Alexandria Engineering Journal 120 (2025) 657–686

Contents lists available at ScienceDirect

Alexandria Engineering Journal

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

Original article

Self-adaptive Emperor Penguin Optimizer with multi-strategy parameter adaptation mechanism for complex optimization problems

Othman Waleed Khalid [a,b], Nor Ashidi Mat Isa [a,*], Wei Hong Lim [c]

[a] School of Electrical and Electronic Engineering, Engineering Campus, Universiti Sains Malaysia, Nibong Tebal, Penang 14300, Malaysia
[b] Department of Electromechanical Engineering Techniques, Baquba Technical College (BTC), Middle Technical University (MTU), Baghdad, Iraq
[c] Faculty of Engineering, Technology and Built Environment, UCSI University, Kuala Lumpur 56000, Malaysia

ARTICLE INFO

Keywords:
Emperor penguin optimizer
Swarm intelligence
Self-adaptive
Muti-strategy
Engineering design problems

ABSTRACT

This study introduces the Self-adaptive Emperor Penguin Optimizer (SA-EPO), a new variant that addresses the exploration–exploitation balance limitations of the original EPO due to its statics control parameters. SA-EPO integrates multiple parameter adaptation strategies with unique features and selection probabilities, enabling dynamic modification of control parameters based on individual solution performance. An intelligent selection mechanism within SA-EPO's framework periodically updates the selection probabilities of these parameter adaptation strategies based on their historical effectiveness in enhancing solution quality, ensuring the optimal strategy is consistently employed. SA-EPO's efficacy is validated against 15 leading optimization algorithms through tests on 41 benchmark functions from the CEC2017 and CEC2022. Furthermore, SA-EPO's capability are demonstrated on seven real-world engineering challenges. Comprehensive non-parametric statistical analyses, including Friedman test and Wilcoxon signed rank test, confirm the superior accuracy and convergence speed of SA-EPO across a range of optimization scenarios. The SA-EPO demonstrates substantial performance enhancements compared with the EPO, with improvements of 47.9 % and 52.4 % in Freidman rank for CEC2017 and CEC2022, respectively. Additionally, the Wilcoxon signed rank test reveals a 100 % improvement, indicating a complete advantage over the EPO in all tested scenarios. These findings highlight its potential to drive industrial and process innovation in diverse optimization tasks.

1. Introduction

Optimization is a crucial aspect of various fields, such as engineering, finance, and computer science, aiming to find the best solution under specific constraints while maximising or minimising objective functions. Traditional methods like gradient descent [1], Newton's method [2], and convex optimization [3] often struggle with complex problems due to issues like getting stuck in local optimum, slow convergence or high computational demands. To address these challenges, researchers have developed nature-inspired algorithms, drawing inspiration from phenomena like genetic evolution, animal behavior, and social interaction [4–9]. These algorithms excel in handling nonlinear, non-convex, and multimodal problems without requiring explicit mathematical models, making them ideal for applications with poorly understood dynamics or complex underlying physics.

Several nature-inspired algorithms are available in the literature. These algorithms can be classified into the following categories based on their inspiration source [10]:

a) Evolutionary algorithms (EAs) are nature-inspired optimization methods based on the concept of "survival of the fittest" [11]. By employing genetic operations such as crossover, mutation, and selection, EAs generate improved offspring solutions. EA is inspired by different natural phenomena, including Genetic Algorithms (GA) [12], Genetic Programming (GP) [13], Evolutionary Strategies (ES) [14], Differential Evolution (DE) [15], and Liver Cancer Algorithm (LCA) [16].
b) Human-based algorithms (HBAs) mimic human behavior in problem-solving, enhancing search efficiency through mechanisms inspired by reasoning, learning, teaching, and communication. Examples include the Jaya algorithm [17], Human-Inspired Algorithms (HIA) [18], Teaching Learning-Based Optimization (TLBO) [19], Socio-Evolution and Learning Optimizations (SELO) [20], and Cognitive Behavior Optimization Algorithms (CBOA) [21].

* Corresponding author.
E-mail address: ashidi@usm.my (N.A.M. Isa).

https://doi.org/10.1016/j.aej.2025.02.046
Received 22 October 2024; Received in revised form 10 February 2025; Accepted 13 February 2025
Available online 26 February 2025

c) Physics-Based Algorithms (PBAs) mimic natural physical phenomena that are grounded in physics, mathematics, and chemistry principles. Notable examples include the Sine Cosine Algorithm (SCA) [22], Tangent Search Algorithm (TSA) [23], Simulated Annealing (SA) [24], Electromagnetism-like Optimization (EMO) [25], Quantum-Inspired Evolutionary Algorithms (QIEA) [26], Polar Lights Optimization (PLO) [27], and RIME Optimization Algorithm (RIME) [28].
d) Swarm intelligence (SI) algorithms draw inspiration from the collective behavior of animals, such as predation and hunting, to solve complex problems. Prominent examples include Particle Swarm Optimization (PSO) [29], Artificial Bee Colony (ABC) [30], and Ant Colony Optimization (ACO) [31]. In recent years, several new SI algorithms have been invented including the Whale Optimization Algorithm (WOA) [32], Harris Hawk Optimization (HHO) [33], Horse Herd Optimization Algorithm (HOA) [34], and Fata Morgana Algorithm (FATA) [35].

One notable swarm intelligence algorithm is the Emperor Penguin Optimizer (EPO), proposed by Dhiman in 2018 [36]. EPO is inspired by the huddling behavior of emperor penguins in the harsh Antarctic climate. As a population-based optimization algorithm, it simulates how penguins huddle to conserve heat, with each penguin representing a potential solution in the search space. The core feature of EPO is the huddle mechanism, which determines movements based on the population's mean position, serving as a reference for navigation. This mechanism has effectively addressed various optimization challenges, including engineering design [37], power systems [38], energy consumption [39], biomedical engineering [40], image processing [41], and wireless sensor network [42]. Despite its promising results, the original EPO has weaknesses, particularly in complex scenarios, such as limited exploration capabilities and a tendency to become trapped in local optima [39]. Enhanced EPO variants have been proposed, but balancing exploration and exploitation remains a challenge. A key factor affecting this balance is the three control parameters (f, l, and M) that govern individual penguin movements [36]. Using fixed values for these parameters limit flexibility and adaptability, hindering effective navigation through complex fitness landscapes [43].

Various enhancements to the EPO have been proposed to improve its performance. For instance, Xing [44] incorporated Levy flight and Gaussian mutation into EPO to enhance exploration and convergence rates, yet this adaptation still struggles with local optima entrapment. Jia [41] developed a mutation-based enhancement to increase population diversity within EPO, but this approach often fails to maintain a balance between exploration and exploitation, limiting its global search effectiveness. The Quantum-Based EPO (QEPO) proposed by Min [39] employs quantum principles to improve exploration capabilities but at the cost of increased computational overhead, which may not be suitable for time-sensitive applications. Additionally, Subha [45] tailored EPO for wireless sensor networks, improving convergence through modified huddling behavior. However, this adaptation's specificity restricts its general applicability. Dhiman [37] combined EPO with the Salp Swarm Algorithm (SSA) to improve search efficiency, but this hybridization complicates parameter tuning and raises concerns about the algorithm's generality across diverse optimization challenges.

These enhancements highlight a recurring theme, i.e., while individual modifications address specific EPO limitations, they introduce new challenges, particularly in balancing exploration with exploitation and maintaining general applicability. This issue is further complicated by the use of multiple search operator schemes. Although conceptually appealing, these schemes often introduce considerable complexity. For instance, the enhanced PSO variant [46] incorporates multiple evolutionary strategies and local search operators, enhancing performance but also introducing substantial parameter tuning inefficiencies. The adaptive EA introduced in [47] features a complex configuration involving multiple search methodologies that, while effective, demand substantial computational resources and meticulous management of operator interactions to avoid redundancy. By contrast, a multiple-parameter adaptation scheme offers streamlined implementation by establishing direct mathematical relationships between control parameters and the algorithm's current state, facilitating adaptive adjustments and simplifying design complexities. This approach avoids uncontrolled search behaviors, such as excessive randomness or premature convergence, often resulting from sub-optimal integration in multiple search operator schemes [48]. By fine-tuning control parameters within known limits, the multiple-parameter adaptation scheme ensures stability and predictability, facilitating consistent performance across different conditions and problem sets.

Recognising these challenges, this research proposes an innovative approach to dynamically adjust control parameters for each solution during the optimization process, addressing the need for systematic adaptive adjustments within the EPO framework. Given the varied exploration and exploitation demands across different subregions of complex fitness landscapes, a single-parameter adaptation scheme proves insufficient [49]. Integrating multiple-parameter adaptation schemes could more effectively navigate complex landscapes. However, the challenge lies in selecting the most suitable scheme for each solution, reflecting the dynamic nature of parameter adaptation strategy selection, which varies with the optimization stage and the specific position of each solution in the search space.

This paper introduces the self-adaptive EPO (SA-EPO) to address these challenges by incorporating a robust parameter adaptation mechanism to enhance EPO's optimization efficacy. Unlike the fixed control parameters in the original EPO, SA-EPO employs a suite of parameter adaptation schemes to dynamically adjust control parameters for each solution throughout the optimization process. It also features an intelligent selection mechanism that allows solutions to transition among these schemes, fine-tuning adjustments based on the fitness landscape and optimization phase. Central to SA-EPO is this selection mechanism, governed by a probability updated periodically based on performance history, enhancing solution quality. With its diverse parameter adaptation schemes and intelligent selection mechanism, SA-EPO aims to better balance exploration and exploitation, enabling it to efficiently tackle a broader range of optimization problems. The main contributions of this paper are summarised as follows:

- We introduce a modified EPO variant named SA-EPO, designed to improve the original EPO's performance. SA-EPO achieves a more effective balance between exploration and exploitation through an advanced multiple-parameter adaptation scheme.
- SA-EPO incorporates five distinct parameter adaptation strategies, each characterised by unique features and selection probabilities. These parameter adaptation strategies enable dynamic adjustment of the control parameters for each SA-EPO solution, in either linear or nonlinear ways, throughout the optimization process.
- An intelligent selection mechanism is developed within the proposed multiple-parameter adaptation framework of SA-EPO to periodically refine the selection probabilities of the five candidate parameter adaptation strategies. This refinement is based on their historical success in enhancing solution quality, ensuring the optimal strategy is employed for each SA-EPO solution during the optimization process.
- The efficacy of SA-EPO is validated against 15 state-of-the-art optimization algorithms through a comprehensive evaluation involving 41 benchmark functions, comprising 29 from CEC2017 and 12 from CEC2022. Furthermore, SA-EPO's capability and versatility are demonstrated through its application to seven real-world engineering challenges: Spring Design, Three Bar Truss, Gear Train Design, Cantilever Beam, Pressure Vessel Design, Speed Reducer Design, and Rolling Element Bearing Design, showcasing its potential in diverse real-world contexts.

The remainder of the paper is structured as follows: Section 2 delineates the fundamental principles of the original EPO algorithm and reviews related work. Section 3 elaborates on the methodology of the proposed SA-EPO in detail. Section 4 details the experimental results. Section 5 details the demerits of the study and discusses the behavior of the proposed algorithm. Finally, Section 6 concludes the paper, summarising the findings and outlining future research directions.

2. Literature review

This section offers a comprehensive review of the original EPO and its variants, delving into their search mechanisms, strengths, and weaknesses. It also broadens the scope to include a survey of other optimization algorithms that employ multi-strategy schemes for improving their performance. This survey analysis lays the groundwork for proposing a new EPO variant, which introduces multiple parameter adaptive schemes and utilises an intelligent selection mechanism to enhance optimization outcomes.

2.1. Original emperor penguin optimizer

The emperor penguin, the largest of its kind, exhibits impressive swimming and diving abilities, often foraging and hunting together [44]. The EPO algorithm draws inspiration from their behavior, simulating huddling to conserve heat in Antarctica [37]. During harsh Antarctic winters, they employ a survival strategy known as huddling [50], where each penguin contributes equally to foster collectiveness [51]. A mathematical model based on the huddling behavior of emperor penguins was developed to determine the most effective mover. The huddle is assumed situated on a 2D L-shape polygonal surface. The process steps proceed as follows:

Step 1: Create the huddle and then determine its boundaries.

During huddling, the penguins move to an L-shaped area within the polygon boundary [50]. Wind flow speed and direction are used to determine the huddling boundary. A mathematical formulation of the boundary can be found in Eq. (1).

$$\varphi = \nabla \vartheta \tag{1}$$

where φ represents the gradient of the wind velocity ϑ; ∇ is a gradient operator. A complex potential is reached by integrating the vector β with ϑ as in Eq. (2).

$$\kappa = \vartheta + a\beta \tag{2}$$

where a is an imaginary constant used in the polygon plane function κ; β is a vector that represents the movement direction of each search agent.

Step 2: Determine the huddle temperature.

When the radius of the polygon R is greater than 1, the huddle temperature T is set as 0. Conversely, when R is less than or equal to 1, T is set as 1 [36]. The huddle temperature of emperor penguin can be expressed mathematically by Eqs. (3) and (4) as

$$T = \begin{cases} 0, \& \text{if} R > 1 \\ 1, \& \text{if} R \le 1 \end{cases} \tag{3}$$

$$T' = T - \frac{I_{\max}}{t - I_{\max}} \tag{4}$$

where t, $I_{\max}$, and T' represent the current iteration number, the maximum number of iterations, and the temperature profile around the huddle, respectively.

Step 3: Calculate the distances between huddling penguins.

The emperor penguins' distance from the optimal solution is determined after creating the huddle boundary. A vectors $P_i = [p_{i1}, p_{i2}, \ldots, p_{ij}, .., p_{iD}]$ represents the position vector of the i-th emperor penguin, whereas $P_{best} = [p_{best1}, p_{best2}, \ldots, p_{bestj}, .., p_{bestD}]$ is the position vector of the fittest emperor penguin with best fitness value. At any t-th iteration, define Dis_i as the current distance between the i-th emperor penguin ($P_{i,t}$) and the best-performing emperor penguin ($P_{best,t}$), i.e.,

$$Dis_i = Abs\left(S_F(A) \cdot P_{best,t} - C \cdot P_{i,t}\right) \tag{5}$$

where $Abs(\bullet)$ is an operator used to calculate the absolute value of a vector; subscript i represents with population index of emperor penguin ranges from 1 to N, where N is the population size of EPO; subscript t represents the iteration index ranges from 1 to I_{max}; $S_F(\ )$ is an operator used to determine the social forces of emperor penguin that is responsible for moving towards the direction of the current best search agent (i.e., $P_{best,t}$). Another two vectors, represented as A and C, are also defined to prevent collisions between emperor penguins, calculated using Eqs. (6)–(8):

$$A = \left(M \times \left(T' + P_{grid}(Accuracy)\right) \times Rand(\ )\right) - T' \tag{6}$$

$$P_{grid}(Accuracy) = Abs\left(P_{best,t} - P_{i,t}\right) \tag{7}$$

$$C = Rand(\ ) \tag{8}$$

where $M = 2$ is a movement parameter that maintains the gap between search agents (i.e., emperor penguins) to avoid collision; $P_{grid}(Accuracy)$ represents the grid accuracy of polygon by calculating the absolute difference between the current best emperor penguin and the i-th emperor penguin; $Rand(\ )$ indicates a random number generator with a range of [0,1]. The function $S_F(\ )$ is determined as in Eq. (9);

$$S_F(A) = \left(\sqrt{f.e^{-\frac{t}{l}} - e^{-t}}\right)^2 \tag{9}$$

where $f \in [2, 3]$ and $l \in [1.5, 2]$ are two control parameters that have crucial roles in balancing the exploration and exploitation searches of EPO.

Step 4: Determine and relocate the effective mover.

At the next iteration of $t + 1$, the position vector of each i-th emperor penguin, denoted as $P_{i,t+1}$ is updated using Eq. (10) based on the vector A and distance value Dis_i obtained earlier, where:

$$P_{i,t+1} = P_{best,t} - A \cdot Dis_i \tag{10}$$

The pseudocode of original EPO is shown in Algorithm 1.

Algorithm 1. : Pseudo code for EPO [36]

Inputs: the emperor penguin population $P_i(i \leftarrow 1, 2, \ldots, N)$

Initialize the parameters $T, T', A, C, S_F(\,), R, and\ I_{max}$
for $i \leftarrow 1\ to\ N$ **do** */*Population initialization*/*
  Randomly generate the initial P_i of each *i*-th emperor penguin;
  Calculate the initial fitness of *i*-th emperor penguin as $fit_i =$ **FITNESS** (P_i);
end for
while $t < I_{max}$ **do** */*Iterative search process of EPO*/*
    Randomly generate the radius of polygon $R \leftarrow Rand\,(\,)$;
    Calculate the huddle temperature *T* using Eq. (3);
    Compute the temperature profile around the huddle *T'* using Eq. (4);
    for $i \leftarrow 1\ to\ N$ **do** */*for each i-th emperor penguin*/*
      Calculate $P_{grid}\ (Accuracy)$ using Eq. (7);
      Calculate A and C using Eqs. (6) and (8), respectively;
      Calculate $S_F(A)$ using Eq. (9);
      Calculate Dis_i using Eq. (5);
      Update the position of *i*-th emperor penguin as $P_{i,t+1}$ using Eq. (10).
      Ensure the updated $P_{i,t+1}$ is within the search boundary;
      Calculate the updated fitness as $fit_{i,t+1} =$ **FITNESS** $(P_{i,t+1})$;
      if updated $P_{i,t+1}$ has better fitness value **then**
        Update $P_{i,t}$ and $P_{best,t}$ along with their fitness values;
      end if
    end for
    Update iteration counter as $t \leftarrow t + 1$;
end while

Output: Best emperor penguin P_{best} and its fitness value.

2.2. Related work

Recent years have seen the development of several EPO variants with improved performance. These variants of EPO can generally be classified into two categories: modified EPO, which involves changes to the original algorithm's structure or parameters, and hybrid EPO, which combines EPO with other optimization techniques. Additionally, the literature documents various metaheuristic algorithm variants that are relevant to the proposed algorithm, particularly those utilising a multi-strategy adaptation scheme.

2.2.1. Variants of emperor penguin optimizer

Although the original EPO algorithm has demonstrated success in addressing various optimization challenges, certain limitations have been recognised. To overcome these issues, modifications have been suggested by researchers. A detailed review of EPO's development and

Table 1
Summary of EPO variants.

Year, Ref.	EPO variant	Strengths	Limitations
2020, [44]	IEPO	• Employing the Levy flight, Gaussian mutation, and OBL to improve the update position formula of the individual agent. • Strengthening the connection between agents to avoid falling into the local optima. • Expands the search domain so the best penguin can move to the best solution quickly.	• More complex than the EPO algorithm and difficult to implement. • Requires specific parameter tuning. It is difficult to find the best parameter settings for the algorithm based on the given problems.
2019, [41]	MSEPO	• Generate a certain number of mutation solutions to enrich the diversity of the EPO population. • Enhancing convergence. • Explore the entire search space to reduce the possibility of falling into local optima.	• Computationally expensive. • Sensitive to the initial population.
2020, [39]	QEPSO	• Quantum-based search is utilized to improve exploration capability. • Reduced premature convergence issue. • Improve efficiency, converging close to the optimal design. • Expedite the convergence rate.	• Using the QEPO method does not give fast convergence results in some applications.
2020, [53]	IEPO	• eQuest helps to choose the optimum values for various envelope criteria • Levy flight aids in exploring the search space efficiently by incorporating random steps	• Increasing execution time due to the complex modifications made associated with the eQuest, Levy flight and a local search method. • Limited to less complex search spaces.
2022, [45]	IEPOACP	• Leverage the huddling behavior of EPO to improve convergence. • Handled the balance between the local and global search.	• Sensitive to the initial values of parameters.
2020, [54]	MEPC1, MEPC2	• Enhance ability to find optimal or near-optimal solutions using Archimedean and hyperbolic spiral movements. • Avoid falling into local optima.	• Converged too fast due to the rapid changes of agents' movement. • Incorporating new movement patterns might introduce additional complexity to the algorithm.
2022, [43]	FA-EPO	• Using FIS to adjust the EPO parameters (f and l) to adapt to the complexity and size of given problems. • Better robustness and scalability.	• The performance heavily depends on the quality of the initial population. • The effectiveness of complex and high-dimensional problems is yet to be tested.
2021, [37]	ESA	• SSA enhanced search capability of EPO in terms of the quality of the solutions. • Outperforms many algorithms in terms of scalability, convergence, and accuracy.	• Expensive computation cost due to the involvement of SSA algorithm. • The effectiveness of ESA might depend heavily on specific parameter settings.
2021, [56]	H-EPC	• EPC integrated with genetic crossover and mutation operators to achieve a better optimal solution. • Utilized the crossover operator that combines solutions to create a new solution. • Increase randomness by using mutation operator.	• Computationally expensive. • Complexity increased due to the involvement of genetic crossover and mutation operators.
2020, [55]	CEPO	• Reduces the influence of the wrong initial estimation to gene selection and classification. • Prevent getting trapped in local optima.	• Only applicable for binary-class data sets. • Slow convergence rate. • Increased computational cost.
2022, [57]	EPO-ASO	• ASO improved performance of the EPO algorithm. • Utilizes the strengths of both EPO and ASO algorithms, and it can find good solutions more quickly than the EPO algorithm alone.	• The search process only relied on the best local positions.
2023, [58]	C-EPO	• Cultural algorithm boosted the exploitation capabilities of EPO. • Less sensitive to parameter tunning.	• Trapped in local optima when the algorithm focused on exploitation. • Computational complexity is high.
2019, [59]	EPO-SVM	• Exploitation performance of EPO is boosted by SEO. • Better classification accuracy than EPO algorithm because it combines the strengths of both the EPO and the SEO algorithm.	• Computational overhead is high. • Requires tuning of the weighting factor. • The algorithm might be trapped in local optima.
2022, [60]	EPO-BES	• BES algorithm improves the quality of the solutions in terms of convergence.	• High sensitivity to parameter settings since extensive experimentations and tuning techniques are required to find the optimal parameter values.
2021, [61]	HEPSO	• Exploit the search space effectively because the PSO particles are attracted to their personal best positions as well as the global best position. • Maintain diverse populations, reducing the chances of getting stuck in local optima.	• Computation complexity increased due to the fusion of EPO and PSO algorithms. • Increased execution time.

its subsequent variants is provided in [52]. Table 1 offers a comparative analysis of the strengths and weaknesses of each EPO variant. This section delves into the specifics of these variants, elucidating their conceptual frameworks, benefits, and limitations as follows:

2.2.1.1. Modified emperor penguin optimizer. In [44], Xing introduced a modified version of the EPO algorithm, incorporating Levy flight, Gaussian mutation, and Opposition-Based Learning (OBL) to enhance the randomness in the position update mechanism for each penguin. The modifications aim to enhance the jumping ability of individual penguins and strengthen their interconnections, reducing the likelihood of entrapment in local optima. Levy flight broadens the search range, facilitating quicker navigation toward the global optimum by the leading penguin. Gaussian mutation was implemented to substitute the original genetic information with random Gaussian values, promoting faster convergence. OBL was also employed to refine the solution quality by concurrently assessing a candidate solution and its counterpart, thus yielding an optimized result. Tang [53] developed an improved EPO method to optimize residential building envelopes, aiming to conserve energy and reduce overall costs. This method integrated the EPO algorithm with the eQuest tool, incorporating Levy flight and a local search strategy based on incidental walks to improve convergence variability and efficiency. However, this EPO variant was tailored to specific design parameters with predetermined values, indicating that adaptation to varied scenarios or inclusion of additional design factors might necessitate significant modifications.

Jia [41] proposed a modification to the EPO algorithm by generating a set number of mutated solutions to enhance population diversity and improve the algorithm's global search capabilities. The study recommended adjusting the huddle temperature, influenced by the emperor

Table 2
Meta-heuristic algorithms with multi-strategy adaptation scheme.

Year, Ref.	Algorithm	Adopted strategies
2023, [63]	SaBO	Inclusion of memory, past experiences, and novel repulsion-based learning for the parameters' updating using strategies: Strategy 1: Gaussian Mutation with DE. Strategy 2: Gaussian Mutation. Strategy 3: DE-like Operation. Strategy 4: Hybrid Operation (Combination of Gaussian Mutation and DE-like Operation).
2023, [64]	SdSCA	Multi-strategy, self-adaptive technique using pool of many strategies including: Strategy 1: Sine-Cosine Algorithm (SCA). Strategy 2: Differential Variation. Strategy 3: Differential Variation with global best solution. Strategy 4: Custom Differential Variation.
2022, [65]	SDPSO	Dynamic multi-strategy consists of different search strategies: Strategy 1: CLPSO. Strategy 2: LIPS. Strategy 3: UPSO. Strategy 4: LDWPSO.
2022, [62]	MSEFA	Multi-strategies and selective ensemble, with search strategies in the pool: Strategy 1: Broad Learning Strategy. Strategy 2: Elite Learning Strategy. Strategy 3: Coordinated Learning Strategy.
2023, [72]	ESO	Enhancement strategies including: Strategy 1: Parameters Dynamic update strategy. Strategy 2: Sine–cosine composite perturbation factors. Strategy 3: Tent-chaos & Cauchy mutation.
2022, [73]	MS-DE	Combination of different characteristic mutation strategies and crossover operators: Strategy 1: Mutation Strategy 1 (scheme1). Strategy 2: Mutation Strategy 2 (scheme2). Strategy 3: Selection of Mutation Schemes. Strategy 4: Adaptive Shift Strategy for Crossover Rate.
2022, [74]	SaWDE	The strategy pool includes ten types of mechanisms employed in evolution. The author named those mechanisms as EnS1 to EnS10, where each strategy contains three different single candidate mutation scenarios.
2022, [75]	HAGCS	The algorithm utilizes strategies Strategy 1: Gradient-based local random walk. Strategy 2: Gradient-Based Local Optimization. Strategy 3: Two self-adaptation and diversity promotion strategies.
2023, [77]	MEWOA	Population is divided into three equal-sized sub-populations and strategies are assigned using different mechanisms: Strategy 1: For exploratory sub-population and global search. Strategy 2: For exploitative sub-population and local search. Strategy 3: For modest sub-population randomly explores or exploits the search space. Strategy 4: For population evolution strategy to improve global optimization ability and avoid local optimum.
2013, [78]	SHADE	Integrates multiple differential mutation strategies, dynamically selecting them based on historical success rates to focus on the most effective approaches. It incorporates a memory mechanism that retains information on optimal parameter settings.
2009, [79]	JDE	Features an adaptive control mechanism for control parameters, allowing for self-adjustment. It also employs a multi-population approach with an aging mechanism, enabling the algorithm to diversify its search by maintaining multiple solution pools.
2009, [80]	JADE	An adaptive DE variant with an optional external archive that maintains diversity and retains valuable historical information. It employs the DE/current-to-p-best mutation strategy, improving the exploration of search space and adaptability to dynamic environments.

distance, to facilitate comprehensive exploration of the search space and minimise the risk of converging to local optima. Min [39] introduced a novel approach known as QEPO to optimize chiller loading operations for reduced power consumption. This technique incorporates principles from quantum theory to enhance the performance of the EPO algorithm. By leveraging quantum-based search methods, the exploration capabilities of EPO are significantly improved, yielding favourable outcomes while preventing premature convergence. Subha [45] introduced a modified EPO for application in wireless sensor networks (WSN). This modification exploits the huddling behavior of the EPO algorithm to enhance its convergence rate. By implementing a huddle formation technique, the population is segmented into smaller groups, facilitating more efficient exploration and exploitation. Additionally, the adoption of an effective mover relocation algorithm contributes to the overall performance enhancement of the EPO. These modifications have led to significant improvements in the efficiency and operational lifespan of WSN.

In recent research by Wahdan [54], two variations of the Emperor Penguin Colony (EPC) algorithm, namely, modified EPC1 (MEPC1) and modified EPC2 (MEPC2), were developed. The primary distinction between these modifications lies in their approach to spiral-like movement. Specifically, the MEPC1 algorithm employs an Archimedean spiral-like pattern, while the MEPC2 algorithm utilises a hyperbolic spiral-like trajectory. In a study by Abdul Kader [43], an improved EPO algorithm for global optimization challenges was introduced. The study advocates for the incorporation of a fuzzy inference system (FIS) to dynamically adjust the control parameters of the EPO algorithm (i.e., f and l), in response to the problem's complexity and size. This enhanced version of EPO utilises FIS to modulate these control parameters based on three criteria: search quality, success rate, and diversity. In [55], Baliarsingh introduced an innovative approach for microarray cancer classification, integrating Fisher score, ReliefF, Extreme Learning Machine (ELM), and a novel Chaotic EPO (CEPO). CEPO enhances the accuracy of ELM by optimizing hyperparameter selection. Comparative analysis with existing methods demonstrated improvements across various datasets. However, this method faces challenges such as slow convergence, and its applicability has been tested solely on binary-class data.

2.2.1.2. Hybrid emperor penguin optimizer. One of the most notable variant categories involves combining EPO with one or more methods from other meta-heuristic techniques, integrating additional search operators to enhance the exploration and exploitation capabilities. To date, EPO has been successfully hybridised with various meta-heuristic methods to create more robust hybrid algorithms.

Dhiman [37] introduced a hybrid bio-inspired algorithm, termed

ESA, by combining the EPO with SSA. ESA leverages the huddling behavior of EPO to augment its search efficiency and offset the computational demands of SSA's parameters. In [56], Sasan introduced a hybrid algorithm for community detection, combining the original EPC algorithm with genetic crossover and mutation operators. This hybridisation facilitates a balance between exploration and exploitation, enhancing the algorithm's ability to achieve optimal solutions. The study introduced two novel operators: a crossover operator that merges two EPC solutions to generate a new one, and a mutation operator that randomly adjusts the parameters of an EPC solution. Angel [57] introduced a hybrid algorithm designed to optimize load balancing, enhance security, and reduce energy consumption in WSN. This approach synergises the EPO with the Atom Search Optimization (ASO) algorithm. Although EPO excels in identifying optimal solutions, its convergence rate can be slow. Conversely, ASO is effective in making local improvements but may become trapped in local minimum. The hybrid method leverages the strengths of both algorithms to achieve superior solutions more rapidly than when using EPO alone. The effectiveness of this technique is assessed based on various metrics, including delivery ratio, network lifetime, overhead, energy consumption, throughput, drop rate, and delay.

Mansour [58] introduced an energy-efficient resource scheduling method utilising Cultural EPO (C-EPO) in a green cloud computing environment. The C-EPO algorithm merges the strengths of the cultural algorithm and the EPO, enhancing the latter's exploitation capabilities. This integrated approach follows a systematic procedure: initialising the penguin population, evaluating each individual's fitness, selecting the top performers for the elite group, applying the cultural algorithm to this group, and finally applying the EPO algorithm to the entire population. The C-EPO algorithm incorporates a specifically designed fitness function to optimize resource scheduling in data centres, aiming to reduce operational and maintenance costs, energy consumption, and heat emission. Baliarsingh [59] introduced a hybrid algorithm that integrates the EPO with Social Engineering Optimization (SEO). In EPO, the exploitation process is determined by calculating the distance between the emperor penguin and the best-performing search agent. This study enhances the EPO by incorporating SEO to fine-tune the neighbourhood regions of the best-performing search agent, thereby improving the results. The methodology involves using EPO to identify the optimal solution, followed by SEO to refine this solution. A novel fitness function was developed, combining classification accuracy with a penalty term for feature selection to mitigate overfitting and reduce feature space dimensionality.

Bagirathan [60] introduced a hybrid routing method for mobile ad-hoc networks named EPO-BES, which merges the Bald Eagle Search (BES) with the EPO. The hybridisation occurs in two phases. Initially, EPO identifies a set of viable paths between the source and destination nodes, utilising a population of penguins where each penguin symbolises a potential path. These penguins traverse the search space, with their positions updated according to their fitness. Subsequently, the BES algorithm refines the paths determined by EPO, starting with an existing path and seeking local enhancements within the search space. Gupta [61] introduced a hybrid algorithm named HEPSO, which merges the EPO with PSO to enhance sensor node placement in underwater acoustic sensor networks. EPO is utilised to calculate the step size for sensor node movement and to balance the exploration and exploitation within the search space, ensuring effective network area coverage. Conversely, PSO determines the movement direction of the sensor nodes, harnessing the swarm's collective intelligence to steer the nodes towards optimal redeployment strategies. The integration of EPO's step size determination and PSO's directional guidance in the HEPSO algorithm capitalises on the strengths of both methods.

2.2.2. Multi-strategy adaptation scheme

Previous research indicates that the performance of metaheuristic algorithms varies with different problems and stages of the optimization process. An algorithm may excel in certain scenarios while underperforming in others. Its performance can also fluctuate even within the same problem at different stages. To address these challenges effectively, robust mechanisms must be designed to adjust the exploration and exploitation capabilities of an algorithm according to the problem type and optimization stage. The adaptation of parameters or search strategies becomes crucial to modify the algorithm's behavior in response to changes in the optimization search space. The reliance on a single parameter adaptation or search strategy is no longer adequate; instead, multi-strategy schemes are essential to address complex real-world optimization problems. These schemes can significantly improve search performance by employing distinct parameter adaptation or search strategies for global search, local search, and balancing both. However, the success of a multi-strategy algorithm hinges on the timely and appropriate selection of each parameter adaptation or search strategy [62]. A sophisticated decision-making process, informed by historical performance, learning periods, and performance metrics, is necessary to select the optimal parameter adaptation or search strategy from the pool. Table 2 presents some of the related works of metaheuristic algorithms with multi-strategy adaptation schemes.

Das [63] introduced a Self-Adaptive Bonobo Optimizer (SaBO), a variant of the Bonobo Optimizer that incorporates multiple search operators to enhance performance. SaBO integrates features such as memory, past experiences, and repulsion-based learning for dynamic search operator selection. This allows the algorithm to adjust its strategies in line with the optimization progress, facilitating the avoidance of local optima and efficient exploration of the search space. The four employed strategies include Gaussian mutation with DE, Gaussian mutation alone, DE-like operation, and a hybrid strategy combining Gaussian mutation and DE-like operation. Akay [64] introduced a Multi-Strategy Self-Adaptive Differential Sine–Cosine Algorithm (SdSCA), which employs a diverse strategy pool, enabling the selection of the most effective strategies for yielding improved solutions. The strategy delivering the most satisfactory outcomes is prioritised for subsequent use, thereby mitigating the algorithm's reliance on a single approach, and offering a more robust SCA variant applicable to a broader range of problems. The four implemented strategies include the standard sine-cosine algorithm, differential variation, differential variation incorporating the best global solution, and a custom differential variation.

Liu [65] developed a PSO variant named Strategy Dynamics PSO (SDPSO) to address complex optimization challenges. SDPSO integrates four distinct search strategies: Competitive Learning PSO (CLPSO) [66], Local Informed Particle Swarm (LIPS) [67], Uniform PSO (UPSO) [68], and Linearly Decreasing Weight PSO (LDWPSO) [69]. The algorithm employs a dynamic selection mechanism for search strategies [70], utilising the principles of replicator dynamics and Evolutionarily Stable Strategy (ESS) from evolutionary game theory. A critical aspect of SDPSO is managing each strategy's adoption probability, which is influenced by player interactions. Strategies that perform better gradually evolve and become predominant. Peng [62] introduced a Multi-Strategy Firefly Algorithm (MSEFA) with selective ensemble designed to enhance performance. MSEFA utilises different search strategies at various stages of the search process, aiming to balance exploration and exploitation. The algorithm encompasses three distinct search strategies within its strategy pool: a broad learning strategy for global search, an elite learning strategy for local search, and a coordinated learning strategy to achieve balancing between global and local searches [71], thereby boosting the algorithm's search performance. MSEFA also employs a selective ensemble approach, creating a priority roulette selection method that identifies and applies the most appropriate search strategies during different phases of the search, facilitating an optimal balance of strategies and yielding improved outcomes.

In advancing multi-strategy optimization, Yao [72] developed the Enhanced Snake Optimizer (ESO) algorithm, which integrates opposition-based learning to improve the global optimization capability

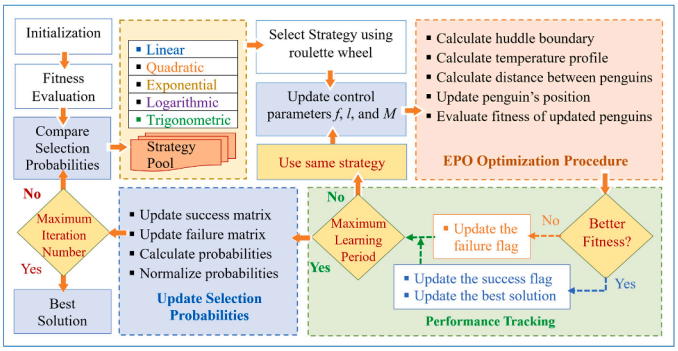

Fig. 1. Overall mechanism of the proposed SA-EPO algorithm.

of the original Snake Optimizer (SO) algorithm. ESO features dynamic update strategies that refine exploration and exploitation, including dynamic parameter updates, sine-cosine composite perturbation factors, and tent-chaos and Cauchy mutation mechanisms. These updates aim to alleviate the limitations of static values in maintaining the exploration-exploitation balance in SO. Specifically, the sine–cosine perturbations address optimization challenges related to dependency on optimal individual positions. ESO also mitigates the issue of offspring prematurely replacing original individuals in SO by incorporating tent-chaos and Cauchy mutations. These enhancements collectively improve the algorithm's fitness evaluation process, thereby enhancing the robustness and adaptability of ESO.

Peng [73] proposed a variant of the DE algorithm named Multi-Strategy Co-Evolutionary DE (MS-DE) to tackle complex optimization problems. This approach features an adaptive selection mechanism that integrates various mutation strategies and crossover operators to simultaneously process continuous and discrete variables. A statistics-based local search method is tailored for discrete variable optimization to boost efficiency and adaptability. Recognising the sensitivity of original DE to parameter configurations, MS-DE discards fixed strategies in favour of an adaptive control mutation strategy and parameter settings. These preferences are designed to match the mutation strategies' characteristics with the diverse requirements of different problems. MS-DE employs an adaptive selection mechanism, using the roulette method to choose from three distinct mutation strategies and two crossover operators, with crossover probability dynamically managed via an adaptive shift strategy. In [74], Wang introduced a Self-Adaptive Weighted DE algorithm, termed (SaWDE), for large-scale feature selection. Initially, a multi-population mechanism was utilised to increase population diversity. Subsequently, a self-adaptive mechanism selects various strategies from a pool, leveraging historical data to accommodate the diverse characteristics of the datasets. To pinpoint critical features, a weighted model is constructed, allowing SaWDE to produce optimal feature selection solutions. The strategy pool comprises 10 evolutionary mechanisms, labelled EnS1 to EnS10, each encompassing three distinct single-candidate mutation scenarios.

Fournier [75] introduced the Hybrid Self-Adaptive Gradient-based Cuckoo Search (HAGCS), which integrates a gradient-based local random walk for initial exploration and subsequent solution refinement. To mitigate the risk of premature convergence associated with gradient-based methods, HAGCS incorporates self-adaptation and diversity enhancement strategies. Its search mechanisms, inspired by the nature-based cuckoo search, feature self-adjusting control parameters informed by the objective function's gradient. The algorithm employs a multi-strategy approach to improve exploration, including Levy flights, DE, the original local random walk, gradient-based local random walk, and deterministic local optimization, alongside promoting parameter diversity [76]. HAGCS is also enhanced with advanced features like a linear population-reduction strategy and a specialised boundary-handling mechanism, further augmenting its capability to tackle a wide array of optimization problems. In [77], Shen introduced the Multi-Population Evolution Whale Optimization Algorithm (MEWOA). The approach begins by dividing individuals into three equal-sized subpopulations based on their fitness: exploratory, exploitative, and modest. Each subpopulation follows distinct search strategies: the exploratory subpopulation conducts global searches, the exploitative subpopulation undertakes local searches, and the modest subpopulation randomly alternates between exploration and exploitation. To enhance global optimization capabilities and prevent convergence to local optima, a population evolution strategy was implemented in MEWOA.

Tanabe [78] proposed the Successful Hierarchical Adaptive DE (SHADE) algorithm, incorporating multiple strategies to enhance performance in complex optimization landscapes. SHADE features a successful strategy adaptation mechanism, dynamically selecting different mutation strategies based on their historical success rates. This allows SHADE to consistently apply the most effective methods throughout the optimization process. Additionally, SHADE incorporates a memory mechanism to retain optimal parameter settings, enabling adaptive adjustments of control parameters such as mutation factor and crossover rate. This hierarchical approach balances exploration and exploitation, enhancing convergence rates and positioning SHADE as a robust solution for diverse engineering and applied sciences challenges. Janez [79] proposed a novel Self-Adaptive Differential Evolution Algorithm (JDE) tailored for dynamic optimization environments. JDE's adaptive control of differential mutation factor and crossover rate allows automatic adjustments during the optimization process. Moreover, its multi-population strategy with an ageing mechanism enhances exploration and mitigates stagnation by maintaining multiple solution pools. Zhang [80] introduced an adaptive DE algorithm with an optional external archive (JADE), advancing the DE framework by addressing typical limitations related to parameter tuning and population diversity. JADE introduces the DE/current-to-p-best mutation strategy, expanding solution selection for broader exploration. The incorporation of an external archive maintains diversity and preserves valuable historical information, essential in dynamics optimization scenarios. JADE's adaptive parameter control mechanism minimises reliance on user input, broadening its applicability across various optimization problems.

3. The proposed SA-EPO algorithm

Different stages of optimization may benefit more from employing a variety of strategies for parameter adjustment than from relying on a single strategy. To enhance the performance of the original EPO, the SA-EPO algorithm has been developed in this study by introducing a novel self-adaptive technique that utilises a multiple-parameter adaptation scheme. This approach involves dynamically adjusting the control parameters of each search agent (i.e., emperor penguin) based on their fitness. It incorporates a candidate strategy pool consisting of multiple effective strategies, each with unique characteristics for parameter tuning. An intelligent selection mechanism has also been devised within the SA-EPO framework to determine the optimal parameter adaptation scheme for each emperor penguin by considering the solution's fitness, historical performance, and the track record of each candidate parameter adaptation strategy. The primary goal of this self-adaptive process is to balance exploration and exploitation throughout the search process to yield optimal solutions. Fig. 1 illustrates the proposed SA-EPO and its underlying mechanisms, providing a detailed view of the algorithm's workflow, emphasising the critical steps in the multiple-parameter adaptation scheme and intelligent selection mechanism.

3.1. Motivations for designing multiple-parameter adaptation schemes

In contrast to the prevalent use of multiple search operator schemes that aim to balance exploration and exploitation, this study introduces a multiple-parameter adaptation scheme in the proposed SA-EPO, driven

by several potential benefits. Multiple search operator schemes, while intuitive, pose significant design and implementation challenges. They require the definition of multiple operators and a deep understanding of their interactions and behaviors, which complicates algorithm development. For instance, Rosso [46] enhanced a PSO variant by incorporating evolutionary strategies and local search operators. Although this improved performance in constrained environments, it significantly increased complexity and risked inefficiencies in parameter tuning. Similarly, Saber [47] introduced an EA with an adaptive configuration employing multiple methodologies, each utilising several search operators. Despite its effectiveness, this approach demanded extensive computational resources and meticulous management of operator interactions to avoid redundancy. Zheng [81] combined multiple candidate generation with a Multiple Strategy DE (MSDE) and adaptive similarity selection. However, this integration was complex, requiring an effective individual selection mechanism and careful parameter calibration to maintain a balance between convergence and diversity. By contrast, the proposed multiple-parameter adaptation scheme simplifies implementation by establishing mathematical relationships between control parameters and the algorithm's current state, such as solution fitness. This mathematical foundation allows for adaptive adjustments within predefined bounds, reducing design complexity and enhancing manageability.

Another significant drawback of multiple search operator schemes is their potential to produce uncontrolled search behavior when suboptimally integrated, leading to conflicting search directions or premature convergence. For instance, the enhanced PSO [46] may stagnate in infeasible regions without finely tuned local search mechanisms. Similarly, inadequate tuning of adaptive parameters in MSDE [81] could compromise the effectiveness of its dual-stage design by either limiting diversity in early stages or impeding convergence in later stages. Moreover, the nature of the EA algorithm in [47], introduced the potential for uncontrolled search behaviors. Rapid adjustments based on performance feedback can lead to erratic search patterns, causing sudden shifts in exploration strategies that lack meaningful context. By contrast, the multiple-parameter adaptation scheme directly influences algorithmic search behavior by fine-tuning control parameters within established limits, thus preventing premature convergence and ensuring sufficient exploration of the solution space. Additionally, the inherent variability and complexity of managing multiple search operator interactions across diverse problem sets can lead to unpredictable results and impact the robustness and reliability of the algorithms. For instance, the interaction of mutation strategies in MSDE frameworks [81] and the adaptive evolutionary mechanisms in the enhanced PSO may suffer from inconsistent convergence patterns in specific types of problems. By contrast, the proposed multiple-parameter adaptation scheme simplifies the adaptive control by focusing on a select set of control parameters, ensuring stability and predictability. By maintaining control within predefined boundaries and understanding the impact of each parameter, SA offers a robust and effective alternative for addressing diverse optimization challenges, ensuring consistent performance across various conditions and problem sets.

3.2. Control parameters on SA-EPO

In the proposed SA-EPO, the control parameters $f, l,$ and$M$ are confined within predefined ranges to effectively govern the exploration and exploitation behaviors of the algorithm [44]. Specifically, these control parameters are set within the following ranges: $f \in [2, 3]$, $l \in [1.5, 2]$ and $M \in [1, 1.5]$. The adjustment of these control parameters is adaptively guided by the fitness values of each emperor penguin (i.e., search agent) of SA-EPO. As the emperor penguins approach the optimal solution, the focus shifts from exploration to exploitation to refine the promising solutions and increase the likelihood of efficiently reaching the global optimum.

The parameter f influences the exploration of algorithms by adjusting the perturbation applied on the emperor penguin's position, where a higher value of f facilitates stronger exploration across the search area and a lower value diminishes it. When the emperor penguins are nearer to optimal solution regions, reducing f helps to focus on fine-tuning promising areas, potentially leading to superior solutions. The parameter l governs exploitation by adjusting the learning rate [43]. Decreasing l intensifies exploitation by concentrating efforts on promising areas, while increasing l promotes broader exploration. For the superior emperor penguins closer to optimal solution regions, lowering l enhances their ability to exploit these areas effectively, facilitating more effective convergence towards the global optimum. Parameter M regulates the movement dynamics among emperor penguins, particularly their collision avoidance behavior [44]. An increased M value encourages exploratory movement through intensified avoidance behavior, while a decreased M value promotes closer interaction among emperor penguins, thus enhancing exploitation efforts in promising areas. For emperor penguins with better fitness and closer proximity to the optimal solution, reducing M emphasises exploitation, facilitating a more effective convergence toward the global optimum.

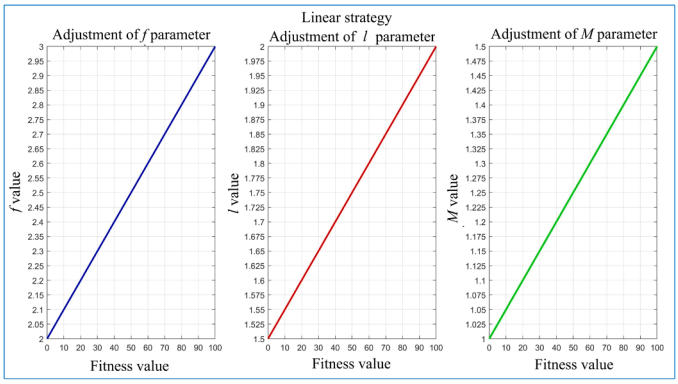

Fig. 2. Variation of control parameters with fitness range [0 100] under linear strategy.

For each i-th emperor penguin, the degree of adjustment for all control parameters (i.e., f, l and M) is determined by the normalised fitness differences, a key metric that gauges the proximity of each i-th emperor penguin to optimality. This metric is calculated by subtracting the fitness value of the i-th emperor penguin from the best (i.e., minimum) fitness among all search agents, as expressed in Eq. (11):

$$diff_i = \quad fit_i - \min_{Fit} \tag{11}$$

where fit_i represents the fitness of the i-th emperor penguin; $\min_{Fit}$ denotes the best (i.e., minimum) fitness value among all search agents; $diff_i$ represents the fitness difference between the i-th emperor penguin and the current best-performing emperor penguin. We denote R_{Fit} as the range of fitness values in the search space and it is determined using Eq. (12):

$$R_{Fit} = \quad \max_{Fit} - \min_{Fit} \tag{12}$$

where $\max_{Fit}$ corresponds to the worst (i.e., maximum) fitness value among all search agents. Finally, the normalised fitness difference for each i-th emperor penguin, denoted as $Norm_i$, is calculated using Eq. (13):

$$Norm_i = \quad \frac{diff_i}{R_{Fit}} \tag{13}$$

This normalisation scales the fitness difference to a value between 0 and 1, significantly influencing the exploration and exploitation behaviors of SA-EPO through adaptive adjustment to the three control parameters. When the i-th emperor penguin is closer to the optimal solution, it typically exhibits a superior (i.e., lower) fitness value. Consequently, a smaller value of $Norm_i$ is assigned to reduce all control

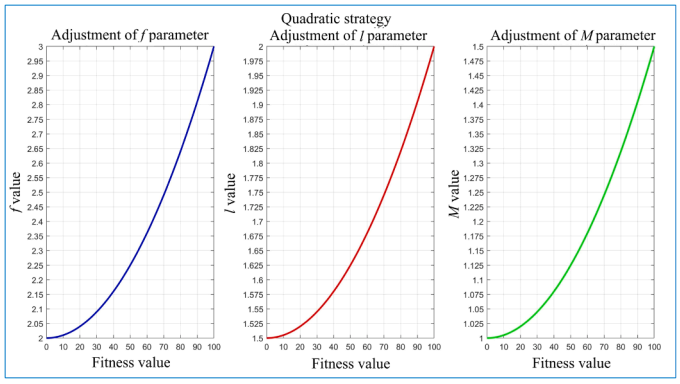

Fig. 3. Variation of control parameters with fitness range [0 100] under quadratic strategy.

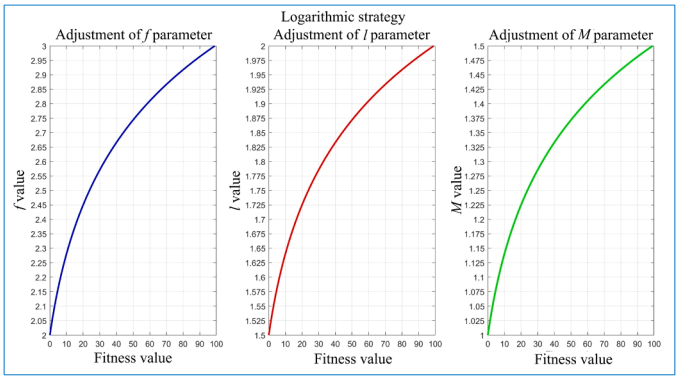

Fig. 5. Variation of control parameters with fitness range [0 100] under logarithmic strategy.

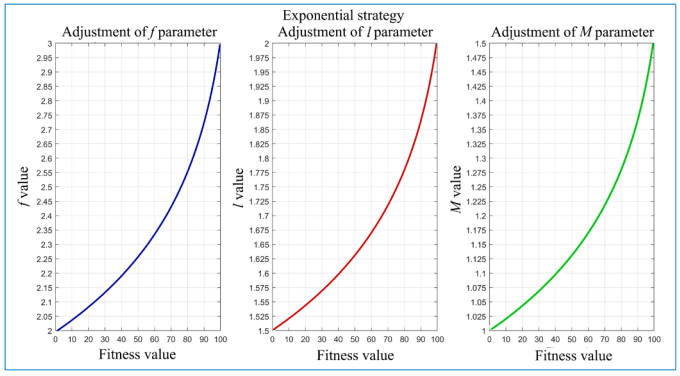

Fig. 4. Variation of control parameters with fitness range [0 100] under exponential strategy.

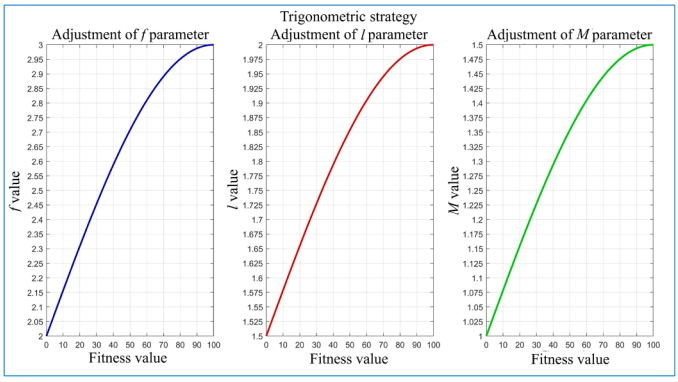

Fig. 6. Variation of control parameters with fitness range [0 100] under trigonometric strategy.

parameters for this search agent, enhancing its focus on refining and exploiting the most promising solutions to efficiently converge towards the global optimum. Conversely, if the *i*-th emperor penguin is further from the global optimum, it usually has an inferior (i.e., larger) fitness value. In this case, a larger value of $Norm_i$ is assigned to increase the control parameters, promoting a broader search and exploration in the solution space to potentially discover superior solutions.

The subsequent subsections will detail how the $Norm_i$ value is used to simultaneously fine tune all three control parameters (f, l, and M) of each *i*-th emperor penguin, using a parameter adaptation strategy determined by the intelligent selection mechanism. This synchronised adjustment ensures that changes to one parameter are coherent with the others, thereby preventing operational complications and enhancing the efficiency of SA-EPO. For clarity and simplicity, the control parameter of each *i*-th emperor penguin subject to adjustment is denoted as $x_i \in \{f_i, l_i, M_i\}$, with $x_{\max} \in \{f_{\max}, l_{\max}, M_{\max}\}$ and $x_{\min} \in \{f_{\min}, l_{\min}, M_{\min}\}$ representing its upper and lower limits, respectively.

3.3. Candidates of parameter adaptation strategies

The proposed SA-EPO algorithm incorporates a multiple-parameter adaptation scheme, featuring a candidate pool of five distinct parameter adaptation strategies: linear, quadratic, exponential, logarithmic, and trigonometric adjustments. These parameter adaptation strategies range from linear adjustments, which change the control parameters proportionally based on the normalised fitness difference of each emperor penguin, to more complex nonlinear adjustments. Linear adjustments provide a straightforward and stable method to fine-tune the control parameters, ensuring gradual changes and maintaining the stability of their values. By contrast, the four nonlinear adjustments offer increased flexibility and responsiveness, enabling the algorithm to dynamically adapt to diverse optimization landscapes and escape from local optima. The strategic integration of these linear and nonlinear schemes is designed to boost the robustness of the SA-EPO across various problem domains. Although linear adjustments may be adequate for simpler scenarios, nonlinear adjustments are crucial in more complex environments, thereby enhancing the versatility and efficacy of the SA-EPO algorithm across a wide range of applications.

3.3.1. Linear strategy

The linear strategy employed by the proposed SA-EPO can linearly adjust the control parameters of each *i*-th emperor penguin within their specified ranges based on the normalised fitness difference $Norm_i$ value by scaling it to fit the parameter range of $(x_{\max} - x_{\min})$ using the linear transformation formula shown in Eq. (14):

$$x_i = x_{\min} + (x_{\max} - x_{\min}) * (Norm_i) \tag{14}$$

Fig. 2 graphically illustrates the linear tuning behavior of the control parameters, showing how each parameter varies with the normalised fitness difference under a linear strategy. The normalised fitness difference is assumed to range from 0 to 100. This visualisation clearly demonstrates that the control parameters increase linearly in response to increases in the normalised fitness difference for each *i*-th emperor penguin.

3.3.2. Quadratic strategy

The quadratic strategy implemented in the proposed SA-EPO algorithm incorporates a quadratic component to adjust the control parameters of each *i*-th emperor penguin. This approach scales the control parameters within their designated ranges based on the square of $Norm_i$,

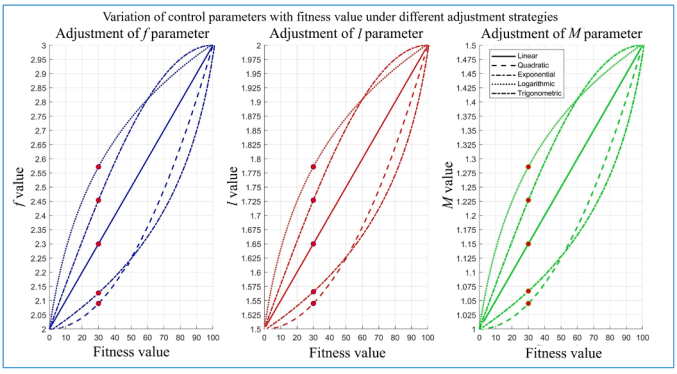

Fig. 7. Variation of control parameters with fitness range [0 100] across different parameter adaptation strategies.

Table 3
Adjusted values of control parameters for the *i*-th emperor penguin with fitness value of 30 using different parameter adaptation strategies.

	Parameter f	Parameter l	Parameter M
Value Range	[2 3]	[1.5 2]	[1 1.5]
Linear Strategy	2.300	1.650	1.150
Quadratic Strategy	2.090	1.545	1.045
Exponential Strategy	2.130	1.565	1.065
Logarithmic Strategy	2.571	1.786	1.286
Trigonometric Strategy	2.454	1.727	1.227

thereby responding to variations in normalised fitness differences with quadratic behavior. The specific formula used for this adjustment is presented in Eq. (15):

$$x_i = x_{\min} + (x_{\max} - x_{\min}) * (Norm_i)^2 \tag{15}$$

Fig. 3 graphically illustrates this quadratic tuning behavior, showing how parameter adjustments adhere to a quadratic function. The inclusion of the square term adds a nonlinear element to the scaling process, intensifying adjustments as the normalised fitness difference increases. This quadratic strategy causes values farther from the minimum threshold to exert a disproportionately larger influence on parameter adjustments compared with those made under a linear strategy.

3.3.3. Exponential strategy

The exponential strategy employs an exponential function to dynamically adjust the control parameters of each *i*-th emperor penguin within their specified ranges, based on its fitness value. This strategy employs the $Norm_i$ value of each *i*-th emperor penguin, negating it before applying an exponential function, as shown in Eq. (16):

$$x_i = x_{\min} + (x_{\max} - x_{\min}) * \exp(-Norm_i) \tag{16}$$

Fig. 4 illustrates the exponential tuning behavior, highlighting how the control parameter adjustments conform to an exponential model. This method facilitates non-linear scaling and transformation, exponentially weighing the influence of deviations from the minimum fitness value.

3.3.4. Logarithmic strategy

The logarithmic strategy employs a log scaling function to adjust the control parameters f, l and M each *i*-th emperor penguin within their specified ranges, aligning with the respective fitness values. To avoid computing the logarithm of zero, a constant of 1 is added to each $Norm_i$ value before applying the logarithmic functions for control parameter adaptation, as defined in Eq. (17):

$$x_i = x_{\min} + (x_{\max} - x_{\min}) * \log(Norm_i + 1) \tag{17}$$

Fig. 5 demonstrates this logarithmic tuning behavior, where the control parameters adjustment follows a logarithmic model. This method compresses the range of parameter adjustments, enhancing the sensitivity to variations, especially those far from the minimum fitness threshold.

3.3.5. Trigonometric strategy

The trigonometric strategy introduces sinusoidal behavior to the nonlinear adjustments of control parameters f, l and m, tailoring them to correspond with the fitness values of each *i*-th emperor penguin. This approach calculates parameter adjustments using a sinusoidal function based on the $Norm_i$ value, as detailed in Eq. (18):

$$x_i = x_{\min} + (x_{\min} - x_{\min}) * \sin\left(\frac{\pi}{2} * Norm_i\right) \tag{18}$$

Fig. 6 illustrates trigonometric tuning behaviors, depicting how adjustments of control parameters follow the ascending part of a sinusoidal curve. This sine function induces oscillations that vary with the normalised fitness value, dynamically adjusting the control parameters within their ranges from the minimum threshold.

3.3.6. Overall comparison of all parameter adaptation strategies

The five parameter adaptation strategies previously described are tailored to adjust the control parameters f, l, and M for each *i*-th emperor penguin uniquely. The degree of adjustment for these control parameters is determined by the normalised fitness differences, which are calculated based on each emperor penguin's fitness value. This process is pivotal in deciding whether an emperor penguin should focus on exploration or exploitation.

We consider a hypothetical scenario where the fitness values of all search agents range from 0 to 100, with the fitness value of *i*-th emperor penguin is set as 30. This setting results in noticeable variations in the adjusted values of the control parameters across different strategies. Fig. 7 illustrates how these control parameters vary with the fitness values under each strategy, with the unique set of adjusted parameter values indicated by red dots. Table 3 provides a detailed summary of the specific values of the tuned control parameters when the fitness value of the *i*-th emperor penguin is 30, clearly demonstrating the variations across different parameter adaptation strategies.

3.4. Intelligent selection mechanism

The intelligent selection mechanism is another critical component of the multiple-parameter adaptation schemes incorporated into the proposed SA-EPO. This mechanism allows each emperor penguin in SA-EPO to self-adaptively select the most suitable parameter adaptation strategy from the candidate pool. The selection is based on the penguin's current fitness value, historical performance, and the efficacy record of each strategy. Essentially, the intelligent selection mechanism identifies the most appropriate parameter adaptation strategy to optimize the algorithm's performance at various stages of the optimization process.

The SA-EPO algorithm incorporates an intelligent selection mechanism characterised by a predefined learning period, applicable to all five parameter adaptation strategies [82]. During each learning period, the selection probabilities for each strategy remain constant. If a new solution generated by a particular parameter adaptation strategy for an emperor penguin outperforms the current solution, the selection probability for that strategy is increased. Conversely, if the new solution underperforms, the selection probability is decreased.

Initially, each parameter adaptation strategy is assigned an equal initial selection probability of $1/N_q$, where N_q is the total number of strategies in the candidate pool. For SA-EPO, N_q is set to 5. Let SSP_q denotes the selection probability of the q-th parameter adaptation strategy, where $q = 1, 2, 3, \ldots, N_q$. The roulette wheel method [24] is then employed to select a parameter adaptation strategy based on these probabilities for each *i*-th emperor penguin, where $i = 1, 2, 3, \ldots, N$. This

selected strategy is subsequently used to adjust the control parameters of the i-th emperor penguin before generating a new solution, which is then evaluated and compared with the current solution.

In each iteration of SA-EPO, the outcome of a comparison between the newly generated solution and the current solution for each i-th emperor penguin is recorded in two matrices, denoted as $Sflag_{N\times N_q}$ and $Fflag_{N\times N_q}$, where the initial elements are set to zero:

$$\begin{cases} Sflag_{N\times N_q} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N\times N_q} \\ Fflag_{N\times N_q} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N\times N_q} \end{cases} \tag{19}$$

If a superior new solution is generated by the i-th emperor penguin using the q-th parameter adaptation strategy, the i-th row and q-th column of $Sflag_{N\times N_q}$ (i.e., element $Sflag_{i,q}$) is set to 1. Conversely, if the new solution fails to outperform the current one, element $Fflag_{i,q}$ is set to 1, indicating that the q-th strategy did not lead to an improvement for the i-th emperor penguin. These flags serve to track the efficacy of each parameter adaptation strategy across the population of emperor penguins in every iteration of SA-EPO.

After each iteration of the SA-EPO, the sum of each q-th column from the matrices $Sflag_{N\times N_q}$ and $Fflag_{N\times N_q}$ are calculated. These sums represent the total number of successful and failed attempts by each q-th parameter selection strategy to improve the solutions of emperor penguin within that iteration. These results are recorded in the k-th row and q-th column of two new matrices, denoted as $Stotal_{N_{LP}\times N_q}$ and $Ftotal_{N_{LP}\times N_q}$, where $k = 1, 2, 3, \ldots, LP$. N_{LP} is the total number of iterations within a predefined learning period. The initial values of the elements in these matrices are set as follows:

$$\begin{cases} Stotal_{N_{LP}\times N_q} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N_{LP}\times N_q} \\ Ftotal_{N_{LP}\times N_q} = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}_{N_{LP}\times N_q} \end{cases} \tag{20}$$

Each element $Stotal_{k,q}$ represents the total number of successful improvements by the q-th parameter adaptation strategy during the k-th iteration of the learning period, while each $Ftotal_{k,q}$ counts the total number of failures. After updating the values in $Stotal_{N_{LP}\times N_q}$ and $Ftotal_{N_{LP}\times N_q}$ for every k-th iteration, the values in $Sflag_{N\times N_q}$ and $Fflag_{N\times N_q}$ are reset to zero as in Eq. (19) to prepare them for the next iteration of data recording.

Throughout the N_{LP} iterations of learning periods, the values in $Stotal_{N_{LP}\times N_q}$ and $Ftotal_{N_{LP}\times N_q}$ matrices are continuously updated. At the end of each learning period, the sums of the q-th columns from these matrices are calculated to determine the total number of successes and failures of each q-th parameter adaptation strategy across all emperor penguins during the last N_{LP} iterations. The updating process is defined by Eq. (21):

$$Stotal_q^{new} \begin{cases} \sum_{k=1}^{N_{LP}} Stotal_{k,q} \Big/ \left(\sum_{k=1}^{N_{LP}} Stotal_{k,q} + \sum_{k=1}^{N_{LP}} Ftotal_{k,q} \right), & if \sum_{k=1}^{N_{LP}} Stotal_{k,q} \neq 0 \\ \varepsilon \Big/ \left(\varepsilon + \sum_{k=1}^{N_p} Ftotal_{k,q} \right), & \text{Otherwise} \end{cases} \tag{21}$$

where ε is a positive constant added to prevent the selection probability of any parameter adaptation strategy from dropping to zero; $Stotal_q^{new}$ represents the new selection probability of the q-th parameter adaptation strategy, calculated as the ratio of successful improvements to total attempts during the last learning period for that strategy.

Next, the newly computed $Stotal_q^{new}$ is normalised to determine the new selection probability SSP_q of the q-th parameter adaptation strategy for the upcoming learning period, as shown in Eq. (22):

$$SSP_q = Stotal_q^{new} \Big/ \sum_{q=1}^{N_q} Stotal_q^{new} \tag{22}$$

Eqs. (21) and (22) facilitate the dynamic update of selection probabilities for each parameter adaptation strategy, enhancing or diminishing their likelihood of selection based on their performance in generating superior solutions during the previous learning period. This intelligent selection mechanism optimizes the self-adaptation strategy's efficacy, ensuring strategies that previously performed well receive a higher probability in subsequent phases.

3.5. Overall workflow of proposed SA-EPO

The overall workflow of the proposed SA-EPO is detailed in Algorithm 2, which integrates multiple-parameter adaptation schemes (Lines 18–27) and an intelligent selection scheme (Lines 40–45 and 47–58) to enhance the performance of the original EPO. These modifications aim to increase the flexibility and adaptability of the SA-EPO, enabling each emperor penguin to effectively select the most suitable parameter adaptation strategy for various problem scenarios. A key variable, defined as LPC, tracks the number of iterations since the last learning period and it plays a crucial role in determining the end of a learning period. Notably, the selection probabilities of each parameter adaptation strategy are dynamically updated after each learning period based on their efficacy in improving the fitness values of emperor penguins during the previous learning period. The intelligent selection mechanism rewards the parameter adaptation strategies that perform well in a previous learning period with higher selection probabilities in subsequent periods. Conversely, the parameter adaptation strategies that do not enhance the fitness of emperor penguins are penalised with lower selection probabilities for the following learning period.

Algorithm 2. : Pseudocode for the proposed SA-EPO algorithm

Inputs: The emperor penguin population $P_i(i \leftarrow 1,2,\ldots,N)$, upper and lower limits of all control parameters (i.e., $f_{min}, f_{max}, l_{min}, l_{max}, M_{min}, and\ M_{max}$), numbers of parameter adaptation strategies N_q, total iteration numbers within a learning period N_{LP},

Initialize the parameters $T, T', A, C, S_F(), R, and\ I_{max}$;
Initialize SSP_q for each ***q***-th parameter selection strategy as $1/N_q$, where ***q***=1,…, N_q;
Initialize $LPC = 0, t = 0, k = 0$;
Initialize all elements of matrices $Sflag_{N\times N_q}$ and $Fflag_{N\times N_q}$ as zeros;
Initialize all elements of matrices $Stotal_{N_{LP}\times N_q}$ and $Ftotal_{N_{LP}\times N_q}$ as zeros;
for $i \leftarrow 1\ to\ N$ **do** ***/*Population initialization of SA-EPO*/***
  Randomly generate the initial P_i of each ***i***-th emperor penguin;
  Calculate the initial fitness of ***i***-th emperor penguin as fit_i = **FITNESS** (P_i);
end for
while $t < I_{max}$ **do** ***/*Iterative search process of SA-EPO*/***
  Determine the best emperor penguin as P_{best} with best fitness value of min_{Fit};
  Determine the worst fitness value among all emperor penguins as max_{Fit};
  Calculate R_{Fit} using Eq. (12);
  Randomly generate the radius of polygon $R \leftarrow Rand()$;
  Calculate the huddle temperature ***T*** using Eq. (3);
  Compute the temperature profile around the huddle ***T'*** using Eq. (4);
  for $i \leftarrow 1\ to\ N$ **do** ***/*for each i-th emperor penguin*/***
    Calculate $diff_i$ and $Norm_i$ using Equations (11) and (13), respectively;
    Select the ***q***-th parameter adaptation strategy with a selection probability of SSP_q using the Roulette Wheel method;
    switch parameter adaptation strategy index ***q***
      Case 1: Adjust ***f***, ***l***, and ***M*** using linear strategy in Eq. (14);
      Case 2: Adjust ***f***, ***l***, and ***M*** using quadratic strategy in Eq. (15);
      Case 3: Adjust ***f***, ***l***, and ***M*** using exponential strategy in Eq. (16);
      Case 4: Adjust ***f***, ***l***, and ***M*** using logarithmic strategy in Eq. (17);
      Case 5: Adjust ***f***, ***l***, and ***M*** using trigonometric strategy in Eq. (18);
    end switch
    Initialize the updated position of the ***i***-th emperor penguin as $P_{i,t+1} \leftarrow \emptyset$;
    for $j \leftarrow d\ to\ D$ **do** ***/*for each j-th dimension of i-th emperor penguin*/***
      Calculate P_{grid} $(Accuracy)$ using Eq. (7);
      Calculate A and ***C*** using Eqs. (6) and (8), respectively;
      Calculate $S_F(A)$ using Eq. (9);
      Calculate Dis_i using Eq. (5);
      Update the j-th dimension of position for the i-th emperor penguin as $P_{i,j,t+1}$ using Eq. (10);
      Ensure the updated $P_{i,j,t+1}$ is within the search boundary;
      $P_{i,t+1} \leftarrow P_{i,t+1} \cup P_{i,j,t+1}$;
    end for
    Calculate the updated fitness as $fit_{i,t+1}$ = **FITNESS** $(P_{i,t+1})$;
    if updated $P_{i,t+1}$ has better fitness value **then**
      Update $P_{i,t}$ and $P_{best,t}$ along with their fitness values;
      Update the corresponding success flag element as $Sflag_{i,q} = 1$
    Else
      Update the corresponding failure flag element as $Fflag_{i,q} = 1$;
    end if
  end for
  Update total success element $Stotal_{k,q}$ and total failure element $Ftotal_{k,q}$ by summing up each q-th column of $Sflag_{N\times N_q}$ and $Fflag_{N\times N_q}$, respectively;
  Reset $Sflag_{N\times N_q}$ and $Fflag_{N\times N_q}$ into zero matrices as in Eq. (19);
  Update iteration counter within a learning period as $k \leftarrow k + 1$;
  Update iteration counter as $t \leftarrow t + 1$;
  if $t - LPC == N_{LP}$ **then** */*After the current learning period has elapsed*/*
    Calculate the $Stotal_q^{new}$ of each q-th parameter selection strategy based on the updated $Stotal_{N_{LP}\times N_q}$ and $Ftotal_{N_{LP}\times N_q}$ using Eq. (21);
    Update the SSP_q of each q-th parameter selection strategy using Eq. (22);
    Reset $Stotal_{N_{LP}\times N_q}$ and $Ftotal_{N_{LP}\times N_q}$ into zero matrices as in Eq. (20);
    Update $LPC \leftarrow t$ and reset $k \leftarrow 0$;
  end if
end while

Outputs: Best emperor penguin P_{best} and its fitness value.

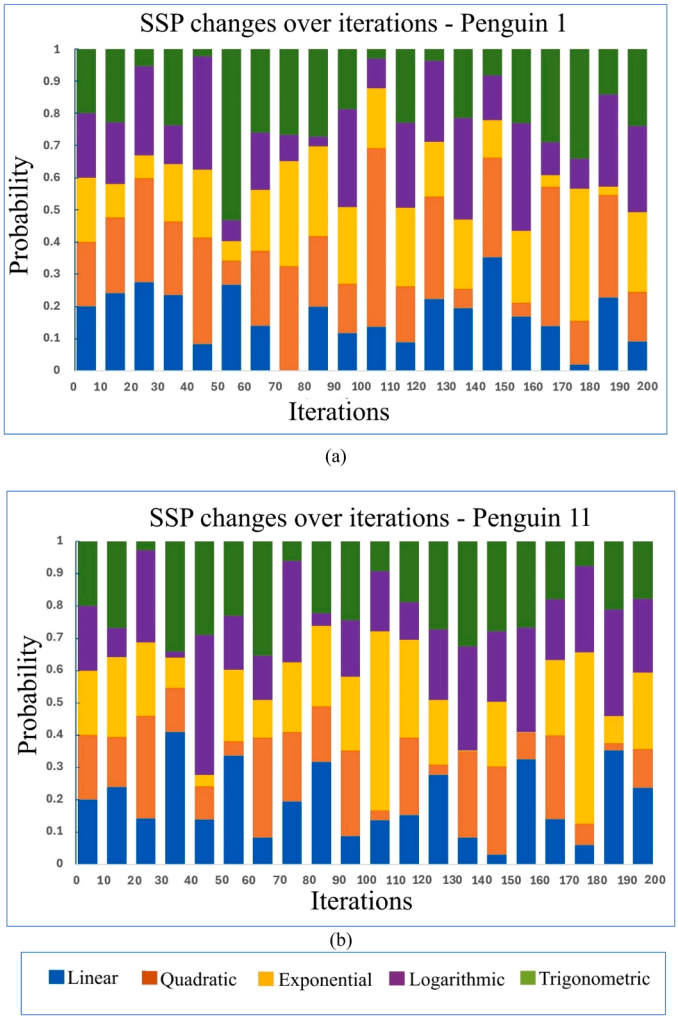

Fig. 8. Continuous change in the strategy selection probability over iterations for emperor penguin with solution indices of: (a) $i = 1$ and (b) $i = 11$.

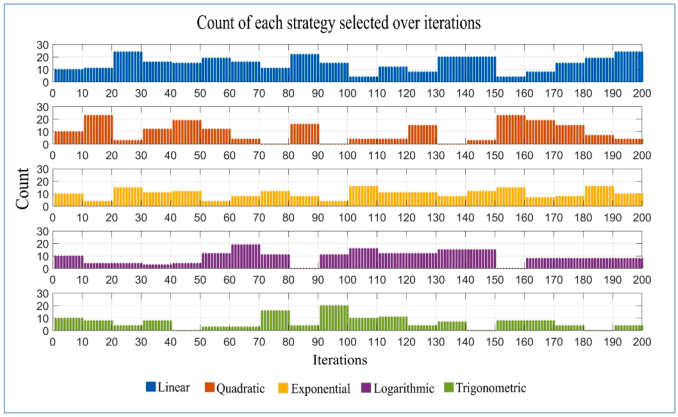

Fig. 9. Frequency of each parameter adaptation strategy selected over iterations.

In the SA-EPO, analysing the parameter adaptation strategy selected by emperor penguins is essential for adjusting their control parameters during optimization. Fig. 8 provides a visual representation of changes in selection probabilities for all parameter strategies across consecutive iterations, considering a population size of $N = 50$ and a maximum iteration number of $I_{\max} = 200$. Additionally, a learning period of $N_{LP} = 10$ is defined, indicating that adjustments in selection probabilities occur every 10 iterations. Importantly, the total sum of probabilities for the five strategies consistently equals one, maintaining probabilistic consistency throughout the optimization process. During each learning period, the strategy with the highest selection probability is more likely to be chosen, ensuring the most suitable strategy is employed at the current optimization stage. This approach guarantees that the chosen strategy meets the optimization demands effectively, thereby enhancing outcomes.

Table 4
Parameter settings of the competing algorithms.

Algorithm	Reference	Year	Parameter Settings
EPO	[36]	2018	$ul \in [0,1]$, $M = 2, f = 3, l = 2$
SaBO	[63]	2023	$ul \in [0,1]$, $p_f = 0.5$, $f_t \in [0.005, 0.14]$, $p_t \in [1.2, 1.8]$
SdSCA	[64]	2023	$ul \in [0,1]$, $a = 2$, $F = 0.8$, $CR = 0.95$
SDPSO	[65]	2023	$ul \in [0,1]$, $\alpha = 0.1$, $\omega = 0.9 \sim 0.2$, $c = 3$, $c_1 = 2.5 \sim 0.5$, $c_2 = 0.5 \sim 2.5$, $r = 4$, $LP = 200$
MSEFA	[62]	2022	$ul \in [0,1]$, $k \in \{0.1, 0.2, 0.3\}$, $\beta_0 = 1$, $\beta_{\min} = 0.2$, $\alpha = 0.5$, $\Upsilon = 1$
ESO	[72]	2023	$ul \in [0,1]$, $Threshold = 0.25$, $Threshold_2 = 0.6$, $C_1 = 0.5$, $C_2 = 0.05$, $C_3 = 2$
MS-DE	[73]	2022	$ul \in [0,1]$, $CR_1 = 0.1$, $CR_2 = 0.9$, $sn_1 = 2$, $sn_2 = 3$, $T = 10$
FAEPO	[43]	2022	$ul \in [0,1]$, $M = 2$
IEPO	[53]	2020	$ul \in [0,1]$, $M = 2, f = 3, l = 2$
ESA	[37]	2019	$ul \in [0,1]$, $M = 2, f \in [2,3]$, $l \in [1.5,2]$, $c_1 \in [0,1]$, $c_2 \in [0,1]$, $c_3 \in [0,1]$
CEPO	[55]	2019	$ul \in [0,1]$, $M = 2, f = 3, l = 2$, $\mu \in \{1,2,3,4\}$
H-EPC	[56]	2021	$ul \in [0,1]$, $M = 2, f = 3, l = 2$, $\varphi = 0.2$, $\zeta\text{rad} = 0.9995$, $\alpha = 1$, $\zeta\alpha = 0.9998$, $\zeta\mu = 0.03$, $a = 0.2$, $b = 0.5$
SHADE	[78]	2013	$ul \in [0,1]$, $c_r = 0.5$, $F_{initial} = 0.5$, $F_{decay} = 0.97$, $Mem_{size} = 5$, $P_{b\_size} = 5$
JDE	[79]	2009	$ul \in [0,1]$, $c_{r\_\min} = 0.45$, $c_{r\_\max} = 0.85$, $F = 0.8$, $k = 2$
JADE	[80]	2009	$ul \in [0,1]$, $c_{r\_\mu} = 0.5$, $F_\mu = 0.6$, $c_{weight} = 0.1$
Proposed SA-EPO	-	-	$ul \in [0,1]$, $M \in [1, 1.5]$, $f \in [2,3]$, $l \in [1.5,2]$

In addition to analysing the variation in selection probabilities of parameter adaptation strategies for each emperor penguin across iterations, assessing how frequently specific strategies are chosen to evaluate their effectiveness is vital. This evaluation involves determining which parameter adaptation strategies are most used across multiple iterations, offering essential insights into their performance. Fig. 9 depicts the frequency of each parameter adaptation strategy's selection across various iterations, illustrating these patterns clearly. Notably, the total number of strategy selections per iteration consistently aligns with the population size of 50, ensuring that each emperor penguin is uniquely paired with a strategy for adjusting its control parameters.

4. Experimental results

This section thoroughly details the experiments conducted on 41 challenging benchmark test functions and seven real-world engineering problems to evaluate the efficiency and effectiveness of the proposed SA-EPO. Additionally, the results are benchmarked against well-known metaheuristic algorithms. Comprehensive descriptions of the results, test functions, metrics, and statistical analyses are provided in the following subsections.

Table 5
Mean and *SD* for 100-*D* benchmark test functions CEC2017.

Function	F1		F2		F3	
	Mean	SD			Mean	SD
EPO	1.35E+ 11	4.12E+ 08	F2 has been excluded because it shows unstable behavior especially for higher dimensions, and significant performance variations for the same algorithm [83].		2.78E+ 14	1.38E+ 13
SaBO	1.35E+ 11	3.87E+ 08			2.78E+ 14	1.41E+ 13
SdSCA	1.35E+ 11	4.25E+ 08			2.69E+ 14	1.33E+ 13
SDPSO	1.35E+ 11	4.86E+ 08			2.67E+ 14	1.70E+ 13
MSEFA	1.35E+ 11	4.47E+ 08			2.69E+ 14	1.82E+ 13
ESO	1.35E+ 11	3.83E+ 08			2.66E+ 14	1.69E+ 13
MS-DE	1.35E+ 11	3.02E+ 08			2.89E+ 14	1.19E+ 13
FAEPO	1.35E+ 11	4.48E+ 08			2.66E+ 14	1.63E+ 13
IEPO	1.35E+ 11	4.44E+ 08			2.66E+ 14	1.60E+ 13
ESA	1.35E+ 11	4.50E+ 08			2.66E+ 14	1.63E+ 13
CEPO	1.35E+ 11	4.39E+ 08			2.66E+ 14	1.68E+ 13
H-EPC	1.35E+ 11	4.28E+ 08			2.66E+ 14	1.79E+ 13
SHADE	*1.33E+ 11*	1.87E+ 09			2.64E+ 14	4.99E+ 13
JDE	1.35E+ 11	3.36E+ 08			*2.62E+ 14*	1.02E+ 13
JADE	1.35E+ 11	2.68E+ 08			2.63E+ 14	1.03E+ 13
SA-EPO	**1.32E+ 11**	7.49E+ 09			**1.53E+ 14**	7.15E+ 13
	F4		F5		F6	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	6.79E+ 04	3.00E+ 02	9.39E+ 02	7.54E+ 00	1.12E+ 02	3.10E+ 00
SaBO	6.77E+ 04	2.62E+ 02	9.37E+ 02	6.73E+ 00	1.12E+ 02	2.94E+ 00
SdSCA	6.77E+ 04	2.48E+ 02	9.38E+ 02	5.08E+ 00	1.12E+ 02	2.70E+ 00
SDPSO	6.78E+ 04	3.22E+ 02	9.39E+ 02	8.12E+ 00	1.12E+ 02	3.28E+ 00
MSEFA	6.77E+ 04	3.17E+ 02	9.37E+ 02	8.86E+ 00	1.09E+ 02	3.26E+ 00
ESO	6.77E+ 04	3.05E+ 02	9.39E+ 02	9.21E+ 00	1.09E+ 02	3.46E+ 00
MS-DE	6.78E+ 04	2.45E+ 02	9.39E+ 02	5.07E+ 00	1.11E+ 02	2.31E+ 00
FAEPO	6.77E+ 04	3.52E+ 02	9.37E+ 02	8.65E+ 00	1.12E+ 02	3.28E+ 00
IEPO	6.77E+ 04	2.92E+ 02	9.39E+ 02	8.11E+ 00	1.12E+ 02	3.42E+ 00
ESA	6.77E+ 04	3.15E+ 02	9.37E+ 02	8.91E+ 00	1.09E+ 02	3.45E+ 00
CEPO	6.77E+ 04	3.08E+ 02	9.38E+ 02	7.86E+ 00	1.10E+ 02	2.98E+ 00
H-EPC	6.78E+ 04	3.39E+ 02	9.37E+ 02	7.69E+ 00	1.09E+ 02	3.48E+ 00
SHADE	*6.73E+ 04*	1.09E+ 03	9.32E+ 02	2.59E+ 01	1.11E+ 02	1.06E+ 01
JDE	6.75E+ 04	1.75E+ 02	*9.17E+ 02*	4.88E+ 00	1.09E+ 02	2.50E+ 00
JADE	6.75E+ 04	2.03E+ 02	9.18E+ 02	4.89E+ 00	*1.07E+ 02*	2.44E+ 00
SA-EPO	**6.42E+ 04**	4.31E+ 03	**8.61E+ 02**	8.62E+ 01	**9.71E+ 01**	1.15E+ 01
	F7		F8		F9	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.78E+ 03	2.90E+ 01	1.09E+ 03	8.46E+ 00	5.18E+ 04	5.67E+ 03
SaBO	1.76E+ 03	3.61E+ 01	1.07E+ 03	8.52E+ 00	4.94E+ 04	4.72E+ 03
SdSCA	1.77E+ 03	3.19E+ 01	1.08E+ 03	6.58E+ 00	4.96E+ 04	3.49E+ 03
SDPSO	1.79E+ 03	3.88E+ 01	1.09E+ 03	7.70E+ 00	5.28E+ 04	4.72E+ 03
MSEFA	1.76E+ 03	3.03E+ 01	1.07E+ 03	8.56E+ 00	4.58E+ 04	4.81E+ 03
ESO	1.76E+ 03	3.89E+ 01	1.08E+ 03	8.85E+ 00	4.57E+ 04	5.08E+ 03
MS-DE	1.74E+ 03	2.89E+ 01	1.08E+ 03	6.92E+ 00	4.63E+ 04	3.24E+ 03
FAEPO	1.76E+ 03	3.47E+ 01	1.08E+ 03	9.46E+ 00	4.60E+ 04	4.60E+ 03
IEPO	1.76E+ 03	3.35E+ 01	1.09E+ 03	7.45E+ 00	4.58E+ 04	4.86E+ 03
ESA	1.75E+ 03	3.46E+ 01	1.08E+ 03	8.41E+ 00	4.60E+ 04	4.63E+ 03
CEPO	1.76E+ 03	3.73E+ 01	1.07E+ 03	8.22E+ 00	4.64E+ 04	4.62E+ 03
H-EPC	1.76E+ 03	3.06E+ 01	1.08E+ 03	8.09E+ 00	4.58E+ 04	4.58E+ 03
SHADE	1.76E+ 03	1.04E+ 02	1.07E+ 03	3.23E+ 01	5.41E+ 04	8.89E+ 03
JDE	*1.70E+ 03*	2.44E+ 01	*1.06E+ 03*	6.70E+ 00	**4.41E+ 04**	3.21E+ 03
JADE	1.71E+ 03	3.09E+ 01	*1.06E+ 03*	6.38E+ 00	**4.41E+ 04**	2.78E+ 03
SA-EPO	**1.65E+ 03**	1.12E+ 02	**9.80E+ 02**	1.27E+ 02	*4.45E+ 04*	6.56E+ 03
	F10		F11		F12	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.93E+ 04	2.01E+ 02	6.88E+ 05	3.74E+ 04	1.42E+ 11	3.66E+ 08
SaBO	1.91E+ 04	2.39E+ 02	6.80E+ 05	3.85E+ 04	1.41E+ 11	3.72E+ 08
SdSCA	1.91E+ 04	2.13E+ 02	6.79E+ 05	4.44E+ 04	1.42E+ 11	2.51E+ 08
SDPSO	1.95E+ 04	1.93E+ 02	6.89E+ 05	7.80E+ 04	1.42E+ 11	4.01E+ 08
MSEFA	1.92E+ 04	2.72E+ 02	6.83E+ 05	4.76E+ 04	1.41E+ 11	4.08E+ 08
ESO	1.92E+ 04	2.30E+ 02	6.79E+ 05	5.66E+ 04	1.41E+ 11	3.78E+ 08
MS-DE	1.92E+ 04	1.51E+ 02	6.81E+ 05	4.15E+ 04	1.42E+ 11	2.88E+ 08
FAEPO	1.91E+ 04	2.15E+ 02	6.78E+ 05	5.42E+ 04	1.41E+ 11	3.55E+ 08
IEPO	1.92E+ 04	2.19E+ 02	6.80E+ 05	4.66E+ 04	1.42E+ 11	3.59E+ 08
ESA	1.91E+ 04	2.07E+ 02	6.81E+ 05	6.79E+ 04	1.42E+ 11	3.46E+ 08
CEPO	1.91E+ 04	2.01E+ 02	6.79E+ 05	6.03E+ 04	1.41E+ 11	3.55E+ 08
H-EPC	1.93E+ 04	2.28E+ 02	6.80E+ 05	5.27E+ 04	1.42E+ 11	3.79E+ 08
SHADE	1.92E+ 04	8.35E+ 02	*3.47E+ 05*	1.91E+ 05	*1.40E+ 11*	2.06E+ 09
JDE	1.91E+ 04	1.83E+ 02	6.38E+ 05	2.96E+ 04	1.41E+ 11	2.78E+ 08

(*continued on next page*)

Table 5 (*continued*)

JADE	*1.89E+ 04*	1.80E+ 02	6.48E+ 05	2.89E+ 04	1.41E+ 11	2.95E+ 08
SA-EPO	**1.72E+ 04**	2.80E+ 03	**7.29E+ 04**	1.35E+ 05	**1.38E+ 11**	9.01E+ 09
	F13		F14		F15	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.13E+ 11	1.99E+ 08	1.45E+ 09	4.00E+ 06	2.36E+ 10	3.85E+ 07
SaBO	1.13E+ 11	1.31E+ 08	1.44E+ 09	4.37E+ 06	2.35E+ 10	2.87E+ 07
SdSCA	1.13E+ 11	1.21E+ 08	1.44E+ 09	3.87E+ 06	2.35E+ 10	4.16E+ 07
SDPSO	1.13E+ 11	1.71E+ 08	1.45E+ 09	4.87E+ 06	2.35E+ 10	4.48E+ 07
MSEFA	1.13E+ 11	1.60E+ 08	1.44E+ 09	3.51E+ 06	2.36E+ 10	3.51E+ 07
ESO	*1.12E+ 11*	1.64E+ 08	1.44E+ 09	4.65E+ 06	2.35E+ 10	3.98E+ 07
MS-DE	1.13E+ 11	2.27E+ 08	1.44E+ 09	4.38E+ 06	2.35E+ 10	5.05E+ 07
FAEPO	1.13E+ 11	2.04E+ 08	1.44E+ 09	4.26E+ 06	2.35E+ 10	3.06E+ 07
IEPO	*1.12E+ 11*	1.41E+ 08	1.44E+ 09	3.65E+ 06	2.36E+ 10	3.26E+ 07
ESA	1.13E+ 11	1.49E+ 08	1.44E+ 09	4.71E+ 06	2.35E+ 10	5.41E+ 07
CEPO	*1.12E+ 11*	1.77E+ 08	1.44E+ 09	4.91E+ 06	2.35E+ 10	3.44E+ 07
H-EPC	1.13E+ 11	1.49E+ 08	1.44E+ 09	3.65E+ 06	2.36E+ 10	4.46E+ 07
SHADE	1.13E+ 11	1.27E+ 09	1.44E+ 09	5.24E+ 07	*2.34E+ 10*	3.67E+ 08
JDE	*1.12E+ 11*	1.22E+ 08	*1.43E+ 09*	2.01E+ 06	2.35E+ 10	2.63E+ 07
JADE	*1.12E+ 11*	1.51E+ 08	*1.43E+ 09*	4.20E+ 06	2.35E+ 10	2.40E+ 07
SA-EPO	**1.07E+ 10**	4.18E+ 09	**1.27E+ 09**	1.10E+ 08	**2.17E+ 10**	1.12E+ 09
	F16		F17		F18	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	2.27E+ 04	1.17E+ 02	1.58E+ 05	1.70E+ 03	1.94E+ 09	2.70E+ 07
SaBO	2.26E+ 04	1.19E+ 02	1.56E+ 05	2.20E+ 03	1.93E+ 09	2.05E+ 07
SdSCA	2.25E+ 04	9.88E+ 01	1.57E+ 05	2.44E+ 03	1.92E+ 09	2.07E+ 07
SDPSO	2.27E+ 04	1.12E+ 02	1.59E+ 05	1.82E+ 03	1.95E+ 09	1.86E+ 07
MSEFA	2.26E+ 04	1.13E+ 02	1.55E+ 05	1.62E+ 03	1.93E+ 09	2.01E+ 07
ESO	2.25E+ 04	1.14E+ 02	1.56E+ 05	1.94E+ 03	1.92E+ 09	2.18E+ 07
MS-DE	2.26E+ 04	6.86E+ 01	1.59E+ 05	2.30E+ 03	1.94E+ 09	1.85E+ 07
FAEPO	2.25E+ 04	1.28E+ 02	1.57E+ 05	2.13E+ 03	1.93E+ 09	1.86E+ 07
IEPO	2.26E+ 04	1.20E+ 02	1.56E+ 05	1.58E+ 03	1.94E+ 09	1.77E+ 07
ESA	2.26E+ 04	1.29E+ 02	1.58E+ 05	2.38E+ 03	1.93E+ 09	1.81E+ 07
CEPO	2.25E+ 04	1.26E+ 02	1.58E+ 05	1.84E+ 03	1.93E+ 09	1.90E+ 07
H-EPC	2.26E+ 04	1.21E+ 02	1.57E+ 05	1.95E+ 03	1.94E+ 09	1.52E+ 07
SHADE	2.25E+ 04	5.82E+ 02	*1.54E+ 05*	1.17E+ 04	*1.88E+ 09*	9.40E+ 07
JDE	2.25E+ 04	8.06E+ 01	1.56E+ 05	1.49E+ 03	1.90E+ 09	1.70E+ 07
JADE	*2.24E+ 04*	9.22E+ 01	1.55E+ 05	1.41E+ 03	1.90E+ 09	9.63E+ 06
SA-EPO	**2.07E+ 04**	2.90E+ 03	**1.23E+ 05**	3.18E+ 04	**1.65E+ 09**	3.25E+ 08
	F19		F20		F21	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.38E+ 10	2.38E+ 07	3.39E+ 03	2.45E+ 01	2.19E+ 03	1.11E+ 01
SaBO	1.35E+ 10	2.14E+ 07	3.38E+ 03	2.57E+ 01	2.18E+ 03	1.13E+ 01
SdSCA	1.36E+ 10	2.76E+ 07	3.39E+ 03	2.27E+ 01	2.18E+ 03	8.03E+ 00
SDPSO	1.37E+ 10	2.75E+ 07	3.39E+ 03	1.99E+ 01	2.20E+ 03	1.21E+ 01
MSEFA	1.35E+ 10	2.54E+ 07	3.36E+ 03	2.41E+ 01	2.19E+ 03	1.26E+ 01
ESO	1.36E+ 10	3.28E+ 07	3.37E+ 03	2.58E+ 01	2.18E+ 03	1.19E+ 01
MS-DE	1.38E+ 10	3.07E+ 07	3.35E+ 03	2.26E+ 01	2.19E+ 03	7.85E+ 00
FAEPO	1.36E+ 10	2.79E+ 07	3.36E+ 03	2.31E+ 01	2.19E+ 03	1.19E+ 01
IEPO	1.38E+ 10	2.95E+ 07	3.36E+ 03	1.71E+ 01	2.18E+ 03	1.13E+ 01
ESA	1.38E+ 10	2.12E+ 07	3.38E+ 03	2.19E+ 01	2.18E+ 03	1.14E+ 01
CEPO	1.35E+ 10	2.95E+ 07	3.35E+ 03	2.63E+ 01	2.18E+ 03	1.18E+ 01
H-EPC	1.36E+ 10	2.24E+ 07	3.34E+ 03	1.83E+ 01	2.18E+ 03	1.08E+ 01
SHADE	*1.34E+ 10*	2.54E+ 08	*3.21E+ 03*	1.51E+ 02	2.19E+ 03	6.03E+ 01
JDE	1.36E+ 10	1.61E+ 07	3.32E+ 03	1.53E+ 01	*2.17E+ 03*	9.53E+ 00
JADE	1.35E+ 10	1.59E+ 07	3.31E+ 03	1.34E+ 01	*2.17E+ 03*	8.62E+ 00
SA-EPO	**1.26E+ 10**	5.82E+ 08	**2.81E+ 03**	2.01E+ 02	**2.12E+ 03**	2.45E+ 02
	F22		F23		F24	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.89E+ 04	1.49E+ 02	7.38E+ 03	8.08E+ 01	4.45E+ 03	7.25E+ 00
SaBO	1.86E+ 04	1.65E+ 02	7.37E+ 03	6.92E+ 01	4.44E+ 03	6.76E+ 00
SdSCA	1.85E+ 04	1.48E+ 02	7.36E+ 03	6.62E+ 01	4.44E+ 03	6.26E+ 00
SDPSO	1.89E+ 04	1.57E+ 02	7.39E+ 03	8.92E+ 01	4.45E+ 03	6.37E+ 00
MSEFA	1.87E+ 04	1.66E+ 02	7.36E+ 03	6.34E+ 01	4.44E+ 03	5.43E+ 00
ESO	1.86E+ 04	1.60E+ 02	7.29E+ 03	7.27E+ 01	4.44E+ 03	6.57E+ 00
MS-DE	1.87E+ 04	1.32E+ 02	7.34E+ 03	6.84E+ 01	4.44E+ 03	5.50E+ 00
FAEPO	1.87E+ 04	1.47E+ 02	7.36E+ 03	6.40E+ 01	4.44E+ 03	6.87E+ 00
IEPO	1.88E+ 04	1.75E+ 02	7.33E+ 03	8.63E+ 01	4.44E+ 03	8.44E+ 00
ESA	1.87E+ 04	1.57E+ 02	7.38E+ 03	7.72E+ 01	4.44E+ 03	7.61E+ 00
CEPO	1.87E+ 04	1.50E+ 02	7.27E+ 03	8.77E+ 01	4.44E+ 03	6.68E+ 00
H-EPC	1.87E+ 04	1.47E+ 02	7.36E+ 03	7.00E+ 01	4.44E+ 03	6.93E+ 00
SHADE	1.88E+ 04	5.72E+ 02	7.33E+ 03	2.70E+ 02	4.45E+ 03	2.88E+ 01

(*continued on next page*)

Table 5 (*continued*)

JDE	*1.84E+ 04*	1.21E+ 02	*7.13E+ 03*	6.14E+ 01	*4.43E+ 03*	3.98E+ 00
JADE	1.85E+ 04	1.23E+ 02	7.23E+ 03	6.04E+ 01	4.44E+ 03	4.06E+ 00
SA-EPO	**1.76E+ 04**	1.10E+ 03	**6.46E+ 03**	9.17E+ 02	**4.39E+ 03**	1.07E+ 02
	F25		F26		F27	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.72E+ 04	8.97E+ 01	1.76E+ 04	6.05E+ 01	1.63E+ 04	1.39E+ 02
SaBO	1.71E+ 04	9.09E+ 01	1.75E+ 04	7.05E+ 01	1.63E+ 04	1.12E+ 02
SdSCA	1.73E+ 04	7.90E+ 01	1.75E+ 04	6.55E+ 01	1.63E+ 04	9.64E+ 01
SDPSO	1.71E+ 04	9.56E+ 01	1.76E+ 04	6.56E+ 01	1.63E+ 04	1.45E+ 02
MSEFA	1.72E+ 04	8.52E+ 01	1.75E+ 04	7.90E+ 01	1.61E+ 04	1.40E+ 02
ESO	1.71E+ 04	8.23E+ 01	1.76E+ 04	6.22E+ 01	1.61E+ 04	1.34E+ 02
MS-DE	*1.70E+ 04*	6.09E+ 01	1.75E+ 04	5.98E+ 01	1.62E+ 04	1.05E+ 02
FAEPO	1.71E+ 04	8.72E+ 01	1.75E+ 04	6.58E+ 01	1.61E+ 04	1.10E+ 02
IEPO	1.71E+ 04	9.00E+ 01	1.76E+ 04	8.24E+ 01	1.63E+ 04	1.22E+ 02
ESA	1.72E+ 04	8.06E+ 01	1.75E+ 04	6.91E+ 01	1.62E+ 04	1.35E+ 02
CEPO	1.71E+ 04	9.12E+ 01	1.76E+ 04	7.05E+ 01	1.62E+ 04	1.24E+ 02
H-EPC	1.72E+ 04	9.24E+ 01	1.75E+ 04	8.42E+ 01	1.63E+ 04	1.47E+ 02
SHADE	1.72E+ 04	3.21E+ 02	1.76E+ 04	2.53E+ 02	1.60E+ 04	5.27E+ 02
JDE	*1.70E+ 04*	7.02E+ 01	*1.74E+ 04*	4.68E+ 01	*1.59E+ 04*	1.03E+ 02
JADE	*1.70E+ 04*	6.24E+ 01	*1.74E+ 04*	5.27E+ 01	*1.59E+ 04*	9.46E+ 01
SA-EPO	**1.65E+ 04**	1.44E+ 03	**1.73E+ 04**	8.18E+ 02	**1.46E+ 04**	1.67E+ 03
	F28		F29		F30	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.73E+ 04	7.88E+ 01	6.39E+ 06	4.44E+ 04	2.46E+ 10	6.56E+ 07
SaBO	1.72E+ 04	7.22E+ 01	6.34E+ 06	4.18E+ 04	2.46E+ 10	6.26E+ 07
SdSCA	1.72E+ 04	5.95E+ 01	6.39E+ 06	2.93E+ 04	2.46E+ 10	5.61E+ 07
SDPSO	1.73E+ 04	7.31E+ 01	6.38E+ 06	3.52E+ 04	2.47E+ 10	8.90E+ 07
MSEFA	1.73E+ 04	7.45E+ 01	6.31E+ 06	4.19E+ 04	2.46E+ 10	8.15E+ 07
ESO	1.72E+ 04	7.77E+ 01	6.33E+ 06	4.53E+ 04	2.46E+ 10	9.71E+ 07
MS-DE	1.72E+ 04	5.54E+ 01	6.34E+ 06	3.78E+ 04	2.46E+ 10	4.21E+ 07
FAEPO	1.73E+ 04	8.01E+ 01	6.35E+ 06	4.04E+ 04	2.46E+ 10	1.00E+ 08
IEPO	1.72E+ 04	7.00E+ 01	6.34E+ 06	4.14E+ 04	2.46E+ 10	8.45E+ 07
ESA	1.72E+ 04	8.84E+ 01	6.38E+ 06	4.64E+ 04	2.46E+ 10	8.71E+ 07
CEPO	1.72E+ 04	7.13E+ 01	6.34E+ 06	4.66E+ 04	2.46E+ 10	9.40E+ 07
H-EPC	1.72E+ 04	6.94E+ 01	6.33E+ 06	4.50E+ 04	2.46E+ 10	9.16E+ 07
SHADE	1.73E+ 04	3.38E+ 02	*6.18E+ 06*	3.34E+ 05	2.46E+ 10	4.02E+ 08
JDE	*1.71E+ 04*	5.98E+ 01	6.28E+ 06	2.25E+ 04	*2.45E+ 10*	5.11E+ 07
JADE	1.72E+ 04	4.91E+ 01	6.27E+ 06	3.10E+ 04	*2.45E+ 10*	4.81E+ 07
SA-EPO	**1.66E+ 04**	6.94E+ 02	**5.26E+ 06**	8.16E+ 05	**2.36E+ 10**	1.57E+ 09
	w/t/l					*#BMF*
EPO	29/0/0					0
SaBO	29/0/0					0
SdSCA	29/0/0					0
SDPSO	29/0/0					0
MSEFA	29/0/0					0
ESO	29/0/0					0
MS-DE	29/0/0					0
FAEPO	29/0/0					0
IEPO	29/0/0					0
ESA	29/0/0					0
CEPO	29/0/0					0
H-EPC	29/0/0					0
SHADE	29/0/0					0
JDE	28/0/1					1
JADE	28/0/1					0
SA-EPO						**28**

4.1. Benchmark functions and simulation settings

The proposed algorithm was evaluated using 41 benchmark test functions to demonstrate its applicability and efficiency. These functions comprise 29 from the IEEE CEC2017 suite [83] and 12 additional functions from IEEE CEC2022 [84], categorised into four types: unimodal, multimodal, hybrid, and composition. Each category, featuring single and multiple local solutions, enables a thorough investigation of the algorithm's capability to navigate local optimal challenges. The assessment of algorithm performance is conducted using the mean fitness value (*Mean*) and standard deviation (*SD*). Specifically, *Mean* quantifies the mean error value between the fitness value of the best solution attained by an algorithm and the fitness value associated with the global optimum for a given test function across multiple simulation runs. *SD* measures the consistency of an algorithm in achieving high search precision across these simulation runs. Lower values of *Mean* and *SD* are indicative of an algorithm's ability to consistently and accurately address test functions, respectively.

This study compares the proposed SA-EPO with several well-established algorithms, including EPO [36], SaBO [63], SdSCA [64], SDPSO [65], MSEFA [62], ESO [72], MS-DE [73], FAEPO [43], IEPO [53], ESA [37], CEPO [55], H-EPC [56], SHADE [78], JDE [79], and JADE [80]. The parameter settings for these algorithms were adopted as recommended in their respective original publications. Details of parameter settings are provided in Table 4. To ensure fairness in comparison, the population size (N), maximum number of iterations (I_{max}),

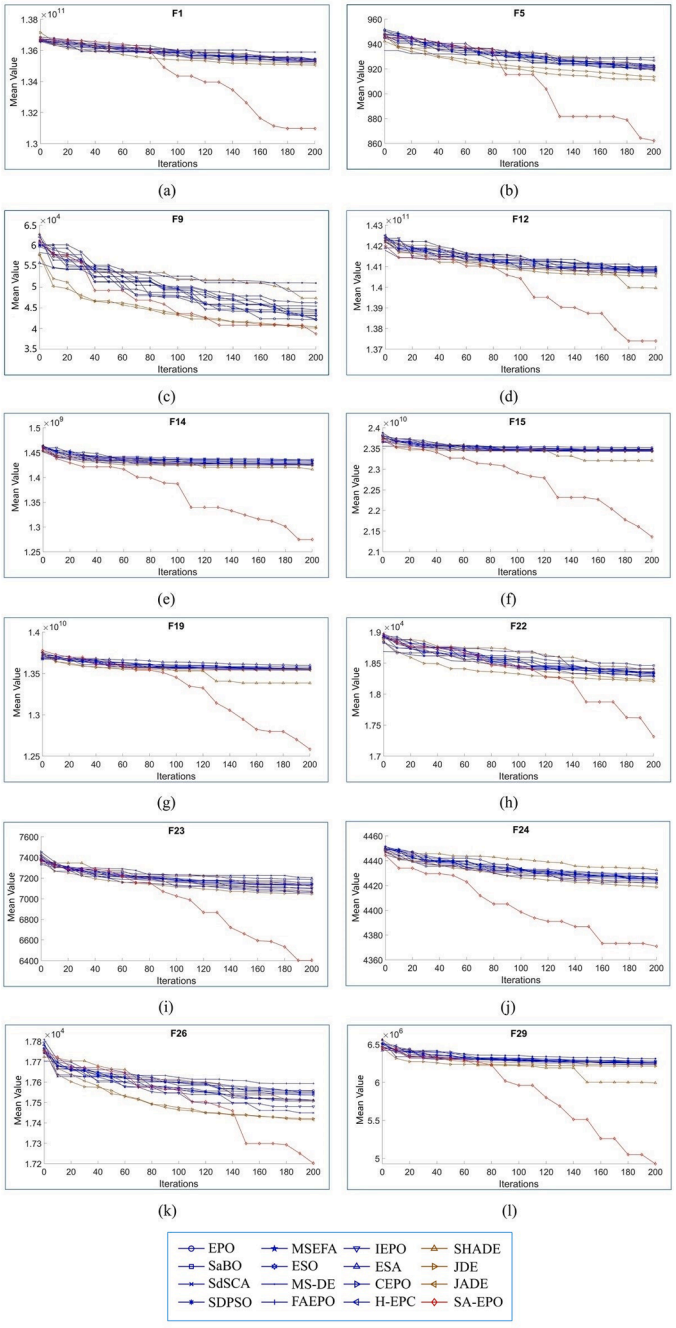

Fig. 10. Convergence analysis of SA-EPO and peer algorithms when solving the CEC2017 benchmark functions of: (a) F1 shifted and rotated bent cigar, (b) F5 shifted and rotated Rastrigin, (c) F9 shifted and rotated Levy, (d) F12 hybrid function 2 ($N = 3$), (e) F14 hybrid function 4 ($N = 4$), (f) F15 hybrid function 5 ($N = 4$), (g) F19 hybrid function 9 (N = 5), (h) F22 composition function 2 ($N = 3$), (i) F23 composition function 3 ($N = 4$), (j) F24 composition function 4 ($N = 4$), (k) F26 composition function 6 ($N = 5$), and (l) F29 composition function 9 ($N = 3$).

and problem dimension (D) are set at 50, 200, and 100 respectively. All simulations were conducted for 30 independent runs for each compared algorithm on a system running Windows 11, equipped with a 64-bit Core i7 processor at 3.20 GHz and 16 GB of RAM, using MATLAB 2020a. The source code for this study is available on GitHub at: https://github.com/OthmanWaleed/SA-EPO.git.

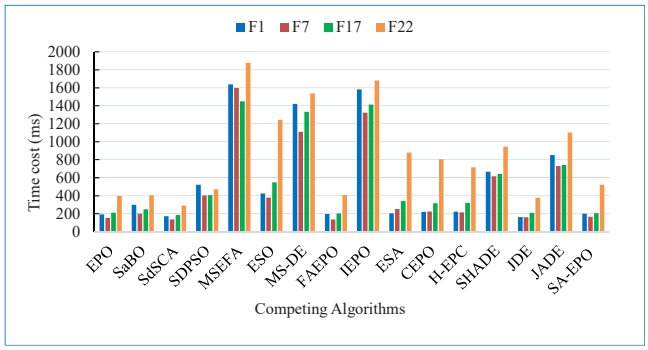

Fig. 11. Computational time consumed by each algorithm in solving the selected benchmark functions.

4.2. Performance evaluation

To demonstrate the efficacy of the SA-EPO algorithm, a comprehensive comparison was conducted with the previously mentioned well-known optimization algorithms using test functions from the CEC2017 and CEC2022 benchmark suites. The results include detailed tables of the *Mean* and *SD* values for the best solutions obtained.

4.2.1. Evaluation of IEEE CEC2017 test functions

An evaluation of the proposed SA-EPO algorithm was first conducted using the IEEE CEC2017 test functions, which are specifically designed to assess optimization algorithm performance. The *Mean* and *SD* values obtained by the proposed SA-EPO on each function are recorded in Table 5, alongside results from other state-of-the-art algorithms. The best simulation results are highlighted in bold, with the second-best results underlined, providing a clear visual representation of the algorithm's performance. The table further summarises the performance comparison between SA-EPO and other algorithms, using *w/t/l* and *#BMF* metrics. Particularly, the *w/t/l* metric indicates that SA-EPO outperforms other algorithms in *w* functions, ties in *t* functions and is outperformed in *l* functions. The *#BMF* counts the test function where each algorithm achieves the best *Mean* results. The proposed SA-EPO consistently outperforms other competing algorithms on the majority of benchmark functions. It emerges as the top performer in 28 out of 29 test functions, specifically F1, F3 to F8, and F10 to F29.

For the unimodal functions F1 and F3, the proposed SA-EPO algorithm demonstrates superior performance, achieving the best *Mean* value for F1 and F3. SHADE and JADE algorithms showed promise in addressing unimodal functions because they yield the second-best results. In the case of multimodal functions ranging from F4 to F10, SA-EPO stands out for its exceptional *Mean* value results in handling these seven functions, generating the six best *Mean* values for functions F4 to F8, and F10. By contrast, the JDE algorithm exhibits promising results in solving multimodal functions by producing the best *Mean* value for functions F9, as well as the second-best *Mean* values for functions F5, F7, and F8. JADE achieved the second-best *Mean* value for functions F6, F9 and F10. Additionally, SHADE demonstrates satisfactory performance in solving function F4 with the second-best *Mean* value.

As for the hybrid functions F11 to F20, the proposed SA-EPO algorithm effectively solves all these functions, achieving the best *Mean* values for all hybrid functions. SHADE has proven competitive in handling hybrid functions, securing the second-best *Mean* value for F11, F12, F15, F17, F18, F19, and F20. JDE scored the second-best *Mean* value for F13 and F14 as a tie with JADE that also attains the second-best *Mean* value F16. For the composite functions F21 to F30, the proposed SA-EPO excels by achieving the best *Mean* values for all 10 functions F21 to F30. JADE records the second-best *Mean* value for F21 and F26. JDE achieves the second-best *Mean* values for F29.

The search efficiency of all competing algorithms can be further

Table 6
Mean and *SD* for 20-*D* benchmark test functions CEC2022.

Function	F1		F2		F3	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	9.29E+ 12	4.60E+ 08	6.94E+ 03	1.54E+ 13	1.43E+ 02	2.93E+ 02
SaBO	9.20E+ 12	3.94E+ 08	6.93E+ 03	1.50E+ 13	1.44E+ 02	2.61E+ 02
SdSCA	9.17E+ 12	2.40E+ 08	6.93E+ 03	1.22E+ 13	1.42E+ 02	3.24E+ 02
SDPSO	9.28E+ 12	3.98E+ 08	6.95E+ 03	1.62E+ 13	1.41E+ 02	3.25E+ 02
MSEFA	9.19E+ 12	3.94E+ 08	6.92E+ 03	1.55E+ 13	1.44E+ 02	3.15E+ 02
ESO	9.22E+ 12	4.84E+ 08	6.91E+ 03	1.73E+ 13	1.40E+ 02	3.15E+ 02
MS-DE	9.21E+ 12	3.46E+ 08	6.92E+ 03	1.41E+ 13	1.41E+ 02	2.83E+ 02
FAEPO	9.15E+ 12	4.03E+ 08	6.90E+ 03	1.72E+ 13	1.41E+ 02	2.97E+ 02
IEPO	9.18E+ 12	3.66E+ 08	6.91E+ 03	1.43E+ 13	1.45E+ 02	3.16E+ 02
ESA	9.14E+ 12	4.32E+ 08	6.90E+ 03	1.72E+ 13	1.43E+ 02	2.94E+ 02
CEPO	9.15E+ 12	4.46E+ 08	6.90E+ 03	1.57E+ 13	1.43E+ 02	2.79E+ 02
H-EPC	9.20E+ 12	3.91E+ 08	6.92E+ 03	1.78E+ 13	1.41E+ 02	3.29E+ 02
SHADE	*8.86E+ 12*	1.74E+ 09	*6.74E+ 03*	4.44E+ 13	*1.29E+ 02*	1.13E+ 03
JDE	9.10E+ 12	3.43E+ 08	6.86E+ 03	1.04E+ 13	1.40E+ 02	2.15E+ 02
JADE	9.11E+ 12	3.36E+ 08	6.84E+ 03	1.26E+ 13	1.39E+ 02	2.13E+ 02
SA-EPO	**8.05E+ 12**	5.18E+ 09	**6.55E+ 03**	7.62E+ 13	**9.74E+ 01**	4.89E+ 03
	F4		F5		F6	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	2.62E+ 02	9.03E+ 00	5.94E+ 03	3.13E+ 00	8.58E+ 09	3.68E+ 01
SaBO	2.59E+ 02	7.20E+ 00	5.99E+ 03	3.41E+ 00	8.54E+ 09	3.65E+ 01
SdSCA	2.59E+ 02	6.42E+ 00	6.11E+ 03	3.05E+ 00	8.54E+ 09	2.93E+ 01
SDPSO	2.64E+ 02	8.50E+ 00	6.15E+ 03	3.12E+ 00	8.59E+ 09	3.30E+ 01
MSEFA	2.60E+ 02	7.83E+ 00	6.12E+ 03	3.05E+ 00	8.55E+ 09	3.57E+ 01
ESO	2.59E+ 02	8.05E+ 00	6.04E+ 03	3.13E+ 00	8.54E+ 09	3.36E+ 01
MS-DE	2.61E+ 02	5.52E+ 00	6.21E+ 03	2.77E+ 00	8.56E+ 09	3.54E+ 01
FAEPO	2.59E+ 02	8.47E+ 00	6.06E+ 03	3.43E+ 00	8.58E+ 09	3.32E+ 01
IEPO	2.63E+ 02	8.20E+ 00	5.81E+ 03	2.79E+ 00	8.58E+ 09	3.59E+ 01
ESA	2.69E+ 02	7.71E+ 00	5.80E+ 03	3.41E+ 00	8.55E+ 09	3.49E+ 01
CEPO	2.59E+ 02	8.70E+ 00	6.03E+ 03	3.34E+ 00	8.59E+ 09	3.88E+ 01
H-EPC	2.59E+ 02	7.56E+ 00	5.66E+ 03	3.53E+ 00	8.57E+ 09	3.40E+ 01
SHADE	*2.39E+ 02*	2.98E+ 01	5.51E+ 03	1.02E+ 01	8.59E+ 09	1.07E+ 02
JDE	2.58E+ 02	5.47E+ 00	5.49E+ 03	2.31E+ 00	8.54E+ 09	2.70E+ 01
JADE	2.56E+ 02	6.30E+ 00	*5.42E+ 03*	2.41E+ 00	*8.52E+ 09*	3.14E+ 01
SA-EPO	**2.19E+ 02**	8.76E+ 01	**5.21E+ 03**	1.48E+ 01	**8.29E+ 09**	1.24E+ 02
	F7		F8		F9	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	6.56E+ 02	8.85E+ 00	1.88E+ 05	4.93E+ 03	4.33E+ 03	2.51E+ 02
SaBO	6.56E+ 02	9.05E+ 00	1.88E+ 05	4.88E+ 03	4.31E+ 03	2.05E+ 02
SdSCA	6.49E+ 02	8.41E+ 00	1.88E+ 05	3.62E+ 03	4.31E+ 03	2.20E+ 02
SDPSO	6.59E+ 02	7.82E+ 00	1.90E+ 05	4.88E+ 03	4.34E+ 03	2.11E+ 02
MSEFA	6.49E+ 02	8.68E+ 00	1.88E+ 05	5.26E+ 03	4.32E+ 03	2.47E+ 02
ESO	6.48E+ 02	8.83E+ 00	1.88E+ 05	4.67E+ 03	4.31E+ 03	1.92E+ 02
MS-DE	6.52E+ 02	7.93E+ 00	1.88E+ 05	3.99E+ 03	4.32E+ 03	1.89E+ 02
FAEPO	6.52E+ 02	9.00E+ 00	1.88E+ 05	4.38E+ 03	4.31E+ 03	2.17E+ 02
IEPO	6.48E+ 02	9.00E+ 00	1.89E+ 05	4.92E+ 03	4.34E+ 03	2.33E+ 02
ESA	6.51E+ 02	8.76E+ 00	1.93E+ 05	4.07E+ 03	4.35E+ 03	2.13E+ 02
CEPO	6.49E+ 02	8.12E+ 00	1.89E+ 05	4.35E+ 03	4.33E+ 03	2.05E+ 02
H-EPC	6.47E+ 02	9.53E+ 00	1.88E+ 05	4.95E+ 03	4.36E+ 03	2.21E+ 02
SHADE	**5.31E+ 02**	3.37E+ 01	*1.83E+ 05*	9.63E+ 03	*4.17E+ 03*	8.46E+ 02
JDE	6.47E+ 02	8.15E+ 00	1.88E+ 05	3.02E+ 03	4.31E+ 03	1.65E+ 02
JADE	6.43E+ 02	8.53E+ 00	1.88E+ 05	3.86E+ 03	4.29E+ 03	1.98E+ 02
SA-EPO	*5.32E+ 02*	9.49E+ 01	**1.75E+ 05**	6.75E+ 03	**3.95E+ 03**	2.68E+ 03
	F10		F11		F12	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	8.23E+ 03	5.41E+ 04	7.90E+ 03	3.98E+ 08	6.41E+ 03	1.92E+ 08
SaBO	8.23E+ 03	2.76E+ 04	7.98E+ 03	3.55E+ 08	6.34E+ 03	1.41E+ 08
SdSCA	8.22E+ 03	3.28E+ 04	7.98E+ 03	3.47E+ 08	6.34E+ 03	1.37E+ 08
SDPSO	8.25E+ 03	5.62E+ 04	7.92E+ 03	4.20E+ 08	6.36E+ 03	1.54E+ 08
MSEFA	8.22E+ 03	4.85E+ 04	7.99E+ 03	3.34E+ 08	6.37E+ 03	1.96E+ 08
ESO	8.28E+ 03	3.80E+ 04	7.98E+ 03	3.82E+ 08	6.34E+ 03	1.83E+ 08
MS-DE	8.22E+ 03	2.41E+ 04	7.84E+ 03	2.49E+ 08	6.39E+ 03	2.06E+ 08
FAEPO	8.24E+ 03	4.99E+ 04	7.97E+ 03	4.27E+ 08	6.37E+ 03	1.77E+ 08
IEPO	8.21E+ 03	3.53E+ 04	7.93E+ 03	3.37E+ 08	6.39E+ 03	1.74E+ 08
ESA	8.29E+ 03	4.85E+ 04	7.86E+ 03	3.44E+ 08	6.47E+ 03	1.89E+ 08
CEPO	8.29E+ 03	5.85E+ 04	7.86E+ 03	3.59E+ 08	6.35E+ 03	1.49E+ 08
H-EPC	8.26E+ 03	6.44E+ 04	7.89E+ 03	3.13E+ 08	6.35E+ 03	1.64E+ 08
SHADE	*8.09E+ 03*	2.05E+ 05	*7.54E+ 03*	1.56E+ 09	*6.23E+ 03*	1.02E+ 09
JDE	8.18E+ 03	4.81E+ 04	7.86E+ 03	2.29E+ 08	6.33E+ 03	1.34E+ 08

(*continued on next page*)

Table 6 (*continued*)

JADE	8.14E+ 03	2.98E+ 04	7.83E+ 03	3.02E+ 08	6.35E+ 03	1.27E+ 08
SA-EPO	**7.87E+ 03**	1.27E+ 05	**7.45E+ 03**	5.50E+ 09	**6.15E+ 03**	3.72E+ 09
	w/t/l					*#BMF*
EPO	12/0/0					0
SaBO	12/0/0					0
SdSCA	12/0/0					0
SDPSO	12/0/0					0
MSEFA	12/0/0					0
ESO	12/0/0					0
MS-DE	12/0/0					0
FAEPO	12/0/0					0
IEPO	12/0/0					0
ESA	12/0/0					0
CEPO	12/0/0					0
H-EPC	12/0/0					0
SHADE	11/0/1					1
JDE	12/0/0					0
JADE	12/0/0					0
SA-EPO						11

enhanced through detailed visual examination. Convergence curves displayed in Fig. 10 show each algorithm's search efficiency in tackling the challenging CEC2017 benchmark test functions, specifically F1, F5, F9, F12, F14, F15, F19, F22, F23, F24, F26, and F29. These curves facilitate an in-depth analysis of the algorithms' convergence behaviors. Notably, the proposed SA-EPO demonstrates competitive convergence speeds, effectively identifying promising solution regions of these benchmark functions at various stages of the optimization process. These empirical results corroborate the findings in Table 5, thereby reinforcing the efficacy of the SA-EPO approach.

An evident pattern of steep decline can be observed from all convergence curves of SA-EPO, highlighting its exceptional ability to swiftly identify promising solution regions of the benchmark functions in the early stage of the optimization process. Fig. 10 (c) demonstrates SA-EPO's robust diversity preservation capability through the consistent spread of solutions during the optimization process. Diversity is evident in the ability of the algorithm to maintain exploration across the search space, as observed in the convergence patterns. Unlike other algorithms that converge prematurely or exhibit stagnation, SA-EPO balances exploration and exploitation effectively, avoiding getting trapped in local optima. These curves also illustrates how SA-EPO achieves a gradual decline in the mean value during the early iterations while maintaining variability, showcasing its capacity to explore diverse regions of the search space. This diversity helps improve convergence toward the global optimum in later iterations.

To rigorously evaluate the computational complexity of the SA-EPO algorithm, an experiment using distinct test functions selected from the CEC2017 benchmark, namely, F1, F7, F17, and F22, are performed. These test functions, representing unimodal, multimodal, hybrid, and composition types, were specifically chosen to demonstrate the algorithm's performance across varied scenarios. The experiments aim to measure the CPU runtime, quantified in milliseconds, to analyse the execution time of the SA-EPO algorithm relative to its peers. As illustrated in Fig. 11, the results highlight the SA-EPO's computational efficiency, showing its runtime is generally lower than competing algorithms such as SaBO, SDPSO, MSEFA, ESO, MS-DE, IEPO, ESA, CEPO, H-EPC, SHADE, and JADE. These findings underscore the SA-EPO's capability to maintain competitive execution times while delivering robust optimization results. The overall time complexity of SA-EPO is considered acceptable and compares favourably with other well-regarded algorithms within the same computational context. The detailed comparison of algorithmic performance, coupled with the time complexity data, conclusively shows that SA-EPO not only achieves superior optimization outcomes but does so with a computational demand comparable to its contemporaries. This balance of efficiency and effectiveness positions SA-EPO as a viable solution for addressing complex optimization problems, with its computational burden influenced by key factors such as the number of iterations, population size, and problem dimensionality each essential to understanding the algorithm's operational efficacy.

4.2.2. Evaluation of IEEE CEC2022 test functions

An additional evaluation of the proposed SA-EPO algorithm is performed using the CEC2022 benchmark functions. These functions are designed to evaluate the effectiveness of optimization algorithms and are considered more challenging than those from earlier benchmarks. The *Mean* and *SD* values achieved by SA-EPO for each function are presented in Table 6. The problem dimension D is set at 20 for all algorithms compared. The best results are highlighted in bold, while the second-best results are underlined. Table 6 also summarises SA-EPO's performance relative to its peers in terms of wins (w), ties (t), and losses (l). The *#BMF* indicates the number of benchmark functions where each algorithm achieves the best *Mean* results. Upon careful examination, the performance of SA-EPO is notably impressive. The analysis of the results demonstrates that SA-EPO outperforms other algorithms on 11 benchmark functions.

For the unimodal functions F1 and F2, the proposed SA-EPO exhibits superior performance, achieving the best *Mean* value for both functions. SHADE also shows good potential in handling unimodal functions by producing the best *Mean* value for F1 and F2 functions. For multimodal functions F3 to F5, the proposed SA-EPO continues to excel, and it achieves the best *Mean* values for all three functions. Notably, SHADE outperforms others in addressing functions F3 and F4 by achieving the second-best *Mean* value, with JADE closely following with the second-best *Mean* results for the F5 function.

In the domain of hybrid functions F6 to F8, the proposed SA-EPO algorithm effectively achieves the best *Mean* values for F6 and F8 functions, indicating its robust performance, as well as the second-best *Mean* value for F7. SHADE also demonstrates strong performance in this function category, securing the best *Mean* value for F7, in addition to the second-best *Mean* value for F8, while JADE achieves the second-best *Mean* value for F6. Moving to the composite functions F9 to F12, the proposed SA-EPO continues to excel and produces the best *Mean* values for all four functions. SHADE performs relatively well by achieving the second-best *Mean* value for all the composite functions.

Fig. 12 illustrates the convergence curves of the proposed SA-EPO and other competing algorithms on the CEC2022 benchmark functions (F1 to F12). The proposed SA-EPO demonstrates a consistent pattern of rapid convergence across all functions, emphasising its ability to quickly identify promising solution regions in the early stages of the optimization process. These observations corroborate the results presented in Table 6, thus affirming the efficacy of SA-EPO. Notably, Fig. 12(g)

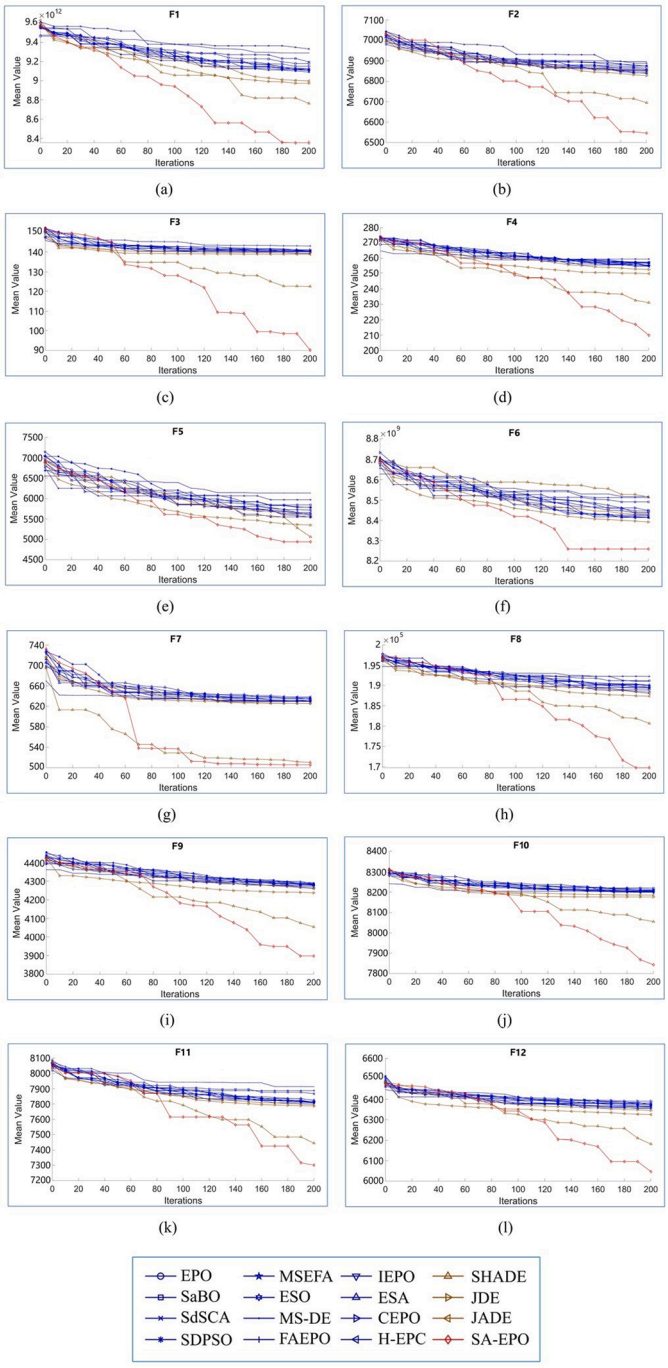

Fig. 12. Convergence analysis of SA-EPO and peer algorithms when solving the CEC2022 benchmark functions of: (a) F1 shifted and rotated Zakharov, (b) F2 shifted and rotated Rosenbrock, (c) F3 Schaffer F7, (d) F4 step Rastrigin, (e) F5 Levy, (f) F6 hybrid function (HF02), (g) F7 hybrid function (HF10), (h) F8 hybrid function (HF06), (i) F9 composition function (CF01), (j) F10 composition function (CF02), (k) F11 composition function (CF06) and (l) F12 composition function (CF07).

demonstrates SA-EPO's capacity for maintaining competitive diversity preservation. The proposed algorithm effectively escapes from local optima, maintaining agility in its search even after temporary entrapments in suboptimal regions. These findings highlight the algorithm's superior performance and robustness in addressing complex optimization challenges, offering a clear advantage over existing optimization algorithms.

Table 7
Results obtained by the Wilcoxon signed rank test for the SA-EPO algorithm (CEC2017).

SA-EPO vs.	$R+$	$R-$	P-value	h-value
EPO	435.0	00.0	2.00E−06	+
SaBO	435.0	00.0	2.00E−06	+
SdSCA	435.0	00.0	2.00E−06	+
SDPSO	435.0	00.0	2.00E−06	+
MSEFA	435.0	00.0	2.00E−06	+
ESO	435.0	00.0	2.00E−06	+
MS-DE	435.0	00.0	2.00E−06	+
FAEPO	435.0	00.0	2.00E−06	+
IEPO	435.0	00.0	2.00E−06	+
ESA	435.0	00.0	2.00E−06	+
CEPO	435.0	00.0	2.00E−06	+
H-EPC	435.0	00.0	2.00E−06	+
SHADE	435.0	00.0	2.00E−06	+
JDE	427.0	08.0	5.00E−06	+
JADE	327.0	08.0	4.00E−06	+

Table 8
Results obtained by the Wilcoxon signed rank test for the SA-EPO algorithm (CEC2022).

SA-EPO vs.	$R+$	$R-$	P-value	h-value
EPO	78.0	00.0	1.94E−03	+
SaBO	78.0	00.0	1.75E−03	+
SdSCA	78.0	00.0	1.94E−03	+
SDPSO	78.0	00.0	1.94E−03	+
MSEFA	78.0	00.0	1.75E−03	+
ESO	78.0	00.0	1.75E−03	+
MS-DE	78.0	00.0	1.75E−03	+
FAEPO	78.0	00.0	1.94E−03	+
IEPO	78.0	00.0	1.94E−03	+
ESA	78.0	00.0	1.94E−03	+
CEPO	78.0	00.0	1.94E−03	+
H-EPC	78.0	00.0	1.94E−03	+
SHADE	77.0	01.0	2.29E−03	+
JDE	78.0	00.0	1.75E−03	+
JADE	78.0	00.0	1.94E−02	+

4.3. Statistical analyses

Non-parametric statistical analyses are performed to compare the proposed SA-EPO comprehensively against its peer algorithms. The Wilcoxon signed rank test facilitates pairwise comparisons between SA-EPO and each competing algorithm. Additionally, the Friedman test [85] is applied to evaluate the average rank of all algorithms, enabling multiple comparison using non-parametric statistical methods. Subsequent post-hoc analyses, including the Bonferroni–Dunn, Holm and Hochberg methods, are conducted to identify significant differences between the proposed SA-EPO and each of its peers, further elucidating its comparative performances.

4.3.1. Wilcoxon signed rank test

To highlight the benefits of SA-EPO, the rankings of each algorithm across every test function have been computed. Tables 7 and 8 display the outcomes of the Wilcoxon signed rank test for SA-EPO within the CEC2017 and CEC2022 benchmarks, respectively. This test provides pairwise comparisons between SA-EPO and competing algorithms at a significant level of $\alpha = 0.05$, presenting metrics such as $R+$, $R-$, p *and* h values. Specifically, $R+$ represents the total rank for cases where the SA-EPO algorithm outperformed its peers, while $R-$ represents the total rank where it underperformed. The raised values in these tables are due to the scoring methodology of the ranking process. In the Wilcoxon test, the sum of ranks can accumulate to relatively high numbers when assessing performance across multiple comparisons or algorithms. Each algorithm's performance is ranked separately based on their results, leading to significant sums. The p-value indicates the probability of

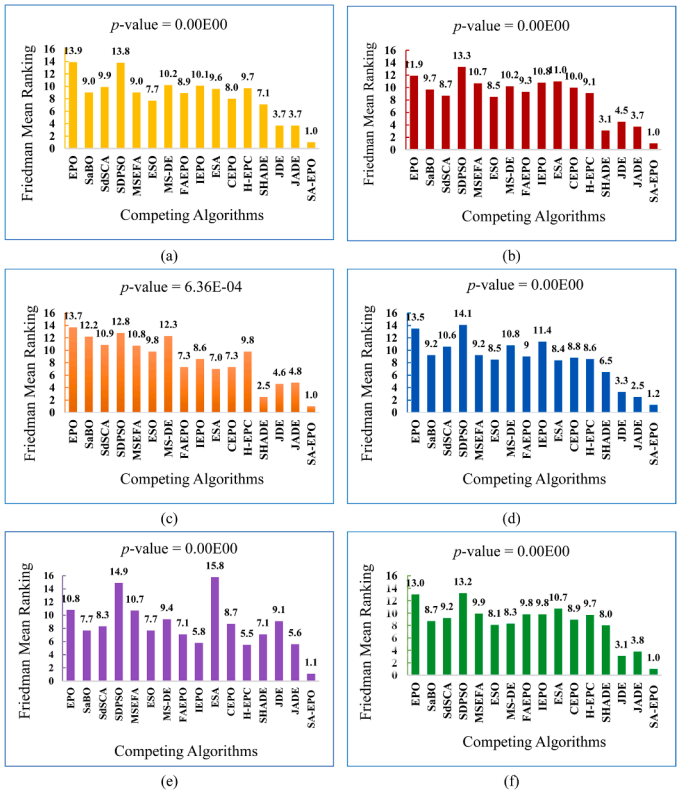

Fig. 13. Friedman mean rank for: (a) all CEC2017 benchmark functions, (b) all CEC2022 benchmark functions, (c) unimodal functions selected from CEC2017 and CEC2022, (d) multimodal functions selected from CEC2017 and CEC2022, (e) hybrid functions selected from CEC2017 and CEC2022 and (f) composition functions selected from CEC2017 and CEC2022.

Table 9
p-values obtained for Bonferroni–Dunn, Holm and Hochberg procedures (CEC2017).

SA-EPO vs.	Unadjusted *p*	Bonferroni-Dunn *p*	Holm *p*	Hochberg *p*
EPO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
SDPSO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
MS-DE	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
IEPO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
SdSCA	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
H-EPC	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
ESA	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
MSEFA	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
SaBO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
FAEPO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
CEPO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
ESO	0.00E+ 00	1.00E−05	0.00E+ 00	0.00E+ 00
SHADE	1.00E−05	1.70E−04	3.00E−05	3.00E−05
JDE	2.93E−02	4.40E−01	5.87E−02	3.37E−03
JADE	3.37E−02	5.05E−01	5.87E−02	3.37E−02

observing the data if no real effect exists, aiding in identifying significant performance differences. Performance is statistically significant if the *p*-value is smaller than α. The *h* value, derived from the *p* and α values, indicates whether SA-EPO's performance is significantly superior (*h* = '+'), not significant (*h* = '=') or significantly inferior (*h* = '−') relative to its peers.

The statistical results from the Wilcoxon signed rank test, as detailed in Tables 7 and 8, confirm the superior performance of SA-EPO over other peer algorithms including EPO, SaBO, SdSCA, SDPSO, MSEFA, ESO, MS-DE, FAEPO, IEPO, ESA, CEPO, H-EPC, SHADE, JDE and JADE. across CEC2017 and CEC2022 benchmark functions. Notably, SA-EPO

Table 10
p-values obtained for Bonferroni–Dunn, Holm and Hochberg procedures (CEC2022).

SA-EPO vs.	Unadjusted *p*	Bonferroni-Dunn *p*	Holm *p*	Hochberg *p*
SDPSO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
EPO	0.00E+ 00	0.00E+ 00	0.00E+ 00	0.00E+ 00
ESA	0.00E+ 00	5.00E−06	4.00E−06	4.00E−06
IEPO	1.00E−06	0.00E−06	7.00E−06	7.00E−06
MSEFA	1.00E−06	1.10E−05	8.00E−06	8.00E−06
MS-DE	3.00E−06	4.00E−05	2.70E−05	2.70E−05
CEPO	4.00E−06	6.10E−04	3.60E−05	3.60E−05
SaBO	8.00E−06	1.24E−03	6.60E−05	6.60E−05
FAEPO	2.20E−05	3.28E−03	1.53E−04	1.53E−04
H-EPC	3.50E−05	5.27E−03	2.11E−04	2.11E−04
SdSCA	8.00E−05	4.00E−03	4.00E−04	4.00E−04
ESO	1.14E−04	4.56E−03	4.56E−04	4.56E−04
JDE	7.88E−02	2.36E−01	2.36E−01	2.36E−01
JADE	1.77E−01	3.53E−01	3.53E−01	2.83E−01
SHADE	2.84E−01	3.53E−01	3.53E−01	2.83E−01

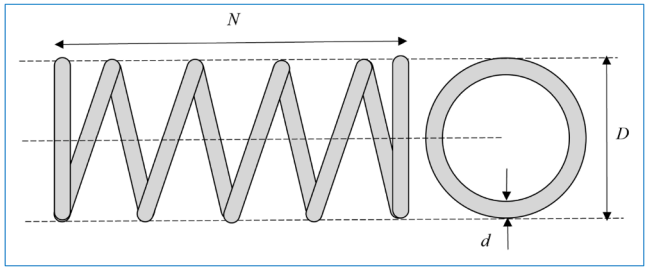

Fig. 14. Schematic view of spring design problem.

demonstrates a significant advantage, evidenced by the positive *h* values ("+") at a significance level of α = 0.05, when solving both sets of benchmark functions compared to other algorithms. These findings affirm substantial performance differences between the proposed SA-EPO and its competitors, providing robust insights into its effectiveness across diverse optimization scenarios.

4.3.2. *Friedman test*

The Friedman test is another non-parametric statistical analysis employed to determine the mean rank values of all competing algorithms. The Friedman mean rank results for all competing algorithms on CEC2017 benchmark test functions are displayed in Fig. 13-a. The rankings place SA-EPO and its counterparts based on their mean ranks as follows: SA-EPO, JADE, JDE, SHADE, ESO, CEPO, FAEPO, SaBO, MSEFA, MS-DE, ESA, H-EPC, SdSCA, IEPO, SDPSO and EPO. The *p*-value from the Friedman test is reported as 0.00E+ 00, signifying statistical significance at a level below α = 0.05. These results confirm significant performance differences among the algorithms from a global perspective.

Additionally, Friedman mean rank assessments for the CEC2022 benchmark test functions are conducted and illustrated in Fig. 13-b. The rankings place SA-EPO and its counterparts as follows: SA-EPO, SHADE, JADE, JDE, ESO, SdSCA, H-EPC, FAEPO, SaBO, CEPO, MS-DE, MSEFA, IEPO, ESA, EPO and SDPSO. The derived *p*-value is reported as 0.00E+ 00, indicating statistical significance below the threshold α = 0.05, thus confirming substantial differences among the algorithms. Furthermore, specific Friedman rankings are conducted for different types of test functions, namely, unimodal, multimodal, hybrid and composite, as shown in Fig. 12-c, d, e and f, respectively. Their corresponding *p*-values are 6.36E-04, 0.00E+ 00, 0.00E+ 00, 0.00E+ 00, all indicating significant differences at levels below α = 0.05. These four types of functions are selected from CEC2017 and CEC2022. The results highlight SA-EPO's superior performance across all function types, demonstrating its enhanced balance between exploitation and

Table 11
Simulation results obtained from competing algorithms to solve seven real-world engineering design problems.

Function	Spring Design		Three Bar Truss Design		Gear Train Design	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	2.82E−02	1.61E+ 13	263.897	3.11E+ 02	2.35E−08	8.23E+ 00
SaBO	2.24E−02	7.09E+ 13	263.896	2.93E+ 03	2.33E−08	5.45E+ 01
SdSCA	2.20E−02	6.29E+ 13	263.896	1.91E+ 03	2.34E−08	4.76E+ 01
SDPSO	2.82E−02	1.69E+ 13	263.897	3.12E+ 02	2.37E−08	8.03E+ 00
MSEFA	2.16E−02	6.59E+ 13	263.895	3.06E+ 03	2.32E−08	7.03E+ 01
ESO	2.13E−02	7.04E+ 13	263.895	3.42E+ 03	2.31E−08	6.54E+ 01
MS-DE	**1.90E−02**	6.38E+ 13	263.895	3.22E+ 03	2.31E−08	5.39E+ 01
FAEPO	2.79E−02	1.03E+ 13	263.897	2.40E+ 02	2.36E−08	5.00E+ 00
IEPO	2.72E−02	1.26E+ 13	263.897	2.39E+ 02	2.25E−08	5.42E+ 00
ESA	2.72E−02	1.50E+ 13	263.897	2.18E+ 02	2.32E−08	6.14E+ 00
CEPO	2.96E−02	1.28E+ 13	263.897	2.03E+ 02	2.36E−08	5.62E+ 00
H-EPC	2.75E−02	1.12E+ 13	263.897	2.13E+ 02	2.35E−08	4.94E+ 00
SHADE	2.23E−02	6.74E+ 13	*263.894*	2.35E+ 03	*2.16E−08*	5.93E+ 01
JDE	2.75E−02	1.04E+ 13	263.896	2.16E+ 02	2.36E−08	5.92E+ 00
JADE	2.72E−02	1.22E+ 13	263.897	2.22E+ 02	2.39E−08	5.53E+ 00
SA-EPO	*1.91E−02*	7.43E+ 13	**263.893**	4.12E+ 03	**2.05E−08**	6.03E+ 01
	Cantilever Beam Design		Pressure Vessel Design		Speed Reducer Design	
	Mean	*SD*	*Mean*	*SD*	*Mean*	*SD*
EPO	1.33998	3.67E+ 00	3.35E+ 04	3.09E+ 01	4.33E+ 03	8.76E+ 00
SaBO	1.33997	1.31E+ 01	*2.94E+ 04*	1.12E+ 02	3.97E+ 03	5.05E+ 01
SdSCA	1.33998	1.12E+ 01	3.12E+ 04	1.22E+ 02	4.12E+ 03	5.71E+ 01
SDPSO	1.33998	3.46E+ 00	3.49E+ 04	3.38E+ 01	4.40E+ 03	8.23E+ 00
MSEFA	*1.33995*	1.30E+ 01	3.15E+ 04	1.28E+ 02	4.10E+ 03	7.56E+ 01
ESO	*1.33995*	1.21E+ 01	3.53E+ 04	1.15E+ 02	**3.57E+ 03**	7.36E+ 01
MS-DE	1.33997	1.24E+ 01	3.25E+ 04	1.08E+ 02	4.12E+ 03	6.87E+ 01
FAEPO	1.33997	2.40E+ 00	3.63E+ 04	2.70E+ 01	4.03E+ 03	6.47E+ 00
IEPO	1.33997	2.33E+ 00	3.62E+ 04	3.08E+ 01	4.51E+ 03	6.67E+ 00
ESA	1.33998	2.89E+ 00	3.25E+ 04	3.20E+ 01	4.58E+ 03	8.37E+ 00
CEPO	1.33998	2.88E+ 00	3.33E+ 04	2.72E+ 01	*3.73E+ 03*	6.52E+ 00
H-EPC	1.33997	2.18E+ 00	3.54E+ 04	2.96E+ 01	3.94E+ 03	7.88E+ 00
SHADE	1.33996	1.26E+ 01	**2.43E+ 04**	1.14E+ 02	3.78E+ 03	4.68E+ 01
JDE	1.33997	1.57E+ 00	3.52E+ 04	2.76E+ 01	4.38E+ 03	6.89E+ 00
JADE	1.33998	2.44E+ 00	3.36E+ 04	2.60E+ 01	4.32E+ 03	8.06E+ 00
SA-EPO	**1.33992**	1.31E+ 01	**2.43E+ 04**	1.30E+ 02	3.75E+ 03	1.18E+ 02
	Rolling Element Bearing Design					
	Mean	*SD*				
EPO	1.43E+ 18	1.92E+ 02				
SaBO	1.23E+ 18	1.69E+ 03				
SdSCA	1.24E+ 18	1.27E+ 03				
SDPSO	1.42E+ 18	2.08E+ 02				
MSEFA	**1.08E+ 18**	1.58E+ 03				
ESO	**1.08E+ 18**	2.04E+ 03				
MS-DE	1.24E+ 18	1.86E+ 03				
FAEPO	1.31E+ 18	2.00E+ 02				
IEPO	1.42E+ 18	1.87E+ 02				
ESA	1.44E+ 18	1.75E+ 02				
CEPO	1.43E+ 18	1.94E+ 02				
H-EPC	1.45E+ 18	1.60E+ 02				
SHADE	1.22E+ 18	1.25E+ 03				
JDE	1.40E+ 18	1.21E+ 02				
JADE	1.41E+ 18	2.21E+ 02				
SA-EPO	*1.11E+ 18*	2.44E+ 03				

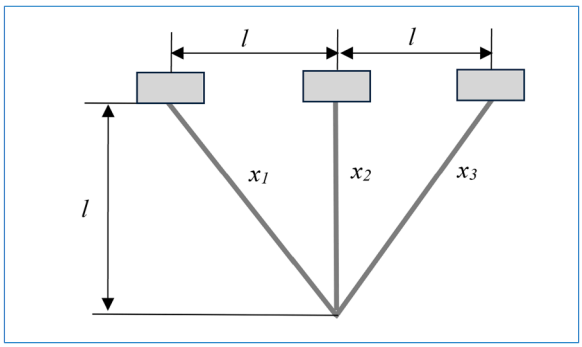

Fig. 15. Schematic view of three-bar truss problem.

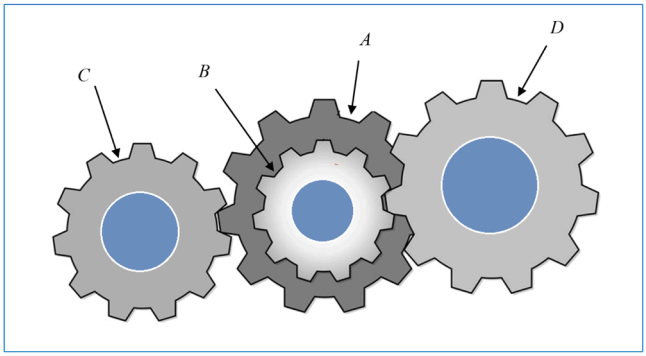

Fig. 16. Schematic view of gear train design problem.

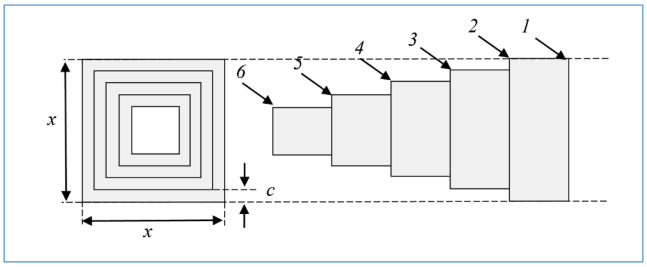

Fig. 17. Schematic view of cantilever beam design problem.

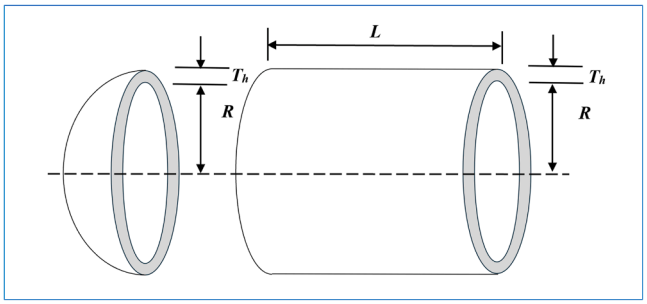

Fig. 18. Schematic view of pressure vessel design problem.

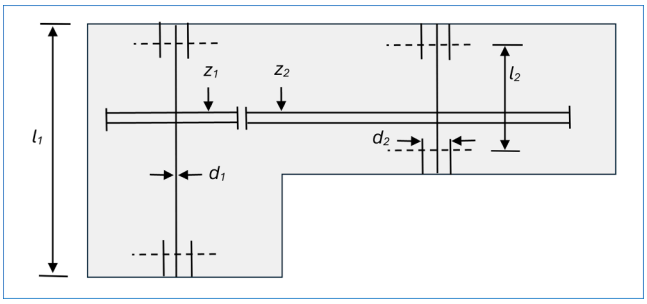

Fig. 19. Schematic view of speed reducer design problem.

exploration, which outperforms competing algorithms.

Referring to the outcomes of the Friedman tests, significant global differences among the algorithms are detected based on the p and α values. Further analyses are conducted using Bonferroni–Dunn, Holm and Hochberg post-hoc tests to examine performance discrepancies through adjusted p-values. These non-parametric statistical tests are employed to identify significant differences among competing algorithms following an initial analysis, such as the Friedman test, indicating overall variances among algorithms. The results, displayed in Tables 9 and 10, present the findings from these post-hoc comparisons for the CEC2017 and CEC2022 benchmarks, respectively. Given the significance level $\alpha = 0.05$, all post-hoc tests substantiate the exceptional performance of SA-EPO relative to its competitors.

4.4. Application of the proposed SA-EPO in real-world engineering design problems

To substantiate the effectiveness and robustness of the proposed SA-EPO algorithm, this subsection conducts a detailed comparative analysis across seven complex engineering optimization challenges. This comparative study aims to demonstrate SA-EPO's remarkable capability in solving complex real-world optimization problems. Each simulation involves over 200 iterations, utilising a population size of 50 agents, and is repeated for 30 independent runs to ensure robust statistical validity.

These seven engineering challenges encompass diverse constraints and problem natures, presenting a significant test of any optimization

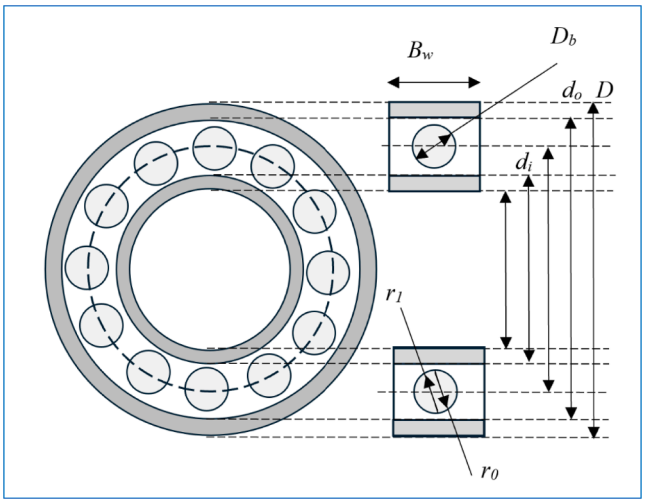

Fig. 20. Schematic view of rolling element bearing design problem.

algorithm's ability to balance feasibility with optimality. Effective constraint management is crucial because algorithms must search extensive spaces to pinpoint solutions that not only meet all constraints

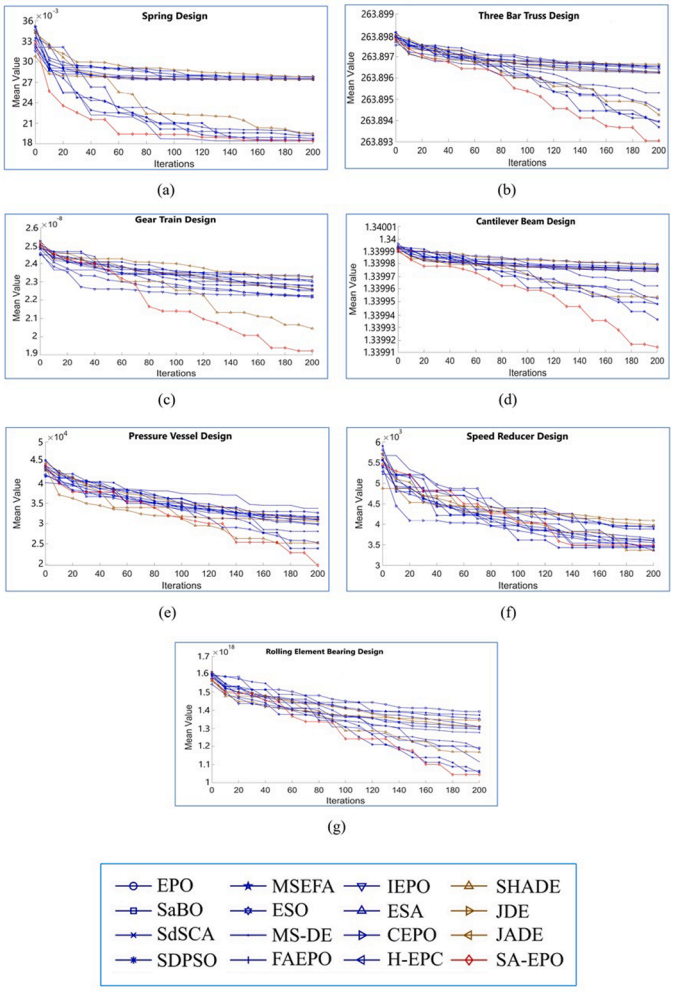

Fig. 21. Convergence analysis of SA-EPO and peer algorithms when solving seven real-world engineering problems of (a) Spring Design, (b) Three-Bar Truss Design, (c) Gear Train Design, (d) Cantilever Beam Design (e) Pressure Vessel Design, (f) Speed Reducer Design (g) Rolling Element Bearing Design.

but also optimize the objective functions. The complexity, dimensionality and number of constraints inherent in these problems escalate the difficulty of this task. To manage these constraints, various penalty functions are employed [86], such as static penalty, dynamic penalty, annealing penalty, adaptive penalty, co-evolutionary penalty and death penalty. Particularly, the death penalty function is noteworthy for its straightforwardness and minimal computational demand, where it discards infeasible solutions and does not utilise any information from such solutions [36]. The proposed SA-EPO and other competing algorithms utilise this death penalty function to effectively handle constrained engineering design problems. The formulation of the penalised objective function $F(x)$ is given by

$$F(x) = f(x) + P(x) \tag{23}$$

where $f(x)$ is the original objective function, and $P(x)$ is the penalty term, defined as

$$P(x) = \sum_{i=1}^{m} \lambda_i g_i(x)^2 + \sum_{j=1}^{n} \mu_j h_j(x)^2 \tag{24}$$

where $g_i(x) \leq 0$ represents inequality constraints and $h_j(x) = 0$ represents equality constraints, with $\lambda_i$andμ_j being the respective penalty coefficients. These coefficients are carefully chosen to strike a balance between ensuring constraint satisfaction and optimizing the objective function. High penalty coefficients aggressively penalise constraint violations to ensure solution feasibility, but may impede convergence by excessively focusing on constraint satisfaction. Conversely, low coefficients may allow infeasible solutions, thereby compromising the effectiveness of the algorithm.

4.4.1. Spring design problem

The spring design problem is a classic constrained optimization problem. This problem consists of minimising the weight of a tension/compression spring subject to constraints on shear stress, surge frequency and minimum deflection. It involves three decision variables: wire diameter (d), average coil diameter (D) and the number of active coils (N) as illustrated in Fig. 14. This problem requires the optimization algorithm to maintain a fine balance between exploration and exploitation to navigate the feasible region effectively. The mathematical formulation of this problem is presented in Eq. (25):

$$\text{Minimise} : f_1(x) = \left(0.25\pi^2 x_2{}^2 x_3 (\quad x_1 + 2)\right) \tag{25}$$

$$\text{Minimise} : f_2(x) = 8K \quad P_{\max} \frac{x_3}{\pi \quad x_2{}^3}$$

$$\text{Subjected to} : g_1(x) = 1.05 x_2 (x_1 + 2) + k P_{\max} - l_{\max}$$

$$g_2(x) = d_{\min} - x_2$$

$$g_3(x) = x_2 + x_3 - D_{\max}$$

$$g_4(x) = 3 \quad - \quad C$$

$$g_5(x) = del_p \quad - \quad del_p$$

$$g_6(x) = del_w \quad - \quad \frac{P_{\max} \quad - \quad P}{k}$$

$$g_7(x) = 8K \quad P_{\max} \frac{x_3}{Pix_2{}^3} \quad - \quad S$$

$$g_8(x) = \left(0.25pi^2 x_2{}^2 x_3 (\quad x_1 + 2)\right) \quad - \quad V_{\max}$$

$$\text{where} : P = 300, \quad D_{\max} = 3, \quad V_{\max} = 30, \quad P_{\max} = 1000, \quad del_w = 1.25,$$

$$I_{\max} = 14, \quad del_{pm} = 6, \quad d_{\min} = 0.2, \quad S = 189000, \quad G = 11500000,$$

$$C = \frac{x_3}{x_2}, k = \quad G * \frac{x_2{}^4}{8 x_1 x_2{}^3} \quad , \quad del_p = \quad \frac{P}{k}, \quad K \quad \frac{4C-1}{4C-4} \quad + \left(0.615 \frac{x_2}{x_3}\right)$$

$$\forall \quad g \leq 0; 1 \leq x_1 \leq 32; 1 \leq x_3 \leq 30; x_1 \in \quad I$$

$x_2 \in \{0.009, 0.0095, 0.0104, 0.0118, 0.0128, 0.0132, 0.014, 0.015, 0.0162, 0.0173, 0.018, 0.020, 0.023, 0.025, 0.028, 0.032, 0.035, 0.041, 0.047, 0.054, 0.063, 0.072, 0.080, 0.092, 0.105, 0.120, 0.135, 0.148, 0.162, 0.177, 0.192, 0.207, 0.225, 0.244, 0.263, 0.283, 0.307, 0.331, 0.362, 0.394, 0.4375, 0.5\}$

The simulation results of applying SA-EPO and other competing algorithms to the spring design problem are detailed in Table 11, Notably, SA-EPO produces a competitive spring weight that meets the required constraints. Furthermore, the convergence curves shown in Fig. 21-a emphasise the superior performance of the proposed SA-EPO in comparison with most of its competitors.

4.4.2. Three-bar truss problem

The primary objective of the three-bar truss design problem is to identify the minimum weight configuration necessary for constructing a three-bar truss. This challenge involves optimizing two distinct decision variables while adhering to various constraints. The schematic view of this problem is illustrated in Fig. 15. This problem is a constrained optimization scenario where feasible solutions occupy a small fraction of the search space. Detailed information about the mathematical framework and the essential constraints associated with these parameters is provided in Eq. (26):

$$\text{Minimise} : f(x) = (2\sqrt{2x_1} + x_2) * l \tag{26}$$

$$\text{Subjected to} : g_1(x) = \frac{\sqrt{2x_1} + x_2}{\sqrt{2x_1{}^2 + 2x_1 x_2}} \quad P - \sigma \leq 0$$

$$g_2(x) \quad = \quad \frac{x_2}{\sqrt{2x_1{}^2 + 2x_1 x_2}} \quad P \quad - \sigma \leq 0$$

$$g_3(x) \quad = \quad \frac{1}{\sqrt{2x_2 \quad + 2x_1}} \quad P \quad - \sigma \leq 0$$

$$\text{where} : l \quad = \quad 100cm, P = \quad 2KN/cm^2, \ \sigma \quad = \quad 2KN/cm^2, \ 0 \leq x_1, \ x_2 \leq 1$$

The outcomes of the comparative analysis between SA-EPO and other peer algorithms for the three-bar truss problem are presented in Table 11. The proposed SA-EPO has proven to be the most effective algorithm to tackle this design problem, surpassing other competitors such as MS-DE, ESO, MSEFA, SaBO, SHADE, JDE and JADE. Fig. 21-b illustrates the convergence characteristic of SA-EPO algorithm and its counterparts, clearly demonstrating SA-EPO's superior ability to tackle the complexities of the three-bar truss problem effectively.

4.4.3. Gear train design problem

The gear train design problem is a fundamental challenge in structural optimization within engineering and mechanical systems. The main objective in optimizing gear trains is to determine the optimal number of teeth for each gear in the system. This problem is characterised by four integer decision variables involving optimizing two distinct objective functions simultaneously. This engineering problem introduces discrete variables and nonlinear relationships, further increasing complexity. The overall schematic of this problem is illus-

trated in Fig. 16. A detailed mathematical representation of this problem is provided in Eq. (27), as shown below:

$$\text{Minimise} : f_1(x) = \max(x_1, x_2, x_3, x_4) \quad (27)$$

$$\text{Minimise} : f_2(x) = \left(\left(\frac{1}{6.931}\right) - \left(\frac{x_1 x_2}{x_3 x_4}\right)\right)^2$$

$$\text{where} : 12 \leq x_1, x_2, x_3, x_4 \leq 60$$

The first objective function in the gear train design problem seeks to minimise the maximum size of any individual gear, while the second objective function focuses on minimising the error in the gear ratio relative to a predefined reference of 1/6.931 [87]. The effectiveness of various algorithms in solving the gear train design problem is evaluated and presented in Table 11, where SA-EPO demonstrates superior performance in handling this optimization challenge. Additionally, Fig. 21-c illustrates the convergence behavior of SA-EPO, highlighting its dominance over most competing algorithms. This analysis affirms the SA-EPO's efficacy and reliability in addressing the design issue of gear trains, establishing it as a robust solution in structural optimization and engineering applications.

4.4.4. Cantilever beam design problem

The Cantilever Beam Design problem (CBD) has been extensively studied within intelligent optimization algorithms. The cantilever beam is composed of five square hollow sections, where the first section is fixed and the fifth bears a vertical load. The objective of this design problem is to minimise the weight of the beam while maintaining structural integrity. This involves optimizing five design variables (x_1, x_2, x_3, x_4, x_5) that define the dimensions of each section's cross-sectional area, subject to constraints ensuring the sections' stability. This problem requires careful handling of multi-modal solutions to avoid premature convergence. The schematic view of the CBD problem is illustrated in Fig. 17. The mathematical formulation of the CBD optimization problem is detailed in Eq. (28) presented below:

$$\text{Minimise} : f(x) = 0.6334(x_1 + x_2 + x_3 + x_4 + x_5) \quad (28)$$

$$\text{Subjected to} : g_1(x) = \frac{61}{{x_1}^3} + \frac{27}{{x_2}^3} + \frac{19}{{x_3}^3} + \frac{7}{{x_4}^3} + \frac{1}{{x_5}^3} - 1 \leq 0$$

$$g_2(x) = 0.01 \leq x_1 \leq 100$$

$$g_3(x) = 0.01 \leq x_2 \leq 100$$

$$g_4(x) = 0.01 \leq x_3 \leq 100$$

$$g_5(x) = 0.01 \leq x_4 \leq 100$$

$$g_6(x) = 0.01 \leq x_5 \leq 100$$

For the CBD problem, the comparative simulation results of the SA-EPO and other algorithms are detailed in Table 11. The proposed SA-EPO outperforms all competing algorithms in terms of stability and effectiveness in addressing the CBD problem. Additionally, the convergence curves depicted in Fig. 21-d highlights the superiority of SA-EPO, showcasing its faster convergence and consistent performance compared with other competing algorithms.

4.4.5. Pressure vessel design

The pressure vessel design problem, initially introduced by Kannan [88], aims to minimise the fabrication costs of a vessel capped with hemispherical heads. Fig. 18 provides a schematic representation of this pressure vessel. The challenge lies in the nonlinearity and strict constraints of the design variables, which include the thickness of the shell ($z_1 = T_s$), thickness of the head ($z_2 = T_h$), inner radius ($z_3 = R$) and length of the cylindrical section without considering the head ($z_4 = L$). $R$and$L$ are continuous variables, while $T_s$andT_h are discrete, being multiples of 0.0625 in.

The objective function and constraints of this problem are mathematically formulated in Eq. (29):

$$\text{Minimise} : f(z) = 0.6224 z_1 z_3 z_4 + 1.7781 z_2 {z_3}^2 + 3.1661 {z_1}^2 z_4 + 19.84 {z_1}^2 z_3 \quad (29)$$

$$\text{Subjected to} : g_1(z) = -z_1 + 0.0193 z_3 \leq 0$$

$$g_2(z) = -z_3 + 0.00954 z_3 \leq 0$$

$$g_3(z) = -\pi {z_3}^2 z_4 - \frac{4}{3}\pi {z_3}^2 + 1,296,000 \leq 0$$

$$g_4(z) = z_4 - 240 \leq 0$$

$$\text{where} : 1 \times 0.625 \leq z_1, z_2 \leq 99 \times 0.625, 10.0 \leq z_3, z_4 \leq 200.0$$

Table 11 compares the results obtained from SA-EPO with those from other competing algorithms such as EPO, SaBO, SdSCA, SDPSO, MSEFA, ESO, MS-DE, FAEPO, IEPO, ESA, CEPO, H-EPC, SHADE, JDE and JADE. The analysis shows that SA-EPO efficiently identifies the optimal design with minimised cost, outperforming all compared algorithms except one, where it ranks second-best. Additionally, the convergence curves in Fig. 21(e) demonstrate SA-EPO's rapid convergence and consistent performance, which distinctly surpasses that of its competitors in this engineering challenge.

4.4.6. Speed reducer design problem

The speed reducer design problem [89], recognised for its complexity, involves optimizing seven interrelated design variables to minimise the weight of a speed reducer while adhering to stringent mechanical and functional constraints [90]. This problem, illustrated in Fig. 19, includes limits on gear teeth bending stress, surface stress, shaft deflections and overall shaft stress. The seven design variables considered in the speed reducer design problem, denoted as z_1 through z_7, each representing critical geometric and structural parameters. These include the face width ($z_1 = b$), the module of teeth ($z_2 = m$), the number of teeth in the pinion ($z_3 = p$), the length of the first shaft between bearings ($z_4 = l_1$), the length of the second shaft between bearings ($z_5 = l_2$), the diameter of the first shaft ($z_6 = d_1$)*and* the diameter of the second shaft ($z_7 = d_2$). Notably, the third variable z_3, representing the number of teeth in the pinion (p), is an integer, which introduces an additional layer of complexity to the optimization process.

The goal of the speed reducer design problem is to minimise the following objective functions while satisfying several constraints that ensure the reducer's operational viability:

$$\text{Minimise} : f(z) = 0.7854 z_1 {z_2}^2 (3.3333 {z_3}^2 + 14.9334 z_3 - 43.0934) \quad (30)$$

$$-1.508z_1\left(z_6{}^2+\quad z_7{}^2\right)+7.4777\left(\quad z_6{}^2+\quad z_7{}^2\right)+0.7845\quad\left(\quad z_4z_6{}^2+\quad z_5z_7{}^2\right)$$

$$\text{Subjected to}: g_1(z)=\frac{27}{z_1z_2{}^2z_3}-1\leq\ 0$$

$$g_2(z)=\frac{397.5}{z_1z_2{}^2z_3{}^2}-1\leq\quad 0$$

$$g_3(z)=\frac{1.93z_4{}^3}{z_2z_6{}^4z_3}-1\leq\quad 0$$

$$g_4(z)=\frac{1.93z_5{}^3}{z_2z_7{}^4z_3}-1\leq\quad 0$$

$$g_5(z)=\frac{\left[\left(745\left(z_{4/z_2z_3}\right)\right)^2\quad+\quad 16.9\quad\times\quad 10^6\right]^{1/2}}{110z_6{}^3}-1\leq\quad 0$$

$$g_6(z)=\frac{\left[\left(745\left(z_{5/z_2z_3}\right)\right)^2\quad+\quad 157.5\quad\times\quad 10^6\right]^{1/2}}{85z_7{}^3}-1\leq\quad 0$$

$$g_7(z)=\frac{z_2z_3}{40}-1\leq\quad 0$$

$$g_8(z)=\frac{5z_2}{z_1}-1\leq\quad 0$$

$$g_9(z)=\frac{z_1}{12z_2}-1\leq\quad 0$$

$$g_{10}(z)=\frac{1.5z_2+1.9}{z_4}-1\leq\quad 0$$

$$g_{11}(z)=\frac{1.1z_2+1.9}{z_5}-1\leq\quad 0$$

where : $2.6\leq z_1\leq 3.6,\quad 0.7\leq z_2\leq 0.8,\quad 17\leq z_3\leq 28,\quad 7.3\leq z_4\leq 8.3,$

$7.3\leq z_5\leq 8.3,\quad 2.9\leq z_6\leq 3.9,\quad 5.0\leq z_7\leq 5.5$

Table 11 and Fig. 21(f) detail SA-EPO's performance relative to other algorithms like EPO, SaBO and SdSCA, demonstrating its superior ability to find the optimal design cost-effectively. The convergence curves in Fig. 21(f) particularly showcase the rapid convergence and robust performance of SA-EPO against competing algorithms, further affirming its effectiveness in tackling this multifaceted engineering challenge.

4.4.7. Rolling element bearing design problem

The rolling element bearing design problem focuses on maximising the dynamic load-carrying capacity, which involves complex multi-objective optimization. This problem is depicted schematically in Fig. 20 and comprises 10 decision variables, including the pitch diameter (D_m), ball diameter (D_b), number of balls (Z), inner (f_i) and outer (f_o) raceway curvature coefficients and others (i.e., $K_{D\min}$, $K_{D\max}$, ε, e *and* ζ). These variables interact in a complex optimization landscape that demands effective management of competing objectives.

The objective function of the rolling element bearing design problem is designed to maximise the dynamic capacity (C_d). Several constraints are also formulated in this design problem to ensure mechanical integrity and operational efficacy, encompassing aspects such a s allowable stress and geometric tolerance. Mathematically, the rolling element bearing design problem is expressed as

$$\text{Maximise}\quad C_d=\begin{cases} f_c\ \ Z^{2/3}D_b{}^{1.8}, if D\ \ \leq\ \ 25.4mm \\ 3.64f_cZ^{2/3}D_b{}^{1.4}, if D\ \ >\ \ 25.4mm \end{cases} \tag{31}$$

$$\text{Subjected to}: g_1(z)=\frac{\phi_0}{2\sin^{-1}\left(D_b/D_m\right)}-Z+1\leq\quad 0$$

$$g_2(z)=2D_b-K_{D\min}(D-d)\geq\quad 0$$

$$g_3(z)=K_{D\max}\left(D-d\right)-2D_b\geq\quad 0$$

$$g_4(z)=\zeta B_w-D_b\geq\quad 0$$

$$g_5(z)=D_m-0.5\left(D+d\right)\geq\quad 0$$

$$g_6(z)=\left(0.5+e\right)\left(D+d\right)-D_m\geq\quad 0$$

$$g_7(z)=0.5(D-D_m-D_b)-\varepsilon D_b\geq\quad 0$$

$$g_8(z)=f_i\geq\quad 0.515$$

$$g_9(z)=f_o\geq\quad 0.515$$

where : f_c

$$=37.91\left[1+\left[1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right]^{\frac{10}{3}}\right]^{-0.3}$$

$$\times\left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{\frac{1}{3}}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{1.41}$$

$$x=\left[\left\{\frac{D-d}{2}-3\left(\frac{T}{4}\right)\right\}^2+\left\{\frac{d}{2}-\frac{T}{4}-D_b\right\}^2-\left\{\frac{d}{2}+\frac{T}{4}\right\}^2\right]$$

$$y=2\left\{\frac{D-d}{2}-3\left(\frac{T}{4}\right)\right\}\times\left\{\frac{D}{2}-\frac{T}{4}-D_b\right\}$$

$$\phi_o=2\pi-\quad 2\cos^{-1}\left(\frac{x}{y}\right)$$

$$\gamma=\frac{D_b}{D_m},\quad f_i=\frac{r_i}{D_b},\quad f_o=\frac{r_o}{D_b},\quad T=D-d-2D_b$$

$$D=160,\quad d=90,\quad B_w=30,\quad r_i=\quad r_o=\quad 11.033$$

$$0.5(D-d)\leq D_m\leq 0.6(D-d),\quad 0.15(D-d)\leq D_b\leq 0.45(D-d),$$

$$4\leq Z\leq 50,\quad 0.515\leq f_i and f_o\quad\leq 0.6$$

$$0.4\leq K_{D\min}\leq 0.5,\quad 0.6\leq\quad K_{D\max}\leq 0.7,\quad 0.3\leq e\leq 0.4,$$

$$0.02\leq e\leq 0.1,\quad 0.6\quad\leq\zeta\leq 0.85.$$

To validate the SA-EPO algorithm's effectiveness, we compared it against several established optimization algorithms across various simulation runs. The performance data are tabulated in Table 11, and

Fig. 21(g) visually represents the convergence characteristics of these algorithms, highlighting the SA-EPO's superior performance in addressing this complex problem.

5. Discussion

The introduction of SA-EPO signifies a notable advancement in adaptive optimization algorithms. Empirical results demonstrate its effectiveness in navigating complex fitness landscapes, achieving an optimal balance between exploration and exploitation, and delivering competitive convergence rates. SA-EPO particularly excels in unimodal, hybrid and composition functions due to its dynamic parameter adaptation strategies that effectively mitigate local optima and accelerate convergence. For example, in unimodal functions, the proposed SA-EPO algorithm demonstrates enhanced fine-tuning capability of control parameters, contributing to its superior convergence speed and solution accuracy.

However, SA-EPO encounters challenges with multimodal functions characterised by numerous local optima, such as F9 from the CEC 2017 benchmark set. The computational load increases significantly with these functions due to the frequent parameter adjustments required in high-dimensional problems, often leading to diminishing returns as the complexity of problems increases. Furthermore, performance inconsistencies observed in specific hybrid functions such as F7 of CEC 2022 indicate that the adaptive mechanisms of SA-EPO may not always effectively handle peculiar problem characteristics, such as discontinuities or deceptive landscapes. These observations necessitate further research into tailoring the parameter adaptation strategies to better suit specific features of the problems. Despite these limitations, SA-EPO's robust performance across the diverse benchmark functions reflects its general applicability. Nonetheless, pinpointing the factors that contribute to its effectiveness, particularly its adeptness at balancing local and global search, remains crucial for future development in adaptive optimization algorithms.

Bridging this gap between theoretical effectiveness and practical utility, SA-EPO not only excels in computational experiments but also holds significant potential for complex optimization tasks across industries such as logistics, manufacturing and finance. It enables more efficient resource allocation and decision-making. Its ability to rapidly generate accurate solutions enhances operational efficiencies, optimizes supply chain logistics and improves scheduling and resource distribution. The inherent self-adaptive nature of SA-EPO also suits it for real-time applications, allowing businesses to adapt strategies dynamically in response to evolving operational conditions, thereby boosting agility and competitiveness in rapidly changing markets.

6. Conclusion and future work

This study introduces the SA-EPO, a significant advancement in adaptive optimization algorithms. By integrating new mechanisms that adaptively fine-tune algorithmic parameters during the search process, SA-EPO effectively navigates complex optimization landscapes. This dynamic adaptation achieves an optimal balance between exploring new potential solutions and exploiting already-found promising solutions, crucial for efficient optimization. Empirical evaluations on various benchmark functions, including CEC2017 and CEC2022, as well as real-world engineering challenges, demonstrate the superior performance of SA-EPO. It consistently outperforms other peer algorithms with better accuracy and faster convergence rates. The adaptability of SA-EPO makes it particularly effective across diverse problem types, including unimodal, multimodal, hybrid and composite functions. This versatility highlights its potential for broad applicability in diverse optimization scenarios. The success of SA-EPO is largely attributed to its innovative self-adaptive mechanism, which effectively adjusts its control parameters throughout the optimization process. This mechanism enables each SA-EPO solution to select the most suitable parameter adaptation scheme based on the success rate and historical performance of each scheme during previous learning periods. Consequently, each solution can learn from its own performance and adapt its behavior accordingly. This self-adjusting capability not only enhances the efficiency of the search process but also significantly improves the robustness of the algorithm against different types of optimization problems. The results highlight the superior performance of the SA-EPO algorithm over the EPO version. Improvements in the Friedman rank are notable, with increases of 47.9 % for CEC2017 and 52.4 % for CEC2022. Additionally, the Wilcoxon signed rank tests show a complete 100 % improvement, confirming the SA-EPO's effectiveness in efficiently navigating the complexities of the benchmarked optimization problems.

Future research should focus on refining the SA-EPO methodology to streamline its parameter adaptation mechanism, thereby strengthening robust performance. One approach is the development of hybrid strategies that dynamically adjust control parameters tailored to specific problem characteristics, enhancing the algorithm's adaptability across varied scenarios. Additionally, incorporating multi-population approaches that utilise both fitness value and solution diversity could significantly enhance exploration and exploitation capabilities of SA-EPO, especially in complex and high-dimensional optimization problems. Exploring the application of SA-EPO to more complex optimization challenges presents another fruitful direction. Potential applications include optimizing electric vehicle charging scheduling, conducting network architecture searches for deep learning models and enhancing feature selection methods. Moreover, adapting SA-EPO to dynamic optimization environments, such as real-time resource allocation and evolving supply chain management, would broaden its practical application, making it highly relevant for industries facing rapidly changing conditions. Lastly, integrating SA-EPO within ensemble or cooperative frameworks, where it could interact with multiple optimization algorithms, may significantly enhance its robustness and generalisability. This integration could lead to more resilient solutions that are well-suited to a broader range of problems, thereby expanding the scope and impacts of SA-EPO in both theoretical and applied optimization fields.

CRediT authorship contribution statement

Lim Wei Hong: Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis. **Mat Isa Nor Ashidi:** Writing – review & editing, Validation, Supervision, Project administration, Investigation, Conceptualization. **Khalid Othman Waleed:** Writing – original draft, Visualization, Software, Resources, Project administration, Methodology, Formal analysis, Data curation, Conceptualization.

Declaration of Competing Interest

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

Acknowledgement

The study was funded by the Malaysian Ministry of Higher Education through the Fundamental Research Grant Scheme (FRGS/1/2024/ICT02/UCSI/02/1).

References

[1] J. Straub, Gradient descent training expert system, Softw. Impacts 10 (2021) 100121, https://doi.org/10.1016/j.simpa.2021.100121.
[2] Y. Chalco-Cano, G.N. Silva, A. Rufián-Lizana, On the Newton method for solving fuzzy optimization problems, Fuzzy Sets Syst. 272 (2015) 60–69, https://doi.org/10.1016/j.fss.2015.02.001.

[3] W. Qian, Q. Ye, Y. Li, J. Huang, S. Dai, Relevance-based label distribution feature selection via convex optimization, Inf. Sci. (N. Y) 607 (2022) 322–345, https://doi.org/10.1016/j.ins.2022.05.094.

[4] A. Machmudah, et al., Cyclic path planning of hyper-redundant manipulator using whale optimization algorithm, Int. J. Adv. Comput. Sci. Appl. 12 (8) (2021), https://doi.org/10.14569/IJACSA.2021.0120879.

[5] A. Machmudah, et al., Design optimization of a gas turbine engine for marine applications: off-design performance and control system considerations, Entropy 24 (12) (2022) 1729, https://doi.org/10.3390/e24121729.

[6] A. Machmudah, E.A. Bakar, R. R, W.H. Nugroho, M.I. Solihin, A. Ghofur, Control system optimisation of biodiesel-based gas turbine for ship propulsion, IAES Int. J. Artif. Intell. (IJ-AI) 13 (2) (2024) 1992, https://doi.org/10.11591/ijai.v13.i2.pp1992-2002.

[7] Z. Danin, A. Sharma, M. Averbukh, A. Meher, Improved moth flame optimization approach for parameter estimation of induction motor, Energ. (Basel) 15 (23) (2022) 8834, https://doi.org/10.3390/en15238834.

[8] A. Sharma, et al., Improved moth flame optimization algorithm based on opposition-based learning and Lévy flight distribution for parameter estimation of solar module, Energy Rep. 8 (2022) 6576–6592, https://doi.org/10.1016/j.egyr.2022.05.011.

[9] C. Yuan, et al., Artemisinin optimization based on malaria therapy: algorithm and applications to medical image segmentation, Displays 84 (2024) 102740, https://doi.org/10.1016/j.displa.2024.102740.

[10] Yang and X.S., Nature-Inspired Metaheuristic Algorithms. 2008.

[11] P.A. Vikhar, Evolutionary algorithms: a critical review and its future prospects, 2016 Int. Conf. Glob. Trends Signal Process. Inf. Comput. Commun. (ICGTSPICC) IEEE (2016) 261–265, https://doi.org/10.1109/ICGTSPICC.2016.7955308.

[12] D. Whitley, A genetic algorithm tutorial, Stat. Comput. 4 (2) (1994), https://doi.org/10.1007/BF00175354.

[13] N.T. Hien, C.T. Tran, X.H. Nguyen, S. Kim, V.D. Phai, N.B. Thuy, N.V. Manh, Genetic programming for storm surge forecasting, Ocean Eng. 215 (2020), https://doi.org/10.1016/j.oceaneng.2020.107812.

[14] F. Gorunescu, S. Belciug, Evolutionary strategy to develop learning-based decision systems. Application to breast cancer and liver fibrosis stadialization, J. Biomed. Inf. 49 (2014) 112–118, https://doi.org/10.1016/j.jbi.2014.02.001.

[15] A.K. Qin and P.N. Suganthan, Self-adaptive differential evolution algorithm for numerical optimization, in 2005 IEEE Congress on Evolutionary Computation, IEEE, pp. 1785–1791. https://doi.org/10.1109/CEC.2005.1554904.

[16] E.H. Houssein, D. Oliva, N.A. Samee, N.F. Mahmoud, M.M. Emam, Liver cancer algorithm: a novel bio-inspired optimizer, Comput. Biol. Med 165 (2023) 107389, https://doi.org/10.1016/j.compbiomed.2023.107389.

[17] Y. Zhang, A. Chi, S. Mirjalili, Enhanced Jaya algorithm: a simple but efficient optimization method for constrained engineering design problems, Knowl. Based Syst. 233 (2021) 107555, https://doi.org/10.1016/J.KNOSYS.2021.107555.

[18] L.M. Zhang, C. Dahlmann, Y. Zhang, Human-Inspired Algorithms for continuous function optimization, 2009 IEEE Int. Conf. Intell. Comput. Intell. Syst. IEEE (2009) 318–321, https://doi.org/10.1109/ICICISYS.2009.5357838.

[19] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems, Comput. -Aided Des. 43 (3) (2011) 303–315, https://doi.org/10.1016/J.CAD.2010.12.015.

[20] M. Kumar, A.J. Kulkarni, S.C. Satapathy, Socio evolution & learning optimization algorithm: a socio-inspired optimization methodology, Future Gener. Comput. Syst. 81 (2018) 252–272, https://doi.org/10.1016/J.FUTURE.2017.10.052.

[21] M. Li, H. Zhao, X. Weng, T. Han, Cognitive behavior optimization algorithm for solving optimization problems, Appl. Soft Comput. 39 (2016) 199–222, https://doi.org/10.1016/J.ASOC.2015.11.015.

[22] S. Mirjalili, SCA: a sine cosine algorithm for solving optimization problems, Knowl. Based Syst. 96 (2016) 120–133, https://doi.org/10.1016/J.KNOSYS.2015.12.022.

[23] P. Pachung, J.C. Bansal, An improved tangent search algorithm, MethodsX 9 (2022) 101839, https://doi.org/10.1016/j.mex.2022.101839.

[24] D.B. Fogel, An introduction to simulated evolutionary optimization, IEEE Trans. Neural Netw. 5 (1) (1994) 3–14, https://doi.org/10.1109/72.265956.

[25] B. Xing and W.-J. Gao, Electromagnetism-like Mechanism Algorithm, 2014, pp. 347–354. https://doi.org/10.1007/978-3-319-03404-1_21.

[26] B. Narottama, D.K. Hendraningrat, S.Y. Shin, Quantum-inspired evolutionary algorithms for NOMA user pairing, ICT Express 8 (1) (2022) 11–17, https://doi.org/10.1016/j.icte.2022.01.014.

[27] C. Yuan, D. Zhao, A.A. Heidari, L. Liu, Y. Chen, H. Chen, Polar lights optimizer: algorithm and applications in image segmentation and feature selection, Neurocomputing 607 (2024) 128427, https://doi.org/10.1016/j.neucom.2024.128427.

[28] H. Su, et al., RIME: a physics-based optimization, Neurocomputing 532 (2023) 183–214, https://doi.org/10.1016/j.neucom.2023.02.010.

[29] J. Kennedy, R. Eberhart, and bls gov, 'Particle Swarm Optimization, 1995.

[30] Y. Shi, C.M. Pun, H. Hu, H. Gao, An improved artificial bee colony and its application, Knowl. Based Syst. 107 (2016) 14–31, https://doi.org/10.1016/J.KNOSYS.2016.05.052.

[31] K. Ye, C. Zhang, J. Ning, X. Liu, Ant-colony algorithm with a strengthened negative-feedback mechanism for constraint-satisfaction problems, Inf. Sci. (N. Y) 406–407 (2017) 29–41, https://doi.org/10.1016/J.INS.2017.04.016.

[32] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016) 51–67, https://doi.org/10.1016/j.advengsoft.2016.01.008.

[33] A.A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: algorithm and applications, Future Gener. Comput. Syst. 97 (2019) 849–872, https://doi.org/10.1016/j.future.2019.02.028.

[34] F. MiarNaeimi, G. Azizyan, M. Rashki, Horse herd optimization algorithm: a nature-inspired algorithm for high-dimensional optimization problems, Knowl. Based Syst. 213 (2021) 106711, https://doi.org/10.1016/j.knosys.2020.106711.

[35] A. Qi, D. Zhao, A.A. Heidari, L. Liu, Y. Chen, H. Chen, FATA: an efficient optimization method based on geophysics, Neurocomputing 607 (2024) 128289, https://doi.org/10.1016/j.neucom.2024.128289.

[36] G. Dhiman, V. Kumar, 'Emperor penguin optimizer: a bio-inspired algorithm for engineering problems', Knowl. Based Syst. 159 (2018) 20–50, https://doi.org/10.1016/j.knosys.2018.06.001.

[37] G. Dhiman, ESA: a hybrid bio-inspired metaheuristic optimization approach for engineering problems, Eng. Comput. 37 (1) (2021) 323–353, https://doi.org/10.1007/s00366-019-00826-w.

[38] N.R. Babu, T. Chiranjeevi, R. Devarapalli, S.K. Bhagat, Optimal location of FACTs devices in LFC studies considering the application of RT-Lab studies and emperor penguin optimization algorithm, J. Eng. Res. (2023) 100060, https://doi.org/10.1016/j.jer.2023.100060.

[39] S. Min, Z. Tang, B. Daneshvar Rouyendegh, Inspired-based optimisation algorithm for solving energy-consuming reduction of chiller loading, Int. J. Ambient Energy (2020) 1–11, https://doi.org/10.1080/01430750.2020.1730954.

[40] H. Shaheen, K. Ravikumar, N. Lakshmipathi Anantha, A. Uma Shankar Kumar, N. Jayapandian, S. Kirubakaran, An efficient classification of cirrhosis liver disease using hybrid convolutional neural network-capsule network, Biomed. Signal Process Control 80 (2023) 104152, https://doi.org/10.1016/j.bspc.2022.104152.

[41] H. Jia, K. Sun, W. Song, X. Peng, C. Lang, Y. Li, Multi-strategy emperor penguin optimizer for RGB histogram-based color satellite image segmentation using masi entropy, IEEE Access 7 (2019) 134448–134474, https://doi.org/10.1109/ACCESS.2019.2942064.

[42] K. Cheena, T. Amgoth, G. Shankar, Emperor penguin optimized self-healing strategy for WSN based smart grids, 2020., and .

[43] A. Md, K.Z. Kader, Zamli, B.Y. Alkazemi, An experimental study of a fuzzy adaptive emperor penguin optimizer for global optimization problem, IEEE Access 10 (2022) 116344–116374, https://doi.org/10.1109/ACCESS.2022.3213805.

[44] Z. Xing, An improved emperor penguin optimization based multilevel thresholding for color image segmentation, Knowl. Based Syst. 194 (2020) 105570, https://doi.org/10.1016/J.KNOSYS.2020.105570.

[45] S. R, A. H, Improved EPOA clustering protocol for lifetime longevity in wireless sensor network, Sens. Int. 3 (2022) 100199, https://doi.org/10.1016/j.sintl.2022.100199.

[46] M.M. Rosso, R. Cucuzza, A. Aloisio, G.C. Marano, Enhanced multi-strategy particle swarm optimization for constrained problems with an evolutionary-strategies-based unfeasible local search operator, Appl. Sci. 12 (5) (2022) 2285, https://doi.org/10.3390/app12052285.

[47] S.M. Elsayed, R.A. Sarker, D.L. Essam, Adaptive Configuration of evolutionary algorithms for constrained optimization, Appl. Math. Comput. 222 (2013) 680–711, https://doi.org/10.1016/j.amc.2013.07.068.

[48] V. Stanovov, S. Akhmedova, E. Semenkin, The automatic design of parameter adaptation techniques for differential evolution with genetic programming, Knowl. Based Syst. 239 (2022) 108070, https://doi.org/10.1016/j.knosys.2021.108070.

[49] Y. Zhang, X. Song, A multi-strategy adaptive comprehensive learning PSO algorithm and its application, Entropy 24 (7) (2022) 890, https://doi.org/10.3390/e24070890.

[50] A. Waters, F. Blanchette, A.D. Kim, Modeling huddling penguins, PLoS One 7 (11) (2012) e50277, https://doi.org/10.1371/journal.pone.0050277.

[51] M. Kalra, V. Kumar, M. Kaur, S. Ahmed Idris, Ş. Öztürk, H. Alshazly, A novel binary emperor penguin optimizer for feature selection tasks, Comput. Mater. Contin. 70 (3) (2022) 6239–6255, https://doi.org/10.32604/cmc.2022.020682.

[52] O.W. Khalid, N.A.M. Isa, H.A. Mat Sakim, Emperor penguin optimizer: a comprehensive review based on state-of-the-art meta-heuristic algorithms, Alex. Eng. J. 63 (2023) 487–526, https://doi.org/10.1016/j.aej.2022.08.013.

[53] F. Tang, J. Li, N. Zafetti, Optimization of residential building envelopes using an improved Emperor Penguin Optimizer, Eng. Comput. (2020), https://doi.org/10.1007/s00366-020-01112-w.

[54] H. Wahdan, H. Abdelslam, T. Abou-El-Enien, S. Kassem, Two-modified emperor penguins colony optimization algorithms, Rev. D. 'Intell. Artif. 34 (2) (2020) 151–160, https://doi.org/10.18280/ria.340205.

[55] S.K. Baliarsingh, S. Vipsita, Chaotic emperor penguin optimised extreme learning machine for microarray cancer classification, IET Syst. Biol. 14 (2) (2020) 85–95, https://doi.org/10.1049/iet-syb.2019.0028.

[56] S. Harifi, J. Mohammadzadeh, M. Khalilian, S. Ebrahimnejad, Hybrid-EPC: an Emperor Penguins Colony algorithm with crossover and mutation operators and its application in community detection, Prog. Artif. Intell. 10 (2) (Jun. 2021) 181–193, https://doi.org/10.1007/s13748-021-00231-9.

[57] M.A. Angel, T. Jaya, An enhanced emperor penguin optimization algorithm for secure energy efficient load balancing in wireless sensor networks, Wirel. Pers. Commun. 125 (3) (2022) 2101–2127, https://doi.org/10.1007/s11277-022-09647-5.

[58] R.F. Mansour, H. Alhumyani, S.A. Khalek, R.A. Saeed, D. Gupta, Design of cultural emperor penguin optimizer for energy-efficient resource scheduling in green cloud computing environment, Clust. Comput. 26 (1) (Feb. 2023) 575–586, https://doi.org/10.1007/s10586-022-03608-0.

[59] S.K. Baliarsingh, W. Ding, S. Vipsita, S. Bakshi, A memetic algorithm using emperor penguin and social engineering optimization for medical data classification, Appl. Soft Comput. J. 85 (2019), https://doi.org/10.1016/j.asoc.2019.105773.

[60] K. Bagirathan, A. Palanisamy, Opportunistic routing protocol based EPO–BES in MANET for optimal path selection, Wirel. Pers. Commun. 123 (1) (2022) 473–494, https://doi.org/10.1007/s11277-021-09140-5.

[61] B. Gupta, K.K. Gola, M. Dhingra, 'HEPSO: an efficient sensor node redeployment strategy based on hybrid optimization algorithm in UWASN, Wirel. Netw. 27 (4) (2021) 2365–2381, https://doi.org/10.1007/s11276-021-02584-4.
[62] H. Peng, et al., Multi-strategy firefly algorithm with selective ensemble for complex engineering optimization problems, Appl. Soft Comput. 120 (2022) 108634, https://doi.org/10.1016/j.asoc.2022.108634.
[63] A.K. Das, S. Sahoo, D.K. Pratihar, An improved design of knee orthosis using self-adaptive bonobo optimizer (SaBO), J. Intell. Robot Syst. 107 (1) (2023) 8, https://doi.org/10.1007/s10846-022-01802-1.
[64] R. Akay, M.Y. Yildirim, Multi-strategy and self-adaptive differential sine–cosine algorithm for multi-robot path planning, Expert Syst. Appl. 232 (2023) 120849, https://doi.org/10.1016/j.eswa.2023.120849.
[65] Z. Liu, T. Nishi, Strategy dynamics particle swarm optimizer, Inf. Sci. (N. Y) 582 (2022) 665–703, https://doi.org/10.1016/j.ins.2021.10.028.
[66] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evolut. Comput. 10 (3) (2006) 281–295, https://doi.org/10.1109/TEVC.2005.857610.
[67] B.Y. Qu, P.N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, IEEE Trans. Evolut. Comput. 17 (3) (Jun. 2013) 387–402, https://doi.org/10.1109/TEVC.2012.2203138.
[68] K.E. Parsopoulos, M.N. Vrahatis, UPSO: a unified particle swarm optimization scheme, 1st ed. Int. Conf. Comput. Methods Sci. Eng. (ICCMSE 2004) (2004) 868–873.
[69] Y. Liu, Z. Qin, Z. Shi, J. Lu, Center particle swarm optimization, Neurocomputing 70 (4–6) (2007) 672–679, https://doi.org/10.1016/j.neucom.2006.10.002.
[70] A.J. Yousif, G. Jabbar, A. Subhi, Design of linear phase high pass fir filter using weight improved particle swarm optimization, Int. J. Adv. Comput. Sci. Appl. 9 (9) (2018), https://doi.org/10.14569/IJACSA.2018.090936.
[71] A.J. Yousif, M.H. Al-Jammas, A lightweight visual understanding system for enhanced assistance to the visually impaired using an embedded platform, Diyala J. Eng. Sci. (2024) 146–162, https://doi.org/10.24237/djes.2024.17310.
[72] L. Yao, P. Yuan, C.-Y. Tsai, T. Zhang, Y. Lu, S. Ding, ESO: an enhanced snake optimizer for real-world engineering problems, Expert Syst. Appl. 230 (2023) 120594, https://doi.org/10.1016/j.eswa.2023.120594.
[73] H. Peng, Y. Han, C. Deng, J. Wang, Z. Wu, Multi-strategy co-evolutionary differential evolution for mixed-variable optimization, Knowl. Based Syst. 229 (2021) 107366, https://doi.org/10.1016/j.knosys.2021.107366.
[74] X. Wang, Y. Wang, K.-C. Wong, X. Li, A self-adaptive weighted differential evolution approach for large-scale feature selection, Knowl. Based Syst. 235 (2022) 107633, https://doi.org/10.1016/j.knosys.2021.107633.
[75] B. She, A. Fournier, M. Yao, Y. Wang, G. Hu, A self-adaptive and gradient-based cuckoo search algorithm for global optimization, Appl. Soft Comput. 122 (2022) 108774, https://doi.org/10.1016/j.asoc.2022.108774.
[76] A.J. Yousif, M.H. Al-Jammas, Real-time Arabic video captioning using CNN and transformer networks based on parallel implementation, Diyala J. Eng. Sci. (2024) 84–93, https://doi.org/10.24237/djes.2024.17108.
[77] Y. Shen, C. Zhang, F. Soleimanian Gharehchopogh, S. Mirjalili, An improved whale optimization algorithm based on multi-population evolution for global optimization and engineering design problems, Expert Syst. Appl. 215 (2023) 119269, https://doi.org/10.1016/j.eswa.2022.119269.
[78] R. Tanabe, A. Fukunaga, Success-history based parameter adaptation for differential evolution, 2013 IEEE Congr. Evolut. Comput. IEEE (2013) 71–78, https://doi.org/10.1109/CEC.2013.6557555.
[79] J. Brest, A. Zamuda, B. Boskovic, M.S. Maucec, V. Zumer, Dynamic optimization using self-adaptive differential evolution, 2009 IEEE Congr. Evolut. Comput. IEEE (2009) 415–422, https://doi.org/10.1109/CEC.2009.4982976.
[80] Jingqiao Zhang, A.C. Sanderson, JADE: adaptive differential evolution with optional external archive, IEEE Trans. Evolut. Comput. 13 (5) (2009) 945–958, https://doi.org/10.1109/TEVC.2009.2014613.
[81] L. Zheng, Y. Wen, A multi-strategy differential evolution algorithm with adaptive similarity selection rule, Symmetry (Basel) 15 (9) (2023) 1697, https://doi.org/10.3390/sym15091697.
[82] Y. Xue, S. Zhong, Y. Zhuang, B. Xu, An ensemble algorithm with self-adaptive learning techniques for high-dimensional numerical optimization, Appl. Math. Comput. 231 (2014) 329–346, https://doi.org/10.1016/j.amc.2013.12.130.
[83] Noor H. Awad, MostafaZ. Ali, J.J. Liang, B.Y. Qu, PonnuthuraiN. Suganthan, Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization, Nanyang Technol. Univ. Jordan Univ. Sci. Technol. Zhengzhou Univ. Tech. Rep. (2016).
[84] Jacob Christian, Illustrating Evolutionary Computation with Mathematica, Elsevier, 2001, https://doi.org/10.1016/B978-1-55860-637-1.X5013-9.
[85] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (1) (2011) 3–18, https://doi.org/10.1016/j.swevo.2011.02.002.
[86] C.A. Coello Coello, Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art, Comput. Methods Appl. Mech. Eng. 191 (11–12) (2002) 1245–1287, https://doi.org/10.1016/S0045-7825(01)00323-1.
[87] S. Gupta, K. Deep, A hybrid self-adaptive sine cosine algorithm with opposition based learning, Expert Syst. Appl. 119 (Apr. 2019) 210–230, https://doi.org/10.1016/j.eswa.2018.10.050.
[88] B.K. Kannan, S.N. Kramer, An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, J. Mech. Des. 116 (2) (1994) 405–411, https://doi.org/10.1115/1.2919393.
[89] A.H. Gandomi and X.-S. Yang, Benchmark Problems in Structural Optimization, 2011, pp. 259–281. https://doi.org/10.1007/978-3-642-20859-1_12.
[90] E. Mezura-Montes and C.A.C. Coello, Useful Infeasible Solutions in Engineering Optimization with Evolutionary Algorithms, 2005, pp. 652–662. https://doi.org/10.1007/11579427_66.