Improved Competitive Swarm Optimizer with Linear Population Reduction for Large-scale Optimization

Rui Zhong
Information Initiative Center
Hokkaido University
Sapporo, Japan
zhongrui@iic.hokudai.ac.jp

Jun Yu*
Institute of Science and Technology
Niigata University
Niigata, Japan
yujun@ie.niigata-u.ac.jp

Xingbang Du
Graduate School of Information Science and Technology
Hokkaido University
Sapporo, Japan
xingbang.du.d1@elms.hokudai.ac.jp

Enzhi Zhang
Graduate School of Information Science and Technology
Hokkaido University
Sapporo, Japan
enzhi.zhang.n6@elms.hokudai.ac.jp

Abdelazim G. Hussien
Department of Computer and Information Science
Linköping University
Linköping, Sweden
Faculty of Science
Fayoum University
Fayoum, Egypt
abdelazim.hussien@liu.se

2025 IEEE Congress on Evolutionary Computation (CEC) | 979-8-3315-3431-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/CEC65147.2025.11042946

***Abstract*—Competitive swarm optimizer (CSO) is an efficient and effective swarm intelligence approach, especially for large-scale optimization. This paper presents an enhanced version of CSO termed improved CSO with linear population reduction (L-ICSO). The novel triple-individuals competitive mechanism is introduced to strengthen the optimization performance of L-ICSO, and the linear population reduction mechanism from L-SHADE is integrated into L-ICSO to highlight the explorative search in the initial phase of optimization and emphasize the exploitative behavior in the late phase. We conduct comprehensive numerical experiments in 100-dimensional CEC2017 benchmark functions. Ten state-of-the-art optimizers such as L-SHADE, jSO, L-SHADE-cnEpSin, and the original CSO are employed as competitor algorithms. The Mann–Whitney U and Holm multiple comparison tests are used to measure the statistical significance between L-ICSO and competitor algorithms. The experimental results and statistical analysis confirm the efficiency and effectiveness of our proposed L-ICSO in addressing large-scale optimization problems. The source code of L-ICSO can be found at https://github.com/RuiZhong961230/L-ICSO.**

***Index Terms*—Competitor swarm optimizer (CSO), Triple-individuals competitive mechanism, Linear population reduction**

I. Introduction

Particle swarm optimization (PSO), as an efficient and flexible population-based and stochastic optimization technique, has been widely applied to robotic arm trajectory planning [1], solar photovoltaic parameter estimation [2], and other real-world applications [3]. A classic framework of PSO can be formulated using Eq. (1).

$$\begin{aligned} \boldsymbol{v}_i^{t+1} &= \omega \boldsymbol{v}_i^t + c_1 \boldsymbol{r}_1 \cdot (\boldsymbol{pb}_i^t - \boldsymbol{x}_i^t) + c_2 \boldsymbol{r}_2 \cdot (\boldsymbol{gb}^t - \boldsymbol{x}_i^t) \\ \boldsymbol{x}_i^{t+1} &= \boldsymbol{x}_i^t + \boldsymbol{v}_i^{t+1} \end{aligned} \tag{1}$$

where t denotes the iteration, i is the index of the particle swarm, and $\boldsymbol{v}$ and $\boldsymbol{x}$ are vectors representing the velocity and position of the particle individuals, respectively. ω is the inertia weight, c_1 and c_2 are acceleration coefficients, $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are two random vectors, where elements in $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are in [0, 1]. $\boldsymbol{pb}_i^t$ is the current best solution of $\boldsymbol{x}_i$ indicating the cognitive coefficient, and $\boldsymbol{gb}^t$ is the best solution of the swarm so far representing the social coefficient. However, the performance of PSO is significantly influenced by the choice of search operators and parameters, and research on PSO has been ongoing for over 25 years [4].

Among numerous variants of particle swarm optimization (PSO), the competitive swarm optimizer (CSO) stands out for its exceptional capability and adaptability in high-dimensional search spaces [5]. The unique competitive mechanism in CSO effectively balances exploitation and exploration and significantly accelerates optimization convergence. Meanwhile, many scholars and researchers have identified some limitations of CSO and have proposed improved versions. Mohapatra et al. [6] proposed a modified CSO (MCSO) in which two-thirds of the population is updated in each iteration using a tri-competitive operator. This small modification aims to maintain a higher proportion of exploration and achieve faster convergence, which significantly improves the performance of the original CSO. Xiong et al. [7] recognized that superior individuals might inadvertently mislead the optimization di-

* Dr. Jun Yu is the corresponding author.

rection in deceptive fitness landscapes and proposed an orthogonal learning CSO (OLCSO) for addressing power system economic dispatch problems. The search strategies in CSO can identify the knowledge from the winner and loser particle individuals, and the orthogonal learning scheme constructs promising solutions to ensure the convergence acceleration for OLCSO. Huang et al. [8] presented an efficient three-phase competitive swarm optimizer (TPCSO). In TPCSO, population diversity is emphasized in the first phase to explore unknown areas. In the second phase, effective search operators are introduced to two sub-swarms to exploit promising areas. In the third phase, individuals focus on accelerating convergence by learning from the best individual. The integration of a novel three-phase cooperative scheme strengthens the performance of the original CSO in large-scale complex optimization problems. Pan et al. [9] integrated surrogate-assisted techniques into CSO and proposed a surrogate-assisted CSO (SACSO) to deal with expensive optimization problems. In SACSO, the global, local, and opposition-based search concepts are three different principles for selecting model-construction particle individuals. The utilization of the generalized surrogate model (GSM) and the elite surrogate model (ESM) effectively enhances the performance of SACSO. The detailed development history of CSO can be referred to [10].

This paper presents an improved CSO with linear population reduction (L-ICSO) for large-scale optimization. The motivation is integrating the population resizing scheme from success-history adaptive differential evolution with linear population reduction (L-SHADE) [11] to CSO. This simple yet efficient principle is expected to switch the balance between exploration and exploitation automatically and intelligently. Additionally, we introduce a triple-individuals competitive mechanism to replace the double-individuals competitive mechanism in the original CSO. This modification utilizes more knowledge from particle swarm to construct offspring individuals, which is expected to significantly improve the performance of L-ICSO. We conduct comprehensive numerical experiments in 100-dimensional CEC2017, and state-of-the-art optimizers, such as CSO, L-SHADE, and L-SHADE-cnEpSin, are employed as competitor algorithms. The experimental results and rigorous statistical analysis confirm the effectiveness and efficiency of our proposed L-ICSO.

The remainder of this paper is organized as follows: Section II introduces the original CSO. Section III presents our proposed L-ICSO in detail. The numerical experiments are conducted in Section IV. Finally, Section V concludes this paper.

II. Competitive swarm optimizer (CSO)

CSO retains the main structure of PSO, where each particle individual exhibits attributes of location and velocity. The key difference lies in the integration of a competitive mechanism in the iterative updating stage. Simply, CSO randomly selects a particle individual $\boldsymbol{x}_{r1}^t$ and its competitor $\boldsymbol{x}_{r2}^t$. The particle individual with better fitness is denoted as $\boldsymbol{x}_w^t$ and directly survives to the next iteration, and the particle individual with worse fitness is denoted as $\boldsymbol{x}_l^t$ and is updated using Eq. (2).

$$\begin{aligned} \boldsymbol{v}_l^{t+1} &= \boldsymbol{r}_1 \boldsymbol{v}_l^t + \boldsymbol{r}_2 \cdot (\boldsymbol{x}_w^t - \boldsymbol{x}_l^t) + \phi \cdot \boldsymbol{r}_3 \cdot (\boldsymbol{x}_{mean}^t - \boldsymbol{x}_l^t) \\ \boldsymbol{x}_l^{t+1} &= \boldsymbol{x}_l^t + \boldsymbol{v}_l^{t+1} \end{aligned} \tag{2}$$

where $\boldsymbol{x}_{mean}^t$ is the centroid of the particle swarm and ϕ is a control parameter fixed at (0, 0.3] as suggested in [5]. The knowledge information from the superior particle individual $\boldsymbol{x}_w^t$ will transfer to the next iteration, and the inferior particle individual $\boldsymbol{x}_l^t$ learns from the $\boldsymbol{x}_w^t$ to construct offspring individuals. This simple yet effective motivation is the foundation of CSO. In summary, the pseudocode of CSO for minimization problems is presented in Algorithm 1.

Algorithm 1 CSO

Require: Population size:N, Dimension:D, Max. iteration:T
Ensure: Optimum: $\boldsymbol{x}_{best}^t$

```
Initialize the particle swarm X
t = 0
x_best^t ← best(X)
while t < T do
    P ← copy(X)
    while P is not empty do
        Randomly pop two particle individual x_r1^t and x_r2^t
        if f(x_r1^t) ≤ f(x_r2^t) then
            x_w^t = x_r1^t, x_l^t = x_r2^t,
        else
            x_w^t = x_r2^t, x_l^t = x_r1^t,
        end if
        Survive x_w^t to X^{t+1}
        Update x_l^t using Eq. (2)
        Save x_l^{t+1} to X^{t+1}
    end while
    x_best^t ← best(X)
    t ← t + 1
end while
return x_best^t
```

III. Our proposal: L-ICSO

This section provides a detailed introduction to our proposed L-ICSO. First, we present the flowchart of L-ICSO in Fig. 1, with the main modifications highlighted in red. L-ICSO inherits the primary architecture of CSO and incorporates a triple-individuals competitive mechanism and a linear population reduction scheme to enhance optimization performance. Subsequently, we will explain these two strategies in the following contexts.

A. Triple-individuals competition

Motivated by the double-individuals competition in the original CSO, we developed a triple-individuals competition scheme to simulate the competitive behaviors among particle individuals with a more complex mechanism. Simply, L-ICSO randomly pops three particle individuals and denotes them as

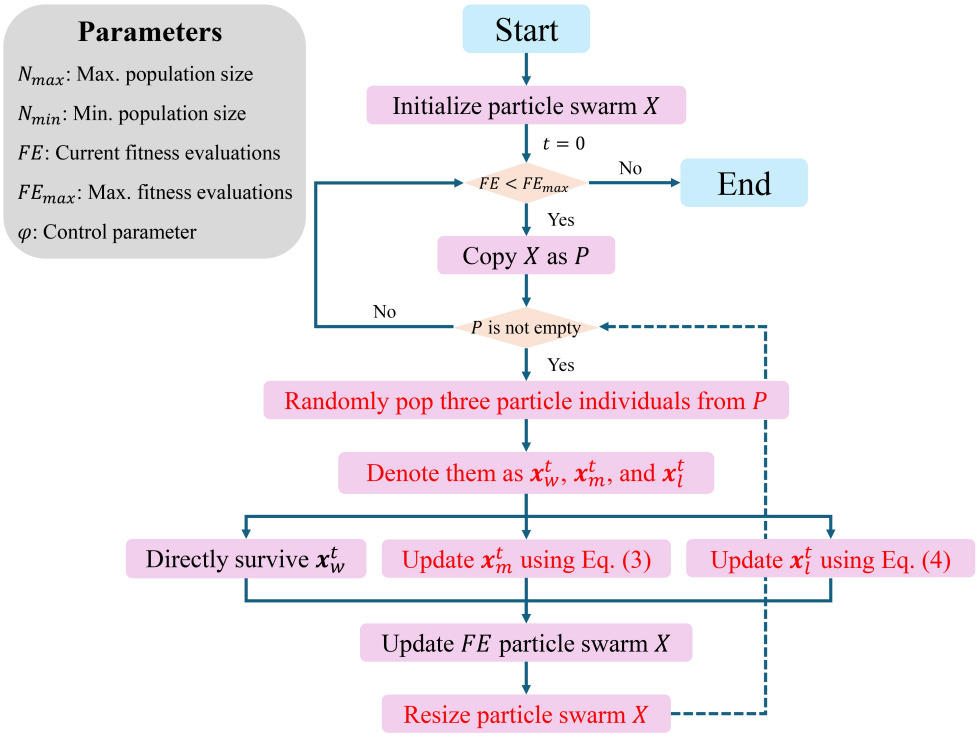

Fig. 1: The flowchart of L-ICSO.

the winner $\boldsymbol{x}_w^t$, the middle $\boldsymbol{x}_m^t$, and the loser $\boldsymbol{x}_l^t$ based on the fitness value. While the winner $\boldsymbol{x}_w^t$ directly survives to the next iteration, the middle $\boldsymbol{x}_m^t$ and the loser $\boldsymbol{x}_l^t$ are updated using Eqs. (3) and (4), respectively.

$$\begin{aligned}
&\text{If } \textbf{rand}(0,1) < 0.5 \\
&\quad \boldsymbol{v}_m^{t+1} = \boldsymbol{r}_1 \boldsymbol{v}_m^t + \boldsymbol{r}_2 \cdot (\boldsymbol{x}_w^t - \boldsymbol{x}_m^t) + \phi \cdot \boldsymbol{r}_3 \cdot (\boldsymbol{x}_{mean}^t - \boldsymbol{x}_m^t) \\
&\quad \boldsymbol{x}_m^{t+1} = \boldsymbol{x}_m^t + \boldsymbol{v}_l^{t+1} \\
&\text{Else} \\
&\quad \boldsymbol{x}_m^{t+1} = \boldsymbol{x}_m^t
\end{aligned} \tag{3}$$

where $\boldsymbol{x}_m^t$ has a 50% chance of surviving to the next iteration directly and a 50% chance of being updated. The term $\boldsymbol{r}_2 \cdot (\boldsymbol{x}_w^t - \boldsymbol{x}_m^t)$ moves $\boldsymbol{x}_m^t$ towards the winner $\boldsymbol{x}_w^t$ and enhance the exploitative search capacity. Meanwhile, the term $\phi \cdot \boldsymbol{r}_3 \cdot (\boldsymbol{x}_{mean}^t - \boldsymbol{x}_m^t)$ moves $\boldsymbol{x}_m^t$ towards the centroid $\boldsymbol{x}_{mean}^t$ and facilitate the exploration of unknown areas.

$$\begin{aligned}
\boldsymbol{v}_l^{t+1} &= \boldsymbol{r}_1 \boldsymbol{v}_l^t + \boldsymbol{r}_2 \cdot (\boldsymbol{x}_w^t - \boldsymbol{x}_l^t) + \phi \cdot \boldsymbol{r}_3 \cdot (\boldsymbol{x}_{best}^t - \boldsymbol{x}_l^t) \\
\boldsymbol{x}_l^{t+1} &= \boldsymbol{x}_l^t + \boldsymbol{v}_l^{t+1}
\end{aligned} \tag{4}$$

where $\boldsymbol{x}_{best}^t$ is the best particle swarm found so far. In this search operator, both terms of $\boldsymbol{r}_2 \cdot (\boldsymbol{x}_w^t - \boldsymbol{x}_l^t)$ and $\phi \cdot \boldsymbol{r}_3 \cdot (\boldsymbol{x}_{best}^t - \boldsymbol{x}_l^t)$ aim to drive the loser $\boldsymbol{x}_l^t$ towards promising area and accelerate the optimization convergence. A demonstration of the proposed triple-individuals competitive mechanism is presented in Fig. 2. The proposed triple-individuals competitive mechanism utilizes extensive information from multiple individuals during the offspring construction process. By integrating three candidates - the winner $\boldsymbol{x}_w^t$, the middle $\boldsymbol{x}_m^t$, and the loser $\boldsymbol{x}_l^t$ - the mechanism enhances genetic diversity, fosters a more comprehensive information inheritance strategy, and improves the balance between exploration and exploitation.

B. Linear population reduction

L-ICSO incorporates the linear population reduction scheme from L-SHADE using Eq. (5).

$$N^t = \textbf{round}\left(\frac{N_{min} - N_{max}}{FE_{max}} \cdot FE + N_{max}\right) \tag{5}$$

N_{max} and N_{min} represent the maximum and minimum population sizes, respectively. FE_{max} and FE denote the maximum and current fitness evaluations, respectively. Given that three particle individuals will participate in the competition, the N_{max} is fixed at 600, and N_{min} is set to 6. During the initial optimization stage, a large population size allows L-ICSO to explore the search domain more thoroughly and reduce the probability of becoming trapped in local optima. In the later stage, a smaller population size enhances the exploitation capacity of L-ICSO and guides the algorithm to better optimal solutions.

In summary, the pseudocode of L-ICSO is presented in Algorithm 2.

IV. Numerical Experiments

Numerical experiments are conducted in this section to investigate the performance of L-ICSO comprehensively. Sec-

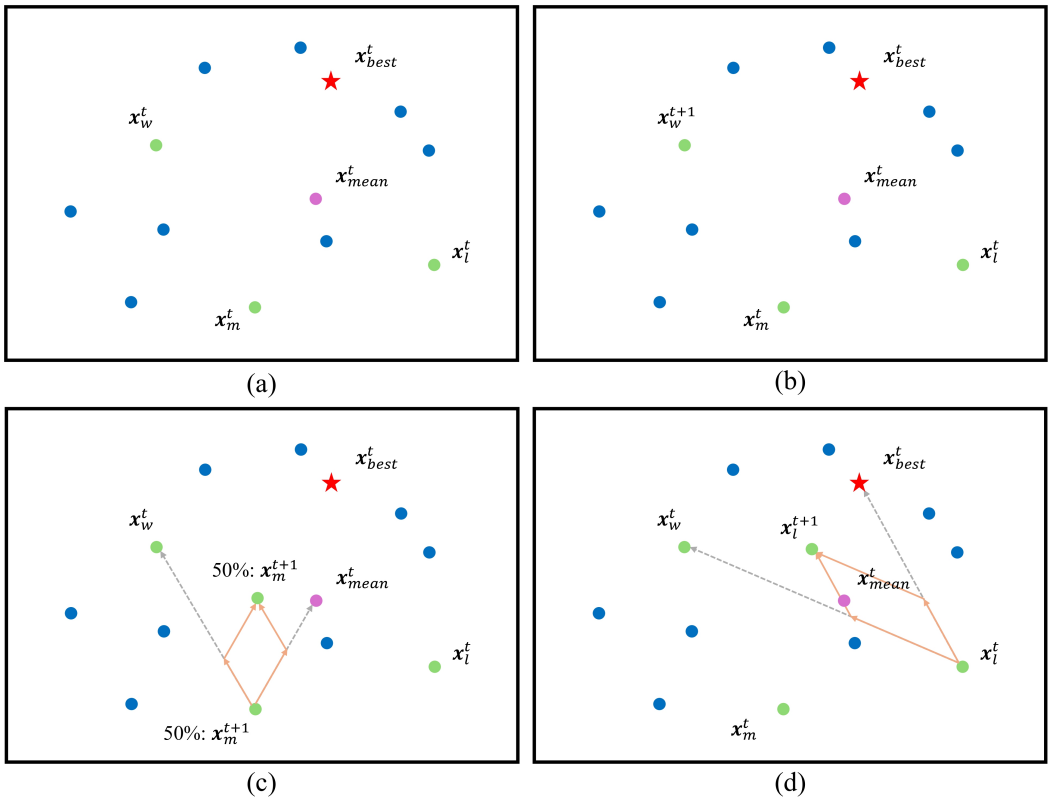

Fig. 2: A demonstration of the triple-individuals competitive mechanism ($\boldsymbol{v}_m^t$ and $\boldsymbol{v}_m^t$ are assumed as zero vectors). (a) The distribution of particle swarms. (b) $\boldsymbol{x}_w^t$ survives to the next iteration. (c) $\boldsymbol{x}_m^t$ survives or updates with equal probability using Eq. (3). (d) $\boldsymbol{x}_l^t$ updates using Eq. (4).

Algorithm 2 L-ICSO

Require: Max. and Min. population size: N_{max} and N_{min}, Dimension: D, Max. fitness evaluations: FE_{max}
Ensure: Optimum: $\boldsymbol{x}_{best}^t$

```
Initialize the particle swarm X
FE = 0
x_best^t ← best(X)
while FE < FE_max do
    P ← copy(X)
    while P is not empty do
        Randomly pop three particle individual x_r1^t, x_r2^t, and x_r3^t
        Denote them as x_w^t, x_m^t, and x_l^t based on fitness
        Survive x_w^t to X^{t+1}
        Update x_m^t using Eq. (3)
        Update x_l^t using Eq. (4)
        Save x_m^{t+1} and x_l^{t+1} to X^{t+1}
    end while
    x_best^t ← best(X)
    Update FE
    Resize the particle swarm X
end while
return x_best^t
```

tion IV-A summarizes the experimental settings and Section IV-B presents the experimental results and statistical analysis.

A. Experimental settings

1) Experimental environments and implementations: We conducted numerical experiments on a Lenovo Legion R9000P equipped with an AMD R9 7945HX CPU, 16GB DDR5 RAM, and an NVIDIA GeForce RTX 4060 GPU. The operating system is Windows 10. These detailed experimental conditions ensure the reproducibility of our research.

2) Benchmark and competitor algorithms: 100-D CEC2017 functions are employed as the benchmark to evaluate the performance of L-ICSO in high-dimensional space. This benchmark contains 29 functions that f_1 is unimodal, f_3 to f_9 are simple multimodal, f_{10} to f_{19} are hybrid, and f_{20} to f_{30} are composite functions. Notably, f_2 is discarded, and f_{30} is as a replacement. State-of-the-art optimizers including covariance matrix adaptation evolution strategy (CMA-ES) [12], hierarchical PSO with time-varying acceleration coefficients (HPSO-TVAC) [13], comprehensive learning PSO (CL-PSO) [14], JADE [15], L-SHADE [11], improved L-SHADE (iL-SHADE) [16], multi-population ensemble DE (MPEDE) [17], jSO [18], L-SHADE-cnEpSin [19], and the original CSO are employed as competitors. CMA-ES, HPSO-TVAC, and CL-PSO are available in the MEALPY library [20], and Advanced DE

variants are provided by the PyADE library[1]. The maximum fitness evaluation is fixed at 1000 $\cdot D$ and each algorithm is independently repeated 30 times. Table I summarizes the parameter setting of algorithms.

TABLE I: Parameter settings of algorithms

Algorithms	Parameters	Value
CMA-ES	population size N	100
	σ	1.3
HPSO-TVAC	population size N	100
	c_{init} and c_{final}	0.5 and 0
CL-PSO	population size N	100
	local coefficient c_{local}	1.2
	max. and min. weight	0.9 and 0.4
JADE	population size N	100
	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.1 and 0.1
L-SHADE	N_{max} and N_{min}	18 $\cdot$ D and 4
	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.1 and 0.1
CSO	population size N	500
	ϕ	0.15
iL-SHADE	N_{max} and N_{min}	12 $\cdot$ D and 4
	μ_F and μ_{Cr}	0.5 and 0.8
	σ_F and σ_{Cr}	0.1 and 0.1
MPEDE	N_{max} and N_{min}	250 and 20
	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.1 and 0.1
	λ_1, λ_2, and λ_3	0.2, 0.2, and 0.2
jSO	N_{max} and N_{min}	12 $\cdot$ D and 4
	p_{max} and p_{min}	0.25 and 0.125
	$\boldsymbol{M}_F$	0.3
L-SHADE-cnEpSin	N_{max} and N_{min}	18 $\cdot$ D and 4
	μ_F and μ_{Cr}	0.5 and 0.5
	σ_F and σ_{Cr}	0.1 and 0.1
	memory size H	5
	ps and pc	0.5 and 0.4
L-ICSO	N_{max} and N_{min}	600 and 6
	ϕ	0.15

B. Experimental results and statistical analysis

This section summarizes the experimental results and analyzes the statistical significance between L-ICSO and competitor algorithms. Table II presents the mean and standard deviation (std) in the CEC2017 benchmark, and the best value is in bold. Fig. 3 presents the convergence curves and boxplots in representative functions. Additionally, we adopt the Mann–Whitney U test to measure the statistical significance between every pair of algorithms, and the Holm multiple comparison test is further employed to correct the p-value obtained from the Mann–Whitney U test. Symbols $+$, $\approx$, and $-$ denote that our proposed L-ICSO is significantly better, has no significance, and is significantly worse than the specific competitor algorithm.

The results summarized in Table II, L-ICSO achieved a statistical significance summary of 27/0/2 against CMA-ES, 28/1/0 against HPSO-TVAC, 26/3/0 against CL-PSO, 29/0/0 against JADE, 29/0/0 against L-SHADE, 18/6/5 against CSO, 23/1/5 against iL-SHADE, 26/0/3 against MPEDE, 29/0/0 against jSO, and 27/0/2 against L-SHADE-cnEpSin. These results demonstrate the superiority and competitiveness of our proposed L-ICSO in addressing large-scale optimization problems. Furthermore, the comparison between CSO and L-ICSO confirms the effectiveness of integrating the novel triple-individuals competitive mechanism and the linear population reduction mechanism into CSO, which significantly accelerates optimization convergence and improves the quality of the optimum.

Meanwhile, we have noticed some shortcomings of L-ICSO. For example, L-ICSO is significantly worse than iL-SHADE in f_3, f_{11}, f_{14}, f_{18}, and f_{26}. However, a remarkable improvement between CSO and L-ICSO in these functions has been observed. Therefore, the introduction of these two strategies is significant. We infer that the primary reason for the observed inferiority is the inherited architecture from CSO, which may not be well-suited for tackling these specific functional problems. This finding is consistent with the No Free Lunch (NFL) Theorem. Additionally, the pronounced performance degradation of L-ICSO compared to CSO in f_5, f_8, f_{10}, f_{21}, and f_{22} highlights the inefficacy of the integrated strategies in these specific problem instances. This suggests that the designed modifications may not generalize well across different landscapes, potentially due to their sensitivity to problem characteristics such as modality, separability, and ruggedness. Moreover, this observation is consistent with the NFL Theorem, which asserts that no single optimization algorithm can outperform all others across all possible functions, which reinforces the necessity of problem-specific algorithmic tuning.

V. Conclusion

This paper introduces an enhanced version of CSO termed improved competitive swarm optimizer with linear population reduction (L-ICSO). The novel triple-individuals competitive mechanism and linear population reduction mechanism are integrated to enhance the global optimization capacity of CSO. The triple-individuals competitive mechanism aims to accelerate optimization convergence, while the linear population reduction mechanism is expected to balance exploration and exploitation across different optimization phases. We conducted numerical experiments in the CEC2017 benchmark and compared L-ICSO with state-of-the-art optimizers such as CSO, L-SHADE, and L-SHADE-cnEpSin. The experimental results and statistical analysis confirm the efficiency and effectiveness of L-ICSO in handling large-scale optimization problems.

In future research, we will continue to develop efficient optimizers based on CSO. This theme includes but is not limited to

[1]https://github.com/xKuZz/pyade

TABLE II: Experimental results in 100-D CEC2017.

Func.		CMA-ES	HPSO-TVAC	CL-PSO	JADE	L-SHADE	CSO	iL-SHADE	MPEDE	jSO	L-SHADE-cnEpSin	L-ICSO
f_1	mean	8.708e+10 +	3.940e+09 +	9.087e+06 +	2.236e+10 +	1.054e+10 +	7.290e+03 +	4.849e+07 +	4.259e+07 +	1.903e+11 +	8.777e+09 +	**3.975e+03**
	std	1.133e+10	1.717e+09	4.143e+06	1.922e+09	4.484e+10	3.203e+03	1.724e+07	1.595e+07	8.446e+09	1.963e+09	2.279e+03
f_3	mean	6.763e+05 +	3.270e+05 +	5.218e+05 +	3.195e+05 +	2.489e+05 +	3.045e+05 +	**8.753e+04** −	2.477e+05 +	3.350e+05 +	2.960e+05 +	2.051e+05
	std	5.912e+04	3.946e+04	4.521e+04	5.128e+03	1.192e+05	1.589e+04	1.305e+04	5.574e+04	3.235e+03	9.300e+03	2.650e+04
f_4	mean	9.478e+03 +	1.515e+03 +	8.291e+02 +	3.742e+03 +	1.569e+04 +	7.795e+02 +	8.223e+02 +	8.875e+02 +	5.514e+04 +	2.015e+03 +	**6.946e+02**
	std	2.572e+03	2.549e+02	2.263e+01	2.751e+02	2.001e+04	2.062e+01	4.926e+01	5.053e+01	5.472e+03	3.654e+02	4.305e+01
f_5	mean	1.696e+03 +	1.078e+03 +	1.425e+03 +	1.467e+03 +	1.256e+03 +	**5.727e+02** −	1.337e+03 +	1.394e+03 +	1.622e+03 +	1.363e+03 +	6.126e+02
	std	4.128e+01	5.974e+01	3.508e+01	2.503e+01	9.841e+01	7.575e+00	1.404e+01	9.236e+01	2.627e+01	1.115e+01	1.612e+01
f_6	mean	6.704e+02 +	6.427e+02 +	6.060e+02 +	6.790e+02 +	6.436e+02 +	6.000e+02 ≈	6.652e+02 +	6.689e+02 +	6.733e+02 +	6.672e+02 +	**6.000e+02**
	std	5.568e+00	5.520e+00	8.758e-01	1.006e+00	3.217e+01	1.587e-03	8.956e-01	1.683e+01	7.198e-01	5.759e-01	4.802e-03
f_7	mean	2.747e+03 +	1.987e+03 +	1.755e+03 +	3.754e+03 +	1.935e+03 +	**9.602e+02** ≈	2.455e+03 +	1.686e+03 +	3.393e+03 +	3.251e+03 +	9.785e+02
	std	1.253e+02	2.056e+02	3.789e+01	2.035e+01	7.435e+02	7.457e+01	7.814e+02	2.699e+01	1.887e+01	1.426e+01	4.599e+01
f_8	mean	2.037e+03 +	1.387e+03 +	1.704e+03 +	1.959e+03 +	1.783e+03 +	**8.733e+02** −	1.878e+03 +	1.817e+03 +	2.138e+03 +	1.906e+03 +	9.032e+02
	std	4.889e+01	9.540e+01	3.888e+01	1.841e+01	1.407e+02	5.559e+00	1.742e+01	8.681e+01	1.997e+01	5.130e+00	2.189e+01
f_9	mean	4.400e+04 +	4.888e+04 +	5.645e+04 +	6.004e+04 +	3.013e+04 +	9.195e+02 +	2.408e+04 +	4.650e+04 +	3.324e+04 +	3.028e+04 +	**9.008e+02**
	std	6.159e+03	7.575e+03	5.302e+03	1.700e+03	1.064e+04	1.860e+01	3.092e+02	2.514e+04	6.675e+02	2.033e+03	1.206e+00
f_{10}	mean	3.232e+04 +	1.774e+04 +	2.654e+04 +	1.831e+04 +	1.796e+04 +	**8.067e+03** −	1.600e+04 +	1.599e+04 +	2.347e+04 +	1.653e+04 +	1.221e+04
	std	3.726e+02	5.758e+03	3.651e+02	2.933e+02	4.347e+03	1.250e+03	3.761e+02	3.408e+02	4.393e+02	2.145e+02	2.347e+03
f_{11}	mean	1.157e+04 +	6.261e+03 +	4.302e+04 +	1.005e+05 +	1.782e+05 +	2.743e+04 +	**2.492e+03** −	2.835e+04 +	1.597e+05 +	2.674e+04 +	3.667e+03
	std	2.450e+03	1.330e+03	6.279e+03	7.684e+03	3.966e+04	3.527e+03	2.067e+02	3.538e+04	1.249e+04	5.278e+03	7.222e+02
f_{12}	mean	3.480e+09 +	2.161e+08 +	3.712e+07 +	5.261e+09 +	2.596e+10 +	2.045e+07 +	1.720e+07 +	4.673e+07 +	1.279e+11 +	4.573e+08 +	**9.939e+06**
	std	1.596e+09	1.697e+08	5.441e+06	5.656e+08	3.971e+10	5.745e+06	8.304e+06	1.443e+07	1.449e+10	1.722e+08	3.662e+06
f_{13}	mean	1.044e+07 +	1.606e+04 +	5.137e+03 ≈	3.022e+08 +	3.255e+09 +	5.490e+03 +	1.695e+04 +	3.739e+04 +	3.103e+10 +	5.826e+04 +	**4.725e+03**
	std	3.314e+06	5.403e+03	1.377e+03	4.040e+07	6.400e+09	1.300e+03	5.942e+03	1.453e+04	2.667e+09	2.076e+04	1.931e+03
f_{14}	mean	2.761e+03 −	1.717e+06 +	6.683e+06 +	7.403e+06 +	1.565e+07 +	2.417e+06 +	**1.934e+03** −	1.129e+04 −	2.950e+07 +	5.242e+05 −	7.072e+05
	std	2.100e+02	9.324e+05	1.984e+06	8.836e+05	1.933e+07	3.695e+05	1.072e+02	6.664e+03	8.451e+06	2.132e+05	1.783e+05
f_{15}	mean	4.356e+04 +	4.554e+03 +	2.660e+03 ≈	4.533e+07 +	3.052e+09 +	**2.363e+03** ≈	4.378e+03 +	4.909e+04 +	1.468e+10 +	2.169e+04 +	2.450e+03
	std	8.340e+03	2.199e+03	4.269e+02	7.666e+06	4.649e+09	5.753e+02	8.879e+02	2.970e+04	2.884e+09	4.583e+03	6.296e+02
f_{16}	mean	1.107e+04 +	5.259e+03 +	7.096e+03 +	1.040e+04 +	1.265e+04 +	3.368e+03 +	5.863e+03 +	9.349e+03 +	1.966e+04 +	6.723e+03 +	**3.111e+03**
	std	3.888e+02	6.808e+02	2.965e+02	3.526e+02	2.141e+03	4.352e+02	1.060e+03	3.811e+02	2.412e+03	4.341e+02	4.037e+02
f_{17}	mean	7.906e+03 +	4.831e+03 +	5.177e+03 +	7.456e+03 +	1.514e+05 +	3.143e+03 +	5.394e+03 +	6.859e+03 +	2.244e+06 +	5.196e+03 +	**2.895e+03**
	std	3.127e+02	5.114e+02	2.759e+02	2.455e+02	9.703e+04	4.156e+02	3.011e+02	2.971e+02	1.303e+06	5.622e+02	3.574e+02
f_{18}	mean	5.758e+05 −	1.711e+06 ≈	6.692e+06 +	6.139e+06 +	2.065e+07 +	2.449e+06 +	**3.297e+04** −	1.308e+05 −	5.476e+07 +	7.396e+05 −	1.615e+06
	std	1.822e+05	1.093e+06	2.060e+06	1.388e+06	2.926e+07	5.616e+05	8.396e+03	3.965e+04	2.457e+07	2.503e+05	5.500e+05
f_{19}	mean	3.857e+05 +	6.041e+03 +	**2.808e+03** ≈	4.838e+07 +	4.130e+09 +	3.057e+03 ≈	3.336e+03 ≈	2.386e+04 +	1.423e+10 +	5.134e+04 +	**2.994e+03**
	std	1.166e+05	3.864e+03	5.061e+02	8.175e+06	4.030e+09	9.372e+02	8.154e+02	1.897e+04	2.718e+09	3.088e+04	1.079e+03
f_{20}	mean	7.687e+03 +	4.785e+03 +	5.512e+03 +	6.370e+03 +	6.981e+03 +	**3.049e+03** ≈	6.205e+03 +	6.257e+03 +	6.486e+03 +	6.252e+03 +	3.170e+03
	std	2.118e+02	8.297e+02	2.999e+02	2.389e+01	7.380e+02	3.459e+02	1.190e+02	3.375e+02	3.599e+01	9.077e+00	4.372e+02
f_{21}	mean	3.645e+03 +	2.935e+03 +	3.217e+03 +	3.469e+03 +	4.353e+03 +	**2.395e+03** −	2.977e+03 +	3.147e+03 +	5.934e+03 +	3.221e+03 +	2.431e+03
	std	8.717e+01	6.673e+01	3.395e+01	2.263e+01	8.876e+01	9.113e+00	6.310e+01	2.439e+01	3.508e+02	5.986e+01	1.963e+01
f_{22}	mean	3.423e+04 +	1.925e+04 +	2.954e+04 +	2.354e+04 +	2.286e+04 +	**9.054e+03** −	2.117e+04 +	2.415e+04 +	2.683e+04 +	2.205e+04 +	1.304e+04
	std	4.986e+02	3.648e+03	4.634e+02	3.073e+02	1.020e+04	4.618e+03	4.738e+02	8.750e+03	3.788e+02	2.639e+02	1.802e+03
f_{23}	mean	4.397e+03 +	3.724e+03 +	3.588e+03 +	8.630e+03 +	4.635e+03 +	2.975e+03 +	7.051e+03 +	3.687e+03 +	9.751e+03 +	8.210e+03 +	**2.946e+03**
	std	6.274e+01	1.082e+02	2.871e+01	2.086e+02	1.036e+01	1.893e+01	9.070e+02	2.823e+01	1.689e+02	7.079e+01	1.958e+01
f_{24}	mean	5.155e+03 +	4.439e+03 +	4.122e+03 +	4.634e+03 +	5.300e+03 +	**3.337e+03** ≈	3.629e+03 +	4.132e+03 +	1.338e+04 +	4.310e+03 +	3.353e+03
	std	2.046e+02	1.828e+02	3.667e+01	4.672e+01	6.187e+01	2.334e+01	1.347e+02	2.907e+01	3.411e+02	1.078e+02	2.635e+01
f_{25}	mean	8.905e+03 +	4.084e+03 +	3.563e+03 +	5.320e+03 +	3.747e+03 +	3.467e+03 +	3.529e+03 +	3.593e+03 +	1.833e+04 +	4.473e+03 +	**3.328e+03**
	std	1.125e+03	2.272e+02	2.487e+01	1.549e+02	3.443e+02	3.252e+01	4.824e+01	4.242e+01	9.449e+02	2.020e+02	2.808e+01
f_{26}	mean	2.259e+04 +	2.111e+04 +	1.555e+04 +	1.315e+04 +	9.710e+03 +	7.134e+03 +	**3.837e+03** −	5.226e+03 −	4.879e+04 +	1.110e+04 +	6.358e+03
	std	8.720e+02	3.235e+03	4.789e+02	1.352e+03	9.377e+02	2.939e+02	3.098e+02	2.984e+03	1.104e+03	1.999e+03	2.398e+02
f_{27}	mean	3.864e+03 +	4.170e+03 +	3.686e+03 +	6.225e+03 +	4.081e+03 +	3.463e+03 +	3.438e+03 +	3.471e+03 +	1.982e+04 +	1.604e+04 +	**3.358e+03**
	std	9.959e+01	1.294e+02	3.124e+01	1.205e+03	7.785e+01	2.121e+01	2.988e+01	2.922e+01	4.274e+02	3.466e+02	1.401e+01
f_{28}	mean	1.056e+04 +	4.909e+03 +	3.905e+03 +	7.012e+03 +	4.538e+03 +	3.679e+03 +	3.647e+03 +	3.799e+03 +	2.528e+04 +	5.382e+03 +	**3.447e+03**
	std	3.018e+03	5.564e+02	5.033e+01	2.554e+02	2.349e+03	4.329e+01	4.575e+01	7.337e+01	1.393e+03	3.317e+02	2.895e+01
f_{29}	mean	1.105e+04 +	7.446e+03 +	8.218e+03 +	1.101e+04 +	3.812e+04 +	5.001e+03 +	7.348e+03 +	8.637e+03 +	1.457e+05 +	9.031e+03 +	**4.568e+03**
	std	3.733e+02	4.946e+02	3.669e+02	2.748e+02	2.598e+04	2.517e+02	2.908e+02	4.129e+02	6.941e+04	6.102e+02	3.184e+02
f_{30}	mean	8.636e+06 +	2.365e+05 +	4.190e+05 +	3.298e+08 +	4.986e+09 +	6.191e+04 +	5.972e+04 +	6.544e+05 +	2.712e+10 +	6.385e+06 +	**1.321e+04**
	std	1.872e+06	1.789e+05	2.169e+05	3.063e+07	6.335e+09	2.360e+04	2.561e+04	2.792e+05	2.847e+09	2.771e+06	2.903e+03
+/≈/−:		27/0/2	28/1/0	26/3/0	29/0/0	29/0/0	18/6/5	23/1/5	26/0/3	29/0/0	27/0/2	-

introducing the cooperative coevolution (CC) framework and hybridizing with efficient search operators to solve larger-scale optimization problems.

VI. Acknowledgment

This work was supported by JST SPRING Grant Number JPMJSP2119.

References

[1] Özge Ekrem and B. Aksoy, "Trajectory planning for a 6-axis robotic arm with particle swarm optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 122, p. 106099, 2023.

[2] M. Qaraad, S. Amjad, N. K. Hussein, M. Farag, S. Mirjalili, and M. A. Elhosseini, "Quadratic interpolation and a new local search approach to improve particle swarm optimization: Solar photovoltaic parameter estimation," *Expert Systems with Applications*, vol. 236, p. 121417, 2024.

[3] R. Zhong, Z. Wang, A. G. Hussien, E. H. Houssein, I. Al-Shourbaji, M. A. Elseify, and J. Yu, "Success history adaptive competitive swarm optimizer with linear population reduction: Performance benchmarking and application in eye disease detection," *Computers in Biology and Medicine*, vol. 186, p. 109587, 2025.

[4] J. Nayak, H. Swapnarekha, B. Naik, G. Dhiman, and S. Vimal, "25 years of particle swarm optimization: Flourishing voyage of two decades," *Archives of Computational Methods in Engineering*, vol. 30, no. 3, pp. 1663–1725, 2023.

[5] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale

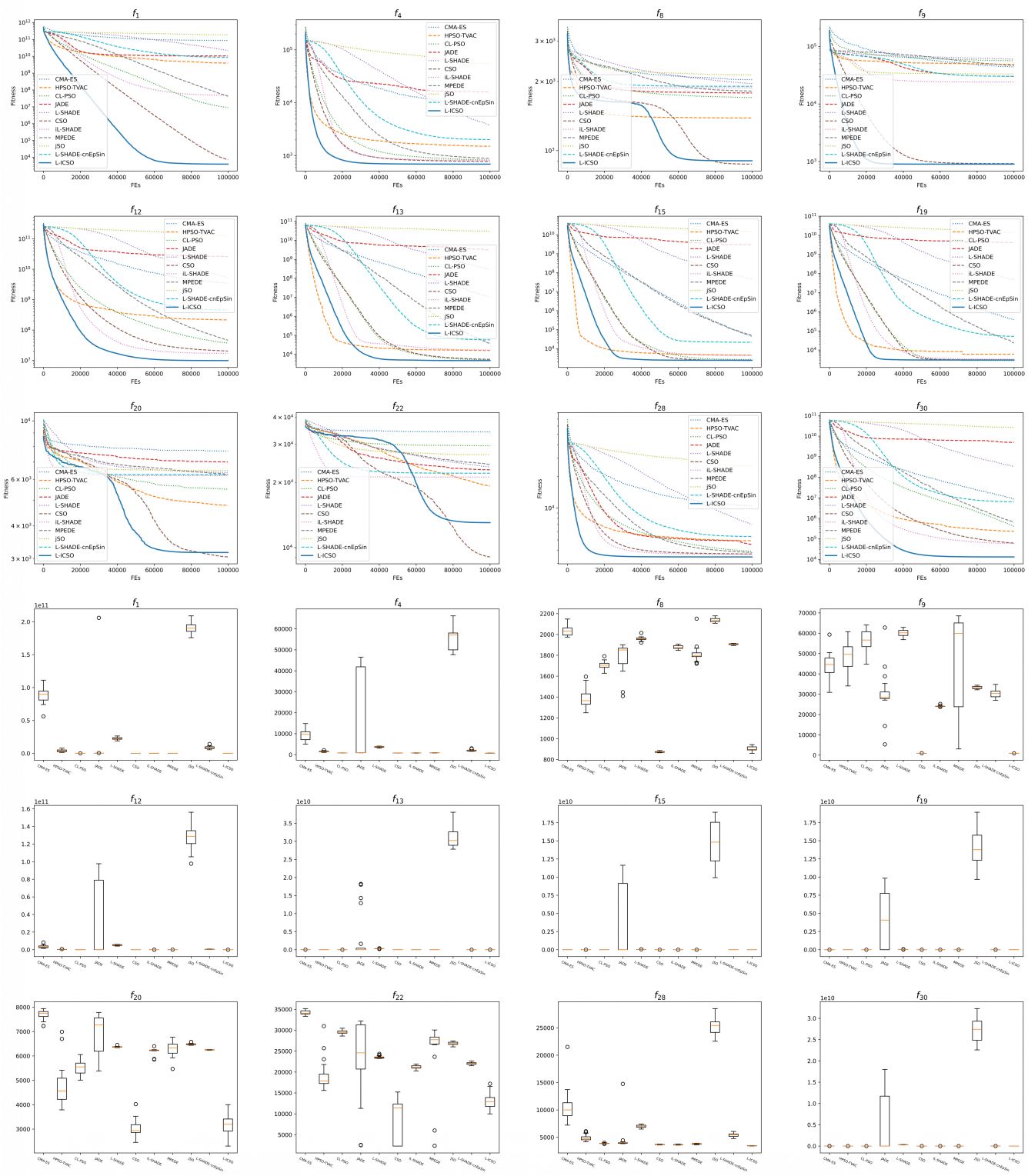

Fig. 3: Convergence curves and boxplots of representative functions in the CEC2017 benchmark (f_1: unimodal function; f_4, f_8, and f_9: multimodal functions; f_{12}, f_{13}, f_{15}, and f_{19}: hybrid functions; f_{20}, f_{22}, f_{28}, and f_{30}: unimodal functions).

optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2015.

[6] P. Mohapatra, K. Nath Das, and S. Roy, "A modified competitive swarm optimizer for large scale optimization problems," *Applied Soft Computing*, vol. 59, pp. 340–362, 2017.

[7] G. Xiong and D. Shi, "Orthogonal learning competitive swarm optimizer for economic dispatch problems," *Applied Soft Computing*, vol. 66, pp. 134–148, 2018.

[8] C. Huang, X. Zhou, X. Ran, Y. Liu, W. Deng, and W. Deng, "Co-evolutionary competitive swarm optimizer with three-phase for large-scale complex optimization problem," *Information Sciences*, vol. 619, pp. 2–18, 2023.
[9] J.-S. Pan, Q. Liang, S.-C. Chu, K.-K. Tseng, and J. Watada, "A multi-strategy surrogate-assisted competitive swarm optimizer for expensive optimization problems," *Applied Soft Computing*, vol. 147, p. 110733, 2023.
[10] D. Chauhan, Shivani, and R. Cheng, "Competitive swarm optimizer: A decade survey," *Swarm and Evolutionary Computation*, vol. 87, p. 101543, 2024.
[11] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, 2014, pp. 1658–1665.
[12] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.
[13] A. Ratnaweera, S. Halgamuge, and H. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.
[14] J. Liang, A. Qin, P. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.
[15] J. Zhang and A. C. Sanderson, "Jade: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.
[16] J. Brest, M. S. Maučec, and B. Bošković, "il-shade: Improved l-shade algorithm for single objective real-parameter optimization," in *2016 IEEE Congress on Evolutionary Computation (CEC)*, 2016, pp. 1188–1195.
[17] G. Wu, R. Mallipeddi, P. Suganthan, R. Wang, and H. Chen, "Differential evolution with multi-population based ensemble of mutation strategies," *Information Sciences*, vol. 329, pp. 329–345, 2016, special issue on Discovery Science.
[18] J. Brest, M. S. Maučec, and B. Bošković, "Single objective real-parameter optimization: Algorithm jso," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 1311–1318.
[19] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, 2017, pp. 372–379.
[20] N. Van Thieu and S. Mirjalili, "Mealpy: An open-source library for latest meta-heuristic algorithms in python," *Journal of Systems Architecture*, vol. 139, p. 102871, 2023.