This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

A Level-based Learning Swarm Optimizer for Large Scale Optimization

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

***Abstract*—In pedagogy, teachers usually separate mixed-level students into different levels, treat them differently and teach them in accordance with their cognitive and learning abilities. Inspired from this idea, we consider particles in the swarm as mixed-level students and propose a level-based learning swarm optimizer to settle large scale optimization, which is still considerably challenging in evolutionary computation. At first, a level-based learning strategy is introduced, which separates particles into a number of levels according to their fitness values and treats particles in different levels differently. Then, a new exemplar selection strategy is designed to randomly select two predominant particles from two different higher levels in the current swarm to guide the learning of particles. The cooperation between these two strategies could afford great diversity enhancement for the optimizer. Further, the exploration and exploitation abilities of the optimizer are analyzed both theoretically and empirically in comparison with two popular particle swarm optimizers. Extensive comparisons with several state-of-the-art algorithms on two widely used sets of large scale benchmark functions confirm the competitive performance of the proposed optimizer in both solution quality and computational efficiency. Finally, comparison experiments on problems with dimensionality increasing from 200 to 2000 further substantiate the good scalability of the developed optimizer.**

***Index Terms*—Large scale optimization, particle swarm optimization, level-based learning swarm optimizer, exemplar selection, high dimensional problems**

I. Introduction

PARTICLE swarm optimization (PSO) has been extensively researched and also has been widely applied to solve real-world problems [1-4], since it was first proposed by Kennedy and Eberhart in 1995 [5, 6]. Imitating the swarm behaviors of social animals, such as bird flocking, particles in the swarm traverse the whole solution space to find the global optimum of the problem to be optimized.

Specifically, each particle in the swarm represents a candidate solution and is denoted by two attributes: position and velocity, which are updated as:

$$v_i^d \leftarrow wv_i^d + c_1 r_1 (pbest_i^d - x_i^d) + c_2 r_2 (nbest_i^d - x_i^d) \quad (1)$$

$$x_i^d \leftarrow x_i^d + v_i^d \quad (2)$$

where $\boldsymbol{X}_i=[x_i^1,...,x_i^d,...,x_i^D]$ and $\boldsymbol{V}_i=[v_i^1,...,v_i^d,...,v_i^D]$ are the position vector and the velocity vector of the ith particle respectively. $\boldsymbol{pbest}_i=[pbest_i^1,...,pbest_i^d,...,pbest_i^D]$ is its personal best position and $\boldsymbol{nbest}_i=[nbest_i^1,...,nbest_i^d,...,nbest_i^D]$ is the best position of its neighbors, which are determined by the adopted topology [7, 8]. As for the parameters, D is the dimension size, w is termed as the inertia weight [9], c_1 and c_2 are two acceleration coefficients [5], and r_1 as well as r_2 is uniformly randomized within [0,1]. Kennedy and Eberhart [6] considered the second part and the third part in the right of Eq. (1) as the cognitive component and the social component, respectively.

By means of Eq. (1), one particle in the swarm learns from its own experienced knowledge and the social knowledge to traverse the search space to seek the global optimum of the optimized problem. However, researchers found that the above learning strategy is not an efficient way to tackle complicated multimodal problems, because this strategy easily leads to stagnation or premature convergence [10].

To further improve the efficacy of PSO in handling complicated problems, many researchers sought inspirations from nature and human society and have proposed an ocean of novel learning or updating strategies for PSO [11-17]. To name a few, enlightened from the social learning in animal society, an incremental social learning strategy was put forward in [12] by adopting a population size increasing approach; inspired by the phenomenon that the interactive learning behavior takes place among different groups in human society, Qin *et al.* developed an interswarm interactive learning strategy in [16], where two swarms dynamically learn from each other when stagnation is detected; inspired from the orthogonal experimental design, an orthogonal learning PSO (OLPSO) [13] was designed by conducting orthogonal experimental design on ***pbest*** and ***gbest*** (or ***nbest***) to obtain more efficient exemplars for particles. In addition, Liang *et al.* developed a comprehensive learning PSO (CLPSO) [11] and further Lynn *et al.* devised heterogeneous CLPSO (HCLPSO) [18] to enhance the exploration and exploitation of CLPSO.

Although these PSO variants show better performance than the classical PSO, they remain effective only in low dimensional space. When encountering high dimensional

Manuscript received December 2, 2016; revised March 14, 2017, May 30, 2017 and August 3, 2017; accepted August 14, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61622206, Grant 61379061, and Grant 61332002, the Natural Science Foundation of Guangdong under Grant 2015A030306024, the Guangdong Special Support Program No. 2014TQ01X550, and the Guangzhou Pearl River New Star of Science and Technology No. 201506010002. (*Corresponding Authors: Wei-Neng Chen and Jun Zhang*).

Q. Yang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, 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 510006, China (email: cwnraul634@aliyun.com; junzhang@ieee.org).

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.

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

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

problems, their performance deteriorate drastically [19-22]. This phenomenon is usually a result of "the curse of dimensionality" [20]. On the one hand, as the dimension size grows, the search space increases exponentially. Such huge and wide space greatly challenges the search efficiency of the current PSO variants [21, 23]. On the other hand, increasing dimensionality may also bring in the explosively increased number of local optima surrounded by capacious local areas, which is especially common for large scale multimodal problems. Such phenomenon may give rise to great chance of premature convergence. Therefore, to solve high dimensional problems effectively, high diversity preservation is highly required for EAs to escape from local traps.

Taking inspirations from nature and human society as well, some researchers have proposed novel learning strategies for PSO [21, 23, 24] to deal with large scale optimization. Generally, these PSO variants can preserve higher diversity than the former PSO variants [11-16]. For consistency, they will be elucidated in details in the following section. Although these PSO variants are promising for large scale optimization, premature convergence is still the main challenge.

To solve large scale optimization problems more efficiently, this paper proposes a Level-based Learning Swarm Optimizer (LLSO) based on two motivations.

First, in education, it is common that different students have different cognitive or learning abilities, and thus teachers should treat their students differently in accordance with their aptitude [25, 26]. In particular, in the mixed-level learning methodology which has been widely used in education practice [25, 26], it is suggested that students should be grouped into different levels with tiered teaching and learning methods. Similarly, in one swarm, particles are usually in different evolution states, and particles in different states generally have different potential in exploring and exploiting the search space. Thus, they should be treated differently as well. Inspired from this, a level-based learning strategy is introduced into LLSO, which groups particles into different levels based on their fitness values and treats those in different levels differently.

Second, instead of using the historically best positions (such as ***pbest***, ***gbest***, or ***nbest***) to update particles, two popular and recent PSO variants, CSO [23] and SL-PSO [21], directly adopt predominant particles in the current swarm to guide the learning of particles. Since particles in the swarm are generally updated in each generation, the diversity of these two optimizers is greatly promoted and thus they show good performance in dealing with large scale optimization. However, these two optimizers utilize only one predominant particle to replace one exemplar in Eq. (1) to guide the learning of particles, while the other exemplar is the mean position of the swarm, which is shared by all particles and thus is not beneficial for further diversity enhancement. In the developed level-based learning strategy, since particles in different levels have diverse potential in exploration and exploitation, they possess diverse evolutionary information to evolve the swarm and thus could be utilized as candidates to respectively replace the two exemplars in Eq. (1) to direct the learning of particles. To this end, a new exemplar selection method is incorporated into the learning strategy, which first randomly selects two different higher levels and then selects one exemplar from each level, so that two diverse predominant particles in the swarm could be selected to guide the learning of particles. In this way, the search diversity is likely promoted.

Together, the proposed LLSO directly utilizes two predominant particles in the current swarm to guide the learning of particles. In this manner, this learning strategy can enhance the diversity of the swarm. In particular, it can compromise exploration and exploitation to search the space in two levels: the particle level and the swarm level. In the particle level, one particle can enhance its potential in exploiting the space by learning from the superior one between the two selected exemplars and consolidate its potential in exploring the space via learning from the relatively inferior one. In the swarm level, particles in different levels have different numbers of exemplars in higher levels to learn from, resulting in that particles in lower levels focus on exploring the space, while those in higher levels concentrate on exploiting the space. The exploration and exploitation abilities of LLSO are both analyzed theoretically and verified empirically in comparison with GPSO [6] and CSO [23].

To verify the efficiency and effectiveness of LLSO, extensive experiments are conducted by comparing LLSO with several state-of-the-art large scale algorithms on CEC'2010 [27] and CEC'2013 [28] large scale benchmark sets. Furthermore, experiments on the CEC'2010 [27] benchmark problems with dimensionality increasing from 200 to 2000 are performed to testify the scalability of LLSO.

The rest of this paper is organized as follows. Various related EAs dealing with large scale optimization are reviewed in Section II. Section III elucidates the whole framework of LLSO in details, following which is the theoretical analysis about its exploration and exploitation abilities in Section IV. Then, extensive experiments are conducted in Section V to verify the effectiveness, efficiency and good scalability of LLSO. Finally, conclusions and discussions are given in Section VI.

II. Related Work on Large Scale Optimization

Without loss of generality, in this paper, we consider the minimization problems defined as follows:

$$\min f(\boldsymbol{X}), \boldsymbol{X} = [x^1, x^2, \ldots, x^D] \tag{3}$$

where D is the number of variables to be optimized. In addition, the function value is taken as the fitness value of each particle.

With D increasing, the above defined problem becomes more and more difficult to optimize, because on the one hand, the search space is exponentially increased; on the other hand, the number of local optima surrounded by wide local areas may be also explosively increased [20, 21, 23], especially for multimodal problems [29, 30]. So far, to locate the global optima of high dimensional problems efficiently, researchers attempted to seek solutions from two perspectives: 1) proposing cooperative coevolutionay algorithms (CCEAs), which divide the whole decision vector into several groups and evolve each variable group separately, and 2) proposing novel updating strategies for traditional EAs, which evolve all variables as a whole and preserve high diversity to escape from local areas.

A. Cooperative Coevolutionary Algorithms (CCEAs)

Since Potter [31] proposed the cooperative coevolution (CC) framework, which adopts the divide-and-conquer technique to

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

decompose problems into smaller sub-problems, various CCEAs have come into being by combining CC with different EAs, such as cooperative coevolutionary PSO (CCPSO) [32, 33], and cooperative coevolutionary DE (DECC) [34].

Bergh and Engelbrecht [32] first combined CC with PSO and proposed CCPSO-S_K, which randomly divides the whole decision vector into D/K subcomponents with each containing K variables and then utilizes the canonical PSO to separately optimize each subcomponent. Following CCPSO-S_K, they further developed CCPSO-H_K, where the classical PSO and CCPSO-S_K update the swarm in an alternative manner. However, for different problems, the optimal number of subcomponents is usually different. To ameliorate this issue, Li and Yao [33] proposed CCPSO2 by designing a group size pool, which contains different group sizes.

Since in CCEAs, each variable group is individually optimized, the interdependent variables should be placed into the same group and optimized simultaneously [20]. This indicates that the decomposition strategy is the most crucial component for CCEAs to achieve good performance. As a consequence, the research on CCEAs mainly concentrates on devising a good decomposition strategy and thus, many decomposition strategies have shown up [20, 35-38]. Among these strategies, differential grouping (DG) [20] and its variants, XDG [37] and GDG[38] are the most popular ones because they can detect variable dependency and thus can separate variables into groups more accurately.

Though CCEAs are promising for large scale optimization, they encounter two limitations, which restrict their wide application. For one thing, the performance of CCEAs seriously relies on the decomposition strategy, and to detect the interdependency among variables, a good decomposer usually consumes a large number of function evaluations [20],[37],[38]. For another, a good CCEA usually costs plenty of function evaluations, particularly when the number of variable groups is large. This is because not only the adopted decomposition strategy consumes a large number of function evaluations, but also the optimization process consumes a lot of function evaluations to evolve variables so that satisfactory performance can be obtained.

B. Novel Learning or Updating Strategies for EAs

From the other perspective, some researchers are devoted to developing new learning or updating strategies, which can preserve high diversity, to aid traditional EAs to cope with large scale optimization problems.

Liang and Suganthan proposed a dynamic multi-swarm PSO [39], where the swarm is randomly divided into multiple small subswarms and then the local version PSO [40] is utilized to evolve each subswarm. Enlightened from the competition in human society, Cheng and Jin [41], [23] developed a novel competitive learning strategy. First, they applied this strategy into a multi-swarm PSO [41], where pairwise competition is performed between two particles randomly selected from two swarms. After the competition, the loser is updated by a convergence strategy, while the winner is updated through a mutation strategy. Subsequently, they introduced a competitive swarm optimizer (CSO) [23], where the pairwise competition is executed among particles in a single swarm, and only the loser in one competition is updated, while the winner enters the next generation directly. Specifically, the loser is updated as:

$$v_l^d \leftarrow r_1 v_l^d + r_2(x_w^d - x_l^d) + \phi r_3(\bar{x}^d - x_l^d) \quad (4)$$

$$x_l^d \leftarrow x_l^d + v_l^d \quad (5)$$

where $\boldsymbol{X}_l=[x_l^1,...,x_l^d,...x_l^D]$ and $\boldsymbol{V}_l=[v_l^1,...,v_l^d,...v_l^D]$ are the position and speed of the loser respectively; $\boldsymbol{X}_w=[x_w^1,...,x_w^d,...x_w^D]$ is the position of the winner; $\bar{\boldsymbol{x}}=[\bar{x}^1,...,\bar{x}^d,...\bar{x}^D]$ is the mean position of the swarm, r_1, r_2, and r_3 are three random variables within [0,1] and ϕ is one parameter controlling the influence of $\bar{\boldsymbol{x}}$.

Inspired from the social learning behavior among social animals, a social learning PSO (SL-PSO) [21] was developed. In this algorithm, all particles are sorted according to their fitness values and then for each particle, the first exemplar in Eq. (1) is randomly selected from all better particles, while the second exemplar is also the mean position of the whole swarm as CSO displayed in Eq. (4). Taking further observation on SL-PSO and CSO, we find that these two optimizers neither utilize ***pbest*** nor ***gbest*** (or ***nbest***) to guide the learning of particles. Instead, they directly adopt predominant particles in the current swarm and the mean position of the swarm to lead particles to find the global optima.

In addition, taking advantage of the invasive weed optimization algorithm [42] and the quantum-behaved particle swarm optimization algorithm [43], Lian *et al.* developed a quantum-behaved invasive weed optimization algorithm (QIWO) [44], which correspondingly adjusts and improves the quantum models of these two algorithms.

Except for PSO variants in handling large scale optimization, many other EA variants were also developed. Since too many works exist, we cannot review them all. Here, to save space, we only list some typical and recent works on large scale optimization. For a comprehensive review of large scale optimizers, readers can refer to [45] and [22].

Hansen and Ostermeier proposed an algorithm named CMA-ES [46], which makes use of adaptive mutation parameters through computing a covariance matrix and correlated step sizes in all dimensions to preserve high diversity. Although it is promising for high dimensional problems, it is very time-consuming owing to the computation of the covariance matrix with time complexity $O(D^2)$, where D is the dimension size. To relieve the high computational burden, its variant called sep-CMA-ES [47] came up, which only computes the diagonal elements of the covariance matrix, leading to the reduction of complexity from $O(D^2)$ to $O(D)$.

Subsequently, Molina *et al.* proposed a memetic algorithm named MA-SW-Chains [48], which combines a steady-state GA with a local search method. LaTorre *et al.* developed a multiple offspring generation framework [49, 50], named MOS, via hybridizing different algorithms to deal with different large scale optimization problems. In [51], Brest and Maučec developed a self-adaptive DE named jDElscop to solve large scale problems, which employs three mutation strategies and a population size reduction mechanism to evolve the population. Zhao *et al.* proposed another self-adaptive DE called SaDE-MMTS [52] by hybridizing the mutation strategy in JADE [53] with a modified multi-trajectory search (MMTS) algorithm. Then, Ali *et al.* introduced a multi-population DE called mDE-bES [54] to tackle large scale optimization. In this algorithm, the population is divided into independent

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

subgroups, and different subgroups are evolved with different mutation strategies.

Even though numerious works exist in dealing with large scale optimization, falling into local optima and permature convergence are still the main challenges in large scale optimizaiton. In this paper, we propose a Level-based Learning Swarm Optimizer (LLSO) to try to alleviate the above issue.

III. Level-based Learning Swarm Optimizer

A. Motivation

When the dimension size D becomes larger and larger (more than 500 [23]), optimization problems become more and more difficult to optimize. On the one hand, with D increasing, the computational complexity of the problem becomes higher and higher and the search space of the problem also increases exponentially, which takes an optimizer a larger number of fitness evaluations to locate the optima [21, 23]. On the other hand, for high dimensional multimodal problems, the number of local optima is generally explosively increased and it is likely that these local optima are surrounded by wide local areas, which may easily cause local traps or premature convergence for optimizers [22, 45]. Thus, to tackle this kind of problems efficiently, an optimizer is especially required to preserve high diversity, so that local traps can be avoided. At the same time, fast convergence is also a necessity for the optimizer, so that with limited resources, such as the restricted number of fitness evaluations, the global optimum can be fast located. However, these two requirements conflict with each other [23, 55]. As a consequence, a good optimizer should make a good compromise between these two aspects to fast traverse the search space.

In order to figure out an effective learning strategy, we seek inspirations from nature and human society. In particular, in pedagogy, different students generally have different cognitive or learning abilities, and thus teachers should treat these students differently in accordance of their aptitude [25, 26]. In particular, in the mixed-level learning methodology which has been widely used in education practice [25, 26], students should be grouped into different levels with tiered teaching and learning methods. Similarly, during the evolution, particles are usually in different evolution states and have different potential in exploring and exploiting the search space. Thus, they should be treated differently as well.

Moreover, taking close observation on SL-PSO [21] and CSO [23], we find that these two optimizers neither utilize ***pbest*** nor ***gbest*** (or ***nbest***) to guide the learning of particles. Instead, they directly adopt predominant particles in the current swarm to update particles and show good potential in dealing with high dimensional problems due to the enhanced diversity. However, these two optimizers utilize only one predominant particle to replace one exemplar in Eq. (1) to guide the learning of particles, while the other exemplar is the mean position of the swarm, which is shared by all particles and thus is not beneficial for further diversity enhancement. Since particles in different evolution states possess diverse potential in exploring and exploiting the search space, they could own diverse evolutionary information to guide the swarm to seek the optima and thus could be utilized as candidates to replace the two exemplars in Eq. (1) to update the swarm, so that the diversity could be further enhanced.

Motivated by the above phenomenon and observation, we propose a level-based learning strategy for PSO, leading to Level-based Learning Swarm Optimizer (LLSO), which separates particles into different levels, treats them differently and utilizes two predominant particles in the current swarm to guide the learning of particles to find the global optima. Accompanying with this learning strategy, a new exemplar selection method is also developed to aid LLSO. The concrete elucidation of each component is presented as follows.

B. Level-based Learning

During the evolution, particles are usually in different evolution states, and have different potential in exploring and exploiting the search space. To tell them apart, we first partition particles into different levels according to their fitness values.

Assume that NP particles are divided into NL levels with each level denoted by L_i $(1 \le i \le NL)$. Before the partition, particles in the swarm are first sorted in ascending order of fitness as in SL-PSO [21]. Then, better particles belong to higher levels and the higher the level is, the smaller level index it has. So, L_1 is the highest level, and L_{NL} is the lowest level. To make it simple, we assume that all levels have the same number of particles. This number is called "level size" and denoted by LS. Clearly, $LS=NP/NL$[1].

Subsequently, we take deep insight into particles in different levels. On the one hand, more promising positions usually can be found around better particles in the current swarm [21, 23, 56]. In other words, particles in higher levels usually hold more beneficial information to guide the swarm towards the global optimum area. Consequently, particles in higher levels should guide those in lower levels to search the whole solution space, so that fast convergence can be achieved and promising positions can be located. This is the first idea behind the level-based learning strategy (LL).

On the other hand, observing particles in different higher levels, we find that the higher the level that a particle belongs to, the more likely the particle may be close to the global optimum area. That is, particles in different levels have different strength in exploitation. Likewise, particles in different levels have different strength in exploration. In general, exploration and exploitation are in the opposite direction [21]. In other words, particles having more potential in exploitation usually have less potential in exploration, and vice versa. So a particle from a lower level should learn from those from different higher levels to make a compromise between exploration and exploitation. This is the second idea behind the level-based learning strategy.

Combining the above two together, the framework of LL is displayed in Fig. 1. From this figure, we can see that particles in lower levels can potentially learn from all those in higher levels, and the number of candidate exemplars for particles in different levels is different. Specifically, as the level that a particle belongs to goes higher, this particle has fewer particles in the higher levels in total to learn from, which matches the expectation that better particles should do more exploitation rather than exploration.

[1] Note that the whole swarm may not be equally partitioned by NP/NL. In this situation, we just add the $NP\%NL$ particles into the lowest level.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

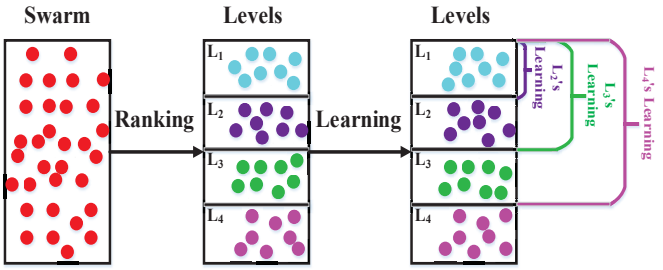


Fig. 1. The framework of the Level-based Learning (LL) strategy. First, particles in the swarm are sorted in ascending order of fitness and then they are equally partitioned into 4 levels (L_1, L_2, L_3 and L_4). Then, particles in L_4 learn from those in L_1, L_2, and L_3, particles in L_3 learn from those in L_1 and L_2, and particles in L_2 learn from those in L_1. It should be noticed that in order to protect the most promising particles from being wrongly updated, particles in L_1 are not updated and directly enter the next generation.

Overall, this level-based mechanism may encourage more exploration among particles in lower levels and more exploitation among those in higher levels. The effectiveness of the LL strategy will be further reinforced by the random selection mechanism for exemplars to be presented next.

C. Exemplar Selection

Besides the learning strategy, another key component for PSO is the exemplar selection strategy. As aforementioned, particles in different levels perform different roles in the evolution process. Generally, superior particles show more potential in exploitation, so they should be used to guide the search direction. While for inferior particles, even though they perform relatively badly in exploiting, they usually show more potential in exploring more directions and larger space, which is potentially profitable for dragging particles away from local areas. Enlightened by these, we propose a new exemplar selection method to select two different exemplars to replace ***pbest*** and ***nbest*** in Eq. (1) to update particles.

To utilize the property that particles in different levels have different strength in exploration and exploitation, we allow each particle in level L_i to learn from two particles $\boldsymbol{X}_{rl_1,k_1}$ and $\boldsymbol{X}_{rl_2,k_2}$ randomly selected from two different higher levels L_{rl_1} and L_{rl_2}, respectively, where rl_1 and rl_2 are randomly selected from $[1, i\text{-}1]$ and k_1 and k_2 are randomly selected from $[1, LS]$. Then, to take advantage of the property that superior particles have more potential in guiding the search direction while inferior ones have more potential in helping particles escape from local traps, with the assumption that rl_1 is higher than rl_2, we use the superior one between $\boldsymbol{X}_{rl_1,k_1}$ and $\boldsymbol{X}_{rl_2,k_2}$, namely $\boldsymbol{X}_{rl_1,k_1}$ to replace ***pbest*** in Eq. (1) and use the inferior one, namely $\boldsymbol{X}_{rl_2,k_2}$ to substitute ***nbest*** in Eq. (1).

Note that, in order to further promote the potential in enhancing the diversity, we use randomness on both the selection of two different higher levels (rl_1 and rl_2) and the selection of exemplars from the selected levels (k_1 and k_2).

On the one hand, this exemplar selection strategy provides two exemplars from different higher levels for each particle in lower levels, offering a potential compromise between exploration and exploitation. On the other hand, the randomness embedded in the level selection and the exemplar selection may contribute to enhancing diversity, which plays a significant role in large scale optimization [23].

D. LLSO

Combining the above two strategies together, LLSO is developed with the update of particles defined as follows:

$$v_{i,j}^d \leftarrow r_1 v_{i,j}^d + r_2 (x_{rl_1,k_1}^d - x_{i,j}^d) + \phi r_3 (x_{rl_2,k_2}^d - x_{i,j}^d) \quad (6)$$

$$x_{i,j}^d \leftarrow x_{i,j}^d + v_{i,j}^d \quad (7)$$

where $\boldsymbol{X}_{i,j}=[x_{i,j}^1,\ldots,x_{i,j}^d,\ldots,x_{i,j}^D]$ is the position of the jth particle from the ith level L_i and $\boldsymbol{V}_{i,j}=[v_{i,j}^1,\ldots,v_{i,j}^d,\ldots,v_{i,j}^D]$ is its speed. $\boldsymbol{X}_{rl_1,k_1}=[x_{rl_1,k_1}^1,\ldots,x_{rl_1,k_1}^d,\ldots,x_{rl_1,k_1}^D]$ randomly selected from level L_{rl_1} and $\boldsymbol{X}_{rl_2,k_2}=[x_{rl_2,k_2}^1,\ldots,x_{rl_2,k_2}^d,\ldots,x_{rl_2,k_2}^D]$ randomly selected from level L_{rl_2} are the two selected exemplars with rl_1 and rl_2 denoting two different higher level indexes selected within $[1, i\text{-}1]$, and k_1 and k_2 representing two particle indexes randomly selected within $[1, LS]$. r_1, r_2, and r_3 are three random variables ranging within $[0, 1]$ and ϕ is the control parameter within $[0,1]$ in charge of the influence of the second exemplar. Note that $rl_1<rl_2<i$, which indicates that L_{rl_1} is higher than L_{rl_2}, and both are higher than L_i, and also suggests that $\boldsymbol{X}_{rl_1,k_1}$ is better than $\boldsymbol{X}_{rl_2,k_2}$ and both are better than $\boldsymbol{X}_{i,j}$.

Generally, superior particles have more potential in exploiting the search space, while inferior particles have more potential in exploring the search space. Thus, the learning

Algorithm 1: The framework of LLSO

Input: swarm size *NP*, number of levels *NL*, level size *LS*, maximum number of fitness evaluations *MAX_FES*, control parameter ϕ.

Output: The final solution $\boldsymbol{x}$ and its fitness $f(\boldsymbol{x})$

```
fes = 0;
Initialize the swarm randomly and calculate the fitness values of particles;
fes += NP;
x is the best particle of the swarm and f(x) is its fitness;
While fes < MAX_FES do
    Sort particles in ascending order of fitness and divide them into NL levels;
    //Update particles in L_NL,..., L_3;
    For i = {NL,...,3} do
        For j = {1, ..., LS} do
            Select two levels from the top (i-1) levels: rl_1, rl_2;
            If ( rl_2<rl_1) then
                Swap (rl_1, rl_2);
            End If
            Randomly select two particles from rl_1, rl_2: X_{rl_1,k_1}, X_{rl_2,k_2};
            Update particle X_{i,j} according to Eq. (6) and Eq. (7);
            Calculate the fitness value f(X_{i,j}) of this particle;
            If (f(X_{i,j})<f(x)) then
                x = X_{i,j};
            End If
        End For
        fes += LS;
    End For
    //Update the second level
    For j = {1, ..., LS} do
        Select two particles from the first level: X_{1,k_1}, X_{1,k_2};
        If (f(X_{1,k_2}) <f(X_{1,k_1}) ) then
            Swap (X_{1,k_1}, X_{1,k_2});
        End If
        Update particle X_{2,j} according to Eq. (6) and Eq. (7);
        Calculate the fitness value f(X_{2,j}) of this particle;
        If (f(X_{2,j}) <f(x)) then
            x = X_{2,j};
        End If
    End For
    fes += LS;
End While
```

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

strategy displayed in Eq. (6) gives rise to a potential compromise between exploration and exploitation for each particle. This is because the second part in the right hand of Eq. (6) allows one particle to promote its potential in exploitation by learning from a superior exemplar, while the third part enables the particle to enhance its potential in exploration through learning from a relatively inferior exemplar, and the degree of such learning is controlled by the parameter ϕ.

Additionally, both the second and third parts in the right hand of Eq. (6) can be seen as the cognitive parts like in PSO (Eq. (1)). Though there is no obvious social learning part in LLSO, actually, the social part is embedded in these two items because, on the one hand, the two levels where the selected exemplars come are randomly chosen from all higher levels; on the other hand, the two exemplars are randomly selected from the corresponding levels. Such two random selections possibly offer a special kind of social learning.

Obviously Eq. (6) is not directly suitable for the update of particles in the first and second levels. To deal with this situation, we adopt different extra techniques for the two levels.

First, since the particles in the first level are the best of the whole swarm in the current generation and better solutions are usually found near these ones, we just leave these particles unchanged to preserve the most useful information and protect them from being weakened. Thus, the particles in the first level directly enter the next generation.

Second, as for the particles in the second level, a similar adaption follows. Instead of randomly choosing two exemplars from two randomly selected higher levels, the two exemplars for these particles are both randomly selected from the first level. Then, the superior one acts as the first exemplar and the inferior one acts as the second exemplar in Eq. (6).

The pseudo code of LLSO is outlined in **Algorithm 1**, which is simple to implement due to the maintenance of the classical PSO framework. In this algorithm, Lines 7 to 21 are for the update of particles in levels L_{NL} to L_3, while Lines 22 to 32 are for the update of particles in the second level.

E. Differences between LLSO and Other PSO Variants.

The main unique property of LLSO is the level-based learning mechanism along with the exemplar selection method. It treats particles differently and directly utilizes two predominant particles from two different higher levels in the swarm to guide the learning of particles in lower levels by taking advantage of their different strength in exploration or exploitation. Specifically, the following characteristics make it distinguishable from the current PSO variants.

1) **Particles are grouped into different levels and those in different levels are treated differently via learning from different numbers (in total) of particles in higher levels.** Specifically, the lower the level one particle belongs to, the more the candidate exemplars (both exemplars in Eq. (6)) this particle could learn from, and vice versa. Through this, particles in lower levels could focus on exploring the search space, while those in higher levels could concentrate on exploiting the search space. However, in most PSO variants [11-14],[41, 57, 58], [59], particles have the same number of candidate exemplars to learn from and thus are treated equally.
2) **Two current superior particles act as the exemplars to guide the learning of inferior particles, which is beneficial for exploration enhancement.** Instead of learning from ***pbest***, ***nbest***, or ***gbest*** in most PSO variants, such as hierarchical PSO [59], multi-swarm PSO variants [41, 57, 58] and new learning strategy based PSOs [11-14], particles in LLSO learn from the superior ones in the current swarm. ***pbest***, ***nbest***, or ***gbest*** may easily lead to premature convergence [23], because they may remain unchanged for many generations, especially when the evolution goes into late stages on multimodal problems. However, particles in the swarm are usually updated at each generation. Thus LLSO may preserve higher diversity and thus has relatively less probability to fall into local areas. In addition, different from CSO [23] and SL-PSO [21] which only adopt one superior particle and the mean position of the swarm (shared by all particles) to guide the learning of particles, LLSO directly utilizes two superior particles randomly selected from two different higher levels to guide the learning of particles, leading to higher diversity preservation than these two optimizers.
3) **Two kinds of compromises between exploration and exploitation exist in LLSO.** Observing Eq. (6) and Eq. (7), we can find LLSO could compromise exploration and exploitation to search the space in two aspects. 1) **Particle-level compromise**: Each particle in lower levels can enhance its potential in exploitation by learning from the better one between the two superior exemplars, and at the same time consolidate its potential in exploration by learning from the relatively worse one. Thus, a compromise in exploring and exploiting the search space exists in the learning process of each particle. 2) **Swarm-level compromise**: Particles in different levels have different numbers of candidate exemplars to learn from. More specifically, particles in the lowest level have the most candidate exemplars to learn from, while particles in the second level have the fewest candidate exemplars to learn from and particles in the highest level (namely the first level) are not updated and directly enter the next generation for preserving the best information. Thus, we can see that particles in the lower levels mainly concentrate on exploring the search space, while particles in the higher levels mainly focus on exploiting the search space. Thus, a compromise in exploring and exploiting the search space exists in the whole swarm, which many other PSO variants do not have.
4) **Last but not at least, two hierarchical randomness exists in the exemplar selection.** First, two higher levels are randomly selected. Then, based on the selected levels, one random particle is selected from each level and thus two different particles in total are randomly selected. Together, the randomness of the level selection and that of the particle selection cooperate with each other, and can potentially provide particles with diverse exemplars, which benefits the diversity promotion.

F. Complexity Analysis

Given a fixed number of fitness evaluations, the time complexity of an EA [14, 21, 23] is generally calculated by analyzing the extra time in each generation without considering the time of function evaluations, which is problem-dependent.

Thanks to the maintenance of the algorithmic simplicity of PSO in LLSO, it is straightforward to compute the time complexity of LLSO. From **Algorithm 1**, we can see that it takes $O(NPlog(NP)+NP)$ to rank the swarm and divide the swarm into NL levels at each generation in Line 6. During the update of particles in all levels, except for those in the first level that directly enter the next generation, it takes $O(NP\times D)$ (Lines 7 to 32). Overall, we can see that LLSO only takes extra $O(NPlog(NP)+NP)$ in each generation compared with PSO, which takes $O(NP\times D)$ in each generation.

As for the space complexity, LLSO needs much smaller space than PSO, because it does not store the personal best position of each particle, which takes $O(NP\times D)$ space.

In conclusion, LLSO remains computationally efficacious in time and is relatively more efficient in space in comparison with the classical PSO.

G. Dynamic Version of LLSO

Comparing LLSO (Eq. (6)) with PSO (Eq. (1)), we find that LLSO only introduces two parameters that need fine-tuning, namely the number of levels NL and the control parameter ϕ.

Given the population size is NP, a small NL gives rise to a large number of particles in each level. This may bring two consequences: 1) promoting diversity in the exemplar selection conducted on the two selected levels, owing to the large number of particles in each level; and 2) reducing diversity in the level selection owing to the small number of levels. On the contrary, a large NL brings two opposite consequences: 1) enhancing diversity in the level selection, on account of the large number of levels; and 2) reducing diversity in the exemplar selection, due to the small number of particles in each level.

Comparing these two kinds of diversity, we consider that they play different roles in the evolution process. Compared with the diversity in the exemplar selection, the diversity in the level selection is more important when the swarm explores the search space or when the swarm falls into local areas and thus needs to jump out. This is because compared with the diversity in the exemplar selection, the diversity in the level selection can provide particles to be updated with more diverse exemplars that preserve diverse potential in exploration and exploitation. On the contrary, when exploiting the search space, the diversity in the exemplar selection becomes more important, which is beneficial for the swarm to exploit the search space more intensively without serious loss of diversity.

Therefore, we can see that for a single problem, the proper NL may vary during the evolution process. Let alone that the proper NL for different problems with different features is different. This motivates us to design a dynamic setting for NL.

In this paper, for simplicity, we design a pool containing different integers to realize the dynamism of NL, which is denoted as $\boldsymbol{S}=\{l_1,...,l_s\}$with s different candidate numbers of levels. Then, at each generation, LLSO will select a number from the pool based on their probabilities, and at the end of the generation, the performance of LLSO with this level number is recorded to update the probability of this number. With this mechanism, LLSO can select a proper NL despite of different features of different problems or different evolution stages for a single problem.

In order to compute the probabilities of different level numbers in $\boldsymbol{S}$, we define a record list $\boldsymbol{R_s}=\{r_1,...,r_s\}$, where each $r_i\in\boldsymbol{R_s}$ is associated with each $l_i\in\boldsymbol{S}$, to record the relative performance improvement under the selected l_i. At the initialization stage, each $r_i\in\boldsymbol{R_s}$ is set to 1, and then, each r_i is updated at each generation as follows as in [36]:

$$r_i=\frac{|F-\tilde{F}|}{|F|} \tag{8}$$

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 $\boldsymbol{P_s}=\{p_1,...,p_s\}$ is computed as in [36]

$$p_i=\frac{e^{7*r_i}}{\sum_{j=1}^{s}e^{7*r_j}} \tag{9}$$

Based on $\boldsymbol{P_s}$, we conduct the roulette wheel selection to select a number from $\boldsymbol{S}$ as the level number in each generation.

Observing Eq. (8) and Eq. (9), we can notice that: 1) The value of each r_i is within [0,1] since Eq. (8) 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 level number in this generation is very appropriate and thus should have a high probability to be selected in the next generation, which is implied by the probability computed in Eq. (9). 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 level number in this generation is not so advisable and thus the probability of this selection should be small, which can be implied by the probability computed in Eq. (9) as well. In this way, LLSO can potentially make an appropriate choice of NL for different problems or for a single problem at different stages.

As for our algorithm, when NL is fixed, it is denoted as LLSO; and when it uses a dynamic NL, we denote it as DLLSO. As shown later in Section V.C, the performance comparison between these two versions favors DLLSO.

IV. Theoretical Analysis

In this section, we take investigation about LLSO by analyzing its exploration and exploitation abilities via making comparisons with the global PSO (GPSO) [6] and one recent and popular PSO variant named CSO [23].

A. Exploration Ability

Exploration plays an important role when the swarm explores the search space. Enhancing the exploration ability of an EA is to promote the diversity of the swarm, so that it can escape from local areas and find the global or promising areas easily. In particular, the exploration ability is considerably important when EAs tackle multimodal problems or when the swarm needs to jump out of local areas, so that stagnation or premature convergence can be avoided. The exploration ability of EAs can be implied by the diversity of the exemplars used to guide the learning or updating of particles or individuals [23].

To investigate the exploration ability of LLSO, we rewrite Eq. (6) as follows:

$$v_{i,j}^{d} \leftarrow r_1 v_{i,j}^{d}+\theta_1(p_1-x_{i,j}^{d}) \tag{10}$$

$$\theta_1 = r_2+\phi r_3 \tag{11}$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

$$p_1 = \frac{r_2}{r_2 + \phi r_3} x^d_{rl_1,k_1} + \frac{\phi r_3}{r_2 + \phi r_3} x^d_{rl_2,k_2} \tag{12}$$

Similarly, we can also rewrite the update formula of GPSO (utilizing ***gbest*** to replace ***nbest*** in Eq. (1)) into Eq. (13) and that of CSO (Eq. (4)) into Eq. (16):

$$v^d_i \leftarrow w v^d_i + \theta_2 (p_2 - x^d_i) \tag{13}$$

$$\theta_2 = c_1 r_1 + c_2 r_2 \tag{14}$$

$$p_2 = \frac{c_1 r_1}{c_1 r_1 + c_2 r_2} pbest^d_i + \frac{c_2 r_2}{c_1 r_1 + c_2 r_2} gbest^d \tag{15}$$

$$v^d_l \leftarrow r_1 v^d_l + \theta_3 (p_3 - x^d_l) \tag{16}$$

$$\theta_3 = r_2 + \phi r_3 \tag{17}$$

$$p_3 = \frac{r_2}{r_2 + \phi r_3} x^d_w + \frac{\phi r_3}{r_2 + \phi r_3} \bar{x}^d \tag{18}$$

From Eq. (10), Eq. (13) and Eq. (16), we can see that the difference between $p_i(i=1,2,3)$ and the particle to be updated provides the main source of diversity. First, comparing Eq. (10) with Eq. (13) and Eq. (16), we can see that LLSO has potential to preserve higher diversity. On the one hand, as for the first part in p_1, p_2, and p_3, the randomly selected exemplar $\boldsymbol{X}_{rl_1,k_1}$ offers chances for each particle in lower levels to learn from various better particles in LLSO. However, in GPSO, ***pbest*** of each particle is updated only when the particle finds a better position, which indicates that it is possible that ***pbest*** of the particle may be unchanged for many generations. In CSO, the loser can only learn from its corresponding winner. Therefore, in terms of the first part, LLSO and CSO preserve competitive or comparable diversity and both potentially own higher diversity than GPSO.

On the other hand, as for the second part of $p_i(i=1,2,3)$, ***gbest*** in GPSO is updated only when the swarm finds a better position. It is more likely that ***gbest*** remains unchanged than ***pbest***. In addition, ***gbest*** is shared by all particles. These two limitations do great harm to the diversity maintenance for GPSO [23]. For CSO, though the mean position of the swarm $\bar{\boldsymbol{x}}$ is updated at each generation, it is also shared by all particles. However, in LLSO, the second exemplar $\boldsymbol{X}_{rl_2,k_2}$ is randomly selected for each particle. Thus, in terms of the second part, LLSO probably possesses higher diversity.

In addition, compared with other PSO variants that use ***nbest*** to guide the learning of particles [7, 8] or that divide the swarm into sub-swarms and then use ***gbest*** or the center of the sub-swarm to guide the updating of particles [58], LLSO still potentially preserves better exploration ability, because ***nbest*** or ***gbest*** of a sub-swarm may remain unchanged for many generations as well.

In short, we can see that the diversity of the exemplars used to guide the learning of particles in LLSO is potentially higher, which may benefit for strengthening the exploration ability. Thus, LLSO can potentially find the promising areas faster and have greater chance to jump out of local optimum areas.

B. Exploitation Ability

With limited computational resources, such as function evaluations, exploitation is necessary when the swarm exploits the searching areas. Enhancing the exploitation ability of an EA is to fully exploit the found promising areas fast, so that better solutions can be located as fast as possible. The exploitation ability is very important when an EA deals with simple unimodal functions or when the swarm finds the global optimum areas. Generally, the exploitation ability can be indicated by the difference between the exemplar and the updated particle [23]. The smaller the difference is, the more the particle focuses on exploiting the area.

To analyze the exploitation ability of LLSO, we assume that for the jth particle $\boldsymbol{X}_{i,j}$ from the ith level L_i, two random exemplars $\boldsymbol{X}_{rl_1,k_1}$ and $\boldsymbol{X}_{rl_2,k_2}$ are selected from two randomly selected higher levels L_{rl_1} and L_{rl_2} ($rl_1<rl_2<i$). Then, we have

$$f(\boldsymbol{X}_{rl_1,k_1}) \le f(\boldsymbol{X}_{rl_2,k_2}) \le f(\boldsymbol{X}_{i,j}) \tag{19}$$

Comparing $\boldsymbol{X}_{rl_1,k_1}$ and $\boldsymbol{X}_{rl_2,k_2}$ with ***pbest*** and ***gbest*** defined in PSO, we have the following formula:

$$\begin{cases} f(\boldsymbol{gbest}) \le f(\boldsymbol{pbest}_{i,j}) \le f(\boldsymbol{X}_{i,j}) \\ f(\boldsymbol{gbest}) \le f(\boldsymbol{pbest}_{rl_1,k_1}) \le f(\boldsymbol{X}_{rl_1,k_1}) \\ f(\boldsymbol{gbest}) \le f(\boldsymbol{pbest}_{rl_2,k_2}) \le f(\boldsymbol{X}_{rl_2,k_2}) \end{cases} \tag{20}$$

where $\boldsymbol{pbest}_{i,j}$, $\boldsymbol{pbest}_{rl_1,k_1}$ and $\boldsymbol{pbest}_{rl_2,k_2}$ are the personal best positions of $\boldsymbol{X}_{i,j}$, $\boldsymbol{X}_{rl_1,k_1}$ and $\boldsymbol{X}_{rl_2,k_2}$, respectively.

When it reaches the late stages where all particles may converge together, the following relationship holds:

$$\boldsymbol{pbest}_{rl_1,k_1} \approx \boldsymbol{pbest}_{rl_2,k_2} \approx \boldsymbol{pbest}_{i,j} \approx \boldsymbol{gbest} \tag{21}$$

Therefore, for GPSO, we have

$$\begin{aligned} \Delta F_{GPSO} &= |f(\boldsymbol{X}_{i,j}) - f(\boldsymbol{gbest})| \\ &= |f(\boldsymbol{X}_{i,j}) - f(\frac{\boldsymbol{gbest} + \boldsymbol{gbest}}{2})| \\ &\approx |f(\boldsymbol{X}_{i,j}) - f(\frac{\boldsymbol{gbest} + \boldsymbol{pbest}_{i,j}}{2})| \\ &= |f(\boldsymbol{X}_{i,j}) - f(p'_2)| \end{aligned} \tag{22}$$

where p'_2 is the expected value of p_2 in Eq. (15).

Similarly, for CSO and LLSO, we can derive the following:

$$\begin{aligned} \Delta F_{CSO} &= |f(\boldsymbol{X}_{i,j}) - f(\boldsymbol{X}_{w_{i,j}})| \\ &= |f(\boldsymbol{X}_{i,j}) - f(p'_3)| \end{aligned} \tag{23}$$

$$\begin{aligned} \Delta F_{LLSO} &= |f(\boldsymbol{X}_{i,j}) - f(\boldsymbol{X}_{rl_1,k_1})| \\ &= |f(\boldsymbol{X}_{i,j}) - f(p'_1)| \end{aligned} \tag{24}$$

where $\boldsymbol{X}_{w_{i,j}}$ is the corresponding winner of $\boldsymbol{X}_{i,j}$ in CSO, p'_3 is the expected value of p_3 with ϕ=0 for CSO in Eq. (18) and p'_1 is the expected value of p_1 with ϕ=0 for LLSO in Eq. (12). Since $\boldsymbol{X}_{rl_1,k_1}$ is selected from level L_{rl_1}, which is higher than L_i, where $\boldsymbol{X}_{i,j}$ comes, the expected value of $\boldsymbol{X}_{rl_1,k_1}$ is better than that of $\boldsymbol{X}_{w_{i,j}}$, which is only better than $\boldsymbol{X}_{i,j}$. Therefore, we have $f(\boldsymbol{X}_{rl_1,k_1}) \le f(\boldsymbol{X}_{w_{i,j}})$.

Combining the above formula together, we can derive:

$$\Delta F_{LLSO} \le \Delta F_{CSO} \le \Delta F_{GPSO} \tag{25}$$

Such formula indicates that compared with GPSO and CSO, LLSO potentially has a better exploitation ability to refine solutions within a smaller gap between two positions whose fitness values are very similar.

The above analysis has separately demonstrated that LLSO can preserve good exploration and exploitation abilities. However, during the evolution process, these two abilities

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

usually conflict with each other. Thus, during evolution, an EA generally needs to make a compromise between these two abilities to search the space [16, 60, 61]. It should be mentioned that such a compromise should not be fixed, but be dynamically adjusted according to different features of the problems to be optimized or different requirements in the evolution process. In Section V.A, the good exploration and exploitation abilities of LLSO will be verified empirically in comparison with GPSO [6] and CSO [23].

V. Experiments

To verify the feasibility and efficiency of the proposed LLSO, a series of experiments are conducted on two widely used sets of large scale optimization problems: the CEC'2010 [27] and the CEC'2013 [28] benchmark sets. The latter is the extension of the former through introducing new features, such as overlapping functions. Consequently, functions in the latter set are much more complicated and harder to optimize. The main properties of these two function sets are summarized in Table SI and Table SII in the *Supplemental Material*, respectively. For details of these functions, readers are referred to [27], [28].

In this section, we first empirically substantiate the good exploration and exploitation abilities of LLSO in Section V.A. Then, we respectively investigate the key parameter settings for DLLSO in Section V.B and the influence of the dynamism of *NL* on LLSO in Section V.C. After all the preliminary investigation, we make comparisons between DLLSO and other state-of-the-art algorithms dealing with large scale optimization in Section V.D. In Section V.E, the scalability comparison between DLLSO and the compared algorithms is conducted on the CEC'2010 benchmark functions with dimensionality increasing from 200 to 2000. At last, the computational time comparison is made between DLLSO and some compared algorithms on the CEC'2010 problems with the dimensionality increasing from 200 to 2000 as well.

In addition, unless otherwise stated, the maximum number of fitness evaluations is set to $3000\times D$ (where D is the dimension size). What's more, for fair comparisons, median, mean and standard deviation (Std) values over 30 independent runs are used to evaluate the performance of different algorithms. In the comparisons between two different algorithms, Wilcoxon rank sum test is performed at a significance level of α=0.05.

Additionally, it is worth mentioning that all algorithms are conducted on a PC with 4 Intel(R) Core(TM) i5-3470 3.20GHz CPUs, 4Gb memory and Ubuntu 12.04 LTS 64-bit system.

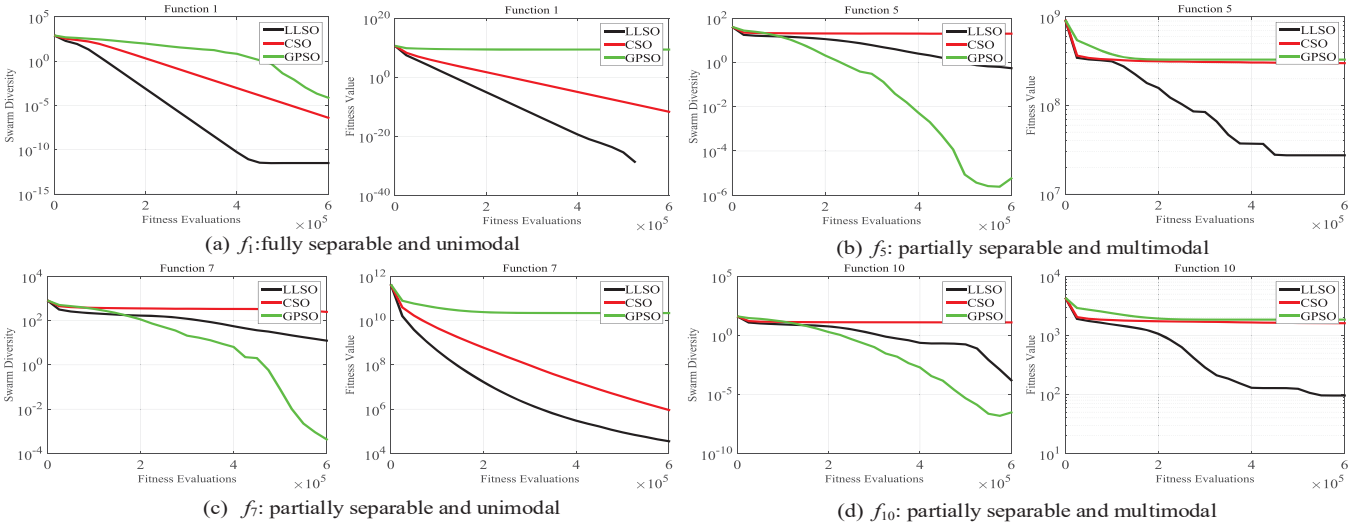
A. Exploration and Exploitation Investigation in LLSO

Before experiments, it should be noted that exploration and exploitation generally conflict with each other. Thus, an EA generally needs to make a compromise between these two aspects to search the space. In general, such a compromise should be dynamically adjusted according to different features of problems or different requirements in the evolution process [16, 60]. Besides, this compromise does not mean that exploration and exploitation should be the same or similar during all evolution stages or on all problems either.

When coping with unimodal problems, exploitation should be put properly more emphasis to seek fast convergence. However, when tackling multimodal problems, exploration should be properly biased to avoid falling into local areas. For a single problem, when most particles locate in local areas, exploration should be appropriately biased to let the swarm jump out of local areas. Nevertheless, when the swarm finds promising areas, exploitation should be appropriately biased to refine the obtained solutions [16, 60].

Then, to verify that LLSO can preserve good exploration and exploitation abilities and could compromise these two abilities properly, we conduct comparison experiments among LLSO, CSO, and GPSO on four CEC'2010 benchmark functions (fully separable and unimodal F_1, partially separable and multimodal F_5, partially separable and unimodal F_7, and partially separable and multimodal F_{10}) with 200 dimenisons in regard to swarm diversity along with the global best fitness value. These functions are selected because we want to make a comprehensive comparsion on various kinds of functions, like fully separable, partially separable, unimodal or multimodal.

In this paper, the diversity is measured as follows [23, 62]:



(a) f_1:fully separable and unimodal (b) f_5: partially separable and multimodal

(c) f_7: partially separable and unimodal (d) f_{10}: partially separable and multimodal

Fig. 2. Swarm diversity and the global best fitness value comparison among LLSO, CSO and GPSO on four functions F_1, F_5, F_7, and F_{10} with 200 dimensions. Please note that for F_1, the fitness value of LLSO is not plotted after the number of fitness evaluations reaches about 5×10^5. This is because the global best fitness value becomes 0, arriving at the global optimum of F_1.

$$D(\boldsymbol{X}) = \frac{1}{NP}\sum_{i=1}^{NP}\sqrt{\sum_{d=1}^{D}(x_i^d - \bar{x}^d)^2} \quad (26)$$

$$\bar{x}^d = \frac{1}{NP}\sum_{i=1}^{NP} x_i^d \quad (27)$$

where $D(\boldsymbol{X})$ represents the diversity of the swarm $\boldsymbol{X}$, and $\bar{\boldsymbol{x}}$ is the mean position of the swarm.

Fig. 2 shows the comparison results of the three algorithms on the four functions with the maximum number of fitness evaluations set as $3000\times D=6\times10^5$. For fairness, the population size is set 300 for all algorithms. From this figure, we can obtain the following findings.

TABLE I
COMPARISON RESULTS OF THE COMPARED ALGORITHMS ON 1000-*D* CEC'2010 FUNCTIONS WITH 3×10^6 FITNESS EVALUATIONS.

Function	Quality	DLLSO	CSO	SL-PSO	MA-SW-Chains	DMS-L-PSO	CCPSO2	DECC-G	MLCC	DECC-DG
F_1	Median	2.93E-22	4.40E-12	7.90E-18	8.23E-21	1.61E+07	7.80E-01	3.53E-07	1.66E-14	1.42E+02
	Mean	3.13E-22	4.50E-12	8.73E-18	9.75E-20	1.63E+07	2.96E+00	3.54E-07	8.65E-13	1.88E+04
	Std	8.03E-23	5.94E-13	3.30E-18	3.33E-19	1.41E+06	6.68E+00	1.44E-07	2.97E-12	4.66E+04
	p-value	-	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_2	Median	9.85E+02	7.33E+03	1.93E+03	5.06E+02	5.53E+03	4.25E+00	1.32E+03	2.43E+00	4.46E+03
	Mean	9.82E+02	7.42E+03	1.93E+03	5.74E+02	5.45E+03	4.30E+00	1.33E+03	2.89E+00	4.43E+03
	Std	4.39E+01	2.86E+02	1.12E+02	1.41E+02	5.38E+02	1.11E+00	2.55E+01	1.52E+00	1.87E+02
	p-value	-	**3.02E-11**$^+$	**3.02E-11**$^+$	3.02E-11$^-$	**3.02E-11**$^+$	3.02E-11$^-$	**3.02E-11**$^+$	3.02E-11$^-$	**3.02E-11**$^+$
F_3	Median	2.89E-14	2.53E-09	1.85E+00	9.42E-13	1.56E+01	4.16E-03	1.14E+00	6.24E-10	1.66E+01
	Mean	2.76E-14	2.60E-09	1.88E+00	1.12E-12	1.56E+01	4.51E-03	1.10E+00	2.10E-07	1.66E+01
	Std	2.38E-15	2.62E-10	3.30E-01	5.73E-13	1.08E-01	1.66E-03	3.35E-01	1.12E-06	3.02E-01
	p-value	-	**9.25E-12**$^+$	**9.25E-12**$^+$	**9.23E-12**$^+$	**9.25E-12**$^+$	**9.25E-12**$^+$	**9.21E-12**$^+$	**9.25E-12**$^+$	**9.25E-12**$^+$
F_4	Median	4.37E+11	7.26E+11	3.04E+11	2.73E+11	4.32E+11	1.45E+12	2.46E+13	1.78E+13	5.08E+12
	Mean	4.40E+11	7.25E+11	2.99E+11	2.74E+11	4.42E+11	1.70E+12	2.59E+13	1.71E+13	5.22E+12
	Std	1.10E+11	1.23E+11	7.16E+10	7.24E+10	8.05E+10	1.04E+12	8.14E+12	5.47E+12	1.89E+12
	p-value	-	**3.20E-09**$^+$	1.49E-06$^-$	9.83E-08$^-$	9.59E-01$^=$	**6.07E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_5	Median	1.19E+07	2.00E+06	3.29E+07	3.54E+07	9.35E+07	3.76E+08	2.50E+08	5.11E+08	1.52E+08
	Mean	1.22E+07	2.86E+06	3.17E+07	3.42E+07	9.25E+07	4.14E+08	2.69E+08	4.99E+08	1.55E+08
	Std	3.43E+06	1.79E+06	6.21E+06	6.69E+06	9.04E+06	1.38E+08	6.84E+07	1.07E+08	2.15E+07
	p-value	-	8.11E-11$^-$	**3.00E-11**$^+$	**3.00E-11**$^+$	**3.00E-11**$^+$	**3.00E-11**$^+$	**3.00E-11**$^+$	**3.00E-11**$^+$	**3.00E-11**$^+$
F_6	Median	4.00E-09	8.23E-07	2.15E+01	7.58E-09	3.66E+01	1.97E+07	4.71E+06	1.97E+07	1.64E+01
	Mean	5.20E-01	8.21E-07	2.08E+01	1.41E+05	3.78E+01	1.71E+07	5.00E+06	1.78E+07	1.63E+01
	Std	7.46E-01	2.68E-08	2.63E+00	3.67E+05	1.21E+01	5.20E+06	1.03E+06	4.37E+06	3.45E-01
	p-value	-	7.39E-02$^=$	**1.79E-11**$^+$	**1.45E-02**$^+$	**1.79E-11**$^+$	**1.79E-11**$^+$	**1.79E-11**$^+$	**1.79E-11**$^+$	**1.79E-11**$^+$
F_7	Median	1.22E+01	2.04E+04	4.06E+04	1.02E+01	3.47E+06	2.67E+06	6.57E+08	1.15E+08	9.20E+03
	Mean	7.19E+02	2.01E+04	6.49E+04	1.04E+01	3.47E+06	2.06E+08	8.14E+08	1.51E+08	1.41E+04
	Std	2.59E+03	3.86E+03	5.60E+04	4.02E+00	1.16E+05	4.31E+08	5.40E+08	1.45E+08	1.26E+04
	p-value	-	**4.50E-11**$^+$	**9.92E-11**$^+$	3.95E-01$^=$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.16E-10**$^+$
F_8	Median	2.34E+07	3.87E+07	7.43E+06	4.06E+06	2.02E+07	2.00E+07	9.06E+07	8.82E+07	1.62E+07
	Mean	2.34E+07	3.87E+07	7.81E+06	1.15E+07	2.03E+07	4.13E+07	8.56E+07	6.59E+07	2.75E+07
	Std	2.46E+05	6.81E+04	1.56E+06	2.04E+07	1.88E+06	3.84E+07	2.64E+07	3.40E+07	2.63E+07
	p-value	-	**3.02E-11**$^+$	3.02E-11$^-$	3.99E-04$^-$	2.44E-09$^-$	1.86E-01$^=$	**8.48E-09**$^+$	**6.77E-05**$^+$	**3.99E-04**$^+$
F_9	Median	4.33E+07	7.05E+07	3.22E+07	3.02E+07	2.08E+07	1.14E+08	4.35E+08	2.48E+08	5.52E+07
	Mean	4.36E+07	7.03E+07	3.30E+07	3.07E+07	2.08E+07	1.02E+08	4.40E+08	2.43E+08	5.59E+07
	Std	4.28E+06	5.73E+06	4.46E+06	3.19E+06	1.58E+06	3.30E+07	4.87E+07	2.16E+07	6.45E+06
	p-value	-	**3.02E-11**$^+$	3.50E-09$^-$	8.99E-11$^-$	3.02E-11$^-$	**3.34E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**7.38E-10**$^+$
F_{10}	Median	8.89E+02	9.59E+03	2.60E+03	1.33E+03	5.09E+03	5.14E+03	1.02E+04	3.97E+03	4.47E+03
	Mean	8.91E+02	9.60E+03	2.56E+03	1.33E+03	5.24E+03	5.09E+03	1.03E+04	4.24E+03	4.49E+03
	Std	3.66E+01	7.67E+01	2.17E+02	5.67E+01	4.26E+02	7.81E+02	3.13E+02	1.45E+03	1.29E+02
	p-value	-	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.00E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$
F_{11}	Median	2.75E+00	3.80E-08	2.30E+01	8.94E+00	1.68E+02	1.98E+02	2.59E+01	1.98E+02	1.02E+01
	Mean	5.80E+00	4.02E-08	2.32E+01	8.66E+00	1.68E+02	1.98E+02	2.59E+01	1.98E+02	1.02E+01
	Std	5.40E+00	5.12E-09	2.10E+00	3.25E+00	1.90E+00	2.12E+00	1.73E+00	1.12E+00	8.71E-01
	p-value	-	3.02E-11$^-$	**3.02E-11**$^+$	**7.96E-03**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**5.57E-03**$^+$
F_{12}	Median	1.24E+04	4.23E+05	1.31E+04	6.22E+04	2.83E+01	2.78E+04	9.69E+04	1.01E+05	2.58E+03
	Mean	1.25E+04	4.37E+05	1.75E+04	6.34E+04	2.39E+01	3.39E+04	9.55E+04	1.03E+05	2.84E+03
	Std	1.46E+03	6.22E+04	9.07E+03	1.00E+04	9.88E+00	1.19E+04	9.55E+03	1.57E+04	1.08E+03
	p-value	-	**3.02E-11**$^+$	2.12E-01$^=$	**3.02E-11**$^+$	3.02E-11$^-$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	3.02E-11$^-$
F_{13}	Median	7.28E+02	5.47E+02	8.48E+02	8.44E+02	1.03E+05	1.36E+03	4.59E+03	2.12E+03	5.06E+03
	Mean	7.35E+02	6.29E+02	9.59E+02	9.89E+02	1.05E+05	1.34E+03	5.96E+03	4.22E+03	6.27E+03
	Std	1.93E+02	2.32E+02	3.74E+02	4.52E+02	6.18E+04	1.72E+02	4.16E+03	4.70E+03	3.65E+03
	p-value	-	1.17E-02$^-$	**2.51E-02**$^+$	**1.22E-02**$^+$	**3.02E-11**$^+$	**7.39E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_{14}	Median	1.25E+08	2.52E+08	8.45E+07	1.68E+08	1.25E+07	3.42E+08	9.72E+08	5.71E+08	3.46E+08
	Mean	1.24E+08	2.49E+08	8.41E+07	1.70E+08	1.19E+07	3.06E+08	9.78E+08	5.70E+08	3.42E+08
	Std	7.38E+06	1.53E+07	6.31E+06	1.29E+07	1.62E+06	1.19E+08	7.52E+07	5.50E+07	2.42E+07
	p-value	-	**3.02E-11**$^+$	3.02E-11$^-$	**3.02E-11**$^+$	3.02E-11$^-$	**4.50E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_{15}	Median	8.40E+02	1.01E+04	1.12E+04	2.68E+03	5.48E+03	1.04E+04	1.24E+04	8.67E+03	5.86E+03
	Mean	8.33E+02	1.01E+04	1.12E+04	2.66E+03	5.54E+03	1.08E+04	1.23E+04	8.90E+03	5.86E+03
	Std	4.31E+01	5.23E+01	8.65E+01	1.50E+02	3.46E+02	1.35E+03	8.24E+02	2.07E+03	1.05E+02
	p-value	-	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$	**3.01E-11**$^+$
F_{16}	Median	3.70E+00	5.75E-08	2.21E+01	6.11E+01	3.18E+02	3.97E+02	6.92E+01	3.96E+02	7.50E-13
	Mean	4.25E+00	5.89E-08	2.51E+01	5.94E+01	3.17E+02	3.96E+02	6.96E+01	3.81E+02	7.53E-13
	Std	2.41E+00	5.61E-09	1.16E+01	1.37E+01	2.04E+00	5.73E-01	6.43E+00	5.76E+01	6.25E-14
	p-value	-	3.02E-11$^-$	**3.69E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	3.00E-11$^-$
F_{17}	Median	9.02E+04	2.22E+06	8.92E+04	8.66E+04	4.75E+01	8.99E+04	3.11E+05	3.47E+05	4.02E+04
	Mean	9.05E+04	2.20E+06	9.00E+04	1.07E+05	4.29E+01	1.25E+05	3.11E+05	3.49E+05	4.03E+04
	Std	3.53E+03	1.55E+05	1.58E+04	6.75E+04	1.15E+01	5.25E+04	2.24E+04	3.11E+04	2.29E+03
	p-value	-	**3.02E-11**$^+$	5.49E-01$^=$	7.48E-02$^=$	3.02E-11$^-$	9.23E-01$^=$	**3.02E-11**$^+$	**3.02E-11**$^+$	3.02E-11$^-$
F_{18}	Median	2.29E+03	1.76E+03	2.66E+03	2.41E+03	2.50E+04	3.10E+03	3.54E+04	1.59E+04	1.47E+10
	Mean	2.55E+03	1.73E+03	2.77E+03	2.53E+03	2.43E+04	3.07E+03	3.83E+04	1.81E+04	1.47E+10
	Std	8.32E+02	5.22E+02	8.33E+02	6.86E+02	1.10E+04	2.45E+02	1.53E+04	9.48E+03	2.03E+09
	p-value	-	6.77E-05$^-$	1.67E-01$^=$	8.77E-01$^=$	**3.02E-11**$^+$	**2.13E-04**$^+$	**3.02E-11**$^+$	**8.15E-11**$^+$	**3.02E-11**$^+$
F_{19}	Median	1.79E+06	1.00E+07	5.11E+06	5.53E+05	2.03E+06	1.51E+06	1.14E+06	2.04E+06	1.75E+06
	Mean	1.80E+06	1.01E+07	5.10E+06	5.46E+05	2.02E+06	1.52E+06	1.14E+06	2.04E+06	1.75E+06
	Std	9.96E+04	5.64E+05	7.05E+05	2.80E+04	1.41E+05	7.10E+04	6.23E+04	1.42E+05	1.10E+05
	p-value	-	**3.02E-11**$^+$	**3.02E-11**$^+$	3.02E-11$^-$	**2.60E-08**$^+$	4.50E-11$^-$	3.02E-11$^-$	**7.09E-08**$^+$	1.02E-01$^=$
F_{20}	Median	1.85E+03	9.85E+02	1.79E+03	1.36E+03	9.82E+02	2.10E+03	4.34E+03	2.27E+03	6.53E+10
	Mean	1.88E+03	1.05E+03	1.85E+03	1.39E+03	9.87E+02	2.11E+03	4.58E+03	2.30E+03	6.41E+10
	Std	1.90E+02	1.49E+02	2.59E+02	1.47E+02	1.40E+01	1.79E+02	8.25E+02	2.26E+02	6.97E+09
	p-value	-	4.50E-11$^-$	1.91E-01$^=$	9.92E-11$^-$	3.02E-11$^-$	**1.64E-05**$^+$	**3.02E-11**$^+$	**1.56E-08**$^+$	**3.02E-11**$^+$
w//t		-	13/6/1	12/4/4	11/6/3	13/6/1	16/2/2	19/1/0	19/1/0	16/3/1

TABLE II
COMPARISON RESULTS OF THE COMPARED ALGORITHMS ON 1000-*D* CEC'2013 FUNCTIONS WITH 3×10^6 FITNESS EVALUATIONS.

Function	Quality	DLLSO	CSO	SL-PSO	MA-SW-Chains	DMS-L-PSO	CCPSO2	DECC-G	MLCC	DECC-DG
F_1	Median	3.86E-22	7.78E-12	1.04E-17	7.90E-21	1.97E+09	2.79E+01	2.06E-06	9.07E-14	6.03E+02
	Mean	3.99E-22	7.71E-12	1.09E-17	1.19E-20	1.98E+09	4.11E+01	3.14E-06	8.60E-10	6.42E+03
	Std	1.32E-22	1.31E-12	2.50E-18	1.11E-20	1.27E+08	3.14E+01	4.27E-06	4.38E-09	1.81E+04
	p-value	-	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_2	Median	1.14E+03	8.55E+03	2.13E+03	6.85E+02	8.61E+03	3.48E+01	1.30E+03	3.57E+00	1.28E+04
	Mean	1.14E+03	8.55E+03	2.13E+03	6.97E+02	8.65E+03	3.50E+01	1.31E+03	3.82E+00	1.27E+04
	Std	5.78E+01	2.65E+02	1.36E+02	5.51E+01	4.88E+02	4.85E+00	3.63E+01	1.73E+00	7.20E+02
	p-value	-	**3.02E-11**$^+$	**3.02E-11**$^+$	3.02E-11$^-$	**3.02E-11**$^+$	3.02E-11$^-$	**7.38E-11**$^+$	3.02E-11$^-$	**3.02E-11**$^+$
F_3	Median	2.16E+01	2.16E+01	2.16E+01	2.03E+01	2.08E+01	2.00E+01	2.02E+01	2.00E+01	2.14E+01
	Mean	2.16E+01	2.16E+01	2.16E+01	2.03E+01	2.08E+01	2.00E+01	2.02E+01	2.00E+01	2.14E+01
	Std	4.07E-03	6.15E-03	1.45E-02	4.36E-02	1.66E-01	1.25E-04	6.18E-03	2.76E-04	1.45E-02
	p-value	-	3.67E-01$^=$	**3.08E-08**$^+$	3.02E-11$^-$	3.02E-11$^-$	2.19E-11$^-$	3.02E-11$^-$	3.02E-11$^-$	3.02E-11$^-$
F_4	Median	6.34E+09	1.28E+10	4.54E+09	5.19E+09	2.97E+11	3.20E+10	2.00E+11	1.99E+11	7.33E+10
	Mean	6.68E+09	1.32E+10	4.35E+09	5.13E+09	2.93E+11	3.49E+10	2.35E+11	2.34E+11	7.70E+10
	Std	1.68E+09	2.54E+09	9.48E+08	1.33E+09	7.25E+10	2.17E+10	1.22E+11	1.26E+11	2.82E+10
	p-value	-	**1.09E-10**$^+$	2.02E-08$^-$	5.56E-04$^-$	**3.02E-11**$^+$	**1.01E-08**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_5	Median	6.60E+05	6.04E+05	8.23E+05	1.74E+06	3.92E+06	1.30E+07	8.44E+06	1.17E+07	5.81E+06
	Mean	7.00E+05	5.91E+05	8.41E+05	1.76E+06	3.95E+06	1.40E+07	8.26E+06	1.27E+07	5.78E+06
	Std	1.28E+05	1.07E+05	1.75E+05	3.26E+05	5.82E+05	4.81E+06	1.14E+06	3.46E+06	3.83E+05
	p-value	-	1.95E-03$^-$	**4.46E-04**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_6	Median	1.06E+06	1.06E+06	1.06E+06	1.05E+06	9.98E+05	1.05E+06	1.06E+06	1.05E+06	1.06E+06
	Mean	1.06E+06	1.06E+06	1.06E+06	1.05E+06	1.00E+06	1.05E+06	1.06E+06	1.05E+06	1.06E+06
	Std	8.28E+02	1.10E+03	1.48E+03	7.00E+03	5.20E+03	5.24E+03	1.84E+03	4.13E+03	1.07E+03
	p-value	-	1.10E-01$^=$	3.01E-01$^=$	3.01E-11$^-$	3.01E-11$^-$	3.01E-11$^-$	**1.32E-04**$^+$	3.01E-11$^-$	**9.26E-04**$^+$
F_7	Median	1.33E+06	5.26E+06	1.40E+06	2.98E+06	1.22E+09	1.29E+08	1.04E+09	1.15E+09	4.25E+08
	Mean	1.60E+06	5.88E+06	1.63E+06	2.91E+06	1.37E+09	4.15E+08	1.04E+09	1.43E+09	4.78E+08
	Std	8.38E+05	2.58E+06	7.05E+05	1.30E+06	7.64E+08	9.38E+08	4.48E+08	1.07E+09	1.92E+08
	p-value	-	**8.99E-11**$^+$	**5.11E-01**$^+$	**3.83E-05**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_8	Median	1.16E+14	2.85E+14	9.97E+13	1.26E+14	1.68E+14	8.16E+14	7.90E+15	8.18E+15	2.89E+15
	Mean	1.20E+14	2.60E+14	1.03E+14	1.28E+14	2.79E+14	1.18E+15	7.50E+15	9.59E+15	3.57E+15
	Std	3.35E+13	5.87E+13	3.62E+13	3.44E+13	5.09E+14	9.99E+14	3.18E+15	6.18E+15	1.85E+15
	p-value	-	**2.37E-10**$^+$	3.78E-02$^-$	4.04E-01$^=$	**3.83E-06**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_9	Median	1.26E+08	5.79E+07	7.94E+07	1.07E+08	3.50E+08	3.63E+09	5.86E+08	8.85E+08	4.95E+08
	Mean	1.30E+08	6.06E+07	8.25E+07	1.09E+08	3.60E+08	3.76E+09	5.96E+08	9.55E+08	4.90E+08
	Std	3.97E+07	1.60E+07	2.03E+07	1.96E+07	4.61E+07	1.02E+09	9.76E+07	2.92E+08	3.18E+07
	p-value	-	3.16E-10$^-$	7.60E-07$^-$	2.32E-02$^-$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_{10}	Median	9.40E+07	9.40E+07	9.36E+07	9.34E+07	9.11E+07	9.29E+07	9.30E+07	9.27E+07	9.45E+07
	Mean	9.40E+07	9.40E+07	9.25E+07	9.34E+07	9.17E+07	9.30E+07	9.29E+07	9.27E+07	9.45E+07
	Std	2.11E+05	1.51E+05	1.67E+06	3.55E+05	1.06E+06	7.01E+05	6.16E+05	6.07E+05	2.46E+05
	p-value	-	3.40E-01$^=$	3.77E-04$^-$	1.61E-10$^-$	3.02E-11$^-$	3.01E-07$^-$	3.82E-10$^-$	7.39E-11$^-$	**2.39E-08**$^+$
F_{11}	Median	9.29E+11	9.35E+11	9.35E+11	4.79E+08	9.44E+10	9.38E+11	1.26E+11	1.90E+11	3.81E+10
	Mean	9.30E+11	9.30E+11	9.33E+11	9.59E+08	1.05E+11	9.37E+11	1.28E+11	2.28E+11	4.83E+10
	Std	9.50E+09	1.03E+10	1.46E+10	1.68E+09	7.53E+10	1.53E+10	7.15E+10	1.53E+11	4.33E+10
	p-value	-	8.88E-01$^=$	8.30E-01$^=$	3.02E-11$^-$	3.02E-11$^-$	**2.71E-02**$^+$	3.02E-11$^-$	3.02E-11$^-$	3.02E-11$^-$
F_{12}	Median	1.79E+03	1.04E+03	1.75E+03	1.34E+03	5.22E+04	2.10E+03	4.19E+03	2.36E+03	1.68E+11
	Mean	1.79E+03	1.07E+03	1.78E+03	1.33E+03	6.99E+04	2.10E+03	4.35E+03	2.49E+03	1.71E+11
	Std	1.39E+02	7.78E+01	1.74E+02	1.00E+02	5.52E+04	1.78E+02	7.83E+02	7.51E+02	2.24E+10
	p-value	-	3.02E-11$^-$	5.30E-01$^=$	5.49E-11$^-$	**3.02E-11**$^+$	**2.83E-08**$^+$	**3.02E-11**$^+$	**6.70E-11**$^+$	**3.02E-11**$^+$
F_{13}	Median	2.70E+08	6.28E+08	4.59E+08	9.72E+08	1.32E+10	3.21E+09	8.67E+09	9.94E+09	2.08E+10
	Mean	3.35E+08	6.67E+08	4.65E+08	1.04E+09	1.34E+10	4.02E+09	9.35E+09	1.06E+10	2.05E+10
	Std	1.71E+08	2.45E+08	2.35E+08	3.28E+08	6.58E+09	2.31E+09	2.78E+09	3.73E+09	5.53E+09
	p-value	-	**1.73E-07**$^+$	**2.15E-02**$^+$	**9.92E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_{14}	Median	1.03E+08	3.15E+09	1.50E+08	5.11E+09	2.21E+11	5.98E+10	1.28E+11	2.06E+11	1.56E+10
	Mean	1.72E+08	3.62E+09	3.28E+08	6.53E+09	2.46E+11	9.10E+10	1.42E+11	2.21E+11	1.92E+10
	Std	1.38E+08	1.44E+09	5.17E+08	5.70E+09	1.26E+11	8.53E+10	5.86E+10	8.54E+10	1.44E+10
	p-value	-	**3.02E-11**$^+$	1.15E-01$^=$	**3.69E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
F_{15}	Median	4.45E+06	7.72E+07	5.88E+07	7.95E+06	1.54E+07	2.72E+06	1.13E+07	1.57E+07	9.52E+06
	Mean	4.48E+06	7.87E+07	5.86E+07	8.48E+06	1.57E+07	4.75E+06	1.16E+07	1.61E+07	9.90E+06
	Std	3.32E+05	6.50E+06	6.11E+06	2.25E+06	3.45E+06	5.07E+06	1.26E+06	1.90E+06	2.30E+06
	p-value	-	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**2.38E-03**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$	**3.02E-11**$^+$
w/l/t		-	8/3/4	7/4/4	6/8/1	11/4/0	11/4/0	12/3/0	10/5/0	13/2/0

First, for unimodal functions, exploitation should be put a little more emphasis, so that fast convergence can be achieved. From Fig. 2(a) and Fig. 2(c), we can see that on the unimodal functions $F_1$and F_7, the exploitation is properly biased, so that LLSO converges much faster than GPSO and CSO with better solutions simultaneously. Specifically, on F_1 (see Fig. 2(a)), the exploitation is biased much more obviously in LLSO. GPSO maintains the highest diversity but obtains the worst performance because of the stagnation of the swarm. CSO perserves higher diversity but slower convergence than LLSO, because the exploitaiton is less empahsized. On F_7 (see Fig. 2(c)), exploitation is appropriately biased without serious loss of exploration in LLSO, resulting in its good performance. However, the exploitation is overemphasized in GPSO and thus the exploration is seriously ignored, leading to its inferior performance. For CSO, the exploitation is less biased and thus slower convergence is obtained than LLSO.

Second, when it arrives at multimodal functions, exploration should be dynamically and properly biased without serious loss of exploitation, so that premature convergence and stagnation can be avoided. From Figs. 2(b) and 2(d), we can find that on multimodal functions F_5 and F_{10}, LLSO still achieves better peformance than GPSO and CSO with respect to both convergence speed and solution quality. This is because LLSO can compromise exploration and exploitation better than GPSO and CSO. Specifically in GPSO, the exploitation is overbiased and thus the exploration is seriously lost on both functions, leading to its poor performance. For CSO, the exploration is overemphasized and thus its exploitation is very poor, resulting in its poor performance in refining the solutions.

Overall, we can see that LLSO can preserve good exploration and exploitation abilities and can particulaly compromise these two well to search the space during the evolution. Such a good ability benefits from the proposed LL strategy, which can offer two kinds of compromises between exploration and exploitation: the particle-level compromise and the swarm-level compromise as stated in Section III.E.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

B. Parameter Settings

In LLSO, only two extra parameters are introduced: the number of levels *NL* and the control parameter ϕ. Since we have proposed a dynamic selection strategy for *NL* in Section III.G, the fine-tuning of the sensitive *NL* can be saved by setting the pool $\boldsymbol{S}$ with a wide range. In the preliminary experiments, we find DLLSO is not so sensitive to $\boldsymbol{S}$, if we keep $\boldsymbol{S}$ in a wide range. In this paper, we set $\boldsymbol{S}$={4,6,8,10,20,50}.

Then, we turn to the setting of the control parameter ϕ and the swarm size *NP*, the common parameter in all EAs [6, 21, 23, 56], which is hard to set, owing to its dependency on the complexity of problems. Thus, to see the effect of ϕ and *NP* on DLLSO, we conduct experiments on DLLSO with *NP* varying from 200 to 600 and ϕ varying from 0.1 to 0.6.

Table SIII in the *Supplemental Material* displays the experimental results of DLLSO with different combinations of ϕ and *NP* on six 1000-*D* benchmark functions from the CEC'2010 set: fully separable and unimodal function F_1, fully separable and multimodal function F_3, partially separable and unimodal function F_7, partially separable and multimodal function F_8, partially separable and unimodal function F_{12}, and partially separable and unimodal function F_{17}. These functions are selected, because we want to investigate the influence of parameters on almost all kinds of problems: fully separable, partially separable, unimodal, and multimodal.

From this table, we can see that: 1) the smaller the swarm size is, the larger value ϕ has. This is because a small swarm size cannot offer high diversity for the swarm, thus a large ϕ is needed to promote the diversity by enhancing the influence of the second exemplar, which owns more potential in exploration than the first one in Eq. (6). 2) For large swarm sizes, the proper ϕ seems to consistently stay at 0.4 as indicated by the results of DLLSO with *NP* within [400,600]. 3) When *NP* is within [400, 600], it seems that ϕ makes no significant difference on DLLSO when it is within [0.1, 0.5]. However, when ϕ is within [0.1, 0.5], it seems that *NP* has great influence on DLLSO, especially for F_3 and F_7.

In conclusion, *NP*=500 and ϕ=0.4 is adopted for DLLSO on 1000-*D* problems, which also makes it fair to compare DLLSO with CSO that adopts the same setting of *NP* [23].

C. Effect of Dynamic Level Numbers

To investigate the effect of the dynamic selection of *NL* on DLLSO, we conduct comparison experiments on two versions of the proposed optimizer: LLSO with a fixed *NL* and LLSO with a dynamic *NL*, namely DLLSO. The former version of LLSO is represented as "LLSO-*NL*", such as LLSO with 4 levels can be denoted as "LLSO-4".

Fig. S1 in the *Supplemental Material* shows the comparison results between the two versions of LLSO on eight 1000-*D* benchmark functions from the CEC'2010 set including the six functions used in the last subsection (adding two extra functions F_{10} and F_{15}). In this experiment, *NP*=500 and ϕ=0.4 is adopted and the maximum number of function evaluations varies from 5×10^5 to 5×10^6. For "LLSO-*NL*", the fixed numbers of levels are set to be the members in $\boldsymbol{S}$.

From this figure, firstly, we can find that on some functions, such as F_{10}, F_{12} and F_{17}, LLSO is not sensitive to *NL*, while on some functions, such as F_3, F_7, F_8 and F_{15}, LLSO is very sensitive to *NL*. Secondly, the optimal *NL* is different for different problems, such as the optimal *NL* is 8 for F_7, while that number is 50 for F_{15}. Thirdly, comparing these two versions, we find that DLLSO can make a good compromise to obtain competitive solutions on almost all the eight problems and even on some functions, such as F_{10} and F_{15}, DLLSO can obtain better solutions than the LLSO with the optimal *NL*.

All in all, we can find that the dynamic selection strategy for *NL* is promising for the proposed optimizer.

D. Comparisons with State-of-the-Art Methods

Subsequently, to comprehensively verify the efficiency and effectiveness of DLLSO, we compare it with various state-of-the-art algorithms dealing with large scale optimization. Specifically, four popular algorithms, namely three PSO variants (CSO [23], SL-PSO[1] [21] and DMS-L-PSO[2] [39]), and a memetic algorithm named MA-SW-Chains [3] [48], concentrating on the second aspect in handling large scale optimization (Section II.B) , and four CCEAs, namely CCPSO2 [33], DECC-DG [4] [20], DECC-G [35] and MLCC [5] [36], focusing on the first aspect in large scale optimization (Section II.A), are selected to make comparisons. For fairness, the key parameters in each algorithm are set as recommended in the corresponding papers. We conduct the comparison experiments on both the CEC'2010 benchmark set [27] and the CEC'2013 benchmark set [28].

Table I and Table II respectively show the comparison results among different algorithms on the two benchmark sets with 1000 dimensions. The highlighted $\boldsymbol{p}$ values mean that DLLSO is significantly better than the corresponding algorithms. Additionally, the symbols, "+", "-", and "=", above the *p* values represent that DLLSO is significantly better than, significantly worse than, and equivalent to the compared algorithms on the associated functions. Furthermore, *w/l/t* in the last row represents that DLLSO wins on *w* functions, loses on *l* functions and ties on *t* functions in total in the competitions with the counterpart methods.

As for the CEC'2010 set, from Table I, we can see that DLLSO outperforms the compared algorithms on most of the 20 functions. In details, compared with CSO, SL-PSO, MA-SW-Chains and DMS-L-PSO, DLLSO shows its great superiority on 13, 12, 11, and 13 functions, respectively. Compared with these algorithms, DLLSO only loses the competition on 6, 4, 6, and 6 functions respectively. In comparison with the four CCEAs (CCPSO2, DECC-G, MLCC, and DECC-DG), DLLSO defeats them on 16, 19, 19 and 16 functions respectively. Besides, DLLSO only loses on 2, 1, 1, and 3 functions respectively competing with these algorithms.

When it arrives at the CEC'2013 set where the functions are more difficult to optimize than those in the former set, DLLSO consistently shows its dominance according to Table II.

[1] The codes of CSO and SL-PSO can be downloaded from http://www.surrey.ac.uk/cs/research/nice/people/yaochu_jin/.

[2] The code of DMS-L-PSO can be downloaded from http://www.ntu.edu.sg/home/epnsugan/.

[3] The code of MA-SW-Chains can be downloaded from http://sci2s.ugr.es/EAMHCO#Complementary.

[4] The codes of CCPSO2 and DECC-DG can be downloaded from https://titan.csit.rmit.edu.au/~e46507/publications.php.

[5] The codes of DECC-G and MLCC can be downloaded from http://staff.ustc.edu.cn/~ketang/codes/.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

Compared with DMS-L-PSO and the four CCEAs, DLLSO shows its significant superiority on at least 10 functions. In comparison with CSO and SL-PSO, DLLSO defeats them down on 8 and 7 functions respectively and only loses the competitions on 3 and 4 functions respectively. Unfortunately, on this set, DLLSO is slightly worse than MA-SW-Chains. However, compared with this algorithm, DLLSO is easier to understand and simpler to implement, due to its maintenance of the framework of the classical PSO, which leads to its superior performance to MA-SW-Chains in computational efficiency that will be verified in the following sections.

Further, we also conduct convergence behavior comparison between DLLSO and the compared algorithms on the two benchmark sets to testify the superiority of DLLSO w.r.t. convergece speed. Fig. S2 and Fig. S3 in the *Supplemental Material* present the comparison results on the CEC'2010 and CEC'2013 benchmark sets respectively.

On the CEC'2010 benchmark set, from Fig. S2, we can observe that: 1) DLLSO converges faster with better solutions than all 8 compared methods on 4 (F_3, F_6, F_{10} and F_{15}) functions. 2) DLLSO achieves faster convergence with higher quality solutions than 7 compared methods (except for only one compared algorithm out of the 8 compared methods) on 6 functions (F_1, F_5, F_7, F_{11}, F_{12}, and F_{14}). 3) Concretely, DLLSO can apparently defeat CSO, SL-PSO, MA-SW-Chains, DMS-L-PSO, CCPSO2, DECC-G, MLCC and DECC-DG with both faster convergence and better solutions on 14, 12, 10, 20, 15, 17, 16, and 14 functions, respectively.

Similarly, on the CEC'2013 benchmark set, from Fig. S3, we can obtain that: 1) DLLSO achieves great superiority to all 8 compared algorithms in both convergence and solution quality on 6 functions (F_4, F_7, F_8, F_{13}-F_{15}). 2) Besides, on F_1 and F_5, DLLSO achieves both better solutions and faster convergence than 7 compared algorithms. 3) DLLSO converges faster with higher solution quality than CSO, SL-PSO, MA-SW-Chains, DMS-L-PSO, CCPSO2, DECC-G, MLCC and DECC-DG on 9, 9, 8, 11, 9, 9, 8, and 12 functions, respectively.

In conclusion, we can see that compared with these state-of-the-art large scale algorithms, DLLSO can achieve competitive or even better performance in both solution quality and convergence speed. The superiority of DLLSO can be attributed to the proposed LL strategy and the proposed exemplar selection strategy. LL groups particles into different levels and treats particles in different levels differently. The exemplar selection strategy allows particles in different levels to learn from various superior particles from different higher levels. From the two selected superior exemplars, one particle could improve its potential in exploitation by learning from the better one, and at the same time consolidate its potential in exploration by learning from the inferior one. In this way, each updated particle may compromise exploration and exploitation in the evolution.

Besides, the cooperation between these two strategies makes particles in different levels learn from different numbers of exemplars. That is, particles in lower levels have more superior particles and a wider range to learn, which is beneficial for exploration, while particles in higher levels have fewer superior particles and a narrower range to learn, which is profitable for exploitation. In this manner, the whole swarm can make a compromise in exploring and exploiting the search space via letting particles in higher levels concentrate on exploiting while letting particles in lower levels focus on exploring.

In short, these two kinds of compromises in exploration and exploitation make DLLSO achieve good performance.

E. Scalability Comparisons with State-of-the-Art Methods

TABLE III
THE PARAMETER SETTINGS OF DLLSO IN DEALING WITH PROBLEMS WITH DIFFERENT DIMENSION SIZES.

Parameters \ *D*	200	500	800	2000
$\boldsymbol{S}$	{4,6,8,10,20,50}			
NP	300	300	500	1000
ϕ	0.5	0.5	0.4	0.4

The above comparison experiments have exhibited the superiority of DLLSO to several state-of-the-art methods in dealing with 1000-*D* problems. To further substantiate the scalability of DLLSO to solve higher dimensional problems, we perform experiments on the CEC'2010 problems with dimensionality increasing from 200 to 2000.

In this series of experiments, the parameters of DLLSO are set as shown in Table III. As for the compared algorithms, the parameters are set as recommended in the corresponding papers. For fairness, the maximum number of fitness evaluations is set as $3000\times D$ when conducting experiments on problems with different dimension sizes. In addition, due to the page limit, we attach all the comparison results to the *Supplemental Material*.

1) Comparsion Results on 200-D Problems

Table SIV presents the comparison results on the CEC'2010 problems with 200 dimensions. From this table, we can see that DLLSO displays its great potential and ability in dealing with 200-*D* problems. Specifically, DLLSO can achieve the global optimum of F_1 in each run and is much superior to CSO, SL-PSO, and the four CCEAs (CCPSO2, DECC-G, MLCC and DECC-DG) on at least 16 functions. Besides, DLLSO also wins the competition with MA-SW-Chains on 12 functions. Compared with DMS-L-PSO, DLLSO is competitive and comparable to this algorithm by defeating it on 9 functions.

2) Comparison Results on 500-D Problems

Table SV shows the comparison results among different algorithms on 500-*D* problems. Observing this table, we can find that DLLSO respectively dominates the 8 compared algorithms on at least 11 functions. Compared with CSO, DLLSO outperforms it on 13 functions and only loses the competition on 3 functions. In comparison to MA-SW-Chains and DMS-L-PSO, DLLSO performs better than them both on 11functions. In particular, DLLSO obtains the global optimum of F_1 in each run as well and is much better than SL-PSO and the four CCEAs on at least 15 functions.

3) Comparison Results on 800-D Problems

Table SVI displays the comparison results on 800-*D* problems. From this table, we can observe that DLLSO is respectively superior to the 8 compared algorithms on at least 12 functions. Particularly, DLLSO dominates CSO, SL-PSO, and DMS-L-PSO on 13 functions respectively and is especially better than the four CCEAs on at least 17 functions.

4) Comparison Results on 2000-D Problems

Table SVII presents the comparison results among all the compared algorithms on 2000-*D* problems, which are particularly harder to optimize than the aforementioned

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

problems. From this table, we can see that DLLSO is still much better than the compared algorithms in dealing with such complicated problems. Specifically, DLLSO respectively wins the competition with the 8 compared methods on at least 13 functions. In particular, DLLSO is significantly superior to SL-PSO, and the four CCEAs on at least 15 functions.

5) Overall Comparisons

From the above comparison results, we can see that DLLSO has a good scalability in tackling problems with different dimension sizes. Particularly, as the dimensionality increases, the number of functions on which DLLSO can respectively dominate the 8 compared algorithms increases as well (for 200-*D*, 500-*D*, 800-*D*, 1000-*D* and 2000-*D* problems, this number is 9, 11, 12, 11 and 13 respectively.).

To have a better view of the comparison results, we plot the changes of the averaged fitness value of each algorithm on each function with the dimensionality increasing from 200 to 2000. The result is shown in Fig. S4 in the *Supplemental Material*.

From this figure, we can find that as expected, on most functions, the performance of all algorithms degrades with the dimensionality increasing, which results from the exponentially increased search space. However, we find that on F_1, F_3, and F_{10}, DLLSO always achieves the best performance as the dimensionality increases in comparison with other algorithms as shown in Figs. S3(a), S3(c) and S3(j). Besides, on F_5, F_6, and F_7, with the dimensionality increasing, the ability of DLLSO does not degrade but instead is improved via the proper parameter settings, which can be clearly seen from Figs. S3(e), S3(f) and S3(g).

Overall, we can conclude that DLLSO preserves good scalability to solve higher dimensional problems. Such superior scalability of DLLSO could be ascribed to the following aspects: 1) First, in LLSO, particles are divided into different levels and are treated differently. 2) Second, the proposed exemplar selection strategy affords two different superior exemplars for each particle to learn, so that the potential in exploitation of one particle may be promoted via learning from the better one and the potential in exploration may be enhanced by learning from the inferior one. In addition, particles in different levels have different numbers of superior particles to learn from, resulting in that particles in lower levels have a wider range to learn, which is beneficial for exploration, and particles in higher levels have a narrower range to learn, which is beneficial for exploitation. Thus, LLSO can compromise exploration and exploitation to search the space from both the particle level and the swarm level, which benefits for achieving competitive performance with the state-of-the-art methods.

F. Time Comparisons with State-of-the-Art Methods

The above experiments have demonstrated the superiority of DLLSO to other algorithms with respect to solution quality. To further validate the competitive efficiency of DLLSO in tackling large scale optimization, we conduct computational cost comparison between DLLSO and three compared algorithms, namely CSO, SL-PSO, and MA-SW-Chains. These algorithms are selected because on the one hand, they were all implemented with C codes; on the other hand, they all contribute to the second aspect in handling large scale optimization as stated in Section II.B. By this means, fair comparison can be obtained.

In the experiments, we record the computing time of each compared algorithm on the CEC'2010 benchmark functions with the dimensionality increasing from 200 to 2000. Table SVIII and Fig. S5 in the *Supplemental Material* present the time comparison results among the four algorithms on each function with different dimension sizes.

From Table SVIII and Fig. S5, we can see that both DLLSO and CSO are much more efficient than the other two algorithms (SL-PSO, and MA-SW-Chains). In details, the computational cost of MA-SW-Chains is the highest. This is because MA-SW-Chains employs many complicated local search methods. Compared with DLLSO and CSO, SL-PSO needs much more time, because it needs to compute the mean position of the swarm, to sort the swarm and to compute the learning probability for each particle, the combination of which leads to its higher computational cost than CSO and DLLSO.

Compared with CSO, we find that DLLSO takes nearly the same time as CSO. However, it is interesting to find that when dealing with low-dimensional problems, CSO is slightly more efficient than DLLSO. Nevertheless, with the dimensionality increasing, the difference between the computational cost of DLLSO and CSO becomes less and less, and even when it comes to 2000 dimensions, DLLSO is a bit more efficient than CSO. This is because CSO needs to compute the mean position of the whole swarm, which takes $O(NP \times D)$ at each generation. Thus, as the dimensionality increases, except for the computation time of fitness functions, the extra computational cost of CSO increases faster than that of DLLSO.

Together, we can see that with respect to the computational cost, DLLSO is also very competitive or even superior to state-of-the-art algorithms, due to its maintenance of the classical PSO framework, which is very easy to understand and simple to implement.

To summarize, we can conclude that the proposed DLLSO is competitive, effective and efficient in dealing with large scale optimization in both solution quality and computational cost.

VI. Conclusion and Discussion

In this paper, we have proposed a level-based learning strategy and an exemplar selection strategy, the combination of which leads to a new optimizer named level-based learning swarm optimizer (LLSO). Besides, to deal with the challenge that the optimal number of levels is problem-dependent, we further added a dynamic selection strategy for the number of levels, leading to DLLSO, a dynamic version of LLSO. Various experiments have been conducted to demonstrate the efficiency and effectiveness of DLLSO in tackling large scale optimization with respect to solution quality, convergence speed, scalability and computational cost.

Though DLLSO shows good performance in coping with large scale optimization, the obtained solutions to some functions are still far from the global optima, which is the common issue for the state-of-the-art algorithms as well, as seen in Table I and Table II. Therefore, how to further improve DLLSO to obtain solutions as near the global optima as possible is the first direction for future investigation. In addition, since dividing the whole swarm into levels is only associated with the population, whether the proposed level-based learning strategy and the exemplar selection

strategy are promising for other population-based EAs, such as DE, is another direction for future investigation.

REFERENCES

[1] L.-Y. Chuang, C.-H. Yang, J.-H. Tsai, and C.-H. Yang, "Operon Prediction Using Chaos Embedded Particle Swarm Optimization," *IEEE/ACM Trans. Comput. Biol. Bioinf.* , vol. 10, no. 5, pp. 1299-1309, 2013.

[2] 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, 2013.

[3] X. Wen, W. N. Chen, Y. Lin, T. Gu, H. Zhang, Y. Li, Y. Yin, and J. Zhang, "A Maximal Clique Based Multiobjective Evolutionary Algorithm for Overlapping Community Detection," *IEEE Trans. Evol. Comput.,* vol. 21, no. 3, pp. 363-377, 2017.

[4] Y. H. Jia, W. N. Chen, T. Gu, H. Zhang, H. Yuan, Y. Lin, W. J. Yu, and J. Zhang, "A Dynamic Logistic Dispatching System With Set-Based Particle Swarm Optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, in press, 2017.

[5] R. C. Eberhart, and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. Int. Symp. MHS*, 1995, pp. 39-43.

[6] J. Kennedy, and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw*, 1995, pp. 1942-1948 vol.4.

[7] J. Kennedy, and R. Mendes, "Population structure and particle swarm performance," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 1671-1676.

[8] J. Kennedy, "Small worlds and mega-minds: effects of neighborhood topology on particle swarm performance," in *Proc. IEEE Congr. Evol. Comput.*, 1999, pp. 1931-1938.

[9] Y. Shi, and R. Eberhart, "A modified particle swarm optimizer," in *Proc. IEEE Congr. Evol. Comput.*, 1998, pp. 69-73.

[10] K. E. Parsopoulos, and M. N. Vrahatis, "On the computation of all global minimizers through particle swarm optimization," *IEEE Trans. Evol. Comput.,* vol. 8, no. 3, pp. 211-224, 2004.

[11] 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, 2006.

[12] M. A. M. d. Oca, T. Stutzle, K. V. d. Enden, and M. Dorigo, "Incremental Social Learning in Particle Swarms," *IEEE Trans. Syst., Man, Cybern. B, Cybern.,* vol. 41, no. 2, pp. 368-384, 2011.

[13] 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, 2011.

[14] 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, 2014.

[15] J. Li, J. Zhang, C. Jiang, and M. Zhou, "Composite Particle Swarm Optimizer With Historical Memory for Function Optimization," *IEEE Trans. Cybern.,* vol. 45, no. 10, pp. 2350 - 2363, 2015.

[16] Q. Qin, S. Cheng, Q. Zhang, L. Li, and Y. Shi, "Particle Swarm Optimization With Interswarm Interactive Learning Strategy," *IEEE Trans. Cybern.,* vol. 46, no. 10, pp. 2238-2251, 2016.

[17] W.-N. Chen, J. Zhang, Y. Lin, N. Chen, Z.-H. Zhan, H. S.-H. Chung, Y. Li, and Y.-h. Shi, "Particle Swarm Optimization With an Aging Leader and Challengers," *IEEE Trans. Evol. Comput.,* vol. 17, no. 2, pp. 241-258, 2013.

[18] N. Lynn, and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm Evol. Comput.,* vol. 24, pp. 11-24, 2015.

[19] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up fast evolutionary programming with cooperative coevolution," in *Proc. IEEE Congr. Evol. Comput.*, 2001, pp. 1101-1108.

[20] 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, 2014.

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

[22] S. Mahdavi, M. E. Shiri, and S. Rahnamayan, "Metaheuristics in large-scale global continues optimization: A survey," *Inf. Sci.,* vol. 295, pp. 407-428, 2015.

[23] R. Cheng, and Y. Jin, "A Competitive Swarm Optimizer for Large Scale Optimization," *IEEE Trans. Cybern.,* vol. 45, no. 2, pp. 191-204, 2015.

[24] Q. Yang, W. N. Chen, T. Gu, H. Zhang, J. D. Deng, Y. Li, and J. Zhang, "Segment-Based Predominant Learning Swarm Optimizer for Large-Scale Optimization," *IEEE Trans. Cybern.,* vol. 47, no. 9, pp. 2896-2910, 2017.

[25] T. O'Brien, and D. Guiney, *Differentiation in teaching and learning: Principles and practice*: Wiley Online Library, 2001.

[26] J. C. Richards, and W. A. Renandya, *Methodology in language teaching: An anthology of current practice*: Cambridge university press, 2002.

[27] 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," *Technical Report, Nature Inspired Computation and Applications Laboratory, USTC, China*, 2010.

[28] 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," *Technical Report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia*, 2013.

[29] Q. Yang, W. N. Chen, Y. Li, C. L. P. Chen, X. M. Xu, and J. Zhang, "Multimodal Estimation of Distribution Algorithms," *IEEE Trans. Cybern.,* vol. 47, no. 3, pp. 636-650, 2017.

[30] Q. Yang, W. N. Chen, Z. Yu, T. Gu, Y. Li, H. Zhang, and J. Zhang, "Adaptive Multimodal Continuous Ant Colony Optimization," *IEEE Trans. Evol. Comput.,* vol. 21, no. 2, pp. 191-205, 2017.

[31] M. A. Potter, "The design and analysis of a computational model of cooperative coevolution," Ph.D. dissertation, George Mason University, 1997.

[32] 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, 2004.

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

[34] Y.-j. Shi, H.-f. Teng, and Z.-q. Li, "Cooperative co-evolutionary differential evolution for function optimization," *Advances in natural computation*, pp. 1080-1088: Springer, 2005.

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

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

[37] Y. Sun, M. Kirley, and S. K. Halgamuge, "Extended Differential Grouping for Large Scale Global Optimization with Direct and Indirect Variable Interactions," in *Proc. Conf. Genet. Evol. Comput.,*, 2015, pp. 313-320.

[38] Y. Mei, M. N. Omidvar, X. Li, and X. Yao, "A Competitive Divide-and-Conquer Algorithm for Unconstrained Large-Scale Black-Box Optimization," *ACM Trans. Math. Softw.,* vol. 42, no. 2, pp. 1-24, 2016.

[39] J. J. Liang, and P. N. Suganthan, "Dynamic multi-swarm particle swarm optimizer with local search," in *Proc. IEEE Congr. Evol. Comput.*, 2005, pp. 522-528.

[40] J. Kennedy, and R. Mendes, "Population structure and particle swarm performance," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 1-1962 Vol. 3.

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

[42] A. R. Mehrabian, and C. Lucas, "A novel numerical optimization algorithm inspired from weed colonization," *Ecol. Inform.,* vol. 1, no. 4, pp. 355-366, 2006.

[43] J. Sun, W. Xu, and B. Feng, "A global search strategy of quantum-behaved particle swarm optimization," in *IEEE Conf. on Cybern. Intell. Syst.*, 2004, pp. 111-116.

[44] K. Lian, X.-Y. Peng, and A. Ouyang, "An Efficient and Effective Algorithm for Large Scale Global Optimization Problems," *Int. J. Pattern Recogn.,* vol. 29, no. 04, pp. 1-22, 2015.

[45] A. LaTorre, S. Muelas, and J.-M. Peña, "A comprehensive comparison of large scale global optimizers," *Inf. Sci.,* vol. 316, pp. 517-549, 2015.

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

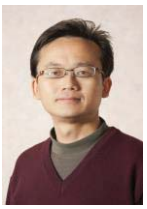
[47] R. Ros, and N. Hansen, "A simple modification in CMA-ES achieving linear time and space complexity," *Parallel Problem Solving from Nature–PPSN X*, pp. 296-305: Springer, 2008.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2743016, IEEE Transactions on Evolutionary Computation

[48] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: Memetic algorithm based on local search chains for large scale continuous global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1-8.
[49] A. LaTorre, S. Muelas, and J. M. Peña, "Multiple Offspring Sampling in Large Scale Global Optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2012, pp. 1-8.
[50] A. LaTorre, S. Muelas, and J. M. Pena, "Large scale global optimization: Experimental results with MOS-based hybrid algorithms," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 2742-2749.
[51] J. Brest, and M. S. Maučec, "Self-adaptive differential evolution algorithm using population size reduction and three strategies," *Soft Comput.,* vol. 15, no. 11, pp. 2157-2174, 2011.
[52] S.-Z. Zhao, P. N. Suganthan, and S. Das, "Self-adaptive differential evolution with multi-trajectory search for large-scale optimization," *Soft Comput.,* vol. 15, no. 11, pp. 2175-2185, 2011.
[53] J. Zhang, and A. C. Sanderson, "JADE: adaptive differential evolution with optional external archive," *IEEE Trans. Evol. Comput.,* vol. 13, no. 5, pp. 945-958, 2009.
[54] M. Z. Ali, N. H. Awad, and P. N. Suganthan, "Multi-population differential evolution with balanced ensemble of mutation strategies for large-scale global optimization," *Appl. Soft Comput.,* vol. 33, pp. 304-327, 2015.
[55] 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, 2014.
[56] R. A. Krohling, and E. Mendel, "Bare bones particle swarm optimization with Gaussian or Cauchy jumps," in *Proc. IEEE Congr. Evol. Comput.*, 2009, pp. 3285-3291.
[57] S.-Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren, "Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 3845-3852.
[58] J. Kennedy, "Stereotyping: improving particle swarm performance with cluster analysis ," in *Proc. IEEE Congr. Evol. Comput.*, 2000, pp. 1507-1512.
[59] S. Janson, and M. Middendorf, "A hierarchical particle swarm optimizer and its adaptive variant," *IEEE Trans. Syst., Man, Cybern. B, Cybern.,* vol. 35, no. 6, pp. 1272-1282, 2005.
[60] Y. V. Pehlivanoglu, "A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks," *IEEE Trans. Evol. Comput.,* vol. 17, no. 3, pp. 436-452, 2013.
[61] C. Segura, C. A. C. Coello, E. Segredo, and A. H. Aguirre, "A Novel Diversity-Based Replacement Strategy for Evolutionary Algorithms," *IEEE Trans. Cybern.,* vol. 46, no. 12, pp. 3233-3246, 2016.
[62] O. Olorunda, and A. P. Engelbrecht, "Measuring exploration/exploitation in particle swarms using swarm diversity," in *Proc. IEEE Congr. Evol. Comput.*, 2008, pp. 1128-1134.

Qiang Yang (S'14) received his M. S. degree from Sun Yat-sen University, China, in 2014, where he is currently pursuing his Ph. D. degree. He is now also a research assistant with School of Computer Science and Engineering, South China University of Technology, China. His current research interests include evolutionary computation algorithms and their applications on real-world problems. So far, he specifically works on large scale optimization algorithms, multimodal optimization algorithms, distributed evolutionary algorithms and their applications on real-world problems.

Wei-Neng Chen (S'07-M'12) received the Bachelor's degree and the Ph.D. degree from Sun Yat-sen University, China, in 2006 and 2012, respectively. He is currently a professor with the School of Computer Science and Engineering, South China University of Technology, China. His current research interests include swarm intelligence algorithms and their applications on cloud computing, operations research and software engineering. Dr. Chen has published more than 70 papers in international journals and conferences, including more than 20 papers in IEEE Transactions journals. His doctoral thesis received the IEEE Computational Intelligence Society (CIS) Outstanding Dissertation Award in 2016. He also received the National Science Fund for Excellent Young Scholars in 2016.



Jeremiah D. Deng (M'00) obtained the B.E. degree from the University of Electronic Science and Technology of China in 1989, and the M.Eng. and D.Eng. from the South China University of Technology (SCUT), Guangzhou, China, in 1992 and 1995, respectively, the latter co-supervised by the University of Hong Kong. From 1995 he was a lecturer at SCUT, and then joined the University of Otago, New Zealand in 1999 as a post-doctoral research fellow. He is now an Associate Professor at the Department of Information Science, University of Otago. Dr. Deng's research interests include machine learning, pattern recognition, and modeling and optimization of computer networks. He has published around 100 refereed research papers in international conference proceedings and journals.

Yun Li (S'87-M'90-SM'17) 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. During 1989-1990, he was with U.K. National Engineering Laboratory and Industrial Systems and Control Ltd. He joined University of Glasgow as Lecturer in 1991, served as two-year Founding Director of University of Glasgow Singapore during 2011-2013. He developed one of the world's first 30 evolutionary computation courses in 1995 and the popular online interactive courseware GA Demo in 1997. He established Evolutionary Computation workgroups for IEEE Control System Society and for European Network of Excellence in Evolutionary Computing (EvoNet) in 1998, and served on the Management Board of EvoNet during 2000-2005. Professor Li has supervised over 30 PhD students in computational intelligence since 1992, has 250 publications, and is a Chartered Engineer in the UK.

Tianlong Gu received the M.Eng. degree from Xidian University, China, in 1987, and the Ph.D. degree from Zhejiang University, China, in 1996. From 1998 to 2002, he was a Research Fellow with the School of Electrical and Computer Engineering, Curtin University of Technology, Australia, and a Post-Doctoral Fellow with the School of Engineering, Murdoch University, Australia. He is currently a Professor with the School of Computer Science and Engineering, Guilin University of Electronic Technology, China. His research interests include formal methods, data and knowledge engineering, software engineering, and information security protocol.

Jun Zhang (M'02-SM'08-F'17) received the Ph.D. degree in Electrical Engineering from the City University of Hong Kong in 2002. From 2004 to 2016, he was a professor with SUN Yat-sen University. Since 2016, he has been with South China University of Technology, Guangzhou, China, where he is currently a Cheung Kong Chair Professor. He has authored seven research books and book chapters, and over 100 technical papers in his research areas. He is Fellow of Institute of Electrical and Electronics Engineers (IEEE). His current research interests include computational intelligence, cloud computing, big data, high performance computing, data mining, wireless sensor networks, operations research, and power electronic circuits.

Professor Zhang was a recipient of the China National Funds for Distinguished Young Scientists from the National Natural Science Foundation of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the IEEE Transactions on Evolutionary Computation, 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 IEEE Beijing (Guangzhou) Section Computational Intelligence Society Chapters. He is the Founding and Current Chair of the ACM Guangzhou Chapter.