Applied Intelligence (2025) 55:777
https://doi.org/10.1007/s10489-025-06674-0

Pulse-strategy collective learning swarm optimizer for large-scale global optimization

Xiaoyu Liu[1] · Qingke Zhang[1] · Shuzhao Pang[1] · Jiajun Sun[2] · Huaxiang Zhang[1]

Accepted: 25 May 2025 / Published online: 19 June 2025

Abstract
Social Learning Particle Swarm Optimization (SLPSO) is an advanced variant of the PSO algorithm, designed to enhance optimization performance in Large-Scale Global Optimization (LSGO) problems. However, SLPSO encounters significant challenges, particularly in maintaining a balanced trade-off between exploration and exploitation, which limits its effectiveness in complex optimization tasks. In response to these limitations, this paper proposes the Pulse-based Collective Learning Swarm Optimizer (PCLSO). The PCLSO introduces two key innovations: collective learning, which replaces the traditional average position updates in SLPSO, thereby enhancing exploration by leveraging knowledge from multiple high-quality particles; and a pulse-strategy, which addresses convergence stagnation by dynamically perturbing the global best solution to improve exploitation capabilities. Extensive experimental evaluations on the CEC'2010 and CEC'2013 benchmark suites demonstrate that PCLSO significantly outperforms SLPSO, as well as several advanced and classical PSO and DE variants, and the latest cooperative coevolutionary (CC)-based algorithms. Furthermore, to validate its practical applicability, PCLSO is applied to the biological Multiple Sequence Alignment (MSA) problem using Hidden Markov Models (HMM), where it exhibits superior performance compared to other state-of-the-art metaheuristic algorithms. In summary, the study presents a robust optimization tool that advances the field of LSGO and shows promise for addressing complex real-world optimization challenges. The source code of the PCLSO algorithm is publicly available at https://github.com/tsingke/PCLSO.

Keywords Large-Scale Global Optimization (LSGO) · Pulse-strategy · Collective learning · Metaheuristic algorithms · Multi Sequence Alignment (MSA)

1 Introduction

With the advancement of intelligence in social development, numerous optimization problems have emerged in various fields [1–3]. These problems require decision variables to be located within a given search space for the purpose of maximizing or minimizing the objective function value [4, 5]. However, real-world applications often face challenges in establishing precise mathematical models for optimization problems due to their complex characteristics such as discontinuity, non-convexity, multimodality, and non-differentiability, which pose challenges to traditional optimization methods [6–8]. Fortunately, metaheuristic algorithms (MAs), such as Evolutionary Computation (EC) and Swarm Intelligence (SI), can better solve these problems [9]. Numerous studies have shown that metaheuristic algorithms, especially evolutionary algorithms, not only perform well in deep learning compared to classical algorithms, but also have tremendous potential in solving LSGO problems. These MAs are not dependent on the precise mathematical models or gradient information of the problems, and have achieved remarkable results in solving complex black-box optimization problems [10, 11].

Currently, the rapid growth in problem dimensions raises significant challenges to the scalability of evolutionary algorithms [12–14]. When dealing with Large-Scale Global Optimization (LSGO) problems, traditional evolutionary algorithms are prone to premature convergence to local optima, leading to a drastic decline in performance [15].

✉ Qingke Zhang
tsingke@sdnu.edu.cn

[1] School of Information Science and Engineering, Shandong Normal University, Jinan 250358, China

[2] School of Science and Engineering, The Chinese University of Hong Kong (Shenzhen), Shenzhen 518172, China

A variety of algorithms have been introduced to deal with LSGO problems [16]. Overall, metaheuristic algorithms designed to solve LSGO problems can generally be classified within two groups. The first group includes Cooperative Coevolutionary (CC) metaheuristic methods, and the second group includes metaheuristic methods with innovative and efficient update strategies (non-CC), as shown in Fig. 1. In the existing literature, efforts have been made to extend existing MAs to deal with LSGO problems [17–19]. Table 1 illustrates some of the algorithms that were proposed over the last few years to solve LSGO problems.

The Particle Swarm Optimization (PSO), initially proposed at 1995, is based on the observed behavior of flocks of birds [37]. It has gained great popularity due to its advantages, including rapid convergence and simple deployment. This method has already been successfully implemented in various practical applications. However, despite its advantages, PSO also has limitations. It often suffers from poor global search capabilities and a tendency to converge prematurely on local optimal solutions. To address these shortcomings, numerous researchers have attempted to improve PSO by introducing various strategies such as crossover, mutation, and learning strategies, leading to the emergence of a variety of PSO variants [38, 39].

The Particle Swarm Optimization (PSO) algorithm, while notable for its straightforward implementation, exhibits several inherent limitations. These include the characteristic oscillation effect and inefficient search patterns (sometimes described as"two steps forward, one step back") [40]. More significantly, the method tends to experience premature convergence due to population diversity depletion, often resulting in suboptimal solutions as particles become trapped in local optima. Furthermore, maintaining an effective equilibrium between global exploration and local exploitation remains a persistent challenge in standard PSO implementations [41].

According to theories in intelligent optimization, the balance between exploration and exploitation is crucial for the success of an optimization algorithm. The question of how to achieve exploration and exploitation of solutions in metaheuristic algorithms remains an open subject [42]. Empirical evidence has shown that the exploration-exploitation balance is closely related to the convergence rate of a search method. While exploitation strategies can speed up convergence to the global optimum, they increase the likelihood of becoming trapped in local optima. Conversely, exploration strategies increase the chances of discovering regions with a higher likelihood of containing the global optimum within the search space, but at the cost of slower convergence.

Among the many LSGO algorithms, one notable algorithm is known as SLPSO [30]. In this algorithm, all particles (besides the best particle) update themselves through learning from each particle in the current population with better performance, thereby increasing the effectiveness of SLPSO in solving LSGO problems by maintaining population diversity. Despite the significant enhancements achieved by SLPSO in optimizing PSO, there are still shortcomings like inadequate balancing between explore and exploit, and lack of effective search. However, large-scale optimization not only requires the algorithm to maintain diversity to search for the global optimal solution in the large search space, but also requires to have fast convergence to refine the solution accuracy. Recent studies demonstrate that hybrid approaches incorporating local search strategies can significantly improve convergence rates. Zhao et al. [43] successfully integrated quasi-Newton methods with PSO, and Molina et

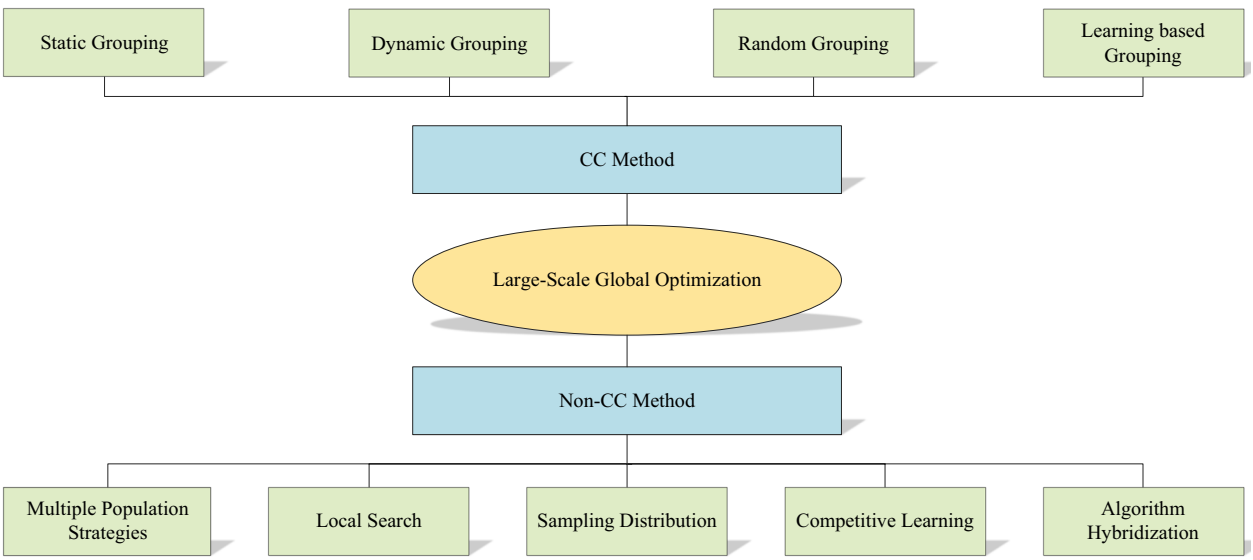


Fig. 1 Taxonomy of LSGO algorithm

Table 1 Summary of optimization algorithms for solving LSGO

Type	Algorithms	Dimensions	Proposed time	References	Real-world applications
CC methods	MLCC	100, 500, 1000	2008	[20]	–
	CCVIL	1000	2010	[21]	–
	DECC-D	100, 500, 1000	2010	[22]	–
	CCPSO2	1000	2012	[23]	–
	DECC-DG	1000	2014	[24]	–
	CCOS	1000	2018	[25]	–
	VVGP-RAC	–	2020	[26]	On-line resource allocation
	MDG	1000	2022	[27]	–
Non-CC methods	CDE	200, 500, 1000	2015	[28]	–
	CSO	100, 500, 1000, 2000	2015	[29]	–
	SLPSO	100, 500, 1000	2015	[30]	–
	MWOA	100, 300, 500, 1000	2018	[31]	–
	DGLDPSO	1000	2019	[1]	Cloud workflow scheduling
	EBA	100, 500, 1000	2019	[32]	–
	APSO_DEE	1000	2021	[33]	–
	EAPSO	30, 50, 100	2023	[34]	Engineering design problems
	TPCSO	1000	2023	[35]	–
	DCSO	100, 500, 1000	2016	[36]	–

al. [44] developed iterative frameworks employing multiple local search operators specifically for LSGO problems.

Inspired by the above observations, a pulse-strategy collective learning swarm optimizer (PCLSO) is introduced in this study. Firstly, collective learning, inspired by the concept of swarm intelligence in biological populations, emphasizing the sharing of knowledge and experience among multiple individuals to enhance problem-solving capabilities, is combined in the proposed algorithm. This approach encourages cooperation and collaboration between individuals in an algorithm, thereby increasing diversity and promoting global exploration of the search space. Secondly, the concept of pulse-strategy represents an innovative way of thinking within optimization algorithms. By introducing pulse-strategies at appropriate moments, algorithms can effectively solve search stagnation and stimulate global search. The combination of these two concepts represents an innovative approach to solving complex optimisation problems. The following sections describe the specific applications of collective learning and pulse strategies within the PCLSO framework, highlighting their significant contributions to algorithm performance.

The experimental results have proven that the incorporation of collective learning and pulse-strategy can significantly improve the performance of SLPSO in applying it to LSGO problems. In addition, the effectiveness using the algorithm in solving actual LSGO problems through HMM-based multi-sequence alignment problems is verified.

The rest of this paper is organized as follows: Section 2 introduces the overview of the PSO and its variants for LSGO problems. Section 3 describes about proposed PCLSO algorithm, including design inspiration and mathematical models. Section 4 provides the results analysis and discussion. The settings of the benchmark function and the test algorithm used are described. Next, extensive statistical experiment results and analysis of PCLSO and comparative algorithms are provided. Section 5 extends PCLSO and advanced MAs to HMM-based MSA problems. Section 6 is a summary of conclusions and further work.

2 Related work

2.1 PSO

Particle Swarm Optimization (PSO) is a widely recognized and utilized metaheuristic optimization algorithm, proposed by Kennedy and Eberhart in 1995 and it is inspired by the social behavior of birds flocking and fish schooling. PSO belongs to the class of swarm intelligence techniques and operates through the collaborative interaction of a population of candidate solutions, referred to as particles. PSO has demonstrated effectiveness in feature selection [45], neural network training [46], and diverse real-world applications. However, as noted in [47], standard PSO may suffer from premature convergence and suboptimal global exploration capabilities. **Algorithm** 1 illustrates the pseudocode of PSO.

Algorithm 1 PSO

Input: $MaxFEs$, N
Output: x_{gbest} and $gbestfitness$
Initialization the particle's population $n \leftarrow 1, 2, ..., N$ and calculate all $fitness$;
$FEs = FEs + popsize$;
while $FEs \leqslant MaxFEs$ **do**
  for $n \leftarrow 1$ *to* N **do**
    Update the position by using (2) ;
    Boundary control;
    Calculate the $fitness(n)$;
    $FEs = FEs + 1$;

Let $\vec{v}_n = (v_{n,1}, v_{n,2}, ..., v_{n,D})$ and $\vec{x}_n = (x_{n,1}, x_{n,2}, ..., x_{n,N})$ represent the vector of velocity and the vector of position of the *n* th particle within the search space of *D* dimensions in PSO, respectively. The evolution formula for the *n*th particle is given by (1) and (2).

$$\begin{aligned}\vec{v}_n(t+1) =& w \times \vec{v}_n(t) + c_1 \times r_1 \times (\vec{x}_{pbestn} - \vec{x}_n(t)) \\ &+ c_2 \times r_2 \times (\vec{x}_{gbest} - \vec{x}_n(t))\end{aligned} \tag{1}$$

$$\vec{x}_n(t+1) = \vec{x}_n(t) + \vec{v}_n(t+1) \tag{2}$$

where r_1 and r_2 represent two random numbers distributed uniformly in [0, 1], *t* is the present value of the iterations, and *w* is inertia weight. *n* ranges from 1 to *N*, where *N* is the size of the population. c_1 and c_2 represent the cognitive and social components respectively. By incorporating both information about its own optimal position ($\vec{x}_{pbestn}$) and the global optimal position ($\vec{x}_{gbest}$), the particles quickly converge to the best position. The simplicity of the calculations in (1) and (2) makes PSO easy to implement. However, the shared best experience of the particles can lead to clustering around local optima, making escape from local optima difficult and leading to loss of diversity.

2.2 SLPSO

SLPSO has demonstrated strong performance in addressing LSGO problems while retaining the benefits of PSO. Within SLPSO, for every particle besides the optimal one, it learns some information from every particle that outperforms its own. The mechanism is commonly referred by the name of social learning mechanism. Unlike standard PSO, where particles typically learn from gbest ($\vec{x}_{gbest}$) and pbest ($\vec{x}_{pbestn}$) only, SLPSO enables particles capable of learning various information obtained from other particles. As a result, SLPSO promotes greater population diversity because not all particles are overly influenced by $\vec{x}_{gbest}$. This helps prevent premature convergence to local optima.

During each generation of SLPSO, each particle within a given swarm is initially sorted by fitness, ranging from poorest to the global optimal. Following this sorting, every particle (excluding the $\vec{x}_{gbest}$) updates each of its dimensions through a process of learning by superior particles. When a particular particle has multiple particles that are superior to it, the particle chooses one randomly to serve as a reference for the updating. It's important to mention that a particle can utilize different superior particles to lead updates in various dimensions. Simplistically, a single particle has the flexibility to study different superior particles, thereby enhancing population diversity.

Supposing the particle *n* chooses the particle *win* for learning in the *d*th dimension, the equation for the updates is given by (3) and (4).

$$\begin{aligned}v_{n,d}(t+1) \quad =& r_1 \times v_{n,d}(t) + r_2 \times (x_{win,d} - x_{n,d}) \\ &+ c \times r_3 \times (x_{mean,j} - x_{n,d})\end{aligned} \tag{3}$$

$$x_{n,d}(t+1) = \begin{cases} x_{n,d}(t) + v_{n,d}(t+1), if\ rand \leqslant PL_n \\ x_{n,d}(t), otherwise \end{cases} \tag{4}$$

where *t* is number of the generation, *x* and *v* denote the position and velocity vectors, respectively. The values of r_1, r_2, and r_3 are generated as uniformly distributed numbers within the interval [0, 1]. Parameter *c* is referred to as the social influence factor, while $x_{mean,d}$ is the mean of the *d*th dimension within the vector of positions for all particles. The parameter PL_n, known as the learning probability, is employed to determine whether a particle requires an update or not, and is associated with its rank. The probability of learning is specific to each particle, particles with better ranks have less probability of learning. As the particles have been arranged from poorest to global optimal, the calculation of PL_n follows (5).

$$PL_n = \left(1 - \frac{n-1}{N}\right)^{\log\left(\frac{D}{M} \times \mu\right)} \tag{5}$$

where *n* denotes the particle's rank within population, sorted in ascending order. *N* represents the parameter size of the population, while μ and *M* are 0.5 and 100, respectively, according to the SLPSO's design. The variable *D* signifies the dimensionality of the problem. When a particle's fitness value is lower, its rank *n* will be higher, resulting in a larger learning probability PL_n. Consequently, poorly performing particles have a greater chance of updating their positions to seek improved solutions. Furthermore, the value of PL_n is influenced by the problem's dimensionality *D*. As *D* increases, PL_n decreases, serving to preserve the diversity of the population and prevent premature convergence in LSGO problems.

Additionally to incorporating the social learning mechanism, the algorithm dynamically adjusts specific parameters in response given the dimensionality of the problem. This adaptive approach aims to reduce the sensitivity of parameters such as the size of the population (*N*) and the factor of social influence (*c*). The values for *N* and *c* are determined as follows:

$$N = M + \frac{D}{10} \tag{6}$$

$$c = \beta \times \frac{D}{M} \tag{7}$$

where β is set to 0.01, following the recommendation from the original SLPSO. This choice is made to ensure that the value of ϵ remains relatively small, thereby preventing premature convergence. Furthermore, both N and C are influenced by the problem's dimensionality D. As the value of D increases, both N and c also increase, enhancing population diversity, particularly for LSGO problems. The pseudocode of SLPSO is shown in **Algorithm** 2.

Algorithm 2 SLPSO

Input: $MaxFEs$, D
Output: x_{gbest} and $gbestfitness$
Calculate the N;
Initialization the particle's population $n \leftarrow 1, 2, ..., N$ and calculate all $fitness$;
$FEs = FEs + popsize$;
while $FEs \leqslant MaxFEs$ **do**
  Sort particles with $fitness$;
  for $n \leftarrow 1$ *to* N **do**
    if $rand \leqslant PL_n$ **then**
      for $d \leftarrow 1$ *to* D **do**
        $\vec{x}_{win,d} = \vec{x}_{randn(n-1),d}$;
      Update the position by using (3) ;
      Boundary control;
      Calculate the $fitness(n)$;
      $FEs = FEs + 1$;

2.3 Other PSO variants for LSGO

Currently, many PSO variants are applied to LSGO [48–53]. Many researchers have modified the standard velocity and position iteration techniques in PSO to improve the diversity of algorithms. Two particle swarm algorithms have been proposed by Cheng et al. for solving LSGO problems, CSO [29] and SLPSO [30]. In the first algorithm, a mechanism of competition between pairs is incorporated, and failed particles learn from winning particles to enhance the population diversity. Huang et al. [35] combined a new multi-stage cooperative evolution technique to develop a new three-stage cooperative evolution method (TPCSO) to improve on the ability of search and convergence with CSO. The improved algorithm divides the swarm equally into two subswarms, and modifies the update mechanism of each subswarm according to the demands of convergence and diversity within it. First stage focuses on exploring more regions to improve diversity, and in the second, two promising areas are developed by introducing excellent particles from the two subswarms. The final stage concentrates on achieving the convergence through learning of $\vec{pos}_{gbest}$. In SLPSO, the particles are ordered in the population before the evolution starts, then the particles learn from all better particles than themselves. The SLPSO algorithm has demonstrated excellent performance. Nevertheless, since it maintains diversity by employing various guiding informations for distinct particles, SLPSO tends to converge slowly. Consequently, the RES is proposed to divide dimensions of the solution into multiple regions and assign a code to each region [54]. In SLPSO, this coding scheme is used to restrict the crossing and mutation between particles to maintain diversity. This method has demonstrated excellent performance in experiments while improving both the speed of convergence of the SLPSO.

Li et al. [33] proposed APSO-DEE, a method that independently manages exploring and exploiting the population, thereby improving optimization performance. In addition, this method introduces two new learning strategies. Firstly, a method of local sparsity measurement suitable for the fitness function is introduced to evaluate the distribution and congestion of the population. According to this method, particles can be guided to sparse areas, thus establishing an exploration strategy. Then, the adaptive exploitation strategy is introduced, and the fitness difference among samples and update particles can be adjusted successfully in the evolution process using a multi-group strategy and subgroup adaptive size adjustment. In [55], a two-stage learning (TPLSO) based group optimizer is proposed. Inspired by human social cooperative learning behavior, this method involves a mass learning stage and an elite learning stage. At the mass learning stage, it chooses three particles at random to build a learning group and updates the particles in the learning group using a competition mechanism. Then, it sorts the entire population and elite particles with superior fitness are selected. Then, during the elite learning stage, the elite particles communicate with each other to explore promising domains. Yiying Zhang proposed the EAPSO [34], which introduces a structured framework that involves the creation of three different elite archives, each tasked with preserving hierarchical particles. These elite archives serve as the basis for designing six distinct learning strategies which are employed to update particle positions iteratively during the optimization process.

Although these PSO variants using novel search strategies address PSO limitations and moderately improve population diversity, there remains ample room for further advances. Given this, the Pulse-Strategy Collective Learning Swarm Optimizer (PCLSO) is proposed.

3 PCLSO algorithm

SLPSO has been shown to perform well on LSGO problems. However, in each iteration, SLPSO updates using different guidance information and the average position of individuals

within the population, even if an individual already possesses a superior fitness value. This to some extent limits the convergence speed. Additionally, in the updating step of the SLPSO, the optimal individual is not subject to updates and stagnation checks. Therefore, there is still space to further improve SLPSO. In order to enhance SLPSO's speed of convergence and accuracy, this work introduces PCLSO. Firstly, a collective learning strategy is proposed to replace the average position of the population of the SLPSO. Next, a pulse-based local search strategy is embedded, perturbing the global best individual when the population convergence stops, further enhancing the algorithm's performance. Subsequently, collective learning will be introduced, followed by an introduction to the pulse-strategy. Finally, the complete PCLSO algorithm will be presented.

3.1 Collective learning

Individual decisions often tend to be less precise compared to decisions made collectively by a majority [56]. Collective intelligence (CI) represents a form of shared intelligence that involves gathering and synthesizing input from groups of individuals to inform decisions. It emerges through the coordinated interactions and information sharing among multiple individuals. CI manifests across animals, humans, and computer networks, taking various forms of cooperative decision-making.

In optimization algorithms based on collective intelligence [57, 58], the search process is guided and optimization performance is improved by utilizing solutions based on CI. These CI-driven solutions are made by using the number of l best solutions within the population.

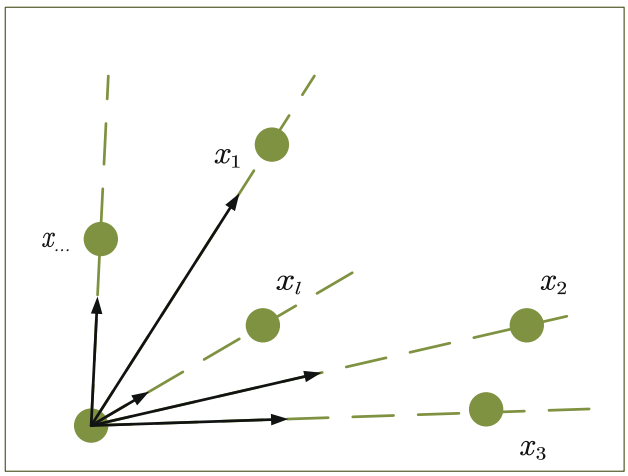


Fig. 2 Collective Learning (index numbers $1, 2, \ldots, l$ represent the individual's fitness ranking, and lengths of the arrow indicate the weights for collective learning from the individual)

As the CI concept from reference [59], a CI-driven best position is defined, which can be computed as (8).

$$\vec{c}_n(t) = \sum_{k=1}^{l} w_k \times \vec{x}_{pbestk} \tag{8}$$

where $n = 1, 2, \ldots, N$, and N represents the size of the population, $l \in [1, n]$ is a randomly chosen integer. $\vec{x}_{pbestk}$ consists of l positions with superior fitness values compared to $\vec{x}_n$, and $\vec{x}_{pbestk}$ is arranged in increasing order. According to (8), the CI-driven best position, $\vec{c}_n(t)$, represents a lineal combination of the first ranked l individual best positions ($\vec{x}_{pbestk}$).

The w_k represents a weight factor which defines the proportion of different $\vec{x}_{pbestk}$ values. It is computed as (9).

$$w_k = \frac{l - k + 1}{\sum_{k=1}^{l} k} \tag{9}$$

In (9), a superior $\vec{x}_{pbestk}$ corresponds to a higher value of w_k, thereby making a more significant contribution to $\vec{c}_n$. Conversely, an inferior pbestk results in a smaller value of w_k, contributing less to $\vec{c}_n$. Figure 2 illustrates the CI-driven strategy.

By substituting the global best position with the CI-driven optimum ($\vec{c}_n$) using (8), a new CI-driven particle search equation is defined as (10).

$$\begin{aligned} v_{n,d}(t+1) \quad &= r_1 \times v_{n,d}(t) + r_2 \times (x_{win,d}(t) - x_{n,d}(t)) \\ &\quad + c \times r_3 \times (c_{n,d}(t) - x_{n,d}(t)) \end{aligned} \tag{10}$$

With the CI-driven search method, the particles utilize the CI-driven optimum position, $\vec{c}_n$, as a reference for their searching. Because of each particle has a distinct $\vec{c}_n$ value, it becomes apparent that the approach described in (10) results in greater population diversity compared to (3). Moreover, $\vec{c}_n$ is derived from the collective knowledge of multiple high-quality individuals, indicating that the strategy in (10) is expected to demonstrate more efficient performance.

3.2 Pulse-strategy

Pulse typically refers to a brief fluctuation of electrical shock (voltage or current), resembling a pulse-like pattern, commonly used in electronics. It primarily refers to a process where a physical quantity undergoes a rapid change and quickly returns to its initial state within a short duration. A pulse is a signal that occurs briefly during an entire signal

cycle, with most of the signal cycle having no signal, much like a human pulse.

Algorithm 3 Pulse-strategy

Input: $\vec{x}_{gbest}$, x, $pmax$, $pmin$, FEs, $MaxFEs$
Output: $\vec{x}_{gbest}$, FEs
while $FEs \leqslant MaxFEs$ **do**
  for $n \leftarrow 1$ *to* N **do**
    for $d \leftarrow 1$ *to* $dimension$ **do**
      if $rand < p_m$ **then**
        Update $\vec{x}_{gbest,d}$ by using (12);
    Calculate fitness of $\vec{x}_{gbest}$;
    FEs = FEs + 1;
    Update σ by using (15);

Pulse-strategy is a time management approach that suggests focusing one's time and energy intensively on an individual or dimension within a specific timeframe, rather than evenly distributing effort among all individuals. The goal is to achieve breakthroughs in the short term. The pulse-strategy allows individuals to establish local advantages and enhance convergence. Individuals with strong learning abilities tend to benefit more from this approach.

Based on the concepts mentioned above, this paper introduces the integration of a pulse-strategy into an evolutionary algorithm to address common weaknesses like insufficient deep search and early convergence of most evolutionary algorithms in LSGO problems. When algorithm convergence stagnation is detected, the pulse-strategy is applied to the the global best particle, shown in Fig. 3. Pseudocode for the pulse strategy is provided in **Algorithm** 3, with specific details outlined below.

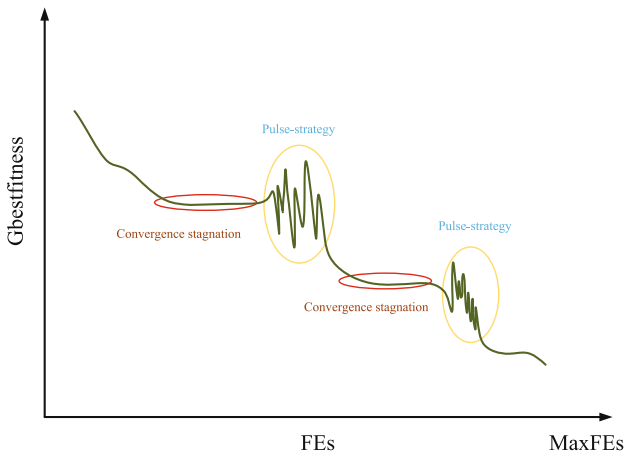


Fig. 3 Pulse-strategy (The x-axis indicates the FEs used during the optimization process, while the y-axis indicates the change of the value of the global best particle fitness (Gbestfitness). The pulse-strategy will be employed when FEs of Gbestfitness remains unchanged for more than a threshold number of function evaluations (e.g. convergence stagnation))

(1) Mutation Rate: In the pulse-strategy, the mutation rate controls the extent of changes made to the global best individual. Since the global best individual already possesses the best fitness value, only small perturbations are applied to it. The probability of these perturbations decreases gradually as the iterations progress. The mutation rate is calculated as shown in (11).

$$p_m = 0.04 \times (1 - FEs/MaxFEs) + 0.01 \tag{11}$$

where *FEs* represent the number of current fitness evaluations and *MaxFEs* denotes the allowed maximum fitness evaluation count. In this manner, all of the dimensions for the global optimal individual could be modified probabilistically, thereby increasing chances for the global optimal individual to obtain a better fitness value.

(2) Mutation Dimension Regeneration: (12) is employed for the regeneration of mutation dimensions. To enhance diversity, the difference between two randomly selected positions is used to perturb the dimensions of the global best individual.

$$x_{n,d} = \vec{x}_{gbest}(d) + \sigma \times rand \times (x_{r1,d} - x_{r2,d}) \tag{12}$$

where *rand* is generated as a random number ranging from 0 to 1, and *r*1 and *r*2 are two individuals that are different from the global optimal. Specifically, *r*2 has a certain probability of being a randomly generated individual that does not come from the existing population. The generation probability is as (13).

$$p = (0.01 + (0.1 - 0.01) \times (1 - FEs/MaxFEs)) \tag{13}$$

Similar to *pm*, the probability p also decreases gradually with an increase in the number of iterations. In this case, the formula for generating particle *r*2 is given by (14).

$$x_{r2,d} = x_{\min} + rand \times (x_{\max} - x_{\min}) \tag{14}$$

where $x_{\max}$ and $x_{\min}$ represent upper and lower bounds of the search space, and *rand* is in the range of 0 and 1. The introduction of this individual further enhances the proposed algorithm's diversity. By periodically incorporating random solutions, the algorithm maintains a broader sampling of the search space, thereby reducing premature convergence and improving the probability of escaping local optima.

Furthermore, in (12), σ represents the contraction factor, and its value changes as indicated in (15).

$$\begin{cases} \sigma = \min(0.9 \times \sigma, \sigma_{max}) & \text{if } newfitness \leq fitness \\ \sigma = \max(1.1 \times \sigma, \sigma_{min}) & \text{otherwise} \end{cases} \tag{15}$$

where *newfitness* and *fitness* denote the objective function values before and after the update, respectively (considering a minimization problem). The parameter σ is initialized to $\sigma = 1$ and adaptively updated according to (15), with its evolution constrained by predefined bounds: $\sigma_{\max} = 5$ as the upper limit and $\sigma_{\min} = 10^{-5}$ as the lower limit. These bounds ensure that σ varies within a controlled range while maintaining the algorithm's stability.

Under influence of the contraction factor, if the previous generation's update is effective, it expands the search space. Conversely, it narrows the search domain. This promotes a better balance of exploitation and exploration in the PCLSO algorithm, further enhancing its convergence capabilities.

Additionally, in the iterative optimization process of algorithms, individuals may sometimes exceed the defined search boundaries, which can affect the optimization process and the quality of solutions obtained. Existing approaches to tackle this problem often involve setting fixed upper or lower limit values for individuals that go beyond the boundary [60] or employing delayed estimation techniques based on boundary values. However, these methods may not always make optimal use of population information.

To overcome this limitation, a novel boundary control strategy is introduced based on individual best position information. This approach utilizes information from the current iteration's population to adjust the positions of individuals outside the boundary, rather than relying on fixed limits or delayed estimation. The proposed method is formalized in (16), which places individuals outside the boundary at the corresponding dimensions of their individual best positions. In the next iteration, they will be searched in positions that are potentially more favorable than those determined by traditional boundary control strategies. Hence, it can enhance PCLSO's exploitation capabilities.

$$x_{n,d} = x_{pbestn,d} \tag{16}$$

where $x_{pbestn,d}$ denotes the corresponding dimension within the optimal position of the particle.

3.3 Procedure of the proposed PCLSO algorithm

On the basis of the above enhancements, PCLSO is introduced. The pseudocode of PCLSO is shown in **Algorithm** 4. The PCLSO algorithm operates as follows:

Algorithm 4 PCLSO

Input: $MaxFEs$, N, $\sigma = 1$, $n_{stag} = 0$
Output: x_{gbest} and $gbestfitness$
1 Initialization the particle's population $n \leftarrow 1, 2, ..., N$ and calculate all $fitness$;
2 $FEs = FEs + popsize$;
3 **while** $FEs \leqslant MaxFEs$ **do**
4 Sort particles with $fitness$;
5 $pastgbestfitness = gbestfitness$;
6 **for** $n \leftarrow 1$ *to* N **do**
7 **if** $rand \leqslant PL_n$ **then**
8 **for** $d \leftarrow 1$ *to* D **do**
9 $\vec{x}_{win,d} = \vec{x}_{randn(n-1),d}$;
10 Update the position by using (10) ;
11 Boundary control;
12 Calculate the $fitness(n)$;
13 $FEs = FEs + 1$;
14 **if** $|pastgbestfitness - gbestfitness| \leqslant pastgbestfitness/1000$ **then**
15 $n_{stag} = n_{stag} + 1$;
16 **if** $n_{stag} > MaxFEs/300$ **then**
17 Execute Pulse-strategy according to **Algorithm** 3;

Step1: Parameter setting and random initialization of population, then compute the fitness for each particle(line 1);
Step2: Sort Particles based on their fitness values (line 4);
Step3: Update particle positions through social and collective learning based on (10) (lines 8–11);
Step4: Perform boundary control according (16) (line 11);
Step5: Calculate the fitness of all particles after the updates (lines 12–13);
Step6: According to **Algorithm** 3, when convergence stagnation occurs, employ the pulse strategy to update the position of the pos_{gbest}(line 17);
Step7: Verify if the algorithm meets the criteria for termination. If it does, conclude the algorithm and report the optimal solution; if not, return to Step 2;

Using the procedure described above, PCLSO possesses several advantages over SLPSO:

(1) Utilization of Collective Information: Collective information is derived from the combined knowledge of multiple high-quality individuals, using the collective wisdom of the population. This can lead to more efficient and informed decision-making. Meanwhile, since each particle has its unique $\vec{c}_i$ value, this approach promotes superior population diversity compared to SLPSO.

(2) Improved Exploration and Enhanced Convergence: PCLSO introduces a pulse-strategy that enhances exploration capabilities by allowing for small perturbations to the global best individual when convergence stagnation is detected. This promotes better global search ability. The combination of pulse-strategy and adaptive boundary control in PCLSO results in improved convergence performance, allowing it to find better solutions in a variety of scenarios.
(3) Adaptive Boundary Control: PCLSO employs a boundary control strategy that adapts to individual best positions, rather than relying on fixed boundary limits or delayed estimations. This approach optimally utilizes population information, ensuring more effective boundary handling.
(4) Increased Diversity: PCLSO's mechanisms, such as mutation dimension regeneration and contraction factor adaptation, contribute to increased diversity within the population, helping the algorithm escape local optima.
(5) Better Handling of Local Optima: By switching between different strategies, PCLSO is better equipped to escape local optima, therefore it is appropriate for a wider variety of problems.

3.4 Complexity analysis

In this paper, when assessing the complexity for PCLSO, the primary focus is on the time required for operations within each generation. The population size and problem dimension are denoted as N, D, respectively. Initially, the cost associated with initializing and evaluating candidate solutions is $O(N \times D)$. Subsequently, the maximum cost incurred during the execution of operators within the for loop is $O((N-1) \times D)$. Additionally, sorting the fitness values involves the time complexity of the quicksort algorithm, taking $O(N \times log(N))$. Thus, the overall complexity of PCLSO can be described as $O(N \times log(N)) + O((N-1) \times D)$.

Table 2 Parameters settings

Algorithm	Parameters settings
CSO	$popsize = 500, \varphi = 0.1$
SLPSO	$popsize = 200, \varphi = 0.1, M = 100, \beta = 0.001$
EAPSO	$popsize = 100$
APSO-DEE	$popzize = 100, \varphi = 0.3$
TPCSO	$popsize = 500$
DCSO	$popsize = 500, \varphi = 0.1, d = 0.45$
EADE	$popsize = 50$
DECC-DG	$popsize = 100, maxFEcycle = 10000$
DECC-RDG	$popsize = 100, maxFEcycle = 10000$
DECC-MDG	$popsize = 100, maxFEcycle = 10000$
PCLSO	$popsize = 200, c = 0.1$

In the worst-case scenario, when convergence stagnation occurs, it is necessary to calculate the time used for the pulse strategy, which is $O(1 \times N)$. In this particular case, the total complexity of PCLSO becomes $O(N \times log(N)) + O(N \times D)$.

4 Experimental results and analysis

To validate efficiency and effectiveness of PCLSO, a number of experiments were performed on two popular LSGO benchmark suites (CEC'2010 [61] and CEC'2013 [62]) from various perspectives.

The PCLSO is compared to a number of recently introduced and advanced algorithms. These algorithms include the EAPSO, APSO_DEE, CSO, SLPSO, TPCSO, DCSO, EADE [63], DECC_DG [24], DECC_RDG [64], and DECC_MDG [27]. Notably, among these comparative algorithms, the last three are algorithms based on cooperative co-evolution. The algorithm is run 25 times independently to get the statistical results.

4.1 Parameter settings

The experiment parameters used in PCLSO and comparison algorithms are as follows. To ensure fairly, the *MaxFEs* is set to 3*E*6. The subsequent subsection further explores the impact of the parameter on PCLSO's performance. Recommended parameter values from their original papers, as shown in Table 2, are adhered to for the compared algorithms.

4.2 Experiment results on CEC'2010

The CEC'2010 LSGO benchmark is composed of 20 functions, with each function having a decision variable dimension of 1000. The characteristics of these functions are as [62].

The test results on the CEC'2010 test suite are displayed in Table 3. The performance comparison between PCLSO and SLPSO was initially observed. For the three fully separable functions (F_1-F_3), PCLSO outperformed SLPSO on all functions. Regarding the 15 partially separable functions (F_4-F_{18}), in 14 functions (specifically, F_4, F_6-F_{18}), PCLSO performed better than SLPSO, with SLPSO surpassing PCLSO only on 1 function. For the last two inseparable functions (F_{19} and F_{20}), PCLSO's performance was also superior to SLPSO.

Besides SLPSO, as shown in Table 3, it can be observed that of the 20 LSGO problems in the CEC'2010 LSGO benchmark test suite, PCLSO exhibits a considerably better performance than CSO, EAPSO, APSO_DEE, TPCSO,

Table 3 Experimental results of PCLSO and the compared algorithms on the CEC'2010 LSGO test suite

Function	Item	CSO	SLPSO	EAPSO	APSO_DEE	TPCSO	DCSO	EADE	DECC_DG	DECC_RDG	DECC_MDG	PCLSO
F_1	Mean	4.69799E-12	1.52174E-14	5.65019E-01	8.66746E+08	1.06477E-20	8.23813E-19	3.00415E-20	2.65222E+01	2.39008E-02	5.60531E-03	**3.12519E-21**
	Sd.	6.25469E-13	2.03724E-15	1.23684E+00	2.88578E+08	1.81240E-21	5.37669E-20	1.89724E-20	2.62429E+01	4.41813E-02	4.94800E-03	**2.67752E-22**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_2	Mean	7.72578E+03	3.02897E+03	6.83328E+03	5.30030E+03	3.25450E+03	2.21084E+03	1.48070E+03	2.91015E+03	3.08732E+03	3.01594E+03	**6.45237E+02**
	Sd.	3.88716E+02	5.22275E+02	1.42088E+02	2.89647E+02	6.70838E+01	8.64015E+01	9.95687E+01	9.90309E+01	7.96678E+01	1.21569E+02	**3.18340E+01**
	h	+	+	+	+	+	+	+	+	+	+	≈
	Sd.	1.16465E-10	1.39487E-11	1.32309E-01	2.39013E-01	2.27814E-01	5.78146E-02	2.24467E-01	3.79985E-01	2.39552E-01	4.22931E-01	**3.17764E-15**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_4	Mean	1.33312E+12	1.45626E+12	5.70190E+11	6.49979E+12	7.53204E+11	7.60227E+11	**9.93429E+10**	2.35763E+13	1.54965E+12	1.45012E+12	5.00274E+11
	Sd.	2.83011E+11	2.88657E+11	8.74137E+10	2.53209E+12	8.73508E+10	3.18286E+11	**4.87645E+10**	5.29028E+12	2.50733E+11	5.29951E+11	5.50112E+10
	h	+	+	≈	+	+	≈	–	+	+	+	≈
F_5	Mean	4.18625E+06	1.17436E+07	1.63991E+08	4.75717E+07	2.84624E+07	2.13154E+07	7.98740E+07	3.03112E+08	1.67764E+08	1.68635E+08	**2.18898E+07**
	Sd.	1.29716E+06	3.61482E+06	2.28769E+07	2.93388E+06	3.27104E+06	3.78743E+06	1.26959E+07	6.10729E+07	1.86894E+07	2.21807E+07	**2.98488E+06**
	h	–	–	+	+	+	≈	+	+	+	+	≈
F_6	Mean	2.19550E-07	6.77190E-08	1.97538E+01	2.05378E+01	1.61840E+00	1.96696E+01	1.90520E+01	1.06544E+01	1.04530E+01	1.07333E+01	**3.80612E-09**
	Sd.	3.58937E-09	1.44819E-09	2.96773E-02	4.69797E-02	1.38346E-01	7.39411E-02	2.24586E-01	3.69501E-01	9.85993E-01	6.83931E-01	**5.19193E-11**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_7	Mean	9.27937E+03	9.35627E+04	3.65903E+00	7.20723E+08	2.88309E+03	3.33511E-01	**1.62995E-01**	1.84448E+06	6.09532E+01	3.71560E+01	1.16117E+01
	Sd.	2.19585E+03	4.48752E+04	3.05518E+00	2.81883E+08	9.67912E+02	1.50028E-01	**2.76470E-01**	4.47053E+05	3.54244E+01	1.27137E+01	5.24162E+00
	h	+	+	–	+	+	–	–	+	+	+	≈
F_8	Mean	3.86187E+07	3.65416E+07	1.92644E+07	4.70732E+07	2.45305E+07	2.09435E+07	**9.12503E-05**	3.11085E+07	4.04315E-01	2.85856E-01	1.89073E+07
	Sd.	1.04991E+05	1.26405E+05	6.83577E+06	7.21077E+05	1.38388E+06	2.44212E+05	**1.93706E-04**	2.75651E+05	4.44476E-01	1.42414E-01	1.72750E+05
	h	+	+	≈	+	+	+	–	+	–	–	≈
F_9	Mean	6.34847E+07	5.86841E+07	5.22921E+07	2.29015E+09	4.01842E+07	5.04494E+07	3.60199E+07	1.90358E+08	1.66169E+08	1.84813E+08	**2.26047E+07**
	Sd.	2.23208E+06	6.88127E+06	4.61024E+06	3.37810E+08	2.10391E+06	4.92217E+06	2.82675E+06	2.84000E+07	2.22706E+07	1.72207E+07	**6.31409E+05**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_{10}	Mean	9.56641E+03	8.95021E+03	7.33726E+03	5.55910E+03	3.31897E+03	**2.31743E+03**	2.74495E+03	6.53532E+03	6.18566E+03	6.22161E+03	6.89173E+03
	Sd.	8.71543E+01	6.54803E+01	4.15692E+02	2.63428E+02	8.25373E+01	**6.11636E+01**	1.50405E+02	5.98752E+01	4.06881E+01	6.85123E+01	1.24083E+03
	h	+	+	≈	–	–	–	–	≈	≈	≈	≈
F_{11}	Mean	4.15348E-08	3.02718E-09	1.71828E+02	1.55028E+02	9.73683E+00	2.21319E+01	1.21714E+02	5.63185E+00	4.56389E+00	4.74337E+00	**2.02505E-13**
	Sd.	4.66363E-09	5.67610E-10	1.32335E+01	9.79110E+00	3.09367E+00	1.73716E+00	8.23249E+00	1.02398E+00	1.37810E-01	6.07776E-01	**6.64652E-15**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_{12}	Mean	5.18544E+05	5.48005E+05	**4.78347E+03**	1.39594E+06	2.84185E+04	1.67641E+04	2.79971E+04	2.78468E+04	2.00341E+04	2.02719E+04	1.48270E+05
	Sd.	4.14523E+04	5.98149E+04	**9.31271E+02**	1.90960E+05	1.72423E+03	1.37621E+03	3.43732E+03	2.66552E+03	1.83475E+03	1.19389E+03	2.25923E+04
	h	+	+	–	+	–	–	–	–	–	–	≈
F_{13}	Mean	9.17917E+02	1.25764E+03	1.34419E+03	2.80010E+08	8.99464E+02	**1.07936E+03**	1.22399E+03	7.93347E+04	8.98744E+02	9.03820E+02	1.14146E+03
	Sd.	1.65849E+02	9.63787E+02	7.15818E+02	1.29586E+08	3.24675E+02	**2.50775E+02**	1.88340E+02	4.80630E+04	1.25437E+02	1.10693E+02	6.95649E+02
	h	≈	≈	≈	+	≈	≈	≈	+	≈	≈	≈
F_{14}	Mean	2.87816E+08	2.85192E+08	1.48737E+08	3.74703E+09	1.47998E+08	1.80124E+08	1.53405E+08	7.47770E+08	7.12626E+08	7.45680E+08	8.86513E+07
	Sd.	1.50426E+07	2.30764E+07	1.05429E+07	2.23687E+08	2.76568E+06	1.35219E+07	6.05609E+06	4.53538E+07	3.92505E+07	4.51063E+07	5.31769E+06

Table 3 (continued)

Function	Item	CSO	SLPSO	EAPSO	APSO_DEE	TPCSO	DCSO	EADE	DECC_DG	DECC_RDG	DECC_MDG	PCLSO
	h	+	+	+	+	+	+	+	+	+	+	≈
F_{15}	Mean	1.00740E+04	1.02113E+04	7.72664E+03	5.77096E+03	3.49285E+03	**2.29599E+03**	3.17610E+03	6.57083E+03	6.56930E+03	6.52158E+03	8.78118E+03
	Sd.	4.09684E+01	7.39515E+01	4.93181E+02	2.59962E+02	8.85635E+01	**9.40291E+01**	1.56962E+02	7.90149E+01	5.48451E+01	8.39417E+01	1.66059E+03
	h	+	+	≈	–	–	–	–	≈	≈	≈	≈
F_{16}	Mean	5.62509E-08	3.83764E-09	3.72377E+02	3.24189E+02	3.63561E+01	7.30248E+01	2.95189E+02	4.44275E-05	1.95856E-05	1.79568E-05	**3.47455E-13**
	Sd.	3.09443E-09	6.29844E-10	6.14637E+00	5.63854E+00	7.62922E+00	7.22317E+00	1.35717E+01	5.40352E-07	3.54060E-07	1.40411E-06	**3.89180E-15**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_{17}	Mean	2.20454E+06	2.53139E+06	3.04952E+04	1.76473E+06	**1.05621E+05**	1.11076E+05	1.45620E+05	2.00354E+05	1.99250E+05	1.95179E+05	1.80239E+06
	Sd.	2.09855E+05	1.16093E+05	2.76239E+03	7.01542E+04	**5.26092E+03**	3.66000E+03	1.28556E+04	8.61655E+03	8.69259E+03	7.34745E+03	2.10348E+05
	h	+	+	–	≈	–	–	–	–	–	–	≈
F_{18}	Mean	1.60539E+03	3.96991E+03	5.14977E+03	3.30548E+10	3.32565E+03	3.04768E+03	2.61337E+03	2.67169E+10	**1.02805E+03**	1.12744E+03	1.71297E+03
	Sd.	2.74180E+02	2.68664E+03	1.09768E+03	5.05896E+09	7.41392E+02	6.26750E+02	3.53150E+02	3.10538E+09	**8.65052E+01**	1.66706E+02	7.27462E+02
	h	≈	≈	+	+	+	+	≈	+	–	≈	≈
F_{19}	Mean	1.00972E+07	9.94787E+06	**9.26446E+05**	4.84194E+06	2.22800E+06	4.99965E+06	1.30561E+06	1.94739E+06	1.90389E+06	1.92451E+06	6.27768E+06
	Sd.	2.70281E+05	5.36571E+05	**6.23440E+04**	2.20430E+05	8.72675E+04	2.18940E+05	4.36945E+04	5.33027E+04	1.08495E+05	7.20865E+04	5.88274E+05
	h	+	+	–	–	–	–	–	–	–	–	≈
F_{20}	Mean	**1.01644E+03**	1.03666E+03	2.49902E+03	3.76872E+10	1.66568E+03	1.95878E+03	2.18432E+03	1.74148E+11	4.27000E+03	4.32436E+03	1.05317E+03
	Sd..	**4.86326E+01**	3.74559E+01	3.07856E+02	7.27286E+09	1.58053E+02	1.53604E+02	1.82477E+02	1.31565E+10	1.45691E+02	3.02945E+02	6.04504E+01
	h	≈	≈	+	+	+	+	+	+	+	+	≈
	w/t/l	**16/3/1**	**16/3/1**	**11/5/4**	**16/1/3**	**14/1/5**	**11/3/6**	**10/2/8**	**15/2/3**	**12/3/5**	**12/4/4**	**0/20/0**
	median Friedman	**6.44**	**6.21**	**6.51**	**9.4**	**4.82**	**4.59**	**4.34**	**7.99**	**5.94**	**6.04**	**3.7**

DCSO, EADE, DECC_DG, DECC_RDG, and DECC_MDG on 8 functions, including F_1-F_3, F_5, F_6, F_9, F_{11}, and F_{16}. Moreover, all the comparison algorithms fail to show a significant advantage over PCLSO on 4 or more functions. Therefore, PCLSO demonstrates outstanding behavior on the CEC'2010 LSGO benchmark test suite, generally surpassing the comparison algorithms.

Furthermore, a Friedman test was performed to further verify the efficiency of PCLSO. Additionally, significance analysis was conducted using the Wilcoxon rank sum test with a $\alpha = 0.05$ significance level to evaluate PCLSO against each of the comparison algorithms. Symbols $+$(superior), $\approx$(approximately equal) to the proposed PCLSO, and $-$ (inferior) were used to denote the results. Figure 7 displays these results graphically. The Friedman and Wilcoxon rank sum test results are presented at the bottom of Table 3. It is evident that among the 20 functions contained in the CEC'2010 test suite, PCLSO obtains the best average rankings.

To demonstrate the accuracy and efficiency of PCLSO on the CEC'2010 LSGO test suite, the evolutionary convergence curves of PCLSO have been plotted along with the comparison algorithms, as shown in Fig. 4. On the fully separable F_1-F_3, PCLSO achieves the best convergence performance and quickly finds superior solutions. For the partially separable functions (F_4-F_{18}), PCLSO still demonstrates good overall performance. Regarding the fully inseparable functions F_{19} and F_{20}, it is evident that PCLSO can achieve superior results compared to the other algorithms, except for EAPSO. The boxplots for all 20 functions on CEC'2010 are shown in Fig. 5. In addition, Fig. 6 depicts a radar chart comparing the performance of PCLSO and ten competitive algorithms on the CEC'2010 test suite in a 1000-dimensional space.

4.3 Experiment results on CEC'2013

The CEC'2013 benchmark functions are an extension of the CEC'2010 benchmark functions to include various subcomponents, subcomponents of unequal sizes, and contribution imbalances between consistent and conflicting overlapping functions. This extension better simulates real-world problems and increases the difficulty of the test set, making it a valuable tool for validating the performance of LSGO algorithms. Despite being proposed in 2013, this test set continues to be widely used in subsequent CEC Large-Scale Optimization Competitions, such as CEC'2018 and CEC'2019.

The CEC'2013 LSGO benchmark consists of 15 functions, with most of them having a decision variable dimension of 1000 (F_1-F_{12} and F_{15} are 1000 dimensions, F_{13} and F_{14} are 905 dimensions). The summary of characteristics of functions in CEC'2013 can be found in [62].

The mean fitness (*Mean*) and standard deviation (*Sd.*) of PCLSO and comparison algorithms across all 15 CEC'2013 test functions are shown in Table 4. Furthermore, the statistical outcomes are shown at the bottom of the table. The bold font has been used to highlight the best results (minimum averages) for each function.

PCLSO was consistently outperformed by SLPSO when the three fully separable functions (F_1-F_3) were initially compared. In the case of the eight partially separable functions (F_4-F_{11}), PCLSO performed better than SLPSO in six functions (specifically, F_4, F_6, F_7, F_8, F_9, and F_{11}), while SLPSO only surpassed PCLSO in two functions (F_5 and F_9). Notably, PCLSO exhibited significantly superior performance to SLPSO, particularly for the function F_6. For the three overlapping functions (F_{12}-F_{14}) and the final fully inseparable function (F_{15}), PCLSO's overall performance also outshone SLPSO.

Besides SLPSO, Table 4 demonstrates that of the 15 LSGO problems in the CEC'2013 LSGO benchmark test set, PCLSO exhibits significantly better performance than CSO, SLPSO, APSO_DEE, TPCSO, DCSO, EADE, DECC_DG, DECC_RDG, and DECC_MDG on F_1, F_2, F_3, F_{12}, and F_{13}, respectively. Moreover, all the comparison algorithms fail to show a significant advantage over PCLSO on four or more functions. As a result, for even complex LSGO problems within the CEC'2013 benchmark test suite, PCLSO maintains excellent performance.

The Friedman test results for the CEC'2013 test set are displayed in the last row of the Table 4. It is clear that PCLSO has the greatest average ranking and the highest frequency of being ranked best among all algorithms. This further underscores that PCLSO generally outperforms the comparison algorithms on the CEC'2013 LSGO test set.

Additionally, to demonstrate the effectiveness and convergence performance of PCLSO on the CEC'2013 LSGO benchmark test set, the curves of PCLSO's evolutionary convergence are plotted along with those of the comparison algorithms. Figure 8 shows the convergence curves for all different algorithms across all fifteen functions. In addition, the boxplots of the different algorithms on all 15 functions are presented in Fig. 9. As depicted in Fig. 8, for the fully separable functions F_1-F_3, PCLSO achieves better convergence results than all the comparison algorithms. In the case of partially separable functions F_5-F_{13}, only PCLSO, CCOS, and SLPSO exhibit faster convergence and more accurate results, with PCLSO still outperforming CCOS and SLPSO. For the overlapping and fully inseparable functions F_{13}-F_{15}, most algorithms demonstrate similar performance, except for CSO.

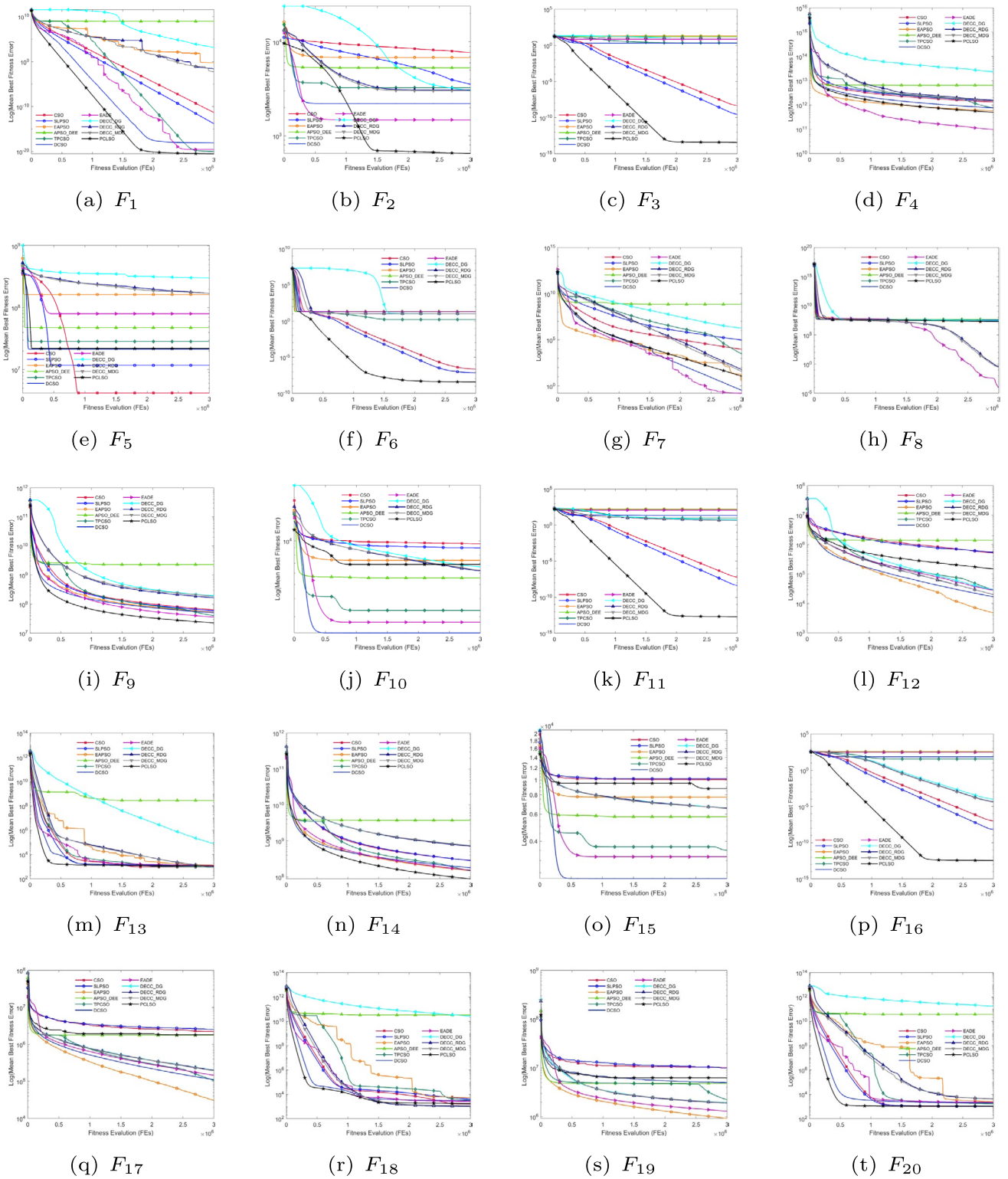
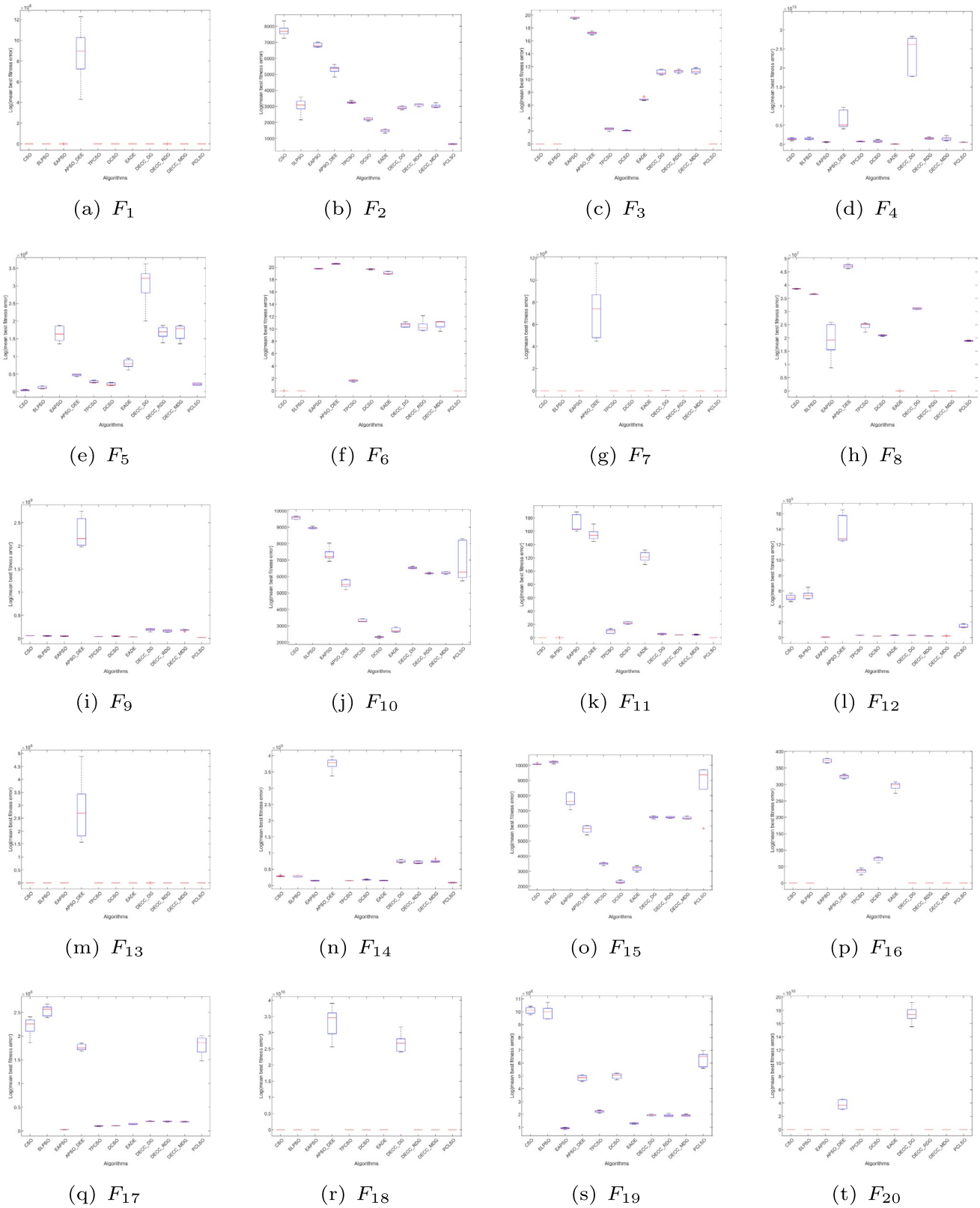


Fig. 4 Convergence curves of different algorithms obtained on the CEC'2010 benchmarks



(a) F_1 (b) F_2 (c) F_3 (d) F_4

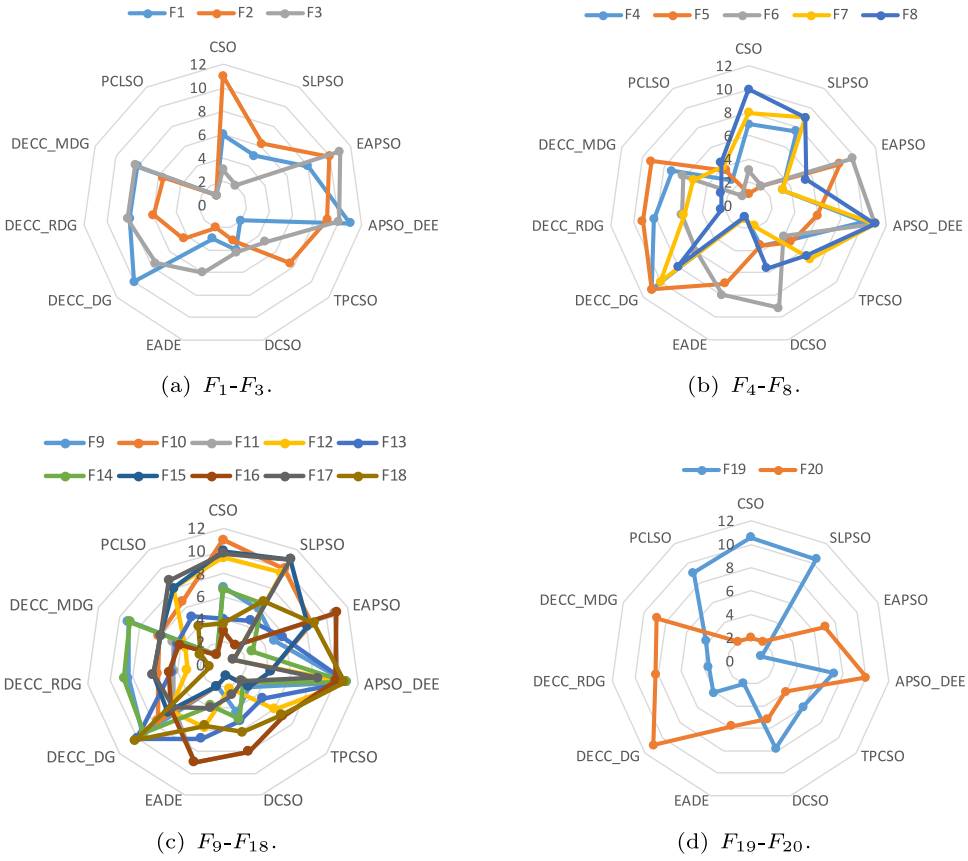
(e) F_5 (f) F_6 (g) F_7 (h) F_8

(i) F_9 (j) F_{10} (k) F_{11} (l) F_{12}

(m) F_{13} (n) F_{14} (o) F_{15} (p) F_{16}

(q) F_{17} (r) F_{18} (s) F_{19} (t) F_{20}

Fig. 5 Boxplot of different algorithms obtained on the CEC'2010 benchmarks



(a) F_1-F_3. (b) F_4-F_8.

(c) F_9-F_{18}. (d) F_{19}-F_{20}.

Fig. 6 Radar graph based on the experimental results from Friedman test obtained by the applied algorithms on CEC'2010

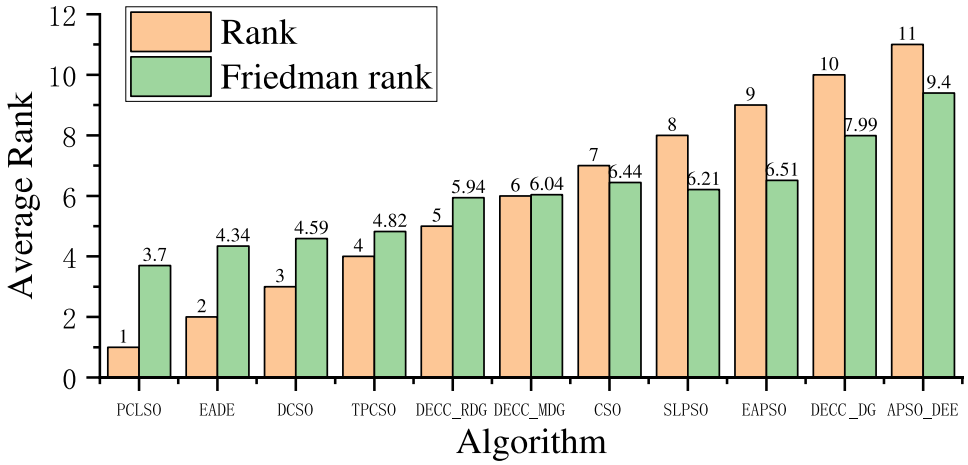


Fig. 7 Friedman rank of PCLSO and compared algorithms on CEC'2010

Figure 10 depicts a radar chart comparing the performance of PCLSO and ten competitive algorithms on the CEC'2013 test suite in a 1000-dimensional space. And Fig. 11 displays the Fridman test results graphically. According to Fig. 11, it is evident that across the selected 15 test functions, PCLSO demonstrates a clear advantage over the comparison algorithms, offering superior convergence performance compared to them.

4.4 Parameter analysis

Recent years have witnessed significant advancements in metaheuristic parameter tuning methodologies, with notable approaches including CRS-Tuning, F-Race, and REVAC gaining prominence in the research community [65]. CRS-Tuning employs a chess-inspired rating system that evaluates configurations through tournament-style comparisons,

Table 4 Experimental results of PCLSO and the compared algorithms on the CEC'2013 LSGO test suite

Function	Item	CSO	SLPSO	EAPSO	APSO_DEE	TPCSO	DCSO	EADE	DECC_DG	DECC_RDG	DECC_MDG	PCLSO
F_1	Mean	8.26001E-12	3.95866E-14	1.07055E-01	8.47216E+08	1.53341E-20	1.25701E-18	1.06165E-18	1.11918E+02	5.09606E-02	1.01052E-01	**2.27350E-21**
	Sd...	1.28982E-12	1.11805E-14	1.64622E-01	1.39447E+08	8.03717E-22	2.13275E-19	2.02217E-18	4.57538E+01	4.33341E-02	9.82706E-02	**1.36519E-22**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_2	Mean	8.62377E+03	6.74185E+03	1.60166E+04	1.17279E+04	5.27540E+03	2.83873E+03	2.17060E+03	7.67137E+03	7.67185E+03	7.72020E+03	**6.40355E+02**
	Sd..	2.51088E+02	2.05896E+02	3.53322E+02	4.32462E+02	7.93369E+02	2.09299E+02	8.66428E+01	5.00489E+02	2.73380E+02	3.94001E+02	**3.28608E+01**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_3	Mean	2.54490E-09	1.73124E-10	1.96010E+01	1.71982E+01	1.82661E+00	1.99951E+00	7.10606E+00	1.14607E+01	1.11014E+01	1.10176E+01	**3.83693E-14**
	Sd.	2.21045E-10	1.46032E-11	1.00270E-01	2.09031E-01	6.68038E-02	1.29449E-01	2.59601E-01	8.55056E-01	5.15173E-01	4.25492E-01	**1.58882E-15**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_4	Mean	1.44711E+10	1.06656E+10	4.99603E+09	7.97759E+10	6.56503E+09	6.57068E+09	**1.10288E+09**	6.43879E+10	5.88556E+10	5.58783E+10	5.51319E+09
	Sd.	4.46214E+09	4.76415E+09	2.40854E+09	2.51422E+10	2.31145E+09	1.32304E+09	**4.64073E+08**	2.01334E+10	5.39362E+09	1.05495E+10	8.85442E+08
	h	+	≈	≈	+	≈	≈	–	+	+	+	≈
F_5	Mean	**5.63464E+05**	7.01124E+05	4.32326E+06	1.98581E+06	1.41158E+06	1.03841E+06	2.08638E+06	6.57978E+06	5.96205E+06	5.62914E+06	8.57174E+05
	Sd.	**6.23649E+04**	1.63967E+05	7.25136E+05	4.21550E+05	4.37040E+04	1.73512E+05	2.83376E+05	3.09007E+05	3.02833E+05	5.13832E+05	1.13014E+05
	h	–	≈	+	+	+	≈	+	+	+	+	≈
F_6	Mean	2.61626E-07	2.99517E-04	5.89826E+04	5.58441E+03	7.53478E+02	8.94889E+02	3.85035E+01	7.74756E+00	7.80126E+00	7.93196E+00	**3.74568E-10**
	Sd.	1.61008E-07	5.04521E-04	8.00081E+03	7.98241E+01	1.25464E+03	1.92367E+03	4.26200E-01	6.51894E-01	6.13710E-01	6.39772E-01	**3.60020E-12**
	h	+	+	+	+	+	+	+	+	+	+	≈
F_7	Mean	5.01644E+06	1.78118E+07	2.08393E+06	1.11579E+09	8.57407E+06	**2.47148E+05**	1.98200E+06	8.38692E+08	8.87394E+07	8.94724E+07	2.13881E+06
	Sd.	2.41625E+06	5.38479E+06	2.73677E+05	2.55574E+08	2.37750E+06	**7.65217E+04**	6.40268E+05	2.05340E+08	2.74879E+07	1.77015E+07	9.48322E+05
	h	+	+	≈	+	+	–	≈	+	+	+	≈
F_8	Mean	2.36602E+14	3.38349E+14	9.30131E+13	1.04867E+15	1.17178E+14	1.17481E+14	**7.15066E+12**	5.82750E+15	7.27045E+15	5.70545E+15	1.29553E+14
	Sd.	9.73217E+13	6.94943E+13	3.02614E+13	6.17344E+14	4.29253E+13	3.34740E+13	**6.31610E+12**	2.16774E+15	3.40213E+15	1.47806E+15	3.39984E+13
	h	+	+	≈	+	≈	≈	–	+	+	+	≈
F_9	Mean	**3.55163E+07**	5.22223E+07	3.25252E+08	2.06529E+08	8.98603E+07	7.48263E+07	1.53795E+08	5.36771E+08	5.38956E+08	5.45158E+08	5.70328E+07
	Sd.	**5.90321E+06**	1.60300E+07	4.03563E+07	2.47526E+07	1.21363E+07	1.66502E+07	1.50317E+07	2.68808E+07	3.30910E+07	3.04264E+07	1.43816E+07
	h	–	≈	+	+	+	≈	+	+	+	+	≈
F_{10}	Mean	**9.02998E-05**	3.24944E-01	1.12937E+04	5.22378E+05	1.57828E+03	9.62470E+03	2.00241E+03	2.60302E-02	5.48840E-02	3.86893E-02	1.02417E-04
	Sd.	**1.55064E-04**	7.11495E-01	1.99387E+04	6.48753E+05	3.52545E+03	1.77010E+04	3.94087E+02	2.80607E-03	1.91377E-02	5.06267E-03	2.29011E-04
	h	≈	+	+	+	+	+	+	+	+	+	≈
F_{11}	Mean	1.94997E+09	1.60327E+10	2.71770E+08	7.78817E+10	2.84735E+08	3.93192E+09	**3.09220E+08**	9.98775E+10	5.71303E+08	1.25170E+10	3.50307E+09
	Sd.	9.04976E+08	1.20828E+10	5.92689E+07	6.45579E+10	4.91774E+07	3.05037E+09	**9.94602E+07**	3.84162E+10	1.10213E+08	7.76792E+09	5.99758E+09
	h	≈	≈	–	+	–	≈	–	+	–	≈	≈
F_{12}	Mean	1.05479E+03	1.15855E+03	2.52636E+03	3.49441E+10	1.62643E+03	1.96267E+03	2.12914E+03	3.81916E+11	4.45001E+03	4.10029E+03	**1.05949E+03**
	Sd.	2.32943E+01	1.24559E+02	1.73752E+02	4.90472E+09	1.39206E+02	1.24501E+02	1.60026E+02	2.64316E+10	4.84076E+02	2.78648E+02	**5.07267E+01**
	h	≈	≈	+	+	+	+	+	+	+	+	≈
F_{13}	Mean	9.95042E+08	1.90919E+09	5.44883E+08	1.41887E+10	8.58255E+08	4.14954E+09	5.66801E+08	2.76134E+10	1.44176E+09	1.49614E+09	**5.95479E+08**
	Sd.	3.88276E+08	1.02136E+09	1.25609E+08	2.82052E+09	3.06590E+08	8.33586E+09	2.45886E+08	5.02108E+09	4.65523E+08	4.63739E+08	**9.56561E+07**
	h	≈	+	≈	+	≈	≈	≈	+	+	+	≈
F_{14}	Mean	2.61513E+09	1.49596E+10	4.96353E+08	2.07476E+11	2.10644E+09	**7.63471E+07**	4.90097E+08	3.02305E+10	3.83531E+09	3.13852E+09	2.65045E+09

Table 4 (continued)

Function	Item	CSO	SLPSO	EAPSO	APSO_DEE	TPCSO	DCSO	EADE	DECC_DG	DECC_RDG	DECC_MDG	PCLSO
	Sd.	8.44996E+08	6.43585E+09	4.45358E+08	5.44903E+10	1.54770E+09	**2.02805E+07**	2.51916E+08	9.40934E+09	2.11269E+09	2.19145E+09	1.78108E+09
	h	≈	+	−	+	≈	−	−	+	≈	≈	≈
F_{15}	Mean	7.56682E+07	6.64111E+07	**2.67909E+06**	3.76786E+07	5.16289E+06	2.80593E+08	3.83358E+06	7.54301E+06	8.08819E+06	7.64300E+06	2.57002E+07
	Sd.	4.68811E+06	2.37949E+06	**3.61276E+05**	7.58279E+06	4.23131E+05	2.35104E+07	5.98996E+05	9.16566E+05	9.47041E+05	7.09134E+05	5.98317E+06
	h	+	+	−	+	−	+	−	−	−	−	≈
	w/t/l	**8/5/2**	**10/5/0**	**8/4/3**	**15/0/0**	**9/4/2**	**7/6/2**	**8/2/5**	**14/0/1**	**12/1/2**	**12/2/1**	**0/15/0**
	median	**4.72**	**5.29**	**6.09**	**9.59**	**4.23**	**4.79**	**4.01**	**8.91**	**7.6**	**7.71**	**3.05**
	Friedman											

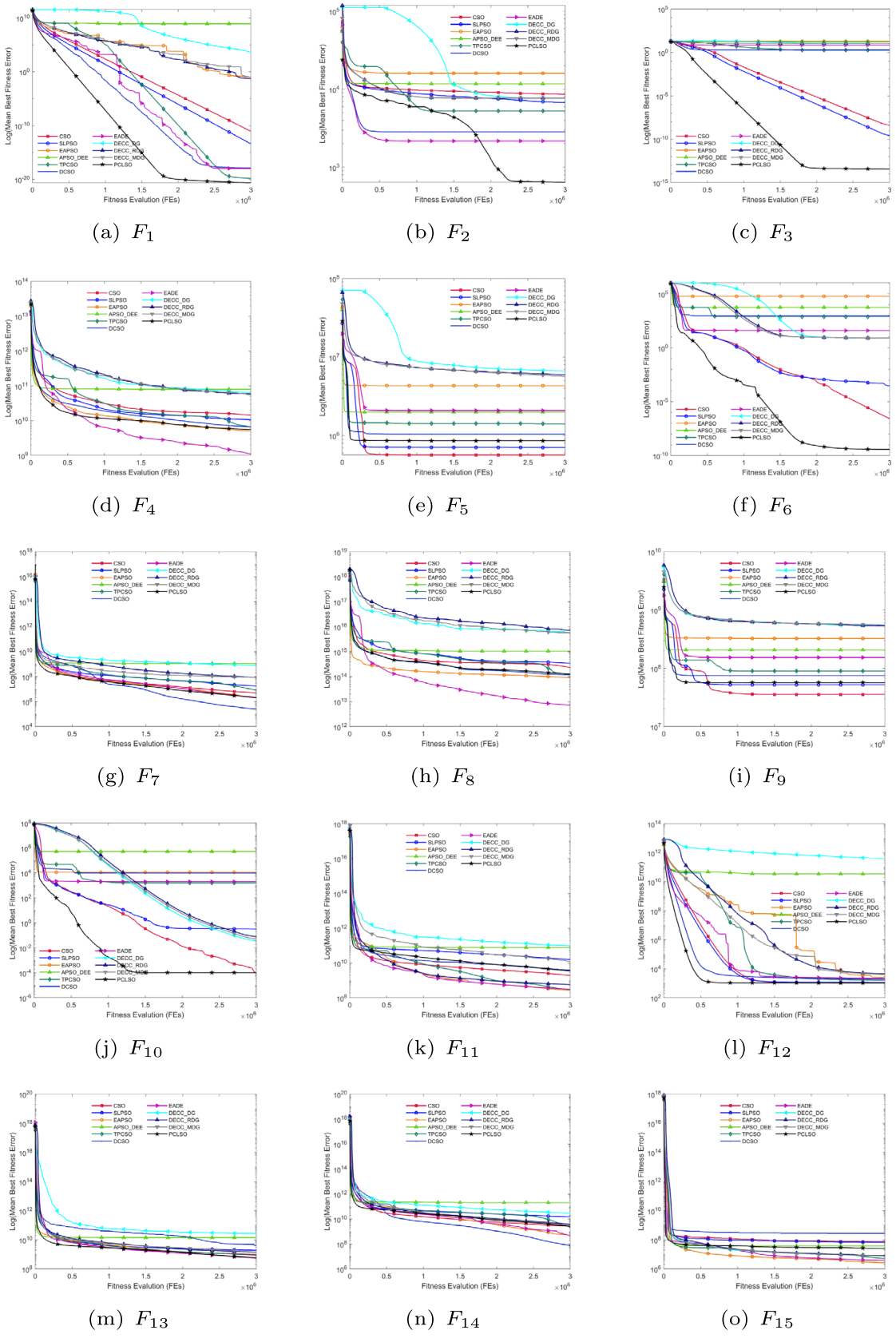
providing relative performance rankings. F-Race implements an efficient elimination mechanism based on Friedman tests, progressively discarding underperforming parameter configurations. REVAC, on the other hand, offers a comprehensive framework for parameter space exploration through variance analysis and entropy-based sampling.

In the present study, the tuning of the PCLSO algorithm was accomplished through the adoption of a parameter sensitivity analysis approach. The choice of parameter sensitivity analysis over automated tuning methods was motivated by three key considerations specific to this research context. First, careful manual examination of parameter interactions was necessitated by the unique characteristics of the optimization problem being addressed - a level of problem-specific adaptation that might not be fully captured by general-purpose tuners such as CRS-Tuning or REVAC. Second, superior interpretability was provided by the sensitivity analysis approach when compared with black-box tuning methods, with clearer insights into individual parameter effects being obtained, which proved invaluable for algorithmic understanding and debugging. Third, practical implementation constraints rendered the computational overhead of comprehensive tuning methods prohibitive for the experimental scale being considered, while sensitivity analysis was found to deliver a more computationally feasible alternative without compromising the quality of parameter optimization. These factors collectively led to the selection of this methodology while ensuring the robustness of the parameter selection process.

The parameters have a significant role in determining the efficiency of meta-heuristic algorithms [66]. Just like CSO and SLPSO, the performance of the proposed PCLSO algorithm is primarily influenced with the control parameter c. Consequently, experiments were conducted to investigate the effect of various values of c on the effectiveness of PCLSO. Throughout the experiments, the value of c was varied in the range of 0.1 to 1.

Table 5 presents the statistical results for PCLSO with the parameter c on five CEC 2013 functions (F_1, F_4, F_8, F_{12}, and F_{15}). Among these functions, F_1 is separable, F_{15} is fully inseparable, and the remaining functions are partially separable. Although these functions were chosen, they cover a wide range of testing problems, including fully separable, partially separable, and fully inseparable functions. Generally, partially separable and fully inseparable functions are challenging to tackle compared to fully separable functions. Most practical optimization problems are multi-modal and inseparable, therefore more attention should be paid to these kinds of functions.

It is clear from Table 5 that PCLSO has distinct performance with different values of the parameters. When $c = 0.1$, PCLSO is suitable for the fully separable function



(a) F_1 (b) F_2 (c) F_3

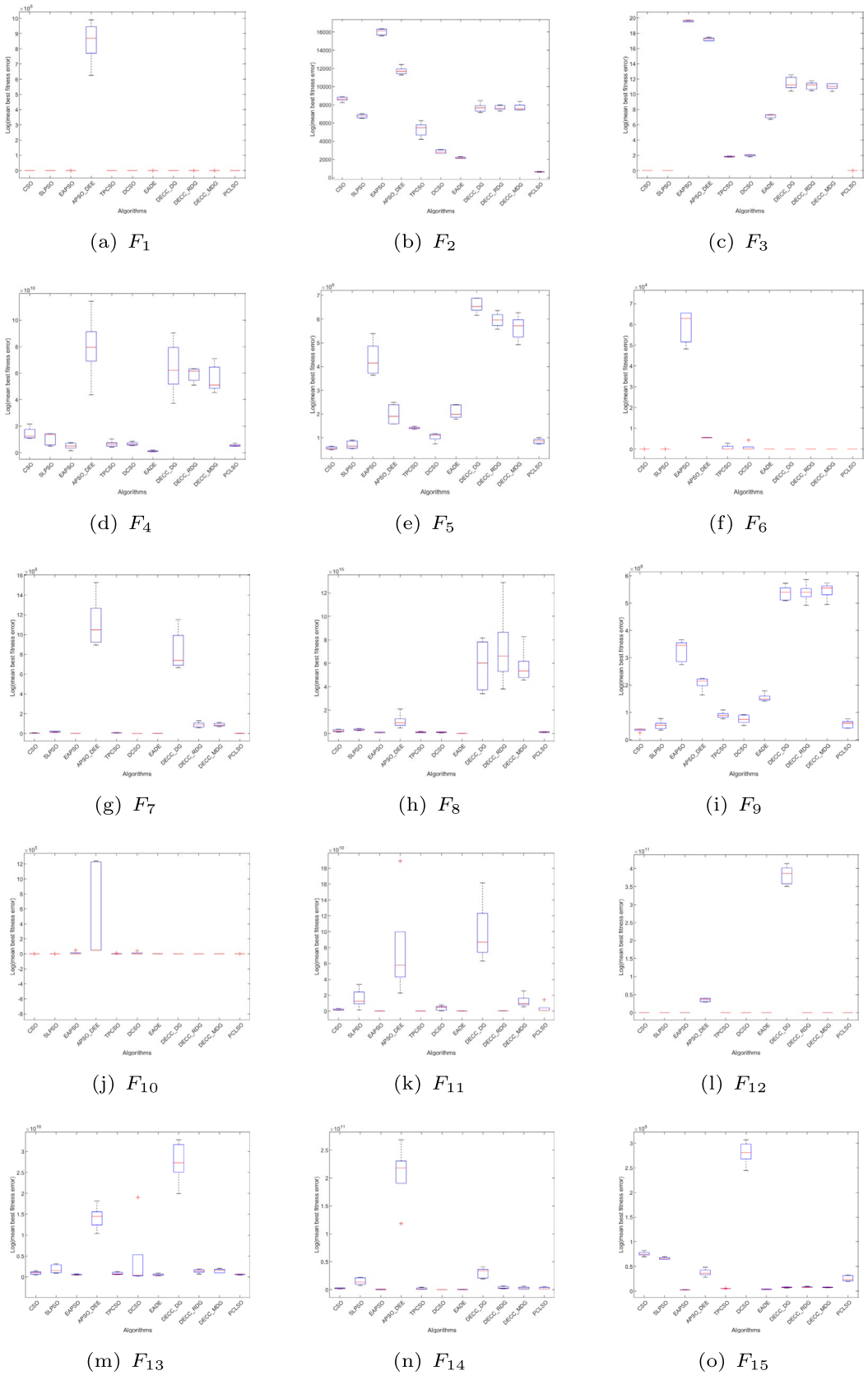
(d) F_4 (e) F_5 (f) F_6

(g) F_7 (h) F_8 (i) F_9

(j) F_{10} (k) F_{11} (l) F_{12}

(m) F_{13} (n) F_{14} (o) F_{15}

Fig. 8 Convergence curves of different algorithms obtained on the CEC'2013 benchmarks



(a) F_1 (b) F_2 (c) F_3

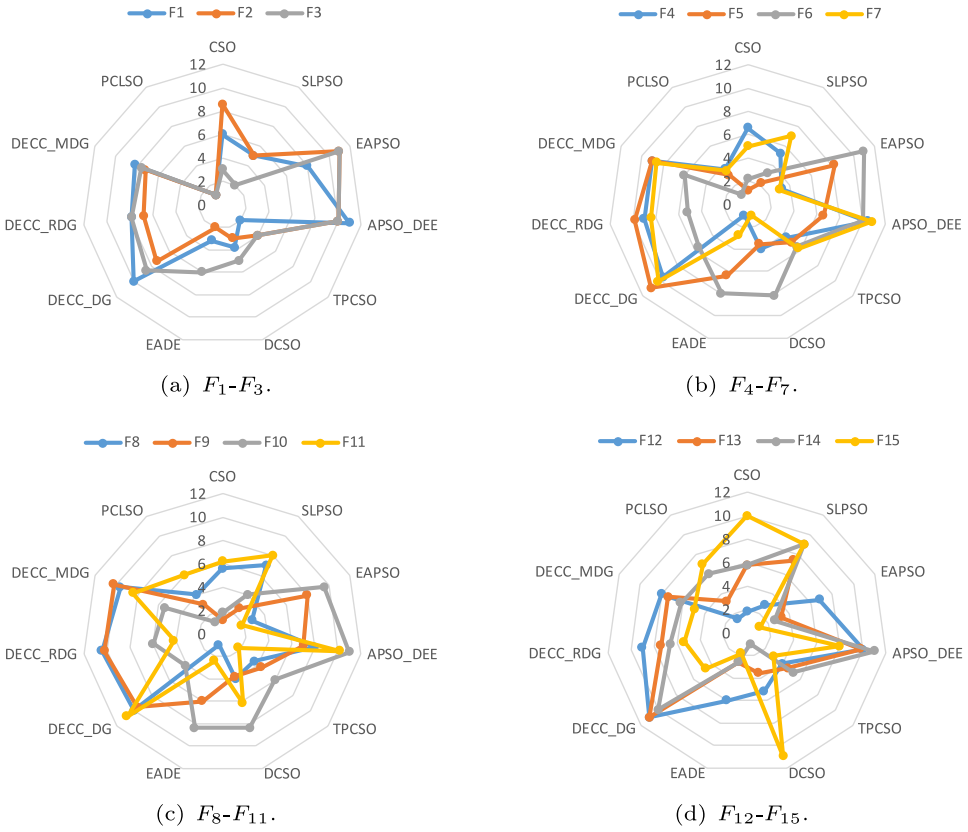
(d) F_4 (e) F_5 (f) F_6

(g) F_7 (h) F_8 (i) F_9

(j) F_{10} (k) F_{11} (l) F_{12}

(m) F_{13} (n) F_{14} (o) F_{15}

Fig. 9 Boxplot of different algorithms obtained on the CEC'2013 benchmarks



(a) F_1-F_3. (b) F_4-F_7. (c) F_8-F_{11}. (d) F_{12}-F_{15}.

Fig. 10 Radar graph based on the experimental results from Friedman test obtained by the applied algorithms on CEC 2013

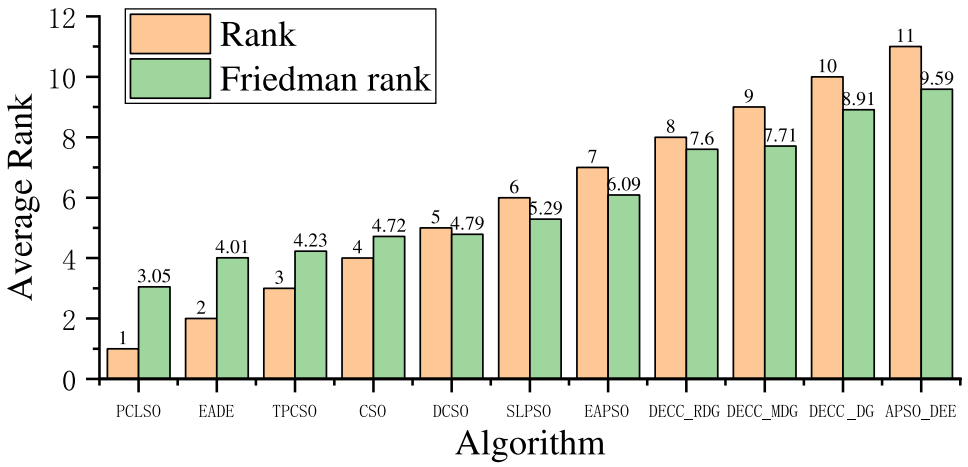


Fig. 11 Friedman rank of PCLSO and compared algorithms on CEC'2013

Table 5 PCLSO on several functions in CEC'2013 test functions of 1000D with the c varying from 0.1 to 1

Function	Item	c=0.1	c=0.3	c=0.5	c=0.7	c=1
F_1	Mean	**2.27E-21**	1.15E+06	8.27E+06	3.41E+06	3.45E+05
	Sd.	**1.37E-22**	3.77E+05	2.25E+06	1.53E+06	3.87E+05
F_4	Mean	5.51E+09	1.54E+10	1.65E+10	1.06E+10	**1.64E+09**
	Sd.	8.85E+08	2.43E+09	2.62E+09	2.70E+09	**5.75E+08**
F_8	Mean	1.30E+14	6.57E+14	5.36E+14	3.15E+14	**1.01E+14**
	Sd.	3.40E+13	2.21E+14	2.36E+14	1.47E+14	**3.13E+13**
F_{12}	Mean	**1.06E+03**	2.36E+05	5.77E+05	4.94E+05	4.20E+03
	Sd.	**5.07E+01**	2.06E+05	1.24E+05	2.11E+05	4.45E+02
F_{15}	Mean	2.57E+07	4.75E+06	5.18E+06	5.33E+06	**3.88E+06**
	Sd.	5.98E+06	2.74E+05	3.18E+05	2.03E+05	**3.03E+05**

F_1 and partially separable function F_{12}. PCLSO with $c = 1$ performs better on partially separable and fully inseparable functions (F_4, F_8, and F_{15}). However, no optimal average results are obtained for $c = 0.3$, $c = 0.5$, and $c = 0.7$. Based on the observations above, parameter c is set to the same 0.1 as SLPSO.

4.5 Effects of the Collective learning and the Pulse-stratage

In order to investigate the effectiveness of the proposed collective learning and pulse-strategy, PCLSO is compared to the original SLPSO, an algorithm not utilizing pulse-strategy (referred to as CLSO). CLSO was tested on four functions from three different groups in CEC'2013, respectively, and the outcomes are compared with those of the standard PCLSO and SLPSO, as presented in Table 6. Results that demonstrate superiority over the others at a level of significance of $\alpha = 0.05$ are highlighted in bold.

As can be seen from Table 12, the performance on four functions from all three groups are improved by the collective learning and pulse-strategy. The results demonstrate that the convergence accuracy of PCLSO is enhanced by both strategies.

Furthermore, the collective learning and pulse-strategy contribute to enhancing the algorithm's convergence speed. The curves of convergence for SLPSO, CLSO, and PCLSO in F_1, F_4, F_8, and F_{15} for all three categories of functions of CEC'2013 are illustrated in Fig. 12.

In Fig. 12, it is observable that PCLSO consistently outperforms CLSO during the convergence process. The combination of Fig. 12 and Table 6 makes it evident that PCLSO achieves not only more accurate but also faster convergence. This improvement is due to the utilization of collective learning and pulse-strategy, which accelerate the convergence process.

Table 6 Experimental results of of SLPSO, CLSO and PCLSO on selected functions

Function	Item	SLPSO	CLSO	PCLSO
F_1	Mean	3.96E-14	2.73E-14	**2.27E-21**
	Sd.	1.12E-14	5.18E-15	**1.37E-22**
	h	+	+	≈
F_4	Mean	1.07E+10	9.77E+09	**5.51E+09**
	Sd.	4.76E+09	1.88E+09	**8.85E+08**
	h	≈	+	≈
F_8	Mean	3.38E+14	2.03E+14	**1.30E+14**
	Sd.	6.95E+13	6.41E+13	**3.40E+13**
	h	+	≈	≈
F_{15}	Mean	6.64E+07	4.86E+07	**2.57E+07**
	Sd.	2.38E+06	4.01E+06	**5.98E+06**
	h	+	+	≈
	w/t/l	**4/1/0**	**4/1/0**	**0/5/0**
	median Friedman	**2.64**	**2.24**	**1.12**

4.6 Exploration and exploitation investigation for PCLSO

The balance between exploration and exploitation in metaheuristics can be assessed through various approaches, including the attraction basin-based measures [67]. While these methods provide valuable insights into search space topology by analyzing solution transitions between optima basins, they present computational challenges for large-scale problems due to requirements for heat map calculation and basin boundary identification. In this study, the dimension-wise diversity measurement was employed [68] as it offers greater computational efficiency while maintaining effectiveness for tracking search behavior. This approach calculates the average sum distance between solutions and their dimensional medians, proving particularly suitable for comparing algorithm performance across different function types. For unimodal problems, metaheuristics should emphasize exploitation for rapid convergence, while multimodal functions require greater exploration to escape local optima.

The diversity used here is described by (17).

$$div = \frac{1}{N \times D} \times \sum_{d=1}^{D} \sum_{n=1}^{N} |median\,(x_d) - x_{n,d}| \tag{17}$$

where $median(x_d)$ is the median of the dimension d of the entire population. $x_{n,d}$ represents the dimension d on the particle n. N denotes the size of the population, and D denotes the size of the problem. This calculations is performed in each iteration.

The comprehensive balance response represents the exploration and exploitation percentages employed by a given MAs. The values calculated in each iteration are given in (18)-(19).

$$exploration\% = \left(\frac{div}{div_{\max}}\right) \times 100 \tag{18}$$

$$exploitation\% = \left(\frac{|div - div_{\max}|}{div_{\max}}\right) \times 100 \tag{19}$$

where $div_{\max}$ is the maximal value of diversity obtained in the whole search.

To demonstrate that PCLSO can properly balance exploration and exploitation, a set of experiments is performed to compare PCLSO on five CEC'2013 functions in regard to the percentage of exploration and exploitation and population diversity. The graphical results are presented in Fig. 13.

Based on Fig. 13, it is evident that in case of F_1, the proposed algorithm primarily engages in exploitation behavior,

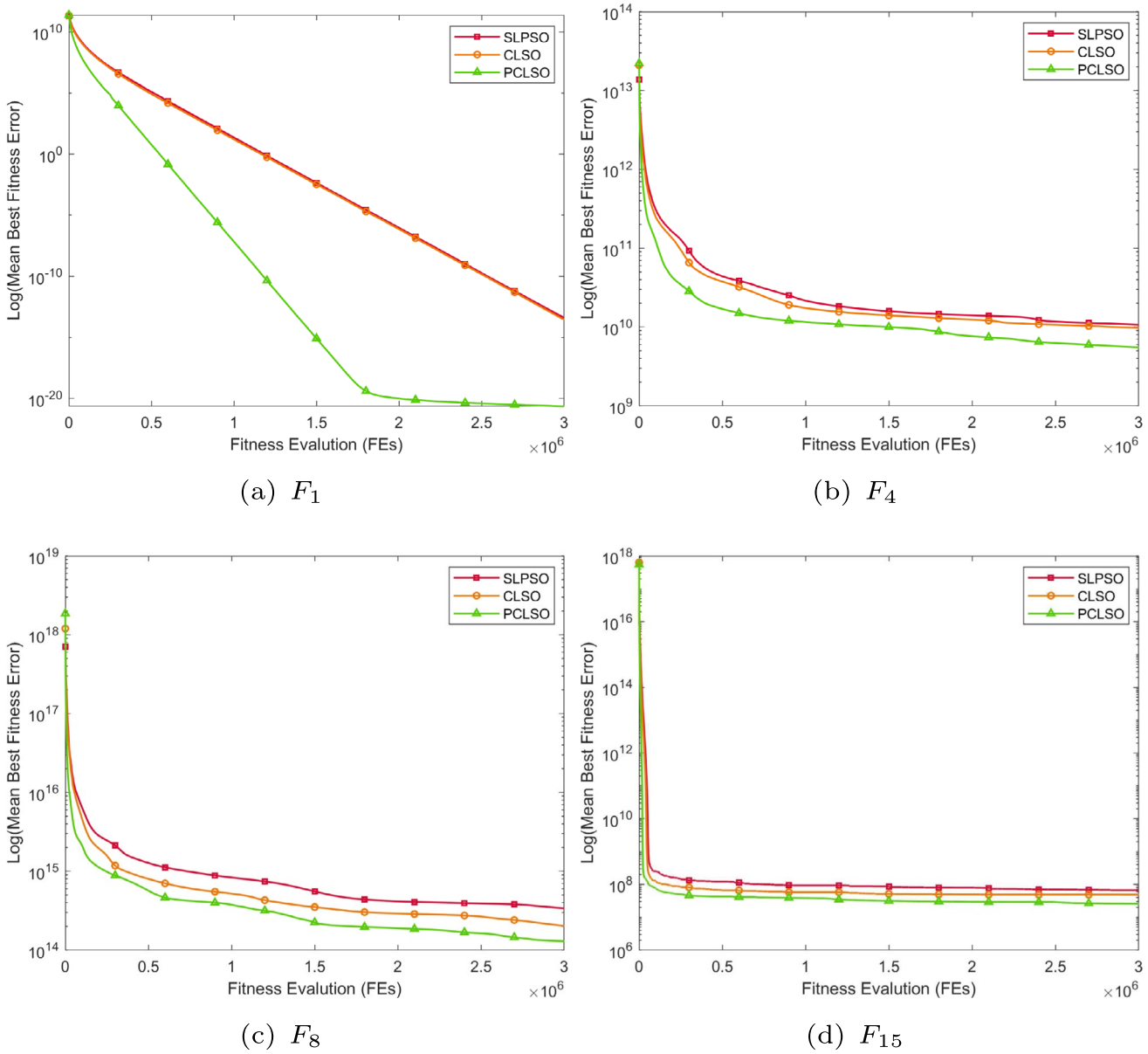


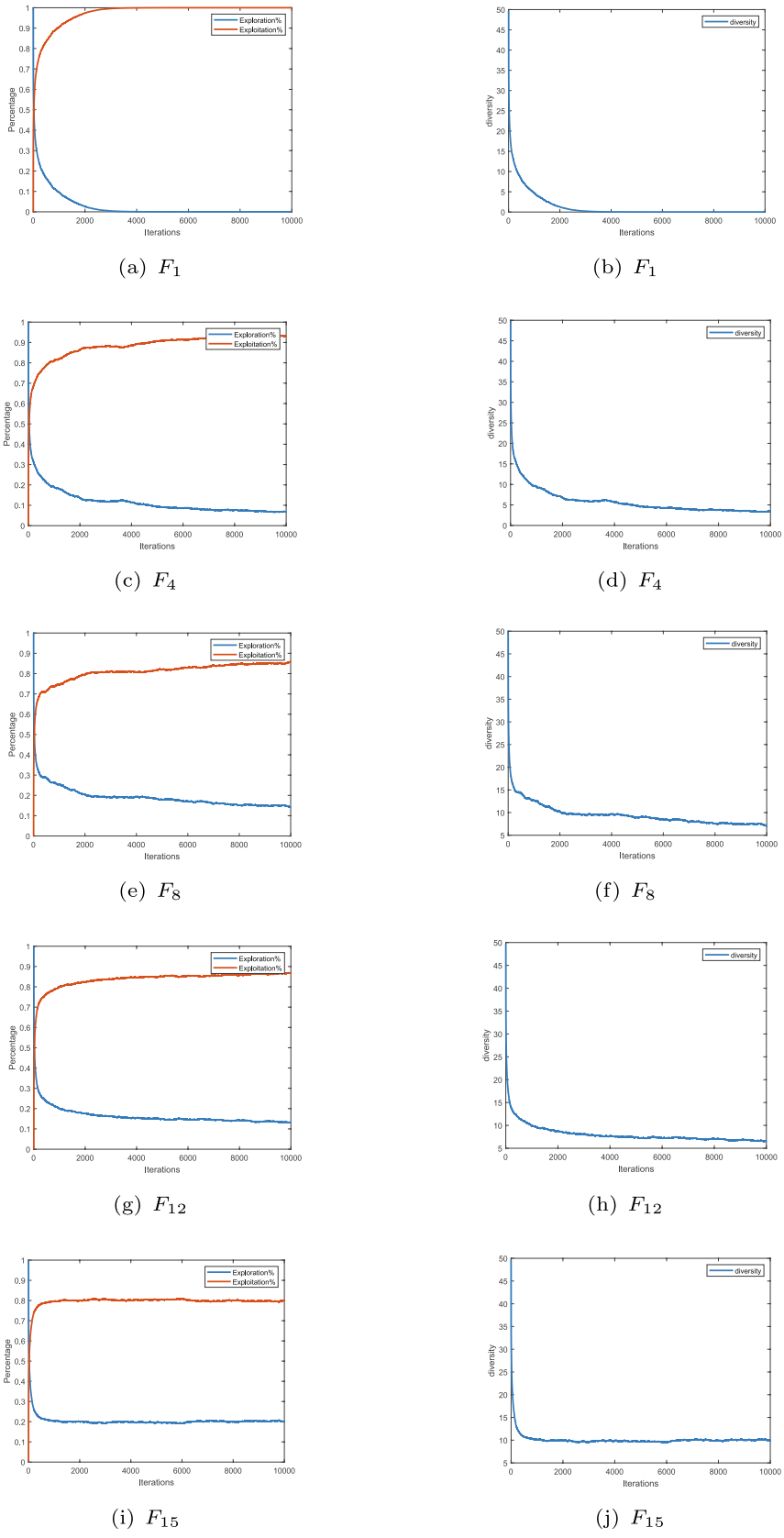
Fig. 12 Convergence process of SLPSO, CLSO and PCLSO on selected functions

which dominates the later iterations of the algorithm. In contrast, exploration behavior is not as prominent. For all types of algorithms, the exploration behavior of PCLSO diminishes rapidly in the middle iterations, possibly because of the existence of many local optima influencing the optimization behavior. When all solutions converge to local optima regions, exploitation behavior takes precedence. On F_4, F_8, F_{12}, and F_{15}, due to the complexity of these functions, even in the later stages of the algorithm, PCLSO maintains a certain level of exploration behavior. This indicates the algorithm is not suffering from stagnation in exploration. In summary, the PCLSO demonstrates a well balance of exploration and exploitation behavior.

The diversity curve also exhibits a similar pattern. For F_1, PCLSO's diversity rapidly decreases, indicating that individuals in the population are inclined towards exploitation. However, during the search process for F_4, F_8, F_{12}, and F_{15}, the diversity of the population is maintained to ensure the population's ability of escaping local optima.

5 PCLSO for multiple sequence alignment

In this section, the PCLSO algorithm is being employed to deal with Multiple Sequence Alignment (MSA). MSA [69] is a fundamental technique used to align multiple biological



(a) F_1 (b) F_1

(c) F_4 (d) F_4

(e) F_8 (f) F_8

(g) F_{12} (h) F_{12}

(i) F_{15} (j) F_{15}

Fig. 13 Exploration and exploitation investigation for PCLSO on CEC'2013

sequences and ascertain their similarities and differences. The primary approach involves arranging three or more DNA or RNA sequences and comparing the bases at each position to determine the degree of similarity among these gene fragments.

5.1 PCLSO trained HMM for MSA

In order to maximize the similarity between gene sequences, spaces may be inserted at certain positions to ensure optimal alignment. In the field of biology, multiple sequence alignment serves as a ubiquitous analytical tool, enabling the exploration of genetic variations among diverse species or individuals. It provides essential data for various research domains, including evolutionary analysis, structure prediction, functional interpretation, and other pertinent investigations. Additionally, multiple sequence alignment constitutes a significant problem in bioinformatics and represents a core aspect in the realms of genomics, systems biology, and related fields.

However, with the continuous advancement of gene sequencing technologies, the length of DNA sequences in biological genomes has been steadily increasing, now reaching several tens of thousands or even hundreds of thousands of base pairs. As a result, the problem of multiple sequence alignment (MSA) has evolved into a large-scale global optimization problem. In this section, the MSA problem will be optimized using several common large-scale global optimization algorithms.

The MSA method based on Hidden Markov Models (HMM) [70] employs a specialized topology known as Profile HMM, which is capable of capturing the similarities and differences between sequences.

Profile HMM constructs a representation by aligning all sequences to a template sequence of equal length, with each position's amino acid or nucleotide corresponding to a hidden Markov model. As a result, all sequences can be represented as matching state sequences on this template. Calculating the sequence's score on the Profile HMM enables comparison of similarity between sequences.

To evaluate the results of multiple sequence alignment, an objective function is employed for quantitative assessment. The choice of scoring criteria may vary based on the considered factors, but the most commonly used objective function relies on the Sum of Pairs (SP) scoring. This method involves aggregating scores column-wise, calculating the score for each aligned column in the sequences, and subsequently summing these scores to obtain a total score known as the Sum of Pairs Score (SPS). Such an approach provides an intuitive representation of the alignment results' quality. Apart from the SP scoring method, other objective functions are utilized to evaluate multiple sequence alignment outcomes, such as Total Column (TC) and Weighted Sum of Pairs (WSP). The SPS scoring function can be expressed as (20).

$$SPS = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} D\left(l_i, l_j\right) \tag{20}$$

where l_i and l_j represent two aligned gene sequences i and j respectively. D is a measure of the difference between the two sequences. n is the number of alignments to be performed.

The flowchart for solving the multiple sequence alignment problem using metaheuristic algorithms is illustrated in Fig. 14.

5.2 Experiment results and analysis

In this work, alignment studies were performed on four sets of biological sequences obtained from the first dataset in the BALIBASE database. The specific information for these four sets of gene sequences is presented in Table 7. The BALIBASE database provides artificially crafted high-quality reference standard sequences for gene 3D structure overlap, which are widely utilized in the field of multiple sequence alignment.

In the Table 7, the first column denotes sequence set names, while the second represents dimension of the MSA problem. The 3rd column, displays the range of sequence lengths within each group, spanning from the minimum to the maximum length. The fourth column presents the sequence identity, with higher values indicating a greater initial similarity among the gene sequences.

The experiment aimed to evaluate the efficiency of PCLSO and other compare algorithms on four sets of gene sequences. To ensure the reliability of the data, 30 independent repeated experiments were conducted, each consisting of 200 iterations. The scoring function SPS was employed as the fitness function for the algorithm. The comparison results between the algorithms on four sequences are presented as Table 8. Additionally, Fig. 15 illustrates the convergence curves during the iterations of the algorithm.

Table 8 reveals that PCLSO achieves the highest average scores on all four sequences, implying that PCLSO is better suited for training Hidden Markov Models (HMMs). The convergence curves on the four sequences are presented in Fig. 15, which indicates that PCLSO exhibits superior performance as its convergence curve oscillates upward, suggesting its ability to escape local optima.

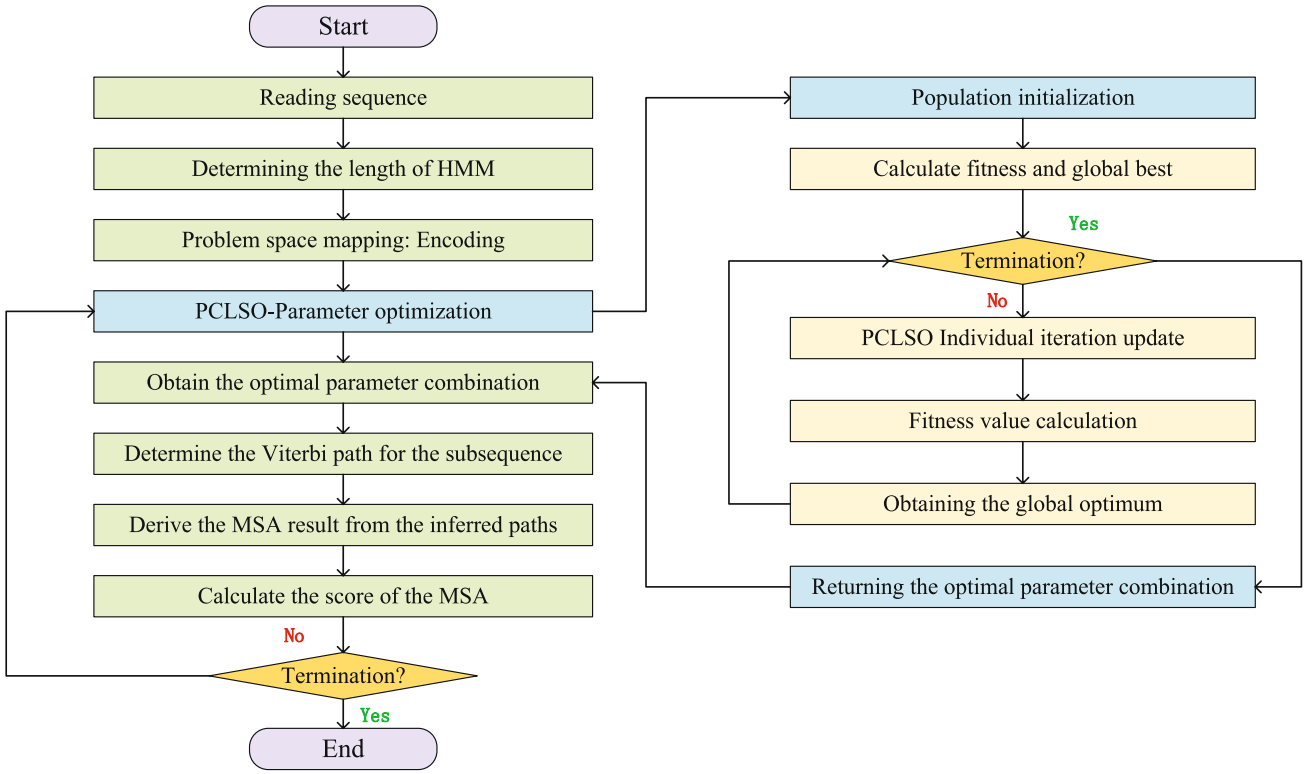


Fig. 14 PCLSO-MSA

Table 7 BAliBASE data sets used in experiments

Sequence name	NSEQ	Dimension	SEQID
kinase	5	16905	42.3
451*c*	5	5345	45.5
1*ad*2	4	13046	48.9
1*hfh*	5	8099	47.0

NSEQ = number of sequences, SEQID = percent residue identity

The results of the PCLSO sequence alignments have been visualized using sequence identity maps (Figs. 16, 17, 18, 19) in order to illustrate the degree of conservation and homology of the sequences. Due to the size of the sequences, aligned segments were extracted from fragments at the head of each sequence for visualization and analysis. The sequence logos map depicts base conservation at each position, with the accumulation of bases reflecting their conservation. The size of the graphic character for a base is proportional to its frequency at a specific position. A solitary graphic character denotes high conservation. The y-axis peak in the sequence logos reaches $\log_2 4 = 2$ bits, corresponding to the four DNA base species.

The consensus position uses symbols to convey information: the symbol! signifies a base frequency exceeding or equal to 80%, $*$ implies similarity among base species, and a blank indicates minimal homology in all sequences at that position. Figures 16, 17, 18, 19 illustrate unique sequence features in various positions, revealing pronounced consistency among specific sequences and suggesting homology among them.

6 Conclusion

In this paper, an efficient pulse-strategy collective learning swarm optimizer (PCLSO) is introduced for large-scale global optimization problems. To increase the population diversity and improve the convergence speed, the proposed

Table 8 Test Results of MSA

	kinase	451c	1ad2	1hfh
CSO	2.110000e+02	7.875000e+02	1.067000e+03	1.137000e+03
SLPSO	2.185000e+02	8.030000e+02	1.044000e+03	1.148500e+03
EAPSO	1.815000e+02	6.875000e+02	8.830000e+02	1.005000e+03
APSO_DEE	2.055000e+02	7.765000e+02	1.028000e+02	1.097500e+03
PCLSO	**2.290000e+02**	**8.040000e+02**	**1.102500e+03**	**1.155000e+03**

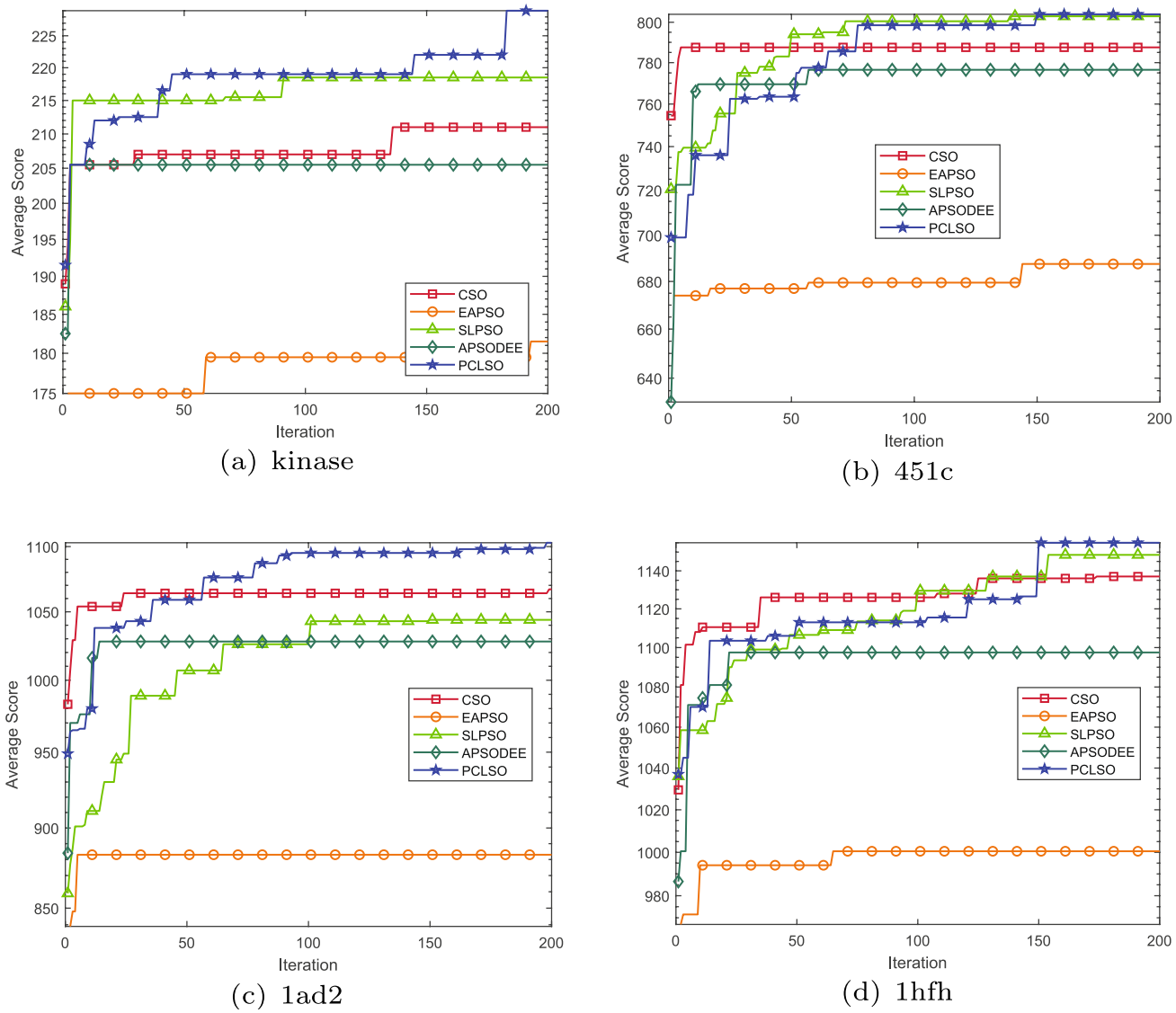


Fig. 15 The algorithm obtains an average score for the quality of the aligned sequences during HMM training

PCLSO employs collective learning for a better balance between exploration and exploitation. Furthermore, to enhance the convergence capabilities of the PCLSO algorithm, this paper introduces a pulse strategy to deal with convergence stagnation. After detecting population convergence, this strategy perturbs the global best particle to help PCLSO escape local optima. Additionally, a novel boundary control strategy is designed to further boost PCLSO's performance. To verify the effectiveness of the proposed PCLSO, it is performed against newly proposed and classical algorithms on the CEC'2010 and CEC'2013 LSGO test suites, including efficient PSO variants, DE variants, and the latest cooperative coevolutionary (CC)-based methods. Experimental results show that on most of all 35 large-scale test functions, the PCLSO algorithm outperforms the compare algorithms. Furthermore, the application of the proposed PCLSO to the gene multiple sequence alignment problem validates its competitive performance in solving real-world large-scale global problems.

According to the No Free Lunch Theorem (NFL), every MAs has its advantages and its limitations. Despite PCLSO's outstanding performance in solving LSGO problems and multiple sequence alignment problems, it also has some evident drawbacks. Like most evolutionary algorithms, the primary constraint of PCLSO is parameter tuning. In Section 4.4, the optimal value for c is determined to be 0.1. However, this is applicable to most benchmark functions but not all of them. The value of c may vary for different problems.

In future work, the focus will be on further enhancing PCLSO to improve its performance in solving LSGO problems, with the aim of achieving as close as possible to global

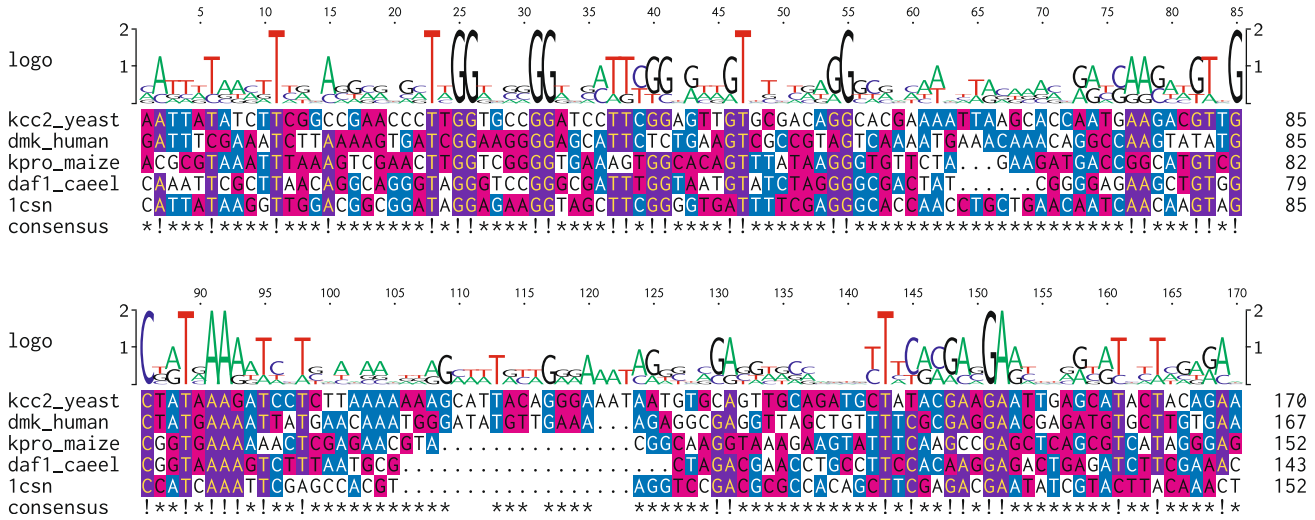


Fig. 16 Sequence of kinase

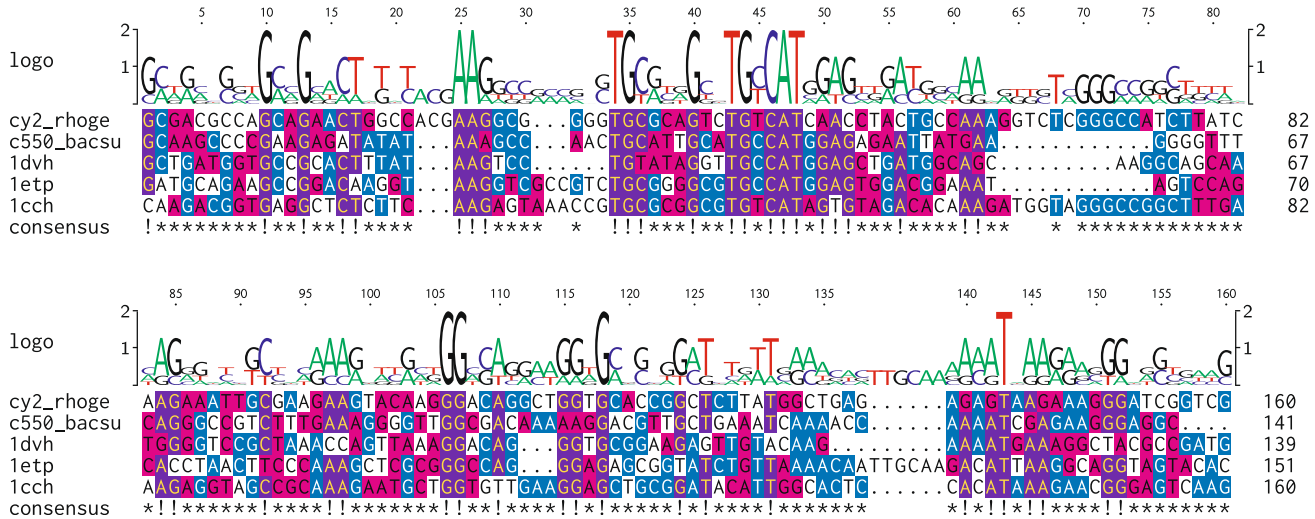


Fig. 17 Sequence of 451c

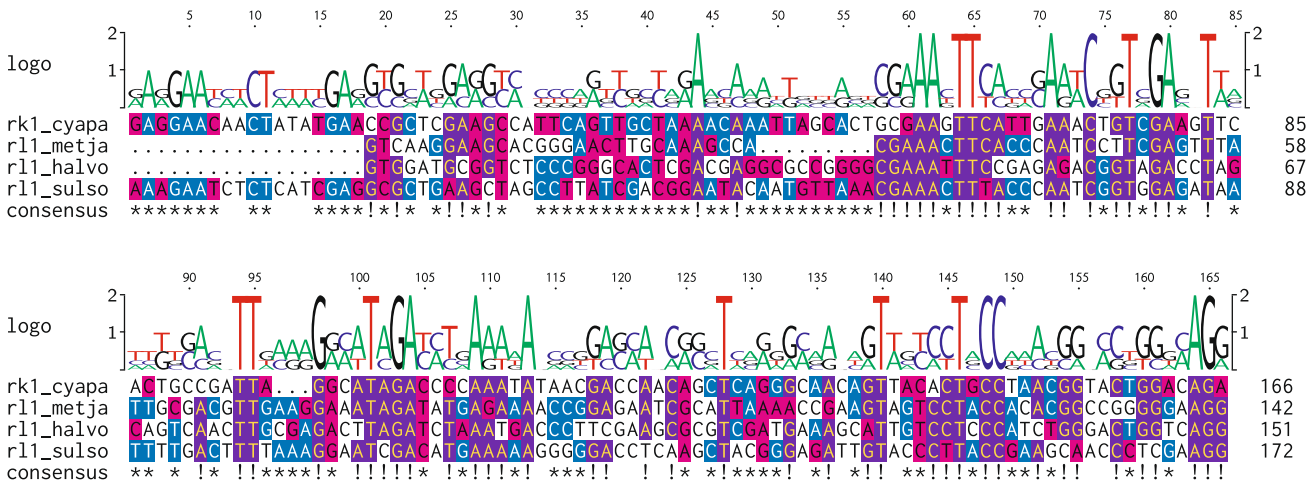


Fig. 18 Sequence of 1ad2

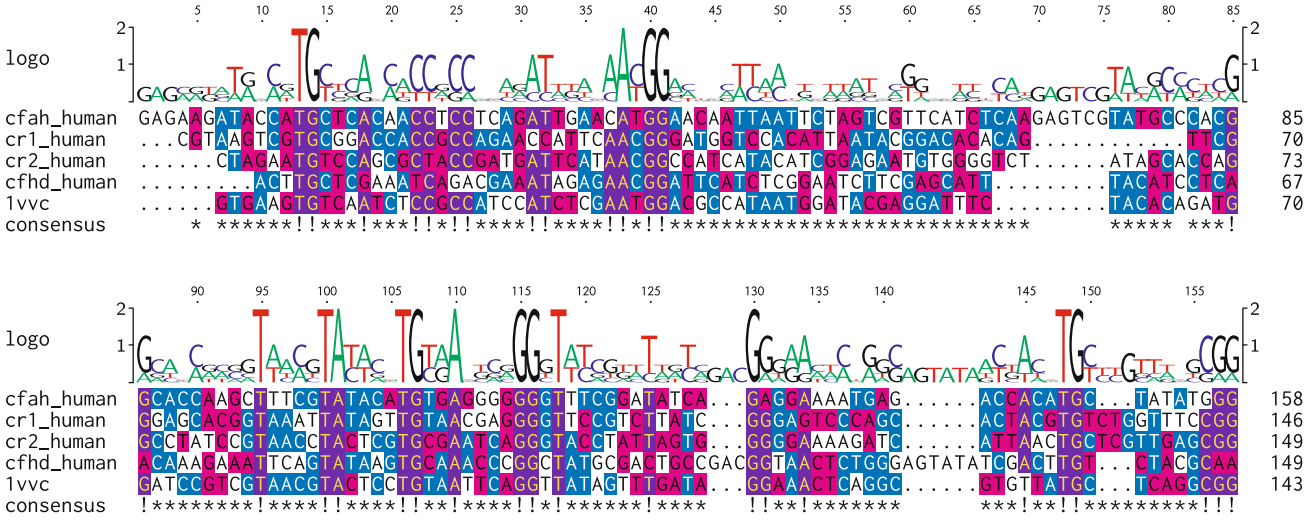


Fig. 19 Sequence of 1hfh

optimal solutions. Furthermore, the proposed algorithm will be applied to more real-world LSGO applications. In addition, future work will investigate hybrid approaches combining the sensitivity analysis with elements of CRS-Tuning's competitive evaluation and REVAC's entropy-based sampling to potentially achieve superior tuning outcomes.

Acknowledgements This work is supported by the National Natural Science Foundation of China (No. 62006144). The Major Fundamental Research Project of Shandong, China (No. ZR2019ZD03), Taishan Scholar Project of Shandong, China (No. ts20190924).

Data availability Data will be made available on request.

References

1. Wang Z-J, Zhan Z-H, Yu W-J, Lin Y, Zhang J, Gu T-L, Zhang J (2019) Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling. IEEE transactions on cybernetics 50(6):2715–2729
2. Zhao S, Zhang T, Ma S, Wang M (2023) Sea-horse optimizer: a novel nature-inspired meta-heuristic for global optimization problems. Appl Intell 53(10):11833–11860
3. Shokri-Ghaleh H, Alfi A, Ebadollahi S, Shahri AM, Ranjbaran S (2020) Unequal limit cuckoo optimization algorithm applied for optimal design of nonlinear field calibration problem of a triaxial accelerometer. Measurement 164:107963
4. Zhong C, Li G, Meng Z, He W (2023) Opposition-based learning equilibrium optimizer with levy flight and evolutionary population dynamics for high-dimensional global optimization problems. Expert Syst Appl 215:119303
5. Wang Z-J, Zhan Z-H, Kwong S, Jin H, Zhang J (2020) Adaptive granularity learning distributed particle swarm optimization for large-scale optimization. IEEE Trans Cybern 51(3):1175–1188
6. Li D, Wang L, Guo W, Zhang M, Hu B, Wu Q (2023) A particle swarm optimizer with dynamic balance of convergence and diversity for large-scale optimization. Appl Soft Comput 132:109852
7. Zhang Q, Gao H, Zhan Z-H, Li J, Zhang H (2023) Growth optimizer: A powerful metaheuristic algorithm for solving continuous and discrete global optimization problems. Knowl-Based Syst 261:110206
8. Abdel-Basset M, Mohamed R, Jameel M, Abouhawwash M (2023) Nutcracker optimizer: A novel nature-inspired metaheuristic algorithm for global optimization and engineering design problems. Knowl-Based Syst 262:110248
9. Sang-To T, Le-Minh H, Wahab MA, Thanh C-L (2023) A new metaheuristic algorithm: Shrimp and goby association search algorithm and its application for damage identification in large-scale and complex structures. Adv Eng Soft 176:103363
10. Zhang E, Nie Z, Yang Q, Wang Y, Liu D, Jeon S-W, Zhang J (2023) Heterogeneous cognitive learning particle swarm optimization for large-scale optimization problems. Inf Sci 633:321–342
11. Deng W, Shang S, Cai X, Zhao H, Zhou Y, Chen H, Deng W (2021) Quantum differential evolution with cooperative coevolution framework and hybrid mutation strategy for large scale optimization. Knowl-Based Syst 224:107080
12. Tenne Y, Goh C-K (2010) Computational intelligence in expensive optimization problems, vol 2. Springer, Berlin
13. Yang Z, Tang K, Yao X (2008) Large scale evolutionary optimization using cooperative coevolution. Inf Sci 178(15):2985–2999
14. Zhao S-Z, Suganthan PN, Das S (2011) Self-adaptive differential evolution with multi-trajectory search for large-scale optimization. Soft Comput 15:2175–2185
15. Wang Z-J, Jian J-R, Zhan Z-H, Li Y, Kwong S, Zhang J (2022) Gene targeting differential evolution: A simple and efficient method for large scale optimization. IEEE Trans Evol Comput 27(4):964–979
16. Jian J-R, Zhan Z-H, Zhang J (2020) Large-scale evolutionary optimization: A survey and experimental comparative study. Int J Mach Learn Cybern 11(3):729–745
17. Yi J-H, Xing L-N, Wang G-G, Dong J, Vasilakos AV, Alavi AH, Wang L (2020) Behavior of crossover operators in nsga-iii for large-scale optimization problems. Inf Sci 509:470–487
18. Liu R, Liu J, Li Y, Liu J (2020) A random dynamic grouping based weight optimization framework for large-scale multi-objective optimization problems. Swarm Evol Comput 55:100684
19. Komodakis N, Pesquet J-C (2015) Playing with duality: An overview of recent primal? dual approaches for solving large-scale optimization problems. IEEE Signal Process Mag 32(6):31–54
20. Yang Z, Tang K, Yao X (2008) Multilevel cooperative coevolution for large scale optimization. In: 2008 IEEE congress on

evolutionary computation (IEEE World Congress on Computational Intelligence), pp 1663–1670. IEEE
21. Chen W, Weise T, Yang Z, Tang K (2010) Large-scale global optimization using cooperative coevolution with variable interaction learning. In: Parallel problem solving from nature, PPSN XI: 11th International Conference, Kraków, Poland, September 11-15, 2010, Proceedings, Part II 11, pp 300–309. Springer
22. Omidvar M.N, Li X, Yao X (2010) Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: IEEE congress on evolutionary computation, pp 1–8. IEEE
23. Li X, Yao X (2011) Cooperatively coevolving particle swarms for large scale optimization. IEEE Trans Evol Comput 16(2):210–224
24. Omidvar MN, Li X, Mei Y, Yao X (2013) Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans Evol Comput 18(3):378–393
25. Sun Y, Kirley M, Li X (2018) Cooperative co-evolution with online optimizer selection for large-scale optimization. In: Proceedings of the genetic and evolutionary computation conference, pp 1079–1086
26. Tan B, Ma H, Mei Y, Zhang M (2020) A cooperative coevolution genetic programming hyper-heuristics approach for on-line resource allocation in container-based clouds. IEEE Trans Cloud Comput 10(3):1500–1514
27. Ma X, Huang Z, Li X, Wang L, Qi Y, Zhu Z (2022) Merged differential grouping for large-scale global optimization. IEEE Trans Evol Comput 26(6):1439–1451
28. Segura C, Coello CAC, Hernández-Díaz AG (2015) Improving the vector generation strategy of differential evolution for large-scale optimization. Inf Sci 323:106–129
29. Cheng R, Jin Y (2014) A competitive swarm optimizer for large scale optimization. IEEE Trans Cyber 45(2):191–204
30. Cheng R, Jin Y (2015) A social learning particle swarm optimization algorithm for scalable optimization. Inf Sci 291:43–60
31. Sun Y, Wang X, Chen Y, Liu Z (2018) A modified whale optimization algorithm for large-scale global optimization problems. Expert Syst Appl 114:563–577
32. Cai X, Zhang J, Liang H, Wang L, Wu Q (2019) An ensemble bat algorithm for large-scale optimization. Int J Mach Learn Cyber 10:3099–3113
33. Li D, Guo W, Lerch A, Li Y, Wang L, Wu Q (2021) An adaptive particle swarm optimizer with decoupled exploration and exploitation for large scale optimization. SwarmEvol Comput 60:100789
34. Zhang Y (2023) Elite archives-driven particle swarm optimization for large scale numerical optimization and its engineering applications. Swarm Evol Comput 76:101212
35. Huang C, Zhou X, Ran X, Liu Y, Deng W, Deng W (2023) Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem. Inf Sci 619:2–18
36. Zhang W-X, Chen W-N, Zhang J (2016) A dynamic competitive swarm optimizer based-on entropy for large scale optimization. In: 2016 Eighth international conference on advanced computational intelligence (ICACI), pp 365–371. IEEE
37. Marini F, Walczak B (2015) Particle swarm optimization (pso). a tutorial. Chem Intell Lab Syst 149:153–165
38. Liu N, Pan J-S, Chu S-C, Hu P (2023) A sinusoidal social learning swarm optimizer for large-scale optimization. Knowl-Based Syst 259:110090
39. Zhang X, Wang X, Kang Q, Cheng J (2019) Differential mutation and novel social learning particle swarm optimization algorithm. Inf Sci 480:109–129
40. Bergh F, Engelbrecht AP (2004) A cooperative approach to particle swarm optimization. IEEE Trans Evol Comput 8(3):225–239
41. Tanweer MR, Suresh S, Sundararajan N (2016) Dynamic mentoring and self-regulation based particle swarm optimization algorithm for solving complex real-world optimization problems. Inf Sci 326:1–24
42. Zhan Z-H, Li J-Y, Zhang J (2022) Evolutionary deep learning: a survey. Neurocomputing 483:42–58
43. Zhao S-Z, Liang JJ, Suganthan PN, Tasgetiren MF (2008) Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In: 2008 IEEE congress on evolutionary computation (IEEE World Congress on Computational Intelligence), pp 3845–3852. IEEE
44. Molina D, Herrera F (2015) Iterative hybridization of de with local search for the cec'2015 special session on large scale global optimization. In: 2015 IEEE congress on evolutionary computation (CEC), pp 1974–1978. IEEE
45. Tran B, Zhang M, Xue B (2016) A pso based hybrid feature selection algorithm for high-dimensional classification. In: 2016 IEEE congress on evolutionary computation (CEC), pp 3801–3808. IEEE
46. Meissner M, Schmuker M, Schneider G (2006) Optimized particle swarm optimization (opso) and its application to artificial neural network training. BMC Bioinformatics 7:1–11
47. Gad AG (2022) Particle swarm optimization algorithm and its applications: a systematic review. Archives Comput Methods Eng 29(5):2531–2561
48. Ge H, Sun L, Tan G, Chen Z, Chen CP (2017) Cooperative hierarchical pso with two stage variable interaction reconstruction for large scale optimization. IEEE Trans Cybern 47(9):2809–2823
49. Yang Q, Chen W-N, Da Deng J, Li Y, Gu T, Zhang J (2017) A level-based learning swarm optimizer for large-scale optimization. IEEE Trans Evol Comput 22(4):578–594
50. Wang F, Wang X, Sun S (2022) A reinforcement learning level-based particle swarm optimization algorithm for large-scale optimization. Inf Sci 602:298–312
51. Mohapatra P, Das KN, Roy S (2017) A modified competitive swarm optimizer for large scale optimization problems. Appl Soft Comput 59:340–362
52. Mousavirad SJ, Rahnamayan S (2020) Cenpso: a novel center-based particle swarm optimization algorithm for large-scale optimization. In: 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp 2066–2071. IEEE
53. Wang Z-J, Yang Q, Zhang Y-H, Chen S-H, Wang Y-G (2023) Superiority combination learning distributed particle swarm optimization for large-scale optimization. Appl Soft Comput 136:110101
54. Jian J-R, Chen Z-G, Zhan Z-H, Zhang J (2021) Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization. IEEE Trans Evol Comput 25(4):779–793
55. Lan R, Zhu Y, Lu H, Liu Z, Luo X (2020) A two-phase learning-based swarm optimizer for large-scale optimization. IEEE Trans Cybern 51(12):6284–6293
56. Blinder AS, Morgan J (2000) Are two heads better than one?: An experimental analysis of group vs. individual decisionmaking. National Bureau of Economic Research Cambridge, Mass, USA
57. Peng ZK, Zhang SX, Zheng SY, Long YL (2019) Collective information-based teaching-learning-based optimization for global optimization. Soft Comput 23:11851–11866
58. Chen X, Li K (2022) Collective information-based particle swarm optimization for multi-fuel chp economic dispatch problem. Knowl-Based Syst 248:108902
59. Zheng LM, Zhang SX, Tang KS, Zheng SY (2017) Differential evolution powered by collective information. Inf Sci 399:13–29
60. Yan S, Yang P, Zhu D, Zheng W, Wu F et al (2021) Improved sparrow search algorithm based on iterative local search. Comput Intell Neurosci, 2021
61. Tang K, Yáo X, Suganthan PN, MacNish C, Chen Y-P, Chen C-M, Yang Z (2007) Benchmark functions for the cec'2010

special session and competition on large-scale global optimization. Nature inspired computation and applications laboratory, USTC, China 24:1–18
62. Li X, Tang K, Omidvar MN, Yang Z, Qin K, China H (2013) Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Gene 7(33):8
63. Mohamed AW (2017) Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm. Complex & Intelligent Systems 3:205–231
64. Sun Y, Kirley M, Halgamuge SK (2017) A recursive decomposition method for large scale continuous optimization. IEEE Trans Evol Comput 22(5):647–661
65. Veček N, Mernik M, Filipič B, Črepinšek M (2016) Parameter tuning with chess rating system (crs-tuning) for meta-heuristic algorithms. Inf Sci 372:446–469. https://doi.org/10.1016/j.ins.2016.08.066
66. Pan J-S, Liu N, Chu S-C, Lai T (2021) An efficient surrogate-assisted hybrid optimization algorithm for expensive optimization problems. Inf Sci 561:304–325
67. Jerebic J, Mernik M, Liu S-H, Ravber M, Baketarić M, Mernik L, Črepinšek M (2021) A novel direct measure of exploration and exploitation based on attraction basins. Expert Syst Appl 167:114353
68. Morales-Castañeda B, Zaldívar D, Cuevas E, Fausto F, Rodríguez A (2020) A better balance in metaheuristic algorithms: Does it exist? Swarm Evol Comput 54:100671. https://doi.org/10.1016/j.swevo.2020.100671
69. Sun J, Wu X, Fang W, Ding Y, Long H, Xu W (2012) Multiple sequence alignment using the hidden markov model trained by an improved quantum-behaved particle swarm optimization. Inf Sci 182(1):93–114
70. Mejía SN (2024) Hidden markov models for early detection of cardiovascular diseases. Ingenier 20(1):1–31