Neural Computing and Applications (2024) 36:11137–11170
https://doi.org/10.1007/s00521-024-09648-4

ORIGINAL ARTICLE

Enhanced prairie dog optimization with Levy flight and dynamic opposition-based learning for global optimization and engineering design problems

Saptadeep Biswas[1] · Azharuddin Shaikh[1] · Absalom El-Shamir Ezugwu[2] · Japie Greeff[3] · Seyedali Mirjalili[4] · Uttam Kumar Bera[1] · Laith Abualigah[5,6,7,8,9,10,11,12]

Received: 20 July 2023 / Accepted: 21 February 2024 / Published online: 30 March 2024

Abstract
This study proposes a new prairie dog optimization algorithm version called EPDO. This new version aims to address the issues of premature convergence and slow convergence that were observed in the original PDO algorithm. To improve performance, several modifications are introduced in EPDO. First, a dynamic opposite learning strategy is employed to increase the diversity of the population and prevent premature convergence. This strategy helps the algorithm avoid falling into local optima and promotes global optimization. Additionally, the Lévy dynamic random walk technique is utilized in EPDO. This modified Lévy flight with random walk reduces the algorithm's running time for the test function's ideal value, accelerating its convergence. The proposed approach is evaluated using 33 benchmark problems from CEC 2017 and compared against seven other comparative techniques: GWO, MFO, ALO, WOA, DA, SCA, and RSA. Numerical results demonstrate that EPDO produces good outcomes and performs well in solving benchmark problems. To further validate the results and assess reliability, the authors employ average rank tests, the measurement of alternatives, and ranking according to the compromise solution (MARCOS) method, as well as a convergence report of EPDO and other algorithms. Furthermore, the effectiveness of the EPDO algorithm is demonstrated by applying it to five design problems. The results indicate that EPDO achieves impressive outcomes and proves its capability to address practical issues. The algorithm performs well in solving benchmark and practical design problems, as supported by the numerical results and validation methods used in the study.

Keywords Enhanced prairie dog optimization · Lévy distribution · Dynamic opposition-based learning · Global optimization · Engineering design problem · MARCOS MCDM method · Benchmark problems

1 Introduction

Optimization issues in the real world are difficult. Due to the growing number of variables, dimensions, time complexity, space complexity, etc., they are becoming increasingly complex. To deal with such challenging optimization problems, one of the best choices is to use meta-heuristics algorithm (MA) [32]. This is due to the stochastic nature of MA, which allows them the ability to show less probability of stagnation in local optimums and provide high exploratory capability [28]. MA is less expensive and more efficient with respect to other traditional optimization algorithms. MA have derivation-free mechanisms in exploring search spaces to find the optimum solution [30]. Multiple solutions or population-based MA are better because they are capable of effective exploration of the search space of the optimization problem [20].

Nowadays, MAs are applied to figure out various technical issues like design and structural optimization problems. Usually, these algorithms initialize their runs with a set of randomly generated solutions and continue the procedure of evaluating the objective functions until the global optimum is obtained [14]. Broadly speaking, MAs are divided into two categories, viz. meta-heuristics with single solution (MSS) and population-based meta-heuristics (PBM) [17]. In MSS methods [such as simulated annealing (SA) [21], tabu search (TS) [13], variable neighbourhood search (VNS) [31]] a single search agent will perform the

Extended author information available on the last page of the article

search operations, and on the contrary, a group of search agents is involved in PBM algorithms. In PBM optimization algorithms, each of the solution's positions is updated using single and social information. Furthermore, search space can be checked using many solutions as quickly as possible. Therefore, PMB's outcomes are exhibited best than MSS. In the literature, PBM is classified into five types. They are evolutionary algorithm (EA), swarm intelligence (SI), physics-based algorithm, human-based algorithm and math-based algorithm, respectively, [8]. The categorization of meta-heuristic algorithms is shown in Fig. 1. Nature-inspired meta-heuristics are categorized into five classes evolutionary algorithm, swarm intelligence, physical and chemical-based algorithm, human-based algorithm and math-based algorithm, respectively [1, 27].

Introducing new advanced MA is to get the optimal value of complicated optimization problems as quickly as possible and achieve a robust optimization process [43, 48]. For every optimization problem, a specific optimal solution exists called global optimal [33]. At the same time, all the solutions obtained from optimization algorithms are not necessarily globally optimal. However, they are acceptable if their solutions get incredibly near to the overall best solution. These unique solutions are called quasi-optimal solutions [5, 6]. Using this idea, we can conclude that an optimization algorithm is better than others in optimizing a particular problem based on the quasi-optimal solution, which is closer to global [7]. This is a new direction of research for developing new optimization algorithms. That question is whether there is still a need for new optimization algorithms to be introduced in light of the numerous MAs published in our scientific literature. As stated by the No Free Lunch theorem [46], no single optimization algorithm can find globally optimal solutions to all existing optimization problems. This explains the fact that MA can solve a set of optimization problems but not another set of optimization problems. Therefore, no optimization algorithm can solve all existing optimization problems. This is another reason for developing novel optimization algorithms to solve optimization issues considering acceptable quasi-optimal solutions. Every year thousands of nature-inspired optimization algorithms (NIOA) are coming into the state of the art of NIOA. So to give them a chance to survive, improving their efficiency and robustness is necessary.

The prairie dog optimization algorithm (PDO) [11] is a new natural-inspired population-based meta-heuristic algorithm. PDO is first introduced by Ezugwu et al. in 2022. In PDO, prairie dog movements of four types are considered to execute the exploration and exploitation process of optimization. The foraging and burrow-building actions of prairie dogs are utilized for exploring the

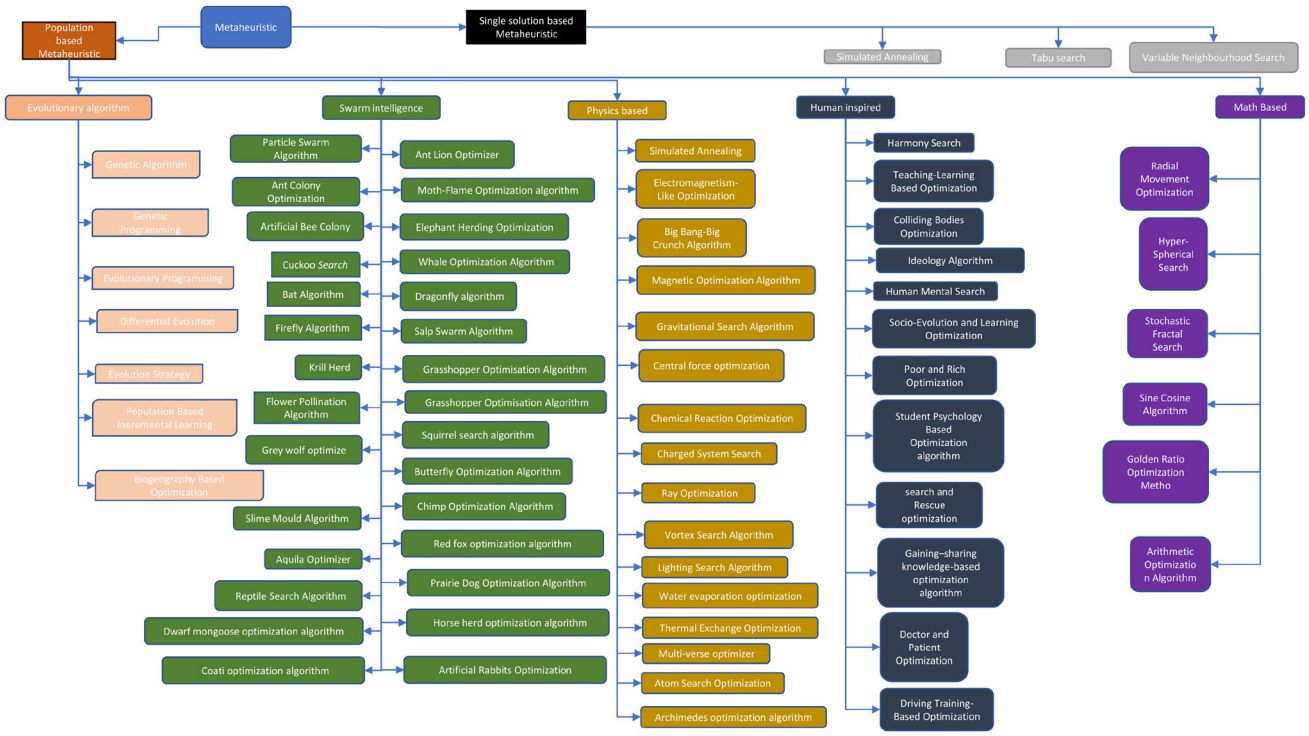

Fig. 1 Classification of meta-heuristics

decision variables in the search space. The exploitation process is executed through the transmission sign of the language of prairie dogs. It has the advantages of the most efficient Lévy flight random walk for producing the new position of prairie dogs, a unique updating process with special attributes like digging strength and predator effect, effective division of labours etc. Based on the results of the original PDO experiment and assessments of alternative algorithms, some issues have been raised. Premature convergence frequently affects the PDO algorithm. Its runtime is long due to the cumulative sum in a random walk. There is an imbalance between exploration and exploitation processes. It faces challenges due to the lack of diversity in its population. It needs modifications for enhancement of the overall performance of the algorithm. So we have proposed an improved design of PDO in this current thesis.

The opposition-based learning (OBL) strategy was first introduced by Tizhoosh [44].

It is a powerful learning strategy which can enhance the potential of the searching power of the MA or other optimization algorithms [23]. The OBL strategy's concept is illustrated in Fig. 2. In literature, many algorithms have been improved in terms of overall performance by using the concept of OBL [10, 18, 34]. OBL is used to increase the MAs' accuracy and convergence rate. Generally, MAs initialize the starting population randomly or on the basis of search domain knowledge. But when this type of knowledge is missing, The ideal solution cannot be reached by MAs. In this situation, OBL can help to overcome this problem by searching opposite direction to the recent solution in the solution search space. It strengthens the exploitation space. There are many different formulations of OBL in the literature. Types of OBL are type-I opposition, type-I quasi-opposition, type-I super-opposition, type-II opposition, generalized OBL, quasi-reflection OBL, centre-based sampling, and OBL using current optimum [37]. The majority of OBL-based optimization approaches described in the literature aim at a potential opposite point using deterministic methods. To improve the exploitation process, it might not be enough. In paper [47], the author presented a dynamic opposite learning (DOL)-based version of the TLBO algorithm whose performance was improved than the original algorithm in terms of convergence rate, solution quality and balance between exploration and exploitation. Later the authors of these papers [3, 9, 38] also utilized the concept of DOL for enhancing the performance of MAs like AO, MFO algorithm, WOA etc. In our current study, we have introduced a version of the DOL strategy which enhances the diversity in the population of our proposed algorithm (E-PDO) and reduces the chances of falling into a locally optimal solution.

The novelty and contribution of this work is the improvement of the new optimization method PDO. An improved design of PDO is proposed with a modified random flight path and a dynamic opposite learning strategy. Various steps of the proposed refinement of PDO are described and modelled mathematically. A set of 33 objective functions, including uni-modal and multi-modal, was used to test the effectiveness of the proposed enhanced PDO. Furthermore, the performance of PDO is compared with known optimization algorithms: GWO [30], MFO [25], ALO [26], WOA [29], DA [27], SCA [28], RSA [1].

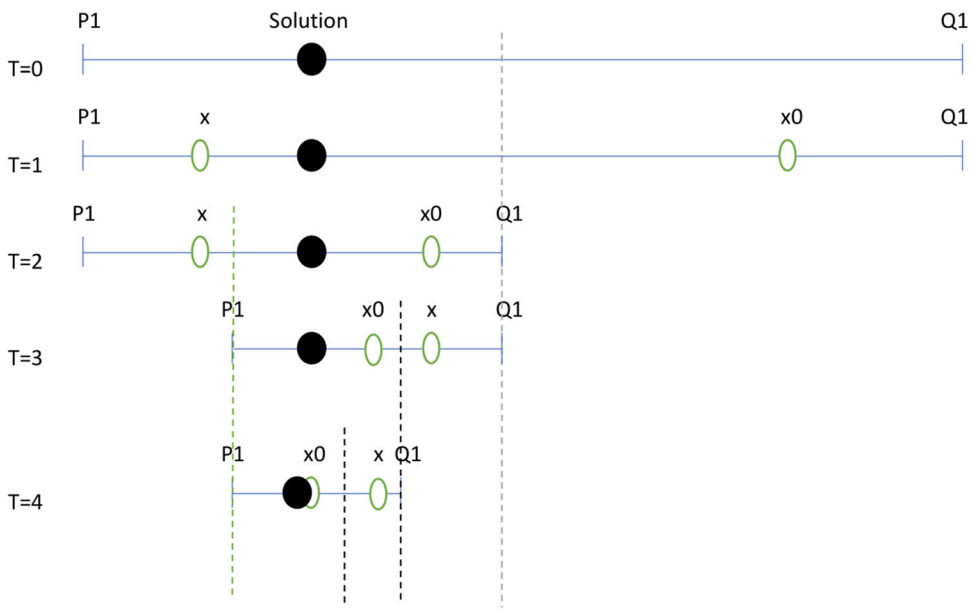

Fig. 2 Determining the optimal estimates x and $x0$ for a one-dimensional function with initial boundary [$P1$, $Q1$] using the concept of OBL strategy where T is iteration count

The contributions of the present study are summarized as follows:

- A modified Lévy flight is proposed to perform a random search for the improved version of the standard PDO algorithm.
- The existing PDO algorithm incorporates a DOL strategy. The DOL strategy serves two key functions: effective population initialization and another efficient generation jump. This will help improve the ability to diversify and enhance the new PDO version, the E-PDO algorithm.
- The performance of the proposed enhanced PDO algorithm is calculated by examining a set of 33 benchmark functions consisting of uni-modal, multi-modal, uni-modal fixed-dimensional and multi-modal fixed-dimensional. Also, the proposed E-PDO algorithm was employed to solve two engineering design problems.
- A statistical mean rank test method and the novel statistical multi-criteria analysis method MARCOS are used to evaluate how well the suggested method performed.

1.1 Paper structure

The rest of the paper is structured as follows. An inspiring synopsis of the original PDO is presented in Sect. 2. A complete mathematical implementation is also discussed in Sect. 2. Section 3 describes the Lévy flight concept and DOL strategy, including a mathematical template. The proposed E-PDO algorithm is explained in Sect. 4. The experimental setup is presented in Sect. 5. Additionally, Sect. 5 includes test work, results, and discussion. Section 5 also presents a statistical analysis of the outcomes of the experiment and the use of E-PDO to solve constrained optimization issues is demonstrated in Sect. 5.6. Section 6 provides conclusions on the current work.

Fig. 3 Burrow entrance of a colony of prairie dogs

2 Prairie dog optimization algorithm

Prairie dogs are ground squirrels [Fig. 3]. Prairie dogs are unique in their foraging and communication abilities. The prairie dog optimization (PDO) algorithm is based primarily on foraging movements, burrowing activities, communication skills, and predator prevention activities of the same coterie members of prairie dogs [11].

2.1 Algorithm assumptions

The assumptions of the prairie dog optimization (PDO) algorithm can be summarized as follows:

- The algorithm assumes that all prairie dogs within a colony are identical regarding their capabilities and behaviours. This assumption allows for a standardized approach to modelling the optimization process.
- Foraging for food and building burrows are the main activities of prairie dogs. These actions are essential for their survival and are mimicked in the optimization algorithm.
- Prairie dogs build their burrows around food sources. This relationship is incorporated into the algorithm, where the solution space exploration is initiated based on the location of the food source.
- Different coteries (subgroups) have their boundary limits within a prairie dog colony. Each coterie is responsible for foraging and burrowing activities within its designated boundary. This helps distribute the optimization process and promotes exploration within different regions.
- The algorithm considers the importance of the latest Burrow position in enhancing coterie operations. This implies that the previous actions and experiences of the prairie dogs influence their future exploration and exploitation strategies.
- Prairie dogs communicate using distinct sounds that convey specific information. These sounds can relate to various scenarios, such as food availability and predator threats. The ability to communicate plays a crucial role in the prairie dogs' survival and adaptation to predators.
- Prairie dogs exhibit different responses to various predators. For example, they may hide in response to a message indicating the presence of a predator within the range of hawks while continuing to observe from their burrows in other situations. This adaptability to different methods of predation is incorporated into the algorithm.

- Specific sounds emitted by prairie dogs trigger movement and response patterns within different coteries. These cycles of movement and response contribute to the exploration and exploitation phases of the PDO algorithm.

By incorporating these assumptions from the behaviour and characteristics of prairie dogs, the PDO algorithm attempts to mimic their foraging and burrowing behaviour to solve optimization problems.

2.2 Implementations of basic PDO

The initialization, fitness function evaluation, exploration, and exploitation of the fundamental PDO algorithm are all covered in this section's mathematical formulation. Prairie dog populations are search agents, and prairie dog locations are possible solutions to the algorithm. Even in the case of PDO, random initialization is considered like other PBM algorithms. All necessary indexes and parameters are specified in Table 1.

2.2.1 Initialization

Suppose N is a prairie dog of a coterie. Each Prairie Dog belongs to M coterie. The search space is filled with N prairie dogs at random within the specified boundary [LB UB] in this stage, where LB signifies the least value of the boundary of the associated variable of the test problem, and UB signifies the highest value of the boundary.

The positions of all coteries within the colony are represented by a coterie matrix (C) in Eq. (1):

$$C = \begin{bmatrix} C_{1,1} & C_{1,2} & \cdots & C_{1,d} \\ C_{2,1} & C_{2,2} & \cdots & C_{2,d} \\ \vdots & \vdots & C_{i,j} & \vdots \\ C_{M-1,1} & C_{M-1,2} & \cdots & C_{M-1,d} \\ C_{M,1} & C_{M,2} & \cdots & C_{M,d} \end{bmatrix} \tag{1}$$

$C_{i,j}$ signifies the i^{th} coterie of j^{th} dimension within the colony.

The locations of all the prairie dogs in a coterie are provided by the Prairie matrix (P) in the Eq. (2):

$$P = \begin{bmatrix} P_{1,1} & P_{1,2} & \cdots & P_{1,d} \\ P_{2,1} & P_{2,2} & \cdots & P_{2,d} \\ \vdots & \vdots & P_{i,j} & \vdots \\ P_{N-1,1} & P_{N-1,2} & \cdots & P_{N-1,d} \\ P_{N,1} & P_{N,2} & \cdots & P_{N,d} \end{bmatrix} \tag{2}$$

$P_{i,j}$ signifies the i^{th} prairie dog at j^{th} location in a coterie.

Equations (3) and (4) are used to assign each coterie and prairie dog location.

$$\begin{aligned} C_{i,j} =& rand(0,1) \times (UB_j - LB_j) \\ &+ LB_j \quad i = 1,2,\cdots,N;\ j = 1,2,\cdots,M; \end{aligned} \tag{3}$$

$$\begin{aligned} P_{i,j} =& rand(0,1) \times (ub_j - lb_j) \\ &+ lb_j \quad i = 1,2,\cdots,N;\ j = 1,2,\cdots,M; \end{aligned} \tag{4}$$

where $N \leq M$, $ub_j = \frac{UB_j}{M}$ and $lb_j = \frac{LB_j}{M}$ and using a uniform distribution, the value of rand(0,1) falls between 0 and 1.

2.2.2 Evaluation function assessment

The Evaluation function values for each prairie dog location in a coterie are enumerated by providing the optimal

Table 1 Nomenclature for proposed algorithm E-PDO

Symbols	Name	Symbols	Name
M or D	Number of coteries (problem dimension)	N	Number of prairie dogs (number of variables)
P	Position or location of Prairie Dogs	i	Index of starting position of Prairie Dogs
j	Index of target position of Prairie Dogs	$rand$	Random numbers from uniform distribution
$randn$	Random numbers from normal distribution	$randi$	Random integers from uniform distribution
t or $iter$	Index for iterations	T	Maximum number of iterations
EC^{Best}	Effect of the current obtained best solution	Ds	Coterie's digging strength
Pe	Predator effect	LF	Lévy distribution (random walk based on Lévy distribution)
$Bool$	− 1 or 1, according to odd or even current iteration	CP	Collective impact of the colony's Prairie Dogs
RP	Position of the randomly chosen solution	τ	Individual prairie dog (P) position difference
ϵ	Food source quality	ϵA	Food source alarm
β	Power law index	w	Weight factor of step length of Lévy flight
ω	Weight factor for DOL generation jumping		

solution to the fitness function ($fit(P)$) in the (5) equation. At each iteration, fitness values are calculated for all prairie dog locations and stored as (n × 1) matrix:

$$fit(P) = \begin{bmatrix} fit_1([P_{1,1} & P_{1,2} & \cdots & P_{1,d}]) \\ fit_2([P_{2,1} & P_{2,2} & \cdots & P_{2,d}]) \\ & \vdots & & \\ fit_N([P_{N,1} & P_{N,2} & \cdots & P_{N,d}]) \end{bmatrix} \quad (5)$$

Each evaluation function's fitness value based on the optimal location of the prairie dog reflects the food source quality, its ability to make new burrows, and its successful response to predators.

2.2.3 Exploration

In PDO, The location of the prairie dog is a probable decision, and the best decision at each stage is considered the best forager as well as the best response to a predator alert. Exploration and use of the PDO algorithm are accomplished through four strategies. These four strategies are applied in four iteration steps. Two strategies for exploration are applied in between $0 < t < \frac{T}{4}$ and $\frac{T}{4} < t < \frac{T}{2}$ and other two for exploitation are used in between $\frac{T}{2} < t < \frac{3T}{4}$ and $\frac{3T}{4} < t < T$.

Burrow building is important for prairie dogs to protect themselves from environmental threats and predators. When their food source runs out, they start looking for new food sources and build new burrows around them. The initial step in the exploration phase is the movement of coterie members from the ward in search of new food sources. The random walk can mimic the behaviour of a prairie dog. This movement with long jumps ensures the effectiveness of the search for food sources. When a food source is found, it makes a unique sound to alert other members. Then they access food source quality, select the best foraging, and build new burrows based on food source quality. The position of the search update is given by the (6) equation during the algorithm's search phase.

$$P^{new}_{(i+1,j+1)} = P^{Best}_{(1,j)} - EC^{Best}_{(i,j)} \times \epsilon A - CP_{(i,j)} \times LF \quad \forall\ t < \frac{T}{4} \quad (6)$$

The formula (7) yields the random collective impact of all colony's prairie dogs as CP.

$$CP_{(i,j)} = rand \times \frac{P^{Best}_{(i,j)} - P_{(randi([1\ N]),j)}}{P^{Best}_{(i,j)} + \epsilon} \quad (7)$$

EC^{Best} measures the effectiveness of the currently obtained optimal solution, as shown in the formula (8).

$$EC^{Best}_{(i,j)} = P^{Best}_{(i,j)} \times \left(\tau + \frac{P_{(i,j)} - mean(P_{(N,M)})}{P^{Best}_{(i,j)} \times (UB_j - LB_j) + \epsilon} \right) \quad (8)$$

A second strategy is to evaluate the quality of previous food sources as well as the intensity of digging. The digging strength is intended to decrease with further iterations, and new burrows are built on top of it. This circumstance aids in limiting the potential number of burrows that can form. The formula (9) provides the location update for building burrow.

$$P^{new}_{(i+1,j+1)} = P^{Best}_{(1,j)} \times RP \times Ds \times LF \quad \forall\ \frac{T}{4} \le t < \frac{T}{2} \quad (9)$$

The coterie's digging intensity is denoted by the variable Ds, which varies on the quality of the food supply and has random values as defined by the Eq. (10).

$$Digging\ strength,\ \ Ds = 1.5 \times randn \times \left(1 - \frac{t}{T}\right)^{\frac{2t}{T}} \times Bool \quad (10)$$

2.2.4 Exploitation

Prairie dogs are capable of producing special types of sounds for, unlike situations. They can able to differentiate those special sounds for varying from the quality of food sources to the dangers of predators. Each prairie dog has an equal response ability to take care of these scenarios. Their communication signals assist them to handle predator-induced fear and also help them to fulfil their nutritional requirements. Once the signal reports that the food source is of good quality and safe, it converges on that signal source to meet nutritional requirements. And when a communication signal identifies a predator, prairie dogs hide in the way of predators while other dogs watch from their burrows.

Two specific behaviours, food alarm and anti-predation alarm cause prairie dogs to converge on specific places (promising if PDO is implemented), and further searches (exploitation) are conducted to find a better or near-optimal solution. The purpose of the exploit mechanism used in PDO is to focus the search on promising areas identified during the exploration phase. This phase is implemented according to the following equations

$$P^{new}_{(i+1,j+1)} = P_{(1,j)}Best - EC^{Best}_{(i,j)} \times \epsilon - CP_{(i,j)} \times rand \quad \forall\ \frac{T}{2} \le t < \frac{3T}{4} \quad (11)$$

$$P^{new}_{(i+1,j+1)} = P^{Best}_{(1,j)} \times Pe \times rand \quad \forall\ \frac{3T}{4} \le t < T \quad (12)$$

Where predator effect (Pe) is defined as

$$Pe = 1.5 \times \left(1 - \frac{t}{T}\right)^{\frac{2t}{T}} \times Bool \tag{13}$$

The PDO algorithm benefits from the most effective Lévy flight random walk for generating the new position of prairie dogs, a particular updating process with distinctive characteristics like digging intensity and predator effect, and an efficient division of labour. There are some concerns based on original PDO experiment findings and comparisons to other algorithms. It is true to say that the PDO algorithm frequently experiences premature convergence. The cumulative sum of a random walk causes its lengthy run-time. The processes of exploration and exploitation are not balanced. Its population is not diverse, which presents problems. Therefore, those challenges have been a focus of our recent research.

3 Foundational framework for advancing prairie dog optimization (PDO) algorithm

3.1 Concepts of Lévy flight

A path consisting of a succession of randomly chosen steps in a mathematical space like integers is referred to as a random walk. This mathematical concept is also known as a stochastic process or random process. This is the common basis for the perturbations of the solution [51]. It is the movement of a particle, which can jump to a neighbouring node in the search space, starting at 0 and taking $+1$ or -1 steps with a given probability each time it moves. Using a normal distribution with a mean of zero and a variance of one, we can calculate the step size S_i of the random walk.

$$S_i \sim \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{14}$$

The current solution is represented here by x and $\sim$ is used to denote the drawing of random numbers according to the distribution. Equation 15 gives the total of successive random walking steps (RW_n).

$$RW_n = \sum_{i=1}^{n} S_i = S_1 + S_2 + \cdots + S_n \tag{15}$$

This is the same as given in the following Eq. (16).

$$RW_n = RW_{n-1} + S_n \tag{16}$$

The new state or transition phase depends only on the existing state (RW_{n-1}) and transition phase (S_n). This random walk is also called the Brownian motion or diffusion process. The search performance of the nature-inspired optimization algorithm (NIOA) depends on a probability distribution of the forager's flight lengths that can only find target sites in the search space [50].

Random walks provide more randomness in both NIOA's exploration and exploitation processes. The powerful model for characterizing non-directed animal motion was established on the conception of Brownian motion for several years [15, 19, 45]. In this configuration, the flight path of an animal in space is assumed to consist of a series of different random directional motion steps produced from a Gaussian distribution [36]. The Lévy flight model is also used to analyse animal movements. Shlesinger et al. [40] first suggested that the movement styles of some organisms could exhibit Lévy flight characteristics. To be precise, these ought to be motion styles of the Lévy Walk. This is because it is a continuous movement (usually of constant speed) rather than individual jumps. Lévy flight involves linear motion in random directions.

Lévy flights are capable of efficient resource searches randomly in uncertain environments [50]. According to simple power law $L(x) \sim |x|^{-1-\beta}$ where $0 < \beta \leq 2$ is an index. It is possible to express the Lévy probability distribution as

$$L(x, \mu, \gamma) = \frac{\sqrt{\gamma/(2 * \pi)}}{(x - \mu)^{3/2}} e^{-\frac{\gamma}{2(x-\mu)}}, \; x > \mu \tag{17}$$

where $\mu > 0$ head the location of the search path and γ is scale factor.

The following is a definition of the Lévy probability function's special case:

$$L(x, \mu, \gamma) \approx \frac{\sqrt{\gamma/(2 * \pi)}}{(x)^{3/2}}, \; x \to \infty \tag{18}$$

For the more general case of exponential β we can define the Lévy distribution using the integral

$$L(x) = \frac{1}{\pi} \int_0^{\infty} \cos(kx) * e^{-\alpha |k|^{\beta}} dk, \; (0 < \beta \leq 2) \tag{19}$$

where $\alpha > 0$ is scale parameter. If in Eq. (19), $\beta = 1$ then it follows a Cauchy distribution, and when $\beta = 2$ Eq. (19) obeys a normal distribution. This inverse integral can only be evaluated for large x. We have also

$$L(x) \to \frac{\alpha\beta\Gamma(\beta)\sin(\frac{\pi\beta}{2})}{\pi |x|^{1+\beta}}, \; x \to \infty \tag{20}$$

where $\Gamma(\beta)$ = Gamma function.

According to the aim of implementation, generating random numbers in Lévy flights includes a pair of steps [49]. These include choosing a random direction and creating steps using the selected Lévy distribution. The uniform distribution should be used to generate directions, but step creation is quite challenging. There are so many measures available to reach this; however, one of the most effective and straightforward techniques for the symmetric

stable Lévy distribution is the Mantegna algorithm [24]. Symmetry signifies both positive and negative actions.

The following equation can be used to calculate the Mantegna algorithm's step length (S).

$$S = \frac{u}{|v|^{1/\beta}} \tag{21}$$

where u and v are taken from the continuous probability distributions known as normal distributions.

$$u \sim N(0, \sigma_u^2); \; v \sim N(0, \sigma_v^2) \tag{22}$$

$$where, \;\; \sigma_u = \left[\frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}}\right]^{\frac{1}{\beta}} \; and \; \sigma_v = 1. \tag{23}$$

In a two-dimensional plane, Fig. 4 shows an illustration of a Lévy flight. Figure 4 illustrates the divergence in motion, which is the most crucial aspect of Lévy flight. In some circumstances, it may considerably improve the search algorithm's efficiency.

3.2 Concepts of dynamic opposite learning strategy (DOL)

3.2.1 Opposition-based learning (OBL)

To improve the search power of algorithms, the OBL (opposition-based learning) strategy is one of the best options for efficient learning. The optimization process of meta-heuristics is improved by this feature, which speeds up convergence. The conceptual idea of two strategies, namely the opposite number and opposite point, is identified in the following way.

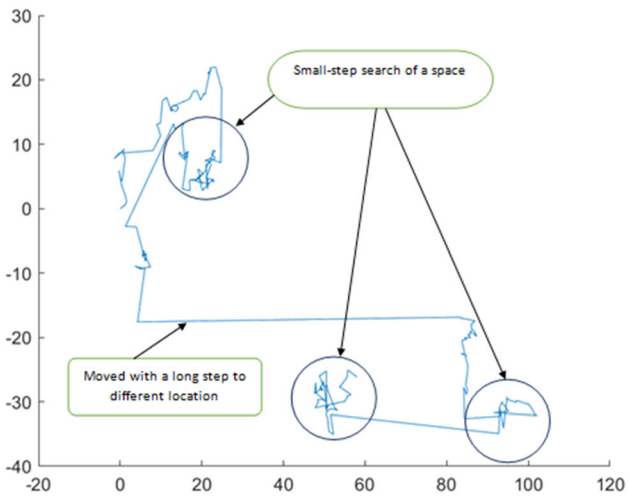

Fig. 4 A trajectory of Lévy flight in a two-dimensional plane

Opposite number. Let X be in the interval $[a, b]$ ($X \in [a, b]$). Then, the opposite number X^O can be defined as eq. (24)

$$X^O = a + b - X \tag{24}$$

where the search region's bottom and higher bounds are a and b, respectively.

Opposite point. Practically, in the application field, X can be a point of a multi-dimensional problem. Let X is a point in D-dimension, $X_j, \cdots, X_D \in [a_j, b_j]$. A multidimensional opposite point is characterized as

$$X_j^O = a_j + b_j - X_j \tag{25}$$

3.2.2 Dynamic opposite learning strategy (DOL)

Considering opposite numbers, the OBL strategy can provide a solution close to the global optimum. The search area between the central and opposite places is expanded using quasi-opposite-based learning (QOBL), and several quasi-opposites are discovered. However, the quasi-reflection numbers are selected from the search space between the current solution location and the average position. In such circumstances, quasi-reflection-based learning (QRBL) can be generated. OBL, QOBL, and QRBL are all affected by the same issue. The search space will converge to local optimum if there is a local optimum between the current solution and its opposite solution. A dynamic oppositional learning (DOL) strategy helps to circumvent this problem. This increases the chances of converging on a global solution. The DOL approach was initially put forth by Xu et al. [47] for the TLBO algorithm.

Inspiration from QOBL and QRBL, there is a higher possibility of dynamical expansion of the search space of OBL to get the closer optimum solution. This is displayed in the Fig. 5 where search space is in between a and b, X is the current position number, X^O is the opposite number, and $X^S = X + rand * (X^O - X)$, $rand \in [0, 1]$ is the same number as before in the new location which is symmetric. Although using this technique increases the likelihood of obtaining the global solution, it is certain that the search space will tend towards the local optimal solution position that lies between the current location and its opposite number. In this regard, we are considering a novel concept called DOL strategy to avoid the local optimum solutions. For this, we need a random opposite number for corresponding X^O. The random opposite number is defined as

$$X^{RO} = rand * X^O, \;\; rand \in [0, 1] \tag{26}$$

X^O is replaced by X^{RO}, which broadens the search area and converts the symmetric search area into an asymmetric

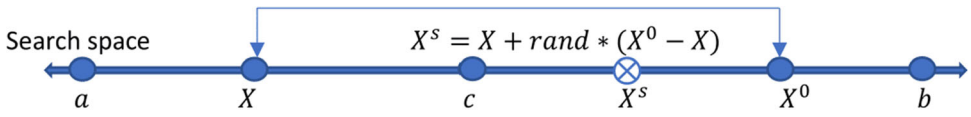

Fig. 5 The symmetric space of DOL strategy

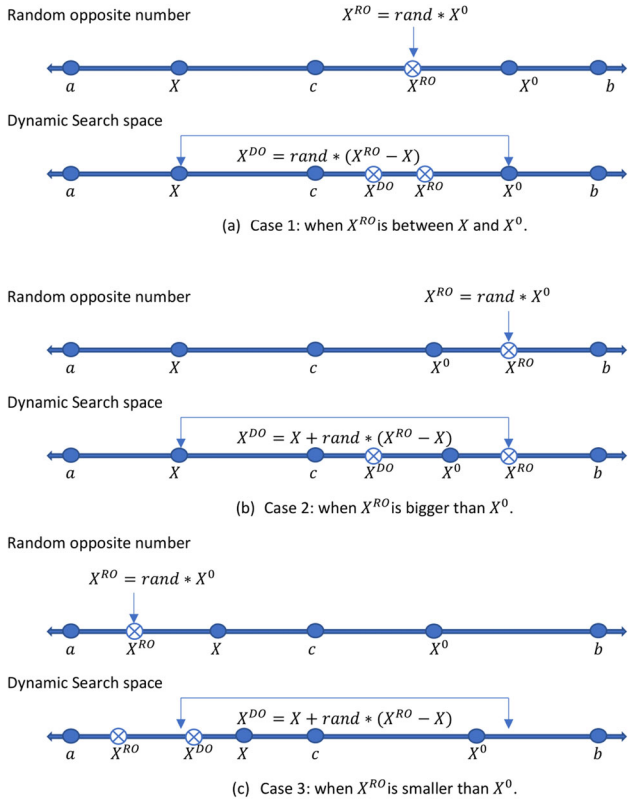

Fig. 6 The asymmetric space of DOL strategy

search area. This is shown in Figs. 6 and 7. This helps create a dynamic search space where the algorithm can avoid reaching a local optimum.

3.3 Mathematical templates for DOL

The DOL strategy can be implemented using the following mathematical model.

Dynamic opposite number The following definition 27 applies to the dynamic opposite number (X^{DO}).

$$X^{DO} = X + \omega * rand * (X^{RO} - X) \tag{27}$$

Where a and b denote the lower and upper boundaries of the value of X, and X is a real number, $X \in [a, b]$. *rand* has the value (0, 1) at random. ω is the weighting factor.

Dynamic opposite point Taking into account the D-dimensional space $(X_1, X_2, \cdots, X_D)$ with $X_j, \cdots, X_D \in [a_j, b_j]$, the dynamic opposite point is defined by Eq. 28, where the lower and upper limits of the current search space, respectively, are a_j and b_j.

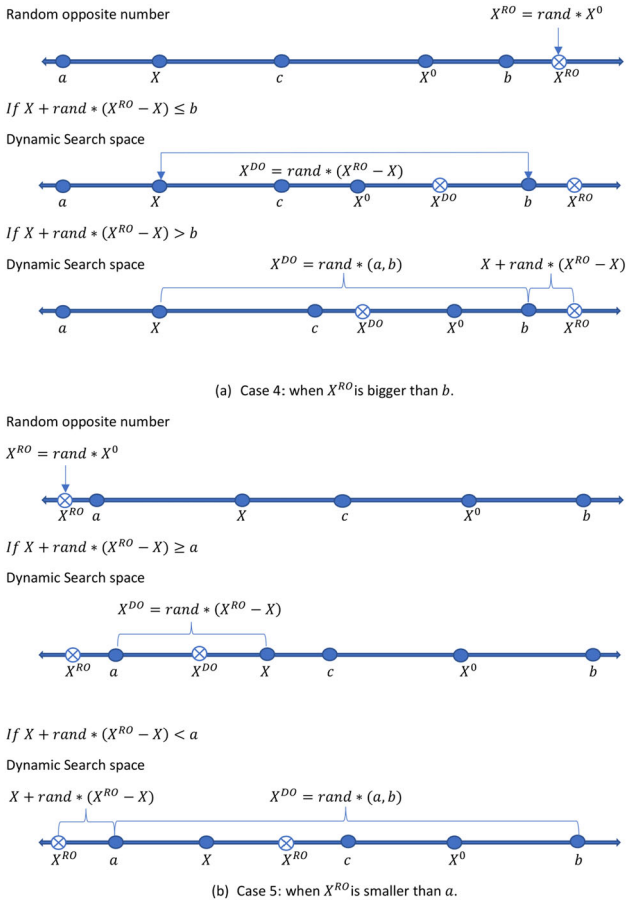

Fig. 7 The asymmetric space of DOL strategy

$$X_j^{DO} = X_j + \omega * rand * (X_j^{RO} - X) \tag{28}$$

DOL-based optimization In D-dimensional space, consider the points $(X_1, X_2, \cdots, X_D)$ and $X_j, \cdots, X_D \in [a_j, b_j]$, where the lower and upper boundaries of the variable X_j are, respectively, a_j and b_j. The dynamic opposite point $X^{DO} = (X_1^{DO}, X_2^{DO}, \cdots, X_D^{DO})$ is determined by Eq. (28) and the update is performed by updating X according to Eq. (28). This step validates X^{DO} value. X^{DO} is acceptable if compared to X, X^{DO} has a better fitness value. Otherwise, X value remains the same.

4 Proposed E-PDO

Exploration and exploitation are the two main phenomena of any optimization algorithm. Exploration is a broader search space perspective that gives the algorithm control

over the entire search space during a search operation. Conversely, exploitation means finding solutions in the local search space. Another important aspect is the stability between the above two operations. Any optimization calculation algorithm should follow these three aspects. PDO is a novel population-based metaheuristic algorithm. It is easy to see that the PDO algorithm undergoes premature convergence. That is, it has a higher tendency to reach a local optimum. Poor quality of population diversity and low accuracy in generating optimal solutions. These issues can result in a poor balance between exploration and exploitation.

To overcome the above difficulties, we proposed an improved PDO algorithm (called E-PDO or enhanced-PDO) by integrating the DOL strategy. Also introduced is a modified random walk that randomizes the locations of prairie dogs. In the subsections below, all details of these modifications of the proposed PDO are described. The introduction of modified Lévy flight and DOL-based strategies for PDO is discussed in detail in Sects. 5.1 and 5.2, respectively.

4.1 Improved random walk

Lévy flight is a highly efficient random walk for nature-inspired optimization algorithms. For the algorithm here, we consider a new improved version of Lévy flight to generate new prairie dog locations in the problem search space. The new flight is called m-Lévy (modified Lévy flight). We have proposed the modified step length S using the concept of the Lévy flight (in Sect. 4.1). It is given by

$$S = \frac{w \times u \times \sigma_u}{|v|^{1/\beta}} \tag{29}$$

where w, is weight factor and u and v are drawn from normal distributions

$$u \sim N(0,1);\ v \sim N(0,1) \tag{30}$$

$$where,\ \sigma_u = \frac{\Gamma(1+\beta) \times \sin(\frac{\pi\beta}{2})}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\frac{\beta-1}{2}}}; \tag{31}$$

Here, we consider the value of $w = 0.001$ for our proposed e-PDO.

Figure 8 shows an illustration of m-Lévy flight in a two-dimensional plane. The steps are made up of numerous small steps and a few long steps because of the Lévy distribution. In some circumstances, these large steps may greatly improve e-PDO's search efficiency as compared to other MAs.

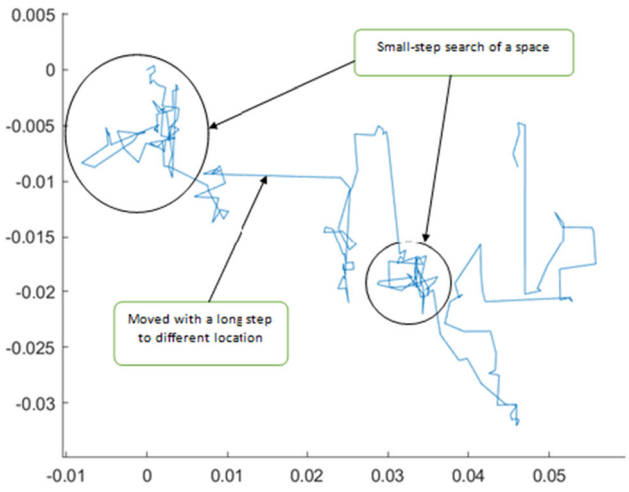

Fig. 8 A trajectory of m-Lévy flight in a two-dimensional plane

4.2 DOL-based strategies for PDO

To prevent premature convergence when solving complex real-world optimization problems, the proposed DOL-based idea is built in PDO to speed up convergence. DOL increases population diversity by avoiding falling into a local optimum. For the proposed algorithm, the DOL-based strategy is composed of two stages: initialization of the population using DOL and generation jump with DOL.

4.2.1 Improved population initialization using DOL strategy

We can consider P to be the initial population which is generated randomly and P^{DOL} to be the population generated by the DOL initial population generation strategy. For each prairie dog $i = 1, 2, \cdots, N$ and $j = 1, 2, \cdots, D$ within maximum iteration over all iterations, the DOL population initialization method is defined as:

$$P_{i,j}^{DOL} = P_{i,j} + r1_i \times \left(r2_i * \left(UB_j + LB_j - P_{i,j}\right) - P_{i,j}\right) \tag{32}$$

where the current search space's upper and lower limits are UB_j and LB_j. $Rand1_i$, and $Rand2_i$ are random numbers. Weight ω is set to 1.

$$P_{i,j}^{DOL} = rand(LB_j, UB_j)\ \ if\ P_{i,j}^{DOL} < LB_j || P_{i,j}^{DOL} > UB_j \tag{33}$$

$P_{i,j}^{DOL}$ must be in the range $[LB_j, UB_j]$. In the initialization phase, the best ones from among $(P \bigcup P^{DOL})$ are selected.

Table 2 Algorithm E-PDO

Algorithm e-PDO: Pseudocode of proposed E-PDO algorithm

1. Start
2. **Input:** Objective function $F(P)$, $P = (P_1, P_2, \cdots, P_D)$, Number of prairie dogs (N), number of coteries (D), individual prairie dogs difference (τ), food source alarm on the basis of food source quality (ϵA), Maximum number of iterations (T).
3. Generate the initial population of prairie dogs (P) using Eq. (4)
4. Initialize the OBest (old fitness value) and the CBest (New fitness value) as ϕ
5. Update individual P according to DOL strategy using the following equation,
6. for i = 1:N N = no. of population
7. for j = 1:D D = dimension
8. $P_{i,j}^{DOL} = P_{i,j} + \omega * Rand1_i * (Rand2_i * (UB_j + LBj - P_{i,j}) - P_{i,j})$
9. check boundaries;
10. end for
11. end for
12. Choose N best P from $(P \bigcup P^{DOL})$;
13. while $t < T + 1$ do
14. initiate *bool*, a stochastic parameter
15. for i = 1:N N = no. of population
16. for j = 1:D D = dimension
17. Determine each P's fitness
18. Discover the best effective solution so far
19. Update digging Strength and predator effect using equations (10) and (13).
20. Update cumulative effect of all P using equation (7)
21. If $(t < T/4)$, then use equation (6) (**foraging activities**) (Exploration)
22. Else if $(T/4 \leq t < T/2)$ then use equation (9) (**burrow building activities**) (Exploration)
23. Else if $(T/2 \leq t < 3T/4)$ then use equation (11) (**food alarm**) (Exploitation)
24. Else use equation (12) (**anti-predation alarm**) (Exploitation)
25. end if
26. end for
27. According to DOL approach, update P
28. $(P_{new})_{i,j}^{DOL} = (P_{new})_{i,j} + \omega \times Rand3_i \times (Rand4_i \times (UB_j + LB_j - (P_{new})_{i,j}) - (P_{new})_j)$
29. check boundaries;
30. Determine each (P_{new})'s fitness
31. Update OBest and CBest
32. end for
33. end while
34. return Best solution
35. end

4.2.2 Improved generation jumping using DOL strategy

We can update the population using DOL Strategy taking into account the jump rate (Jr). If the selection probability at any iteration t is less than Jr, the DOL-generation transition procedure can be carried out as

$$(P_{new})_{i,j}^{DOL} = (P_{new})_{i,j} + \omega \times Rand3_i \times (Rand4_i \times (UB_j + LB_j - (P_{new})_{i,j}) - (P_{new})_j) \quad (34)$$

$Rand3_i$, and $Rand4_i$ are random numbers.To make the DOL efficient, we need to check the range in the same format as Eq. 33.

$$LB_j = min(P_{i,j}),\ UB_j = max(P_{i,j})$$

In the DOL generation jumping phase, to discover the optimum solution, we can choose the best out of $(P \bigcup P^{DOL})$.

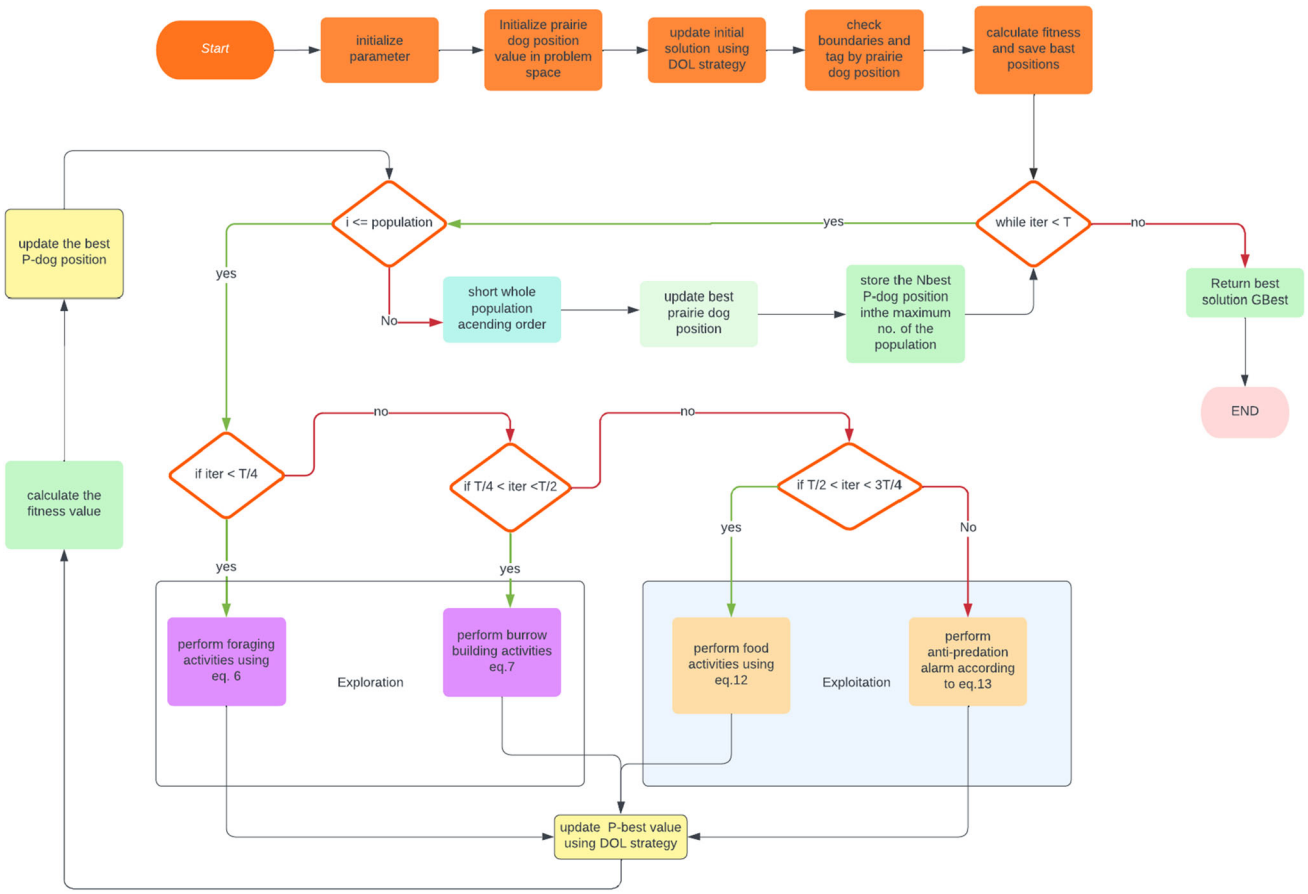

Fig. 9 Flow diagram of the E-PDO algorithm

Table 3 Uni-modal benchmark functions

Function	Description	Dimensions	Range	f_{min}
Sphere (F1)	$f(x) = \sum_{i=1}^{n} x_i^2$	30	[− 100, 100]	0
Schwefel 2.22 (F2)	$f(x) = \sum_{i=0}^{n} \lvert x_i \rvert + \prod_{i=0}^{n} \lvert x_i \rvert$	30	[− 10, 10]	0
Schwefel 1.2 (F3)	$f(x) = \sum_{i=1}^{d} (\sum_{j=1}^{i} x_i)^2$	30	[− 100, 100]	0
Schwefel 2.21 (F4)	$f(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	[− 100, 100]	0
Rosenbrock (F5)	$f(x) = \sum_{i=1}^{n-1} [100(x_i^2 - x_{i+1})^2 + (1 - x_i)^2]$	30	[− 30, 30]	0
Step (F6)	$f(x) = \sum_{i=1}^{n} (\lvert x_i + 0.5 \rvert)^2$	30	[− 100, 100]	0
Quartic (F7)	$f(x) = \sum_{i=1}^{n} i x_i^4 + random[0, 1)$	30	[− 1.28, 1.28]	0

4.3 E-PDO algorithm steps

A new PDO variant called enhanced PDO (E-PDO) has been formulated with the addition of modified random walk, improved population initialization and generation jumping using DOL Strategy. To visualize a specific algorithm, the E-PDO algorithm's steps are provided in Algorithm E-PDO in Table 2. The complete process of the E-PDO algorithm is included in the flow diagram in Fig. 9.

4.4 E-PDOUW

Here, we also consider E-PDO with m-Lévy flight step length with a weight value of $(w = 1)$. We call this E-PDOUW.

4.5 PDOL

The combination of PDO algorithm, DOL strategy and old Lévy flight (based on Eqs. 21 to 23) is examined as PDOL.

Table 4 Multi-modal benchmark functions

Function	Description	Dimensions	Range	f_{min}
Schwefel (F8)	$f(x) = \sum_{i=1}^{n}(-x_i \sin(\sqrt{\lvert x_i \rvert}))$	30	[− 500, 500]	$-418.9829 \times n$
Rastrigin (F9)	$f(x) = \sum_{i=1}^{n}[x_i^2 - 10\cos(2\pi x_i) + 10]$	30	[− 5.12, 5.12]	0
Ackley (F10)	$f(x) = -20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	[− 32, 32]	0
Griewank (F11)	$f(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})$	30	[− 600, 600]	0
Penalized (F12)	$f(x) = \frac{\pi}{n}\{10\sin(\pi y_i)\} + \sum_{i=1}^{n-1}(y_i - 1)^2$ $[1 + 10\sin^2(\pi y_{i+1}) + \sum_{i=1}^{n} u(x_i, 10, 100, 4)]$ *where* $y_i = 1 + \frac{x_i+1}{4}$, $u(x_i, a, k, m) = \begin{cases} \{K(x_i - a)^m, & if\ x_i > a \\ 0, & if -a \le x_i \ge a \\ K(-x_i - a)^m & if -a \le x_i \end{cases}$	30	[− 50, 50]	0
Penalized 2 (F13)	$f(x) = 0.1(\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i - 1)^2$ $[1 + \sin^2(3\pi x_i + 1)](x_n - 1)^2$ $\{1 + \sin^2(2\pi x_n)\}) + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[− 50, 50]	0

4.6 The computational complexity of the E-PDO algorithm

Computational complexity is an important factor to consider when assessing an algorithm's performance. The complexity of the EPDO is determined according to the values of the number of prairie dogs (N), the dimensions (D) and the maximum number of iterations (T). The space complexity of the EPDO algorithm hinges on two key parameters: the number of prairie dogs (N) and the dimensions of the optimization problem (D). This space complexity measurement reflects the memory space requirements, particularly during initialization. Consequently, the expression for EPDO's space complexity is succinctly captured as $O(N \times D)$.

The time complexity is intricately influenced by several factors, including the population size (N), problem dimensions (D), the number of iterations (T), and the cost of function evaluations (C). Consequently, the time complexity (O(EPDO)) can be precisely articulated as the sum of three main components: O(Initialization), O(cost function evaluation), and O(Updating strategy). The complexity as a whole is O(EPDO) = O(Initialization) + $T\times$ (O(Fitness evaluation of all prairie dogs) + O(Generation updating process of all prairie dogs with new strategies)). Hence, the overall computational complexity of the EPDO is

$$O(EPDO) = O((N \times D) + (T \times N \times D \times 4) + (T \times N \times D)) \quad (35)$$

According to a convergence analysis, an algorithm that is combined with the DOL method achieves a fast convergence rate compared to other conventional algorithms. A DOL technique is used to improve the EPDO algorithm's ability to avoid local optima. The results of the runtime study show that the EPDO, when combined with the m-Lévy random walk and DOL approaches, can greatly improve computational efficiency.

According to a convergence analysis, an algorithm that is combined with the DOL method achieves a fast convergence rate compared to other conventional algorithms. A DOL technique is used to improve the e-PDO algorithm's ability to avoid local optima. The results of the runtime study show that the E-PDO, when combined with the m-Lévy random walk and DOL approaches, can greatly improve computational efficiency.

5 Experimental problems, results, and discussions

Various tests are conducted in this section to show how well the e-PDO works and verify the accuracy of solving the global optimization problem. We combined the simulation results of the proposed e-PDO and compared the simulation results of the original PDO with seven other meta-heuristics, including GWO, MFO, ALO, WOA, DA, SCA, and RSA for reference functions, including unimodal, multi-modal, and fixed-dimensional functions.

Table 5 Fixed-dimension multi-modal test functions

Function	Description	Dimensions	Range	f_{min}
Foxholes (F14)	$f(x) = \{(\frac{1}{500}\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}2(x_i-a_{ij})^6})\}^{-1}$	2	[− 65, 65]	1
Kowalik (F15)	$f(x) = \sum_{i=1}^{11}\left[a_i - \frac{x_1(b_i^2+b_i x_2)}{b_i^2+b_i x_3+x_4}\right]^2$	4	[− 5, 5]	0.0003
Six Hump Camel (F16)	$f(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	[− 5, 5]	−1.0316
Branin (F17)	$f(x) = (x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6)^2$ $+10(1-\frac{1}{8\pi})\cos x_1 + 10$	2	[− 5, 5]	0.398
Goldstein-Price (F18)	$f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2$ $-14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2 + 1)^2$ $(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	2	[− 2, 2]	3
Hartman 3 (F19)	$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{i=1}^{3} a_{ij}(x_j - p_{ij})^2\right)$	3	[− 1, 2]	−3.86
Hartman 6 (F20)	$f(x) = -\sum_{i=1}^{4} c_i \exp\left(-\sum_{i=1}^{6} a_{ij}(x_j - p_{ij})^2\right)$	6	[0, 1]	−0.32
Shekel 5 (F21)	$f(x) = -\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	4	[0, 1]	−10.1532
Shekel 7 (F22)	$f(x) = -\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	4	[0, 1]	−10.4028
Shekel 10 (F23)	$f(x) = -\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T + c_i]^{-1}$	4	[0, 1]	−10.5363
Schaffers (F24)	$f(x) = 0.5 + \frac{\sin^2(x_1^2+x_2^2)-0.5}{[1+0.001(x_1^2+x_2^2)]^2}$	2	[− 100, 100]	0
Easom (F25)	$f(x) = -\cos(x_1)\cos(x_2)exp[-(x_1-\pi)^2 \times 9(x_2-\pi)^2]$	2	[− 100, 100]	0
Schwefel 2.26 (F26)	$f(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i\rvert})$	30	[− 500, 500]	−418.982
Bohachevsky (F27)	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	[− 100, 100]	0
Bohachevsky 3 (F28)	$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.3$	2	[− 50, 50]	0
Colville (F29)	$f(x) = 100(x_1 - x_2^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_2)^2$ $+10.1(x_2-1)^2 + (x_4-1)^2 + 19.8(x_2-1)(x_4-1)$	4	[− 10, 10]	0

Table 6 Fixed-dimension uni-modal test functions

Function	Description	Dimensions	Range	f_{min}
Booth (F30)	$f(x) = (2x_1 + x_2 - 5)^2 + (x_1 + 2x_2 - 7)^2$	2	[− 10, 10]	0
Matyas (F31)	$f(x) = 0.26(x_1^2 + x_2^2) - 0.48x_1x_2$	2	[− 10, 10]	0
Zettl (F32)	$f(x) = (x_1^2 + x_2^2 - 2x_1)^2 + 0.25x_1$	2	[− 1, 5]	−0.00379
Leon (F33)	$f(x) = 100(x_2 - x_1^3)^2 + (1-x_1)^2$	2	[− 1.2, 1.2]	0

5.1 Benchmark function

Thirty-three reference functions are chosen and split into three categories: fixed-dimensional benchmark function, multi-modal benchmark function, and uni-modal benchmark function [16]. These benchmark functions are used to test the proposed e-PDO algorithm. With the exception of setting the CEC-2017 functions' dimension to 30, algorithms are compared using identical parameter values. Uni-modal test functions (F1 to F7) are defined in Table 3. Multi-modal test functions (F8 to F13) are defined in Table 4. Tables 5 and 6 provide definitions for fixed-dimension uni-modal (F30 to F33) and multi-modal (F14 to F29) functions.

The uni-modal functions (F1 to F7) contain only one local optimum. The exploitation potential of optimization methods can be assessed with the aid of uni-modal functions. Consequently, a meta-heuristic method with the highest exploitation potential is used to optimise these functions. Selected multi-modal functions (F8 to F13) and fixed-dimensional multi-modal functions (F14 to F29) have many local minima associated with these functions. Due to the fact that these functions' solutions might occasionally become stuck in the local optima and are impossible to escape, they can be more challenging to solve than uni-

Table 7 Algorithm control parameters

Algorithm	Parameter and description	Value	Parameter and description	Value
E-PDO	N (Population Size)	30	T (Maximum Iteration)	1000
	ϵA	0.1	ϵ	$2.2204e^{-16}$
	τ	0.005	β	1.5
	w	0.001	ω	0.001
PDO [11]	N (Population Size)	30	T (Maximum Iteration)	1000
	ϵA	0.1	ϵ	$2.2204e^{-16}$
	τ	0.005	β	1.5
ALO [26]	Population Size	30	Maximum Iteration	1000
	Initial I ratio value	1		
WOA [29]	Population Size	30	Maximum Iteration	1000
	Parameter a	Linear decreasing value from 2 to 0	Parameter $a2$	Linear decreasing value from − 1 to − 2
	Parameter b	1	Parameter l	[− 1, 1]
	Random Number (r')	[0,1]		
DA [27]	Population Size	30	Maximum Iteration	1000
	Food attraction weight f	1		
GWO [30]	Population Size	30	Maximum Iteration	1000
	Convergence parameter a	Linear decreasing value from 2 to 0		
MFO [25]	Population Size	30	Maximum Iteration	1000
	Convergence constant	Linear decreasing value from − 1 to − 2		
SCA [28]	Population Size	30	Maximum Iteration	1000
	Parameter a (constant)	2		
RSA [1]	Population Size	30	Maximum Iteration	1000
	Sensitive parameter α	0.1	Sensitive parameter β	0.005
	Evolutionary sense (ES)	Decreasing random values between 2 and − 2		

modal functions. The complexity level of multi-modal functions also increases as multiple dimensions, search domains, and local optima values increase. Because of their capability to discover new sites, these test the MA's ability to explore. The experimental outcomes are contrasted with those of the original PDO and 7 other well-known algorithms. Results are evaluated using the statistical tests described in the next section.

5.2 Experimental setup

To ensure consistency across all validation tests, we chose (N) = 30 for the population size, (D) = 10 for the size of the dimensions, and (T) = 1000 for the number of iterations. The 30 iterations of each function are rounded to two decimal places to reduce statistical error and produce statistically significant output. Two measures are used to evaluate the algorithm's performance: the mean and standard deviation measures. The best, the worst, the mean, and the standard deviation (SD) are shown as the final experimental results after each method has been run separately 30 times for each function. All tests are conducted using MATLAB R2020a on a computer running Windows 10 and equipped with an Intel(R) Core(TM) i7-4790 CPU clocked at 3.60GHz and 8GB of RAM. All the required parameter values for each algorithm are listed in Table 7.

5.3 Experimental results

In Tables 8 and 9, along with e-PDO and the other seven algorithms, the mean and standard deviation for optimised uni-modal functions are shown. The table clearly shows that, in comparison with other algorithms, e-PDO offered the least values. The functions F1, F2, F3, F4, and F7 get the optimum results when using the e-PDO algorithm. It provides the 5^{th} and 4^{th} best outcomes for functions F5 and F6, respectively. Boldface indicates all average best results. So it stands to reason that our suggested method is a better algorithm than others.

In the case of multi-modal (F8–F13) functions (from Tables 10, 11), it achieves the optimum results for F8–F11 except for F12 and F13 functions. For fixed-dimension

Table 8 Result of uni-modal test functions (F1–F7)(Dimension = 10)

Function	Global	Value	E-PDO	E-PDOUW	PDOL	PDO	ALO	SCA
F1	0	Best	0	0	0	0	7.62×10^{-10}	3.37×10^{-35}
		Worst	0	0	0	0	3.49×10^{-9}	1.73×10^{-25}
		Average	**0**	**0**	**0**	**0**	2.03×10^{-9}	7.12×10^{-27}
		SD	0	0	0	0	7.18×10^{-10}	3.16×10^{-26}
F2	0	Best	0	0	0	0	9.79×10^{-9}	2.98×10^{-24}
		Worst	0	0	0	0	0.0035132	4.62×10^{-17}
		Average	**0**	**0**	**0**	**0**	0.0002046	2.06×10^{-18}
		SD	0	0	0	0	0.000701	8.61×10^{-18}
F3	0	Best	0	0	0	0	4.88×10^{-9}	3.03×10^{-16}
		Worst	0	0	0	0	1.56×10^{-6}	3.71×10^{-8}
		Average	**0**	**0**	**0**	**0**	2.85×10^{-7}	2.47×10^{-9}
		SD	0	0	0	0	3.902×10^{-7}	8.19×10^{-9}
F4	0	Best	0	0	0	0	3.102×10^{-6}	3.04×10^{-11}
		Worst	0	0	0	0	0.000734	3.38×10^{-7}
		Average	**0**	**0**	**0**	**0**	8.94×10^{-5}	2.103×10^{-8}
		SD	0	0	0	0	0.000158	6.39×10^{-8}
F5	0	Best	0.6113	8.0239	7.3773	0.0288	2.48×10^{-6}	6.8286
		Worst	9	8.9803	8.9734	9	469.76	8.7271
		Average	7.7682	8.6304	8.6011	3.7673	54.725	7.4488
		SD	2.3118	0.3493	0.4536	2.7853	128.3108	0.5566
F6	0	Best	0.001705	1.0921	0.8175	0	6.11×10^{-10}	0.0856
		Worst	0.004565	1.8552	1.8336	1.12×10^{-21}	2.51×10^{-9}	0.7659
		Average	0.002923	1.5584	1.58597	3.85×10^{-23}	1.39×10^{-9}	0.3417
		SD	0.000665	0.2014	0.20003	2.07×10^{-22}	4.99×10^{-10}	0.1604
F7	0	Best	1.52×10^{-6}	4.88×10^{-7}	6.0478×10^{-7}	1.13×10^{-6}	0.0017	0.000222
		Worst	4.77×10^{-5}	0.00019	0.000108	0.000161	0.0133	0.007631
		Average	1.73×10^{-5}	3.98×10^{-5}	2.4×10^{-5}	2.9×10^{-5}	0.0055	0.001461
		SD	1.23×10^{-5}	4.22×10^{-5}	2.25×10^{-5}	3.05×10^{-5}	0.00291	0.001592

multi-modal test functions (F14–F29) (from Tables 12, 13, 14 and 15), with the exception of the F18–F19 and F21–F23 functions, e-PDO demonstrates its superiority over other algorithms. For fixed-dimension uni-modal test functions (F30–F33), e-PDO gives best values for F31.

In addition to the above analysis, if we compare e-PDO and other PDO versions on the CEC test problems from Tables 18 and 19, it can be concluded that e-PDO achieves better results than the original PDO algorithm. Tables 8, 9, 10, 11, 12, 13, 14, 15, 16, 17 and 18 further show that, in terms of average and standard deviation indices, our suggested e-PDO algorithm outperformed the other 7 meta-heuristic algorithms. The outcomes showed that by combining the DOL technique and the modified random walk of prairie dogs, e-PDO might produce a better solution.

5.4 Statistical analysis of experimental results

5.4.1 Performance indicators

Various statistical tools are used in this paper, namely the average value of the objective functions ($Average_F$) and the standard deviation (SD_F). Their mathematical formulations are given as follows.

The average measure, which can be expressed by Eq. 36, determines the average of the optimal values resulting from the algorithm and is assessed over a number of predefined runs.

$$Average_F = \frac{1}{R_n}\sum_{i=1}^{R_n} F_i \tag{36}$$

Table 9 Result of uni-modal test functions (F1–F7)

Function	Global	Value	WOA	DA	GWO	MFO	RSA
F1	0	Best	8.15×10^{-190}	0	2.71×10^{-61}	1.06×10^{-34}	0
		Worst	6.101×10^{-171}	152073	3.73×10^{-57}	1.58×10^{-27}	0
		Average	2.06×10^{-172}	5073.3699	1.73×10^{-58}	5.49×10^{-29}	0
		SD	0	27763.799	$6.82\times^{-58}$	2.94×10^{-28}	0
F2	0	Best	3.11×10^{117}	0	1.06×10^{-35}	7.31×10^{-21}	0
		Worst	1.92×10^{-108}	7.8212	2	5.99×10^{-18}	0
		Average	2.02×10^{-109}	1.1079	0.1	4.504×10^{-19}	0
		SD	5.83×10^{-109}	1.6984	0.4026	1.08×10^{-18}	0
F3	0	Best	994.226	0	9.95×10^{-20}	1.77×10^{-13}	0
		Worst	19039.0914	5073.288	1.402×10^{-12}	4.27×10^{-8}	0
		Average	10194.164	197.3722	4.85×10^{-14}	2.81×10^{-9}	0
		SD	4805.966	921.672	2.56×10^{-13}	8.28×10^{-9}	0
F4	0	Best	0.000092	0	4.65×10^{-16}	1.78×10^{-9}	0
		Worst	83.2756	4.1176	8.45×10^{-14}	0.000042	0
		Average	27.6845	1.1733	1.77×10^{-14}	2.39×10^{-6}	0
		SD	24.5755	1.0726	2.07×10^{-14}	8.15×10^{-6}	0
F5	0	Best	26.0043	7.2949	25.3101	0.34088	8.86×10^{-30}
		Worst	27.0043	3137215	36.1552	22.1917	3.6×10^{-29}
		Average	26.544	107877.4572	27.0835	6.92103	2.09×10^{-29}
		SD	0.2959	572385.091	1.8729	5.1194	8.39×10^{-30}
F6	0	Best	0.001365	5.9223×10^{-7}	0.0000228	0	1.3997
		Worst	0.00967	44.8047	1.2533	3.28×10^{-30}	2.5
		Average	0.00375	3.6063	0.5489	2.26×10^{-31}	2.06521
		SD	0.00208	9.7113	0.3366	6.17×10^{-31}	0.2876
F7	0	Best	0.0000446	0.0003736	0.000189	0.000475	3.189×10^{-6}
		Worst	0.005483	0.16272	0.0035398	0.011526	0.0002119
		Average	0.001232	0.01853	0.00091018	0.002612	4.89×10^{-5}
		SD	0.001404	0.02932	0.0006709	0.00198	5.35×10^{-5}

where R_n is the quantity of runs. The optimal result is represented by F_i.

The standard deviation (SD) measurement is used to test whether the algorithm under evaluation can achieve the same best value across all runs, and to examine the consistency of algorithmic output, and can be represented by Eq. 37:

$$SD_F = \sqrt{\frac{1}{R_n - 1}\sum_{i=1}^{R_n}(F_i - Average_F)^2} \tag{37}$$

We specify the average value and standard deviation (SD) of the e-PDO along with seven other existing MAs such as GWO, MFO, ALO, WOA, DA, SCA and RSA for matching. Here, we also include another two versions of e-PDO.

5.4.2 Measurement of alternatives and ranking according to compromise solution (MARCOS) method

More than just scanning and comparing a set of benchmark function lists should be included in choosing the best method. This section provides a description of the MARCOS approach (developed by Stevic et al. [42]). This is a new approach to multi-criteria analysis [41]. The MARCOS technique is built on specifying how alternatives and reference values relate to one another (ideal and anti-ideal alternatives). The utility functions of the alternatives are established based on the aforementioned relationships, and compromise rankings with respect to ideal and anti-ideal solutions are generated. Utility functions are used to define the preference for decisions. Utility functions show a choice's position in relation to ideal and anti-ideal possibilities. The optimal alternative is the one that is farthest from the anti-ideal reference point while also being the

Table 10 Result of multi-modal test functions (F8–F13) (Dimension = 10)

Function	Global	Value	E-PDO	E-PDOUW	PDOL	PDO	ALO	SCA
F8	0	Best	-7.2×10^{112}	-9.98×10^{307}	-9.54×10^{307}	−20124.6231	−4189.83	−2573.584
		Worst	-1.04×10^{65}	-6.42×10^{306}	1.00001×10^{307}	−1805.8916	−1925.85	−1882.2845
		Average	-2.6×10^{111}	-3.7×10^{307}	-4.01×10^{307}	−2588.5597	−2587.96	−2202.04
		SD	1.3×10^{112}	-9.98×10^{307}	2.61×10^{307}	3315.5742	591.9149	149.52
F9	0	Best	0	0	0	0	3.9798	0
		Worst	0	0	0	0	36.8134	0
		Average	0	0	0	0	17.6754	0
		SD	0	0	0	0	7.2211	0
F10	0	Best	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	0.00001	4.44×10^{-15}
		Worst	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	2.5799	1.11×10^{-13}
		Average	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	8.88×10^{-16}	0.2105	1.19×10^{-14}
		SD	4.01×10^{-31}	8.88×10^{-16}	4.01×10^{-31}	4.01×10^{-31}	0.6776	2.32×10^{-14}
F11	0	Best	0	0	0	0	0.071445	0
		Worst	0	0	0	0	0.39623	6.66×10^{16}
		Average	0	0	0	0	0.182341	2.297×10^{15}
		SD	0	0	0	0	0.092088	1.24×10^{16}
F12	0	Best	2.1×10^{-5}	0.36427	0.00005612	0.000104	6.7×10^{-12}	0.017787
		Worst	2.5254	1.9875	2.0135	18386	7.3359	0.11858
		Average	0.446622	0.610027	0.7698932	613.4228	0.001099	0.074222
		SD	0.571419	0.273173	0.5815198	3356.7040	1.65491	0.023585
F13	0	Best	0.000218	0.63082	0.00027928	0.030543	7.73×10^{-12}	0.14377
		Worst	0.9989	0.84509	1	1	0.010987	0.51424
		Average	0.422178	0.76349	0.7725211	0.835132	0.001099	0.280396
		SD	0.455744	0.05483	0.40552581	0.262103	0.003352	0.097908

most likable to the ideal. The following steps comprise the MARCOS methodology:

Step 1. Constructing an initial decision-making matrix. In multi-criteria models, there are n criteria and m alternatives defined. A team of r experts should be put together to evaluate the options in light of the criteria when making decisions as a group. An initial group decision-making matrix is created in the event of group decision-making by merging expert assessment matrices.

Step 2. The formation of a lengthy initial matrix. The original matrix is enlarged at this stage by providing the ideal (AI) and anti-ideal (AAI) solution.

$$\mathbf{X} = \begin{array}{c} \\ AAI \\ A_1 \\ A_2 \\ \vdots \\ A_m \\ AI \end{array} \begin{array}{c} \begin{array}{cccc} \bar{C}_1 & \bar{C}_2 & \cdots & \bar{C}_n \end{array} \\ \begin{pmatrix} x_{aa1} & x_{aa2} & \cdots & x_{aan} \\ x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ x_{m1} & x_{m2} & \cdots & x_{mn} \\ x_{ai1} & x_{ai2} & \cdots & x_{ain} \end{pmatrix} \end{array}$$

The anti-ideal solution (*AAI*) is the worst choice, whereas the ideal solution (*AI*) is the finest alternative. (*AAI*) and (*AI*)s are defined using equations according to the nature of the criteria:

$$AAI = \begin{cases} \min\limits_i x_{ij} & \text{if } j \in B \text{ (for maximizing problem)} \\ \max\limits_i x_{ij} & \text{if } j \in C \text{ (for minimizing problem)} \end{cases}$$

Table 11 Result of multi-modal test functions (F8–F13)

Function	Global	Value	WOA	DA	GWO	MFO	RSA
F8	0	Best	−12569.486	−4068.2796	−6650.3486	−4071.3905	−2162.7606
		Worst	−7549.6955	−2482.911	−2306.5925	−2402.6344	−1743.3803
		Average	−11468.701	−3121.4874	−3572.4041	−3351.6025	−1992.8401
		SD	1565.76069	453.742437	1507.4426	410.55269	99.1028426
F9	0	Best	0	0.50627	0	2.9849	0
		Worst	0	40.6912	0	42.8541	0
		Average	0	19.2120957	0	17.9494833	0
		SD	0	12.9280178	0	11.1621627	0
F10	0	Best	8.88×10^{-16}	7.99×10^{-15}	4.44×10^{-15}	4.44×10^{-15}	8.88×10^{-16}
		Worst	7.99×10^{-15}	4.1544	7.99×10^{-15}	4.44×10^{-15}	8.88×10^{-16}
		Average	4.08×10^{-15}	1.33847207	4.79×10^{-15}	4.44×10^{-15}	8.88×10^{-16}
		SD	2.53×10^{-15}	1.26209177	1.08×10^{-15}	4.01×10^{-30}	4.01×10^{-31}
F11	0	Best	0	0	0	0.051625	0
		Worst	0	1.1121	0.22036	0.27545	0
		Average	0	0.49096463	0.01505519	0.14882998	0
		SD	0	0.31129449	0.0409164	0.07322223	0
F12	0	Best	0.00018172	0.00073046	6.3×10^{-8}	4.71×10^{-32}	0.33533
		Worst	0.020475	3.0483	0.039702	0.62195	2.5621
		Average	0.00148689	0.53167864	0.00363845	0.04146567	0.67915767
		SD	0.00371422	0.68534156	0.00908974	0.13502161	0.40751938
F13	0	Best	0.01718	0.0001682	4.67×10^{-7}	1.35×10^{-32}	2.09×10^{-32}
		Worst	0.47733	0.52355	0.10013	0.010987	4.83×10^{-31}
		Average	0.20148977	0.11798804	0.01970615	0.00292987	1.46×10^{-31}
		SD	0.13040069	0.13075897	0.04008615	0.00494169	2.02×10^{-31}

$$AI = \begin{cases} \max\limits_{i} x_{ij} & \text{if } j \in B \text{ (for maximizing problem)} \\ \min\limits_{i} x_{ij} & \text{if } j \in C \text{ (for minimizing problem)} \end{cases}$$

where B represents a set of criteria for benefits and C represents a set of criteria for costs.

Step 3. The extended starting matrix (X) is normalised. The equations are used to calculate the elements of the normalized matrix $N = [n_{ij}]_{m \times n}$.

$$n_{ij} = \begin{cases} \dfrac{x_{ij}}{x_{ai}} & \text{if } j \in B \text{ (for maximizing problem)} \\ \dfrac{x_{ai}}{x_{ij}} & \text{if } j \in C \text{ (for minimizing problem)} \end{cases}$$

where x_{ij} and x_{ai} are indeed the elements of the matrix X and the ideal solution of the alternative, respectively.

Step 4. $V = [v_{ij}]_{m \times n}$ is the weighted matrix. The weighted matrix V is formed by multiplying the normalized matrix N by weightage. Using the weight coefficients of the criteria w_j, the equation may be solved.

$$v_{ij} = n_{ij} \times w_j$$

Step 5. The utility degree of alternative K_i is calculated. The utility degrees of an alternative in regard to the anti-ideal and ideal solutions are determined using equations K_i^- and K_i^+.

$$K_i^- = \frac{S_i}{S_{aai}}$$

$$K_i^+ = \frac{S_i}{S_{ai}}$$

where $S_i; (i = 1, 2, ..., m)$ denotes the total number of items in the weighted matrix V.

$$S_i = \sum_{i=1}^{n} v_{ij}$$

Step 6. Calculating the utility function of alternatives $f(K_i)$. The utility function is the trade-off between the observable alternative and the ideal and anti-ideal solutions. The equation expresses the utility function of alternatives.

$$f(K_i) = \frac{K_i^+ + K_i^-}{1 + \frac{1-f(K_i^+)}{f(K_i^+)} + \frac{1-f(K_i^-)}{f(K_i^-)}}$$

where $f(K_i^-)$ represents the utility function in relation to the anti-ideal solution and $f(K_i^+)$ represents the utility function in relation to the ideal solution. Equations are used

Table 12 Result of fixed-dimension multi-modal test functions (F14–F21)

Function	Global	Value	E-PDO	E-PDOUW	PDOL	PDO	ALO	SCA
F14	0	Best	0.998	0.998	0.998	0.998	0.998	0.998
		Worst	8.8408	0.998	12.6705	12.6705	0.998	2.9821
		Average	2.0828	0.998	4.2141	3.941490	0.998	1.461689
		SD	2.071605	4.52×10^{-16}	2.8970626	3.939739	4.52×10^{-16}	0.853117
F15	0	Best	0.000308	0.000528	0.00016788	0.000308	0.000307	0.000335
		Worst	0.0016443	0.000991	0.0034221	0.002252	0.063291	0.001526
		Average	0.000529	0.0008003	0.0007761	0.000854	0.005408	0.000943
		SD	0.000248	0.0001099	0.0006627	0.000442	0.012866	0.000371
F16	0	Best	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
		Worst	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
		Average	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
		SD	6.77×10^{-16}	6.77×10^{-16}	6.77×10^{-16}	6.77×10^{-16}	6.77×10^{-16}	6.77×10^{-16}
F17	0	Best	0.39789	0.39789	0.39789	0.39789	0.39789	0.39789
		Worst	0.39789	0.3979	0.39789	0.39789	0.39789	0.4034
		Average	0.39789	0.3978903	0.39789	0.39789	0.39789	0.398671
		SD	1.69×10^{-16}	1.826×10^{-6}	1.69×10^{-16}	1.69×10^{-16}	1.69×10^{-16}	0.001017
F18	0	Best	3	3	3	3	3	3
		Worst	3.0008	3.0019	3.0354	3	3	3
		Average	3.0001	3.00022	3.00144	3	3	3
		SD	0.00017019	0.00043	0.006422	0	0	0
F19	0	Best	−3.8628	−3.8647	−3.869	−3.8628	−3.8628	−3.8618
		Worst	−3.8625	−3.8478	−3.8521	−3.8628	−3.8628	−3.851
		Average	−3.8627	−3.86132	−3.86192	−3.8628	−3.8628	−3.854937
		SD	9.097×10^{-5}	0.0034764	0.002346	3.16×10^{-15}	3.16×10^{-15}	0.002447
F20	0	Best	−3.3211	−3.3028	−3.3167	−3.322	−3.322	−3.1261
		Worst	−3.2763	−2.782	−3.124	−2.6312	−3.2031	−1.4573
		Average	−3.301967	−3.24605	−3.273543	−3.0528	−3.2625	−2.86243
		SD	0.010324	0.109724	0.0386142	0.2224	0.0605	0.371310
F21	0	Best	−5.0112	−5.0541	−5.055	−5.0552	−10.1532	−5.0552
		Worst	−4.4903	−5.0418	−5.0401	−5.0552	−2.6305	−5.0552
		Average	−4.814037	−5.0499867	−5.05037367	−5.0552	−7.2757	−5.0552
		SD	0.114594	0.002946	0.0029264	1.81×10^{-15}	2.8021	1.81×10^{-15}

to calculate utility functions in relation to the ideal and anti-ideal solutions.

$$f(K_i^-) = \frac{K_i^+}{K_i^+ + K_i^-}$$

$$f(K_i^+) = \frac{K_i^-}{K_i^+ + K_i^-}$$

Step 7. The choices are ranked. To rank the options, the utility function's final values are employed. The maximum utility function value that is possible should be assigned to an option.

5.4.3 MARCOS calculation

According to the MARCOS method, CEC test functions are the criteria used here, all algorithms are alternatives, and all problems(CEC test functions) are minimization problems. As a result, these criteria are cost criteria., using this procedure e-PDO is the best algorithm for solving CEC test functions. Table 19 clearly shows the computational supremacy of e-PDO.

5.5 Convergence report

To compare the convergence speed of EPDO with other algorithms (with GWO, MFO, ALO, WOA, SCA, and RSA

Table 13 Result of fixed-dimension multi-modal test functions (F14–F21)

Function	Global	Value	WOA	DA	GWO	MFO	RSA
F14	0	Best	0.998	0.998	0.998	0.988	1.0124
		Worst	107632	0.998	12.6705	6.9033	107632
		Average	3589.81204	0.998	3.93279333	3.26519333	3591.52789
		SD	19650.4323	4.52×10^{-16}	4.24423761	2.49696204	19650.1083
F15	0	Best	0.000308	0.00041312	0.00030749	0.00030855	0.00043333
		Worst	0.40339	0.020363	0.020363	0.0016554	0.0021201
		Average	0.0140329	0.00191875	0.00372748	0.0009831	0.00101859
		SD	0.07353833	0.00352198	0.00757248	0.00036669	0.00037764
F16	0	Best	−1.0316	−1.0316	−1.0316	−1.0316	−1.0316
		Worst	−1.0316	−1.0316	−1.0316	−1.0316	−1.0217
		Average	−1.0316	−1.0316	−1.0316	−1.0316	−1.03066
		SD	6.77×10^{-16}	6.77×10^{-16}	6.77×10^{-16}	6.77×10^{-16}	0.00187425
F17	0	Best	0.39789	0.39789	0.39789	0.39789	0.39789
		Worst	0.39789	0.39789	0.39789	0.39789	0.4327
		Average	0.39789	0.39789	0.39789	0.39789	0.40298259
		SD	1.69×10^{-16}	1.69×10^{-16}	1.69×10^{-16}	1.69×10^{-16}	0.00757068
F18	0	Best	0.3	3	3	3	3
		Worst	3.0006	3	3	3	30.0105
		Average	2.91004	3	3	3	3.90044667
		SD	0.49295787	0	0	0	4.93140179
F19	0	Best	−3.8628	−3.8628	−3.8628	−3.8628	−7.6939
		Worst	−3.8488	−3.8624	−3.8549	−3.8628	−3.6375
		Average	−3.8596167	−3.86276	−3.8614767	−3.8628	−3.9602667
		SD	0.00343	0.00010372	0.00272	3.16×10^{-15}	0.70648447
F20	0	Best	−3.322	−3.322	−3.322	−3.322	−3.0977
		Worst	−3.0156	−2.9529	−3.1376	−3.1376	−1.7753
		Average	−3.2431367	−3.2622367	−3.25783	−3.2264767	−2.7668733
		SD	0.08684007	0.08572106	0.06696046	0.05606244	0.31216593
F21	0	Best	− 101528	−10.1532	−10.1531	−10.1532	−5.0552
		Worst	−2.6305	−5.0552	−2.5531	10.1532	−5.0552
		Average	−7113.6972	−9.3086433	−9.14372	−5.04199	−5.0552
		SD	25725.9409	1.91814072	2.34926702	4.31566475	1.8067E−15

for a few benchmark functions), in Figs. 10 and 11, the values of the function estimate and the objective function are plotted on the horizontal and vertical axes, respectively, the curve is plotted with a dimension of 10. In most cases, EPDO has faster convergence optima compared to other methods. This performance is particularly noteworthy in the case of F1–F4, F7, F9–F11, F14–F20, F24–F29, and F31–F33. It is due to the presence of DOL initialization and generation jumping strategy as well as the m-Lévy. This helps in increasing the exploration capability. The special property of DOL called dynamic change, assists in excellent exploitation. As a result, EPDO is competitive and beats other algorithms in search accuracy, dependability, convergence speed, and exceeding the local optimum.

5.6 Applicability of EPDO for solving engineering design problems

The effectiveness of the EPDO algorithm in resolving engineering issues (constrained optimization problems) has been assessed in this subsection. Five engineering issues in total—the pressure vessel problem, the rolling element bearing design problem, the cantilever beam design problem, the tension/compression spring design problem and the gear train design problem have been used for this. Since the problem above has a lot of inequality constraints, if one of these constraints is violated, MAs use a penalty function that gets a good value. The results obtained by EPDO are compared with other MAs.

Table 14 Result of fixed-dimension multi-modal test functions (F22–F29)

Function	Global	Value	E-PDO	E-PDOUW	PDOL	PDO	ALO	SCA
F22	0	Best	−5.0096	−5.087	−5.0875	−5.0877	−10.4029	−5.0877
		Worst	−4.5863	−5.0733	−5.0765	−5.0877	−1.8376	−5.0877
		Average	−4.858583	−5.082663	−5.082663	−5.0877	−7.8796	−5.0877
		SD	0.085745	0.003007	0.0027866	9.034×10^{-16}	3.2253	9.03×10^{-16}
F23	0	Best	−5.0478	−5.1281	−5.1281	−5.1285	−10.5364	−5.1285
		Worst	−4.6523	−5.1151	−5.1127	−5.1285	−1.6766	−5.1285
		Average	−4.90481	−5.12389	−5.12335	−5.1285	−8.6688	−5.1285
		SD	0.092029	0.0032128	0.0034607	1.81×10^{-16}	3.2052	1.81×10^{-15}
F24	0	Best	0	0	0	0	2.22×10^{-16}	0
		Worst	0	0	0	0	2.8×10^{-14}	0
		Average	0	0	0	0	6.11×10^{-15}	0
		SD	0	0	0	0	6.51×10^{-15}	0
F25	0	Best	− 1	− 1	− 1	− 1	− 1	− 1
		Worst	−0.99704	−0.99679	−0.98873	−0.9515	0	−0.99856
		Average	−0.999839	−0.999607	−0.999194	−0.9962	−0.9667	−0.999628
		SD	0.000614	0.0007555	0.002187	0.0099	0.1826	0.000336
F26	0	Best	0	0	0	0	4.29×10^{-17}	1.03×10^{-6}
		Worst	0	0	0	0	4.29×10^{-8}	4.9855
		Average	0	0	0	0	1.61×10^{-9}	0.5745
		SD	0	0	0	0	7.81×10^{-9}	1.2482
F27	0	Best	0	0	0	0	3.77×10^{-15}	0
		Worst	0	0	0	0	2.09×10^{-10}	0
		Average	0	0	0	0	4.39×10^{-11}	0
		SD	0	0	0	0	5.96×10^{-11}	0
F28	0	Best	−0.6	−0.6	−0.6	−0.6	−0.6	−0.6
		Worst	−0.6	−0.6	−0.6	−0.6	−0.6	−0.6
		Average	−0.6	−0.6	−0.6	−0.6	−0.6	−0.6
		SD	0	0	0	0	0	0
F29	0	Best	−1021.374	−3179.2797	−3108.428	−398.2097	−401.7926	−399.3558
		Worst	−174.4206	−3.5709	−505.3499	−3.8059	−4.2148	−81.4242
		Average	−433.7878	−2154.1784	−2429.7953	−222.0092	−285.9628	−358.8803
		SD	262.168	862.3215	634.3033	146.5904	172.9691	75.9272

5.6.1 Pressure vessel problem

To maintain gases or liquids at a pressure that is often substantially greater than the atmospheric pressure, a closed vessel known as a pressure vessel is utilised (Fig. 12). Hemispherical heads are used to cap off a pressure vessel that is cylindrical at both ends. This optimization issue was put forth by Sandgren [39], and pressure vessels are frequently utilised for engineering purposes.

The construction of this compressed air tank, which has a working pressure of 3000 psi and a minimum capacity of 750 cubic ft, complied with the boiler and pressure vessel code of the American Society of Mechanical Engineers (ASME). The goal is to reduce the overall cost, which is made up of the welding, material, and forming costs. The variables are the shell thickness (T_s), head thickness (T_h), inner radius (R) and length of the part of the vessel without the head (L). The integer multiples of 0.0625 inches are the only ones that the thicknesses (T_s and T_h) can accept. The

Table 15 Result of fixed-dimension multi-modal test functions (F22–F29)

Function	Global	Value	WOA	DA	GWO	MFO	RSA
F22	0	Best	− 102657	−10.4029	−10.4029	−10.4029	−5.0877
		Worst	−2.7658	−2.7659	−10.402	−2.7519	−5.0877
		Average	−3430.131	−9.7949867	−10.40254	−8.1096667	−5.087
		SD	18740.9638	1.88341363	0.00025134	3.34108536	9.03×10^{-16}
F23	0	Best	−10.536	−10.5364	−10.5364	−10.5364	−5.1285
		Worst	−2.4217	−5.1756	−10.5353	−2.8066	−5.1285
		Average	−8.59425	−9.9735167	−10.535947	−9.87511	−5.1285
		SD	2.82784445	1.62938851	0.00027004	2.04078634	1.81×10^{-15}
F24	0	Best	0	0	0	0	0
		Worst	0	2.93×10^{-13}	0	0.009293	0
		Average	0	9.82×10^{-15}	0	0.00124581	0
		SD	0	5.35×10^{-14}	0	0.00239365	0
F25	0	Best	− 1	− 1	− 1	− 1	− 1
		Worst	0	0.99924	− 1	− 1	−0.98664
		Average	−0.9331677	−0.9257007	− 1	− 1	−0.997195
		SD	0.25366432	0.3654357	0	0	0.00365806
F26	0	Best	3.26×10^{-290}	0.7869	2.57×10^{-132}	9.6×10^{-10}	0
		Worst	3.46×10^{-216}	2057557	1.66×10^{-121}	6.27×10^{-5}	2.61×10^{-191}
		Average	1.2×10^{-217}	68706.6764	5.62×10^{-123}	2.39×10^{-6}	9.02×10^{-193}
		SD	0	375633.893	3.02×10^{-122}	1.14×10^{-5}	0
F27	0	Best	0	0	0	0	0
		Worst	0	0.078808	0	0	0
		Average	0	0.00386379	0	0	0
		SD	0	0.01482416	0	0	0
F28	0	Best	−0.6	−0.6	−0.6	−0.6	−0.6
		Worst	−0.6	0.6	−0.6	−0.6	−0.6
		Average	−0.6	−0.5599033	−0.6	−0.6	−0.6
		SD	0	0.2190709	0	0	0
F29	0	Best	−400.616	−401.9667	−402.1814	−402.1819	−222.851
		Worst	−0.0061075	−4.1113	−4.1059	−4.2138	-1.91×10^{-27}
		Average	−183.27279	−332.50502	−256.62211	−371.24806	−47.990069
		SD	149.66081	123.122173	175.170332	99.9505574	60.5660998

following is how this problem is mathematically formulated [22]:

$$\text{Consider } \vec{z} = [T_s, T_h, R, L]$$

$$\text{Minimize } f(\vec{z}) = 0.6224 T_s R L + 1.7781 T_h R^2 + 3.1661 T_s^2 L + 19.84 T_h^2 L$$

subject to

$$g_1(\vec{z}) = -T_s + 0.0193R \le 0,$$

$$g_2(\vec{z}) = -T_h + 0.00954R \le 0,$$

$$g_3(\vec{z}) = -\pi R^2 L - \frac{4}{3}\pi R^3 + 750 \times 11728 \le 0,$$

$$g_4(\vec{z}) = L - 240 \le 0,$$

$$0 \le T_s, T_h \le 99, 0 \le R, L \le 200$$

The optimal results of variables and objective function are inserted in Table 20. EPDO achieves a better result than the original PDO (from Table 20).

5.6.2 Rolling element bearing design problem

This problem (Fig. 13) has 10 parameters and 10 constraints. The primary objective of the current problem is to maximize the load-carrying capacity of bearing [4, 35].

The mathematical formulation of the design issue with rolling elements in bearings is follows:

Table 16 Result of fixed-dimension uni-modal test functions (F30–F33)

Function	Global	Value	E-PDO	E-PDOUW	PDOL	PDO	ALO	SCA
F30	0	Best	1.34×10^{-7}	1.47×10^{-6}	9.198×10^{-7}	1.56×10^{-6}	8.32×10^{-15}	0.000480
		Worst	4.48×10^{-5}	0.0006344	0.002511	0.0304	1.09×10^{-11}	0.021048
		Average	5.47×10^{-6}	0.00013169	0.000176	0.00308	1.29×10^{-12}	0.005183
		SD	1.001×10^{-5}	0.00014095	0.000454	0.007636	2.18×10^{-12}	0.005090
F31	0	Best	0	0	0	0	4.19×10^{-17}	2.16×10^{-129}
		Worst	0	0	0	0	1.05×10^{-14}	3.78×10^{-62}
		Average	0	0	0	0	2.02×10^{-15}	1.26×10^{-63}
		SD	0	0	0	0	2.47×10^{-15}	6.91×10^{-63}
F32	0	Best	−0.003791	−0.0037912	−0.0037912	−0.0037912	−0.0037912	−0.0037912
		Worst	−0.000149	−0.0037912	−0.0037912	−0.0037912	−0.0037912	−0.0037912
		Average	−0.003668	−0.0037912	−0.0037912	−0.0037912	−0.0037912	−0.0037912
		SD	0.000665	1.76×10^{-18}	1.76×10^{-18}	1.76×10^{-18}	1.76×10^{-18}	1.76×10^{-18}
F33	0	Best	4.0608×10^{-7}	4.22×10^{-8}	7.41×10^{-9}	0	5.09×10^{-17}	7.19×10^{-6}
		Worst	3.0316×10^{-5}	0.000070149	0.000128	0	2.18×10^{-14}	0.002285
		Average	1.1663×10^{-5}	9.77×10^{-6}	1.56×10^{-5}	0	3.89×10^{-15}	0.000413
		SD	8.9693×10^{-6}	1.42×10^{-5}	2.69×10^{-5}	0	5.65×10^{-15}	0.000554

Table 17 Result of fixed-dimension uni-modal test functions (F30–F33)

Function	Global	Value	WOA	DA	GWO	MFO	RSA
F30	0	Best	0.000061595	0	6.68×10^{-8}	8.43×10^{-23}	3.60×10^{-6}
		Worst	0.41409	0.015217	6.44×10^{-6}	0.0012647	0.33351
		Average	0.074980698	0.000889105	1.15×10^{-6}	9.92×10^{-5}	0.031177516
		SD	0.101016059	0.003175672	1.21×10^{-6}	0.000307325	0.063240251
F31	0	Best	0	0	2.49×10^{-248}	1.53×10^{-131}	0
		Worst	0	6.43×10^{-6}	2.61×10^{-137}	8.3776	0
		Average	0	7.24×10^{-7}	8.70×10^{-139}	0.279253333	0
		SD	0	1.70×10^{-6}	4.76×10^{-138}	1.529533499	0
F32	0	Best	−0.0037912	−0.0037912	−0.0037912	−0.0037912	−0.0037912
		Worst	−0.0037912	−0.0037424	−0.0037912	−0.0037912	0
		Average	−0.0037912	−0.003789273	−0.0037912	−0.0037912	−0.002265523
		SD	1.76×10^{-18}	8.9×10^{-6}	1.76×10^{-18}	1.76×10^{-18}	0.001872865
F33	0	Best	3.59×10^{-8}	1.42×10^{-14}	1.27×10^{-8}	5.11×10^{-8}	0
		Worst	0.00014661	0.0039227	1.28×10^{-6}	0.022591	4.93×10^{-32}
		Average	1.35×10^{-5}	0.000743198	2.99×10^{-7}	0.00562717	2.13×10^{-32}
		SD	2.78×10^{-5}	0.001517491	2.7×10^{-7}	0.006438524	2.35×10^{-32}

Table 18 Ranking and average ranking

	E-PDO	E-PDOUW	PDOL	PDO	ALO	SCA	WOA	DA	GWO	MFO	RSA
F1	1	1	1	1	10	9	6	11	7	8	1
F2	1	1	1	1	9	8	6	11	10	7	1
F3	1	1	1	1	9	7	11	10	6	8	1
F4	1	1	1	1	9	7	11	10	6	8	1
F5	5	7	6	2	10	4	8	11	9	3	1
F6	4	8	9	2	3	6	5	11	7	1	10
F7	1	4	2	3	10	8	7	11	6	9	5
F8	3	2	1	8	9	10	4	7	5	6	11
F9	1	1	1	1	9	1	1	11	1	10	1
F10	1	1	1	1	10	9	6	11	8	7	1
F11	1	1	1	1	9	11	1	10	7	8	1
F12	5	7	10	11	9	4	1	6	2	3	8
F13	8	9	10	11	2	7	6	5	4	3	1
F14	5	1	9	8	1	4	10	1	7	6	11
F15	1	3	2	4	10	5	11	8	9	6	7
F16	1	1	1	1	1	1	1	1	1	1	11
F17	1	9	1	1	1	10	1	1	1	1	11
F18	8	9	10	2	2	2	1	2	2	2	11
F19	6	9	7	2	2	11	10	5	8	2	1
F20	1	6	2	9	3	10	7	4	5	8	11
F21	11	9	8	5	4	5	1	2	3	10	5
F22	11	10	9	6	5	6	1	3	2	4	6
F23	11	9	10	6	4	6	5	2	1	3	6
F24	1	1	1	1	9	1	1	10	1	11	1
F25	3	5	6	8	9	4	10	11	1	1	7
F26	1	1	1	1	8	10	5	11	7	9	6
F27	1	1	1	1	10	1	1	11	1	1	1
F28	1	1	1	1	1	1	1	11	1	1	1
F29	3	2	1	9	7	5	10	6	8	4	11
F30	3	5	6	8	1	9	11	7	2	4	10
F31	1	1	1	1	9	8	1	10	7	11	1
F32	10	1	1	1	1	1	1	9	1	1	11
F33	6	5	8	1	3	9	7	10	4	11	2
AVG	3.6061	4.0303	3.9697	3.6364	6.0303	6.0606	5.1212	7.576	4.5455	5.3939	5.273
RANK	1	4	3	2	9	10	6	11	5	8	7

Table 19 Ranking using MARCOS method

Algorithms	K_i-	K_i+	$f(K_i-)$	$f(K_i+)$	$f(K_i)$	Rank
E-PDO	0.610285	6.713131	0.916667	0.083333	0.605696	1
E-PDOUW	0.561436	6.175794	0.916667	0.083333	0.557214	4
E-PDOL	0.609885	6.70873	0.916667	0.083333	0.605299	2
PDO	0.606688	6.673569	0.916667	0.083333	0.602127	3
ALO	0.338757	3.726323	0.916667	0.083333	0.33621	9
SCA	0.316056	3.476611	0.916667	0.083333	0.313679	10
WOA	0.479814	5.277958	0.916667	0.083333	0.476207	6
DA	0.24056	2.646164	0.916667	0.083333	0.238752	11
GWO	0.432961	4.762566	0.916667	0.083333	0.429705	7
MFO	0.369364	4.062999	0.916667	0.083333	0.366586	8
RSA	0.507212	5.579329	0.916667	0.083333	0.503398	5

Table 20 Results estimated for pressure vessel design

Algorithms	T_s	T_h	R	L	Best	Worst	Average	SD
EPDO	0.82	0.81	42.38	174.83	7303.75	9676.05	8575.09	610.37
PDO	3.49	1.14	69.95	37.01	33830.63	961258.57	415173.68	266249.23
SCA	0.8873	0.4883	44.4625	161.6696	6100.10	7676.20	6638.60	410.0589
WOA	1.1806	0.6682	55.6093	69.5988	6415.80	11584	7797.30	1078.70
MFO	0.9067	0.4482	46.9801	139.0231	5885.30	7319	6211.50	454.05
RSA	7.4493	36.6616	72.9797	121.3523	152440	118910	482780	276670

$$
\begin{aligned}
&Consider\ [\vec{z}] = [z_1, z_2, z_3, z_4, z_5, z_6, z_7, z_8, z_9, z_{10}] \\
&\qquad = [D_m, D_b, Z, f_i, f_o, K_{D_{min}}, K_{D_{max}}, \varepsilon, e, \zeta] \\
&Maximize \begin{cases} f_{\vec{z}} = f_c Z^{2/3} D_b^{1.8} & if\ D_b \le 25.4mm \\ f_{\vec{z}} = 3.647 f_c Z^{2/3} D_b^{1.4} & if\ D_b > 25.4mm \end{cases}
\end{aligned}
$$

subject to

$$
\begin{aligned}
&g_1(\vec{z}) = \frac{\phi_o}{2\sin^{-1}(D_b/D_m)} - z + 1 \ge 0, \\
&g_2(\vec{z}) = 2D_b - K_{D_{min}}(D-d) \ge 0, \\
&g_3(\vec{z}) = K_{D_{max}}(D-d) - 2D_b \ge 0, \\
&g_4(\vec{z}) = D_m - (0.5-e)(D+d) \ge 0, \\
&g_5(\vec{z}) = (0.5+e)(D+d) - D_m \ge 0, \\
&g_6(\vec{z}) = D_m - 0.5(D+d) \ge 0, \\
&g_7(\vec{z}) = 0.5(D - D_b - D_m) - \varepsilon D_b \ge 0, \\
&g_8(\vec{z}) = \zeta B_w - D_b \le 0, \\
&g_9(\vec{z}) = f_i \ge 0.515, \\
&g_{10}(\vec{z}) = f_o \ge 0.515, \\
&f_c = 37.91 \times \left[1 + \left\{1.04 \times \left(\frac{1-\gamma}{1-\gamma}\right)^{1.72} \times \left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \\
&\qquad \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(\gamma+1)^{1/3}}\right] \times \left[\frac{2f_i}{2f_i-1}\right]^{0.41}, \\
&\gamma = D_b/D_m,\ f_o = r_o/D_b,\ f_i = r_i/D_b, \\
&\phi_o = 2\pi - 2\cos^{-1} \\
&\frac{\{(D-d)/2 - 3(T/4)\}^2 + \{D/2 - (T/4) - D_b\}^2 - \{d/2 + (T/4)\}^2}{2\{(D-d)/2 - 3(T/4)\}\{D/2 - (T/4) - D_b\}}, \\
&T = D - d - 2D_b,\ D = 160,\ d = 90,\ B_w = 30,\ r_i = r_o = 11.033, \\
&0.5(D+d) \le D_m \le 0.6(D+d), 0.15(D-d) \le D_b \le 0.45(D-d), \\
&4 \le Z \le 50, 0.515 \le f_i \le 0.6, \\
&0.515 \le f_o \le 0.6, 0.4 \le K_{D_{min}} \le 0.5, 0.6 \le K_{D_{max}} \le 0.7, 0.3 \le \varepsilon \le 0.4, \\
&0.02 \le e \le 0.1, 0.6 \le \zeta \le 0.85
\end{aligned}
$$

The optimal results of variables and objective function are presented in Table 21. Compared to other algorithms, the optimal function value for this issue, 6.96×10^4, which is obtained via EPDO, can achieve greater solution accuracy.

5.6.3 Cantilever beam design problem

The cantilever beam's free end is subject to a vertical load, while the other side is rigidly supported. Figure 14 illustrates the cantilever beam design problem's structure. The goal is to reduce the weight of the beam, and the vertical displacement is a constraint that shouldn't be exceeded by the ideal design at all. The design of the cantilever beam problem is mathematically described as follows:

$$
\begin{aligned}
&Minimize\, f(z) = 0.0624(z_1 + z_2 + z_3 + z_4 + z_5) \\
&\text{subject to:} \\
&g(z) = \frac{61}{z_1^3} + \frac{37}{z_2^3} + \frac{19}{z_3^3} + \frac{7}{z_4^3} + \frac{1}{z_5^3} - 1 \le 0 \\
&0.01 \le z_i \le 100;\ i = 1, 2, \cdots, 5
\end{aligned}
$$

As compared to other algorithms, the optimal function value for this issue is 1.332 obtained from EPDO, indicating that it can achieve better solution accuracy (Table 22).

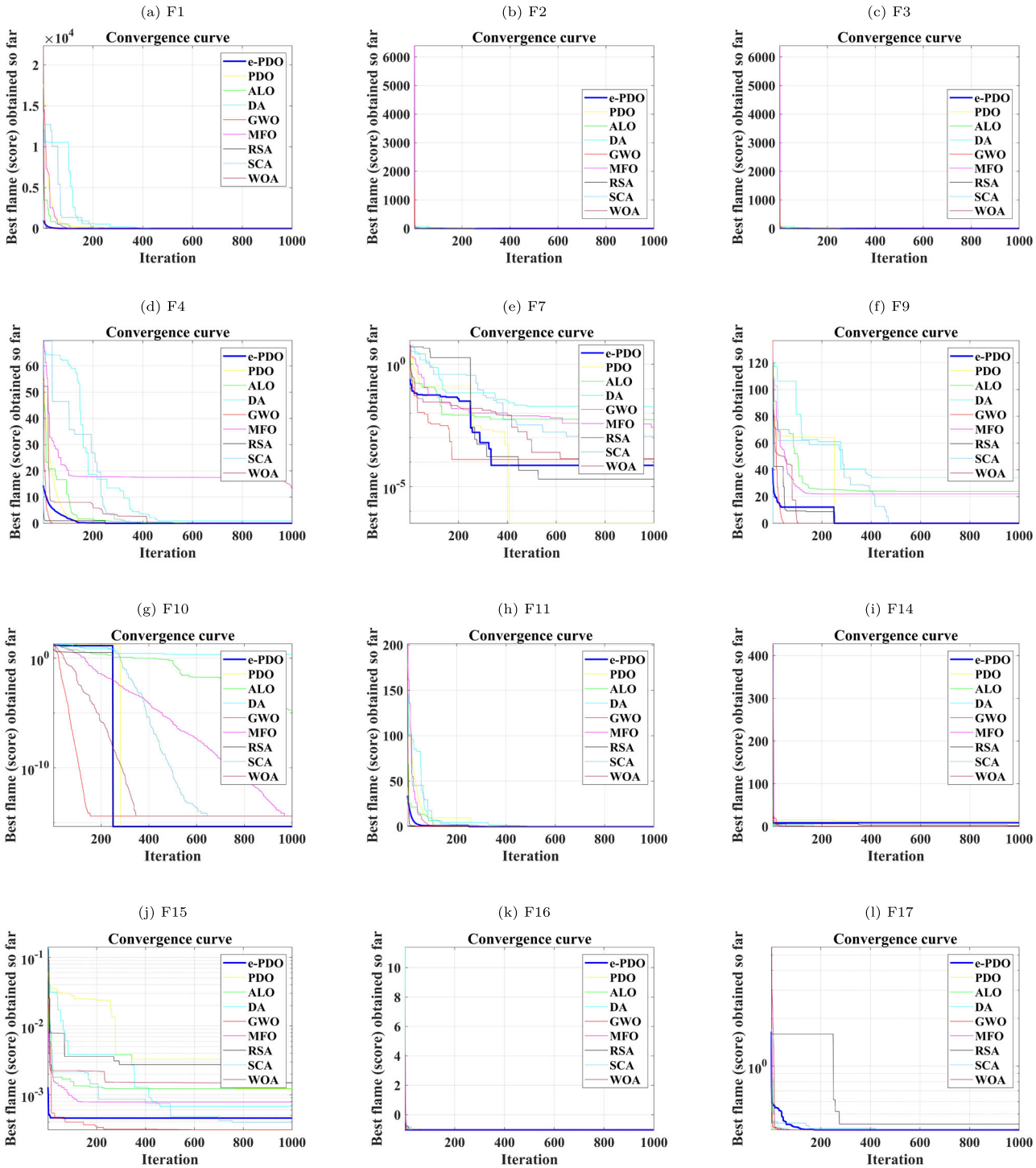

Fig. 10 Convergence curves of EPDO and competitor algorithms for different CEC-2017 test functions

5.6.4 Tension/compression spring design problem

In engineering sciences, the design of tension/compression springs [22] is an optimization challenge with four constraints to lessen the weight of these springs, whose schematic construction is depicted in Fig. 15. The components of this design problem are subject to buckling, stress, and deflection constraints. The problem centres on identifying the optimal values of two variables—the cross-sectional regions of the truss bars.

The design of tension/compression springs is mathematically expressed as follows:

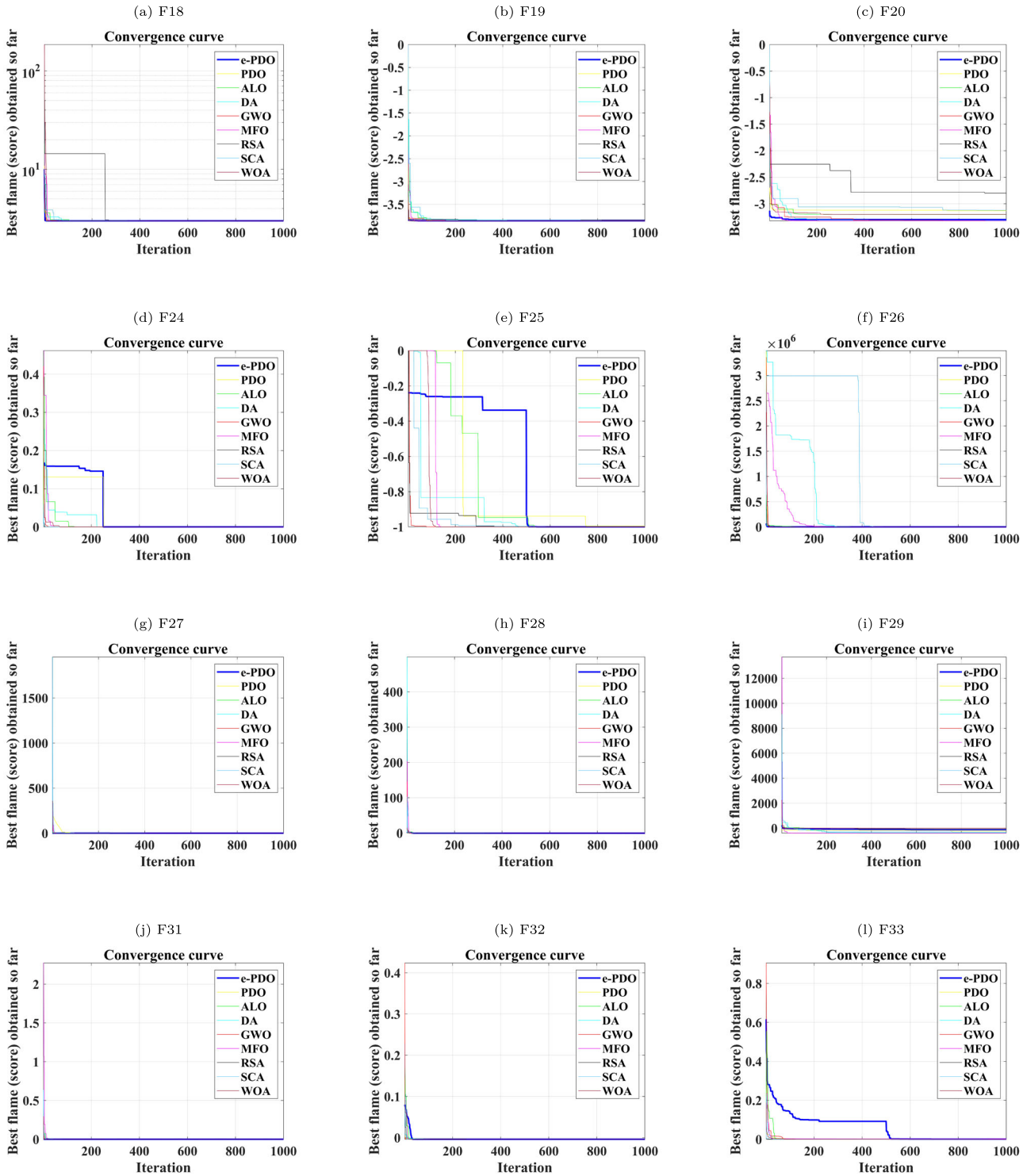

Fig. 11 Convergence curves of EPDO and competitor algorithms for different CEC-2017 test functions

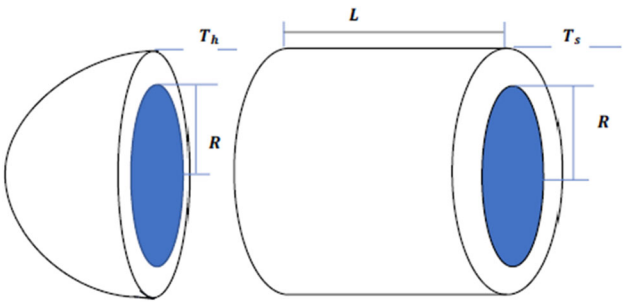

Fig. 12 Pressure vessel design

$$\text{Consider } [z] = [z_1, z_2, z_3] = [d, D, P]$$

$$\text{Minimizef}(z) = (z_3 + 2) z_2 z_1^2$$

subject to:

$$g_1(z) = 1 - \frac{z_2^3 z_3}{71785 z_1^4} \leq 0$$

$$g_2(z) = \frac{4z_2^2 - z_1 z_2}{12566(z_2 z_1^3)} + \frac{1}{5108 z_1^2} - 1 \leq 0$$

$$g_3(z) = 1 - \frac{140.45 z_1}{z_2^2 z_3} \leq 0$$

$$g_4(z) = \frac{z_1 + z_2}{1.5} - 1 \leq 0$$

$$0.05 \leq z_1 \leq 2.00;\ 0.25 \leq z_2 \leq 1.30;\ 2.00 \leq z_3 \leq 15.00$$

Fig. 13 Rolling element bearing design

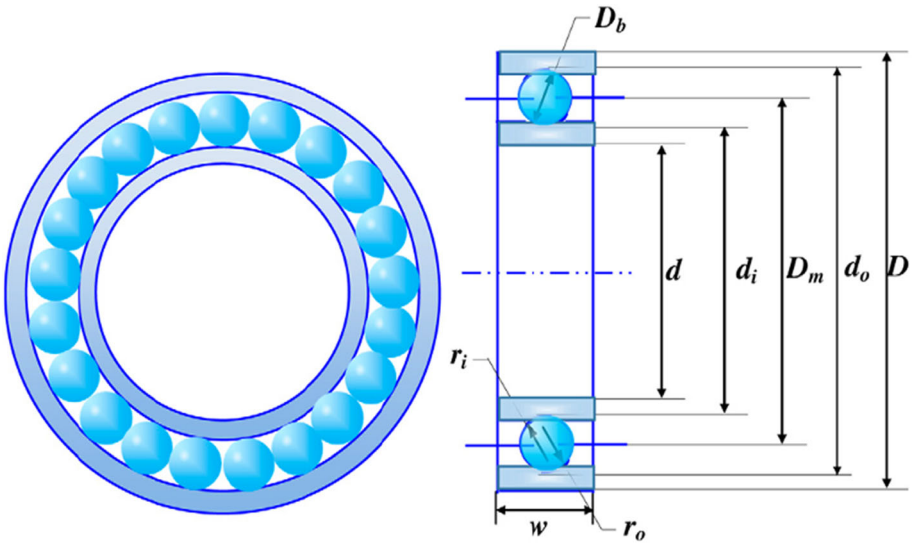

Table 21 Comparative studies estimated by various algorithms for rolling element bearing design

Variables	EPDO	PDO	SCA	WOA	MFO	RSA
D_m	125.5782	126.5857	125	125	125.7191	128.1271
D_b	21.06127	19.68419	21.26432	21.27301	21.42559	19.32694
Z	11	11	11	11	11	9
f_i	0.5166768	0.515533	0.515	0.515	0.515	0.527105
f_o	0.596	0.58185	0.515	0.515037	0.515	0.58492
$K_{D_{min}}$	0.4011888	0.4924716	0.44155	0.460342	0.5	0.495879
$K_{D_{max}}$	0.6437155	0.6317101	0.7	0.7	0.7	0.632665
ε	0.3041295	0.3416889	0.3	0.300022	0.3	0.313165
e	0.03119378	0.0050994	0.02	0.053931	0.02	0.024572
ζ	0.6097202	0.633754	0.6	0.60445	0.600403	0.60377
Best	6.96×10^4	7.25×10^4	−84398.9	−84459.8	−85549.2	-4.93×10^4
Worst	−47921.0721	5.721×10^{21}	−76467.6	−45196.4	−84459.8	1.84×10^{29}
Average	−60636.45057	1.69676×10^{21}	−81556.6	−78770.4	−85221.7	6.12×10^{27}
SD	5792.457846	1.57944×10^{21}	2005.239	8273.617	507.3587	3.35×10^{28}

Table 22 Comparative results for the cantilever beam design estimated using different algorithms

Algorithms	z_1	z_2	z_3	z_4	z_5	Best	Worst	AVG	SD
EPDO	5.0275	4.9958	4.9705	3.2653	3.0869	1.332	1.5902	1.5555	6.70E−03
PDO	5.0744	5.0744	5.0744	5.0744	5.0744	1.56	1.69E+00	1.58E+00	4.22E−02
SCA	6.2214	5.3335	4.6077	3.6606	2.3422	1.3552	1.4045	1.3831	1.23E−02
WOA	6.3496	5.2788	4.6311	3.8787	2.5801	1.3543	1.556	1.4176	0.0536
MFO	6.0222	5.3082	4.5091	3.4863	2.1544	1.34	1.3412	1.3404	3.27E−04
RSA	24.0023	32.1756	24.586	28.7294	18.9943	5.1413	1.01E+01	8.02E+00	1.33E+00

Table 23 Comparative results for the design of tension/compression springs

Algorithms	d	D	P	Best	Worst	AVG	SD
EPDO	0.0547	0.4211	9.1	0.0126	0.014	0.0129	1.23E−04
PDO	0.1025	0.9234	9.1667	0.0313	8.56E+19	2.10E+19	2.92E+19
SCA	0.0514	0.3483	12.5667	0.0127	0.0132	0.0131	1.47E−04
WOA	0.0582	0.5416	6.1667	0.0127	0.0178	0.0139	0.0012
MFO	0.0545	0.4387	9.9	0.0127	0.0178	0.0134	0.0011
RSA	0.1167	0.9296	9.2667	0.0303	9.69E+19	2.88E+19	3.12E+19

In order to find the best solution for the tension/compression spring design variables, the EPDO and competing algorithms' implementation results are provided in Table 23. Based on these findings, EPDO has offered the best solution to the tension/compression spring problem, with the goal function set to 0.0126 and the design variables set to (0.0547, 0.4211, 9.1). Table 23 also displays the statistical outcomes from optimizing this design issue using EPDO and competitive techniques.

5.6.5 Gear train design problem

The main objective of this problem is to minimize the cost of the gear ratio in a gear train, which includes four design variables, namely the number of teeth of gears.

$$\text{Gear ratio} = \frac{n_A \times n_B}{n_C \times n_D}$$

Figure 16 illustrates this design issue. The gearwheel's teeth come in four different varieties A, B, C, and D. n_A, n_B, n_C, and n_D are used to denote the design variables. The range [12, 60] encompasses all of them as integers.

The gear train design challenge [39] is mathematically expressed as follows:

$$Consider\ [n_A, n_B, n_C, n_D]$$

$$Minimize\ f(n_A, n_B, n_C, n_D) = \left(\frac{1}{6.931} - \frac{n_A \times n_B}{n_C \times n_D}\right)^2$$

subject to:

$$12 \leq n_A, n_B, n_C, n_D \leq 60$$

Table 24 presents the statistical data on the outcomes of numerous independent runs in comparison to the outcomes of other algorithms. Compared to other algorithms, it demonstrates that the suggested algorithm performs better.

5.7 Discussions

Incorporating DOL initialization and the generation-jumping strategy significantly improves EPDO's exploration capabilities. Furthermore, utilizing the m-Lévy distribution enhances the algorithm's adaptability and exploration prowess. The dynamic change feature of DOL further aids in achieving excellent exploitation. Consequently, EPDO stands out as a competitive algorithm, excelling in search accuracy, dependability, convergence speed, and its capacity to overcome local optima. EPDO consistently demonstrates superior performance in comparing results across diverse engineering design challenges, including pressure vessel, rolling element bearing, cantilever beam, tension/compression spring, and gear train problems. It exhibits lower objective function values and improved statistical measures compared to the original PDO and other MAs. EPDO's innovations suit real-world engineering optimization challenges well, delivering accurate and efficient solutions, particularly in industries with diverse design constraints. A notable improvement in EPDO is its proficiency in handling engineering design problems with numerous inequality constraints. Introducing a penalty function enables graceful adaptation to constraint violations, significantly expanding its applicability

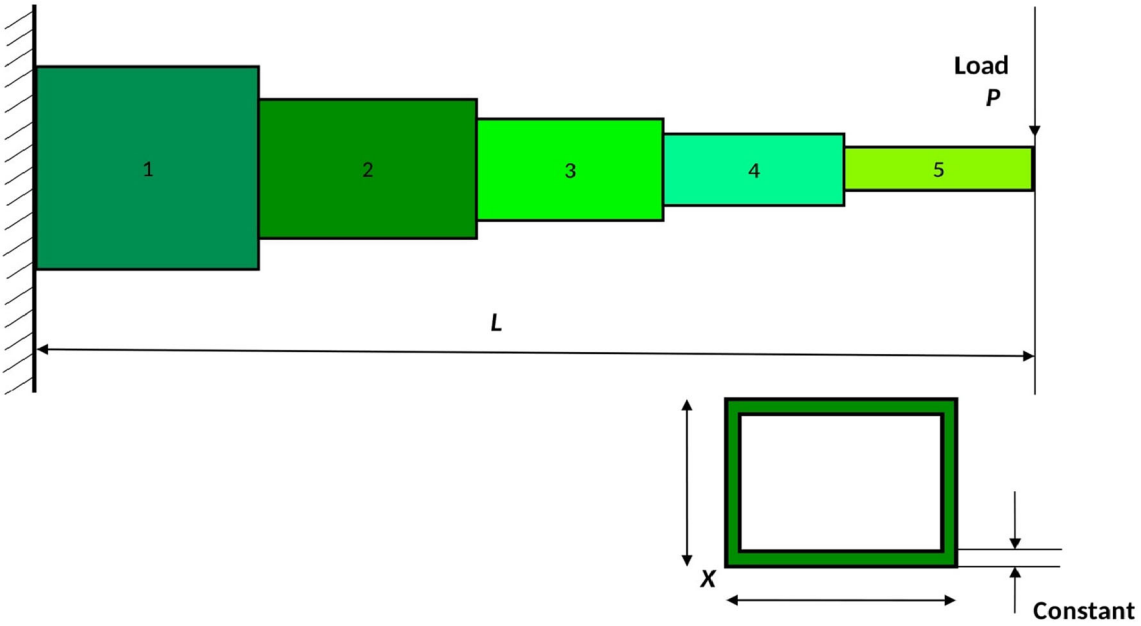

Fig. 14 Structure of cantilever beam

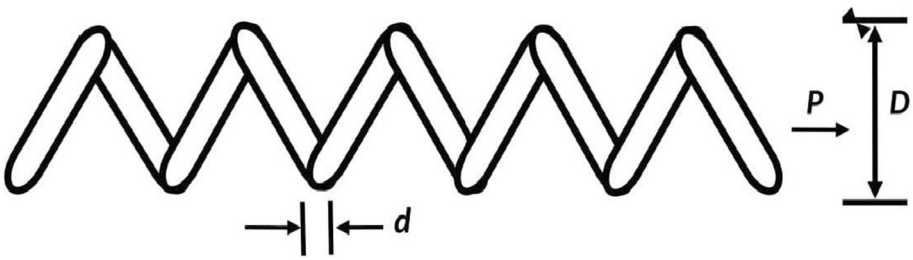

Fig. 15 Tension/compression spring design

to intricate real-world scenarios with stringent constraints. The practical implications of EPDO's enhancements are evident in industries relying on efficient engineering design. By consistently surpassing its predecessor and other algorithms, EPDO emerges as a reliable and robust optimization tool for real-world scenarios emphasizing precision and reliability.

The EPDO algorithm, while showcasing notable advantages, has its limitations. Sensitivity to algorithm tuning poses a challenge, requiring meticulous parameter configuration for optimal performance across diverse problem sets. Additionally, the algorithm's effectiveness is contingent on the quality of initial solutions, making it susceptible to suboptimal convergence if initialization needs to be carefully managed. The variability in EPDO's performance across different optimization problems suggests a degree of problem-specific sensitivity. Although it excels in various benchmarks, its consistency and effectiveness might vary for specific complex, nonlinear, or multimodal functions. These limitations highlight the importance of addressing tuning challenges, improving robustness to diverse initializations, and enhancing performance across a broader range of optimization scenarios to fortify EPDO's reliability and applicability.

Table 24 Comparative results for the gear train design problem

Algorithms	n_A	n_B	n_C	n_D	Best	Worst	AVG	SD
EPDO	46.7	18.8667	17.2333	46.4667	2.70E−12	4.17E−08	7.64E−09	1.24E−08
PDO	47.0667	20.2333	17.9667	43.9667	2.13E−07	2.14E−02	2.20E−03	4.30E−03
SCA	49.4667	19.6	19.8333	50.1333	2.70E−12	2.36E−09	1.02E−09	7.53E−10
WOA	46.3667	17.7667	18.2667	47.1333	2.70E−12	6.51E−09	1.20E−09	1.30E−09
MFO	50.9667	21.7	18.7333	49.9667	2.31E−11	4.47E−08	5.66E−09	9.97E−09
RSA	46.1	21.2667	21.5	49.4667	2.71E−06	4.23E−02	5.30E−03	9.90E−03

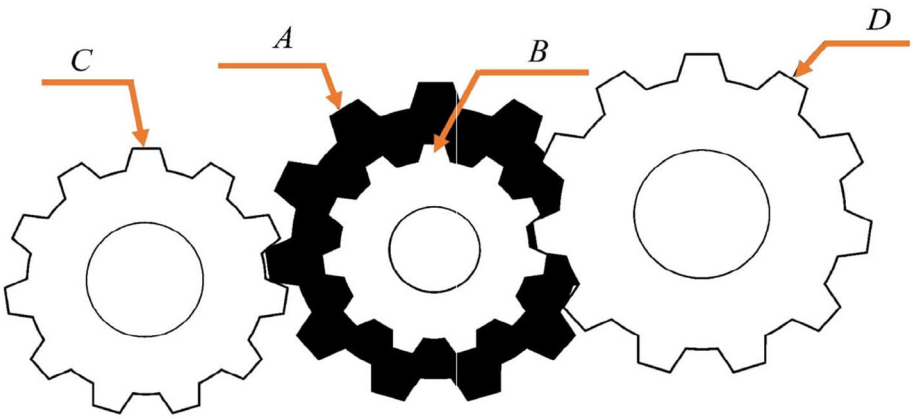

Fig. 16 Gear Train Design

6 Conclusions and plans for the future

In this research article, we have presented the EPDO algorithm as an enhanced variant of the PDO algorithm, addressing its limitations and improving its performance in the exploration and exploitation of global optimization problems. By incorporating the DOL method and a modified Lévy flight technique, EPDO achieves better intensification and diversification capabilities, improving overall optimization performance. To evaluate the effectiveness of EPDO, we conducted extensive experiments using 23 CEC-2017 benchmark test problems and additional benchmark problems. The results demonstrate that EPDO outperforms other state-of-the-art algorithms in terms of global optimization, showing its potential for solving engineering optimization problems and real-world applications. We further validated the performance of EPDO through average rank tests, MARCOS rank tests, and convergence analysis, all of which consistently confirmed its competitive advantage in handling shifting and rotation problems in global optimization.

While the EPDO algorithm demonstrates notable advantages, including improved convergence speed, solution quality, and versatility, several limitations should be acknowledged. EPDO's sensitivity to parameter tuning poses challenges in achieving optimal configurations across diverse problem domains, and its dependency on the quality of initial solutions may lead to suboptimal performance with careless initialization. Additionally, the algorithm's performance may exhibit variability across different optimization problems, particularly complex, nonlinear, or multimodal ones. Addressing these limitations is crucial for further enhancing EPDO's robustness and applicability. Future work should focus on mitigating parameter sensitivity, exploring advanced initialization strategies, adapting the algorithm for specific problem types, investigating hybridization with other techniques, and conducting extensive real-world applications to validate its effectiveness. The algorithm's application in various domains, including manufacturing, logistics, energy systems, financial modelling, healthcare planning, telecommunications network design, and agricultural planning, showcases its potential for optimizing diverse real-world scenarios. Moreover, we recommend exploring its potential for solving specific optimization problems, such as the travelling salesman problem, emission dispatch problem, nonlinear inventory optimization problem [2, 12], portfolio optimization, healthcare resource allocation problem and image segmentation problem. Furthermore, investigating many-objective optimization and conducting theoretical investigations, including Markov process modelling and stability analysis, can provide a stronger theoretical foundation for EPDO, further enhancing its understanding and capabilities.

Funding Open access funding provided by North-West University.

Data availability Data are available from the authors upon reasonable request.

Declarations

Conflict of interest The authors declare that there is no Conflict of interest regarding the publication of this paper.

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

Informed consent Informed consent was obtained from all individual participants included in the study.

References

1. Abualigah L, Elaziz MA, Sumari P, Geem ZW, Gandomi AH (2022) Reptile search algorithm (RSA): a nature-inspired meta-heuristic optimizer. Expert Syst Appl 191:116158
2. Taleizadeh AA, Naghavi-Alhoseiny M-S, Eduardo Cárdenas-Barrón L, Amjadian A (2023) Optimization of price, lot size and backordered level in an EPQ inventory model with rework process. RAIRO-Oper Res 5:803–819
3. Cao D, Xu Y, Yang Z, Dong H, Li X (2022) An enhanced whale optimization algorithm with improved dynamic opposite learning and adaptive inertia weight strategy. Complex Intell Syst 8:767–795
4. Chakraborty I, Kumar V, Nair SB, Tiwari R (2003) Rolling element bearing design through genetic algorithms. Eng Optim 35(6):649–659
5. Coufal P, Hubálovský Š, Hubálovská M, Balogh Z (2021) Snow leopard optimization algorithm: a new nature-based optimization algorithm for solving optimization problems. Mathematics 9(21):2832
6. Dehghani M, Montazeri Z, Hubálovský Š (2021) GMBO: group mean-based optimizer for solving various optimization problems. Mathematics 9(11):1190
7. Dehghani M, Trojovský P (2021) Teamwork optimization algorithm: a new optimization approach for function minimization/maximization. Sensors 21(13):4567
8. Dhal KG, Ray S, Rai R, Das A (2023) Archimedes optimizer: theory, analysis, improvements, and applications. Arch Comput Methods Eng 30(4):2543–2578
9. Dong H, Xu Y, Li X, Yang Z, Zou C (2021) An improved antlion optimizer with dynamic random walk and dynamic opposite learning. Knowl-Based Syst 216:106752
10. Dong W, Kang L, Zhang W (2017) Opposition-based particle swarm optimization with adaptive mutation strategy. Soft Comput 21(17):5081–5090
11. Ezugwu AE, Agushaka JO, Abualigah L, Mirjalili S, Gandomi AH (2022) Prairie dog optimization algorithm. Neural Comput Appl 34(22):20017–20065
12. Gharaei A, Amjadian A, Shavandi A, Amjadian A (2023) An augmented Lagrangian approach with general constraints to solve nonlinear models of the large-scale reliable inventory systems. J Comb Optim 45(2):78
13. Glover F (1994) Tabu search for nonlinear and parametric optimization (with links to genetic algorithms). Discret Appl Math 49(1–3):231–255
14. Hasançebi O, Çarbaş S, Doğan E, Erdal F, Saka MP (2009) Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. Comput Struct 87(5–6):284–302
15. Sirovich IL, Marsden JE (2009) Diffusion and ecological problems: modern perspectives, vol 8. Springer, New York
16. Jamil M, Yang XS (2013) A literature survey of benchmark functions for global optimisation problems. Int J Math Model Numer Optim 4(2):150–194
17. Jaradat G, Ayob M, Almarashdeh I (2016) The effect of elite pool in hybrid population-based meta-heuristics for solving combinatorial optimization problems. Appl Soft Comput J 44:45–56
18. Kang Q, Xiong C, Zhou M, Meng L (2018) Opposition-based hybrid strategy for particle swarm optimization in noisy environments. IEEE Access 6:21888–21900
19. Kareiva PM, Shigesada N (1983) Analyzing insect movement as a correlated random walk. Oecologia 56(2–3):234–238
20. Kashani AR, Camp CV, Rostamian M, Azizi K, Gandomi AH (2022) Population-based optimization in structural engineering: a review. Artif Intell Rev 55(1):1–108
21. Kirkpatrick S, Gelatt CD, Vecchi MP (1983) Optimization by simulated annealing. Science 220(4598):671–680
22. Kumar A, Wu G, Ali MZ, Mallipeddi R, Suganthan PN, Das S (2020) A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. Swarm Evol Comput 56:100693
23. Mahdavi S, Rahnamayan S, Deb K (2018) Opposition based learning: a literature review. Swarm Evol Comput 39:1–23
24. Mantegna RN, Stanley HE (1994) Stochastic process with ultraslow convergence to a gaussian: the truncated Lévy flight. Phys Rev Lett 73(22):2946
25. Mirjalili S (2015) Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm. Knowledge-Based Syst 89:228–249
26. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
27. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput Appl 27(4):1053–1073
28. Mirjalili S (2016) SCA: a Sine Cosine algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
29. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
30. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf Optimizer. Adv Eng Softw 69:46–61
31. Mladenović N, Hansen P (1997) Variable neighborhood search. Comput Oper Res 24(11):1097–1100
32. Osaba E, Villar-Rodriguez E, Del Ser J, Nebro AJ, Molina D, LaTorre A, Suganthan PN, Coello Coello CA, Herrera F (2021) A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems. Swarm Evol Comput 64:100888
33. Pardalos PM, Romeijn HE, Tuy H (2000) Recent developments and trends in global optimization. J Comput Appl Math 124(1–2):209–228
34. Rahnamayan S, Tizhoosh H, Salama M (2008) Opposition-based differential evolution. IEEE Tran Evol Comput 12(1):64–79
35. Rajeswara Rao B, Tiwari R (2007) Optimum design of rolling element bearings using genetic algorithms. Mech Mach Theory 42(2):233–250
36. Reynolds AM, Rhodes CJ (2009) The Lévy flight paradigm: random search patterns and mechanisms. Ecology 90(4):877–887
37. Rojas-Morales N, Riff Rojas MC, Montero Ureta E (2017) A survey and classification of Opposition-Based Metaheuristics. Comput Ind Eng 110:424–435
38. Sahoo SK, Saha AK, Nama S, Masdari M (2022) An improved moth flame optimization algorithm based on modified dynamic opposite learning strategy. Artif Intell Rev 8:2811–2869
39. Sandgren E (1990) Nonlinear integer and discrete programming in mechanical design optimization. J Mech Des, Trans ASME 112(2):223–229
40. Shlesinger MF, Klafter J (1986) Lévy walks versus Lévy flights. In: Stanley HE, Ostrowsky N (eds) On growth and form: fractal and non-fractal patterns in physics, vol 100. Springer, Dordrecht
41. Stević Ž, Brković N (2020) A novel integrated FUCOM-MARCOS model for evaluation of human resources in a transport company. Logistics 4(1):4

42. Stević Ž, Pamučar D, Puška A, Chatterjee P (2020) Sustainable supplier selection in healthcare industries using a new MCDM method: measurement of alternatives and ranking according to COmpromise solution (MARCOS). Comput Ind Eng 140:106231
43. Taleizadeh AA, Amjadian A, Hashemi-Petroodi SE, Moon I (2023) Supply chain coordination based on mean-variance risk optimisation: pricing, warranty, and full-refund decisions. Int J Syst Sci: Oper Logist 10(1):12
44. Tizhoosh HR (2005) Opposition-based learning: a new scheme for machine intelligence. In: Proceedings- international conference on computational intelligence for modelling, control and automation, CIMCA 2005 and international conference on intelligent agents, Web Technologies and Internet
45. Turchin P (1991) Translating foraging movements in heterogeneous environments into the spatial distribution of foragers. Ecology 72(4):1253–1266
46. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
47. Xu Y, Yang Z, Li X, Kang H, Yang X (2020) Dynamic opposite learning enhanced teaching-learning-based optimization. Knowl-Based Syst 188:104966
48. Yang XS (2010) Engineering optimization: an introduction with metaheuristic applications. Wiley, Hoboken
49. Yang XS (2012) Efficiency analysis of swarm intelligence and randomization techniques. J Comput Theor Nanosci 9(2):189–198
50. Yang XS (2020) Nature-inspired optimization algorithms: challenges and open problems. J Comput Sci 46:101104
51. Yang X-S, He X-S (2019) Mathematical foundations of nature-inspired algorithms. Springer, Cham

Authors and Affiliations

Saptadeep Biswas[1] · Azharuddin Shaikh[1] · Absalom El-Shamir Ezugwu[2] · Japie Greeff[3] · Seyedali Mirjalili[4] · Uttam Kumar Bera[1] · Laith Abualigah[5,6,7,8,9,10,11,12]

✉ Absalom El-Shamir Ezugwu
absalom.ezugwu@nwu.ac.za

✉ Laith Abualigah
aligah.2020@gmail.com; abu.aligah@usm.my

Saptadeep Biswas
saptadeepmath.sch@nita.ac.in

Azharuddin Shaikh
Azharuddin@gmail.com

Japie Greeff
japie.greeff@nwu.ac.za

Seyedali Mirjalili
ali.mirjalili@torrens.edu.au

Uttam Kumar Bera
uttam.math@nita.ac.in

[1] Department of Mathematics, National Institute of Technology Agartala, Agartala, Tripura, India

[2] Unit for Data Science and Computing, North-West University, 11 Hoffman Street, Potchefstroom 2520, South Africa

[3] Faculty of Natural and Agricultural Sciences, School of Computer Science and Information Systems, North-West University, Vanderbijlpark, South Africa

[4] Centre for Artificial Intelligence Research and Optimization, Torrens University Australia, Fortitude Valley, Brisbane, QLD 4006, Australia

[5] Artificial Intelligence and Sensing Technologies (AIST) Research Center, University of Tabuk, Tabuk 71491, Saudi Arabia

[6] Computer Science Department, Al al-Bayt University, Mafraq 25113, Jordan

[7] Hourani Center for Applied Scientific Research, Al-Ahliyya Amman University, Amman 19328, Jordan

[8] Applied Science Research Center, Applied Science Private University, Amman 11931, Jordan

[9] Department of Electrical and Computer Engineering, Lebanese American University, Byblos 13-5053, Lebanon

[10] School of Engineering and Technology, Sunway University Malaysia, 27500 Petaling Jaya, Malaysia

[11] College of Engineering, Yuan Ze University, Taoyuan 32003, Taiwan

[12] MEU Research Unit, Middle East University, Amman 11831, Jordan