Region Encoding Helps Evolutionary Computation Evolve Faster: A New Solution Encoding Scheme in Particle Swarm for Large-Scale Optimization

Jun-Rong Jian, *Student Member, IEEE*, Zong-Gan Chen, *Student Member, IEEE*, Zhi-Hui Zhan, *Senior Member, IEEE*, and Jun Zhang, *Fellow, IEEE*

***Abstract*—In the last decade, many evolutionary computation (EC) algorithms with diversity enhancement have been proposed to solve large-scale optimization problems in big data era. Among them, the social learning particle swarm optimization (SLPSO) has shown good performance. However, as SLPSO uses different guidance information for different particles to maintain the diversity, it often results in slow convergence speed. Therefore, this article proposes a new region encoding scheme (RES) to extend the solution representation from a single point to a region, which can help EC algorithms evolve faster. The RES is generic for EC algorithms and is applied to SLPSO. Based on RES, a novel adaptive region search (ARS) is designed to on the one hand keep the diversity of SLPSO and on the other hand accelerate the convergence speed, forming the SLPSO with ARS (SLPSO-ARS). In SLPSO-ARS, each particle is encoded as a region so that some of the best (e.g., the top *P*) particles can carry out region search to search for better solutions near their current positions. The ARS strategy offers the particle a greater chance to discover the nearby optimal solutions and helps to accelerate the convergence speed of the whole population. Moreover, the region radius is adaptively controlled based on the search information. Comprehensive experiments on all the problems in both IEEE Congress on Evolutionary Computation 2010 (CEC 2010) and 2013 (CEC 2013) competitions are conducted to validate the effectiveness and efficiency of SLPSO-ARS and to investigate its important parameters and components. The experimental results show that SLPSO-ARS can achieve generally better performance than the compared algorithms.**

***Index Terms*—Adaptive region search (ARS), evolutionary computation (EC), large-scale optimization problems (LSOPs), region encoding scheme (RES), social learning particle swarm optimization (SLPSO).**

Manuscript received August 18, 2020; revised November 10, 2020, January 12, 2021, and February 16, 2021; accepted March 5, 2021. Date of publication March 12, 2021; date of current version July 30, 2021. This work was supported in part by the National Key Research and Development Program of China under Grant 2019YFB2102102; in part by the Outstanding Youth Science Foundation under Grant 61822602; in part by the National Natural Science Foundations of China (NSFC) under Grant 61772207 and Grant 61873097; in part by the Key-Area Research and Development of Guangdong Province under Grant 2020B010166002; in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003; and in part by the Guangdong-Hong Kong Joint Innovation Platform under Grant 2018B050502006. *(Jun-Rong Jian and Zong-Gan Chen are co-first authors.) (Corresponding authors: Zhi-Hui Zhan; Jun Zhang.)*

Jun-Rong Jian, Zong-Gan Chen, and Zhi-Hui Zhan are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China, and also with Pazhou Laboratory, Guangzhou 510330, China (e-mail: zhanapollo@163.com).

Jun Zhang is with the Chaoyang University of Technology, Taichung 41349, Taiwan, and also with Victoria University, Melbourne, VIC 8001, Australia.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TEVC.2021.3065659.

Digital Object Identifier 10.1109/TEVC.2021.3065659

I. Introduction

In RECENT years, the evolutionary computation (EC) algorithms have been successful in solving various global optimization problems, such as the works in particle swarm optimization (PSO) [1]–[5], ant colony optimization (ACO) [6]–[9], genetic algorithm (GA) [10]–[13], estimation of distribution algorithm (EDA) [14], [15], differential evolution (DE) [16]–[20], and some other algorithms [21], [22]. Among them, PSO is a simple yet efficient algorithm, which designs a swarm of particles and all the particles learn from the currently searched optima to find the optimal solutions [23]. In standard PSO, each particle i needs to maintain two vectors, one is the position vector $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$, and the other is the velocity vector $\boldsymbol{v}_i = \{v_{i1}, v_{i2}, \ldots, v_{iD}\}$, where D is the dimensionality of the problem. In the initialization, each particle randomly initializes its position vector and velocity vector within the corresponding search ranges. In every generation, the velocity vector and position vector of each particle i on dimension d are updated as

$$v_{id}(t+1) = \omega \cdot v_{id}(t) + c_1 \cdot r_{1d} \cdot \left(pbest_{id} - x_{id}(t)\right) + c_2 \cdot r_{2d} \cdot \left(gbest_d - x_{id}(t)\right) \quad (1)$$

$$x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \quad (2)$$

where t is the current generation index, ω is inertia weight [24], c_1 and c_2 are acceleration coefficients [23], r_{1d} and r_{2d} are two random numbers generated uniformly distributed in [0, 1] for the dimension d, and $\boldsymbol{pbest}_i$ (personal best) and $\boldsymbol{gbest}$ (global best) are the best position vectors found so far by particle i and all particles, respectively.

Since PSO has the advantages such as easy implementation and simplicity in solving optimization problems, it has developed rapidly in recent years to solve global optimization problems [25]–[28]. However, as the dimensions of the optimization problems increase in nowadays big data era, the performance of the PSO and also other EC algorithms will deteriorate rapidly [29]. Therefore, adopting some appropriate and efficient methods to improve the ability of PSO and

other EC algorithms in solving large-scale optimization problems (LSOPs) has become a hot research topic. More details about these researches are presented in Section II-A.

Among the numerous large-scale optimization algorithms, there is a representative algorithm named social learning PSO (SLPSO) [30]. In SLPSO, each particle (except the best particle) is updated by learning from any particle in the current swarm that is better than itself, which can help SLPSO become effective to solve LSOPs by keeping population diversity. 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. Therefore, although SLPSO has advantages in keeping diversity, it still has room for improvement on convergence. In the literature, a promising way is to design local search strategies to speed up the population convergence when solving LSOPs. Zhao *et al.* [31] combined the local search strategy (the quasi-Newton method) with a PSO variant to accelerate the convergence speed. Molina and Herrera [32], Molina *et al.* [33] proposed to use multiple local search strategies iteratively to deal with LSOPs.

In this article, to both maintain the diversity and accelerate the convergence speed when solving LSOPs, we propose a novel region encoding scheme (RES) inspired by our previous orthogonal-predictive local search strategy [34] and stochastic coding strategy [35] and apply the RES in the SLPSO. The SLPSO can maintain diversity while the RES can help evolve faster by the novel RES and by using a local search strategy, named adaptive region search (ARS), based on the RES to accelerate the convergence speed. In traditional EC and PSO algorithms, a solution (e.g., a particle in PSO) is encoded as a single point in the search space. However, such an encoding scheme makes the particle represent only one solution in one generation so that the algorithm needs a lot of generations to make up more solutions. This of course results in the slow convergence speed to search for promising solutions. Differently, we propose to use RES to encode a particle as a region rather than only a single point in this article. This way, we can carry out additional operations (e.g., the local search) with the help of the region information to generate more solutions in every generation. This is helpful to speed up the convergence. Based on the RES idea, we propose the novel local search strategy, named ARS, to extend the SLPSO to SLPSO-ARS. In SLPSO-ARS, some of the best (e.g., the top P) particles are selected to carry out region search (RS) at the end of every generation. That is, the top P best particles search the area near their current positions according to the region radius. The ARS strategy can offer the particle with a greater chance to find the better solutions and accelerate the convergence speed. Moreover, each particle has its own region radius which is adaptively controlled during the evolutionary process to make the region more suitable for the search. The experimental results show that the RES and the ARS strategy make SLPSO more efficient in solving LSOPs. More significantly, the RES and the ARS strategy are generic ideas that help evolve faster, leading a new way to enhance EC algorithms in solving LSOPs.

The remainder of this article is structured as follows. Section II introduces some methods for LSOPs and reviews the SLPSO. The detail of the SLPSO-ARS is presented in Section III. Section IV shows the experimental results of SLPSO-ARS and the compared large-scale optimization algorithms on both IEEE Congress on Evolutionary Computation 2010 (CEC 2010) and 2013 (CEC 2013) LSOPs benchmark test sets. Section V draws the conclusion of this article.

II. Background

A. LSOP

LSOP generally has a large amount of decision variables (dimensions), leading to huge search space and many local optima, so it becomes very difficult to be solved by traditional EC algorithms. In general, there are two methods to enhance EC algorithms to better solve LSOPs, one is the cooperative co-evolution (CC) method that decomposes the entire LSOP to reduce problem difficulty, and the other is the non-CC method, that is, to increase population diversity by introducing some additional strategies [36].

In the CC method, the main idea is to use "divide-and-conquer" in EC algorithms to solve LSOPs, such as CCGA [37], CCPSO [38], [39], and DECC [40]. To divide the large-scale variables, random grouping (RG) strategy can be adopted, like in the DECC-G [29] and the multilevel CC (MLCC) [41]. In addition to the RG strategy, some grouping strategies that can detect the relationships between decision variables are also proposed in the literatures, such as delta grouping strategy [42], differential grouping (DG) strategy [43], eXtended DG (XDG) strategy [44], and global DG (GDG) strategy [45]. Moreover, some CC methods have been proposed to allocate different computing resources according to the contribution of the subproblems [46]–[48], while Zhang *et al.* [49] proposed to use the function-independent decomposition strategy to design the CC bare-bones PSO (CCBBPSO).

In the non-CC method, some additional strategies are designed to increase population diversity to improve the performance of the algorithms in solving LSOPs. For example, Takahama and Sakai [50] proposed to control the scaling factor F in DE by detecting landscape modality. Brest *et al.* [51], Brest and Maucec [52] proposed two jDE (a DE variant) variants to self-adapt the population size. In addition, some researchers have designed new operators to solve LSOPs. Yang *et al.* [53] proposed a two-level guidance strategy to modify the standard PSO operators. Similarly, Cheng and Jin [30], [54] also proposed two new PSO operators to solve LSOPs, where each particle could learn from any better particle. Moreover, some works adopt distributed paradigm and multiple populations to deal with LSOPs. Wang *et al.* has proposed two distributed PSO (DPSO) variants, named dynamic group learning DPSO (DGLDPSO) [55] and adaptive granularity learning DPSO (AGLDPSO) [56], which can dynamically change the structure of the subpopulations to increase population diversity. Ge *et al.* [57] combined the

multiple populations strategy with DE and embedded the appropriate migration strategy to solve LSOPs.

B. SLPSO

SLPSO [30] have shown good performance in solving LSOPs and also inherited the advantages of standard PSO. In SLPSO, each particle except the best one learns from any particle that is better than itself. This strategy is also known as the social learning mechanism. That is, SLPSO allows the particles to learn from different particles instead of just learning from ***gbest*** and ***pbest*** like standard PSO. Therefore, SLPSO can increase population diversity because all the particles will not be excessively affected by ***gbest***, which can avoid prematurely falling into the local optima.

In SLPSO, all the particles in the current swarm are sorted based on their fitness values from the worst to the best at the beginning of every generation. Then, all the particles (except the best particle) will update each dimension by learning from a better particle. For each particle, if there are multiple particles better than itself, it will randomly select one of them to guide the update. Note that the particle may use different particles to guide different dimensions. In other words, each particle can learn from multiple better particles, which increases the diversity of the population. Assuming that in the dth dimension, particle i selects to learn from particle k, the updating process is shown as

$$v_{id}(t+1) = r_1 \cdot v_{id}(t) + r_2 \cdot (x_{kd}(t) - x_{id}(t)) + \varepsilon \cdot r_3 \cdot (\bar{x}_d(t) - x_{id}(t)) \tag{3}$$

$$x_{id}(t+1) = \begin{cases} x_{id}(t) + v_{id}(t+1), & \text{if } rand(0,1) \le P_i \\ x_{id}(t), & \text{otherwise} \end{cases} \tag{4}$$

where t represents the current generation, $\boldsymbol{x}$ and $\boldsymbol{v}$ are the position vector and velocity vector like the standard PSO, respectively, r_1, r_2, and r_3 are three random numbers that are generated uniformly distributed in [0, 1]; ε is a parameter named social influence factor, and $\bar{x}_d$ represents the average value of the dth dimension in the position vector of all particles. Note that the k is reselected for each dimension. P_i is a parameter named learning probability used to control whether the particle needs to be updated or not and it is related to the rank of the particle. The learning probability of each particle is different and the learning probability of better particles will be lower. As the particles have been sorted from the worst to the best, P_i is calculated as

$$P_i = \left(1 - \frac{i-1}{N}\right)^{\mu \cdot \log(\lceil \frac{D}{M} \rceil)} \tag{5}$$

where i represents the index of the particle in the sorted population from the worst to the best, N is the population size, μ and M are set to 0.5 and 100 as in the original paper [30], respectively, and D represents the problem dimensionality. When the fitness value of the particle is worse, the rank value of i will be lower, so that the learning probability P_i will be larger. Therefore, the worse particles have a higher chance to update their positions to find better solutions. In addition, P_i is also related to the problem dimensionality D. When D increases, the learning probability P_i will decrease to maintain the population diversity and avoid premature convergence for larger scale problems.

In addition to using social learning mechanism, SLPSO also adapts the relevant parameters according to the dimensionality of the problem which can alleviate the sensitivity of the parameters, such as the values of the population size N and the social influence factor ε. N and ε are calculated as

$$N = M + \left\lfloor \frac{D}{10} \right\rfloor \tag{6}$$

$$\varepsilon = \beta \cdot \frac{D}{M} \tag{7}$$

where β is set to 0.01 according to the suggestion in the original paper [30] because the value of ε should be relatively small to avoid premature convergence. Moreover, N and ε is also related to the problem dimensionality D. The value of N and ε will increase with the increase of the value of D, which can increase the population diversity for larger scale problems.

III. SLPSO-ARS

SLPSO has been proven to have good performance in solving LSOPs. However, in every generation of the algorithm, not all the particles in the swarm can be updated similarly because each particle has different learning probability. Moreover, SLPSO uses different guidance information for different particles and different dimensions to maintain the diversity. So, SLPSO often results in slow convergence speed. Therefore, SLPSO still has room for further improvement. In order to accelerate the convergence speed of SLPSO, we propose to encode the solution by the RES and to execute the novel local search strategy named ARS to further improve the algorithm, forming the SLPSO with ARS (SLPSO-ARS). Subsequently, we will first present the RES and then the ARS strategy. The complete SLPSO-ARS is given at last.

A. RES

The basic idea of RES is that each particle i is no longer a single point as $\boldsymbol{x}_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$, but has a radius r_i to represent a region with $\boldsymbol{x}_i$ as the region center. This way, the region-based solution can cover a wider range of search space. The illustration of point-based encoding scheme and the region-based encoding scheme is compared in Fig. 1(a) and (b).

As shown in Fig. 1, the solid black dots stand for the particles $\boldsymbol{x}_i$. For the point-based encoding scheme, each particle can only represent one position, as shown in Fig. 1(a). For the region-based encoding scheme, as shown in Fig. 1(b), each particle $\boldsymbol{x}_i$ has its own region with radius r_i [as shown by the dotted circle in Fig. 1(b)]. In this way, each particle can occupy a region and find a better solution in the wide search space. Moreover, each particle can execute local search in its own region and quickly find the best position in the region, which is able to greatly speed up the convergence because the local search is sometimes more efficient than evolutionary operations for exploitation. If the global optimum is within the region of particle $\boldsymbol{x}_i$, it can quickly find the global optimal solution. If the RES is not used, the particle $\boldsymbol{x}_i$ may easily

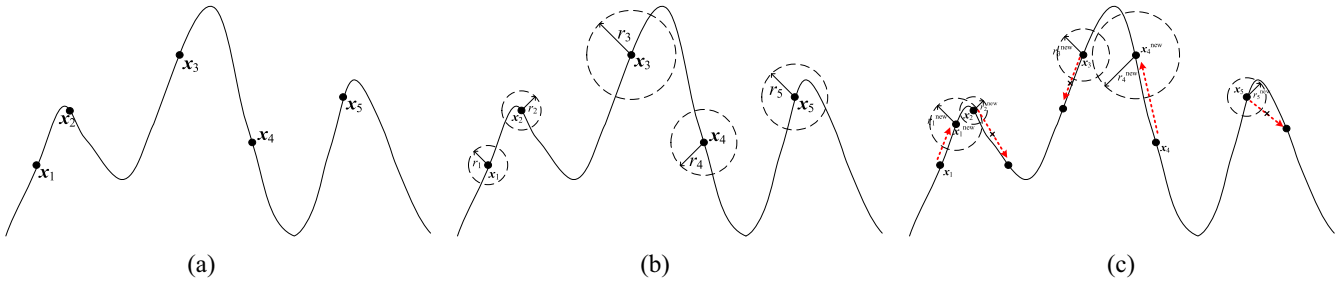


Fig. 1. Two encoding scheme and ARS strategy. (a) Point-based encoding scheme. (b) Region-based encoding scheme. (c) ARS strategy.

Algorithm 1: ARS

Begin

1: **For** $j = 1$ to T
2: $n = rand\_int(1, D)$;
3: **For** $d = 1$ to D
4: Perturb the particle according to (8);
5: **End For**
6: **If** $f(\boldsymbol{x}_i') < f(\boldsymbol{x}_i)$
7: $\boldsymbol{x}_i = \boldsymbol{x}_i$';
8: **End If**
9: $FEs = FEs + 1$;
10: **End For**
11: Modify the region radius r_i of the particle i according to (9).

End

learn from other particles which are currently better than particle $\boldsymbol{x}_i$ but are near the local optima, and then be drawn to the position near local optima. Therefore, RES is very helpful to accelerate the convergence speed of the algorithm.

B. ARS

Based on the RES, the ARS strategy can offer the particle with a greater chance to obtain more promising solutions by the RS. Herein, the adaptation of RS means that in the evolutionary process, the region radius r_i of each particle will be adaptively changed and adjusted according to the RS results. Take Fig. 1(c) as an example to illustrate the ARS strategy. $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, …, $\boldsymbol{x}_5$ are five particles and the top position of the fitness landscape is the optimal solution. In addition, the red dashed arrows represent the finding direction of the particles in the RS. Compared with Fig. 1(b), the particles $\boldsymbol{x}_1$ and $\boldsymbol{x}_4$ have found better positions ${\boldsymbol{x}_1}^{\text{new}}$ and ${\boldsymbol{x}_4}^{\text{new}}$ in the current generation, respectively, which means that the radius are promising for the particles and should be further enhanced. Therefore, the ARS strategy increases their region radii r_1 and r_4 to ${r_1}^{\text{new}}$ and ${r_4}^{\text{new}}$, respectively, in order to use a larger radius to help the particles fast move to the optimal solutions. On the contrary, the particles $\boldsymbol{x}_2$, $\boldsymbol{x}_3$, and $\boldsymbol{x}_5$ cannot find a better position than their original positions, which means their current positions may be in the promising regions and too large radius may make the particles overstep the optimal positions. Therefore, the ARS strategy decreases their region radius r_2, r_3, and r_5 to ${r_2}^{\text{new}}$, ${r_3}^{\text{new}}$, and ${r_5}^{\text{new}}$, respectively, in order to make exploitation in the promising regions with smaller radius. Moreover, the RS is carried out on some of the best (e.g., the top P) particles at the end of every generation. The procedure of the RS on each performed particle is described as follows.

Every time when executing RS, every dimension of the particle will be perturbed with a probability ρ in the region centered at the particle with a radius r_i, namely, search the region near the particle i. Then, if a solution better than particle i is found, the particle i will be replaced by this better solution. Therefore, RS not only can help the population to jump out of the local optimal solutions but also can accelerate the convergence speed of population, so that it can solve LSOPs better.

When particle i is carrying out RS, the dth dimension of the perturbed particle $\boldsymbol{x}_i$' is generated as

$$x_{id}^{'} = \begin{cases} x_{id} + N(0,1) \cdot r_i, & \text{if } d == n \text{ or } rand(0,1) < \rho \\ x_{id}, & \text{otherwise} \end{cases} \tag{8}$$

where $\boldsymbol{x}_i$' and $\boldsymbol{x}_i$ are the position vectors of the perturbed particle i and the original particle i respectively. ρ is a parameter named perturbation probability, which is used to control whether the dimension of the particle is changed when carrying out RS. As we generally only choose some of the best (e.g., the top P) particles in the current swarm to carry out RS, these particles may already have very good information in most of the dimensions and a large perturbation may be harmful. Therefore, a relatively small value of ρ is beneficial to guarantee the properties of the best particles to a large extent and most of the good dimensions will not be destroyed. That is, if the value of ρ is set too large, more dimensions of the best particles will be changed, resulting in poor performance of the perturbed particles due to the more difference with the original best particles. Therefore, the value of ρ is relatively small in SLPSO-ARS. The n is a random integer which is generated uniformly distributed within [1, D], where D represents the dimensionality of the problem. Here, it is to make sure that the perturbed particle $\boldsymbol{x}_i$' is not exactly the same as the original particle $\boldsymbol{x}_i$. $N(0, 1)$ represents the random number generated in the standard Gaussian distribution, and r_i represents the region radius of particle i, which is adaptively changed according to the evolutionary process. When the fitness value of $\boldsymbol{x}_i$' is better than $\boldsymbol{x}_i$, $\boldsymbol{x}_i$ will be replaced with $\boldsymbol{x}_i$'. The above perturbation process will be repeated T times, which is equivalent to generating T new particles in the search region nearby particle i.

If none of the perturbed particles which is better than the original particle i is found after T times of RS, indicating that the original particle i may have found an approximate global or local optimal solution, which is also an approximate optimal solution in its own region, so the region radius r_i of particle i will be decreased. Otherwise, the region radius r_i will be increased to explore the optimal solutions in a larger region. The update rule of region radius r_i is shown as

$$r_i = \begin{cases} r_i \cdot c, & \text{if none of the } T \text{ particles} \\ & \text{is better than the orginal particle } i \\ r_i / c, & \text{otherwise} \end{cases} \tag{9}$$

where c is a parameter within [0, 1], named the scaling factor.

It should be noted that the region radius r has a maximum value $r_{\max}$. If the value of r exceeds $r_{\max}$ when it changes, set $r = r_{\max}$. The value of $r_{\max}$ is gradually reduced over the evolutionary process. $r_{\max}$ is calculated as

$$r_{\max} = r_0 \cdot \frac{maxFEs - FEs + 1}{maxFEs} \tag{10}$$

where r_0 is the initial region radius, and *FEs* and *maxFEs* are the current and maximum number of fitness evaluations (FEs), respectively. The procedure of ARS is shown in Algorithm 1.

C. SLPSO-ARS

SLPSO-ARS is proposed on the basis of SLPSO. This algorithm embeds the new local search strategy named ARS introduced in Algorithm 1 to improve the ability of SLPSO in solving LSOPs, which can accelerate the convergence speed of the population while maintaining population diversity. In SLPSO-ARS, the evolutionary process still adopts SLPSO, but the main difference is the addition of the ARS strategy after updating the particles. At the end of every generation, all the particles in the current swarm are sorted based on their fitness values from the worst to the best. Then, the top P particles are selected to carry out RS. Since we only select a small part of particles in the population to execute RS, other particles remain unchanged, so the algorithm keeps the properties of original population to a large extent for the purpose of maintaining the diversity. At the same time, the algorithm will select the top P particles to carry out RS according to the rank, which has a greater chance to search for the nearby optima. Therefore, the ARS strategy can help the population to converge faster.

Based on the ARS strategy in Algorithm 1, the process of the complete SLPSO-ARS is shown in Algorithm 2. In Algorithm 2, we only select some particles with higher rank to execute RS, which not only can save FEs but also have a higher chance to find the nearby optimal solutions. In addition, it should be noted that the region radius r of each particle is initialized to r_0 at the beginning of the algorithm, and the r_0 is calculated as

$$r_0 = \frac{ubound - lbound}{10} \tag{11}$$

where *ubound* and *lbound* are the upper and lower bounds of the problem space, respectively.

Algorithm 2: SLPSO-ARS

Begin

1: $N = 200$; $\varepsilon = 0.1$; $P = 5$; $T = 5$; $\rho = 0.01$; $c = 0.5$;
2: Population initialization: randomly generate N particles and calculate the fitness value of each particle;
3: Initialize the region radius r of each particle to be r_0;
4: Sort all the particles based on the fitness values from the worst to the best; //This is required by SLPSO
5: $FEs = FEs + N$;
6: **While** $FEs \leq maxFEs$
7: **For** i = 1 to N-1
8: **If** $rand(0, 1) \leq P_i$
9: **For** d = 1 to D
10: $k = randi\_int(i + 1, N)$;
11: Update the particles according to (3) and (4);
12: **End For**
13: Calculate the fitness value of particle i;
14: $FEs = FEs + 1$;
15: **End If**
16: **End For**
17: Sort all the particles according to the fitness values and select the top P particles;
18: **//Carry out ARS**
19: **For** $i = 1$ to P
20: Execute ARS using **Algorithm 1**;
21: **End For**
22: Sort all the particles based on the fitness values from the worst to the best. //This is required by SLPSO
23: **End While**

End

The advantage of SLPSO-ARS over SLPSO is that it introduces a novel RES and a new local search strategy ARS based on the RES. The ARS strategy has two main advantages.

1) By selecting the particles with better fitness values at the end of every generation to execute RS and search the better solutions in the region near the particles, the chance of finding the nearby optimal solutions is increased and the convergence speed of the population can be accelerated. Moreover, a small value of perturbation probability ρ is used to perturb the dimensions of the particles when executing RS, so most of the excellent properties of the original particles are maintained.
2) Each particle has its own region radius r and the region radius r can be changed adaptively during the evolutionary process. Therefore, each particle has the most suitable region radius r, which can ease the sensitivity of this parameter.

IV. Experiments

In this section, we adopt two LSOPs benchmark test sets to validate the effectiveness and efficiency of SLPSO-ARS in solving LSOPs. First, in Section IV-A, we will show the experimental setup and give the parameter settings of SLPSO-ARS. Subsequently, Sections IV-B and IV-C present

TABLE I
EXPERIMENTAL RESULTS OF SLPSO-ARS AND THE COMPARED ALGORITHMS ON THE CEC 2010 LARGE-SCALE BENCHMARK TEST SET

Fun	Properties	SLPSO-ARS	SLPSO	CSO	DMS-L-PSO	CCPSO2	DECC-G	MLCC	DECC-DG
f_1	Mean	6.64E-19	8.73E-18(+)	4.50E-12(+)	4.88E+09(+)	1.64E+00(+)	1.36E-14(+)	**0(-)**	1.61E+01(+)
	Std	2.31E-20	5.19E-19	7.78E-13	2.31E+08	1.69E+00	1.73E-15	**0**	1.19E+01
f_2	Mean	2.42E+03	1.93E+03(-)	7.42E+03(+)	6.17E+03(+)	7.49E+00(-)	4.88E+01(-)	**3.32E-01(-)**	4.47E+03(+)
	Std	9.11E+01	1.12E+02	2.86E+02	1.87E+02	1.74E+00	1.31E+01	**5.43E-01**	2.09E+02
f_3	Mean	**3.51E-13**	1.88E+00(+)	2.60E-09(+)	1.74E+01(+)	8.89E-03(+)	1.71E+00(+)	8.17E-02(+)	1.67E+01(+)
	Std	**4.25E-15**	1.66E-01	4.52E-10	5.61E-02	1.17E-02	2.88E-01	3.11E-01	2.71E-01
f_4	Mean	3.25E+11	**2.99E+11(-)**	7.25E+11(+)	3.90E+13(+)	1.15E+12(+)	1.25E+13(+)	1.54E+13(+)	3.82E+12(+)
	Std	4.93E+10	**7.16E+10**	1.23E+11	4.40E+12	7.41E+11	2.85E+12	6.57E+12	6.75E+11
f_5	Mean	1.19E+07	3.17E+07(+)	**1.15E+07(≈)**	1.04E+08(+)	4.53E+08(+)	2.60E+08(+)	3.13E+08(+)	1.54E+08(+)
	Std	2.67E+06	6.21E+06	**1.62E+06**	7.83E+06	1.18E+08	8.16E+07	1.09E+08	1.90E+07
f_6	Mean	**7.32E-08**	2.08E+01(+)	8.21E-07(+)	1.66E+06(+)	1.92E+07(+)	4.76E+06(+)	1.61E+07(+)	1.64E+01(+)
	Std	**2.38E-09**	3.98E+00	4.61E-08	4.42E+05	1.68E+06	6.14E+05	4.40E+06	2.73E-01
f_7	Mean	**1.69E+02**	6.49E+04(+)	2.01E+04(+)	2.42E+10(+)	1.70E+08(+)	8.39E+06(+)	1.81E+06(+)	5.81E+03(+)
	Std	**3.62E+01**	5.60E+04	3.86E+03	1.63E+09	3.23E+08	7.56E+06	2.85E+06	2.66E+03
f_8	Mean	2.30E+07	**7.81E+06(-)**	3.87E+07(+)	1.43E+08(+)	3.31E+07(-)	4.72E+07(+)	3.76E+07(≈)	3.94E+07(≈)
	Std	2.50E+05	**1.56E+06**	6.81E+04	3.42E+07	2.97E+07	3.08E+07	3.27E+07	2.98E+07
f_9	Mean	7.84E+07	**3.30E+07(-)**	7.03E+07(-)	5.79E+09(+)	1.14E+08(+)	2.54E+08(+)	1.19E+08(+)	5.95E+07(-)
	Std	2.68E+06	**4.46E+06**	5.73E+06	2.02E+08	3.60E+07	1.01E+07	1.44E+07	9.21E+06
f_{10}	Mean	**1.91E+03**	2.56E+03(+)	9.60E+03(+)	5.88E+03(+)	5.71E+03(+)	9.22E+03(+)	2.98E+03(+)	4.55E+03(+)
	Std	**2.34E+02**	2.17E+02	7.67E+01	2.48E+02	1.03E+03	4.37E+02	3.70E+02	1.21E+02
f_{11}	Mean	**3.32E-12**	2.32E+01(+)	4.02E-08(+)	1.82E+02(+)	1.98E+02(+)	2.52E+01(+)	1.96E+02(+)	1.13E+01(+)
	Std	**8.86E-14**	2.10E+00	6.92E-09	8.59E+00	2.39E-01	1.08E+00	3.07E+00	5.04E-01
f_{12}	Mean	6.50E+04	1.75E+04(-)	4.37E+05(+)	2.84E+06(+)	2.78E+04(-)	3.91E+04(-)	3.60E+04(-)	**2.53E+03(-)**
	Std	5.89E+03	9.07E+03	6.22E+04	1.10E+05	7.58E+03	5.81E+03	6.49E+03	**3.14E+02**
f_{13}	Mean	8.45E+02	9.59E+02(+)	**6.29E+02(-)**	9.68E+07(+)	1.28E+03(+)	3.13E+03(+)	2.37E+03(+)	4.86E+03(+)
	Std	4.78E+02	3.74E+02	**2.32E+02**	2.62E+07	1.82E+02	1.15E+03	1.64E+03	2.73E+03
f_{14}	Mean	3.71E+08	**8.41E+07(-)**	2.49E+08(-)	5.02E+09(+)	3.22E+08(≈)	5.77E+08(+)	3.24E+08(-)	3.40E+08(-)
	Std	2.06E+07	**6.31E+06**	1.53E+07	3.43E+08	1.46E+08	2.18E+07	1.97E+07	1.85E+07
f_{15}	Mean	**1.99E+03**	1.12E+04(+)	1.01E+04(+)	6.21E+03(+)	1.02E+04(+)	9.79E+03(+)	7.17E+03(+)	5.84E+03(+)
	Std	**1.76E+02**	8.65E+01	5.23E+01	2.76E+02	8.90E+02	2.74E+03	1.14E+03	6.02E+01
f_{16}	Mean	5.77E-12	2.51E+01(+)	5.89E-08(+)	3.39E+02(+)	3.97E+02(+)	8.50E+01(+)	3.81E+02(+)	**7.23E-13(-)**
	Std	1.00E-13	2.42E+00	6.34E-09	7.88E-01	4.64E-01	1.04E+01	4.77E+01	**4.89E-14**
f_{17}	Mean	3.07E+05	9.00E+04(-)	2.20E+06(+)	2.67E+06(+)	1.41E+05(-)	1.63E+05(-)	1.56E+05(-)	**4.18E+04(-)**
	Std	1.47E+04	1.58E+04	1.55E+05	1.54E+05	5.81E+04	9.60E+03	1.03E+04	**1.14E+03**
f_{18}	Mean	2.28E+03	2.77E+03(+)	**1.73E+03(-)**	2.82E+09(+)	2.87E+03(+)	9.00E+03(+)	6.83E+03(+)	1.51E+10(+)
	Std	9.66E+02	8.33E+02	**5.22E+02**	5.30E+08	3.73E+02	1.09E+03	5.99E+03	1.86E+09
f_{19}	Mean	1.91E+06	5.10E+06(+)	1.01E+07(+)	1.63E+07(+)	1.41E+06(-)	**7.33E+05(-)**	1.31E+06(-)	1.71E+06(-)
	Std	5.03E+04	7.05E+05	5.64E+05	6.70E+05	8.90E+04	**4.61E+04**	1.05E+05	1.04E+05
f_{20}	Mean	**9.95E+02**	1.85E+03(+)	1.05E+03(+)	4.10E+09(+)	1.97E+03(+)	3.47E+03(+)	2.03E+03(+)	6.17E+10(+)
	Std	**1.70E+01**	1.80E+02	1.59E+02	6.34E+08	2.08E+02	2.45E+02	1.96E+02	5.82E+09
+(SLPSO-ARS is significantly better)			13	15	20	14	16	13	13
-(SLPSO-ARS is significantly worse)			7	4	0	5	4	6	6
≈(no significant difference)			0	1	0	1	0	1	1

the experimental results of SLPSO-ARS and several well-known large-scale optimization algorithms on the CEC 2010 and CEC 2013 large-scale benchmark test sets, respectively. Then, comparison experiments between SLPSO-ARS and the winning algorithm in CEC 2010 competition are conducted in Section IV-D. Finally, Section IV-E investigates some important parameters of SLPSO-ARS and the influence of its some components.

A. Parameters and Experimental Settings

In the experiments, we adopt two extensively used LSOPs benchmark test sets (the CEC 2010 large-scale benchmark set [58] and CEC 2013 large-scale benchmark set [59]) to validate the performance of SLPSO-ARS, like many other works [47], [53], [56]. In addition, we compare the experimental results obtained by SLPSO-ARS with seven well-known large-scale optimization algorithms, including four algorithms using the CC method (CCPSO2 [39], DECC-G [29], MLCC [41], and DECC-DG [43]) and three non-CC method algorithms (SLPSO [30], CSO [54], and DMS-L-PSO [31]), to further show the superiority of the new SLPSO-ARS algorithm.

In parameter settings of SLPSO-ARS, the population size N is set to 200, the social influence factor ε is set to 0.1, the number of particles for ARS (i.e., the P) and the times for RS (i.e., the T) are both set to 5, perturbation probability ρ and scaling factor c are set to 0.01 and 0.5, respectively. The parameters tuning and effects of SLPSO-ARS will be investigated in Section IV-E. In terms of the compared algorithms, all the parameters are set based on their original papers because

TABLE II
EXPERIMENTAL RESULTS OF SLPSO-ARS AND THE COMPARED ALGORITHMS ON THE CEC 2013 LARGE-SCALE BENCHMARK TEST SET

Fun	Properties	SLPSO-ARS	SLPSO	CSO	DMS-L-PSO	CCPSO2	DECC-G	MLCC	DECC-DG
f_1	Mean	7.34E-19	1.09E-17(+)	3.67E-12(+)	5.70E+09(+)	5.26E+00(+)	2.16E-12(+)	**8.10E-26(-)**	1.25E+03(+)
	Std	5.04E-20	2.59E-18	1.20E-12	1.44E+08	1.32E+00	8.83E-13	**3.11E-25**	3.99E+03
f_2	Mean	2.79E+03	2.13E+03(-)	7.04E+03(+)	1.24E+04(+)	1.24E+01(-)	4.90E+01(-)	**8.29E+00(-)**	1.28E+04(+)
	Std	2.87E+02	1.36E+02	3.56E+02	2.69E+02	1.00E+00	2.25E+01	**5.58E+00**	4.62E+02
f_3	Mean	2.01E+01	2.16E+01(+)	2.16E+01(+)	2.14E+01(+)	**2.00E+01(-)**	2.01E+01(+)	**2.00E+01(-)**	2.14E+01(+)
	Std	1.98E-02	1.45E-02	5.44E-03	1.92E-02	**4.00E-05**	2.27E-03	**1.65E-03**	1.52E-02
f_4	Mean	1.02E+10	**4.35E+09(-)**	1.26E+10(+)	9.00E+11(+)	1.61E+10(+)	1.42E+11(+)	8.76E+10(+)	5.24E+10(+)
	Std	2.69E+09	**9.48E+08**	1.91E+09	3.78E+10	7.64E+09	6.49E+10	2.85E+10	3.36E+10
f_5	Mean	**7.60E+05**	8.41E+05(+)	8.62E+05(+)	5.49E+06(+)	1.70E+07(+)	7.53E+06(+)	1.02E+07(+)	5.82E+06(+)
	Std	**1.36E+05**	1.23E+05	2.13E+04	4.19E+05	4.09E+06	1.26E+06	1.93E+06	3.49E+05
f_6	Mean	**1.03E+06**	1.06E+06(+)	1.06E+06(+)	**1.03E+06(≈)**	1.05E+06(+)	1.06E+06(+)	1.05E+06(+)	1.06E+06(+)
	Std	**5.20E+03**	1.48E+03	1.05E+03	**3.99E+03**	9.50E+03	5.82E+02	3.67E+03	9.04E+02
f_7	Mean	7.27E+06	**1.63E+06(-)**	7.62E+06(≈)	3.55E+09(+)	1.16E+08(+)	3.98E+08(+)	4.30E+08(+)	8.35E+08(+)
	Std	3.42E+06	**7.05E+05**	1.35E+06	2.61E+08	8.71E+07	3.03E+08	2.18E+08	7.66E+08
f_8	Mean	1.59E+14	**1.03E+14(-)**	3.50E+14(+)	6.79E+15(+)	6.20E+14(+)	2.94E+15(+)	4.59E+15(+)	4.59E+15(+)
	Std	3.76E+13	**3.62E+13**	3.59E+13	1.38E+15	5.84E+14	1.29E+15	3.71E+15	5.25E+14
f_9	Mean	4.84E+07	8.25E+07(+)	**3.94E+07(-)**	5.05E+08(+)	3.18E+09(+)	5.97E+08(+)	8.98E+08(+)	5.00E+08(+)
	Std	9.19E+06	1.06E+07	**6.36E+06**	2.66E+07	6.00E+08	1.19E+08	1.97E+08	2.20E+07
f_{10}	Mean	**9.20E+07**	9.25E+07(≈)	9.41E+07(+)	9.31E+07(+)	9.37E+07(+)	9.30E+07(+)	9.22E+07(≈)	9.46E+07(+)
	Std	**4.09E+05**	1.67E+06	1.49E+05	3.11E+05	4.40E+05	5.53E+05	3.84E+05	3.47E+04
f_{11}	Mean	**1.53E+09**	9.33E+11(+)	3.58E+11(+)	4.96E+11(+)	9.30E+11(+)	5.90E+10(+)	1.20E+11(+)	2.35E+10(+)
	Std	**5.21E+09**	9.02E+09	9.80E+09	4.01E+10	9.58E+09	4.49E+10	2.77E+10	1.32E+10
f_{12}	Mean	**1.06E+03**	1.78E+03(+)	1.28E+03(+)	4.42E+09(+)	2.02E+03(+)	3.36E+03(+)	2.10E+03(+)	1.63E+11(+)
	Std	**8.47E+01**	1.74E+02	8.23E+01	8.34E+08	8.87E+01	2.69E+02	1.99E+02	1.61E+10
f_{13}	Mean	**3.66E+08**	4.65E+08(+)	8.06E+08(+)	1.16E+11(+)	2.04E+09(+)	4.54E+09(+)	8.19E+09(+)	1.98E+10(+)
	Std	**2.55E+08**	3.17E+08	1.02E+08	1.05E+10	5.54E+08	6.47E+08	2.74E+09	6.05E+09
f_{14}	Mean	1.83E+09	**3.28E+08(-)**	5.17E+09(+)	1.25E+12(+)	1.42E+11(+)	7.53E+10(+)	1.18E+11(+)	1.86E+10(+)
	Std	1.33E+09	**5.17E+08**	2.86E+09	1.59E+11	9.87E+10	3.44E+10	6.86E+10	9.40E+09
f_{15}	Mean	7.35E+06	7.87E+07(+)	1.74E+07(+)	1.60E+09(+)	**3.67E+06(-)**	4.76E+06(-)	6.70E+06(-)	9.51E+06(+)
	Std	6.42E+05	8.45E+06	6.53E+05	6.18E+08	**1.73E+06**	4.21E+05	9.17E+05	9.84E+05
+(SLPSO-ARS is significantly better)			9	13	14	12	13	10	15
-(SLPSO-ARS is significantly worse)			5	1	0	3	2	4	0
≈(no significant difference)			1	1	1	0	0	1	0

their parameters have been fine tuned for solving the related LSOPs. For the sake of fairness, the maximum number of FEs of all the algorithms is uniformly set to 3e6. In order to avoid the occasionality of the results, the experimental results of all the algorithms are obtained by running 30 times independently and calculating the average values. Besides, SLPSO-ARS and any of the compared algorithms are performed significance analysis by carrying out Wilcoxon's rank sum test with level $\alpha = 0.05$ in Sections IV-B and IV-C. We use the symbols "+," "−," and "≈" to indicate SLPSO-ARS is significantly better than, significantly worse than, and no significant difference to the compared algorithms in Tables I and II, respectively. While the symbols "+," "−," and "≈" indicate SLPSO-ARS is better than, worse than, and the same as the compared algorithms in Tables III, V, and VI, respectively. Moreover, the statistical results are also given at the end of all the Tables. Note that we also represent the optimal results (minimum mean values) on each function in boldface.

B. Comparisons With Seven Well-Known Algorithms on the CEC 2010 LSOPs Benchmark Test Set

The main properties of all the LSOPs in the CEC2010 large-scale benchmark test set [58] are presented in Table S.I in the supplementary material. There are 20 LSOPs in this benchmark set, including three completely separable functions (f_1–f_3), 15 partially separable functions (f_4–f_{18}), and two completely nonseparable functions (f_{19} and f_{20}). The experimental results of SLPSO-ARS and the compared algorithms on the CEC 2010 large-scale benchmark test set are shown in Table I.

Since SLPSO-ARS is improved on the basis of SLPSO, we first observe the comparison results between these two algorithms. As shown in Table I, for three completely separable functions (f_1–f_3), SLPSO-ARS achieves better performance than SLPSO on functions f_1 and f_3, especially on function f_3. In terms of 15 partially separable functions (f_4–f_{18}), SLPSO-ARS performs better than SLPSO among nine functions (i.e., f_5–$f_7, f_{10}, f_{11}, f_{13}, f_{15}, f_{16}$, and f_{18}), while SLPSO outperforms SLPSO-ARS only on 6 functions. Especially on functions f_6, f_{11}, and f_{16}, the performance of SLPSO-ARS is much better than SLPSO, which are more than nine orders of magnitude better than SLPSO. For the last two nonseparable functions (f_{19} and f_{20}), SLPSO-ARS also performs better than SLPSO.

Overall, among the 20 LSOPs in the CEC 2010 benchmark test set, SLPSO-ARS is significantly better than SLPSO

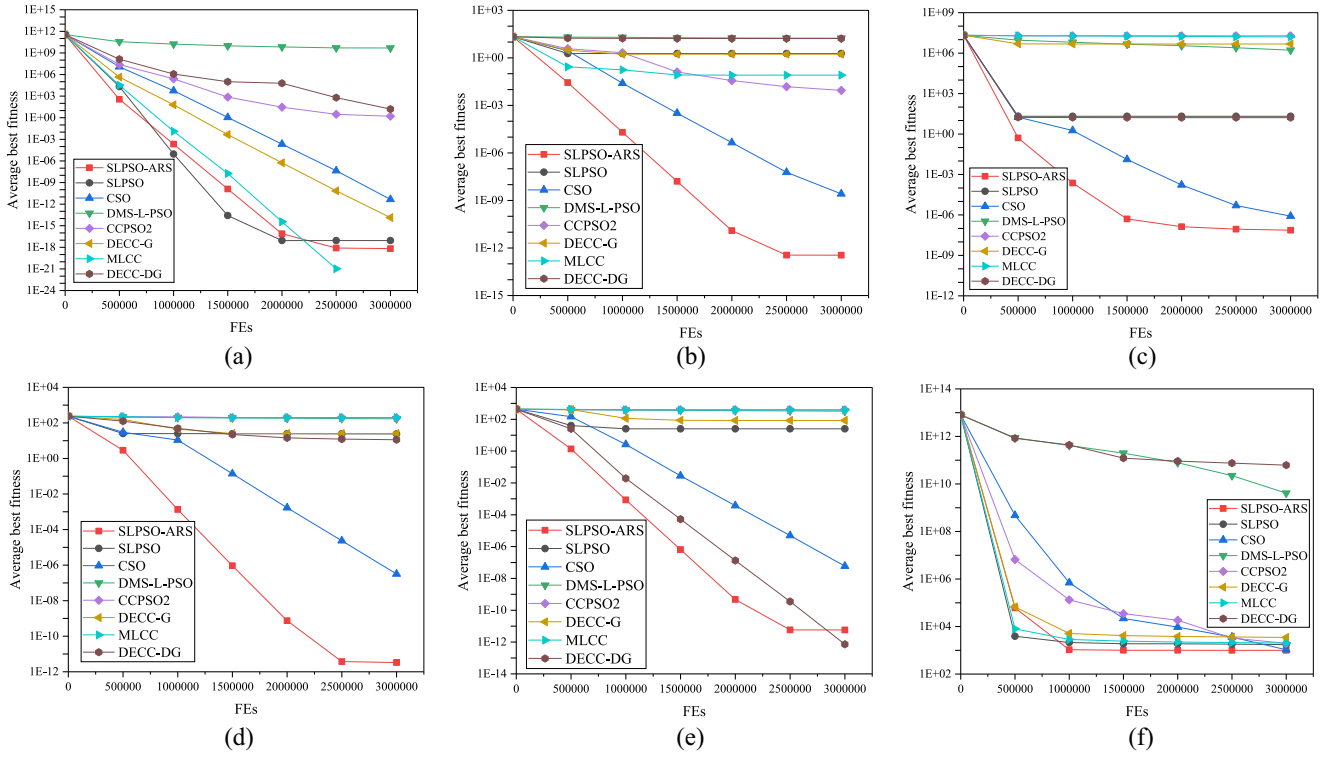


Fig. 2. Convergence curves of SLPSO-ARS and the compared algorithms on six representative functions in the CEC 2010 large-scale benchmark test set. (a) Completely separable function f_1. (b) Completely separable function f_3. (c) Partially separable function f_6. (d) Partially separable function f_{11}. (e) Partially separable function f_{16}. (f) Completely nonseparable function f_{20}.

TABLE III
EXPERIMENTAL RESULTS OF SLPSO-ARS AND MA-SW-CHAINS ON THE CEC 2010 LARGE-SCALE BENCHMARK TEST SET

Fun	SLPSO-ARS	MA-SW-Chains
	Mean±Std	Mean±Std
f_1	**6.64E-19±2.54E-20**	2.10E-14±1.99E-14(+)
f_2	2.42E+03±9.11E+01	**8.10E+02±5.88E+01(-)**
f_3	**3.51E-13±4.82E-15**	7.28E-13±3.40E-13(+)
f_4	**3.25E+11±4.93E+10**	3.53E+11±3.12E+10(-)
f_5	**1.19E+07±2.67E+06**	1.68E+08±1.04E+08(+)
f_6	**7.32E-08±2.98E-09**	8.14E+04±2.84E+05(+)
f_7	1.69E+02±3.62E+01	**1.03E+02±8.70E+01(-)**
f_8	2.30E+07±2.50E+05	**1.41E+07±3.68E+07(-)**
f_9	7.84E+07±2.68E+06	**1.41E+07±1.15E+06(-)**
f_{10}	**1.91E+03±2.34E+02**	2.07E+03±1.44E+02(+)
f_{11}	**3.32E-12±7.38E-14**	3.80E+01±7.35E+00(+)
f_{12}	6.50E+04±5.89E+03	**3.62E-06±5.92E-07(-)**
f_{13}	**8.45E+02±4.78E+02**	1.25E+03±5.72E+02(+)
f_{14}	3.71E+08±2.06E+07	**3.11E+07±1.93E+06(-)**
f_{15}	**1.99E+03±1.76E+02**	2.74E+03±1.22E+02(+)
f_{16}	**5.77E-12±1.22E-13**	9.98E+01±1.40E+01(+)
f_{17}	3.07E+05±1.47E+04	**1.24E+00±1.25E-01(-)**
f_{18}	2.28E+03±9.66E+02	**1.30E+03±4.36E+02(-)**
f_{19}	1.91E+06±5.03E+04	**2.85E+05±1.78E+04(-)**
f_{20}	**9.95E+02±2.51E+01**	1.07E+03±7.29E+01(+)
+(SLPSO-ARS is better)		11
-(SLPSO-ARS is worse)		9
≈(no difference)		0

on 13 functions, while SLPSO achieves significantly better performance than SLPSO-ARS only on seven functions. Especially on the Ackley functions (i.e., f_3, f_6, f_{11}, and f_{16}), the performance of SLPSO-ARS is particularly outstanding.

In addition to SLPSO, from Table I, we can see that among the 20 LSOPs in the CEC 2010 large-scale benchmark test set, SLPSO-ARS performs significantly better than CSO, DMS-L-PSO, CCPSO2, DECC-G, MLCC, and DECC-DG on 15, 20, 14, 16, 13, and 13 functions, respectively, while all the compared algorithms cannot perform significantly better than SLPSO-ARS on more than six functions. Therefore, SLPSO-ARS shows outstanding performance and generally outperforms the compared algorithms on the CEC 2010 LSOPs benchmark test set.

Moreover, we further conduct another two statistical tests, i.e., t-test [60] and the Friedman test [21], to further validate the performance of SLPSO-ARS. The results of the t-test and Friedman test with significance level $\alpha = 0.05$ on the CEC 2010 test set are shown in Tables S.II and S.III in the supplementary material, respectively. The p-value in Table S.II in the supplementary material represents whether the performance of SLPSO-ARS is significantly better than the compared algorithms. From Table S.II in the supplementary material, we can see that SLPSO-ARS performs significantly better than the compared algorithms on at least 12 functions among 20 functions in the CEC 2010 test set, while none of the compared algorithms can perform significantly better than SLPSO-ARS on more than seven functions. In Table S.III in the supplementary material, the values in the parentheses represent the performance rank of the algorithms on the corresponding functions by the Friedman test. The p-value represents whether the performance of SLPSO-ARS is significantly better than the compared algorithms. From Table S.III in the supplementary

material, the p-values show that the performance of SLPSO-ARS is significantly better than CSO, DMS-L-PSO, CCPSO2, and DECC-DG. Although the performance of SLPSO-ARS is not significantly different from SLPSO, MLCC, and DECC-DG according to p-values, SLPSO-ARS has the best average rank on the CEC 2010 test set and has the first rank on seven functions, which is the most among all the algorithms. Therefore, the t-test and Friedman test statistical analyses further show that SLPSO-ARS has an overall better performance than the compared algorithms on the CEC 2010 large-scale test set.

Besides, in order to show the efficiency and convergence speed of SLPSO-ARS on the CEC 2010 large-scale benchmark test set, we plot the evolutionary convergence curves of SLPSO-ARS and the compared algorithms on some representative functions, including two completely separable functions (f_1 and f_3), three partially separable functions (f_6, f_{11}, and f_{16}), and one completely nonseparable function (f_{20}). Fig. 2 displays the evolutionary convergence curves of all the algorithms on these 6 functions. As shown in Fig. 2(a), only MLCC, SLPSO, and SLPSO-ARS are able to converge faster and find the better solutions on the completely separable function f_1. Although the final solution found by SLPSO-ARS does not outperform MLCC, the convergence speed of these two algorithms is similar from the convergence curves in Fig. 2(a). On the completely separable function f_3 and partially separable functions f_6 and f_{11} in Fig. 2(b)–(d), SLPSO-ARS achieves better performance than the compared algorithms in both the convergence speed and the accuracy of the final solution, especially on function f_{11}. On partially separable function f_{16} in Fig. 2(e), most of the algorithms fall into local optima prematurely, only CSO, DECC-DG, and SLPSO-ARS can continuously converge and DECC-DG can find the better solution than other algorithms. However, SLPSO-ARS converges faster than DECC-DG in the early stage on this function. In Fig. 2(f), in terms of completely nonseparable function f_{20}, we can see that most algorithms have the similar performance except DMS-L-PSO and DECC-DG, but SLPSO-ARS can find the better results than the compared algorithms.

Furthermore, we present the experimental results of each algorithm in 30 independent runs by box-plots on some representative functions (the same as those functions for evolutionary convergence curves in Fig. 2) in Fig. S.1 in the supplementary material. As shown in Fig. S.1 in the supplementary material, we can see that there are some outliers (denoted by the red "+" symbol) obtained by the compared algorithms, while the proposed SLPSO-ARS algorithm does not obtain any outlier on these six functions. These results show that SLPSO-ARS not only can find better solutions than the compared algorithms, but also can maintain stable search ability in each independent run.

In conclusion, SLPSO-ARS generally outperforms the compared algorithms on the CEC 2010 LSOPs test set.

C. Comparisons With Seven Well-Known Algorithms on the CEC 2013 LSOPs Benchmark Test Set

The main properties of all the LSOPs in the CEC2013 large-scale benchmark test set [59] are presented in Table S.IV in the supplementary material. The CEC 2013 benchmark test set is the newest benchmark test set and is also used in CEC 2020 competition. There are 15 LSOPs in this test set, including three completely separable functions (f_1–f_3), eight partially separable functions (f_4–f_{11}), three overlapping functions (f_{12}–f_{14}), and one completely nonseparable function (f_{15}). The results of SLPSO-ARS and the compared algorithms on the CEC 2013 large-scale benchmark test set are shown in Table II.

Similarly, we first compare SLPSO-ARS with SLPSO. As shown in Table II, for three completely separable functions (f_1–f_3), the experimental results between these two algorithms are similar to the completely separable functions in the CEC 2010 benchmark test set. In terms of eight partially separable functions (f_4–f_{11}), SLPSO-ARS performs better than SLPSO among the five functions (i.e., f_5, f_6, and f_9–f_{11}), while SLPSO can only surpass SLPSO-ARS on three functions (f_4, f_7, and f_8). Especially on function f_{11}, the performance of SLPSO-ARS is much better than SLPSO. For three overlapping functions (f_{12}–f_{14}) and the last completely nonseparable function (f_{15}), the overall performance of SLPSO-ARS is also better than SLPSO.

Overall, among the 15 LSOPs in the CEC 2013 benchmark test set, SLPSO-ARS is significantly better than SLPSO on nine functions, while SLPSO achieves significantly better performance than SLPSO-ARS only on five functions. Therefore, even on the more complex problems (i.e., from the CEC 2010 to the CEC 2013 benchmark test set), the performance of SLPSO-ARS is still better than SLPSO.

In addition to SLPSO, Table II also displays that among the 15 LSOPs in the CEC 2013 large-scale benchmark test set, SLPSO-ARS performs significantly better than CSO, DMS-L-PSO, CCPSO2, DECC-G, MLCC, and DECC-DG on 13, 14, 12, 13, 10, and 15 functions, respectively, while all the compared algorithms cannot perform significantly better than SLPSO-ARS on more than four functions. Therefore, even on the more complex LSOPs in the CEC 2013 benchmark test set, SLPSO-ARS still has outstanding performance.

Moreover, we also conduct t-test [60] and Friedman test [21] to further validate the performance of SLPSO-ARS. The results of t-test and Friedman test with significance level $\alpha = 0.05$ on the CEC 2013 test set are shown in Tables S.V and S.VI in the supplementary material, respectively. From Table S.V in the supplementary material, we can see that SLPSO-ARS performs significantly better than SLPSO, CSO, DMS-L-PSO, CCPSO2, DECC-G, MLCC, and DECC-DG on 8, 13, 15, 12, 13, 11, and 14 functions among the 15 functions in the CEC 2013 test set, respectively, while all the compared algorithms cannot perform significantly better than SLPSO-ARS on more than 5 functions. From Table S.VI in the supplementary material, the p-values show that SLPSO-ARS performs significantly better than CSO, DMS-L-PSO, CCPSO2, DECC-G, and DECC-DG. Although the performance of SLPSO-ARS is not significantly different from SLPSO and MLCC according to the p-values, SLPSO-ARS still has the best average rank on the CEC 2013 test set and has the first rank on six functions, which is also the most among all the algorithms. Therefore, the t-test and Friedman test statistical analyses further show

TABLE IV
PARAMETERS P AND T INVESTIGATION OF SLPSO-ARS ON THE CEC 2010 LARGE-SCALE BENCHMARK TEST SET

Fun	$P=5$				$P=10$			
	$T=5$	$T=10$	$T=15$	$T=20$	$T=5$	$T=10$	$T=15$	$T=20$
	Mean	Mean	Mean	Mean	Mean	Mean	Mean	Mean
f_1	**6.64E-19**	6.90E-19	3.81E-18	5.27E-15	6.68E-19	1.79E-15	1.22E-10	1.31E-07
f_2	2.42E+03	2.06E+03	2.06E+03	1.95E+03	2.94E+03	2.19E+03	1.94E+03	1.92E+03
f_3	**3.51E-13**	3.53E-13	3.38E-12	2.15E-10	3.55E-13	6.77E-11	2.52E-08	9.80E-07
f_4	3.25E+11	4.90E+11	5.31E+11	4.72E+11	**3.48E+11**	6.62E+11	6.39E+11	8.02E+11
f_5	1.19E+07	9.95E+06	1.21E+07	1.31E+07	1.21E+07	8.96E+06	7.37E+06	**5.38E+06**
f_6	**7.32E-08**	7.84E-08	8.52E-08	1.12E-07	8.38E-08	1.23E-07	3.57E-07	3.46E-06
f_7	**1.69E+02**	9.90E+02	4.17E+03	1.24E+04	1.01E+03	1.19E+04	4.35E+04	9.39E+04
f_8	**2.30E+07**	2.73E+07	3.01E+07	4.44E+07	2.72E+07	3.22E+07	3.54E+07	4.80E+07
f_9	7.84E+07	8.50E+07	1.07E+08	1.23E+08	1.45E+08	2.02E+08	1.90E+08	1.89E+08
f_{10}	1.91E+03	1.74E+03	**1.62E+03**	1.68E+03	2.22E+03	1.78E+03	1.78E+03	1.84E+03
f_{11}	3.32E-12	**3.29E-12**	1.17E-10	5.73E-09	3.84E-12	3.35E-09	6.36E-07	4.49E-05
f_{12}	6.50E+04	3.60E+04	2.67E+04	**2.60E+04**	9.07E+04	6.39E+04	5.95E+04	6.72E+04
f_{13}	8.45E+02	7.78E+02	1.04E+03	**5.62E+02**	1.26E+03	9.53E+02	9.87E+02	6.35E+02
f_{14}	3.71E+08	2.84E+08	2.45E+08	**2.39E+08**	4.81E+08	3.98E+08	3.79E+08	3.85E+08
f_{15}	1.99E+03	1.79E+03	1.69E+03	**1.58E+03**	3.06E+03	1.87E+03	1.91E+03	1.79E+03
f_{16}	**5.77E-12**	5.88E-12	1.13E-09	1.45E-07	5.91E-12	1.15E-07	1.69E-05	1.79E-03
f_{17}	3.07E+05	1.78E+05	1.42E+05	**1.36E+05**	3.96E+05	2.88E+05	2.70E+05	2.58E+05
f_{18}	2.28E+03	2.19E+03	2.34E+03	**2.05E+03**	2.26E+03	4.56E+03	1.20E+04	7.82E+03
f_{19}	1.91E+06	1.45E+06	1.31E+06	**1.25E+06**	2.16E+06	1.70E+06	1.66E+06	1.59E+06
f_{20}	9.95E+02	1.03E+03	1.03E+03	9.92E+02	9.90E+02	1.00E+03	1.08E+03	**9.86E+02**
the number of best result	6	1	1	7	1	0	0	2
Fun	$P=15$				$P=20$			
	$T=5$	$T=10$	$T=15$	$T=20$	$T=5$	$T=10$	$T=15$	$T=20$
	Mean	Mean	Mean	Mean	Mean	Mean	Mean	Mean
f_1	2.51E-18	5.10E-11	2.03E-05	7.54E-03	3.92E-14	1.50E-06	7.70E-03	1.33E+00
f_2	2.42E+03	2.03E+03	5.24E+03	6.18E+03	**1.25E+03**	4.79E+03	6.08E+03	6.80E+03
f_3	1.31E-12	1.36E-08	6.08E-06	1.14E-04	2.80E-10	1.62E-06	1.10E-04	1.33E-03
f_4	4.57E+11	7.24E+11	1.00E+12	1.66E+12	6.36E+11	1.13E+12	1.24E+12	1.76E+12
f_5	8.37E+06	7.77E+06	1.33E+07	1.47E+07	1.25E+07	1.43E+07	1.45E+07	1.23E+07
f_6	1.05E-07	3.44E-07	8.26E-05	1.52E-03	2.76E-07	2.13E-05	1.49E-03	1.97E-02
f_7	4.10E+03	4.61E+04	3.03E+05	6.49E+05	2.47E+04	1.54E+05	7.80E+05	1.87E+06
f_8	3.00E+07	4.53E+07	4.90E+07	3.99E+07	3.25E+07	3.73E+07	3.98E+07	4.15E+07
f_9	2.45E+08	2.53E+08	9.79E+07	1.31E+08	**5.48E+07**	8.55E+07	1.29E+08	1.72E+08
f_{10}	2.22E+03	2.07E+03	9.12E+03	9.22E+03	8.91E+03	9.00E+03	9.16E+03	9.36E+03
f_{11}	4.44E-11	7.00E-07	2.19E-04	4.12E-03	1.20E-08	7.19E-05	4.01E-03	7.91E-02
f_{12}	1.26E+05	1.03E+05	8.14E+05	1.07E+06	5.10E+05	7.94E+05	1.02E+06	1.32E+06
f_{13}	7.03E+02	8.00E+02	1.32E+03	1.04E+03	9.01E+02	1.89E+03	9.43E+02	1.59E+03
f_{14}	5.12E+08	4.88E+08	5.14E+08	6.46E+08	2.80E+08	4.43E+08	6.41E+08	8.23E+08
f_{15}	2.32E+03	1.85E+03	1.03E+04	1.03E+04	1.03E+04	1.03E+04	1.03E+04	1.03E+04
f_{16}	2.35E-10	2.61E-05	2.23E-04	1.79E-01	5.53E-09	5.88E-05	2.99E-03	4.63E-02
f_{17}	4.73E+05	4.06E+05	3.03E+06	3.17E+06	2.47E+06	3.00E+06	3.21E+06	3.42E+06
f_{18}	4.02E+03	1.61E+04	1.46E+04	1.54E+04	3.07E+03	1.59E+04	1.47E+04	2.68E+04
f_{19}	2.32E+06	2.03E+06	1.08E+07	1.13E+07	1.04E+07	1.08E+07	1.15E+07	1.19E+07
f_{20}	1.00E+03	1.08E+03	1.09E+03	1.24E+03	1.17E+03	1.08E+03	1.21E+03	2.25E+03
the number of best result	0	0	0	0	2	0	0	0

that SLPSO-ARS has overall better performance than the compared algorithms on the CEC 2013 large-scale test set.

Besides, to show the efficiency and convergence speed of SLPSO-ARS on the CEC 2013 large-scale benchmark test set, we plot the evolutionary convergence curves of SLPSO-ARS and the compared algorithms on some representative functions, including one completely separable function (f_1), three partially separable functions (f_5, f_9, and f_{11}), one overlapping function (f_{13}), and one completely nonseparable function (f_{15}). Fig. 3 shows the evolutionary convergence curves of all the algorithms on these six functions. As shown in Fig. 3(a), the convergence curves on completely separable function f_1 are similar to those in the CEC 2010 benchmark test set. On partially separable function f_5 and overlapping function f_{13}, as shown in Fig. 3(b) and (e), only SLPSO-ARS, CSO, and SLPSO can converge faster and get more accurate results, but SLPSO-ARS still has better performance than CSO and SLPSO. On partially separable function f_9 in Fig. 3(c), CSO is able to find the best results among all algorithms, but SLPSO-ARS converges faster than the compared algorithms

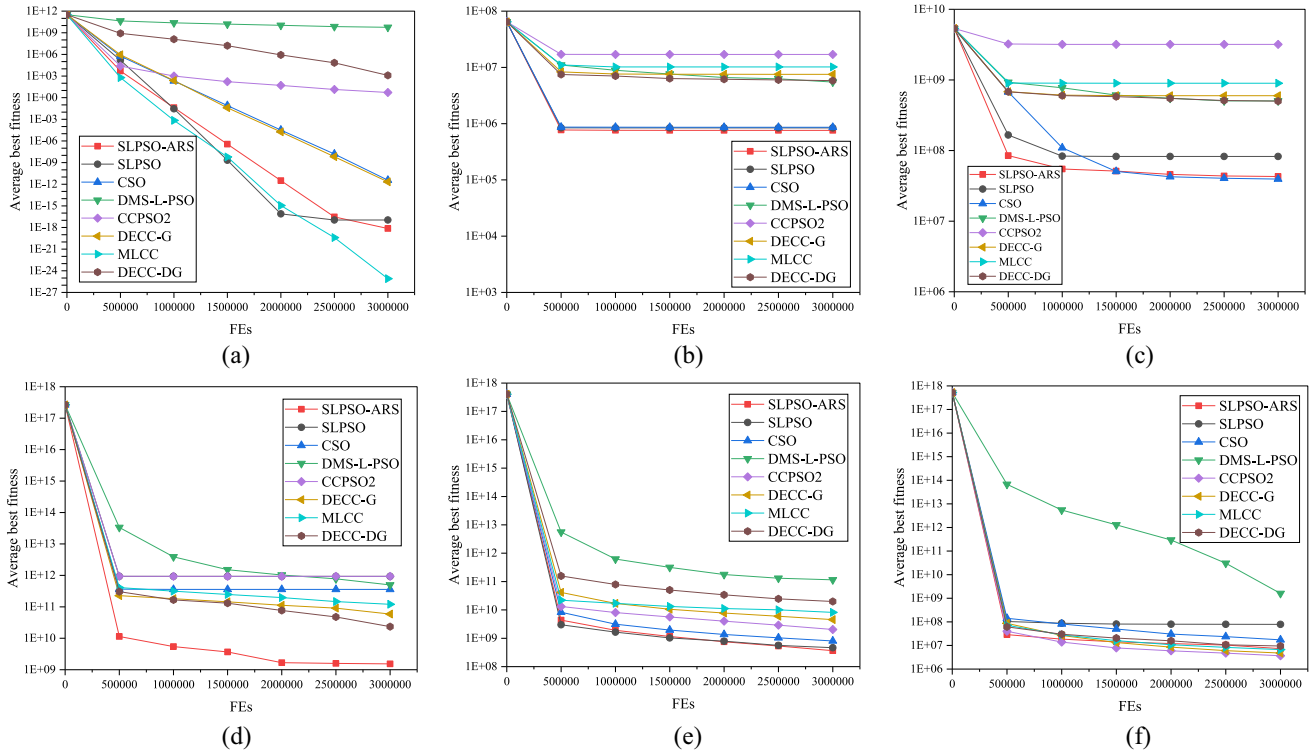


Fig. 3. Convergence curves of SLPSO-ARS and the compared algorithms on six representative functions in the CEC 2013 large-scale benchmark test set. (a) Completely separable function f_1. (b) Partially separable function f_5. (c) Partially separable function f_9. (d) Partially separable function f_{11}. (e) Overlapping function f_{13}. (f) Completely nonseparable function f_{15}.

in the early stage. In Fig. 3(d), in terms of partially separable function f_{11}, SLPSO-ARS performs better than the compared algorithms in both the convergence speed and the accuracy of the final solution. On completely nonseparable function f_{15} in Fig. 3(f), most of the algorithms have the similar performance except DMS-L-PSO. Although CCPSO2 can find the best solution on function f_{15}, the convergence curve of SLPSO-ARS is very close to CCPSO2.

Furthermore, we also present the experimental results of each algorithm in 30 independent runs by box-plots on some representative functions (the same as those functions for evolutionary convergence curves in Fig. 3) in Fig. S.2 in the supplementary material. As shown in Fig. S.2 in the supplementary material, SLPSO-ARS also obtains some outliers (denoted by the red "+" symbol) since CEC 2013 is a more complex and difficult test set than CEC 2010. However, SLPSO-ARS still obtains overall better results than the compared algorithms.

In conclusion, SLPSO-ARS still generally performs better than the compared algorithms on the CEC 2013 LSOPs benchmark test set.

D. Comparisons With the Winning Algorithm in the CEC 2010 Competition

Molina *et al.* [33] proposed a memetic algorithm with multiple local search strategies, which was the winning algorithm in the CEC 2010 competition, named MA-SW-Chains. In order to further highlight the superiority of SLPSO-ARS, we compare SLPSO-ARS with MA-SW-Chains algorithm. The experimental results between SLPSO-ARS and MA-SW-Chains on the CEC 2010 LSOPs benchmark test set are shown in Table III. Note that the experimental results of MA-SW-Chains are directly referred to the original paper [33].

As shown in Table III, among the 20 LSOPs in the CEC 2010 large-scale benchmark test set, SLPSO-ARS achieves better performance than MA-SW-Chains on 11 functions, while MA-SW-Chains dominates SLPSO-ARS only on nine functions. Especially on functions f_1, f_6, f_{11}, and f_{16}, SLPSO-ARS performs much better than MA-SW-Chains, where SLPSO-ARS is more than five orders of magnitude better than MA-SW-Chains. Even on these nine functions, where SLPSO-ARS has worse performance than MA-SW-Chains, the experimental results of SLPSO-ARS on these functions are very close to MA-SW-Chains except functions f_{12} and f_{17}.

Overall, compared with MA-SW-Chains which is the winning algorithm on the CEC 2010 test set, SLPSO-ARS is still very competitive and slightly better than MA-SW-Chains.

E. Parameters and Components Investigation in SLPSO-ARS

1) Investigation on the Number of Particles for ARS P and RS Times T: The ARS strategy has two important parameters, one is the number of (i.e., the top P) particles for ARS and the other is RS times (i.e., T) of each best particle. If P and T are set too small, the local search of the best particles may not be sufficient to find the better solutions. If P and T are set too large, a large number of FEs may be consumed. Therefore, we need to investigate the impact of these two parameters in SLPSO-ARS. We set a series of combinations of P and T where P and T can be 5, 10, 15, or 20. Therefore, there

TABLE V
COMPARISON BETWEEN SLPSO-ARS WITH SLPSO-ARS-*R* ON THE CEC 2010 LARGE-SCALE BENCHMARK TEST SET

Fun	SLPSO-ARS	SLPSO-ARS-*R*					
		$R = 1/2$	$R = 1/5$	$R = 1/10$	$R = 1/20$	$R = 1/50$	$R = 1/100$
	Mean	Mean	Mean	Mean	Mean	Mean	Mean
f_1	**6.64E-19**	2.74E-18(+)	2.30E-18(+)	2.69E-18(+)	2.33E-18(+)	2.62E-18(+)	2.77E-18(+)
f_2	2.42E+03	**9.15E+02(-)**	5.57E+03(+)	4.47E+03(+)	4.66E+03(+)	3.27E+03(+)	1.85E+03(-)
f_3	3.51E-13	3.54E-13(+)	3.51E-13(≈)	3.56E-13(+)	**3.50E-13(-)**	3.52E-13(+)	3.56E-13(+)
f_4	3.25E+11	4.30E+11(+)	3.28E+11(+)	4.79E+11(+)	**2.97E+11(-)**	3.54E+11(+)	3.85E+11(+)
f_5	1.19E+07	1.15E+07(-)	1.45E+07(+)	1.06E+07(-)	1.17E+07(-)	1.11E+07(-)	**1.04E+07(-)**
f_6	**7.32E-08**	1.95E-07(+)	1.86E-07(+)	1.95E-07(+)	1.91E-07(+)	1.97E-07(+)	2.02E-07(+)
f_7	1.69E+02	1.19E+03(+)	**1.64E+02(-)**	1.16E+03(+)	1.70E+02(+)	8.95E+02(+)	6.88E+02(+)
f_8	**2.30E+07**	3.81E+07(+)	3.28E+07(+)	2.71E+07(+)	**2.30E+07(≈)**	3.63E+07(+)	3.63E+07(+)
f_9	7.84E+07	4.28E+07(-)	**3.27E+07(-)**	5.70E+07(-)	4.44E+07(-)	1.17E+08(+)	1.14E+08(+)
f_{10}	**1.91E+03**	9.50E+03(+)	8.22E+03(+)	6.36E+03(+)	5.41E+03(+)	3.27E+03(+)	2.11E+03(+)
f_{11}	**3.32E-12**	4.60E-12(+)	3.52E-12(+)	4.25E-12(+)	3.51E-12(+)	4.52E-12(+)	4.47E-12(+)
f_{12}	6.50E+04	4.28E+05(+)	1.06E+06(+)	6.83E+05(+)	4.11E+05(+)	1.41E+05(+)	**6.18E+04(-)**
f_{13}	8.45E+02	1.05E+03(+)	1.17E+03(+)	9.48E+02(+)	7.61E+02(-)	**7.04E+02(-)**	9.42E+02(+)
f_{14}	3.71E+08	**1.84E+08(-)**	1.94E+08(-)	3.04E+09(+)	1.80E+09(+)	6.40E+08(+)	3.60E+08(-)
f_{15}	**1.99E+03**	1.01E+04(+)	9.27E+03(+)	7.74E+03(+)	6.31E+03(+)	3.27E+03(+)	2.18E+03(+)
f_{16}	**5.77E-12**	6.72E-12(+)	6.45E-12(+)	6.65E-12(+)	6.60E-12(+)	6.59E-12(+)	6.69E-12(+)
f_{17}	3.07E+05	3.09E+06(+)	2.94E+06(+)	1.84E+06(+)	1.27E+06(+)	4.84E+05(+)	**2.35E+05(-)**
f_{18}	2.28E+03	2.00E+03(-)	2.02E+03(-)	2.97E+03(+)	2.13E+03(-)	1.61E+03(-)	**1.59E+03(-)**
f_{19}	1.91E+06	9.01E+06(+)	6.97E+06(+)	4.69E+06(+)	3.90E+06(+)	2.37E+06(+)	**1.67E+06(-)**
f_{20}	9.95E+02	1.04E+03(+)	1.09E+03(+)	1.04E+03(+)	1.11E+03(+)	**9.87E+02(-)**	1.01E+03(+)
+(SLPSO-ARS is better)		15	15	18	13	16	13
-(SLPSO-ARS is worse)		5	4	2	6	4	7
≈(no difference)		0	1	0	1	0	0

are 16 (4×4) SLPSO-ARS variants with different P and T values. Note that each SLPSO-ARS variant is named SLPSO-ARS(x, y), where x and y stand for the value of P and T, respectively. For example, assuming that in one of the SLPSO-ARS variants, P and T are set to 5 and 10, respectively, then this SLPSO-ARS variant is named SLPSO-ARS(5, 10).

We adopt the CEC 2010 LSOPs benchmark test set to test the performance of these 16 SLPSO-ARS variants. The mean results are shown in Table IV. Moreover, the number of functions which obtain the best result for each SLPSO-ARS variant is given at the end of the table.

From Table IV, we can see that when RS times T is fixed, the larger the P is used, the worse the performance of SLPSO-ARS variants is. Therefore, among the four values mentioned above, setting the parameter P to 5 is the best choice for SLPSO-ARS. When the parameter P is fixed, a small value of T is more effective in solving completely separable functions (e.g., f_1–f_3) and some simple partially separable functions (e.g., f_4–f_{11}), while a large value of T is more suitable for solving some complex partially separable functions (e.g., f_{12}–f_{18}) and completely nonseparable functions (e.g., f_{19} and f_{20}). This may be due to that the population easily falls into the local optima when solving complex functions, so increasing the RS times of best particles can help the population jump out of the local optima more effectively. Moreover, from the last row of statistical results in Table IV, SLPSO-ARS(5, 5) and SLPSO-ARS(5, 20) have more best-performing functions than other SLPSO-ARS variants, with six and seven, respectively. Although SLPSO-ARS(5, 5) performs worse than SLPSO-ARS(5, 20) on some complex partially separable functions (e.g., f_{12}–f_{15}, f_{17}, and f_{18}) and completely nonseparable functions (e.g., f_{19} and f_{20}), the results of SLPSO-ARS(5, 5) on these functions are very close to SLPSO-ARS(5, 20). However, SLPSO-ARS(5, 5) is much better than SLPSO-ARS(5, 20) on other functions, especially on functions f_1, f_3, f_{11}, and f_{16}. Therefore, both parameters P and T are set to 5 in SLPSO-ARS.

2) Effect of Adaptive Region Radius r: One of the innovations in the ARS strategy in SLPSO-ARS is that each particle has its own region radius r, and the region radius r of each particle is adaptively adjusted during the evolutionary process. Therefore, the adaptive region radius r allows different particles to have the most suitable region radius r in different evolutionary state or for different problems. In order to validate the effectiveness of the adaptive region radius r, we compare SLPSO-ARS with adaptive region radius r with some SLPSO-ARS variants with fixed region radius r, which are denoted as SLPSO-ARS-R. Note that the value of R represents that the region radius r of each particle is fixed to *(ubound-lbound)* $\times$ R, where *ubound* and *lbound* are the upper and lower bounds of the problem space, respectively. For example, if $R = 1/2$, then the region radius r is set to *(ubound-lbound)* $\times$ 1/2, and r remains unchanged throughout the evolutionary process.

Similarly, we adopt the CEC 2010 LSOPs benchmark test set to test the performance of SLPSO-ARS-R. Table V shows the experimental results and we only present the mean values.

From Table V, we can see that among the 20 LSOPs in the CEC 2010 benchmark test set, SLPSO-ARS performs better than SLPRO-RS-1/2, SLPSO-ARS-1/5, SLPSO-ARS-1/10, SLPSO-ARS-1/20, SLPSO-ARS-1/50, and SLPSO-ARS-1/100 on 15, 15, 18, 13, 16, and 13 functions, respectively, while all the SLPSO-ARS-R variants cannot

TABLE VI
COMPARISON BETWEEN SLPSO-ARS WITH SLPSO-ARS-WITHOUT-$R_{\max}$ ON THE CEC 2010 LARGE-SCALE BENCHMARK TEST SET

Fun	SLPSO-ARS	SLPSO-ARS-without-$r_{\max}$
	Mean±Std	Mean±Std
f_1	**6.64E-19±2.54E-20**	7.57E-19±3.94E-20(+)
f_2	**2.42E+03±9.11E+01**	2.73E+03±2.02E+02(+)
f_3	**3.51E-13±4.82E-15**	**3.51E-13±4.14E-15(≈)**
f_4	**3.25E+11±4.93E+10**	4.77E+11±1.04E+11(+)
f_5	**1.19E+07±2.67E+06**	1.23E+07±2.40E+06(+)
f_6	**7.32E-08±2.98E-09**	7.53E-08±2.40E-09(+)
f_7	**1.69E+02±3.62E+01**	1.26E+03±6.47E+02(+)
f_8	**2.30E+07±2.50E+05**	2.73E+07±2.24E+05(+)
f_9	**7.84E+07±2.68E+06**	8.31E+07±5.16E+06(+)
f_{10}	**1.91E+03±2.34E+02**	2.27E+03±2.36E+02(+)
f_{11}	**3.32E-12±7.38E-14**	3.58E-12±6.43E-13(+)
f_{12}	6.50E+04±5.89E+03	**3.62E+04±1.62E+03(-)**
f_{13}	**8.45E+02±4.78E+02**	1.31E+03±7.54E+02(+)
f_{14}	3.71E+08±2.06E+07	**2.96E+08±3.12E+07(-)**
f_{15}	1.99E+03±1.76E+02	**1.92E+03±7.84E+01(-)**
f_{16}	5.77E-12±1.22E-13	**5.23E-12±4.94E-13(-)**
f_{17}	3.07E+05±1.47E+04	**2.08E+05±1.28E+04(-)**
f_{18}	**2.28E+03±9.66E+02**	2.54E+03±1.13E+03(+)
f_{19}	1.91E+06±5.03E+04	**1.49E+06±4.40E+04(-)**
f_{20}	**9.95E+02±2.51E+01**	1.04E+03±5.70E+01(+)
+(SLPSO-ARS is better)		13
-(SLPSO-ARS is worse)		6
≈(no difference)		1

outperform SLPSO-ARS for more than seven functions. Therefore, SLPSO-ARS with adaptive region radius r has better performance than some SLPSO-ARS variants with fixed region radius r, which validates the effectiveness of the adaptive region radius r.

3) Effect of r_{max}: In the ARS strategy, we set $r_{\max}$ to limit the value of the region radius r and prevent it from expanding infinitely. Moreover, $r_{\max}$ decreases linearly during the evolutionary process. To validate the effectiveness of $r_{\max}$, we compare SLPSO-ARS with SLPSO-ARS variant which is without $r_{\max}$, and denoted as SLPSO-ARS-without-$r_{\max}$.

We also use the CEC 2010 LSOPs benchmark test set to test the performance of SLPSO-ARS-without-$r_{\max}$. The experimental results are presented in Table VI.

From Table VI, we can see that among the 20 LSOPs, SLPSO-ARS achieves better performance than SLPSO-ARS-without-$r_{\max}$ on 13 functions, while SLPSO-ARS-without-$r_{\max}$ can only surpass SLPSO-ARS on six functions. In addition, SLPSO-ARS-without-$r_{\max}$ performs better in solving some complex partially separable functions and completely nonseparable functions, such as functions f_{12}, f_{14}–f_{17}, and f_{19}. When solving such functions, it is easy to get trapped in the local optimal solutions. If without the limit of $r_{\max}$, the region radius r will expand infinitely when the better solutions can be found during the RS, making the population more likely to jump out of the region near the local optimal solutions. However, SLPSO-ARS has a significant advantage in solving completely separable functions and some simple partially separable functions, such as functions f_1–f_{11}, and the overall performance of SLPSO-ARS is better than SLPSO-ARS-without-$r_{\max}$ on the CEC 2010 LSOPs benchmark test set. Therefore, this comparison experiment further shows the effectiveness of the $r_{\max}$.

V. CONCLUSION

In this article, we proposed a novel region-based encoding scheme to extend the solution from a single point to a region, which can help the algorithm evolve faster if the region information can be well utilized. To this aim, the new local search strategy, named ARS, was proposed based on the RES to improve SLPSO, forming the SLPSO-ARS. In SLPSO-ARS, some of the best (i.e., the top P) particles will execute RS at the end of every generation to search the better solutions near their current positions. The ARS strategy not only can accelerate the convergence speed of the population but also have a greater chance to find the nearby optimal solutions. In addition, we adopt both the CEC 2010 and CEC 2013 LSOPs benchmark test sets to compare SLPSO-ARS with some well-known large-scale optimization algorithms. The experimental results showed that SLPSO-ARS achieved generally better performance than the compared large-scale optimization algorithms in terms of the accuracy of the final solution and the convergence speed. Moreover, we also investigated some important parameters of SLPSO-ARS and validated the effectiveness of its related components.

As the RES and the ARS are the generic framework and strategy that may be used in different EC algorithms, in the future, we hope to apply the RES framework and the ARS strategy to other EC algorithms and use them to solve real-world optimization problems.

REFERENCES

[1] T. Blackwell and J. Kennedy, "Impact of communication topology in particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 689–702, Aug. 2019.

[2] X. F. Liu, Z. H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 4, pp. 587–602, Aug. 2019.

[3] J. Y. Li, Z.-H. Zhan, R. D. Liu, C. Wang, S. Kwong, and J. Zhang, "Generation-level parallelism for evolutionary computation: A pipeline-based parallel particle swarm optimization," *IEEE Trans. Cybern.*, early access, Nov. 4, 2020, doi: 10.1109/TCYB.2020.3028070.

[4] S. Chen, A. B. Röhler, J. Montgomery, and T. Hendtlass, "An analysis on the effect of selection on exploration in particle swarm optimization and differential evolution," in *Proc. IEEE Congr. Evol. Comput.*, 2019, pp. 3037–3044.

[5] X. Xia *et al.*, "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4862–4875, Dec. 2020.

[6] Z. G. Chen *et al.*, "Multiobjective cloud workflow scheduling: A multiple populations ant colony system approach," *IEEE Trans. Cybern.*, vol. 49, no. 8, pp. 2912–2926, Aug. 2019.

[7] S. Z. Zhou, Z. H. Zhan, Z. G. Chen, S. Kwong, and J. Zhang, "A multi-objective ant colony system algorithm for airline crew rostering problem with fairness and satisfaction," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 8, 2020, doi: 10.1109/TITS.2020.2994779.

[8] M. Mavrovouniotis, F. M. Muller, and S. Yang, "Ant colony optimization with local search for dynamic traveling salesman problems," *IEEE Trans. Cybern.*, vol. 47, no. 7, pp. 1743–1756, Jul. 2017.

[9] D. Liang, Z.-H. Zhan, Y. Zhang, and J. Zhang, "An efficient ant colony system approach for new energy vehicle dispatch problem," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 11, pp. 4784–4797, Nov. 2020.

[10] J.-Y. Li, Z.-H. Zhan, C. Wang, H. Jin, and J. Zhang, "Boosting data-driven evolutionary algorithm with localized data generation," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 923–937, Oct. 2020.

[11] A. Mohammadi, H. Asadi, S. Mohamed, K. Nelson, and S. Nahavandi, "Multiobjective and interactive genetic algorithms for weight tuning of a model predictive control-based motion cueing algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3471–3481, Sep. 2019.

[12] X.-Y. Zhang, J. Zhang, Y.-J. Gong, Z.-H. Zhan, W.-N. Chen, and Y. Li, "Kuhn-Munkres parallel genetic algorithm for the set cover problem and its application to large-scale wireless sensor networks," *IEEE Trans Evol. Comput.*, vol. 20, no. 5, pp. 695–710, Oct. 2016.

[13] J. Y. Li, Z. H. Zhan, H. Wang, and J. Zhang, "Data-driven evolutionary algorithm with perturbation-based ensemble surrogates," *IEEE Trans. Cybern.*, early access, Aug. 10, 2020, doi: 10.1109/TCYB.2020.3008280.

[14] B. Doerr and M. S. Krejca, "Significance-based estimation-of-distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1025–1034, Dec. 2020.

[15] Z. G. Chen, Y. Lin, Y. J. Gong, Z. H. Zhan, and J. Zhang, "Maximizing lifetime of range-adjustable wireless sensor networks: A neighborhood-based estimation of distribution algorithm," *IEEE Trans. Cybern.*, early access, Apr. 1, 2020, doi: 10.1109/TCYB.2020.2977858.

[16] P. Verma, K. Sanyal, D. Srinivsan, and K. S. Swarup, "Information exchange based clustered differential evolution for constrained generation-transmission expansion planning," *Swarm Evol. Comput.*, vol. 44, pp. 863–875, Feb. 2019.

[17] Z.-H. Zhan, Z. J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020.

[18] Z.-G. Chen, Z.-H. Zhan, H. Wang, and J. Zhang, "Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 708–719, Aug. 2020.

[19] Z.-H. Zhan *et al.*, "Cloudde: A heterogeneous differential evolution algorithm and its distributed cloud version," *IEEE Trans. Parallel Distrib. Syst.*, vol. 28, no. 3, pp. 704–716, Mar. 2017.

[20] H. Zhao *et al.*, "Local binary pattern-based adaptive differential evolution for multimodal optimization problems," *IEEE Trans. Cybern.*, vol. 50, no. 7, pp. 3343–3357, Jul. 2020.

[21] A. K. Ball, S. S. Roy, D. R. Kisku, N. C. Murmu, and L. D. S. Coelho, "Optimization of drop ejection frequency in EHD inkjet printing system using an improved firefly algorithm," *Appl. Soft Comput.*, vol. 94, Sep. 2020, Art. no. 106438.

[22] E. H. D. V. Segundo, V. C. Mariani, and L. D. S. Coelho, "Design of heat exchangers using falcon optimization algorithm," *Appl. Thermal Eng.*, vol. 156, pp. 119–144, Jun. 2019.

[23] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, 1995, pp. 1942–1948.

[24] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE World Congr. Comput. Intell.*, 1998, pp. 69–73.

[25] Y. Guo, J. Y. Li, and Z. H. Zhan, "Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approach," *Cybern. Syst.*, vol. 52, no. 1, pp. 36–57, Oct. 2020.

[26] Y. Lin, Y. S. Jiang, Y. J. Gong, Z. H. Zhan, and J. Zhang, "A discrete multiobjective particle swarm optimizer for automated assembly of parallel cognitive diagnosis tests," *IEEE Trans. Cybern.*, vol. 49, no. 7, pp. 2792–2805, Jul. 2019.

[27] X. F. Liu *et al.*, "Neural network-based information transfer for dynamic optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 5, pp. 1557–1570, May 2020.

[28] Q. Lin *et al.*, "Particle swarm optimization with a balanceable fitness estimation for many-objective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 32–46, Feb. 2018.

[29] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Inf. Sci.*, vol. 178, no. 15, pp. 2985–2999, 2008.

[30] R. Cheng and Y. C. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Inf. Sci.*, vol. 291, pp. 43–60, Jan. 2015.

[31] S. Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 3845–3852.

[32] D. Molina and F. Herrera, "Iterative hybridization of DE with local search for the CEC'2015 special session on large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2015, pp. 1974–1978.

[33] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[34] J. Zhang, J. H. Zhong, and X. M. Hu, "A novel genetic algorithm with orthogonal prediction for global numerical optimization," in *Proc. Conf. Simulat. Evol. Learn.*, 2008, pp. 31–40.

[35] J. H. Zhong and J. Zhang, "Adaptive multi-objective differential evolution with stochastic coding strategy," in *Proc. Conf. Genet. Evol. Comput.*, 2011, pp. 665–672.

[36] J. R. Jian, Z. H. Zhan, and J. Zhang, "Large-scale evolutionary optimization: A survey and experimental comparative study," *Int. J. Mach. Learn. Cybern.*, vol. 11, no. 3, pp. 729–745, Mar. 2020.

[37] M. A. Potter and K. A. D. Jong, "A cooperative coevolutionary approach to function optimization," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 1994, pp. 249–257.

[38] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 225–239, Jun. 2004.

[39] X. Li and X. Yao, "Cooperatively coevolving particle swarms for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 16, no. 2, pp. 210–224, Apr. 2012.

[40] Y. Shi, H. Teng, and Z. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Proc. Int. Conf. Nat. Comput.*, 2005, pp. 1080–1088.

[41] Z. Yang, K. Tang, and X. Yao, "Multilevel cooperative coevolution for large scale optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 1663–1670.

[42] M. N. Omidvar, X. Li, and X. Yao, "Cooperative co-evolution with delta grouping for large scale non-separable function optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1762–1769.

[43] M. Omidvar, X. Li, Y. Mei, and X. Yao, "Cooperative co-evolution with differential grouping for large scale optimization," *IEEE Trans. Evol. Comput.*, vol. 18, no. 3, pp. 378–393, Jun. 2014.

[44] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended differential grouping for large scale global optimization with direct and indirect variable interactions," in *Proc. Conf. Genet. Evol. Comput.*, 2015, pp. 313–320.

[45] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization," *ACM Trans. Math. Softw.*, vol. 42, no. 2, pp. 1–24, 2016.

[46] M. N. Omidvar, B. Kazimipour, X. Li, and X. Yao, "CBCC3—A contribution-based cooperative co-evolutionary algorithm with improved exploration/exploitation balance," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3541–3548.

[47] M. Yang *et al.*, "Efficient resource allocation in cooperative co-evolution for large-scale global optimization," *IEEE Trans. Evol. Comput.*, vol. 21, no. 4, pp. 493–505, Aug. 2017.

[48] B. Kazimipour, M. N. Omidvar, A. K. Qin, X. Li, and X. Yao, "Bandit-based cooperative coevolution for tackling contribution imbalance in large-scale optimization problems," *Appl. Soft Comput.*, vol. 76, pp. 265–281, Mar. 2019.

[49] X. Zhang, K.-J. Du, Z.-H. Zhan, S. Kwong, T.-L. Gu, and J. Zhang, "Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with uncertainties," *IEEE Trans. Cybern.*, vol. 50, no. 10, pp. 4454–4468, Oct. 2020.

[50] T. Takahama and S. Sakai, "Large scale optimization by differential evolution with landscape modality detection and a diversity archive," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 2842–2849.

[51] J. Brest, B. Boškovic, A. Zamuda, I. Fister, and M. S. Maucec, "Self-adaptive differential evolution algorithm with a small and varying population size," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1–8.

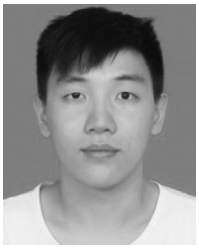
[52] J. Brest and M. S. Maucec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Comput.*, vol. 15, no. 11, pp. 2157–2174, 2011.

[53] Q. Yang, W. N. Chen, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 578–594, Aug. 2018.

[54] R. Cheng and Y. C. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Trans. Cybern.*, vol. 45, no. 2, pp. 191–204, Feb. 2015.

[55] Z.-J. Wang *et al.*, "Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling," *IEEE Trans. Cybern.*, vol. 50, no. 6, pp. 2715–2729, Jun. 2020.

[56] Z. J. Wang, Z. H. Zhan, S. Kwong, H. Jin, and J. Zhang, "Adaptive granularity learning distributed particle swarm optimization for large-scale optimization," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1175–1188, Mar. 2021.
[57] Y.-F. Ge *et al.*, "Distributed differential evolution based on adaptive mergence and split for large-scale optimization," *IEEE Trans. Cybern.*, vol. 48, no. 7, pp. 2166–2180, Jul. 2018.
[58] K. Tang, X. Li, P. N. Suganthan, Z. Yang, and T. Weise, "Benchmark functions for the CEC'2010 special session and competition on large-scale global optimization," Nat. Inspired Comput. Appl. Lab., Univ. Sci. Technol. China, Anhui, China, Rep., 2010. [Online]. Available: https://titan.csit.rmit.edu.au e46507/publications/lsgo-cec10.pdf
[59] X. Li, K. Tang, M. N. Omidvar, Z. Yang, and K. Qin, "Benchmark functions for the CEC'2013 special session and competition on large-scale global optimization," Evol. Comput. Mach. Learn. Group, RMIT Univ., Melbourne, VIC, Australia, Rep., 2013. [Online]. Available: https://www.tflsgo.org/assets/cec2018/cec2013-lsgo-benchmark-tech-report.pdf
[60] J. Carrasco, S. Garcia, M. M. Rueda, S. Das, and F. Herrera, "Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review," *Swarm Evol. Comput.*, vol. 54, May 2020, Art. no. 100665.



Jun-Rong Jian (Student Member, IEEE) received the B.S. degree in computer science and technology from the South China University of Technology, Guangzhou, China, in 2019, where he is currently pursuing the M.S. degree.

His current research interests include evolutionary algorithms, swarm intelligence algorithms, and their applications in large-scale real-world optimization problems.

Zong-Gan Chen (Student Member, IEEE) received the B.S. degree from Sun Yat-sen University, Guangzhou, China, in 2016. He is currently pursuing the Ph.D. degree in computer science and technology with the South China University of Technology, Guangzhou.

His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world optimization problems.

Zhi-Hui Zhan (Senior Member, IEEE) received the bachelor's and Ph.D. degrees in computer science from Sun Yat-sen University, Guangzhou, China, in 2007 and 2013, respectively.

He is currently the Changjiang Scholar Young Professor and the Pearl River Scholar Young Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation algorithms, swarm intelligence algorithms, and their applications in real-world problems, and in environments of cloud computing and big data.

Dr. Zhan was a recipient of the Outstanding Youth Science Foundation from National Natural Science Foundations of China in 2018, the Wu Wen-Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017, and the Doctoral Dissertation was Awarded the IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation and the China Computer Federation Outstanding Ph.D. Dissertation. He is listed as one of the Most Cited Chinese Researchers in Computer Science. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and *Neurocomputing*.

Jun Zhang (Fellow, IEEE) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Visiting Scholar with Chaoyang University of Technology, Taichung, Taiwan, and Victoria University, Melbourne, VIC, Australia. His current research interests include computational intelligence, cloud computing, high-performance computing, operations research, and power electronic circuits.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011, and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON CYBERNETICS.