in Table 2, from which it is evident that the modified EPO out-shined the EPO. The avg of the modified EPO is closer to the actual convergence point, with a small std, mainly because of the linear transformation between the exploration and the exploitation. And by a linear transformation means that the variation of eta is linear over the time t. In the original EPO algorithm, there was less diversity due to the constant value of eta, however, in the modified version, we have given more leverage to the algorithm by introducing another variable term eta, which allows the algorithm to transform between exploration and exploitation robustly.

From Table 1, it is quite evident that the modified EPO algorithm has more promising results than the EPO with constant eta. That is only because, in the revised version, we have better control over the scaling of the search space.

We used the built-in MATLAB command `fmincon` to evaluate the optimal solution of all functions mentioned and then compared those results with both algorithms. Both were efficient and approximated the optimal solution, but the careful Comparison showed that the modified EPO outclassed the simple EPO. In the modified EPO algorithm, we iteratively change the value of eta from 0.9 to 0.8, since all uni-modal function converges to 0 so the amount of accuracy in case of the modified EPO was between range 10^{-20} and 10^{-50} with standard deviation(std) between the range 10^{-20} and 10^{-60}. In contrast, the EPO with constant eta produced results with accuracy in the range 10^{-1} and 10^{-2} with standard deviation (std) between 10^{-1} and 10^{-3}.

2.5.2. Comparison based on multi-modal functions

We applied the algorithms to multi-modal functions as well. In such functions, many local optimum solutions lie at different locations within the search space, but only one optimal or global solution exists. Our algorithm is meta-heuristic-based, so our solution will have randomness despite the same initial conditions. Applying it to multi-modal functions can produce different results, which is why we ran it 30 times and calculated the standard deviation to determine our algorithm's consistency and accuracy. The functions are shown in Table 3, and their respective results are shown in Table 4 (see Table 6).

The results presented in Table 9 further clarify that the modified EPO is more efficient than EPO with constant eta. As mentioned above, multi-modal functions have several optimal solutions within the search space, so it may be difficult for an algorithm, especially meta-heuristic, to identify the most accurate optimal solution. Considering this fact, we defined a parameter "std" to finds out the standard deviation that occurs in 30 runs. It is evident from the result that "std" in the case of the modified EPO is of 10^{-2} value which is comparable with the performance of EPO accuracy. Still, the modified EPO has shown more efficient and consistent results.

We analyzed an improved version of the algorithm as well, mentioned from (11) to (14) and found that as the value of n increases, the algorithm's efficiency decreases. We applied the algorithm on both uni-modal and multi-modal functions and found out the results, shown in Table 5. The parameters kept were: $t_{s(iterations)} = 500$, $res = 0.05$, $dim = 4$, $k_{particles} = 30$, and eta varies from 0.9 to 0.8.

In the next section, we will verify that using EPO, the convergence of a function to its optimality is inevitable.

2.6. Mathematical proof for global convergence

A series converges when the sequence of its partial sums approaches a limit; that means the partial sums become closer to a given number when the number of terms increases. In proof, we will show that the convergence of EPO is inevitable. The solution converges to global optima with probability 1 when time t goes to infinity. Before starting the mathematical manipulations, we need to consider these three points.

1. Monotonically non-reducing function converges if it has an upper bound.
2. The (1) does not imply that the lower bound is the limit. The limit may be greater than it.
3. Here, convergence means the limit exists. It does not imply the limit is exactly the global optima.

Monotonicity recall (7) (8) and (9), with this we know $Y_{Best}(t+1) \geq Y_{Best}(t)$. if condition (7) is not met then, $Y_{Best}(t+1) = Y_{Best}(t)$. For upper bound recall that we are searching for the maximum. If the maximum exists, itself constructs the upper bound.

Table 3
Multi-modal functions.

Function	Dim	Range	f_{min}
$g_4(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sigma_{i=n}^{n}x_i^2}) - exp(\frac{1}{n}\sigma_{i=n}^{n}\cos(2\pi x_i)) + 20 + e$	30	[−5.12 5.12]	0
$g_5(x) = (1(1+x_1+x_2)^2(19-14x_1+3x_1^2-14x_2+6*x_1x_2+3x_2^2))$ $(30+(2x_1-3x_2^2 18-32x_1=12x_1^2+48x_2-36x_1x_2+27x_2^2))$	30	[−10 10]	3
$g_6(x) = \frac{1}{4000}\Sigma_{i=1}^{n}x_i^2 - \prod_i^n \cos(\frac{x_i}{i}) + 1$	30	[−10 10]	0

Table 4
Results of multi-modal functions.

Function			f_x
$g_4(x)$	EPO	avg	8.70×10^{-12}
		std	3.20×10^{-11}
	EPO (mod.)	avg	-1.00×10^{-13}
		std	2.00×10^{-3}
$g_5(x)$	EPO	avg	3.13
		std	4.60×10^{-16}
	EPO (mod.)	avg	3.03
		std	0.00
$g_6(x)$	EPO	avg	0.00
		std	0.00
	EPO (mod.)	avg	5.72×10^{-3}
		std	9.47×10^{-3}

This proof is particularly for proving that the convergence limit is exactly the global optima. In order to proceed let us do a little modification to the (1) as show below.

$$l_{scale} = l_{scale} * eta + L_o \qquad 0 < L_o < 1 \tag{15}$$

The Eqs. (3) and (15) plays the key role in this proof. Consider the set of global optima x_{opt} as follow:

$$x \in R^n, abs(x_i - x_o pt_i) \leq delta \tag{16}$$

where $delta > 0$ and $x = [x_i]$.

Consider the problem: For step t, how large is the probability to cast a sample into the above defined neighborhood.

$$\begin{aligned}\Omega = \{x \in R^n, x_o pt_i - delta < x_i \\ + randn < x_o pt_i + delta\}\end{aligned} \tag{17}$$

Thus the above Probability will be:

$$\begin{aligned}P = P(x_{opt\_1} - delta < x_1 + randn_1 \\ < x_{opt\_1} + delta) * P(x_{opt\_2} - delta \\ < x_2 + randn_2 < x_{opt\_2} + delta) \\ * ... * P(x_{opt\_n} - delta < x_n + randn_n \\ < x_{opt\_n} + delta)\end{aligned} \tag{18}$$

$$\begin{aligned}P = [F(x_{opt\_1} + delta - x_1) - F(x_{opt\_1} \\ - delta - x_1)] * [F(x_{opt\_2} + delta - x_2) \\ - F(x_{opt\_2} - delta - x_2)] * ... * [F(x_{opt\_n} \\ + delta - x_n) - F(x_{opt\_n} - delta - x_n)]\end{aligned}$$

where $F(x)$ is the cdf (cumulative distribution function) of $N(0, L)$, i.e., $F(x) = int\_(-inf)^x * f(y)dy$ (see Table 8).

We made the assumption that searching space is bounded. And the diameter of searching space, which is defined as the largest distance between any two points in the domain, is bounded by D. Then, we can conclude that,

$$\begin{aligned}[F(x + delta) - F(x - delta)] > min\{f(x + delta) \\ , f(x - delta)\} \\ * 2 * delta\end{aligned} \tag{19}$$

For, $[F(x_{opt\_i} + delta - x_i) - F(x_{opt\_i} - delta - x_i)]$, note that $abs(x_{opt\_i} - x_i) \leq D$ according to above definition. Thus,

$$\begin{aligned}[F(x_{opt\_i} + delta - x_i) - F(x_{opt\_i} - delta - x_i)] > \\ min\{f(D + delta), f(D - delta)\} * 2 * delta\end{aligned} \tag{20}$$

where $f(x)$ is the pdf of $N(0, 1)$ Clearly, $f(D + delta) < f(D - delta)$ if $delta \ll D$. So, $min = f(D + delta)$. Therefore, $[F(x_{opt\_i} + delta - x_i) - F(x_{opt\_i} - delta - x_i)] > f(D + delta) * 2 * delta$, for $N(0, 1)$ $f(D + delta)$ Can be computed easily. From the above we know the probability of $Y \geq [f(D + delta) * 2 * delta]^n = epsilon > 0$ where $epsilon > 0$. In order to understand how proof works consider the descriptive form below.

- Consider K particles, and their probability of falling in the neighborhood of optimal solution is $epsilon$.
- The probability of K particles never falling in the neighborhood of optimal solution is $(1 - epsilon)^k$.
- For t number of iterations this probability will be:
  $(1 - epsilon)^k * ...(1 - epsilon)^k = (1 - epsilon)^{kt}$
  This represent the probability that sample will never drop into the optima neighborhood. So far, we have successfully proved that at step t, the probability to sample a point in the neighborhood of the optimum is greater than $epsilon$.
- When t goes to infinity, $(1 - epsilon)^{kt}$ goes to 0. This means that the probability of K particles not falling in the neighborhood of optimal solution reaches to zero as the t approaches to infinity. Thus the probability of particles falling in neighborhood of optimal solution becomes 1.

3. Results and discussion

This section will do the competitive analysis and benchmark the performance of EPO with many other famous meta-heuristic algorithms. We will use the number of test functions. Based on their nature, they are divided into two groups: uni-modal functions and multi-modal functions. Uni-modal functions are those with a single optimum solution, so they are easy to handle since they converge to a single solution. Contrary, multi-modal functions are those with the number of optimal solutions. However, they also have one global convergence, but it is hard to deal with them because of the number of optimal solutions.

For the results verification of EPO, we will compare the results with Ant-Lion Optimizer (ALO) [14], Dragon Fly Optimizer (DA) [15], Particle Swarm Optimizer (PSO) [13], which is the best among the group of swarm optimizers, and GA [37] the best evolution-based optimization algorithm. In addition to them, there are other recently developed optimization techniques which includes; Flower Pollination Algorithm [38], State of Matter Search Algorithm (SMS) [39], Cuckoo Search Algorithm (CS) [40], Bat Algorithm (BA) [41], and Firefly Algorithm (FA) [42]. To quantify the results, we will run each function 30 times and calculate their average (avg) and standard deviation (std).

Each of the function will undergo 30 test run, with 500 iteration, $l_{scale} = 500$, and $res = 0.05$. Here we will employ both the EPO algorithm and compare it with the rest of the algorithms mentioned above.

3.0.1. Comparison result of uni-modal functions

Table 7 shows all the seven test functions used for the comparison. Table 7 shows results obtained after testing the uni-modal functions, and it is quite prominent that EPO outperforms the rest of the algorithms. As mentioned in Section 2.5 the range of accuracy both the EPO algorithm shows is incredible. All the test functions in uni-modal have a global optimum at 0 (zero).

Table 5
Results of further modified EPO algorithm.

Function		$n = 2$	$n = 4$	$n = 6$	$n = 8$	$n = 10$	$n = 12$
$g_2(x)$	avg	$1.82\times^{-24}$	$4.08\times^{-24}$	$2.18\times^{-19}$	$6.08\times^{-08}$	$3.74\times^{-04}$	$3.50\times^{-3}$
$g_3(x)$	avg	$1.75\times^{-42}$	$5.09\times^{-22}$	$5.39\times^{-08}$	$1.13\times^{-05}$	0.67	4.73
$g_4(x)$	avg	0.06	0.05	0.02	0.10	0.19	0.40
$g_5(x)$	avg	2.99	3.01	3.30	3.31	4.09	4.10

Table 6
Uni-modal benchmark functions.

Function	Dim	Range	f_{min}
$F_1(x) = \Sigma_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
$F_2(x) = \Sigma_{i=1}^{n} \lvert x_i^2 \rvert + \Pi_{i=1}^{n} \lvert x_i \rvert$	30	$[-10, 10]$	0
$F_3(x) = \Sigma_{i=1}^{n} (\Sigma_{j-1}^{i} x_j)^2$	30	$[-100, 100]$	0
$F_4(x) = max(\lvert x \rvert)\ 1 \leq i \leq n$	30	$[-100, 100]$	0
$F_5(x) = \Sigma_{i=1}^{n} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0
$F_6(x) = \Sigma_{i=1}^{n} ([x_i + 0.5])^2$	30	$[-100, 100]$	0
$F_7(x) = \Sigma_{i=1}^{n} i x_i^4 + random[0, 1]$	30	$[-1.28, 1.28]$	0

3.0.2. Comparison result of multi-modal functions

The comparison results of the multi-modal functions are shown in Table 9, it is evident from the table that EPO outclassed the rest of the algorithms and its average converges to the global optimum solution with the least standard deviation. It is because of the efficient and vast exploration of the algorithm that then transformed into exploitation when it reaches to the region of optimum solution. As mentioned earlier, multi-modal functions have number of local optimum solution with only one global solution, the results we obtained showed that EPO efficiently avoid the local optima and converges to the global solution only.

4. Analysis of the EPO algorithm

In Section 2, we discussed the performance of both EPO algorithms in detail. We also discuss how EPO (mod.) is more powerful than EPO. In Section 3, we did a comprehensive, competitive analysis of both EPO algorithms with other well-known algorithms, which includes; ant-lion (ALO), dragon-fly, particle swarm , *etc.* From the detailed analysis, we concluded that in both uni and multi-modal functions EPO outperforms the testing algorithms and very efficiently achieve the required optimal point.

In this section, we will do a detailed analysis of the EPO algorithm; we will manipulate the controlling unit of the algorithm and will find out how it affects the output. Before starting, the algorithm's parameters, particularly for this practice, are; $l_scale = 100, res = 0.05, t_{s(iteration)} = 500, particles = 30, and\ dimension = 2$. As explained earlier in a modified version of the algorithm, we manipulated the value of *eta* using (10). The value of *eta*, in fact, faster the transformation from exploration to exploitation of our algorithm, making it more efficient. To explore the effect of *eta* on the working of our algorithm, we choose three functions, $F1, F4, and\ F7$ from the uni-modal function list 1 and run the algorithm with three-four different values of the , *eta*, i.e., 0.9, 0.9–0.8, 0.9–0.7, and 0.9–0.6. The obtained results are shown in Fig. 2.

It is evident from the figure that as the value of eta_{min} decreases, in other words, as the difference between eta_{min} and eta_{max} increases, the test functions converge to their optimal value very fast, and more efficiently. For example, in the case of $F1$, if we run the algorithm with *eta* constant, after 500 iterations, the function ended up at around 10^{-15}, identical results were obtained within 150 iterations when *eta* varies from 0.9 to 0.8, the same trend is followed further. All three functions behave alike, but if we meticulously observe the trend in $F7$, we notice a slight variation, and the trend is not the same as in previous cases. There is a bit of irregularity when we move from }}*eta* varies from 0.9 to 0.8ε to }}*eta* varies from 0.9 to 0.7ε and further }}*eta* varies from 0.9 to 0.6ε. From the first two functions, $F1$ and $F4$, the algorithms become more efficient as the range of *eta* increases. But, $F7$ broke this trend and showed that is only sometimes the case. Namely, as the complexity of a goal function increases, we will find certain anomalies in the algorithm functioning concerning the range of *eta*. From the testing of uni-modal functions, we concluded two things:

- For simple functions, as the range of *eta* increases, the function will converge faster to the optimal point.
- For complex functions, as the range of *eta* increases, we will find anomalies in the system, and there we need to tune the value of *eta* to achieve the optimal point.

We extended the analysis to the multi-modal problems presented in Table 3. The generated results are shown in Fig. 3. It proves our above-mentioned second point that as the complexity of a test function increases, the algorithm does not sustain the general trend of *eta*. As the range of *eta* increases, the convergence of the functions gets affected. Since multi-modal functions are more complex than uni-modal (because they have several local minima and one global minima), it is not wise to accelerate the transformation from the exploration to the exploitation phase, as there is an excellent chance to avoid the optimal global solution. It is observed, in that case, we need to tune the range of *eta* to obtain the optimality.

Function $F8$ shows a complex search space; out of all the tested functions, it is the only one whose optimal solution depends on the number of dimensions employed in the algorithm. When we benchmarked the EPO algorithm, the optimal solution obtained for this function differed from the one we obtained here. This is because the dimension was 30, and here it is 2. $F1$ shows a fluctuation in the results obtained using different ranges of *eta*; despite the fluctuation, if we look at the graph, we can see that *std* of this function is around 370 considering the different ranges of the *eta*. Similarly, $F9$ also shows the same trend. The common thing in both functions was that they were close to their optimal solution despite the fluctuations. The $F10$ shows a very different trend; for the first three cases of the *eta*, it shows more or less the same trend but for }}*eta* varies from 0.9 to 0.6ε it shows a very different trend, almost a constant line and not closer to the optimal solution as compare to other cases.

The other controlling parameters involved in the algorithm are; l_{scale} and *res*; both are directly or indirectly related to *eta*. l_{scale} represents the area of search space that our algorithm will explore, and its value should be manageable and manageable in both cases. It may need help to track the optimum point accurately. *res* is also a controlling variable because if its value is higher than l_{scale}, then we will have $eta > 1$, which means that on each iteration, l_{scale} will become greater and greater, so it will be unable for the algorithm to track the optimal point.

We can further tune the results with the help of t_s and the number of runs. Here, t_s represents the number of iterations to obtain a more stable output; it is better to have it in triple digits, but it varies from function to function so that we can set it accordingly. After running the algorithm number of times, e.g., 10 to 20 times, obtaining the average *avg* and standard deviation *std* of the results, it will produce stable solutions.

5. Constrained optimization using EPO

In this section, we will apply the EPO algorithm to some constrained optimization problems and will compare the results with two other algorithms: ALO (ant-lion optimizer) and DA (dragon optimizer). Each

Table 7
Results of uni-modal benchmark functions.

Function	EPO (mod.)		EPO		DA		ALO		PSO	
	avg	std	avg	std	avg	std	avg	std	avg	std
F1	3.93×10^{-45}	6.56×10^{-61}	1.14×10^{-02}	1.66×10^{-03}	2.85×10^{-18}	7.16×10^{-18}	2.59×10^{-10}	1.65×10^{-10}	2.70×10^{-09}	1.00×10^{-09}
F2	3.26×10^{-22}	4.96×10^{-38}	4.39×10^{-01}	5.83×10^{-02}	1.49×10^{-05}	3.76×10^{-05}	1.84×10^{-06}	6.58×10^{-07}	7.15×10^{-05}	2.26×10^{-05}
F3	1.04×10^{-52}	1.95×10^{-68}	8.47×10^{-09}	2.45×10^{-08}	1.29×10^{-06}	2.10×10^{-06}	6.06×10^{-10}	6.34×10^{-10}	4.71×10^{-10}	$1.49 \times 10^{-06}6$
F4	1.04×10^{-52}	1.95×10^{-68}	8.47×10^{-09}	2.45×10^{-08}	1.29×10^{-06}	2.10×10^{-06}	6.06×10^{-10}	6.34×10^{-10}	4.71×10^{-06}	1.49×10^{-06}
F5	0.00	0.00	0.00	0.00	7.60	6.78	0.34	0.10	0.12	0.21
F6	0.00	0.00	8.03×10^{-03}	3.30×10^{-03}	4.17×10^{-16}	1.32×10^{-01}	2.53×10^{-10}	1.09×10^{-10}	5.23×10^{-07}	2.74×10^{-06}
F7	1.77×10^{-03}	0.00	1.38×10^{-03}	2.64×10^{-03}	1.02×10^{-02}	4.69×10^{-03}	4.292×10^{-03}	5.08×10^{-03}	1.398×10^{-03}	1.26×10^{-03}
	SMS		BA		FPA		CS		FA	
	avg	std	avg	std	avg	std	avg	std	avg	std
F1	0.05	0.01	0.77	0.52	1.06×10^{-07}	1.27×10^{-07}	6.50×10^{-03}	2.05×10^{-04}	0.03	0.01
F2	6.84×10^{-03}	1.577×10^{-03}	0.33	3.81	6.24×10^{-04}	1.76×10^{-04}	0.21	3.98×10^{-02}	0.05	0.01
F3	0.96	0.82	0.11	0.76	5.67×10^{-08}	3.90×10^{-08}	0.24	0.21	4.93×10^{-02}	1.94×10^{-02}
F4	0.27	5.74×10^{-03}	0.19	0.89	3.83×10^{-03}	2.18×10^{-03}	1.12×10^{-05}	8.25×10^{-06}	0.14	0.03
F5	0.08	0.14	0.33	0.30	0.78	0.36	7.19×10^{-03}	7.22×10^{-03}	2.17	1.44
F6	0.12	0.08	0.77	0.67	1.09×10^{-07}	1.25×10^{-07}	5.95×10^{-05}	1.08×10^{-06}	0.05	0.01
F7	3.04×10^{-04}	2.58×10^{-04}	0.13	0.11	3.10×10^{-03}	1.36×10^{-03}	1.32×10^{-03}	7.28×10^{-04}	8.53×10^{-04}	5.04×10^{-04}

Table 8
Multi-modal benchmark functions.

Function	Dim	Range	f_{min}
$F_8(x) = \Sigma_{i=1}^{n} - x_i \sin(\sqrt{\lvert x_i \rvert})$	30	$[-500, 500]$	$-418.9 \times \text{Dim}^a$
$F_9(x) = \Sigma_{i=1}^{n} [x_i^2 - 10\cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0
$F_{10}(x) = -20exp(-0.2\sqrt{\frac{1}{n}\sigma_{i=n}^{n} x_i^2}) - exp(\frac{1}{n}\sigma_{i=n}^{n} \cos(2\pi x_i)) + 20 + e$	30	$[-5.12, 5.12]$	0

Table 9
Results of multi-modal benchmark functions.

Function	EPO (modified)		EPO		DA		ALO		PSO	
	avg	std	avg	std	avg	std	avg	std	avg	std
F8	-6.23×10^{03}	7.67E−12	-7.11×10^{04}	0.00E+00	-2.85×10^{03}	383.6	-1.60×10^{03}	314.4	-1.36×10^{03}	146.4
F9	−290.0	0.00	−286.1	5.99×10^{-14}	16.01	9.47	7.71×10^{-06}	8.45×10^{-06}	0.27	0.21
F10	-1.06×10^{-13}	2.058×10^{-03}	-8.07×10^{-12}	1.47×10^{-11}	0.23	0.48	3.73×10^{-15}	1.50×10^{-15}	1.11×10^{-09}	2.39×10^{-11}
	SMS		BA		FPA		CS		FA	
	avg	std	avg	std	avg	std	avg	std	avg	std
F8	-4.20×10^{-06}	9.36×10^{-16}	-1.06×10^{-03}	8.98×10^{-03}	-1.84×10^{-04}	5.04×10^{-03}	-2.09×10^{-03}	7.61×10^{-03}	-1.24×10^{-03}	3.53×10^{-03}
F9	1.32	0.32	1.23	0.68	0.27	0.06	0.12	0.02	0.26	0.18
F10	8.88×10^{-06}	8.56×10^{-09}	0.12	0.043251	730×10^{-03}	7.09×10^{-03}	8.16×10^{-09}	1.63×10^{-08}	0.16	0.05

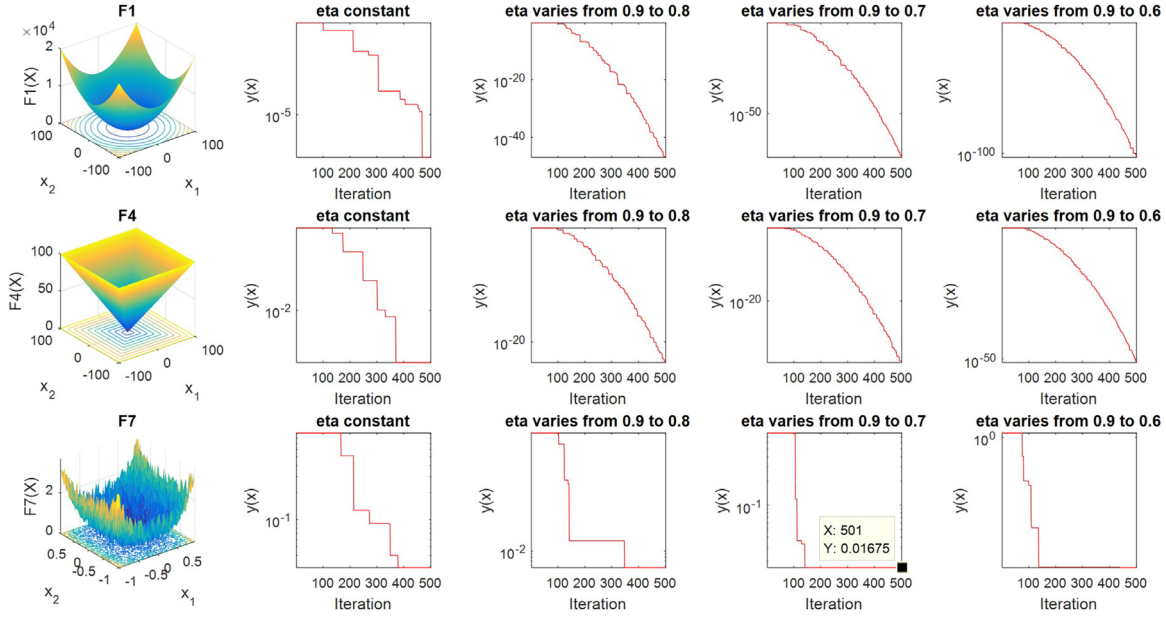

Fig. 2. Search space of uni-modal functions, convergence with constant *eta*, convergence when *eta* varies from 0.9 to 0.8, convergence when *eta* varies from 0.9 to 0.7, convergence when *eta* varies from 0.9 to 0.6.

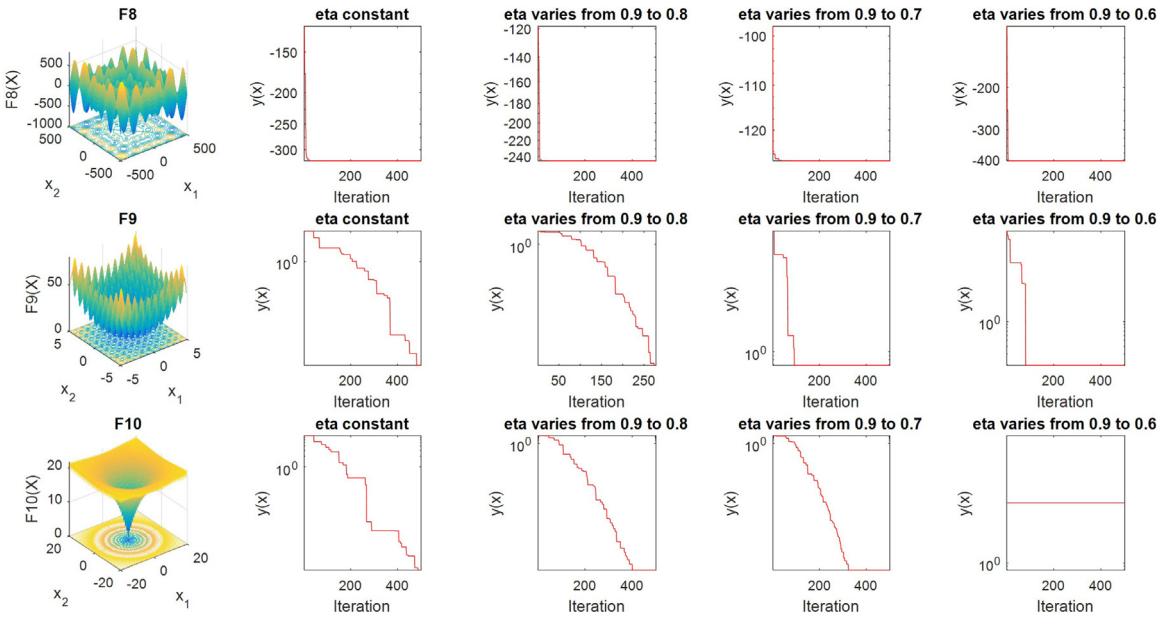

Fig. 3. Search space of multi-modal functions, convergence with constant *eta*, convergence when *eta* varies from 0.9 to 0.8, convergence when *eta* varies from 0.9 to 0.7, convergence when *eta* varies from 0.9 to 0.6.

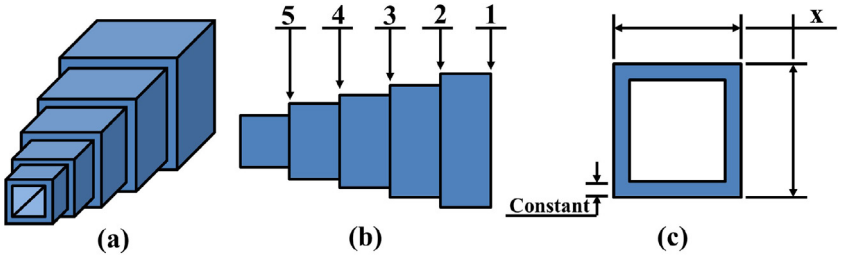

Fig. 4. The cantilever beam design problem. (a) is the design of cantilever, (b) shows that all the five blocks of the cantilever are in decreasing size, (c) represents that the thickness of each square block remains constant whereas the length x is the one needs to optimize.

Table 10
Results of cantilever beam design problem.

Algorithms		x_1	x_2	x_3	x_4	x_5	f_x	t (s)
EPO	avg	6.00	5.25	4.50	3.57	2.13	13.36	1.31
	std	0.93×10^{-15}	0.93×10^{-15}	0.93×10^{-15}	0.93×10^{-15}	0.46×10^{-15}	1.87×10^{-15}	
EPO (mod.)	avg	6.01	5.26	4.53	3.48	2.17	13.36	1.19
	std	0.93×10^{-16}	0.00	0.93×10^{-16}	0.93×10^{-16}	0.46×10^{-16}	1.87×10^{-15}	
ALO	avg	6.03	5.29	4.48	3.50	2.15	13.36	52.26
	std	0.01	0.04	0.03	0.03	0.02	8.62×10^{-04}	
DA	avg	6.22	5.57	4.45	3.37	2.22	13.60	117.3
	std	0.53	1.01	0.23	0.18	0.27	5.00×10^{-01}	

problem has some constraints with it, we will run the problem 10 times, will *avg* (average) their optimal coordinates and their respective solutions. We will also calculate the *std* (standard deviation) and then we will benchmark the performance of algorithms.

5.1. Cantilever beam design problem

Cantilever is the structure of five hollow square boxes mounted on each other with the decreasing sizes as shown in Fig. 4(a). Each box has a constant thickness whereas different length x, as shown in Fig. 4(c). Here the optimization problem is to minimize the weight of the lever. There are two constraint for the objective function: variable constraints and vertical displacement constraint [43]. The mathematical representation of the problem is as follows:

$$Consider \quad \vec{x} = [x_1, x_2, x_3, x_4, x_5]$$

$$Minimize \quad f(\vec{x}) = 0.6224(x_1 + x_2 + x_3 + x_4 + x_5)$$

$$Subject\ to \quad g(\vec{x}) = \frac{61}{x_1^3} + \frac{37}{x_2^3}\frac{19}{x_3^3}\frac{7}{x_4^3}\frac{1}{x_5^3}$$

$$Variable\ range \quad 0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100.$$

The results obtained after testing the function are shown in Table 10. From the results arranged in the table, it is obvious that both the EPO algorithms have outperformed ALO and DA. Although the *avg* value of five blocks of both the EPO algorithm was almost same as that of ALO and DA, but least *std* in both, std_x and std_{f_x}.

The std_x and std_{f_x} are far much better than others which mean that the probability of EPO algorithm deviating from its mean value or optimum value in our case is very small. The time t_s that EPO algorithms took to complete the search is also much lesser than the time corresponding to ALO and DA, this is still a simple engineering

Table 11
Results of three-bar truss design problem.

Function		x_1	x_2	f_x	t (s)
EPO	avg	0.79	0.39	2.63	1.22
	std	0.11E−15	0.058E−15	4.68E−16	
EPO (mod.)	avg	0.79	0.39	2.63	1.67
	std	0.11E−15	0.05E−15	0.00	
ALO	avg	0.78	0.41	2.63	24.20
	std	0.007	0.01	4.97E−04	
DA	avg	0.82	0.34	2.66	94.17
	std	0.06	0.13	6.01E−02	

problem and AO and DA took a lot of time to complete the algorithm consider a complex problem with several inputs and constraints they will take even more time in that case. Contrary to that, EPO algorithm will save a lot of time and will produce the results more efficiently. This is a practical example that shows the extent to which we can apply this algorithm, it is sufficient enough to handle engineering problem with higher efficiency and accuracy.

5.2. Three-bar truss design problem

The second constrained optimization problem is the design of the three-bar truss to minimize its weight, as shown in Fig. 5. The objective function of the problem is very simple, whereas it is highly constrained. Its structure under goes some severe constraints which include: stress, deflection, and buckling constraints [44]. The mathematical model of the problem is shown below.

$$\begin{aligned}
&Consider && \vec{x} = [x_1, x_2] \\
&Minimize && f(\vec{x}) = (2\sqrt{2}x_1 + x_2) * l \\
&Subject\ to && g_1(\vec{x}) = \frac{\sqrt{2}x_1}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \\
& && g_2(\vec{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \le 0 \\
& && g_3(\vec{x}) = \frac{1}{\sqrt{2}x_2 + x_1} P - \sigma \le 0 \\
&Variable\ range && 0 \le x_1, x_2 \le 1.
\end{aligned}$$

We tested the three-bar truss design in doing so we employed four testing algorithms: eagle perching optimization (EPO), modified eagle perching optimization (EPO mod.), ant-lion optimization, and dragonfly optimization (DA). We performed 10 runs on this problem and obtained the *avg* (average), *std* (standard deviation), and total time it took t. The results are shown in Table 11, which shows the promising results of EPO and EPO (mod.) with the *std* of range 10^{-15} to 10^{-16}. The time the considered algorithms took to complete 10 runs is also minimal in case of EPO algorithms around 1.6 s, whereas ALO and DA are not even comparable with them.

5.3. Gear train design problem

This is another problem of constrained optimization, it is related to the making of train gears, shown in Fig. 6 and the optimization problem is to find the optimal number of tooth for the four gears to minimize the gear ratio. It is not highly constrained problem, it has only one constrain the range of number of tooth for the gear [45]. The mathematical formulation of the problem is shown below.

$$\begin{aligned}
&Consider && \vec{x} = [x_1, x_2, x_3, x_4] \\
&Minimize && f(\vec{x}) = \left(\frac{1}{6.931} - \frac{x_3 * x_2}{x_1 * x_4}\right)^2 \\
&Variable\ range && 12 \le x_1, x_2, x_3, x_4 \le 60.
\end{aligned}$$

We tested five algorithms and then compared their results with the built-in MATLAB function `fmincon` which are shown in Table 12, we

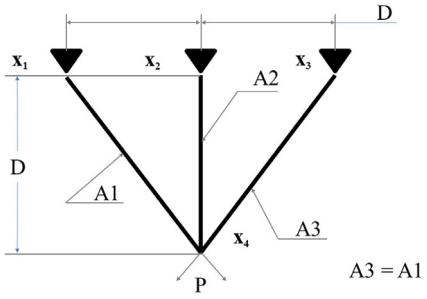

Fig. 5. Three-bar truss design problem.

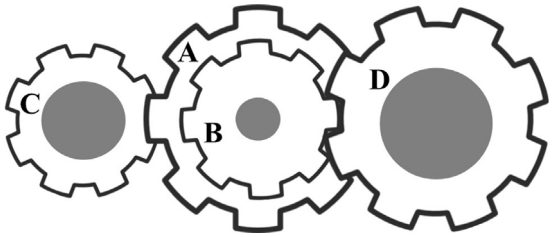

Fig. 6. Gear train design problem.

round of the value since we are dealing with the number of tooth for the gears. From the arranged results, it is evident that EPO algorithms evaluated the results more nearer to the actual solutions. Especially, EPO (mod.) has not only the more accurate solution but also have the least *std* (standard deviation) and the evaluation time $t = 0.9723$.

From all the above-explained three problems, it is clear that EPO can handle the constrained, real-world problems. It also proves that its evaluation time is many-fold less than the other test optimization algorithms. If you further narrow the performance criteria, then EPO (modified) looks more promising than EPO because of the ability of EPO (modified) to explore more areas in less time. In other words, the modified EPO has an efficient transformation from exploration to exploitation.

We would, like to point out some limitations and the future direction of our work. Like any swarm algorithm, EPO cold be computationally expensive and slow when optimizing complex time-taking objective functions, e.g., robotics systems, portfolio optimization. In that case, single particle algorithms like BAS are preferable. In future, we intend to collect the comprehensive data on the design and performance of state-of-the-art heuristic algorithms, so that a comprehensive comparison could be made based on several aspects. Likewise, we will extend our performance measure metrics by including statistical t-test [46] like techniques. We also intend to find a way to find the optimal selection of hyper-parameter, and Hyperband and Bayesian optimization are one of the techniques that cold be employed.

Table 12
Results of Gear train design problem.

Function		x_1	x_2	x_3	x_4	f_x
fmincon	avg	42	16	16	42	3×10^{-12}
	std	NA	NA	NA	NA	NA
EPO	avg	49	15	26	57	3×10^{-18}
	std	0.0×10^{-14}	0.1×10^{-14}	0.0×10^{-14}	0.0×10^{-14}	8×10^{-34}
EPO (mod.)	avg	42	15	18	44	0.0
	std	0.00	0.00	0.30×10^{-14}	0.70×10^{-14}	0.0
ALO	avg	49	19	16	43	2E−12
	std	NA	NA	NA	NA	NA
CS	avg	43	16	19	49	2E−12
	std	NA	NA	NA	NA	NA
MBA	avg	43	16	19	49	2E−12
	std	NA	NA	NA	NA	NA

6. Conclusion

EPO (Eagle Perching Optimization) is a unique nature-inspired optimization algorithm. It is modeled on the behavior of an eagle, which seeks out from the highest spot in the environment towards its prey. We exploited its nature, mathematically formulated it, and suggested two algorithms based on its properties. One is a straightforward EPO algorithm with a constant degradation rate of search space, whereas the modified variant exponentially accelerates this decay and makes it robust. Later, we showed the theoretical soundness of the algorithm. We compared two EPO variants, and it is determined that the latter has better convergence. In addition, we evaluated its performance with several state-of-the-art heuristic optimization techniques using benchmark uni-modal and multi-modal functions. According to the findings, the EPO algorithm is comparable with other algorithms. It also shows that EPO converges to the optimal solution in fewer iterations with smaller standard deviation. We performed a comprehensive investigation of the method and determined that, for uni-modal functions, the general trend is that as the range of *eta* expands, the algorithm's performance improves. However, this is different with complex multi-modal functions, for which the eta range must be fine-tuned. The optimal eta range we have determined for our test functions by hit-and-trial is between 0.9 and 0.8. We further evaluated its performance by deploying it to solve real-world restricted optimization problems and compared its findings to those of other algorithms and MATLAB's built-in function. We discovered that EPO calculated optimal values with greater precision and shorter time, demonstrating the algorithm's increased efficiency.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

References

[1] C. Muro, R. Escobedo, L. Spector, R. Coppinger, Wolf-pack (Canis lupus) hunting strategies emerge from simple rules in computational simulations, Behav. Process. 88 (3) (2011) 192–197.

[2] X. Jiang, S. Li, BAS: Beetle antennae search algorithm for optimization problems. arXiv, 2017, arXiv preprint arXiv:1710.10724.

[3] A.T. Khan, X. Cao, S. Li, B. Hu, V.N. Katsikis, Quantum beetle antennae search: a novel technique for the constrained portfolio optimization problem, Sci. China Inform. Sci. (2020).

[4] A.H. Khan, S. Li, X. Luo, Obstacle avoidance and tracking control of redundant robotic manipulator: An RNN-based metaheuristic approach, IEEE Trans. Ind. Inform. 16 (7) (2019) 4670–4680.

[5] A.H. Khan, X. Cao, S. Li, V.N. Katsikis, L. Liao, BAS-ADAM: an ADAM based approach to improve the performance of beetle antennae search optimizer, IEEE/CAA J. Autom. Sin. 7 (2) (2020) 461–471.

[6] P. Dario, G. Sandini, P. Aebischer, Robots and Biological Systems: Towards a New Bionics?: Proceedings of the NATO Advanced Workshop on Robots and Biological Systems, Held At Il Ciocco, Toscana, Italy, June 26–30, 1989, Vol. 102, Springer Science & Business Media, 2012.

[7] E. Bonabeau, M. Dorigo, G. Theraulaz, G. Theraulaz, Swarm Intelligence: from Natural to Artificial Systems, No. 1, Oxford University Press, 1999.

[8] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz, A. Cosar, A survey on new generation metaheuristic algorithms, Comput. Ind. Eng. 137 (2019) 106040.

[9] Y. Xu, R. Jin, T. Yang, First-order stochastic algorithms for escaping from saddle points in almost linear time, 2017, arXiv preprint arXiv:1711.01944.

[10] O. Bozorg-Haddad, M. Solgi, H.A. Loáiciga, Meta-Heuristic and Evolutionary Algorithms for Engineering Optimization, John Wiley & Sons, 2017.

[11] B. Morales-Castañeda, D. Zaldivar, E. Cuevas, F. Fausto, A. Rodríguez, A better balance in metaheuristic algorithms: Does it exist? Swarm Evol. Comput. 54 (2020) 100671.

[12] H. Chiroma, A.Y. Gital, N. Rana, M.A. Shafi'i, A.N. Muhammad, A.Y. Umar, A.I. Abubakar, Nature inspired meta-heuristic algorithms for deep learning: Recent progress and novel perspective, in: Science and Information Conference, Springer, 2019, pp. 59–70.

[13] J.C. Bansal, Particle swarm optimization, in: Evolutionary and Swarm Intelligence Algorithms, Springer, 2019, pp. 11–23.

[14] L. Abualigah, M. Shehab, M. Alshinwan, S. Mirjalili, M. Abd Elaziz, Ant lion optimizer: a comprehensive survey of its variants and applications, Arch. Comput. Methods Eng. (2020) 1–20.

[15] K. Chatra, V. Kuppili, D.R. Edla, Texture image classification using deep neural network and binary dragon fly optimization with a novel fitness function, Wirel. Pers. Commun. 108 (3) (2019) 1513–1528.

[16] M.H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, An improved grey wolf optimizer for solving engineering problems, Expert Syst. Appl. 166 (2021) 113917.

[17] M.A. Mellal, E. Zio, System reliability-redundancy optimization with cold-standby strategy by an enhanced nest cuckoo optimization algorithm, Reliab. Eng. Syst. Saf. 201 (2020) 106973.

[18] A.T. Khan, X. Cao, B. Liao, A. Francis, Bio-inspired machine learning for distributed confidential multi-portfolio selection problem, Biomimetics 7 (3) (2022) 124.

[19] A.T. Khan, X. Cao, S. Li, Using quadratic interpolated beetle antennae search for higher dimensional portfolio selection under cardinality constraints, Comput. Econ. (2022) 1–23.

[20] A.T. Khan, X. Cao, I. Brajevic, P.S. Stanimirovic, V.N. Katsikis, S. Li, Non-linear activated beetle antennae search: A novel technique for non-convex tax-aware portfolio optimization problem, Expert Syst. Appl. 197 (2022) 116631.

[21] A.T. Khan, S. Li, X. Cao, Control framework for cooperative robots in smart home using bio-inspired neural network, Measurement 167 (2021) 108253.

[22] A.T. Khan, S. Li, S. Kadry, Y. Nam, Control framework for trajectory planning of soft manipulator using optimized RRT algorithm, IEEE Access 8 (2020) 171730–171743.

[23] A.T. Khan, S. Li, Z. Li, Obstacle avoidance and model-free tracking control for home automation using bio-inspired approach, in: Advanced Control for Applications: Engineering and Industrial Systems, Wiley Online Library, e63.

[24] A.T. Khan, S. Li, Human guided cooperative robotic agents in smart home using beetle antennae search, Sci. China Inform. Sci. (2021).

[25] A.T. Khan, S. Li, X. Zhou, Trajectory optimization of 5-link biped robot using beetle antennae search, IEEE Trans. Circuits Syst. II (2021) 1, http://dx.doi.org/10.1109/TCSII.2021.3062639.

[26] A.H. Khan, S. Li, D. Chen, L. Liao, Tracking control of redundant mobile manipulator: An RNN based metaheuristic approach, Neurocomputing 400 (2020) 272–284.

[27] A.H. Khan, X. Cao, V.N. Katsikis, P. Stanimirović, I. Brajević, S. Li, S. Kadry, Y. Nam, Optimal portfolio management for engineering problems using non-convex cardinality constraint: a computing perspective, IEEE Access 8 (2020) 57437–57450.
[28] A.H. Khan, S. Li, X. Cao, Tracking control of redundant manipulator under active remote center-of-motion constraints: an RNN-based metaheuristic approach, Sci. China Inf. Sci. 64 (3) (2021) 1–18.
[29] A. D'Ambrosio, D. Spiller, F. Curti, Improved magnetic charged system search optimization algorithm with application to satellite formation flying, Eng. Appl. Artif. Intell. 89 (2020) 103473.
[30] M. Mareli, B. Twala, An adaptive Cuckoo search algorithm for optimisation, Appl. Comput. Inf. 14 (2) (2018) 107–115.
[31] E. Rashedi, E. Rashedi, H. Nezamabadi-pour, A comprehensive survey on gravitational search algorithm, Swarm Evol. Comput. 41 (2018) 141–158.
[32] R. Burman, S. Chakrabarti, S. Das, Democracy-inspired particle swarm optimizer with the concept of peer groups, Soft Comput. 21 (12) (2017) 3267–3286.
[33] D. Yousri, D. Allam, M. Eteiba, P.N. Suganthan, Static and dynamic photovoltaic models' parameters identification using chaotic heterogeneous comprehensive learning particle swarm optimizer variants, Energy Convers. Manage. 182 (2019) 546–563.
[34] F.H. Aguiar-Silva, T.M. Sanaiotti, O. Jaudoin, A.C. Srbek-Araujo, G. Siqueira, A. Banhos, Harpy Eagle Sightings, Traces and Nesting Records at the" Reserva Natural Vale", a Brazilian Atlantic Forest Remnant in Espírito Santo, Brazil, Vol. 20, No. 2, Revista Brasileira de Ornitologia, 2012, pp. 148–155.
[35] S. McAlpine, I. Smail, R.G. Bower, A. Swinbank, J.W. Trayford, T. Theuns, M. Baes, P. Camps, R.A. Crain, J. Schaye, The nature of submillimetre and highly star-forming galaxies in the EAGLE simulation, Mon. Not. R. Astron. Soc. 488 (2) (2019) 2440–2454.
[36] K.M. Laurent, B. Fogg, T. Ginsburg, C. Halverson, M.J. Lanzone, T.A. Miller, D.W. Winkler, G.P. Bewley, Turbulence explains the accelerations of an eagle in natural flight, Proc. Natl. Acad. Sci. 118 (23) (2021) e2102588118.
[37] S. Mirjalili, Genetic algorithm, in: Evolutionary Algorithms and Neural Networks, Springer, 2019, pp. 43–55.
[38] T. Adithiyaa, D. Chandramohan, T. Sathish, Flower pollination algorithm for the optimization of stair casting parameter for the preparation of AMC, Mater. Today Proc. 21 (2020) 882–886.
[39] Ş. Gulcu, Training of the artificial neural networks using states of matter search algorithm, Int. J. Intell. Syst. Appl. Eng. 8 (3) (2020) 131–136.
[40] M. Mareli, B. Twala, An adaptive Cuckoo search algorithm for optimisation, Appl. Comput. Inf. 14 (2) (2018) 107–115.
[41] Y. Wang, P. Wang, J. Zhang, Z. Cui, X. Cai, W. Zhang, J. Chen, A novel bat algorithm with multiple strategies coupling for numerical optimization, Mathematics 7 (2) (2019) 135.
[42] H. Wang, W. Wang, X. Zhou, H. Sun, J. Zhao, X. Yu, Z. Cui, Firefly algorithm with neighborhood attraction, Inform. Sci. 382 (2017) 374–387.
[43] C.A. McCormick, M.R. Shepherd, Optimization of an acoustic black hole vibration absorber at the end of a cantilever beam, J. Acoust. Soc. Am. 145 (6) (2019) EL593–EL597.
[44] H. Fauzi, U. Batool, A three-bar truss design using single-solution simulated Kalman filter optimizer, Mekatronika 1 (2) (2019) 98–102.
[45] R.V. Rao, R.B. Pawar, Optimal weight design of a spur gear train using rao algorithms, in: International Conference on Sustainable and Innovative Solutions for Current Challenges in Engineering & Technology, Springer, 2019, pp. 351–362.
[46] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18.