Yang, Qiang and Chen, Wei Neng and Gu, Tianlong and Zhang, Huaxiang and Deng, Jeremiah D. and Li, Yun and Zhang, Jun (2017) Segment-based predominant learning swarm optimizer for large-scale optimization. IEEE Transactions on Cybernetics, 47 (9). pp. 2896-2910. ISSN 2168-2275 , http://dx.doi.org/10.1109/TCYB.2016.2616170

Segment-Based Predominant Learning Swarm Optimizer for Large-Scale Optimization

Qiang Yang, *Student Member, IEEE*, Wei-Neng Chen, *Member, IEEE*, Tianlong Gu, Huaxiang Zhang, Jeremiah D. Deng, *Member, IEEE*, Yun Li, *Member, IEEE*, and Jun Zhang, *Senior Member, IEEE*

***Abstract*—Large-scale optimization has become a significant yet challenging area in evolutionary computation. To solve this problem, this paper proposes a novel segment-based predominant learning swarm optimizer (SPLSO) swarm optimizer through letting several predominant particles guide the learning of a particle. First, a segment-based learning strategy is proposed to randomly divide the whole dimensions into segments. During update, variables in different segments are evolved by learning from different exemplars while the ones in the same segment are evolved by the same exemplar. Second, to accelerate search speed and enhance search diversity, a predominant learning strategy is also proposed, which lets several predominant particles guide the update of a particle with each predominant particle responsible for one segment of dimensions. By combining these two learning strategies together, SPLSO evolves all dimensions simultaneously and possesses competitive exploration and exploitation abilities. Extensive experiments are conducted on two large-scale benchmark function sets to investigate the influence of each algorithmic component and comparisons with several state-of-the-art meta-heuristic algorithms dealing with large-scale problems demonstrate the competitive efficiency and effectiveness of the proposed optimizer. Further the scalability of the optimizer to solve problems with dimensionality up to 2000 is also verified.**

***Index Terms*—Global numerical optimization, large-scale optimization, particle swarm optimization (PSO), segment-based predominant learning swarm optimizer (SPLSO).**

Manuscript received March 22, 2016; revised July 4, 2016; accepted October 3, 2016. Date of publication October 24, 2016; date of current version August 16, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61379061, and Grant 61332002, in part by the Natural Science Foundation of Guangdong under Grant 2015A030306024, and in part by the Guangdong Special Support Program under Grant 2014TQ01X550. This paper was recommended by Associate Editor Y. Tan. *(Corresponding authors: Wei-Neng Chen; Jun Zhang.)*

Q. Yang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 51006, China, and also with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou 510006, China.

W.-N. Chen and J. Zhang are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 51006, China (e-mail: cwnraul634@aliyun.com; junzhang@ieee.org).

T. Gu is with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin 541004, China.

H. Zhang is with the School of Information Science and Engineering, Shandong Normal University, Jinan 250014, China.

J. D. Deng is with the Department of Information Science, University of Otago, Dunedin 9054, New Zealand.

Y. Li is with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan 523808, China.

This paper has supplementary downloadable multimedia material available at http://ieeexplore.ieee.org provided by the authors. This includes a PDF file which contains additional tables and figures not included within the paper itself. The total size of the file is 1.048 MB.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2016.2616170

I. Introduction

EVOLUTIONARY optimization has witnessed great success in many optimization problems [1]–[6] and has been applied to many real-world applications [7]–[12] in recent years. Owing to its easiness of understanding and implementation, a lot of attention has been devoted to evolutionary algorithm (EA) researches and subsequently many different EAs have been developed, such as particle swarm optimization (PSO) algorithms [13]–[21], differential evolution (DE) algorithms [22], [23], ant colony optimization (ACO) algorithms [24], [25], estimation of distribution algorithms [26], [27], firefly algorithms [5], [28], etc. In particular, since PSO was first developed by Eberhart and Kennedy [17], [18], an ocean of PSO variants have been proposed, such as clustering-based PSO [29], scatter learning PSO [30], adaptive PSO [13], [31], comprehensive learning PSO (CLPSO) [32], and bare bone PSO [14], [15].

Although traditional EAs have shown their excellent abilities and feasibilities in low-dimensional space (less than 100-D), their performance would deteriorate dramatically when it comes to high-dimensional space [33], which is usually called “the curse of dimensionality.” Such inferior performance can be attributed to the drastic and exponential increase of the search space with the growing dimensionality, forming large and massive local regions that easily cause the search process to stagnate into local optima and result in premature convergence for traditional EAs [33], [34]. Therefore, to tackle large-scale problems (more than 500-D), new evolution or learning strategies need to come up for EAs to escape from being trapped at local optima.

In recent years, works on large-scale optimization problems mainly followed two approaches.

1) Proposing cooperative coevolution (CC) strategies, which mainly concentrate on decomposing dimensions into groups and evolve each dimension group separately.
2) Proposing new learning strategies for traditional EAs evolving all dimensions together, which have great power in enhancing population diversity.

CC-based algorithms adopt the divide-and-conquer strategy to decompose high-dimensional problems into smaller subproblems. Since Potter [35] proposed such a promising CC framework, researchers have developed many variants by utilizing different EAs (CCEAs) for large-scale optimization, such as cooperative coevolutionary genetic algorithm [36]–[38], cooperative coevolutionary PSO (CCPSO) [16], [33], and cooperative coevolutionary

DE (DECC) [39]–[43]. Currently, the research of CCEAs mainly focuses on proposing a good decomposition strategy, which aims to group independent variables into different groups and simultaneously group interdependent variables into the same group. So far, there are four kinds of decomposition strategies [41]: 1) random grouping [40], [42], [43], 2) perturbation grouping [44], [45], 3) interaction adaptation grouping [41], [46], [47], 4) model building grouping [48]. Even though CCEAs are promising for large-scale problems, they encounter three limitations.

1) CCEAs usually need a considerable number of fitness evaluations to obtain satisfactory solutions due to evolving each subproblem separately, especially when the number of dimension groups is large.
2) A good decomposer has to detect dependency among variables by using a certain number of fitness evaluations [41], [45], at the sacrifice of fitness evaluations used for evolving.
3) When the fitness landscape of the problem becomes more and more complicated, e.g., the dynamic variable dependency in piecewise functions, the current dimension grouping strategies embedded in CCEAs would lose their efficiency in detecting the dependency among variables.

The above limitations restrict the wide application of CCEAs when faced with limited resources, such as the limited number of fitness evaluations.

To relieve the above dilemma in CCEAs, from another aspect, researchers attempt to develop new learning strategies for traditional EAs to enhance the diversity, which contribute to promoting the chances of escaping from local optima for the population. Such new learning strategies are usually embedded in the update of particles, which is in the form of learning from potential particles. Along this way, [49]–[54] show their feasibility and efficiency. CMA-ES [49] equips the evolution strategy (ES) with self-adaptive mutation parameters through computing a covariance matrix and correlated step sizes, which takes $O(D^2)$ (D is the dimension size), while sep-CMA-ES [50] is a simple modification of CMA-ES, which reduces the update of the whole covariance matrix to the update of its diagonal components. Recently, a multiswarm PSO based on feedback evolution (FBE) [54] and a competitive swarm optimizer (CSO) [53] were brought up with competitive learning strategies based on pairwise competition between particles, which will be detailed in the next section.

Though these proposed new learning strategies have shown feasibility on large-scale optimization, early-stagnation is still a main challenge for EAs. To prevent from being trapped in local optima, specific diversity enhancement strategies have to be designed to further improve search diversity.

In order to circumvent the dilemmas that CCEAs [16], [35], [41], [42], [45] and traditional EAs [50], [53], [54] are confronted with, this paper intends to take advantage of these two kinds of approaches and develop a novel segment-based predominant learning (SPL) swarm optimizer (SPLSO). More specifically, This paper contains the following components.

1) *Segment-Based Learning (SL):* Inspired from the facts that exemplars made up by combining dimensions from different exemplars would lead to potentially better directions in orthogonal learning PSO (OLPSO) [55] and that learning from different exemplars can greatly enhance the diversity in CLPSO [32], an SL strategy is proposed to learn segments of potentially useful information from different exemplars. First, SL randomly divides dimensions into segments. Then, for each segment of a particle, SL lets one exemplar guide the update of this part. Together, SL allows several exemplars to simultaneously guide the learning of one particle. Through this, on one hand the potentially beneficial evolutionary information embedded in different exemplars may be gathered and learned by particles; on the other hand, the potentially useful information in different dimensions of an exemplar may be preserved and gathered together. In this way, SL performs a new way of coevolution for different dimension segments.
2) *SPL:* Inspired by the competitive learning strategy in CSO [53], this paper further couples SL with a predominant learning strategy (PL), leading to a novel learning strategy called "SPL." SPL divides particles into relatively good ones and poor ones. Then each poor particle performs SL to learn from different good ones. That is, each dimension segment of this particle is updated by randomly selecting a relatively good particle as its exemplar. In this way, SPL not only potentially introduces high diversity as all predominant particles can potentially become exemplars, but also enables an effective interaction among different dimension segments of predominant particles, which may contribute to a fast convergence speed.

To verify the proposed SPLSO, a series of experiments are conducted on widely used CEC'2010 and CEC'2013 large-scale benchmark functions. The experimental results in comparison to state-of-the-art algorithms coping with large-scale optimization demonstrate the competitive efficiency and effectiveness of SPLSO. In addition, the comparison results between SPLSO and CSO [53] on 20 CEC'2010 problems with 2000-D further substantiate the good scalability of SPLSO to higher dimensionality.

The remainder of this paper is organized as follows. Section II introduces the classical PSO and its variants in dealing with large-scale optimization. Then, SPLSO is stated in detail in Section III. In Section IV, experiments are conducted to verify the effectiveness of SPLSO in comparison with five state-of-the-art algorithms. At last, Section V concludes this paper.

II. Related Work

Without loss of generality, a D-dimensional minimization problem considered in this paper can be formulated as

$$\min f(\mathbf{x}), \mathbf{x} \in R^D \tag{1}$$

where D is the dimension size. In this paper, the function value is considered as the fitness value of a particle.

A. PSO

PSO [17], [18] simulates the swarm behaviors of social animals such as bird flocking, and is modeled on an abstract framework of "collective intelligence." Usually, particles in a swarm represent points in the D-dimensional search space and two attributes, namely position and velocity, are assigned to each particle. Suppose the position and velocity of the swarm are denoted as $\mathbf{X}$ and $\mathbf{V}$, with the ith particle identified as $\mathbf{X_i}$ and $\mathbf{V_i}$, respectively, and then the update of these two attributes are

$$v_i^d \leftarrow wv_i^d + c_1r_1\left(\text{pbest}_i^d - x_i^d\right) + c_2r_2\left(\text{gbest}^d - x_i^d\right) \tag{2}$$

$$x_i^d \leftarrow x_i^d + v_i^d \tag{3}$$

where $\mathbf{X_i} = [x_i^1, \ldots, x_i^d, \ldots, x_i^D]$ is the position of the ith particle, and $\mathbf{V_i} = [v_i^1, \ldots, v_i^d, \ldots, v_i^D]$ is its velocity, D is the dimension size, $\mathbf{pbest_i} = [\text{pbest}_i^1, \ldots, \text{pbest}_i^d, \ldots, \text{pbest}_i^D]$ is the personal best position of the ith particle, and $\mathbf{gbest} = [\text{gbest}^1, \ldots, \text{gbest}^d, \ldots, \text{gbest}^D]$ is the global best position of the whole swarm. Among parameters, c_1 and c_2 are two acceleration coefficients [17], r_1 and r_2 are uniformly randomized within $[0, 1]$, and w is termed as the inertia weight [56]. Kennedy and Eberhart [18] referred to the second and the third part in the right of (2) as the cognitive component and the social component, respectively.

An outstanding characteristic of PSO is the fast convergent behavior and inherent adaptability. Theoretical analysis of PSO [57] has shown that particles in a swarm can keep a balance between exploration and exploitation. However, due to the strong influence of the global best position **gbest** on the convergence speed [33], premature convergence remains a major issue in PSO. Thus, some researchers proposed to utilize the neighbor best position [17], [57], [58] to replace the global best position in updating the velocity of each particle.

Further, some researchers even put forward strategies to construct new exemplars to lead the learning direction. Along this line, two influential representatives are OLPSO [55] and CLPSO [32]. OLPSO constructs a new exemplar by combining dimensions from the personal best position and the neighbor best position of each particle via orthogonal experimental design (OED). As for the construction of the new exemplar in CLPSO, with respect to each dimension of the exemplar, two personal best positions are first randomly selected and the value of the dimension from the better one fills in the corresponding dimension of the new exemplar. Thus, the velocity update in CLPSO is formulated as

$$v_i^d \leftarrow wv_i^d + cr_i^d\left(\text{pbest}_{f_{i(d)}}^d - x_i^d\right) \tag{4}$$

where $\mathbf{f_i} = [f_{i(1)}, \ldots, f_{i(d)}, \ldots, f_{i(D)}]$ defines a series of **pbest** of different particles that the ith particle should follow, r_i^d is randomly generated within [0, 1], and w and c are the inertia weight and the acceleration coefficient as in (2), respectively. Both OLPSO and CLPSO show great efficiency in low-dimensional space; however, they are not suitable for large-scale problems because, on one hand OLPSO needs a lot of fitness evaluations in OED to seek the potentially useful combination of dimensions, which is impractical for large-scale problems; on the other hand, CLPSO converges considerably slowly owing to the construction strategy that each dimension of the new exemplar corresponds to a randomly selected personal best position. Nevertheless, the way to learn from different exemplars behind these two optimizers inspires us to propose the SL strategy in this paper.

B. PSO Variants for Large-Scale Optimization

When it comes to high-dimensional space (larger than 500-D), the classical PSO [17], [18] and most of its variants [13], [31], [32], [55] will lose their effectiveness due to the drastic and exponential enlargement of the search space when the dimensionality grows. That massive local optima exist in the high-dimensional space is another challenge, leading to easily being trapped into local areas for EAs [53].

As for the first approach to large-scale optimization, some works have been done to combine CC framework [35] with the classical PSO and its representative variants, resulting in CCPSO. CCPSO-S_K [33] randomly divides the whole dimensions into K groups with each group containing D/K dimensions. Then for each dimension group, the classical PSO is applied to optimize the subspace, while dimensions in the other groups are fixed to the corresponding part of the global best solution **gbest**. Through this cooperation among dimension groups, CCPSO-S_K is promising to solve large-scale optimization. In CCPSO-H_K [33], the classical PSO and CCPSO-S_K are used in an alternating manner, with CCPSO-S_K executed for one iteration, followed by the classical PSO in the next generation. While in CCPSO2, Li and Yao [16] designed a decomposer pool consisting of different group sizes, and the algorithm will randomly select a group size from the pool every time **gbest** is not improved. In addition, the offspring of each parent for each dimension group is randomly sampled around the personal best position or the neighbor best position according to the Cauchy [59] or Gaussian distributions [14].

Currently, the research of CCEAs including CCPSO [16], [33] mainly focuses on proposing good decomposers to divide dimensions into groups, which is the key of the CC framework. So far, CC with variable interaction learning [45] and differential grouping (DG) [41] are the most representative ones, which can potentially detect the dependency among variables. While CCEAs are promising for solving high-dimensional problems, they pay huge cost in fitness evaluations owing to optimizing each subcomponent individually, especially when the number of groups is large.

To relieve the above situation, from another perspective, Cheng *et al.* [53], [54] proposed a competitive learning strategy to enhance diversity. First, they proposed a multiswarm evolutionary framework based on a feedback mechanism (FBE) [54], where two particles randomly chosen from two populations compete with each other and then the loser is updated using a convergence strategy, while the winner is updated using a mutation strategy. Further, they proposed a CSO [53], where two particles randomly selected in a single population compete with each other and then

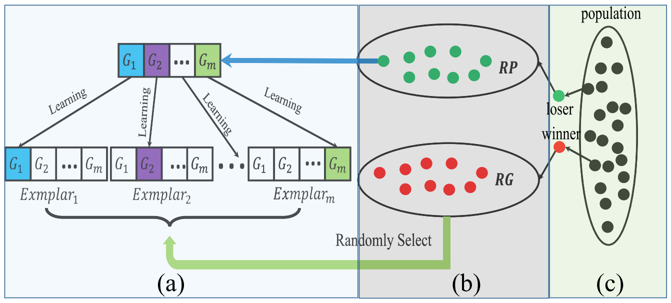

Fig. 1. Visual framework of SPLSO containing three parts. (a) SL. (b) PL. (c) Particle competition.

only the loser is updated, while the winner directly enters the next generation. The update process of the loser is formulated as

$$v_l^d \leftarrow r_1 v_l^d + r_2\left(x_w^d - x_l^d\right) + \phi r_3\left(\bar{x}^d - x_l^d\right) \tag{5}$$

$$x_l^d \leftarrow x_l^d + v_l^d \tag{6}$$

where $\mathbf{V_l} = [v_l^1, \ldots, v_l^d, \ldots, v_l^D]$ is the velocity of the loser, $\mathbf{X_l} = [x_l^1, \ldots, x_l^d, \ldots, x_l^D]$ and $\mathbf{X_w} = [x_w^1, \ldots, x_w^d, \ldots, x_w^D]$ are the positions of the loser and the winner, respectively, $\bar{\mathbf{x}} = [\bar{x}^1, \ldots, \bar{x}^d, \ldots, \bar{x}^D]$ is the mean position of the current population, r_1–r_3 are three uniformly random variables ranging within [0, 1], and ϕ is a parameter controlling the influence of $\bar{\mathbf{x}}$. In this formula, the second part of (5) enforces the loser move toward the winner, possibly resulting in a tendency to approach to the global optima, while the third part may potentially make the particle dragged away from the local area. Such randomized pairwise competition can greatly enhance the diversity of the population, which is very beneficial for large-scale optimization.

However, we find that during one competition, the loser learns deterministically only from the winning counterpart. As a consequence, the learning ability of the loser is restricted. Such an observation, along with considerations elaborated in Section II-A, motivates us to propose the SPL swarm optimizer (SPLSO).

III. Segment-Based Predominant Learning Swarm Optimizer

SPLSO intends to gather the potentially useful evolutionary information, which is concealed in different exemplars, to guide the learning of each particle, so that both diversity and convergence can be promoted to address large-scale optimization problems. The whole framework of SPLSO is shown in Fig. 1. First, an SL method displayed in Fig. 1(a) is introduced in SPLSO, which draws lessons from two representative PSO variants (OLPSO [55] and CLPSO [32]). Second, to get a fast convergence speed, a PL strategy, as shown in Fig. 1(b) accompanies SL, resulting in an SPL strategy. In addition, to deal with the difficulty in determining the optimal number of segments in SPL, a segment number pool consisting of different numbers of segments is designed. It is noticed that this paper contributes to the second approach to large-scale optimization. The detailed techniques adopted in SPLSO are presented as follows.

A. Segment-Based Learning

From the excellent performance of OLPSO [55] and CLPSO [32] in low-dimensional space, we find that learning from different exemplars through combining dimensions can potentially offer good directions for particles. However, originally both OLPSO and CLPSO are not suitable for large-scale optimization, because, for one thing, they all learn information for each dimension separately; for another, OLPSO costs a lot of fitness evaluations to seek the best combination of dimensions, while CLPSO proceeds to the global optima very slowly. Such observation affords the inspiration for the SL method we are proposing.

As illustrated in Fig. 1(a), SL first divides all dimensions of one particle into m disjoint segments with each segment containing D/m dimensions,[1] i.e., $\mathbf{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_j, \ldots, \mathbf{G}_m\}$, where $\mathbf{G}_j$ is the jth dimension segment containing a set of variables. Then, the position $\mathbf{X}_i = [x_i^1, \ldots, x_i^d, \ldots, x_i^D]$ of the ith particle is accordingly divided into m subvectors $\mathbf{X}_i^{\mathbf{G}_1}, \ldots, \mathbf{X}_i^{\mathbf{G}_j}, \ldots, \mathbf{X}_i^{\mathbf{G}_m}$, where $\mathbf{X}_i^{\mathbf{G}_j}$ is a vector of variables from $\mathbf{X}_i$ that belong to the jth segment. Likewise, the velocity $\mathbf{V}_i = [v_i^1, \ldots, v_i^d, \ldots, v_i^D]$ is divided into m subvectors $\mathbf{V}_i^{\mathbf{G}_1}, \mathbf{V}_i^{\mathbf{G}_2}, \ldots, \mathbf{V}_i^{\mathbf{G}_m}$, as well. Subsequently, during the update of velocity and position in SPLSO, SL can be characterized by the following three rules.

1) For the dimensions in segment $\mathbf{G}_j$ of the ith particle, the particle's corresponding velocity subvector $\mathbf{V}_i^{\mathbf{G}_j}$ and position subvector $\mathbf{X}_i^{\mathbf{G}_j}$ are updated as a whole by learning from the same exemplar.
2) For different dimension segments, different exemplars can be used to update different velocity and position subvectors.
3) During one generation, though the number of segments is fixed, the random segmentation on dimensions is executed for each particle to be updated, which means that the components of each counterpart segment are different for different particles, namely the segment $\mathbf{G}_j$ of the ith particle may be different from that of the kth particle.

Compared with traditional learning strategies in PSO variants, especially in OLPSO [55] and CLPSO [32], SL can potentially increase the probability of preserving the useful information embedded in one exemplar together according to rule 1. Rule 2 allows particles to learn from different exemplars, which can potentially collect useful evolution information existing in different exemplars as indicated in OLPSO [55] and CLPSO [32]. Such learning behavior may promote the diversity to some extent. As for rule 3, it is an auxiliary to rules 1 and 2, because for different exemplars, the segments of potentially useful information may be different. On one hand, it can enhance the probability to preserve the potentially beneficial information in different exemplars together; on the other hand, such behavior may improve the diversity to some extent.

[1]If $D\%m \neq 0$, we just add the rest $D\%m$ into the last segment.

Remark: Using the above three rules, SL also enforces a special kind of coevolution among the dimension segments, which differs from the CC framework significantly. It should be noted that although both SL and CCEAs need to divide the dimensions into disjoint parts, they work in very different ways. CCEAs consider each dimension group as a separated subproblem and evolve each subproblem individually. Thus, interdependent variables should be gathered into the same group while independent variables should be put into different groups according to variable dependency. However, SL mainly aims to let each relatively poor particle learn a segment of potentially useful information from different predominant exemplars, so that the useful evolutionary information in different exemplars can potentially be gathered and learned by particles. Thus, this strategy is to increase the probability of gathering the potentially useful information in different predominant exemplars together, but not to increase the probability of putting interdependent variables into the same group as in CCEAs. In high-dimensional problems, it is likely that in SL, a segment of variables from one predominant exemplar containing beneficial evolutionary information includes interdependent and independent variables simultaneously. Obviously, this is totally different from or even violates the aim of the decomposers in CCEAs. Thus, it is likely that the decomposition methods in CCEAs are not suitable for the proposed SL. In addition, instead of treating each segment as a subproblem and evolving each group individually in CCEAs, SL treats all dimensions as a whole and evolves them simultaneously.

In a word, the difference both in dividing dimensions into segments and in evolving variables between SL and CCEAs makes SPLSO very different from CCEAs. Since currently, no effective indicator exists to measure the usefulness of one dimension of an exemplar, we just adopt random segmentation in this paper to divide the dimensions of each particle into m segments. Through this random recombination, on one hand, the potentially useful information in different dimensions of an exemplar may be simultaneously gathered; on the other hand, the potentially useful information in different exemplars may be gathered together with a probability.

B. Segment-Based Predominant Learning

Following SL, a crucial issue is how to select different exemplars for different dimension segments. To solve this issue, we further put forward the PL strategy to cooperate with SL, leading to SPL.

In general, diversity enhancement is accompanied with slow convergence, which is another concern for traditional EAs [13], [14], [30], [53]. In nature, it is common that a particle follows the experience from those which are better than itself. Enlightened by such an idea, we propose the PL strategy.

First, the whole swarm is organized into pairs randomly and the particles in each pair are compared with each other. When making pairwise comparisons between particles, usually one particle is dominated by the other. Therefore, two separated sets, the relatively good particles **RG** and the relatively poor ones **RP**, are generated with the better one entering **RG** while the worse one belonging to **RP** as shown in Fig. 1(b) and (c). Generally speaking, particles in **RG** usually hold more potentially useful information. Therefore, particles in **RP** should update their positions through utilizing beneficial information from particles in **RG** as much as possible. This is the main idea of PL.

Combining SL and PL together, SPL tries to make one poor particle learn segments of useful information from different good particles, which can be formulated as follows:

$$\mathbf{V}_{\mathrm{RP}_j}^{\mathbf{G}_i} \leftarrow r_1 \mathbf{V}_{\mathrm{RP}_j}^{\mathbf{G}_i} + r_2\left(\mathbf{X}_{\mathrm{RG}_{g(j,i)}}^{\mathbf{G}_i} - \mathbf{X}_{\mathrm{RP}_j}^{\mathbf{G}_i}\right) + \phi r_3\left(\hat{\mathbf{x}}^{\mathbf{G}_i} - \mathbf{X}_{\mathrm{RP}_j}^{\mathbf{G}_i}\right) \quad (7)$$

$$\mathbf{X}_{\mathrm{RP}_j}^{\mathbf{G}_i} \leftarrow \mathbf{X}_{\mathrm{RP}_j}^{\mathbf{G}_i} + \mathbf{V}_{\mathrm{RP}_j}^{\mathbf{G}_i} \quad (8)$$

where RP_j denotes the jth relatively poor particle in **RP**, $\mathbf{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_i, \ldots, \mathbf{G}_m\}$ is the dimension segments with totally m segments, $g(j, i)$ indicates the index of the relatively good particle in **RG** that the jth relatively poor particle will learn from for the segment $\mathbf{G}_i$, $\hat{\mathbf{x}}$ is the weighted mean position of the whole population, r_1–r_3 are three random variables ranging within [0, 1], and ϕ is the controlling parameter in charge of the influence of $\hat{\mathbf{x}}$.

The first part in the right of (7) is similar to the inertia term in PSO (2), which is mainly responsible for the stability of the search process. The second part can also be called the cognitive component like in PSO. Instead of learning from **pbest**, SPLSO learns from all potentially good particles through dimension segments. This part offers the chances of providing particles with better directions, which may contribute to fast convergence. As for the third part, we can also name it as the social component as in PSO. This part takes the responsibility to drag the particles away from local optimum areas. Instead of using mean position $\bar{\mathbf{x}}$ in CSO [53], SPLSO shares the social knowledge through the weighted mean position of the whole swarm $\hat{\mathbf{x}} = [\hat{x}^1, \ldots, \hat{x}^d, \ldots, \hat{x}^D]$, which is defined as

$$\hat{x}^d = \sum_{i=1}^{\mathrm{NP}} \frac{\mathrm{fit}(\mathbf{X}_i)}{\sum_{j=1}^{\mathrm{NP}} \mathrm{fit}(\mathbf{X}_j)} x_i^d \quad (9)$$

where NP is the population size, and $\mathrm{fit}(\cdot)$ is the fitness function, defined as the function to be optimized in this paper.

For simplicity, the fitness values of particles are utilized to determine the weight of each particle. To deal with problems with negative fitness values, we adjust (9) as follows:

$$\hat{x}^d = \sum_{i=1}^{\mathrm{NP}} \frac{\mathrm{fit}(\mathbf{X}_i) + |\mathrm{fit}_{\min}| + \eta}{\sum_{j=1}^{\mathrm{NP}} \left(\mathrm{fit}(\mathbf{X}_j) + |\mathrm{fit}_{\min}| + \eta\right)} x_i^d \quad (10)$$

where $\mathrm{fit}_{\min}$ is the minimum fitness value of the swarm and η is a small positive value used to avoid a zero denominator.

In (9) or (10), $\hat{\mathbf{x}}$ puts more emphasis on inferior particles, which may be beneficial for dragging particles away from local optimum areas. In fact, this weighted mean position $\hat{\mathbf{x}}$ is functioned on the third part in the right of (7), namely the social component, which is mainly for the promotion of search diversity. On one hand, the diversity of a swarm generally relies on inferior particles, which possess more powerful exploration ability than superior ones. On the other hand, in large-scale problems, usually, many local optima exist. Thus, superior particles may easily fall into local areas, leading to premature convergence. However, this situation can be alleviated and the chances of escaping from local traps can be enhanced through putting more weight on inferior particles. In Section IV-B2, the influence of $\hat{\mathbf{x}}$ will be investigated through comparing against $\bar{\mathbf{x}}$ and the experimental result favors $\hat{\mathbf{x}}$.

Note that SPL only operates on particles in **RP**, indicating that only half of the swarm is updated in each generation.

Though (7) has characterized SPL, some detailed techniques have to be mentioned here.

1) During each generation, the relatively good particle set **RG** and the relatively poor particle set **RP** are generated through random pairwise comparison with **RG** associated with the better ones and **RP** related to the worse ones.
2) During the update of poor particles, for each segment $\mathbf{G}_i$ of the jth poor particle $\mathbf{X}_{\mathrm{RP}_j}$, a random good particle $\mathbf{X}_{\mathrm{RG}_{\mathrm{rand}}}$ is selected from **RG**, and then we compare the fitness of this randomized good particle $\mathrm{fit}(\mathbf{X}_{\mathrm{RG}_{\mathrm{rand}}})$ with that of the poor particle's corresponding dominator $\mathrm{fit}(\mathbf{X}_{\mathrm{RG}_{\mathrm{RP}_j}})$. If $\mathrm{fit}(\mathbf{X}_{\mathrm{RG}_{\mathrm{rand}}}) < \mathrm{fit}(\mathbf{X}_{\mathrm{RG}_{\mathrm{RP}_j}})$, we will update the poor particle using the randomized good particle $\mathbf{X}_{\mathrm{RG}_{\mathrm{rand}}}$ in (7), otherwise, we will use its corresponding dominator $\mathbf{X}_{\mathrm{RG}_{\mathrm{RP}_j}}$ to update this particle.
3) For each poor particle in **RP**, the random segmentation on dimensions is conducted, resulting in m segments $\mathbf{G} = \{\mathbf{G}_1, \ldots, \mathbf{G}_i, \ldots, \mathbf{G}_m\}$. Note that for different poor particles, the partition may be different, but the number of segments is the same.

First, the random pairwise comparison is adopted in technique 1 because such randomness can allow some poor particles to survive, which may contribute to the promotion of diversity. However, this also may result in that some relatively good particles in **RG** may be dominated by some relatively poor ones in **RP**. Thus, not all relatively good particles are valuable for the relatively poor ones. Generally speaking, the better the learned particle is for one poor particle, the faster this poor particle may approach to the global optimum area.

Thus, to accelerate the learning speed of poor particles, in technique 2, we only allow poor particles to learn from those predominant ones that can dominate their dominators that win in the corresponding pairwise comparisons. To reduce the possibility that this learning strategy may lead to falling into local optima, we take advantage of two methods to enhance the diversity.

1) The good particle learned by one poor particle is randomly chosen from **RG** as in technique 2).
2) In technique 3), for each poor particle, the random dimension segmentation is conducted along with SPL, namely rule 3 in SL.

The whole framework of SPL is displayed in Fig. 2. Overall, SPL can potentially bring the following advantages to SPLSO.

1) SPL can allow poor particles to learn useful information comprehensively, and potentially gather the useful evolutionary information concealed in one predominant exemplar together.
2) SPL may accelerate the learning speed for poor particles through PL, leading to fast convergence.
3) With SL allowing poor particles to learn several segments of information from different good ones, SPL may also enhance the diversity of the swarm to some extent.

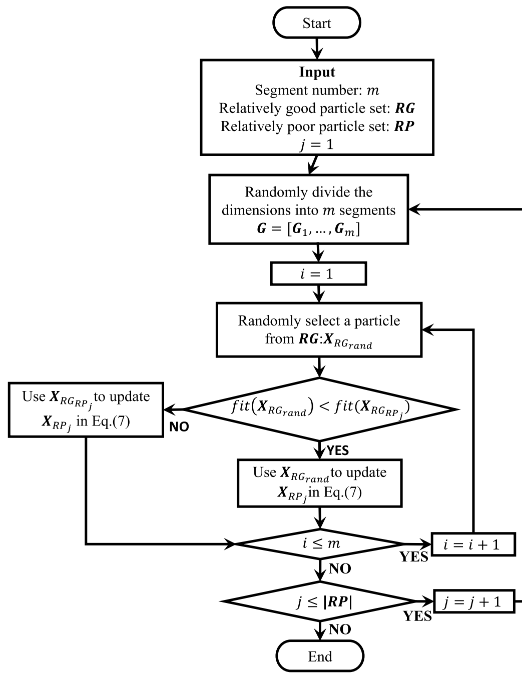
Start
Input
Segment number: m
Relatively good particle set: $\boldsymbol{RG}$
Relatively poor particle set: $\boldsymbol{RP}$
$j = 1$
Randomly divide the dimensions into m segments $\boldsymbol{G} = [\boldsymbol{G}_1, \ldots, \boldsymbol{G}_m]$
$i = 1$
Randomly select a particle from $\boldsymbol{RG}$: $\boldsymbol{X}_{RG_{rand}}$
$fit(\boldsymbol{X}_{RG_{rand}}) < fit(\boldsymbol{X}_{RG_{RP_j}})$
NO
YES
Use $\boldsymbol{X}_{RG_{RP_j}}$ to update $\boldsymbol{X}_{RP_j}$ in Eq.(7)
Use $\boldsymbol{X}_{RG_{rand}}$ to update $\boldsymbol{X}_{RP_j}$ in Eq.(7)
$i \le m$
YES
$i = i + 1$
NO
$j \le |\boldsymbol{RP}|$
YES
$j = j + 1$
NO
End

Fig. 2. Framework of SPL.

Remark: Since this paper is mainly inspired from CSO [53] and CLPSO [32], in this part, we elucidate the main differences between the proposed SPL and the learning strategies in these two PSO variants.

1) Comparison Between SPLSO and CSO: Specifically, compared with the competitive learning strategy in CSO [53] displayed in (5) and (6), the proposed SPL formulated as (7) and (8) differs from it mainly in two aspects.

1) Distinguishing from CSO, where only one particle is selected as the exemplar to guide the learning of a particle, the developed SL first randomly partitions each particle into several segments and then allows different particles to guide the learning of different segments of a particle. In other words, several particles can simultaneously guide the learning of one particle. Through SL, on one hand, the potentially useful information embedded in different exemplars may be gathered together; on the other hand, the potentially useful information embedded in different dimensions of an exemplar may be preserved and gathered together. This random recombination may provide promising directions to guide particles to find promising areas fast and provide chances for particles to escape from local areas.
2) Differing from CSO, where each loser can only learn from its corresponding winner, the proposed PL utilizes pairwise competition to partition the whole swarm

into two separate sets: a) the relatively good particle set **RG** and b) the relatively poor particle set **RP**. Then, each particle in **RP** is updated by SL where the exemplars guiding the learning of this particle are randomly selected from **RG**. The cooperation between SL and PL leads to SPL, which may afford potential to grasp the useful information embedded in different predominant particles in **RG**.

2) Comparison Between SPLSO and CLPSO: Comparing SPL (7) with the comprehensive learning strategy (4) in CLPSO [32], we can observe three main differences.

1) Instead of using particles' personal best positions (**pbest**) to guide the learning of particles in CLPSO, in SPL, each particle is guided by the predominant particles in the current swarm. Since particles are generally updated in each generation and the predominant ones are usually not the same during two consecutive generations, the diversity of the swarm can be enhanced through letting particles learn from predominant ones in the swarm.
2) Instead of constructing the exemplar of one particle *dimension by dimension* through selecting from different **pbest**s, SPL constructs the exemplar of one particle in **RP** *segment by segment* through selecting from different predominant particles. Through this, the potentially useful information embedded in several dimensions of a predominant particle can be preserved together to guide the learning of particles.
3) Different from CLPSO which updates the whole swarm in each generation, SPL utilizes the pairwise competition to partition the swarm into two sets: **RP** and **RG** and only the particles in **RP** are updated through learning from particles in **RG** in each generation. In other words, only half of the swarm are updated in each generation. Through this, the potentially promising particles can be preserved and protected from being weakened.

In particular, in Section IV-B1, the benefit of the proposed SPL is verified by the experiments in comparison with CSO and CLPSO.

C. Dynamically Determining the Number of Segments

Observing SPLSO, we can see that only two parameters are introduced, namely the number of segments m and the control parameter ϕ. With respect to ϕ, we leave it to be fine-tuned in the experiments like in CSO [53]. For the segment number m in the dimension segmenting technique, we can see that a small m leads to a large number of elements in each segment but a small number of different exemplars that a particle learns from because the learning of each segment of one particle is guided by one exemplar. Such a number may be beneficial for gathering the potentially useful information in different dimensions of an exemplar together, but not beneficial for gathering potentially information in different exemplars due to the small number of selected exemplars. On the contrary, a large m results in a small number of elements in each segment but a large number of different exemplars that one particle learns from. This may be beneficial for gathering the potentially information in different exemplars, but not beneficial for gathering the potentially information in different dimensions of one exemplar. Thus, a proper m is needed. However, such a number is usually hard to set, because the prior knowledge about how many useful dimensions exist in one selected exemplar is not known. In addition, for different exemplars, the proper m may be different. Let alone that the proper m for different problems is different.

Borrowing ideas from [16] and [43], we design a segment number pool denoted as $\mathbf{S} = \{s_1, \ldots, s_t\}$ containing t different segment numbers, to aid SPLSO alleviate the above concerns. Then, at each generation, SPLSO will probabilistically select a number from **S**. When updating particles in **RP**, m is fixed to be the selected number. At the end of the generation, the relative performance improvement of the proposed optimizer using this segment number is recorded to update the probability. With this mechanism, SPLSO may self-adaptively select a proper segment number despite of different features of different problems and different evolution stages for a single problem.

Therefore, in order to compute the probability of different segment numbers in **S**, we define a relative performance improvement list $\mathbf{R_s} = \{r_1, \ldots, r_t\}$, where $r_i \in \mathbf{R_s}$ is associated with $s_i \in \mathbf{S}$. At the initialization stage, each $r_i \in \mathbf{R_s}$ is set to 1, and then, r_i is updated as in [43]

$$r_i = \frac{\left|F - \tilde{F}\right|}{|F|} \tag{11}$$

where F is the global best fitness of the last generation, while $\tilde{F}$ is the global best fitness of the current generation.

Then the probability of $s_i \in \mathbf{S}$ is computed as in [43]

$$p_i = \frac{e^{7r_i}}{\sum_{j=1}^{t} e^{7r_j}}. \tag{12}$$

On the basis of $\mathbf{P_s} = \{p_1, \ldots, p_t\}$, we conduct roulette wheel selection to select a segment number m at each generation. Observing (11) and (12), we can notice that: 1) the value of each r_i is within [0, 1], because (11) calculates the relative performance improvement using the global best fitness values between two consecutive generations and 2) if the global best fitness value differs a lot between two consecutive generations, r_i is close to 1. This indicates that the selected segment number in this generation is very appropriate and thus should have a high probability to be selected in next generation, which can be implied by the probability computed in (12). On the contrary, when the global best fitness value differs little between two consecutive generations, r_i is close to 0. This indicates that the selection of the segment number in this generation is not so advisable and thus the probability of this selection should be small, which can also be implied by the probability computed in (12). Therefore, through this, SPLSO can potentially make an appropriate choice of m for different problems or for a single problem at different stages.

As for our algorithm, when the number of segments is fixed and keeps unchanged, we call it as SPLSO; when it uses a

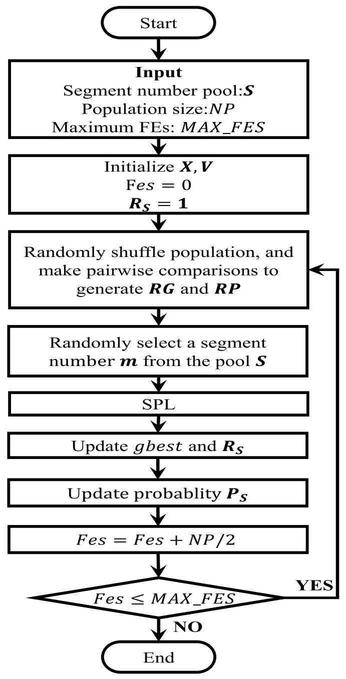

Fig. 3. Framework of DSPLSO.

dynamic segment number, we call as DSPLSO. The performance of both versions are compared in Section IV-B3 and the comparison results favor DSPLSO.

D. Overall Optimizer and Complexity Analysis

The overall procedure of the proposed DSPLSO is exhibited in Fig. 3. The execution process of DSPLSO during each generation can be described as follows.

Step 1: First, the whole swarm is randomly organized into pairs and pairwise comparison is executed in each pair to generate the relatively poor particle set **RP** and the relatively good particle set **RG**.

Step 2: Then, the roulette wheel selection is conducted to select a segment number from the pool **S**.

Step 3: Subsequently, SPL is executed for each relatively poor particle in **RP** as shown in Fig. 2.

Step 4: After each generation, **gbest** is updated and the probabilities $\mathbf{P_s}$ of all segment numbers in the pool are recalculated according to (12).

Step 5: If the termination condition is met, the whole procedure exits; otherwise, goes to step 1 to continue.

1) Complexity Analysis: As for the computational complexity, during one generation, from the above steps, we can see that $O(\text{NP})$ is needed to shuffle the swarm and generate **RP** and **RG** in step 1, where NP is the swarm size. Step 2 takes constant time and step 3 needs $O(\text{NP} * D)$ to update **RP** with $O(\text{NP} * D/2)$ to segment the dimensions of NP/2 particles to be updated and another $O(\text{NP} * D/2)$ to update these particles. To update the probability set $\mathbf{P_s}$ in step 4, $O(t)$ is needed with t being the size of segment number set **S**, which is a constant and much smaller than NP. To sum up, the complexity of DSPLSO is $O(\text{NP} * D + \text{NP} + t)$. Compared with the traditional PSO with complexity $O(\text{NP} * D)$, DSPLSO only needs $O(\text{NP} + t)$ extra in each generation, where the population size NP and the pool size t are constants far smaller than $\text{NP} * D$. Therefore, DSPLSO remains efficient in time complexity.

IV. Experiments

To verify the efficiency and effectiveness of DSPLSO, a series of experiments are conducted on CEC'2010 [60] and CEC'2013 [61] large-scale benchmark problems. The CEC'2013 benchmark set (containing 15 functions) is the extension of the CEC'2010 set (consisting of 20 functions). The functions from the CEC'2013 set are more complicated and harder to optimize because of introducing a number of new features, such as imbalance between subcomponents and overlapping functions. The main characteristics of these two function sets are summarized in Tables SI and SII in the supplemental material, respectively. For details of these functions, readers can refer to [60] and [61].

In this section, we first empirically investigate the influence of two key parameters in DSPLSO, namely, the population size NP and the control parameter ϕ in (7). Then, we will observe the importance of the weighted mean position $\hat{\mathbf{x}}$ and the dynamic segment number selection strategy in Sections IV-B2 and IV-B3, respectively. Subsequently, we will make a wide comparison between DSPLSO and other state-of-the-art algorithms dealing with large-scale optimization in Section IV-C. At last, in Section IV-D, we will observe the scalability of DSPLSO on higher dimensional problems in comparison with CSO [53].

In addition, unless otherwise stated, the maximum number of fitness evaluations is set to 3×10^6 so that we can make comparisons between DSPLSO and other algorithms which are benchmarked against the same test suite, by citing the results reported in the corresponding papers. Mean value and standard deviation (Std) of results from 30 independent runs are used to evaluate the performance of different algorithms. Besides, in the comparisons between two different algorithms, two-tailed t-tests are performed at a significance level of $\alpha = 0.05$, at which the critical t value with 30 samples is 2.064. Additionally, it is worth mentioning that all algorithms are conducted on a PC with 4 Intel Core i5-3470 3.20 GHz CPU, 4 GB memory and Ubuntu 12.04 LTS 64-bit operating system. To save space, some detailed results are not included here but attached to the supplemental material.

A. Parameter Settings

Except for the swarm size NP, which all EAs are sensitive to, there are only two key parameters in DSPLSO: 1) the segment number pool **S** and 2) the control parameter ϕ in (7). **S** is designed mainly because for different problems, the optimal number of segments may be different and even for a single problem, this number may be different at different evolution stages. So, to well adapt to different problems, the number in **S** should have a large range. In this paper, enlightened by the settings of the decomposer pool in CCPSO2 [16] and MLCC [43], we set $\mathbf{S} = \{1, 10, 20, 50, 100, 250\}$, with a wide range of segment numbers. Users or readers can also use other sets, but there seems no significant difference according to our preliminary experiments if we keep **S** in a wide range.

Thus, there remains only one parameter introduced newly in DSPLSO that needs to be fine-tuned, namely ϕ in (7). A large ϕ can improve the influence of weighted mean position $\hat{\mathbf{x}}$, possibly leading to a good ability to escape from local optima, but this may also lead to slow convergence. Conversely, a small ϕ may contribute to fast convergence, but it may result in premature convergence easily. So, ϕ should be tuned. As for the common parameter NP for all EAs, a small swarm size may maintain fast convergence, but cannot afford enough diversity, resulting in premature convergence. In contrast, although a large swarm size can increase the diversity of the swarm, it may slow down the convergence speed. Thus, a tradeoff between the diversity and the convergence speed should be obtained through selecting a proper NP.

Consequently, to obtain empirical insight into NP and ϕ, experiments are conducted on DSPLSO with NP varying from 100 to 600 and ϕ varying from 0 to 0.4 on six CEC'2010 benchmark functions: fully separable and unimodal f_1, fully separable and multimodal f_3, partially separable and unimodal f_7, and partially separable and multimodal f_6, f_{11}, and f_{16}. Among these functions, more multimodal functions are selected because they are more difficult to optimize than unimodal functions, resulting in that EAs are more sensitive to parameters on these functions.

Table SIII in the supplemental material exhibits the results with the best ones highlighted for different population sizes along with the corresponding best ϕ. From this table, we can draw three conclusions.

1) With ϕ fixed to be a nonzero value, a large NP is preferred. This is because a large NP can afford enough diversity to drag the swarm out of the local areas. However, comparing the results in column "$\phi = 0.1$," we can see that when NP exceeds 500, the performance of DSPLSO degrades. This may be caused by the slow convergence resulted from the excessive diversity.
2) With NP fixed and larger than 200, $\phi \neq 0$ is preferred, indicating the importance of the social learning part in (7). However, when ϕ is larger than 0.1, the performance of DSPLSO deteriorates dramatically, especially when $\phi > 0.2$. This is because the social learning part is over emphasized due to the larger ϕ, which may result in that particles are dragged far away from the current promising area and the convergence is slowed down. Thus, a large ϕ is not preferred.
3) When NP takes a fixed value within [300, 500], the proper ϕ remains unchanged at 0.1.

Based on the above observation, in the following experiments related to 1000-D problems, NP = 500 along with $\phi = 0.1$ is adopted for fair comparison with CSO [53], which shares some similarities with DSPLSO, like evolving all dimensions together, and for which the optimal population size is 500 as well.

B. Observations of DSPLSO

1) Usefulness of the Proposed SPL: In this part, we aim to verify the usefulness of the developed SPL. Specifically, we first develop two special cases of the developed SPLSO: 1) SPLSO-1 and 2) SPLSO-1000. The former is the SPLSO with only one segment, which indicates that the whole dimensions are considered as a segment and only one predominant particle in **RG** is selected to guide the learning of particles in **RP** in (7). The latter is the SPLSO with 1000 segments, which indicates that each dimension is a separate segment for 1000-D problems and SPL would select one predominant particle in **RG** for the update of each dimension of particles in **RP** as shown in (7). Then, we can verify the usefulness of SPL through two comparisons.

1) The comparison between SPLSO-1 and CSO, which can verify the usefulness of SPL under the condition that the whole dimensions of one particle are guided by only one exemplar.
2) The comparison between SPLSO-1000 and CLPSO, which can verify the usefulness of SPL under the condition that each dimension of a particle is updated by one exemplar.

Then, we conduct experiments on the twenty 1000-D CEC'2010 functions. To make fair comparisons, the swarm size is set the same (500) for all compared algorithms. Fig. S1 in the supplemental material shows the comparison results.

From this figure, we can see the following.

1) Compared with CSO, SPLSO-1 performs much better on almost all functions, except for five functions, namely f_5, where SPLSO-1 is a little inferior to CSO, and f_{13}, f_{15}, f_{18}, and f_{20}, where SPLSO-1 and CSO perform very similarly.
2) Compared with CLPSO, SPLSO-1000 are much superior on almost all function as well, except for f_{10} where SPLSO-1000 is worse than CLPSO.
3) Further, we also can observe that both SPLSO-1 and SPLSO-1000 are much better than CSO and CLPSO on more than 13 functions.

The above experimental results demonstrate that under two extreme conditions, namely considering the whole dimensions as one segment and considering each dimension as a segment, SPLSO is much better than both CSO and CLPSO, verifying the great usefulness of SPL. Compared with the competitive learning strategy in CSO and the comprehensive learning in CLPSO, SPL mainly benefits from the developed PL, where the whole swarm is partitioned into two separate sets **RP** and **RG** and each particle in **RG** can potentially be the exemplar to guide the learning of the particles in **RP**. These help the swarm in SPLSO maintain high diversity, so that local traps can be avoided, leading to the promising performance of SPLSO.

2) Influence of the Weighted Mean Position: To substantiate the usefulness of weighted mean position $\hat{\mathbf{x}}$ of the swarm as proposed in (7), first, we use the mean position $\bar{\mathbf{x}}$ to substitute $\hat{\mathbf{x}}$ in (7), leading to a version of DSPLSO, denoted as DSPLSO-$\bar{\mathbf{x}}$. To have a better view on the comparison, the original DSPLSO using $\hat{\mathbf{x}}$ is represented as DSPLSO-$\hat{\mathbf{x}}$. Subsequently, these two versions of DSPLSO are compared on six functions used in Section IV-A. Fig. S2 in the supplemental material shows the comparison results with the number of fitness evaluations varying from 5×10^5 to 5×10^6.

From this figure, we can see that at the early stage, $\hat{\mathbf{x}}$ and $\bar{\mathbf{x}}$ have the same effect; but in the late stage, especially when the best solution of the swarm is close to the global or local optima, the solutions obtained by DSPLSO-$\hat{\mathbf{x}}$ are much better than those obtained by DSPLSO-$\bar{\mathbf{x}}$, especially on multimodal functions as shown in Fig. S2(b), (c), (e), and (f) in the supplemental material. This is mainly because instead of treating all particles equally in $\bar{\mathbf{x}}$, more emphasis is put on inferior particles in $\hat{\mathbf{x}}$, which potentially enhances the force to drag the swarm to escape from the local areas. Since this weighted mean position operated on the third part in (7) is mainly for promoting the diversity of the swarm, DSPLSO-$\hat{\mathbf{x}}$ and DSPLSO-$\bar{\mathbf{x}}$ may perform similarly on unimodal functions, such as f_7 [shown in Fig. S2(d) in the supplemental material], because of the consistence that all particles in the swarm converge to the same direction in the unimodal functions. However, DSPLSO-$\hat{\mathbf{x}}$ performs better on multimodal functions.

Overall, we can see that the weighted mean position $\hat{\mathbf{x}}$ is promising for DSPLSO in promoting the exploration ability of the swarm, leading to superior performance to $\bar{\mathbf{x}}$.

3) Influence of the Dynamic Segment Number: To investigate the effectiveness of the dynamic segment number selection strategy in DSPLSO, first, we denote SPLSO with fixed segment number m as "SPLSO-m," e.g., SPLSO with only one segment can be expressed as "SPLSO-1," which indicates that the whole dimensions are considered a segment and only one predominant exemplar in **RG** is selected to guide the learning of each particle in **RP**. Then we make comparisons between "SPLSO-m" with $m \in \mathbf{S}$ and DSPLSO on f_1, f_2, f_5, f_8, f_{13}, and f_{18} from the CEC'2010 benchmark set. Fig. S3 in the supplemental material displays the comparison results.

From this figure, it can be found that: 1) on unimodal function f_1, SPLSO-1 performs slightly better with respect to the convergence speed, but performs very similarly to other versions of SPLSO, such as DSPLSO and SPLSO with other fixed numbers of segments, in regard to the solution quality. However, when it comes to multimodal functions, like f_2, f_5, f_8, f_{13}, and f_{18}, SPLSO-1 performs worse than other versions of SPLSO. This is because, for complicated multimodal problems, cooperated with PL, the developed SL can introduce higher diversity owing to letting particles in **RP** learn from several predominant particles in **RG** and 2) the optimal number of segments is different for different functions, such as for f_5, the number is 100, while for f_{13}, it is 20. Employing this strategy, DSPLSO can make a good compromise for different functions, which is indicated by that DSPLSO performs very close to or the same as the SPLSO with the optimal segment number. Additionally, the dynamic segment number strategy relieves DSPLSO from the sensitivity to m, liberating users from the tedious effort of fine-tuning m for different problems.

Thus, it is beneficial for SPLSO to adopt the dynamic segment number strategy.

C. Comparisons With State-of-the-Art Methods

To better demonstrate the efficiency of DSPLSO, we chose five state-of-the-art and representative algorithms for comparison. Two of them are PSO variants proposed recently: CSO [53] focusing on the second approach to large-scale optimization as DSPLSO and CCPSO2 [16] concentrating on the first approach, and the other three are DE variants based on the CC framework [35]: 1) DECC-DG [41]; 2) DECC-G [42]; and 3) MLCC [43], which contribute to the first approach to large-scale optimization with different decomposers.

TABLE I
PARAMETER SETTINGS OF THE COMPARED ALGORITHMS FOR 1000-D PROBLEMS

Algorithms	Parameters
CSO	$NP = 500, \phi = 0.1$
CCPSO2	$NP = 30$, Group Size Set $\tilde{\mathbf{S}} = \{2, 5, 10, 50, 100, 250\}$
DECC-DG	$NP = 50$, $\varepsilon = 1.0E-3$
DECC-G	$NP = 100$, group size $s = 100$
MLCC	$NP = 50$, Group Size Set $\tilde{\mathbf{S}} = \{5, 10, 25, 50, 100\}$

Furthermore, the parameters in each algorithm are set as recommended in the corresponding papers for fair comparisons, which is shown in Table I. For MLCC and DECC-G, we directly use the reported results in the Special Session on Large-Scale Global Optimization at CEC'2010[2] for the CEC'2010 benchmark set, while for the CEC'2013 benchmark set, owing to the missing of reported results of MLCC in the CEC'2013 Special Session, we have to only report the results of DECC-G.[3] For CSO, CCPSO2, and DECC-DG, because they were developed in recent two years and have shown their superiority to other methods, we put more emphasis on the comparisons between these three methods and DSPLSO.

Tables II and III present the comparison results among different algorithms on problems with 1000-D in the CEC'2010 set and the CEC'2013 set, respectively. The highlighted t values mean that DSPLSO is significantly better than the corresponding compared algorithms judged by t values. In addition, in the last two columns of Table II and in the last column of Table III, we use the highlighted **N/A**s to indicate DSPLSO is much better than MLCC or DECC-G judged by the mean values instead of t values, owing to the absence of the detailed results of MLCC and DECC-G in the corresponding special sessions. When using mean values as the comparison standard, if the order of magnitude of mean value of DSPLSO is lower than that of MLCC or DECC-G, we think DSPLSO is significantly better; if the magnitude of mean value of DSPLSO is higher, we think DSPLSO is worse; otherwise, we consider DSPLSO is equivalent to MLCC or DECC-G. Furthermore, we use $w/t/l$ in the last row of both tables to give the number of wins, ties and losses when DSPLSO compares against the counterpart methods. When compared with MLCC and DECC-G, $w/t/l$ are counted by comparing the mean values of DSPLSO with those of MLCC or DECC-G according to the above mentioned standard.

From these two tables, undoubtedly, we can conclude that DSPLSO outperforms the recent algorithms: CSO (21/35), CCPSO2 (23/35), and DECC-DG (21/35) on most of the functions. As for MLCC, DSPLSO displays its superiority on eight functions, and only loses its advantage on two functions on the

[2]http://nical.ustc.edu.cn/cec10ss.php

[3]http://goanna.cs.rmit.edu.au/~xiaodong/cec13-lsgo/competition/lsgo2013-decc-g.html

TABLE II

COMPARISON RESULTS BETWEEN DSPLSO AND THE COMPARED ALGORITHMS ON 20 CEC'2010 BENCHMARK FUNCTIONS WITH 1000-D. THE MEAN VALUE AND STANDARD DEVIATION ALONG WITH TWO TAILED t-TEST AT SIGNIFICANCE LEVEL OF $\alpha = 0.05$ WITH RESPECT TO FUNCTION VALUES ARE REPORTED OVER 30 INDEPENDENT RUNS. THE BOLDED t VALUES MEAN THAT DSPLSO IS SIGNIFICANTLY BETTER THAN THE CORRESPONDING ALGORITHM

F	Quality	DSPLSO	CSO	CCPSO2	DECC-DG	MLCC	DECC-G
f_1	Mean	7.73E-20	4.75E-12	1.88E+00	7.80E+03	1.53E-27	2.93E-07
	Std	7.07E-21	7.90E-13	2.26E+00	2.66E+04	7.66E-27	8.62E-08
	t-test	-	**3.29E+01**	**4.54E+00**	1.61E+00	N/A	**N/A**
f_2	Mean	4.45E+02	7.48E+03	5.06E+00	4.43E+03	5.57E-01	1.31E+03
	Std	1.65E+01	2.63E+02	1.10E+00	1.98E+02	2.21E+00	3.26E+01
	t-test	-	**1.46E+02**	-1.46E+02	**1.10E+02**	N/A	**N/A**
f_3	Mean	2.52E-13	2.57E-09	5.61E-03	1.67E+01	9.88E-13	1.39E+00
	Std	1.89E-14	1.85E-10	1.73E-03	3.10E-01	3.70E-12	9.73E-02
	t-test	-	**7.61E+01**	**1.77E+01**	**2.96E+02**	N/A	**N/A**
f_4	Mean	4.30E+11	6.87E+11	2.14E+12	4.95E+12	9.61E+12	1.70E+13
	Std	8.31E+10	1.79E+11	1.54E+12	1.33E+12	3.43E+12	5.37E+12
	t-test	-	**7.13E+00**	**6.05E+00**	**1.86E+01**	**N/A**	**N/A**
f_5	Mean	6.30E+06	2.46E+06	4.56E+08	1.49E+08	3.84E+08	2.63E+08
	Std	1.76E+06	1.35E+06	1.20E+08	2.02E+07	6.93E+07	8.44E+07
	t-test	-	-9.48E+00	**2.05E+01**	**3.85E+01**	**N/A**	**N/A**
f_6	Mean	9.45E-09	8.16E-07	1.79E+07	1.63E+01	1.62E+07	4.96E+06
	Std	1.20E-09	2.60E-08	4.50E+06	3.82E-01	4.97E+06	8.02E+05
	t-test	-	**1.70E+02**	**2.18E+01**	**2.34E+02**	**N/A**	**N/A**
f_7	Mean	4.76E+02	2.13E+04	2.48E+08	1.14E+04	6.89E+05	1.63E+08
	Std	1.31E+02	4.60E+03	3.97E+08	8.52E+03	7.37E+05	1.37E+08
	t-test	-	**2.48E+01**	**3.43E+00**	**7.03E+00**	**N/A**	**N/A**
f_8	Mean	3.11E+07	3.86E+07	1.55E+07	2.30E+07	4.38E+07	6.44E+07
	Std	9.36E+04	8.33E+04	1.85E+07	2.16E+07	3.45E+07	2.89E+07
	t-test	-	**3.31E+02**	-4.62E+00	-2.04E+00	N/A	N/A
f_9	Mean	4.59E+07	6.68E+07	1.00E+08	5.90E+07	1.23E+08	3.21E+08
	Std	3.04E+06	4.47E+06	3.93E+07	9.97E+06	1.33E+07	3.38E+07
	t-test	-	**2.12E+01**	**7.58E+00**	**6.91E+00**	**N/A**	**N/A**
f_{10}	Mean	7.99E+03	9.58E+03	5.11E+03	4.55E+03	3.43E+03	1.06E+04
	Std	1.28E+02	6.68E+01	8.80E+02	1.10E+02	8.72E+02	2.95E+02
	t-test	-	**6.02E+01**	-1.78E+01	-1.12E+02	N/A	**N/A**
f_{11}	Mean	3.04E-12	3.98E-08	1.98E+02	1.04E+01	1.98E+02	2.34E+01
	Std	2.89E-13	3.25E-09	3.94E-01	8.19E-01	6.98E-01	1.78E+00
	t-test	-	**6.72E+01**	**2.76E+03**	**6.99E+01**	**N/A**	**N/A**
f_{12}	Mean	9.52E+04	4.37E+05	4.09E+04	2.56E+03	3.49E+04	8.93E+04
	Std	6.69E+03	6.61E+04	1.27E+04	5.31E+02	4.92E+03	6.87E+03
	t-test	-	**2.82E+01**	-2.07E+01	-7.56E+01	N/A	N/A
f_{13}	Mean	5.48E+02	5.53E+02	1.32E+03	5.64E+03	2.08E+03	5.12E+03
	Std	1.69E+02	1.78E+02	1.81E+02	3.02E+03	7.27E+02	3.95E+03
	t-test	-	1.17E-01	**1.70E+01**	**9.23E+00**	**N/A**	**N/A**
f_{14}	Mean	1.60E+08	2.46E+08	2.58E+08	3.40E+08	3.16E+08	8.08E+08
	Std	8.50E+06	1.31E+07	1.34E+08	2.23E+07	2.77E+07	6.07E+07
	t-test	-	**3.00E+01**	**4.00E+00**	**4.13E+01**	N/A	N/A
f_{15}	Mean	9.91E+03	1.01E+04	1.05E+04	5.86E+03	7.11E+03	1.22E+04
	Std	6.70E+01	5.84E+01	1.36E+03	9.20E+01	1.34E+03	8.97E+02
	t-test	-	**1.04E+01**	**2.19E+00**	-1.95E+02	N/A	**N/A**
f_{16}	Mean	4.68E-12	5.68E-08	3.97E+02	7.57E-13	3.76E+02	7.66E+01
	Std	4.49E-13	6.32E-09	6.18E-01	6.67E-14	4.71E+01	8.14E+00
	t-test	-	**4.92E+01**	**3.51E+03**	-4.74E+01	**N/A**	**N/A**
f_{17}	Mean	6.84E+05	2.21E+06	1.32E+05	4.05E+04	1.59E+05	2.87E+05
	Std	3.63E+04	2.10E+05	5.37E+04	2.05E+03	1.43E+04	1.98E+04
	t-test	-	**3.90E+01**	-4.67E+01	-9.70E+01	N/A	N/A
f_{18}	Mean	1.35E+03	1.64E+03	2.91E+03	1.46E+10	7.09E+03	2.46E+04
	Std	3.87E+02	8.27E+02	3.43E+02	2.82E+09	4.77E+03	1.05E+04
	t-test	-	1.75E+00	**1.65E+01**	**2.84E+01**	N/A	**N/A**
f_{19}	Mean	8.20E+06	9.86E+06	1.53E+06	1.74E+06	1.36E+06	1.11E+06
	Std	4.69E+05	5.13E+05	8.83E+04	9.89E+04	7.35E+04	5.15E+04
	t-test	-	**1.31E+01**	-7.66E+01	-7.39E+01	N/A	N/A
f_{20}	Mean	1.06E+03	1.07E+03	2.15E+03	6.28E+10	2.05E+03	4.06E+03
	Std	1.79E+02	1.72E+02	2.08E+02	8.48E+09	1.80E+02	3.66E+02
	t-test	-	2.00E-01	**2.17E+01**	**4.06E+01**	N/A	N/A
$w/t/l$		-	16/3/1	14/0/6	12/2/6	8/10/2	14/6/0

When compared with the first three methods, CSO, CCPSO2, DECC-DG, t values are utilized. However, owing to the absence of all results of 30 runs for MLCC and DECC-G in CEC'2010 competition, we use the mean values to determine the wins or loses.

TABLE III

COMPARISON RESULTS BETWEEN DSPLSO AND THE COMPARED ALGORITHMS ON 15 CEC'2013 BENCHMARK FUNCTIONS WITH 1000-D. THE MEAN VALUE AND STANDARD DEVIATION ALONG WITH TWO TAILED t-TEST AT SIGNIFICANCE LEVEL OF $\alpha = 0.05$ WITH RESPECT TO FUNCTION VALUES ARE REPORTED OVER 30 INDEPENDENT RUNS. THE BOLDED t VALUES MEAN THAT DSPLSO IS SIGNIFICANTLY BETTER THAN THE CORRESPONDING ALGORITHM

F	Quality	DSPLSO	CSO	CCPSO2	DECC-DG	DECC-G
f_1	Mean	1.18E-19	7.88E-12	3.53E+01	4.88E-03	2.03E-13
	Std	1.06E-20	1.21E-12	2.33E+01	1.45E-02	1.78E-14
	t-test	-	**3.58E+01**	**8.32E+00**	1.85E+00	**N/A**
f_2	Mean	1.06E+03	8.58E+03	3.57E+01	1.28E+04	1.03E+03
	Std	4.45E+02	1.79E+02	5.22E+00	5.43E+02	2.26E+01
	t-test	-	**8.58E+01**	-1.26E+01	**9.16E+01**	N/A
f_3	Mean	2.16E+01	2.16E+01	2.00E+01	2.13E+01	2.87E-10
	Std	7.53E-03	5.99E-03	9.00E-05	1.97E-02	1.38E-11
	t-test	-	1.04E+00	-1.16E+03	-7.10E+01	N/A
f_4	Mean	9.40E+09	1.35E+10	3.41E+10	4.16E+10	2.60E+10
	Std	1.89E+09	3.17E+09	1.97E+10	2.39E+10	1.47E+10
	t-test	-	**6.07E+00**	**6.84E+00**	**7.37E+00**	**N/A**
f_5	Mean	6.30E+05	5.97E+05	1.64E+07	4.97E+06	7.28E+14
	Std	1.02E+05	1.04E+05	4.22E+06	4.00E+05	1.51E+05
	t-test	-	-1.25E+00	**2.05E+01**	**5.76E+01**	**N/A**
f_6	Mean	1.06E+06	1.06E+06	1.05E+06	1.06E+06	4.85E+04
	Std	8.05E+02	1.20E+03	7.68E+03	1.52E+03	3.98E+04
	t-test	-	-1.21E+00	-1.02E+01	-1.71E+00	N/A
f_7	Mean	5.50E+06	5.81E+06	3.04E+08	2.50E+08	6.07E+08
	Std	2.26E+06	3.09E+06	4.42E+08	1.74E+08	4.09E+08
	t-test	-	4.49E-01	**3.69E+00**	**7.71E+00**	**N/A**
f_8	Mean	1.55E+14	2.46E+14	1.02E+15	3.21E+15	4.26E+14
	Std	2.96E+13	8.86E+13	5.48E+14	1.97E+15	1.53E+14
	t-test	-	**5.34E+00**	**8.67E+00**	**8.49E+00**	N/A
f_9	Mean	8.07E+07	6.08E+07	3.72E+09	4.41E+08	4.27E+08
	Std	2.24E+07	1.31E+07	1.01E+09	3.39E+07	9.89E+07
	t-test	-	-4.19E+00	**1.98E+01**	**4.85E+01**	**N/A**
f_{10}	Mean	9.39E+07	9.40E+07	9.32E+07	9.44E+07	1.10E+07
	Std	2.26E+05	2.25E+05	4.92E+05	2.44E+05	4.00E+06
	t-test	-	1.73E+00	-6.92E+00	**6.96E+00**	N/A
f_{11}	Mean	9.27E+11	9.29E+11	9.32E+11	1.60E+10	2.46E+11
	Std	9.48E+09	9.80E+09	1.08E+10	2.49E+10	2.03E+11
	t-test	-	7.84E-01	1.88E+00	-1.87E+02	N/A
f_{12}	Mean	1.05E+03	1.08E+03	2.15E+03	7.39E+10	1.04E+03
	Std	5.37E+01	7.51E+01	2.13E+02	9.86E+09	5.76E+01
	t-test	-	1.71E+00	**2.74E+01**	**4.11E+01**	N/A
f_{13}	Mean	1.20E+09	7.48E+08	4.13E+09	1.40E+10	3.42E+10
	Std	4.99E+08	2.89E+08	1.70E+09	4.23E+09	6.41E+09
	t-test	-	-4.32E+00	**9.03E+00**	**1.65E+01**	**N/A**
f_{14}	Mean	8.31E+09	3.67E+09	9.37E+10	7.06E+09	6.08E+11
	Std	6.67E+09	3.38E+09	1.02E+11	7.75E+09	2.06E+11
	t-test	-	-3.40E+00	**4.59E+00**	-6.72E-01	**N/A**
f_{15}	Mean	4.13E+07	7.61E+07	6.95E+06	6.68E+06	6.05E+07
	Std	3.11E+06	6.24E+06	5.80E+06	1.48E+06	6.45E+06
	t-test	-	**2.73E+01**	-2.86E+01	-5.50E+01	N/A
$w/t/l$		-	5/7/3	9/1/5	9/3/3	7/6/2

When compared with the first three methods, CSO, CCPSO2, DECC-DG, t values are utilized. However, owing to the absence of all results of 30 runs for DECC-G in CEC'2013 competition, we use the mean values to determine the wins or loses.

CEC'2010 benchmark set. As for DECC-G, DSPLSO dominates on 21 functions among all 35 functions. Additionally, we notice that for f_1 from both CEC'2010 set and CEC'2013 set, though the t-test value between DSPLSO and DECC-DG is smaller than the critical value 2.064, DSPLSO still performs significantly better than DECC-DG in terms of mean value and standard deviation. Taking a closer observation, we can see that DSPLSO is better than the CC-based algorithms (CCPSO2, DECC-DG, MLCC, and DECC-G). Compared with CSO, though both algorithms concentrate on the second approach to large-scale optimization, DSPLSO exhibits its advantages over CSO. The good performance of DSPLSO in comparison with these methods benefits from the proposed SPL strategy. On one hand, SPL allows poor particles to learn segments of potentially useful evolution information from different predominant exemplars, resulting in promising enhancement in both diversity (resulted from learning from different exemplars) and convergence (benefited from learning from predominant exemplars). On the other hand, the random segmentation embedded in SL and performed on each relatively poor particle along with the random competition affords potential diversity enhancement. All these together provide strength for DSPLSO to compete with other methods.

Then, we further conduct experiments on both CEC'2010 and CEC'2013 benchmark sets to compare the convergence behavior of different methods with the number of fitness evaluations varying from 5×10^5 to 5×10^6. Figs. S4 and S5 in the supplemental material show the results on 20 CEC'2010 functions and 15 CEC'2013 functions, respectively. In addition, it should be noticed that in Fig. S4 in the supplemental material, the results of DECC-DG are absent on f_1–f_3 and f_5 and f_6 when the number of fitness evaluations is fewer than 1.5×10^6 and 1×10^6, respectively. This is because DECC-DG costs more than 1×10^6 on f_1–f_3 and 5×10^5 on f_5 and f_6 fitness evaluations to partition dimensions into groups.

TABLE IV
COMPARISON RESULTS BETWEEN DSPLSO AND CSO ON 20 CEC'2010 BENCHMARK FUNCTIONS WITH 2000-D. THE MEAN VALUE AND STANDARD DEVIATION ALONG WITH TWO TAILED t-TEST AT SIGNIFICANCE LEVEL OF $\alpha = 0.05$ WITH RESPECT TO FUNCTION VALUES ARE REPORTED OVER 30 INDEPENDENT RUNS. THE BOLDED t VALUES MEAN THAT SPLSO IS SIGNIFICANTLY BETTER THAN CSO

Methods	Quality	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}
DSPLSO	Mean	1.39E-19	2.25E+03	2.01E-13	5.03E+11	9.48E+06	1.49E-08	2.16E+03	2.72E+07	1.55E+08	2.21E+03
	Std	4.71E-21	5.22E+01	2.91E-15	7.71E+10	2.03E+06	4.79E-10	7.38E+02	1.91E+05	6.27E+06	5.74E+01
CSO	Mean	6.84E-08	9.58E+03	2.19E-02	6.56E+11	4.55E+06	3.71E-06	1.29E+05	3.94E+07	2.08E+08	1.87E+04
	Std	6.66E-09	5.37E+02	1.17E-03	9.34E+10	1.65E+06	1.05E-07	2.34E+04	8.43E+04	1.12E+07	1.42E+02
t-test		**5.62E+01**	**7.44E+01**	**1.03E+02**	**6.95E+00**	-1.03E+01	**1.92E+02**	**2.96E+01**	**3.20E+02**	**2.26E+01**	**5.91E+02**
Methods	Quality	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}	f_{16}	f_{17}	f_{18}	f_{19}	f_{20}
DSPLSO	Mean	1.65E-12	1.72E+05	2.18E+03	4.27E+08	2.17E+03	2.56E-12	6.06E+05	6.26E+03	7.72E+06	3.50E+03
	Std	3.01E-14	5.68E+03	7.10E+02	1.50E+07	4.95E+01	3.68E-14	1.26E+04	1.00E+03	2.86E+05	3.12E+02
CSO	Mean	6.00E-06	5.72E+05	1.89E+03	6.43E+08	2.03E+04	8.42E-06	3.11E+06	8.51E+03	3.10E+07	2.23E+03
	Std	3.21E-07	2.46E+04	5.29E+02	1.83E+07	7.60E+01	3.87E-07	1.25E+05	4.90E+03	1.27E+06	2.50E+02
t-test		**1.02E+02**	**8.68E+01**	-1.80E+00	**5.03E+01**	**1.09E+03**	**1.19E+02**	**1.09E+02**	**2.46E+00**	**9.79E+01**	-1.74E+01

From Fig. S4 in the supplemental material, we can see that DSPLSO defeats all the three algorithms on seven functions (f_1,f_3,f_4,f_6,f_7,f_{11}, and f_{14}) in terms of the quality of solutions and on f_{13},f_{18}, and f_{20}, though DSPLSO achieves the same best results as CSO, it converges faster. Separately, from the perspective of the quality of solutions, DSPLSO shows its superiority to CCPSO2 [16] on 14 functions, dominates CSO [53] on 15 functions, and beats DECC-DG [41] on 12 functions, respectively.

From Fig. S5 in the supplemental material, we can find that DSPLSO defeats all the three methods on three functions (f_1,f_4, and f_8) and on f_5,f_7, and f_{12}, both DSPLSO and CSO achieve the best results. In detail, DSPLSO defeats CSO on six functions, dominates CCPSO2 on nine functions, and beats DECC-DG on ten functions, respectively. Besides, DSPLSO performs similarly to CSO, CCPSO2 and DECC-DG on 5(f_3,f_5–f_7,f_{12}), 1(f_{11}), and 1(f_6) functions, respectively.

Comprehensively, on one hand, DSPLSO can find competitive or even better solutions than other methods, e.g., shown in Figs. S4(a), (c), and (f) and S5(a) and (d) in the supplementary material; on the other hand, DSPLSO can possess a competitive or even faster convergence speed, e.g., shown in Figs. S4(c), (m), (r), and (t) and S5(a) and (l) in the supplemental material. All these can be potentially attributed to the proposed SPL, which allows particles to learn from different predominant exemplars, enlarging the learning ability of particles. This learning strategy not only brings benefits in enhancing diversity for the swarm, leading to good exploration ability, but also potentially promotes the exploitation ability, probably resulting in fast convergence and good solutions.

Overall, the experimental results in Tables II and III and the convergence plots in Figs. S4 and S5 in the supplemental material, demonstrate the efficiency and effectiveness of DSPLSO in dealing with large-scale optimization.

D. Scalability to Higher Dimensionality

To further evaluate the scalability of DSPLSO to higher dimensionality, we conduct experiments on DSPLSO for optimizing 2000-D problems by modifying the dimension size in the CEC'2010 function generators to 2000.

First, we take a look at the parameter settings of NP and ϕ on 2000-D problems with the segment number set $\mathbf{S}$ the same as the one used for 1000-D problems. Table SIV in the supplemental material, presents the experimental results of DSPLSO with NP varying from 200 to 1000 and ϕ ranging from 0.1 to 0.35 on the six functions that are also used for observing the influence of NP and ϕ on 1000-D problems in Table SIII in the supplemental material, From this table, similar conclusions can be drawn.

1) With NP fixed, a proper ϕ is needed. When NP is smaller than 600, $\phi = 0.1$ is preferred; when NP is medium, such as within [600, 800], $\phi = 0.15$ is preferred; and when NP is large, such as 1000, $\phi = 0.2$ is preferred. This indicates that when NP becomes large, ϕ should also choose a properly large value as well.
2) With ϕ fixed, a large NP is preferred. Comparing the results of DSPLSO with different NP and the corresponding best ϕ, we can find that a large NP is needed, such as 1000, to afford enough diversity for the optimizer to locate the global optima. Above all, we find that NP $= 1000$ and $\phi = 0.2$ is the most proper setting for DSPLSO on 2000-D problems.

Then, the comparison between DSPLSO and CSO [53] is conducted. Here, only CSO is selected to make a comparison because that not only is it the state-of-the-art, but also it belongs to the same approach to large-scale optimization as DSPLSO, which evolves all dimensions together. In addition, NP $= 1000$ and $\phi = 0.15$ is adopted for CSO as recommended in [53]. In this series of experiments, the maximum number of fitness evaluations is set to 5×10^6.

Table IV exhibits the comparison results with highlighted t values indicating DSPLSO is significantly better than CSO. From this table, we can see that DSPLSO significantly outperforms CSO on almost all functions, except for f_5 and f_{20} on which DSPLSO performs worse than CSO and f_{13} on which DSPLSO and CSO achieve similar performance.

Further, to compare the convergence behaviors of DSPLSO and CSO on 2000-D problems, we conduct experiments on these problems with the number of fitness evaluations varying from 1×10^6 to 1×10^7. Fig. S6 in the supplemental material shows the comparison results.

From Fig. S6 in the supplemental material, we can see that DSPLSO is much better than CSO either in terms of the quality of solutions or from the perspective of the convergence speed on almost all functions, except for f_5, f_{13}, and f_{20} where DSPLSO is a little inferior to CSO. Particularly, we can find that DSPLSO converges considerably faster than CSO with better solutions on f_1, f_3, f_6, f_{10}, f_{11}, f_{15}, and f_{16}.

The above verified superiority of DSPLSO over CSO mainly benefits from two aspects.

1) Compared with CSO, where the loser is limited to only learn from its corresponding winner, the relatively poor particles in **RP** in DSPLSO possess better learning ability, due to learning from different predominant exemplars, which is driven by PL.
2) Instead of updating all dimensions using only one exemplar (the corresponding winner for a loser) in CSO, DSPLSO divides the dimensions of each particle to be updated into several segments, and then evolves the dimensions in each segment together by learning from a randomly selected predominant exemplar, which is driven by SL. For different dimension segments, the exemplars may be different. This learning strategy has the potential to gather the useful information in different exemplars together. The cooperation between SL and PL leads to SPL, which can afford promising exploration and exploitation abilities.

All in all, this series of experiments demonstrate the good scalability of DSPLSO to higher dimensionality.

V. Conclusion

This paper has proposed a novel SPLSO. To self-adaptively determine the appropriate number of segments for different problems, borrowing ideas from MLCC [43] and CCPSO2 [16], we designed a segment number pool to dynamically select a proper segment number, leading to DSPLSO. This new optimizer allows the relatively poor particles to learn from different good particles through the SPL. The SPL strategy may contribute to fast convergence and high diversity to some extent, which are verified by the experiments on two widely used large-scale problem sets—the CEC'2010 and CEC'2013 benchmark function sets. The comparison results between DSPLSO and different state-of-the-art algorithms demonstrate the competitive feasibility and efficiency of the new optimizer. Additionally, the experimental results on 2000-D problems further substantiate the competitive scalability of DSPLSO to higher dimensionality.

Though DSPLSO has shown its ability in dealing with large-scale optimization, at times it still falls into local optima. For instance, the results on f_5, f_8, f_9, and f_{14} shown in Table II and on f_4, f_8, f_{11}, and f_{13} shown in Table III are far away from the global optima. This is unfortunately a common drawback for other optimization algorithms as well. Our future research is to investigate how to mitigate the trapping at local optima and further enhance the performance of DSPLSO.

References

[1] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.

[2] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. New York, NY, USA: Springer, 2006.

[3] M. Clerc, *Particle Swarm Optimization*, vol. 93. Hoboken, NJ, USA: Wiley, 2010.

[4] R. A. Sarker, M. Mohammadian, and X. Yao, *Evolutionary Optimization*, vol. 48. New York, NY, USA: Springer, 2002.

[5] I. Fister, Jr., M. Perc, S. M. Kamal, and I. Fister, "A review of chaos-based firefly algorithms: Perspectives and research challenges," *Appl. Math. Comput.*, vol. 252, pp. 155–165, Feb. 2015.

[6] Y. Zhang, Y.-J. Gong, H. Zhang, T.-L. Gu, and J. Zhang, "Towards fast niching evolutionary algorithms: A locality sensitive hashing-based approach," *IEEE Trans. Evol. Comput.*, in press, 2016.

[7] P. Faria, J. Soares, Z. Vale, H. Morais, and T. Sousa, "Modified particle swarm optimization applied to integrated demand response and DG resources scheduling," *IEEE Trans. Smart Grid*, vol. 4, no. 1, pp. 606–616, Mar. 2013.

[8] M. Shen *et al.*, "Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks," *IEEE Trans. Ind. Electron.*, vol. 61, no. 12, pp. 7141–7151, Dec. 2014.

[9] J. Zhang, C. Zhang, T. Chu, and M. Perc, "Resolution of the stochastic strategy spatial prisoner's dilemma by means of particle swarm optimization," *PloS One*, vol. 6, no. 7, 2011, Art. no. e21787.

[10] I. Fister *et al.*, "Particle swarm optimization for automatic creation of complex graphic characters," *Chaos Solitons Fractals*, vol. 73, pp. 29–35, Apr. 2015.

[11] X. Wen *et al.*, "A maximal clique based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Trans. Evol. Comput.*, in press, 2016.

[12] X.-Y. Zhang *et al.*, "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] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 39, no. 6, pp. 1362–1381, Dec. 2009.

[14] M. Campos, R. A. Krohling, and I. Enriquez, "Bare bones particle swarm optimization with scale matrix adaptation," *IEEE Trans. Cybern.*, vol. 44, no. 9, pp. 1567–1578, Sep. 2014.

[15] J. Kennedy, "Bare bones particle swarms," in *Proc. IEEE Swarm Intell. Symp.*, Indianapolis, IN, USA, 2003, pp. 80–87.

[16] 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.

[17] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, vol. 1. Nagoya, Japan, 1995, pp. 39–43.

[18] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. 4. Piscataway, NJ, USA, 1995, pp. 1942–1948.

[19] Y.-J. Gong *et al.*, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[20] W.-N. Chen *et al.*, "A novel set-based particle swarm optimization method for discrete optimization problems," *IEEE Trans. Evol. Comput.*, vol. 14, no. 2, pp. 278–300, Apr. 2010.

[21] W.-N. Chen *et al.*, "Particle swarm optimization with an aging leader and challengers," *IEEE Trans. Evol. Comput.*, vol. 17, no. 2, pp. 241–258, Apr. 2013.

[22] Q. Fan and X. Yan, "Self-adaptive differential evolution algorithm with zoning evolution of control parameters and adaptive mutation strategies," *IEEE Trans. Cybern.*, vol. 46, no. 1, pp. 219–232, Jan. 2016.

[23] X. Qiu, K. C. Tan, and J.-X. Xu, "Multiple exponential recombination for differential evolution," *IEEE Trans. Cybern.*, in press, 2016.

[24] Q. Yang *et al.*, "Adaptive multimodal continuous ant colony optimization," *IEEE Trans. Evol. Comput.*, in press, 2016.

[25] T. Liao, K. Socha, M. A. M. de Oca, T. Stützle, and M. Dorigo, "Ant colony optimization for mixed-variable optimization problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 4, pp. 503–518, Aug. 2014.

[26] P. Yang, K. Tang, and X. Lu, "Improving estimation of distribution algorithm on multimodal problems by detecting promising areas," *IEEE Trans. Cybern.*, vol. 45, no. 8, pp. 1438–1449, Aug. 2015.

[27] Q. Yang *et al.*, "Multimodal estimation of distribution algorithms," *IEEE Trans. Cybern.*, in press, 2016.

[28] I. Fister, I. Fister, Jr., X.-S. Yang, and J. Brest, "A comprehensive review of firefly algorithms," *Swarm Evol. Comput.*, vol. 13, pp. 34–46, Dec. 2013.

[29] J. Kennedy, "Stereotyping: Improving particle swarm performance with cluster analysis," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. La Jolla, CA, USA, 2000, pp. 1507–1512.

[30] Z. Ren, A. Zhang, C. Wen, and Z. Feng, "A scatter learning particle swarm optimization algorithm for multimodal problems," *IEEE Trans. Cybern.*, vol. 44, no. 7, pp. 1127–1140, Jul. 2014.

[31] M. Hu, T. Wu, and J. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Trans. Evol. Comput.*, vol. 17, no. 5, pp. 705–720, Oct. 2013.

[32] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Trans. Evol. Comput.*, vol. 10, no. 3, pp. 281–295, Jun. 2006.

[33] 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.

[34] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Portland, OR, USA, 2004, pp. 1980–1987.

[35] M. A. Potter, "The design and analysis of a computational model of cooperative coevolution," Ph.D. dissertation, Dept. Comput. Sci., George Mason Univ., Fairfax, VA, USA, 1997.

[36] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *Parallel Problem Solving From Nature—III*. Heidelberg, Germany: Springer, 1994, pp. 249–257.

[37] Y. Yu and Y. Xinjie, "Cooperative coevolutionary genetic algorithm for digital IIR filter design," *IEEE Trans. Ind. Electron.*, vol. 54, no. 3, pp. 1311–1318, Jun. 2007.

[38] Z. Cai and Z. Peng, "Cooperative coevolutionary adaptive genetic algorithm in path planning of cooperative multi-mobile robot systems," *J. Intell. Robot. Syst.*, vol. 33, no. 1, pp. 61–71, 2002.

[39] Y.-J. Shi, H.-F. Teng, and Z.-Q. Li, "Cooperative co-evolutionary differential evolution for function optimization," in *Advances in Natural Computation*. Heidelberg, Germany: Springer, 2005, pp. 1080–1088.

[40] Z. Yang, K. Tang, and X. Yao, "Differential evolution for high-dimensional function optimization," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, 2007, pp. 3523–3530.

[41] M. N. 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.

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

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

[44] K. Weicker and N. Weicker, "On the improvement of coevolutionary optimizers by learning variable interdependencies," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Washington, DC, USA, 1999, pp. 1627–1632.

[45] W. Chen, T. Weise, Z. Yang, and K. Tang, "Large-scale global optimization using cooperative coevolution with variable interaction learning," in *Parallel Problem Solving From Nature, PPSN XI*. Heidelberg, Germany: Springer, 2010, pp. 300–309.

[46] J. Smith and T. C. Fogarty, "An adaptive poly-parental recombination strategy," in *Evolutionary Computing*. Heidelberg, Germany: Springer, 1995, pp. 48–61.

[47] G. R. Harik, "Learning gene linkage to efficiently solve problems of bounded difficulty using genetic algorithms," Ph.D. dissertation, Dept. Elect. Eng., and Computer Science, Univ. Michigan, Ann Arbor, MI, USA, 1997.

[48] M. Pelikan and D. E. Goldberg, "Hierarchical Bayesian optimization algorithm," in *Scalable Optimization via Probabilistic Modeling*. Heidelberg, Germany: Springer, 2006, pp. 63–90.

[49] N. Hansen and A. Ostermeier, "Completely derandomized self-adaptation in evolution strategies," *Evol. Comput.*, vol. 9, no. 2, pp. 159–195, 2001.

[50] R. Ros and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," in *Parallel Problem Solving From Nature—PPSN X*. vol. 5199. Heidelberg, Germany: Springer, 2008, pp. 296–305.

[51] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Edinburgh, U.K., 2005, pp. 1769–1776.

[52] N. Hansen, A. Auger, R. Ros, S. Finck, and P. Pošík, "Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009," in *Proc. Conf. Genetic Evol. Comput.*, Portland, OR, USA, 2010, pp. 1689–1696.

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

[54] R. Cheng, C. Sun, and Y. Jin, "A multi-swarm evolutionary framework based on a feedback mechanism," in *Proc. IEEE Congr. Evol. Comput.*, Cancun, Mexico, 2013, pp. 718–724.

[55] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 15, no. 6, pp. 832–847, Dec. 2011.

[56] Y. Shi and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput.*, Anchorage, AK, USA, 1998, pp. 69–73.

[57] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Washington, DC, USA, 1999, pp. 1958–1962.

[58] J. Kennedy and R. Mendes, "Population structure and particle swarm performance," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2. Honolulu, HI, USA, 2002, pp. 1671–1676.

[59] R. A. Krohling and E. Mendel, "Bare bones particle swarm optimization with Gaussian or cauchy jumps," in *Proc. IEEE Congr. Evol. Comput.*, Trondheim, Norway, 2009, pp. 3285–3291.

[60] 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," Nature Inspired Comput. Appl. Lab., Univ. Sci. Technol. China, Anhui, China, Tech. Rep., 2009.

[61] 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, Tech. Rep., 2013.

Qiang Yang (S'14) received the M.S. degree from Sun Yat-sen University, Guangzhou, China, in 2014, where he is currently pursuing the Ph.D. degree.

He is currently a Research Assistant with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. His current research interests include evolutionary computation algorithms and their applications on real-world problems, large-scale optimization algorithms, multimodal optimization algorithms, distribute evolutionary algorithms, and their applications on real-world problems, like intelligent transportation.

Wei-Neng Chen (S'07–M'12) received the bachelor's and Ph.D. degrees from Sun Yat-sen University, Guangzhou, China, in 2006 and 2012, respectively.

He is currently a Professor with the School of Computer Science and Engineering, South China University of Technology, Guangzhou. He has published 50 papers in international journals and conferences. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research, and software engineering.

Dr. Chen was a recipient of the Pearl River New Star in Science and Technology in 2014, the Natural Science Foundation for Distinguished Young Scientists of Guangdong Province, China, in 2015, the Guangdong Special Support Program for Outstanding Young Scientists in 2015, and the IEEE Computational Intelligence Society Outstanding Dissertation Award in 2016.

Tianlong Gu received the M.Eng. degree from Xidian University, Xi'an, China, in 1987, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 1996.

He was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Perth, WA, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Perth, from 1998 to 2002. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, Guilin, China. His current research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Huaxiang Zhang received the Ph.D. degree from Shanghai Jiaotong University, Shanghai, China, in 2004.

He is currently a Professor with the School of Information Science and Engineering, Shandong Normal University, Jinan, China where he was an Associated Professor with the Department of Computer Science, from 2004 to 2005. He has authored over 100 journal and conference papers and holds eight invention patents. His current research interests include machine learning, pattern recognition, evolutionary computation, and Web information processing.

Jeremiah D. Deng (M'00) received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 1989, and the M.Eng. and D.Eng. degrees from the South China University of Technology (SCUT), Guangzhou, China, in 1992 and 1995, respectively, under the supervision of the University of Hong Kong.

He has been a Lecturer with the SCUT, since 1995. He then joined the University of Otago, Dunedin, New Zealand, in 1999, as a Post-Doctoral Research Fellow, where he is currently an Associate Professor with the Department of Information Science. He has published around 100 refereed research papers in international conference proceedings and journals. His current research interests include machine learning, pattern recognition, and modeling and optimization of computer networks.

Yun Li (S'87–M'90) received the B.S. degree in radio electronics science from Sichuan University, Chengdu, China, in 1984, the M.Eng. degree in electronic engineering from the University of Electronic Science and Technology of China (UESTC), Chengdu, in 1987, and the Ph.D. degree in parallel processing for control engineering from the University of Strathclyde, Glasgow, U.K., in 1990.

He is currently a Professor with the School of Computer Science and Network Security, Dongguan University of Technology, Dongguan, China. He was a Professor with the School of Engineering, University of Glasgow, Glasgow, U.K. From 1989 to 1990, he was with the U.K. National Engineering Laboratory and Industrial Systems, Glasgow, and Control Ltd., Glasgow. He joined the University of Glasgow, in 1991, as a Lecturer. He served as the two-year Founding Director of the University of Glasgow at Singapore, Singapore, from 2011 to 2013. He was the Interim/Founding Director of the University's first joint programme in China, in 2013, with the UESTC. He has been a Visiting Professor with Kumamoto University, Kumamoto, Japan, UESTC, and Sun Yat-sen University, Guangzhou, China. He has supervised over 20 Ph.D. students and has over 200 publications.

Prof. Li has established Evolutionary Computation Workgroups for the IEEE Control System Society and European Network of Excellence in Evolutionary Computing (EvoNet) in 1998 and served on the Management Board of EvoNet from 2000 to 2005. He is a Chartered Engineer in U.K.

Jun Zhang (M'02–SM'08) received the Ph.D. degree in electrical engineering from the City University of Hong Kong, Hong Kong, in 2002.

He is currently a Professor with the South China University of Technology, Guangzhou, China. His current research interests include computational intelligence, cloud computing, wireless sensor networks, operations research, and power electronic circuits. He has authored seven research books and book chapters, and over 50 IEEE TRANSACTIONS papers in the above areas.

Prof. Zhang was a recipient of the National Science Fund for Distinguished Young Scholars in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He was also appointed as the Changjiang Chair professor in 2013. He is currently an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON CYBERNETICS. He is the Founding and Current Chair of the IEEE Guangzhou Subsection and the IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters. He is the Founding and Current Chair of ACM Guangzhou Chapter.