Learning-aided Evolution for Optimization

Zhi-Hui Zhan, *Senior Member*, *IEEE*, Jian-Yu Li, *Member*, *IEEE*, Sam Kwong, *Fellow*, *IEEE*, Jun Zhang, *Fellow*, *IEEE*

Abstract—Learning and optimization are the two essential abilities of human beings for problem solving. Similarly, computer scientists have made great efforts to design artificial neural network (ANN) and evolutionary computation (EC) to simulate the learning ability and the optimization ability for solving real-world problems, respectively. These have been two essential branches in artificial intelligence (AI) and computer science. However, in humans, learning and optimization are usually integrated together for problem solving. Therefore, how to efficiently integrate these two abilities together to develop powerful AI remains a significant but challenging issue. Motivated by this, this paper proposes a novel learning-aided evolutionary optimization (LEO) framework that plus learning and evolution for solving optimization problems. The LEO is integrated with the evolution knowledge learned by ANN from the evolution process of EC to promote optimization efficiency. The LEO framework is applied to both classical EC algorithms and some state-of-the-art EC algorithms including a champion algorithm, with benchmarking against the IEEE Congress on Evolutionary Computation competition data. The experimental results show that the LEO can significantly enhance the existing EC algorithms to better solve both single-objective and multi-/many-objective global optimization problems, suggesting that learning plus evolution is more intelligent for problem solving. Moreover, the experimental results have also validated the time efficiency of the LEO, where the additional time cost for using LEO is greatly deserved. Therefore, the promising LEO can lead to a new and more efficient paradigm for EC algorithms to solve global optimization problems by plus learning and evolution.

***Index Terms*—Evolutionary computation, learning-aided evolution, artificial neural network, particle swarm optimization, differential evolution, single-objective optimization, multi-objective optimization, many-objective optimization**

I. Introduction

Learning and optimization are two basic abilities of human beings for problem solving [1]. Similarly, artificial neural networks (ANNs) for learning and evolutionary computation (EC) algorithms for optimization are the two typical branches of artificial intelligence (AI), and both have obtained significant developments along with the development of computer science [2]. In general, the ANN is a kind of connectionism AI that simulates the brain structure to assist machine in learning knowledge, while the EC is a kind of evolutionism AI that simulates the evolution phenomena and intelligent behaviors of humans or swarm animals for problem solving [2]-[4], both of which are essential tasks of AI.

Currently, the ANN-based learning branch has aroused great attention due to the success of deep learning (DL) in various real-world applications [5]-[7]. More significantly, EC algorithms have also made the great pace in research and applications [8]-[13]. EC algorithms were born in the 1960s, when computer scientists designed EC algorithms such as the genetic algorithm (GA) [14][15], evolution strategy [16], and evolutionary programming [17]-[19] for solving optimization problems. Since then, EC algorithms have attracted great attention and interest in the global optimization community. EC algorithms are promising because they do not require the strict mathematical characteristics of the problem and can find the global optimum or near-global optimum within an acceptable time. Generally, EC is a common framework that simulates the evolutionary mechanism of biology (e.g., GA [14] and differential evolution (DE) [20]-[23]) and the swarm intelligence behaviors of animals/insects (e.g., particle swarm optimization (PSO) [24]-[26] and ant colony optimization (ACO) [27]-[30])). The core idea of EC algorithms is "survival of the fittest". That is, new solutions are generated by simulating evolutionary phenomena such as crossover and mutation. Then, the new solutions (i.e., offspring) and the old solutions (i.e., parents) compete to survive in the next generation. This is similar to the principle of natural selection in nature. By doing so, the solutions in the new generation are expected to be better than those in the old generations. Consequently, EC algorithms can gradually approach the global optimum generation by generation. The generic evolutionary framework of an EC algorithm is illustrated in the top of Fig. 1, where g is the generation index and will be increased by $g=g+1$ after the selection until the algorithm termination criteria are met.

However, solving the problem only by evolution may be inefficient. For example, in nature, evolution may need thousands of years to improve a species, while learning can help accelerate evolution dramatically. Concerning this, an insightful question is: Can we combine learning with evolution to solve complex problems more efficiently, just like the combination of the learning and optimization abilities of human beings? Moreover, as mentioned before, optimization using EC

Manuscript received XXXX; revised XXXX; accepted XXXX. This work was supported in part by the National Natural Science Foundations of China under Grant 62176094 and Grant 61873097, in part by the Guangdong Natural Science Foundation Research Team under Grant 2018B030312003, in part by the National Research Foundation of Korea (NRF-2022H1D3A2A01093478), in part by the Hong Kong GRF-RGC General Research Fund under Grant 11209819 (CityU 9042816) and Grant 11203820 (CityU 9042598), and in part by the Hong Kong Innovation and Technology Commission (InnoHK Project CIMDA). (*Corresponding authors: Zhi-Hui Zhan; Jun Zhang.*)

Zhi-Hui Zhan and Jian-Yu Li are with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: zhanapollo@163.com).

Sam Kwong is with the Department of Computer Science, City University of Hong Kong, Hong Kong.

Jun Zhang is with the Zhejiang Normal University, Jinhua 321004, China, and also with Hanyang University, Ansan, 15588, South Korea.

and learning using ANN are two main streams of AI, thus their combination is promising for designing powerful AI for achieving greater problem-solving ability.

In the history of AI, the two branches of EC and ANN mainly develop independently and sometimes interact with each other, resulting in the evolutionary neural network [31]-[33], evolutionary DL [34], evolutionary machine learning [35], neuroevolution [36], and machine learning-enhanced EC [37]. However, few enhanced EC algorithms have employed ANNs to evolve solutions. As ANNs have already performed quite well in learning tasks, it is worth researching their ability to learn "*how to evolve well*" from EC process to assist optimization tasks. In this way, the performance of EC algorithms can be surely improved by using ANNs. This can be a new way to design more intelligent systems for knowledge discovery and problem solving. Therefore, we propose a learning-aided evolutionary optimization (LEO) framework that can upgrade the performance of EC algorithms by plus using an ANN learning system. The LEO is a new EC paradigm that drives EC optimization using both the evolutionary fitness from problems and the learned evolution knowledge obtained by an ANN. The general idea of the LEO framework is given in Fig. 1.

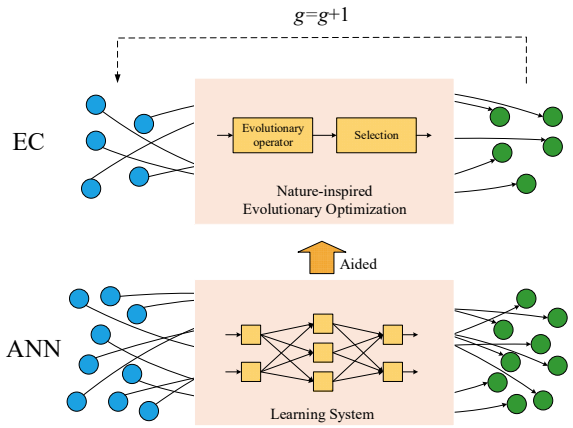

Fig. 1. The main idea of LEO.

As shown in Fig. 1, the learning system in LEO aims to learn knowledge from the EC algorithm process. Herein, the learned knowledge is the "evolution knowledge" that can guide the EC algorithm "how to evolve well" (e.g., to efficiently help the EC algorithm better approach the global optimum). To this aim, we collect evolutionary data during the EC process as evolutionary pattern (EP) and use the ANN to learn from the successful EP (SEP) to obtain the evolution knowledge. Specifically, the EP of a solution can be represented by a pair of solutions that contains the original version of this solution in the current generation and the changed version of this solution in the next generation. In this sense, the SEP is defined as a successful pair of solutions whose changed version has better fitness than its original version. As shown in Fig. 1, in the learning system, if a pair of solutions (i.e., the old solution and the new solution) is successful, i.e., the new solution has a better fitness value than the old solution, then the relationship of this pair can be regarded as a kind of SEP that can help a solution evolve better. During the evolutionary process of the EC algorithm, many such successful pairs of solutions (i.e., SEPs) can be collected. Therefore, the learning system can be trained by using these collected SEPs, where the learned evolution knowledge from SEPs can be used to promote the evolutionary process.

In order to learn from these SEPs, in this paper, the learning system in LEO is configured by using a very simple yet efficient ANN as an example. The training input of the ANN is the old solution of the successful pair while the output of the ANN is the corresponding new solution of the pair. After the training process with all the pairs, the ANN becomes a model that can help an input solution evolve better. That is, by learning a large number of SEPs, the ANN can be trained into a model that can receive a solution as input and then output a solution with better fitness. This is helpful to push the EC algorithms to better regions of the search space. Therefore, LEO can also be interpreted as "Learning + Evolution → Optimization", which is an emerging and promising AI paradigm in the future EC era, as shown in Fig. 2. Fig. 2 also shows that the EC algorithms are different from and more efficient than the random search algorithms before the EC era due to the fitness-driven mechanism of evolution. Generally speaking, fitness information can be regarded as problem-related knowledge because it is designed by domain experts and is used for driving the evolution of the EC population. Nevertheless, the EC population also yields data and information during the evolution process, which can be learned and regarded as evolution-related knowledge to further enhance the EC algorithm. Therefore, in the future EC era, the next generational EC algorithms not only will be fitness-driven (data and knowledge derived from the problem), but also will be learning-aided (data and knowledge derived from the evolution), i.e., learning plus evolution, such as the LEO new paradigm proposed in this paper.

In the experiments, the proposed LEO is evaluated on not only the single-objective optimization problems but also the multi- and many-objective optimization problems. To be specific, the single-objective optimization problems are actually the newest IEEE Congress on Evolutionary Computation (CEC) 2021 competition benchmark problems for single-objective real parameter numerical optimization [38]. These benchmark problems consist of the most challenging complex optimization problems from the previous IEEE CEC competitions, e.g., IEEE CEC 2014, 2017, and 2018 competitions. Moreover, they are the most challenging and well-known benchmarks in the EC community. As for multi-/many-objective problems, the widely-used benchmark suite, i.e., DTLZ [39], is adopted in this paper with the number of objectives set as 5, 10, 15, and 20, respectively. Moreover, we apply the LEO framework not only to the classical algorithms for single-objective optimization (i.e., PSO and DE) and multi-objective optimization (NSGA-II [40] and MOEA/D [41]), but also to the state-of-the-art algorithms, e.g., the champion algorithm named NL-SHADE-RSP [42] on IEEE CEC 2021 competition.

The rest contents are organized as follows: Section II gives a

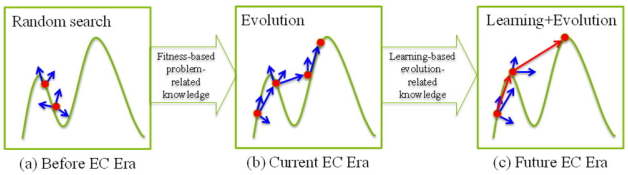

Fig. 2. The development of EC.

brief introduction of the EC, ANN, and related work. Section III details the LEO, while Section IV illustrates the experiments with results, comparisons, and analyses. Finally, the conclusion is provided in Section V.

II. Background and Related Work

A. Evolutionary Computation

EC algorithms can be divided into two categories, including evolutionary algorithm (EA) and swarm intelligence (SI) [2]. Herein, the following contents introduce two typical and widely-used EC algorithms for optimization problems, i.e., the PSO [24] as a typical SI and the DE [23] as a typical EA.

1) Particle Swarm Optimization

PSO is a classical and efficient SI algorithm in the EC family [24]. Modeling the behavior of bird flocks, PSO adopts a set of particles as a swarm to search for the global optimum. Each particle has a position vector to represent the candidate solution in the whole search space (i.e., the problem space) and uses a velocity vector to indicate the search direction.

First, the PSO initializes the positions and velocities of particles randomly within their corresponding upper bound and lower bound to represent their initial status. Then, the fitness values of all particles will be evaluated with the objective function of the problem. After the fitness evaluation, the personally best position (*pBest*) of each particle in their experience and the globally best position (*gBest*) of the whole swarm will be determined. Then, the PSO goes into the loop to conduct the evolutionary operators, such as velocity and position updates, new solution evaluations, and *pBest* and *gBest* updates, generation by generation. These processes are iteratively performed to evolve particles to find better solutions. Finally, when the algorithm meets the termination conditions, the algorithm will output the newest *gBest* as the final optimal solution. In particular, the velocity and position updates are performed according to Eq.(1) and Eq.(2), respectively.

$$V_{i,j} = w \cdot V_{i,j} + c_1 \cdot r_{1,j} \cdot (pBest_{i,j} - X_{i,j}) + c_2 \cdot r_{2,j} \cdot (gBest_j - X_{i,j}) \quad (1)$$

$$X_{i,j} = X_{i,j} + V_{i,j} \quad (2)$$

where $X_{i,j}$ and $V_{i,j}$ are the j^{th} dimension's position and velocity of the i^{th} particle, respectively; ω represents the inertia weight; c_1 and c_2 are predefined coefficients; and $r_{1,j}$ and $r_{2,j}$ are two values randomly sampled in the range [0,1].

2) Differential Evolution

DE is an easy-to-implement and efficient EA in the EC family. It treats candidate solutions as individuals, i.e., X_i, and evolves individuals based on the individual difference.

After the random initialization of individuals in the search space, DE iteratively evolves individuals by three evolutionary operators until the stop criteria are met. These three operators include mutation, crossover, and selection, as introduced as follows.

Mutation: a mutation vector T_i is computed for each individual X_i with the position of other individuals in every generation. "DE/rand/1" is a frequently-used strategy for computing the mutation vector, which is listed as below:

$$T_i = X_{r1} + F \cdot (X_{r2} - X_{r3}) \quad (3)$$

where r_1, r_2, and r_3 represent the three randomly-selected individual indexes different from each other and different from i, F is the mutation parameter.

Crossover: A trial vector U_i will be generated for each X_i via crossover based on X_i and T_i. The widely-used crossover operation is described as:

$$U_{i,j} = \begin{cases} T_{i,j} & \text{if } rand_j \le CR \text{ or } j = j_{rand} \\ X_{i,j} & \text{otherwise} \end{cases} \quad (4)$$

where $rand_j$ is a value sampled from [0,1] randomly, j_{rand} is the random dimension index, which guarantees that U_i adopts at least one dimension from T_i, CR is the crossover rate.

Selection: In the selection, individuals with better fitness will be selected to enter into the next generation. Herein, the one-to-one selection in DE is considered, i.e., the better one between X_i and U_i will be selected.

B. Artificial Neural Network

As a learning system that simulates the human brain, ANN can map the input data to the targeted data after training the weights parameters of neurons based on the training data (i.e., input-output pairs). Therefore, given the successful evolution pairs of solution positions, the ANN can learn how to map poor solution positions to better solution positions through training. As this paper adopt the multi-layer feed-forward NN (a widely-used ANN) [43] as the learning system in LEO, this part briefly introduces the multi-layer feed-forward NN.

In general, a multi-layer feed-forward NN contains several layers with corresponding neurons and connections. For example, Fig. 3 presents a multilayer feed-forward NN with N inputs and M outputs, where each connection (i.e., arrow) between the neurons in different layers means that the output of the neuron in the previous layer is the input of the neuron of the next layer. In addition, to provide the NN with nonlinear mapping ability, the activation function is used in the neuron of the hidden layer. In this paper, we consider the widely-used function, i.e., the sigmoid function, as the activation function in the neurons of hidden layers. Given the input as z, the sigmoid function can be written as

$$\sigma(z) = \frac{1}{1+e^{-z}} \quad (5)$$

C. Related Work

To date, the research into using learning into EC has caused certain attention [37]. According to the learning purposes, the existing works can be mainly categorized into learning the problem knowledge and learning the solution knowledge.

In the first category, learning the fitness as problem knowledge is a popular research topic. This is also known as data-driven EC because it uses evaluated individuals as data to learn the fitness for building a fitness surrogate, which is promising for solving expensive optimization problems [4]. For example, Wang *et al.* [44] proposed a selective ensemble

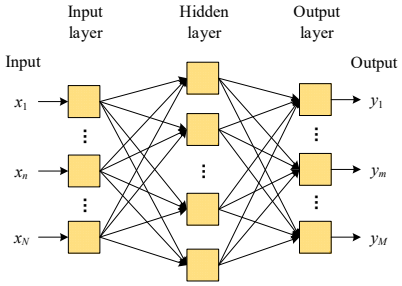

Fig. 3. An example of multi-layer feed-forward neural network with N inputs and M outputs.

surrogate to learn the fitness knowledge efficiently. Li *et al.* [45] proposed a data-driven EC with localized data generation method to learn the fitness knowledge from few-shot evaluated solution data. Also, to learn the fitness knowledge from few-shot evaluated data, Li *et al.* [46] proposed a multiple surrogate method based on data perturbation, which has obtained promising results. Moreover, Ji *et al.* [47] proposed a dual-layer cooperative surrogate model to learn the fitness knowledge of multimodal problems. Li and Zhang [48] proposed to use multiple local surrogates to learn the fitness and constraint knowledge from the constrained problems. However, data-driven EC mainly focuses on approximating the fitness and/or constraint evaluations via learning from the evaluated data. Differently, the proposed LEO mainly focuses on learning the SEP so as to help for better population reproduction and variation. Therefore, to better show their differences, the differences between the LEO and data-driven EC algorithms are briefly summarized in Table S.I of the supplemental material.

In the second category, the solution knowledge is learned mainly for the purposes of reusing past knowledge, transferring assisted knowledge, and predicting promising solutions. For example, Feng *et al.* [49] proposed to learn past knowledge from the solutions of the already solved vehicle routing problems and then reuse them to produce routing solutions for the current problem in a more efficient way. Zhou *et al.* [50] proposed to generate promising solutions via knowledge transfer from the promising solutions of other problems. Moreover, Feng *et al.* [51] proposed to predict the moving of optimal solutions by learning the dynamic change of the old optimal solutions in the old dynamic environments, so as to benefit the population reproduction in the current environment. Liu *et al.* [52] proposed a neural network-based information transfer method to transfer the solutions in the past environment to the promising solutions in the new environment. Also reusing knowledge of old solutions, Zhan *et al.* [53] proposed the adaptive distributed DE, where the promising individual positions in previous generations are maintained in the archive and then are reused for the individual evolution in the current generation. In addition, Ghosh *et al.* [54] proposed a difference vector reuse mechanism to reuse the successful differential vector learned from the solutions in the past generation to promote the evolution of the current population. Xia *et al.* [55] proposed a PSO with triple archives to store the promising particle positions as knowledge in different archives and reuse them to help better evolution.

The above methods have obtained potential results for improving the performance of EC, which suggests the significance of learning in evolution. Different from the above methods that focus on learning the problem knowledge and solution knowledge, the LEO proposed in this paper aims to learn the evolution knowledge, i.e., "*how to evolve well*", so as to promote population reproduction and optimization efficiency.

III. Learning-Aided Evolutionary Optimization

A. The LEO Framework

The overall LEO framework is illustrated in Fig. 4, which contains two parts, i.e., the evolution part and the learning part. When compared with the traditional EC framework, the major novelty of LEO lies in that it uses not only the traditional evolutionary operator but also the learning-aided evolutionary operator to generate new promising individuals. In the learning part, the LEO trains the learning system with the successful solution pairs collected from the evolution part, so as to learn the SEPs. By learning from a large number of successful solution pairs, the ANN can learn enough SEPs to help achieve better individual evolution, which may be more straightforward and efficient than traditional evolutionary operators.

In the LEO framework, the learning system can be simple by using simple ANN (as studied in this paper), and also can be complex if complex learning systems like deep networks are adopted (which can be studied in the future). Moreover, the learning-aided evolutionary operator is a generic operator and should be particularly designed based on the particular EC algorithm. The particular realization is described as follows.

B. Successful Solution Pairing

As data is significant for training the learning system, the quality of successful evolution pairs will greatly influence the performance of the learning system in learning-aided evolution. In fact, the definition of successful solution pair can vary, because that many individuals in the new generation can be regarded as the successful evolution of the individuals with worse fitness in the old generation. Without loss of generality, this paper defines the successful solution pair as the SEP, which is described as follows. Given the i^{th} individual in the generation g and g+1, denoted as $X_{g,i}$ and $X_{g+1,i}$, respectively, ($X_{g,i}$, $X_{g+1,i}$) is defined as a successful solution pair (i.e., a SEP) if the fitness of $X_{g+1,i}$ is better than $X_{g,i}$.

Note that if in PSO, the X means the personal best position of the particle. That is, the successful solution pair ($pBest_{g,i}$, $pBest_{g+1,i}$) is regarded as the SEP if $pBest_{g+1,i}$ is better than $pBest_{g,i}$. Moreover, for multi-objective problems, the ($X_{g,i}$, $X_{g+1,i}$) is defined as a successful solution pair if $X_{g+1,i}$ dominates $X_{g,i}$.

C. Learning SEPs

The learning system in LEO aims to learn the SEP by using successful pairs of solutions. Therefore, in every generation, LEO collects all the successful pairs of solutions as training data and uses the collected data to update the ANN, i.e., trains the weights of the ANN. However, two issues should be considered in the data collection. The first is that the number of collected successful solution pairs will increase rapidly as the evolution goes on, and the second is that the SEPs learned in

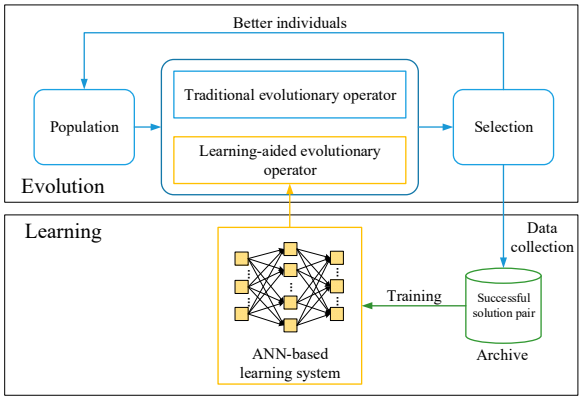

Fig. 4. The framework of LEO.

earlier generations may not be suitable for the solutions in later generations. With these two concerns, we use an external archive to store the successful solution pairs and update the archive periodically. By doing so, the number of available successful solution pairs can be adaptively controlled, and the out-of-date pairs can be naturally dropped. Specifically, after the data collection in every generation, LEO will check whether the number of collected successful solution pairs exceeds the maximum size of the archive, say *arch_size*. If exceeds, LEO only reserves the newest *arch_size* pairs and drops the old ones. Otherwise, the archive will not be changed.

With the data (i.e., the successful solution pairs) in the archive, the ANN can adaptively update its weights to learn SEPs better. Specifically, the ANN conducts one backpropagation (i.e., one epoch) in every generation with all the data in the archive to update its weights. The loss function of the ANN is the mean squared error between the network output and the target output, as can be written as:

$$MSE=\frac{1}{arch\_size}\sum_{(X_i,X_i')\in arch}|L(X_i)-X_i'|^2 \tag{5}$$

where (X_i, X_i') is a successful solution pair stored in the *arch*, $L(X_i)$ is the output of the learning system with the solution X_i as the input. Note that other loss functions can be used if needed.

As the generation goes on, the ANN will conduct many epochs of training, and each training will use the newest successful solution pairs as the training data.

D. Learning-aided Evolutionary Operator

The learning-aided evolutionary operator aims to use the learning system to drive the evolution of candidate solutions. By learning the SEPs, the learning system in LEO is able to know how to better optimize the position of an individual to enhance its fitness. That is, given an individual as an input, the learning system can output a new individual with better fitness. To fully utilize the learned evolution knowledge, we propose a learning-aided mutation (*LM*) operator and a learning-aided crossover (*LC*) operator. The *LM* operator can be written as:

$$\begin{aligned} newX &= LM(X_A, X_B, X_C) \\ &= L(X_A)+\alpha\cdot(X_B-X_C) \end{aligned} \tag{6}$$

where X_A, X_B, and X_C are three solutions as the inputs of *LM*, the $L(X_A)$ is the output of the learning system with the solution X_A as the input, and α is a parameter for the learning-aided mutation. Moreover, the *LC* operator can be written as:

$$newX = LC(X_A, X_B)$$

$$\text{where } newX_j=\begin{cases} X_{A,j}, & \text{if } rand_j \le \beta \\ X_{B,j}, & \text{otherwise} \end{cases} \tag{7}$$

where X_A and X_B are the two solutions as the inputs of *LC*, j represents the dimension index, $rand_j$ is a random value within [0,1], and β is a parameter in the learning-aided crossover.

Based on the above, the *LM* and *LC* in PSO are implemented as:

$$newX_i = LM(pBest_i, pBest_{r1}, pBest_{r2}) \tag{8}$$

$$newX_i = LC(pBest_i, newX_i) \tag{9}$$

where $pBest_i$ is the personally best position of the ith particle, $pBest_{r1}$ and $pBest_{r2}$ are the personally best positions of two different random particles.

As for DE, there are no personally best positions of individuals. Therefore, the individuals themselves in the current generation are used for implementing the *LM* and *LC*. That is, the *LM* and *LC* operators in DE can be rewritten as

$$newX_i = LM(X_i, X_{r1}, X_{r2}) \tag{10}$$

$$newX_i = LC(X_i, newX_i) \tag{11}$$

where X_i is the ith individual of the current generation, and X_{r1} and X_{r2} are the positions of two different random individuals in the current generation. Note that the Eq.(6) to Eq.(11) are examples to show how to design the learning-aided evolutionary operator, and without loss of generality, these operators can be substituted by other evolutionary operators with the output of the learning system (e.g., $L(pBest_i)$ and $L(X_i)$) to develop more efficient learning-aided evolutionary operator to enhance the LEO-based algorithms.

E. Combination of Traditional Evolutionary Operator and Learning-aided Evolutionary Operator

As mentioned above, some individuals can be successfully evolved by the well-trained learning system via the learning-aided evolutionary operator, which can be more efficient and straightforward than the traditional evolutionary operator. However, as the individuals and their successful evolution differ from generation to generation, the learning system needs to update itself adaptively with new SEPs in the latest generations. Otherwise, the learning system trained with out-of-date SEPs will become unsuitable for driving the evolution of individuals in current and future generations and stages. In order to collect more latest SEPs to update the learning system, the traditional evolutionary operator is still essential to explore new successful solution pairs. Therefore, LEO uses both the traditional evolutionary operator and the learning system to evolve the individuals instead of only using the learning system (i.e., only using the learning-aided evolutionary operator). With this concern, a parameter of learning probability *lp* is proposed for controlling the use of traditional evolutionary operator and learning-aided evolutionary operator in each generation. To be specific, for each generation, if a value randomly sampled within [0, 1] is smaller than *lp*, then the individuals will be evolved by the traditional evolutionary operator, so as to help explore the unknown landscapes to discover better successful solution pairs. Otherwise, the individuals will be evolved by the

Algorithm 1: LEO-based EC algorithm

```
1:Begin
2:    // Initialization
3:    g ←1; // the generation index
4:    Initialize population X_g and evaluate the fitness;
5:    Initialize the weights of ANN randomly;
6:    Initialize arch as an empty set; // to store solution pairs
7:    While stop criteria not satisfied Do
8:       g ←g + 1;
9:       // Individual Evolution
10:      Sample r uniformly on [0,1];
11:      If g >1 and r< lp Then
12:         newX ← Evolve X_g by learning-aided evolutionary operator;
13:      Else
14:         // use operators in traditional EC, e.g., PSO or DE
15:         newX ← Evolve X_g by traditional evolutionary operator;
16:      End If
17:      Evaluate the fitness of individuals in newX;
18:      // Selection
19:      X_{g+1} ← selection among X_g and newX;
20:      // SEP Collection
21:      For each individual i in X_{g+1} Do
22:         If X_{g+1,i} is better than X_{g,i} Then
23:            Add (X_{g,i}, X_{g+1,i}) in arch;
24:         End
25:      End For
26:      If number of SEPs in arch > arch_size Then
27:         Arch ← the newest arch_size solution pairs;
28:      End If
29:      // Learning System Update
30:      Train the ANN with all data in arch for one epoch;
31:   End While
32:End
```

learning-aided evolutionary operator, so that the learned evolution knowledge can be well utilized. By doing so, the traditional and learning-aided evolutionary operator can work cooperatively to evolve individuals more efficiently.

F. The Whole LEO-based Algorithm

Herein, the whole LEO-based EC algorithm is presented in **Algorithm 1**. In the algorithm, there are five main procedures: initialization, individual evolution, selection, SEP collection, and learning system update. In the individual evolution, the learning system (i.e., ANN) will be used to evolve individuals via learning-aided evolutionary operators only when the index of the generation number is larger than one (i.e., g>1) and a randomly generated value is smaller than lp (i.e., r<lp), as shown in line 11 of **Algorithm 1**. The reason is that if g>1, then the ANN has been surely updated with the solution pairs in the archive, as shown in line 30 of **Algorithm 1**. In other words, the ANN will not be used to help the individual evolution until its first update, which can guarantee that the ANN has at least learned evolution knowledge from the successful evolution pairs once. After the individual evolution, the algorithm will carry out selection and SEP collection. In this procedure, the successful pairs of solutions are added in the archive *arch* as training data. The above four procedures (except the initialization) will be carried out iteratively until the algorithm meets the stop criteria. Lastly, the algorithm outputs the best-found solution as the result.

IV. Experimental Studies

A. Experimental Settings

To evaluate LEO, experiments are conducted on not only single-objective optimization problems but also multi-/many-objective optimization problems. Moreover, corresponding state-of-the-art single-objective and multi-/many-objective algorithms are used in the experiments. To be more specific, the experimental settings are given as follows:

1) Experimental Settings for Single-objective Optimization

For experiments on single-objective optimization problems, the LEO-based algorithms will be tested on benchmark problems of the IEEE CEC 2021 Single-Objective Real Parameter Numerical Optimization Competition [38]. This benchmark set, which is the latest and most well-known benchmark in the EC community, includes 10 complex minimization problems that have been selected on purpose from the IEEE 2014, 2017, and 2018 CEC competitions [38], [42]. As these problems span a wide variety of function characteristics, such as multimodal and non-separable, they are suitable for investigating and evaluating the effectiveness and general performance of the proposed LEO framework from different aspects. More significantly, the well-known and typical properties of these problems will provide in-depth observations on how LEO-based algorithms may behave in various situations. In addition, each problem can be set as a 10- or 20-dimensional minimization problem, and the lower and upper bound of each dimension are –100 and 100, respectively, as recommended by the official of the CEC 2021 Competition [38]. Therefore, in this paper, each of the 10 problems is set with 10 and 20 dimensions (i.e., 10D and 20D), and these 20 test problems are all adopted for the evaluation of LEO. In addition, on these benchmark problems, we apply LEO not only to PSO and DE, but also to the state-of-the-art algorithm NL-SHADE-RSP [42], which is a champion algorithm in IEEE CEC 2021 competition, to further investigate the effectiveness of LEO. Note that the population size of both PSO and DE is set as 100, while the NL-SHADE-RSP uses a dynamic population size due to its integrated population size reduction mechanism. To obtain fair comparisons, 2×10^5 and 1×10^6 are set as the maximum number of fitness evaluations for 10D and 20D problems, respectively, as suggested by the official of the CEC 2021 Competition [38].

2) Experimental Settings for Multi-/Many-objective Optimization

For experiments on multi-/many-objective optimization problems, the LEO will be combined with the classical multi-objective algorithm NSGA-II [40], MOEA/D [41], and a state-of-the-art algorithm named RVEA [56]. The LEO-based algorithms are tested on a widely-used multi-objective benchmark suit, i.e., the DLTZ benchmark set [39]. Note that as problem 8 and problem 9 in the DLTZ benchmark contain constraints, algorithms that are not designed for constrained problems, such as the MOEA/D, may generate infeasible solutions. Therefore, this paper only uses the DTLZ1 to DTLZ7 to conduct the experiments. To further challenge the LEO-based algorithms, the number of objectives in each of the seven problems are set as 5, 10, 15, and 20. That is, 7×4=28 multi-/many-objective optimization problems will be used in the experiments. All the experiments are conducted with the maximum number of evaluations set as 1×10^4 for each problem. Furthermore, the settings of NSGA-II, MOEA/D, and RVEA

TABLE I
COMPARISONS BETWEEN LEO-BASED ALGORITHMS AND THE ORIGINAL ALGORITHMS ON SINGLE-OBJECTIVE OPTIMIZATION PROBLEMS

Problem	L-PSO	PSO	L-DE	DE	L-NL-SHADE-RSP	NL-SHADE-RSP
F01(10D)	**0±0**	2.38E-05±6.01E-05 (+)(l)	**0±0**	**0±0 (≈)(s)**	**0±0**	**0±0 (≈)(s)**
F02(10D)	**4.10E+00±2.16E+01**	8.71E+02±2.70E+02 (+)(l)	**3.14E-01±8.09E-02**	3.45E+02±1.85E+02 (+)(l)	**0±0**	**0±0 (≈)(s)**
F03(10D)	**8.41E+00±4.72E+00**	7.40E+01±9.91E+00 (+)(l)	**1.10E+01±2.63E-01**	2.32E+01±4.04E+00 (+)(l)	**0±0**	**0±0 (≈)(s)**
F04(10D)	**4.06E-01±5.93E-02**	6.52E+00±1.04E+00 (+)(l)	**4.20E-01±6.02E-02**	1.50E+00±5.02E-01 (+)(l)	**3.62E-03±7.10E-03**	1.43E-02±2.14E-02 (+)(l)
F05(10D)	**5.19E+01±5.91E+01**	8.67E+01±8.02E+01 (+)(m)	1.53E-01±1.54E-01	**1.32E-01±1.77E-01 (-)(s)**	**0±0**	**0±0 (≈)(s)**
F06(10D)	**3.68E-01±2.17E-01**	2.24E+01±3.39E+01 (+)(l)	3.66E-01±2.43E-01	**1.61E-01±1.89E-01 (-)(l)**	**7.17E-02±1.80E-01**	**7.17E-02±1.80E-01(≈)(s)**
F07(10D)	3.62E+01±5.49E+01	**1.64E+01±3.60E+01 (-)(l)**	1.47E-01±1.92E-01	**1.30E-01±1.76E-01 (-)(s)**	**6.33E-04±1.04E-03**	1.38E-03±1.66E-03 (+)(l)
F08(10D)	**0±0**	3.37E+02±7.96E+01 (+)(l)	**0±0**	**0±0 (≈)(s)**	**0±0**	**0±0 (≈)(s)**
F09(10D)	**0±0**	5.43E-04±1.18E-03 (+)(l)	**0±0**	**0±0 (≈)(s)**	**0±0**	**0±0 (≈)(s)**
F10(10D)	**4.81E+01±1.20E-01**	5.25E+01±5.84E-01 (+)(l)	**4.81E+01±1.20E-01**	4.85E+01±1.59E-01 (≈)(s)	**1.88E-03±1.00E-03**	1.93E-03±9.41E-04(+)(s)
F01(20D)	**0±0**	1.97E+08±5.10E+07 (+)(l)	**0±0**	**0±0 (≈)(s)**	**0±0**	**0±0 (≈)(s)**
F02(20D)	**4.22E+00±2.16E+01**	2.74E+03±2.92E+02 (+)(l)	**3.71E-01±6.50E-02**	6.43E+00±2.77E+00 (+)(l)	**0±0**	**0±0 (≈)(s)**
F03(20D)	**1.60E+01±8.78E+00**	1.87E+02±1.79E+01 (+)(l)	**2.02E+01±0.00E+00**	2.34E+01±4.32E+00 (+)(l)	**0±0**	**0±0 (≈)(s)**
F04(20D)	**8.14E-01±1.13E-01**	1.73E+01±1.80E+00 (+)(l)	**7.65E-01±8.46E-02**	2.75E+00±2.32E+00 (+)(l)	**1.62E-02±1.70E-02**	3.16E-02±1.52E-02 (+)(l)
F05(20D)	**1.18E+02±1.04E+02**	1.19E+03±3.36E+02 (+)(l)	**8.46E-01±9.04E-01**	1.97E+00±2.58E+00 (+)(l)	**0±0**	**0±0 (≈)(s)**
F06(20D)	**2.10E+00±1.84E+00**	3.13E+02±1.20E+02 (+)(l)	**8.47E-01±5.75E-01**	2.11E+00±2.38E+00 (+)(l)	**2.13E-02±2.48E-02**	3.90E-02±1.82E-02 (+)(l)
F07(20D)	**5.51E+01±8.32E+01**	2.95E+02±1.58E+02 (+)(l)	**8.01E-01±1.45E-01**	1.38E+00±2.13E+00 (+)(l)	**9.71E-03±1.18E-02**	1.52E-02±1.05E-02 (+)(s)
F08(20D)	**1.72E+00±3.91E+00**	1.08E+03±1.49E+02 (+)(l)	**0±0**	3.29E+00±5.78E+00 (+)(l)	**0±0**	**0±0 (≈)(s)**
F09(20D)	**0±0**	5.55E+01±1.28E+01 (+)(l)	**0±0**	**0±0 (≈)(s)**	**0±0**	**0±0 (≈)(s)**
F10(20D)	**5.01E+01±4.73E+00**	5.86E+01±1.64E+00 (+)(l)	**4.88E+01±6.53E-02**	**4.88E+01±2.70E-02 (≈)(s)**	**4.12E-03±1.72E-03**	4.29E-03±1.54E-03 (+)(s)
+/≈/-	NA	19/0/1	NA	10/7/3	NA	7/13/0
Adjusted *p*-value	NA	0.0008	NA	0.0201	NA	0.0707

#"s", "m", and "l" mean the effect size d is regarded as: small if $0.2 \leqslant d < 0.5$; medium if $0.5 \leqslant d < 0.8$, and large if $0.8 < d$, respectively.

are set according to the default settings in their original references, and the MOEA/D uses the PBI reference vectors.

3) Other Experimental Settings

As for the setting of LEO-based algorithms, the *arch_size* is set as 100, while α and β in the learning-aided evolutionary operator are set as 0.5 and 0.9 respectively. The learning system ANN is set with three layers, which include one input layer, one hidden layer, and one output layer. As both the input and output of the ANN are candidate solutions, the dimension of the input and output layer will be D, where D is the problem dimension. Besides, the number of neurons in the hidden layer is set as $3 \times D$. Note that the input and output values of the ANN should be in the range of [0,1]. Therefore, each dimension value of solutions will be transformed from the original search space to [0,1] to be the input of ANN and the output values of the ANN will be transformed from [0,1] back to the corresponding search space to be the generated individual. For the training and implementation of ANN, as mentioned earlier in Section III-C, the LEO-based algorithms will randomly initialize the ANN at the beginning and then perform one epoch of backpropagation in every generation with all the data in the archive. To reduce the statistical errors, each algorithm will be conducted 30 times on each single or multi-/many-objective optimization problem independently, so as to get the statistical results for the comparison. Besides, this paper uses the Wilcoxon's rank sum test with a significance level of α=0.05 to determine whether the algorithms with LEO are significantly better than, statistically similar to, and significantly worse than the algorithms of original versions on each problem in terms of the optimization accuracy. In addition, the Friedman test with Holm post-hoc analysis with a significant level of 0.05 is also used to compare the algorithm performance on multiple problems. Based on this, the compared algorithm performs significantly differently from the proposed algorithm if the corresponding adjusted p-value is larger than 0.05, otherwise, the two algorithms have similar performance on multiple problems. Note that, all the LEO-based EC algorithms are named as L-EC where EC is the original name of the algorithm. To quantify the performance improvements, the effect size is computed based on Cohen's d statistic [57] and reported in this paper. Similar to existing work [58], the effect size is regarded as: small if $0.2 \leqslant d < 0.5$; medium if $0.5 \leqslant d < 0.8$, or large if $0.8 < d$, which are denoted with symbols "s", "m", or "l", respectively.

B. Comparisons on Single-objective Optimization Problems

Table I provides the comparison results among LEO-based algorithms and their original versions on single-objective optimization problems. In Table I, the LEO is very effective in improving the optimization efficiency of EC algorithms. As can be seen in Table I, L-PSO, L-DE, and L-NL-SHADE-RSP can significantly outperform PSO, DE, and NL-SHADE-RSP on 19, 10, and 7 problems, and only have worse results on 1, 2, and 0 problems, respectively. Moreover, although NL-SHADE-RSP is the champion algorithm in the CEC competitions with these test problems, the L-NL-SHADE-RSP can still obtain better optimization results and update the optimization result records obtained by NL-SHADE-RSP on 7 problems, i.e., F04(10D), F07(10D), F10(10D), F04(20D), F06(20D), F07(20D), and F20(10D). This suggests the great effectiveness and practical potential of LEO. Overall, the LEO is promising for improving EC algorithms on single-objective optimization problems.

TABLE II
COMPARISONS OF IGD RESULTS BETWEEN LEO-BASED ALGORITHMS AND THE ORIGINAL ALGORITHMS ON MULTI-OBJECTIVE OPTIMIZATION PROBLEMS

Prob.	# obj.	L-NSGA-II	NSGA-II	L-MOEA/D	MOEA/D	L-RVEA	RVEA
DTLZ1	5	2.33E+00±1.46E+00	**1.28E+00±7.33E-01(-)(l)**	**2.34E-01±3.63E-01**	2.88E-01±3.67E-01(+)(s)	**4.63E-01±2.75E-01**	4.69E-01±3.06E-01(≈)(s)
	10	**2.12E+01±6.97E+00**	2.30E+01±9.76E+00(+)(m)	3.39E-01±3.18E-01	**1.03E-01±2.23E-03(-)(l)**	4.96E-01±4.33E-01	**1.95E-01±9.76E-02(-)(l)**
	15	**2.30E+01±1.06E+01**	3.27E+01±1.47E+01(+)(l)	**3.19E-01±1.82E-01**	3.92E-01±3.14E-01(+)(m)	**6.10E-01±4.77E-01**	7.15E-01±8.64E-01(≈)(s)
	20	**2.66E+01±1.31E+01**	3.39E+01±1.43E+01(+)(l)	**3.72E-01±1.90E-01**	4.79E-01±3.66E-01(+)(m)	**8.20E-01±6.15E-01**	9.75E-01±8.96E-01(+)(m)
DTLZ2	5	**2.54E-01±1.30E-02**	2.83E-01±1.37E-02(+)(l)	**2.13E-01±5.04E-04**	2.13E-01±5.35E-04(≈)(s)	**2.12E-01±8.15E-04**	2.14E-01±5.52E-04(≈)(l)
	10	1.44E+00±1.85E-01	**1.37E+00±1.36E-01(-)(m)**	5.36E-01±5.05E-02	**4.13E-01±2.89E-03(-)(l)**	5.33E-01±5.97E-02	**4.25E-01±1.59E-03(-)(l)**
	15	1.51E+00±1.57E-01	**1.40E+00±9.42E-02(-)(l)**	**1.10E+00±8.98E-02**	1.11E+00±9.05E-02(≈)(s)	**9.22E-01±1.31E-01**	9.56E-01±1.08E-01(+)(m)
	20	1.54E+00±1.24E-01	**1.51E+00±9.12E-02(≈)(s)**	**8.93E-01±2.02E-02**	9.00E-01±7.49E-02(≈)(s)	**1.00E+00±1.00E-01**	1.01E+00±1.04E-01(≈)(s)
DTLZ3	5	**6.13E+01±1.10E+01**	3.67E+01±1.57E+01(-)(l)	**1.17E+01±8.53E+00**	1.93E+01±1.14E+0(+)(l)	**1.24E+01±5.93E+00**	1.44E+01±6.82E+00(+)(m)
	10	**7.84E+02±1.40E+02**	7.85E+02±1.76E+02(≈)(s)	9.42E+00±5.75E+00	**1.40E+00±1.35E+00(-)(l)**	1.29E+01±5.38E+00	**9.39E+00±4.08E+00(-)(l)**
	15	**6.77E+02±1.75E+02**	7.66E+02±1.78E+02(+)(l)	1.10E+01±8.23E+00	**1.00E+01±6.62E+00(≈)(s)**	2.21E+01±1.27E+01	**2.10E+01±1.30E+01(-)(s)**
	20	**6.65E+02±2.10E+02**	7.35E+02±1.36E+02(+)(l)	1.43E+01±7.74E+00	**1.28E+01±6.90E+00(-)(s)**	**2.44E+01±1.37E+01**	2.63E+01±9.97E+00(+)(s)
DTLZ4	5	**2.61E-01±2.52E-02**	2.80E-0±1.26E-02(+)(l)	**6.42E-01±2.29E-01**	6.55E-01±2.04E-01(+)(s)	**2.22E-01±3.89E-02**	2.72E-01±1.12E-01(+)(l)
	10	**9.56E-01±8.38E-02**	1.40E+00±1.10E-01(+)(l)	9.20E-01±1.33E-01	**7.33E-01±1.09E-01(-)(l)**	5.92E-01±5.81E-02	**4.43E-01±2.67E-03(-)(l)**
	15	**1.13E+00±7.83E-02**	1.44E+00±1.11E-01(+)(l)	**1.10E+00±1.38E-01**	1.11E+00±1.32E-01(≈)(s)	**8.82E-01±8.40E-02**	8.95E-01±7.39E-02(≈)(s)
	20	**1.22E+00±7.30E-02**	1.46E+00±8.17E-02(+)(l)	**1.16E+00±1.27E-01**	1.19E+00±1.21E-01(≈)(m)	**8.87E-01±5.84E-02**	9.01E-01±5.89E-02(≈)(m)
DTLZ5	5	1.62E-01±4.20E-02	**1.39E-01±3.34E-02(-)(l)**	**2.16E-02±1.60E-03**	2.22E-02±1.73E-03(≈)(m)	**2.12E-01±6.63E-02**	2.16E-01±7.93E-02(≈)(s)
	10	3.36E-01±9.32E-02	**2.58E-01±7.55E-02(-)(l)**	7.71E-02±8.47E-03	**1.87E-02±3.19E-04(-)(l)**	4.05E-01±1.63E-01	**3.85E-01±7.49E-02(≈)(s)**
	15	**4.74E-01±1.85E-01**	6.26E-01±1.61E-01(+)(l)	**3.73E-01±1.96E-01**	3.74E-01±1.89E-01(≈)(s)	**6.59E-01±1.25E-01**	6.65E-01±1.31E-01(≈)(s)
	20	**5.36E-01±2.23E-01**	6.46E-01±2.26E-01(+)(m)	2.65E-01±1.10E-01	**2.37E-01±5.54E-03(-)(m)**	6.40E-01±1.44E-01	**6.35E-01±1.44E-01(≈)(s)**
DTLZ6	5	**4.53E+00±1.16E+00**	4.88E+00±7.50E-01(+)(m)	**2.12E-01±3.77E-01**	2.42E-01±4.53E-01(+)(s)	3.59E-01±2.25E-01	**2.95E-01±2.07E-01(≈)(m)**
	10	7.50E+00±8.91E-01	**6.03E+00±7.91E-01(-)(l)**	**1.95E-01±3.81E-01**	2.45E-01±7.14E-01(+)(s)	5.20E-01±4.00E-01	**3.50E-01±2.21E-01(≈)(l)**
	15	7.85E+00±6.35E-01	**7.78E+00±7.45E-01(≈)(s)**	4.91E-01±2.28E-01	**4.59E-01±2.26E-01(-)(s)**	6.77E-01±3.74E-01	**6.64E-01±2.74E-01(≈)(s)**
	20	8.00E+00±7.11E-01	**7.88E+00±5.91E-01(≈)(s)**	**2.38E-01±6.53E-04**	2.38E-01±1.56E-03(≈)(s)	**9.37E-01±7.37E-01**	1.07E+00±7.31E-01(+)(s)
DTLZ7	5	6.91E-01±8.87E-02	**6.74E-01±5.62E-02(≈)(m)**	1.07E+00±1.91E-01	**1.04E+00±1.36E-01(≈)(s)**	**6.04E-01±6.74E-02**	6.08E-01±6.75E-02 (≈)(s)
	10	9.08E+00±2.81E+00	**6.75E+00±2.12E+00(-)(l)**	**2.12E+00±3.71E-01**	2.36E+00±5.08E-01(+)(l)	1.68E+00±1.49E-01	**1.67E+00±2.58E-01(≈)(s)**
	15	**2.17E+01±3.82E+00**	2.45E+01±4.27E+00(+)(l)	6.56E+00±8.88E-01	**6.54E+00±1.12E+00(≈)(s)**	**2.53E+00±3.33E-01**	2.61E+00±3.76E-01(≈)(m)
	20	**3.43E+01±5.26E+00**	3.45E+01±7.29E+00(≈)(m)	**3.38E+00±6.65E-01**	3.59E+00±5.59E-01(+)(m)	**3.25E+00±8.65E-01**	3.27E+00±8.40E-01(≈)(s)
+/≈/-		NA	14/6/8	NA	9/11/8	NA	6/17/5
Adjusted *p*-value		NA	0.0325	NA	0.4227	NA	0.1967

C. Comparisons on Multi-/Many-objective Optimization Problems

Table II gives the comparison results among LEO-based algorithms and their original versions on multi-objective optimization problems. The results indicate that the LEO is also very effective in improving the optimization efficiency of EC algorithms on multi-/many-objective problems. As shown in Table II, L-NSGA-II, L-MOEA/D, and L-RVEA can obtain significantly better IGD results than NSGA-II, MOEA/D, and RVEA on 14, 9, and 6 problems, and have worse results only on 8, 8, and 5 problems, respectively. Moreover, Table II shows that the NSGA-II has more improvement than the MOEA/D and RVEA when integrated with LEO. This may be due to that when the number of objectives increases, the number of the non-dominated solution increases dramatically and the NSGA-II (i.e., the non-dominated selection-based algorithm) is difficult to conduct environmental selection to select high-quality solutions into the next generation for better evolution. However, the LEO can directly evolve solutions toward the promising region, and therefore can improve the NSGA-II significantly. Overall, the results have shown the effectiveness of LEO on multi-/many-objective optimization problems.

D. Comparisons on High-dimensional Optimization Problems

To examine the proposed LEO on more higher-dimensional problems, this part compares L-PSO, L-DE, and L-NL-SHADE-RSP with PSO, DE, and NL-SHADE-RSP, respectively, on optimization problems with 100 dimensions. The number of fitness evaluations for different algorithms is set the same as 1×10^6. Table III gives the comparison results between LEO-based algorithms and their original versions. As can be seen in Table III, L-PSO, L-DE, and L-NL-SHADE-RSP can significantly outperform PSO, DE, and NL-SHADE-RSP on 10, 8, and 6 problems with 100 dimensions, respectively, but have worse results on none of the problems. Moreover, according to the p-values, L-PSO, L-DE, and L-NL-SHADE-RSP have significantly better overall performance than PSO, DE, and NL-SHADE-RSP on the 10 problems. Therefore, the experimental results further verify the superiority of the LEO on higher-dimensional problems.

E. Time Efficiency Analysis of the LEO

The above results have experimentally verified the

TABLE III
COMPARISONS BETWEEN LEO-BASED ALGORITHMS AND THE ORIGINAL ALGORITHMS ON PROBLEMS WITH 100 DIMENSIONS

Problem	L-PSO	PSO	L-DE	DE	L-NL-SHADE-RSP	NL-SHADE-RSP
F01	**0±0**	7.33E+09±7.75E+08(+)(l)	**0±0**	2.54E-07±1.65E-07(+)(l)	**0±0**	**0±0(≈)(s)**
F02	**1.17E+03±3.65E+02**	2.74E+04±1.03E+03(+)(l)	**3.51E+00±4.70E-01**	2.55E+04±1.75E+03(+)(l)	2.85E-06±1.69E-07	**3.15E-07±3.82E-08(≈)(l)**
F03	**2.00E+02±4.52E+01**	1.32E+03±4.45E+01(+)(l)	**1.66E+02±2.20E+01**	8.49E+02±4.64E+01(+)(l)	**1.78E-07±4.49E-08**	2.37E-07±9.57E-08(≈)(l)
F04	**6.08E+01±1.32E+01**	1.98E+02±2.71E+01(+)(l)	**2.51E+01±9.67E-01**	7.50E+01±3.05E+00(+)(l)	**4.34E-01±5.57E-01**	2.14E+00±4.77E+00(+)(l)
F05	**2.16E+03±5.63E+02**	1.28E+08±3.60E+07(+)(l)	**2.64E+02±1.32E+02**	2.67E+03±1.24E+03(≈)(l)	**4.17E+01±9.53E+00**	1.48E+02±4.49E+02(+)(m)
F06	**2.56E+02±1.77E+02**	6.99E+03±5.03E+02(+)(l)	**1.79E+01±4.80E+00**	4.00E+03±1.05E+03(+)(l)	**4.98E+00±1.47E+00**	2.24E+01±2.67E+00(+)(l)
F07	**1.34E+03±2.98E+02**	1.09E+08±3.40E+07(+)(l)	**1.36E+02±1.80E+02**	6.71E+02±6.71E+02(+)(l)	**5.12E-02±8.99E-03**	9.14E+01±1.57E+01(+)(l)
F08	**1.39E+03±8.20E+01**	1.09E+04±5.53E+02(+)(l)	**1.25E+02±2.57E+01**	2.66E+02±7.12E+01(+)(l)	**1.39E+00±2.20E+00**	3.23E+01±6.95E+02(+)(s)
F09	**1.52E+02±3.46E+01**	1.05E+03±4.15E+01(+)(l)	**1.88E+00±4.60E+00**	7.27E+00±1.40E+01(+)(l)	**1.52E-02±3.46E-01**	1.93E-02±3.51E-04(≈)(s)
F10	**6.66E+02±4.67E+01**	1.05E+03±8.74E+01(+)(l)	**5.24E+02±2.66E+01**	5.44E+02±3.51E+01(≈)(l)	**2.67E-02±3.67E-01**	3.86E-02±1.41E-01(+)(s)
+/≈/-	NA	10/0/0	NA	8/2/0	NA	6/4/0
Adjusted *p*-value	NA	0.0016	NA	0.0016	NA	0.0269

effectiveness of LEO in improving the optimization results. Therefore, this part further investigates the time efficiency of the LEO for solving optimization problems. To study the time efficiency of LEO, this paper proposes a novel indicator metric named *e*rror reduction versus *t*ime increase *r*ate (i.e., ETR), which can measure the benefit of additional time cost when using LEO (i.e., can be treated as a measurement of cost performance). Mathematically, given two algorithms A and B, where A requires more computational time cost than B but has better performance than B, the ETR of A over B is defined as

$$\mathrm{ETR}(A,B)=\frac{\mathrm{ERR}(A,B)}{\mathrm{TIR}(A,B)} \tag{12}$$

where ERR(A, B) is the optimization *e*rror *r*eduction *r*ate of A over B, and the TIR(A, B) is the *t*ime cost *i*ncrease *r*ate of A over B. For a minimization optimization problem, the ERR(A, B) can be computed as

$$\mathrm{ERR}(A,B)=\begin{cases}\dfrac{E_B-E_A}{E_B}\times 100\%, & \text{if } E_B>E_A\\ 0, & \text{otherwise}\end{cases} \tag{13}$$

where E_A and E_B are the optimization error of A and B, respectively. If ERR(A, B) is 0, then A has no error reduction over B, i.e., no accuracy improvement when compared with B. Moreover, the TIR(A, B) can be calculated as:

$$\mathrm{TIR}(A,B)=\frac{T_A-T_B}{T_B}\times 100\% \tag{14}$$

where T_B and T_A represent the computational time cost of A and B, respectively. Therefore, ETR(A, B) can measure how many percentages of optimization error reduction A can obtain over B for each increased percentage of time cost.

Based on the ETR, this part studies the time efficiency of LEO by analyzing the ETR of L-PSO over PSO in different situations. To be specific, this part analyzes the ETR(L-PSO, PSO) on F01 to F10 at 10D with five computational time cost settings in fitness evaluation (i.e., the time cost of each fitness evaluation in the five scenarios requires 0.1, 0.5, 1, 1.5, and 2 ms, respectively). The five settings of time cost are implemented by using a time-delay operation in the fitness evaluation. For brevity and simplicity, the following contents simply refer ETR as ETR(L-PSO, PSO). The ETR results are plotted in Fig. 5, and the running time of L-PSO and PSO are also given in Table S.III.

From Fig. 5, we can have two important observations. First, the ETR increases nearly linearly as the time cost of fitness evaluation increases on most problems. This suggests that the more expensive the fitness evaluation cost is, the more efficient the LEO method will be (one percentage increase in time cost can bring in more percentages of error reduction). In fact, the computational time cost of one fitness evaluation in various real-world applications can be very expensive, e.g., several hours to several days [4], which are much more expensive than the highest settings in this experiment (i.e., 2 ms). Second, when the time cost of each evaluation is larger than 0.5 ms, the ETR is very remarkable, and when the time cost of each evaluation is 2 ms, the ETR is about 14 on most problems, i.e., every increased percentage of time cost when using LEO can result in about 14 percentage of optimization error reduction. These results have shown the great potential and advantage of LEO. In addition, the running time given in Table S.III shows that the L-PSO just requires a small percentage of additional running time when compared with PSO when the time cost of fitness evaluation is larger than 1 ms. Based on the above, it can be concluded that the additional time cost for using LEO is considerably economical and deserved, and the LEO is very efficient in turning the computational cost into the performance improvement of the algorithm.

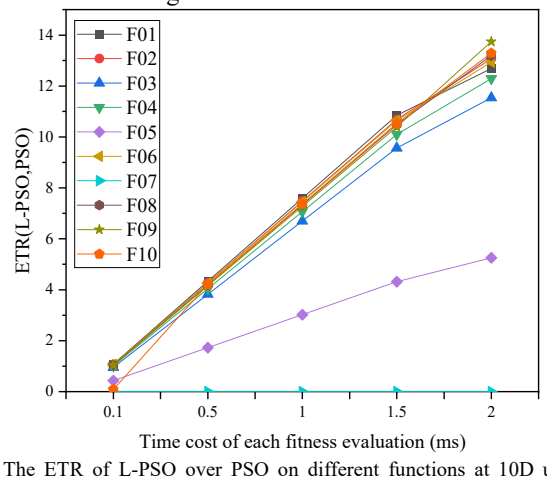

Fig. 5. The ETR of L-PSO over PSO on different functions at 10D under different time costs in each fitness evaluation, where the larger ETR value means the better efficiency.

In addition, to provide a fair comparison from the aspect of the same running time, this part further compares the LEO-PSO, LEO-DE, and LEO-NL-SHADE-RSP with their original version under the same running time. To do this, the running time of all algorithms are set the same as 240 seconds, i.e., 4 minutes. Similar to the setting for Fig. 5, the time cost of each fitness evaluation is set as 1 ms controlled by a time-delay operation. The comparison results are given in Table S.II of the supplementary material. As shown in Table S.II, when using the same running time, L-PSO, L-DE, and L-NL-SHADE-RSP can significantly outperform PSO, DE, and NL-SHADE-RSP on 9, 5, and 3 problems, and only have worse results on 1, 1, and 0 problems, respectively. Moreover, according to the adjusted *p*-value, the L-PSO and L-DE are significantly better than PSO and DE over the multiple problems, respectively. Although the L-NL-SHADE-RSP and NL-SHADE-RSP have similar results over multiple problems based on the adjusted *p*-value, the L-NL-SHADE-RSP can still outperform NL-SHADE-RSP on F04, F07, and F10. Based on the above, the LEO-based algorithms are still efficient when compared with the baseline algorithms with the same running time.

F. Effectiveness of Learning-aided Evolutionary Operator

This paper proposes to use the learning-aided evolutionary operator with ANN to generate promising individuals, rather than directly using the output solutions of ANN to make up the new population. Therefore, this part studies whether using the output solutions of ANN directly can work. To do this, a new L-PSO variant named ANN-PSO is developed, where during the learning-aided process (refer to line 12 of Algorithm 1), the learning-aided evolutionary operator is not used while the output solution of ANN is used directly as the new individual. That is, the Eq. (6) and Eq. (7) are not used.

The comparison results between L-PSO and ANN-PSO are given in Table IV. As can be seen, according to the adjusted *p*-value, the L-PSO is significantly different from the ANN-PSO on the multiple test problems. Moreover, the L-PSO greatly outperforms ANN-PSO on 8 of the 10 problems. In addition, the L-PSO obtains the best results on all the 10 problems. These results show the great superiority of L-PSO over ANN-PSO. This may be due to that although an ANN is used to learn how to generate promising individuals in the ANN-PSO, the ANN is misled by the local optima found in the early evolutionary stage and then produce new individuals around local optima, which results in the poor performance of the ANN-PSO. Differently, the learning-aided evolutionary operator used in L-PSO can generate new individuals based on the individuals produced by ANN and the individuals in current populations, which has more diversity and can enhance the optimization results. Overall, the learning-aided evolutionary operator is effective for enhancing the algorithm results.

TABLE IV
COMPARISONS BETWEEN L-PSO AND ANN-PSO

Problem	L-PSO	ANN-PSO
F01(10D)	**0±0**	**0±0 (≈)(s)**
F02(10D)	**4.10E+00±2.16E+01**	1.17E+02±1.46E+02(+)(l)
F03(10D)	**8.41E+00±4.72E+00**	1.12E+01±4.27E+00(+)(l)
F04(10D)	**4.06E-01±5.93E-02**	9.41E-01±5.04E-01(+)(l)
F05(10D)	**5.19E+01±5.91E+01**	8.67E+01±9.09E+01(+)(m)
F06(10D)	**3.68E-01±2.17E-01**	6.99E+00±2.16E+01(+)(m)
F07(10D)	**3.62E+01±5.49E+01**	5.63E+01±8.55E+01(+)(m)
F08(10D)	**0±0**	1.37E+01±1.63E+01(+)(l)
F09(10D)	**0±0**	5.12E-01±1.58E+00(+)(m)
F10(10D)	**4.81E+01±1.20E-01**	**4.81E+01±2.95E-01(≈)(s)**
Number of +/≈/-	NA	8/2/0
Adjusted *p*-value	NA	0.0044

G. Influence of Learning Probability

To study the influence of *lp* in LEO-based algorithms, this part compares the L-PSO with its variants using different *lp* values. That is, the L-PSO(*lp*=0.5) is compared with L-PSO(*lp*=0.1), L-PSO(*lp*=0.3), L-PSO(*lp*=0.7), L-PSO(*lp*=0.9). The comparison results are shown in Table V. In general, according to the adjusted *p*-value, the L-PSO(*lp*=0.5) has no significant difference from the rest four variants over the 10 problems. This shows that the proposed LEO is not sensitive to the learning probability. Moreover, L-PSO(*lp*=0.5) can generate significantly better results than L-PSO(*lp*=0.1), L-PSO(*lp*=0.3), and L-PSO(*lp*=0.9) on 4, 5, and 3 problems, and produce worse results only on 2, 0, and 3 problems, respectively. Moreover, when compared with L-PSO(*lp*=0.7), L-PSO(*lp*=0.5) has a similar performance in general, e.g., has similar results on 8 problems. The above results may be due to that both too small *lp* (e.g., *lp*=0.1) and too large *lp* (e.g., *lp*=0.9) will deteriorate the algorithm performance and an appropriate *lp* (e.g., *lp*=0.5 or 0.7) can make a good balance between the traditional evolutionary process and learning-aided evolutionary process to obtain better results. Moreover, the experimental results also suggest that *lp*=0.5 can be a good setting, and therefore *lp*=0.5 is recommended in this paper.

H. Influence of Archive Size

To investigate the influence of *arch_size* in LEO-based algorithms, this part compares the L-PSO with its variants using different *arch_size*. To be specifically, the L-PSO(*arch_size*=100) is compared with L-PSO(*arch_size*=50), L-PSO(*arch_size*=200), L-PSO(*arch_size*=300), and L-PSO(*arch_size*=400). The comparison results can be seen in Table VI. In general, according to the adjusted *p*-value, the different L-PSO variants generate similar optimization results over the ten test problems. To be more specifically, L-PSO(*arch_size*=100) produce significantly better results than L-PSO(*arch_size*=50), L-PSO(*arch_size*=200), L-PSO(*arch_size*=300), and L-PSO(*arch_size*=400) on 3, 1, 1, and 2 problems, and worse results on 1, 1, 2, and 2 problems, respectively. Moreover, the L-PSO(*arch_size*=100) performs similar to L-PSO(*arch_size*=50), L-PSO(*arch_size*=200), L-PSO(*arch_size*=300), and L-PSO(*arch_size*=400) on 6, 8, 7, and 6 problems, respectively. The above results indicate that the L-PSO is not that sensitive to the settings of *arch_size*, and *arch_size*=100 can be enough for learning the evolution knowledge in PSO. In addition, as a larger *arch_size* (i.e., more successful solution pairs for training) requires more computation cost, the *arch_size*=100 can be adopted to obtain a better balance between the optimization results and computational burden, and therefore is recommended herein.

TABLE V
COMPARISONS AMONG L-PSO VARIANTS WITH DIFFERENT LEARNING PROBABILITIES

Problem	L-PSO(lp=0.5)	L-PSO(lp=0.1)	L-PSO(lp=0.3)	L-PSO(lp=0.7)	L-PSO(lp=0.9)
F01(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F02(10D)	4.10E+00±2.16E+01	3.46E+01±6.82E+01 (+)(l)	8.66E+00±2.99E+01 (+)(s)	1.30E-01±8.35E-02 (-)(m)	**1.25E-01±1.30E-01 (-)(m)**
F03(10D)	8.41E+00±4.72E+00	**8.10E+00±4.97E+00 (-)(s)**	8.43E+00±4.73E+00 (≈)(s)	9.30E+00±4.23E+00 (+)(s)	1.05E+01±2.88E+00 (+)(l)
F04(10D)	4.06E-01±5.93E-02	6.91E-01±1.39E-01 (+)(l)	4.86E-01±8.03E-02 (+)(l)	3.57E-01±5.84E-02 (≈)(l)	**3.38E-01±5.46E-02 (-)(l)**
F05(10D)	5.19E+01±5.91E+01	6.21E+01±7.53E+01 (+)(s)	6.16E+01±6.72E+01 (+)(s)	5.33E+01±5.72E+01 (≈)(s)	**3.40E+01±5.35E+01 (-)(m)**
F06(10D)	3.68E-01±2.17E-01	9.76E-01±2.10E+00 (+)(m)	3.94E-01±2.36E-01 (+)(s)	**3.61E-01±1.75E-01 (≈)(s)**	4.28E-01±2.50E-01 (+)(m)
F07(10D)	3.62E+01±5.49E+01	**2.28E+01±4.41E+01(-)(m)**	4.01E+01±6.33E+01 (+)(s)	3.78E+01±5.97E+01 (≈)(s)	4.05E+00±2.16E+01 (+)(l)
F08(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F09(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F10(10D)	**4.81E+01±1.20E-01**	**4.81E+01±1.41E-01 (≈)(s)**	**4.81E+01±1.97E-01 (≈)(s)**	**4.81E+01±1.51E-01 (≈)(s)**	4.82E+01±3.28E-01 (≈)(m)
Number of +/≈/-	NA	4/4/2	5/5/0	1/8/1	3/4/3
Adjusted p-value	NA	0.3222	0.3222	0.8875	0.9436

TABLE VI
COMPARISONS AMONG L-PSO VARIANTS WITH DIFFERENT ARCHIVE SIZES FOR COLLECTING TRAINING DATA

Problem	L-PSO ($arch\_size$=100)	L-PSO ($arch\_size$=50)	L-PSO ($arch\_size$=200)	L-PSO ($arch\_size$=300)	L-PSO ($arch\_size$=400)
F01(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F02(10D)	**4.10E+00±2.16E+01**	8.60E+00±4.56E+01 (+)(s)	1.06E+01±5.70E+01 (+)(s)	1.05E+01±5.70E+01 (+)(s)	8.49E+00±4.56E+01 (+)(s)
F03(10D)	**8.41E+00±4.72E+00**	8.78E+00±4.47E+00 (+)(s)	8.48E+00±4.76E+00 (≈)(s)	8.50E+00±4.77E+00 (≈)(s)	8.45E+00±4.74E+00 (≈)(s)
F04(10D)	4.06E-01±5.93E-02	4.06E-01±5.22E-02 (≈)(s)	**3.97E-01±7.60E-02 (≈)(s)**	4.08E-01±8.04E-02 (≈)(s)	4.04E-01±6.29E-02 (≈)(s)
F05(10D)	5.19E+01±5.91E+01	5.18E+01±6.03E+01 (≈)(s)	5.21E+01±6.69E+01 (≈)(s)	5.10E+01±6.45E+01 (≈)(s)	**4.74E+01±5.92E+01 (-)(s)**
F06(10D)	3.68E-01±2.17E-01	4.24E-01±2.56E-01 (+)(m)	3.49E-01±2.33E-01 (≈)(s)	**3.27E-01±1.89E-01 (-)(m)**	3.82E-01±2.52E-01 (+)(s)
F07(10D)	3.62E+01±5.49E+01	**2.44E+01±4.80E+01(-)(m)**	3.16E+01±5.92E+01 (-)(s)	**2.44E+01±4.79E+01(-)(m)**	2.84E+01±5.07E+01 (-)(s)
F08(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F09(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F10(10D)	**4.81E+01±1.20E-01**	**4.81E+01±1.08E-01 (≈)(s)**	**4.81E+01±9.43E-02 (≈)(s)**	**4.81E+01±1.06E-01 (≈)(s)**	**4.81E+01±1.06E-01 (≈)(s)**
Number of +/≈/-	NA	3/6/1	1/8/1	1/7/2	2/6/2
Adjusted p-value	NA	0.6206	0.7237	1.0000	0.6206

I. Influence of Learning Parameters

This part studies the influence of two learning parameters, i.e., α and β. To begin with, to study the influence of α in LEO-based algorithms, this part compares the L-PSO with its variants with different α values. That is, the L-PSO(α=0.5) is compared with L-PSO(α=0.1), L-PSO(α=0.3), L-PSO(α=0.7), and L-PSO(α=0.9). The comparison results are shown in Table VII. In general, according to the adjusted p-value, the L-PSO(α=0.5) has no significant difference from the other four variants over the 10 problems. Moreover, the L-PSO(α=0.5) performs similarly to L-PSO(α=0.1), L-PSO(α=0.3), L-PSO(α=0.7), and L-PSO(α=0.9) on 8, 9, 9, and 10 problems, respectively. This shows that the LEO is not sensitive to the setting of α. Moreover, as L-PSO(α=0.5) can generate the best results on 6 problems (as marked in **boldface**) among the five algorithms, which is the most, the α=0.5 is used in this paper.

Moreover, this part also studies the influence of β by comparing the L-PSO with its variants with different β values. That is, the L-PSO(β=0.9) is compared with L-PSO(β=0.1), L-PSO(β=0.3), L-PSO(β=0.5), and L-PSO(β=0.7). The comparison results are shown in Table VIII. As can be seen, the L-PSO(β=0.9) significantly outperforms L-PSO(β=0.1) and L-PSO(β=0.3) on 3 and 5 problems, and performs similarly to L-PSO(β=0.5) and L-PSO(β=0.7) on 9 and 8 problems. This suggests that larger β can yield better results. This may be due to that a larger β encourages the individuals to learn more dimensions from the individual produced by ANN, which enhance the optimization efficiency. In addition, according to the adjusted p-value, the L-PSO(β=0.9) has no significant difference with L-PSO(β=0.1), L-PSO(β=0.5), and L-PSO(β=0.7) over the multiple problems, and only have significant difference with L-PSO(β=0.3). Therefore, the L-PSO is not that sensitive to the β value.

J. Influence of ANN Hyperparameter

This part studies the influence of the ANN hyperparameter. The L-PSO is compared with its variants using different numbers of hidden neurons. That is, the L-PSO(N_h=3×D) is compared with L-PSO(N_h=1×D), L-PSO(N_h=2×D), L-PSO(N_h=4×D), and L-PSO(N_h=5×D), where N_h denotes the number of hidden neurons and D is the problem dimension. The comparison results are shown in Table S.IV of the supplementary material. As can be seen in Table S.IV, the L-PSO(N_h=3×D) performs similarly to L-PSO(N_h=1×D), L-PSO(N_h=2×D), L-PSO(N_h=4×D), and L-PSO(N_h=5×D) on 10, 9, 10, and 10 problems, respectively. Moreover, according to the adjusted p-value, the L-PSO(N_h=3×D) performs similarly to the rest four variants. That is, the results produced by L-PSO variants with different numbers of hidden neurons are very similar. This may be due to the fact that the ANN is a powerful learning system and can still learn the evolution knowledge efficiently when using a slightly different hyperparameter.

TABLE VII
COMPARISONS AMONG L-PSO VARIANTS WITH DIFFERENT α VALUES

Problem	L-PSO(α=0.5)	L-PSO(α=0.1)	L-PSO(α=0.3)	L-PSO(α=0.7)	L-PSO(α=0.9)
F01(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F02(10D)	4.10E+00±2.16E+01	4.22E+00±2.16E+01(≈)(s)	**3.46E+00±9.47E+00(≈)(s)**	4.20E+00±1.33E+01(≈)(s)	4.10E+00±2.16E+01(≈)(s)
F03(10D)	8.41E+00±4.72E+00	8.87E+00±4.52E+00(≈)(s)	8.66E+00±4.71E+00(≈)(s)	**8.35E+00±5.13E+00(≈)(s)**	8.41E+00±4.72E+00(≈)(s)
F04(10D)	**4.06E-01±5.93E-02**	6.00E-01±1.46E-01(+)(l)	4.49E-01±7.69E-02(≈)(l)	4.14E-01±5.74E-02(≈)(s)	**4.06E-01±5.93E-02(≈)(s)**
F05(10D)	**5.19E+01±5.91E+01**	6.88E+01±7.77E+01(+)(m)	6.75E+01±8.55E+01(+)(m)	6.44E+01±5.24E+01(+)(m)	**5.19E+01±5.91E+01(≈)(s)**
F06(10D)	3.68E-01±2.17E-01	4.10E-01±2.88E-01(≈)(s)	3.91E-01±2.02E-01(≈)(s)	**3.55E-01±1.43E-01(≈)(s)**	3.68E-01±2.17E-01(≈)(s)
F07(10D)	3.62E+01±5.49E+01	3.24E+01±5.30E+01(≈)(s)	3.45E+01±5.45E+01(≈)(s)	**3.20E+01±3.61E+01(≈)(s)**	3.62E+01±5.49E+01(≈)(s)
F08(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F09(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F10(10D)	**4.81E+01±1.20E-01**	4.82E+01±1.74E-01(≈)(m)	4.82E+01±2.70E-01(≈)(m)	**4.81E+01±1.26E-01(≈)(s)**	**4.81E+01±1.20E-01(≈)(s)**
Number of +/≈/-	NA	2/8/0	1/9/0	1/9/0	0/10/0
Adjusted *p*-value	NA	0.1612	0.8665	1.0000	1.0000

TABLE VIII
COMPARISONS AMONG L-PSO VARIANTS WITH DIFFERENT β VALUES

Problem	L-PSO(β=0.9)	L-PSO(β=0.1)	L-PSO(β=0.3)	L-PSO(β=0.5)	L-PSO(β=0.7)
F01(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F02(10D)	**4.10E+00±2.16E+01**	3.46E+01±6.82E+01(+)(l)	4.37E+01±7.31E+01(+)(l)	8.03E+00±3.00E+01(≈)(s)	1.33E+01±7.82E-02(+)(l)
F03(10D)	8.41E+00±4.72E+00	**8.10E+00±4.97E+00(≈)(s)**	1.10E+01±3.77E+00(+)(l)	9.80E+00±4.29E+00(≈)(m)	9.48E+00±4.32E+00(≈)(m)
F04(10D)	**4.06E-01±5.93E-02**	4.91E-01±1.39E-01(≈)(m)	4.45E-01±1.99E-01(≈)(m)	4.70E-01±9.18E-02(≈)(m)	4.70E-01±9.18E-02(≈)(m)
F05(10D)	**5.19E+01±5.91E+01**	6.21E+01±7.53E+01(+)(s)	1.19E+02±1.30E+02(+)(l)	8.45E+01±8.62E+01(+)(m)	8.45E+01±8.62E+01(+)(m)
F06(10D)	**3.68E-01±2.17E-01**	9.76E-01±2.10E+00(+)(m)	4.58E-01±4.28E-01(+)(s)	4.13E-01±5.30E-01(≈)(s)	4.13E-01±5.30E-01(≈)(s)
F07(10D)	3.62E+01±5.49E+01	**2.28E+01±4.41E+01(-)(m)**	4.36E+01±7.28E+01(+)(s)	3.23E+01±5.30E+01(≈)(s)	3.23E+01±5.30E+01(≈)(s)
F08(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F09(10D)	**0±0**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**	**0±0 (≈)(s)**
F10(10D)	**4.81E+01±1.20E-01**	**4.81E+01±1.41E-01(≈)(s)**	4.82E+01±4.20E-01(≈)(m)	4.82E+01±3.35E-01(≈)(m)	4.82E+01±3.35E-01(≈)(m)
Number of +/≈/-	NA	3/6/1	5/5/0	1/9/0	2/8/0
Adjusted *p*-value	NA	0.3222	0.0015	0.6093	0.6093

Overall, the LEO is robust and is not sensitive to the network hyperparameter, and the original setting (i.e., N_h=3×D) is recommended.

V. CONCLUSION

In this paper, an innovative LEO framework for EC algorithms has been proposed. In the LEO, the ANN is adopted as the learning system to learn the SEPs collected from the evolutionary process, and then the learned evolution knowledge is applied to promote the solution evolution. As a result, the candidate solution can evolve directly toward a more promising region, which is more straightforward and efficient than traditional evolutionary operators.

To evaluate the LEO framework, the performance of some state-of-the-art and champion algorithms are compared with those of their LEO-based variants on both single-objective and multi-/many-objective optimization problems. In these experiments, the LEO-based algorithms achieve better performance than those without LEO. The experimental results show that solutions can not only avoid the dilemma of being unable to extract enough information to enhance evolution but also achieve faster convergence speed via the LEO framework, especially when encountered with high-dimensional problems. Moreover, a novel ETR indicator has been proposed to measure the time efficiency of the LEO, and the simulation experimental results have shown that the additional time cost for using LEO is greatly economical and deserved, especially when the fitness evaluation is very expensive. Therefore, although the concept and core idea of the proposed LEO framework is simple and easy to understand, it is very effective and efficient, which leads to a novel and more efficient paradigm for EC algorithms in solving global optimization problems. Moreover, the ETR indicator is a general metric that can be promoted in the EC community for measuring the cost performance of different EC algorithms.

Nevertheless, there may still have some room for enhancing the proposed LEO framework and LEO-based algorithms. First, the difficulties of complex problems may pose new challenges to the LEO-based algorithms, e.g., multi-objective problems with irregular feasible regions [58], complex Pareto fronts [59], and a large number of objectives [60], which should be well-studied. Second, if the fitness evaluation of the problem is expensive to access, there will be only few-shot solutions that can be evaluated with the real fitness evaluation. In such case, how to efficiently use the limited training data should be well-studied.

Therefore, in the future, we will not only enhance the LEO-based algorithms to well address the above issues, but also hope to extend the idea of knowledge learning to more research aspects of EC and real-world application problems, such as large-scale problems [25]-[63], multimodal problems [64][65], and multi-task problems [66][67]. Moreover, the automatic optimization of the network architecture and parameter of the LEO learning system for different scenarios will be worthy of study [68]. Additionally, the implementation of the LEO with powerful computation techniques, such as

distributed computing [69][70], is also a potential research direction for future work.

REFERENCES

[1] Z. H. Zhan *et al.*, "Matrix-based evolutionary computation," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 6, no. 2, pp. 315-328, April 2022.

[2] Z. H. Zhan, L. Shi, K. C. Tan, and J. Zhang, "A survey on evolutionary computation for complex continuous optimization," *Artif. Intell. Rev.*, vol. 55, pp. 59–110, Jan. 2022.

[3] D. Guirguis *et al.*, "Evolutionary black-box topology optimization: Challenges and promises," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 613-633, Aug. 2020.

[4] J. Y. Li, Z. H. Zhan, and J. Zhang, "Evolutionary computation for expensive optimization: A survey," *Mach. Intell. Res.*, vol. 19, no. 1, pp. 3–23, 2022.

[5] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[6] L. Feng, Q. Shang, Y. Hou, K. Tan, and Y. S. Ong, "Multi-space evolutionary search for large-scale optimization with applications to recommender systems," *IEEE Trans. Artif. Intell.*, 2021, DOI: 10.1109/TAI.2022.3156952.

[7] X. F. Liu, Z. H. Zhan, and J. Zhang, "Resource-aware distributed differential evolution for training expensive neural-network-based controller in power electronic circuit," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 33, no. 11, pp. 6286–6296, Nov. 2022.

[8] A. Ahrari, S. Elsayed, R. Sarker, D. Essam, and C. A. C. Coello, "Static and dynamic multimodal optimization by improved covariance matrix self-adaptation evolution strategy with repelling subpopulations," *IEEE Trans. Evol. Comput.*, 2021, DOI: 10.1109/TEVC.2021.3117116.

[9] Z. G. Chen, Z. H. Zhan, S. Kwong, and J. Zhang, "Evolutionary computation for intelligent transportation in smart cities: A survey," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 83–102, 2022.

[10] Q. Lin, X. Wu, L. Ma, J. Li, M. Gong, and C. A. C. Coello, "An ensemble surrogate-based framework for expensive multiobjective evolutionary optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 4, pp. 631–645, Aug. 2022.

[11] S. C. Liu, Z. G. Chen, Z. H. Zhan, S. W. Jeon, S. Kwong, and J. Zhang, "Many-objective job shop scheduling: A multiple populations for multiple objectives-based genetic algorithm approach," *IEEE Trans. Cybern.*, 2021, DOI: 10.1109/TCYB.2021.3102642.

[12] J. G. Falcón-Cardona, H. Ishibuchi, C. A. Coello Coello, and M. Emmerich, "On the effect of the cooperation of indicator-based multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 681-695, Aug. 2021.

[13] X. Zhang, Z. H. Zhan, W. Fang, P. Qian, and J. Zhang, "Multi population ant colony system with knowledge based local searches for multiobjective supply chain configuration," *IEEE Trans. Evol. Comput.*, vol. 26, no. 3, pp. 512–526, Jun. 2022.

[14] J. H. Holland, *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. Cambridge, MA, USA: MIT Press, 1975.

[15] J. G. Falcón-Cardona, H. Ishibuchi, C. A. Coello Coello, and M. Emmerich, "On the effect of the cooperation of indicator-based multiobjective evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 681-695, Aug. 2021.

[16] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES)," *Evol. Comput.* vol. 11, pp.1-18, 2003.

[17] Y. Bi, B. Xue, and M. Zhang, "Multitask feature learning as multiobjective optimization: A new genetic programming approach to image classification," *IEEE Trans. Cybern.*, 2022, DOI: 10.1109/TCYB.2022.3174519.

[18] Y. Bi, B. Xue, and M. Zhang, "Dual-tree genetic programming for few-shot image classification," *IEEE Trans. Evol. Comput.*, vol. 26, no. 3, pp. 555–569, 2022.

[19] Y. Bi, B. Xue, and M. Zhang, "Multitask feature learning as multiobjective optimization: A new genetic programming approach to image classification," *IEEE Trans. Cybern.*, vol. 26, no. 2, pp. 218–232, 2022.

[20] R. Storn and K. V. Price, "Differential evolution—A simple and efficient heuristic for global optimization over continuous spaces," *J. Glob. Optim.*, vol. 11, no. 4, pp. 341–359, 1997.

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

[22] J. Y. Li, Z. H. Zhan, K. C. Tan, and J. Zhang, "A meta-knowledge transfer-based differential evolution for multitask optimization," *IEEE Trans. Evol. Comput.,* vol. 26, no. 4, pp. 719–734, Aug. 2022.

[23] K. J. Du, J. Y. Li, H. Wang, and J. Zhang, "Multi-objective multi-criteria evolutionary algorithm for multi-objective multi-task optimization," *Complex Intell. Syst.*, 2022, DOI: 10.1007/s40747-022-00650-8.

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

[25] J. R. Jian, Z. G. Chen, Z. H. Zhan, and J. Zhang, "Region encoding helps evolutionary computation evolve faster: A new solution encoding scheme in particle swarm for large-scale optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 779-793, Aug. 2021.

[26] J. Y. Li, Z. H. Zhan, R. D. Liu, C. Wang, S. Kwong, and J. Zhang, "Generation level parallelism for evolutionary computation: A pipeline-based parallel particle swarm optimization," *IEEE Trans. Cybern.*, vol. 51, no. 10, pp. 4848-4859, Oct. 2021.

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

[28] J. Y. Li *et al.*, "A multipopulation multiobjective ant colony system considering travel and prevention costs for vehicle routing in COVID-19-like epidemics," *IEEE Trans. Intell. Transp. Syst.*, 2022, DOI: 10.1109/tits.2022.3180760.

[29] L. Shi, Z. H. Zhan, D. Liang, and J. Zhang, "Memory-based ant colony system approach for multi-source data associated dynamic electric vehicle dispatch optimization," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17491–17505, Oct. 2022.

[30] L. J. Wu *et al.*, "Real environment-aware multisource data-associated cold chain logistics scheduling: A multiple population-based multiobjective ant colony system approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 12, pp. 23613–23627, Dec. 2022.

[31] J. Y. Li, Z. H. Zhan, J. Xu, S. Kwong, and J. Zhang, "Surrogate-assisted hybrid-model estimation of distribution algorithm for mixed-variable hyperparameters optimization in convolutional neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, 2021, DOI: 10.1109/TNNLS.2021.3106399.

[32] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 921–934, 2019.

[33] Y. Sun, G. G. Yen, and Z. Yi, "Evolving unsupervised deep neural networks for learning meaningful representations," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 89–103, 2019.

[34] Z. H. Zhan, J. Y. Li, and J. Zhang, "Evolutionary deep learning: A survey," *Neurocomput.*, vol. 483, pp. 42–58, 2022.

[35] H. Al-Sahaf *et al.*, "A survey on evolutionary machine learning," *J. R. Soc. New Zeal.*, vol. 49, no. 2, pp. 205–228, 2019

[36] E. Galván and P. Mooney, "Neuroevolution in deep neural networks: Current trends and future challenges," *IEEE Trans. Artif. Intell.*, vol. 2, no. 6, pp. 476-493, Dec. 2021.

[37] J. Zhang *et al.*, "Evolutionary computation meets machine learning: A survey," *IEEE Comput. Intell. Mag.*, vol. 6, no. 4, pp. 68–75, 2011.

[38] A. Wagdy *et al.*, "Problem definitions and evaluation criteria for the CEC 2021 special session and competition on single objective bound constrained numerical optimization", *Tech. Rep.*, Nanyang Technological University, Singapore.

[39] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multiobjective optimization test problems," in *Proc. IEEE Congr. Evol. Comput.*, 2002, pp. 825–830.

[40] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[41] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[42] V. Stanovov, S. Akhmedova, and E. Semenkin, "NL-SHADE-RSP algorithm with adaptive archive and selective pressure for cec 2021 numerical optimization," in *Proc. Proc. IEEE Congr. Evol. Comput.*, 2021, pp. 809-816.

[43] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, 1989.

[44] H. Wang, Y. Jin, C. Sun, and J. Doherty, "Offline data-driven evolutionary optimization using selective surrogate ensembles," *IEEE Trans. Evol. Comput.*, vol. 23, no. 2, pp. 203–216, 2018.

[45] J. Y. Li, Z. H. Zhan, C. Wang, H. Jin, and J. Zhang, "Boosting data-driven evolutionary algorithm with localized data generation," *IEEE Trans. Evol.*

Comput., vol. 24, no. 5, pp. 923-937, Oct. 2020.
[46] J. Y. Li, Z. H. Zhan, H. Wang, and J. Zhang, "Data-driven evolutionary algorithm with perturbation-based ensemble surrogates," *IEEE Trans. Cybern.*, vol. 51, no. 8, pp. 3925-3937, Aug. 2021.
[47] X. Ji, Y. Zhang, D. Gong, and X. Sun, "Dual-surrogate assisted cooperative particle swarm optimization for expensive multimodal problems," *IEEE Trans. Evol. Comput.*, 2021, DOI: 10.1109/TEVC.2021.3064835.
[48] G. Li and Q. Zhang, "Multiple penalties and multiple local surrogates for expensive constrained optimization," *IEEE Trans. Evol. Comput.*, 2021, DOI: 10.1109/TEVC.2021.3066606.
[49] L. Feng *et al.*, "Towards faster vehicle routing by transferring knowledge from customer representation," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 2, pp. 952–965, 2022.
[50] L. Zhou, L. Feng, A. Gupta, and Y. S. Ong, "Learnable evolutionary search across heterogeneous problems via kernelized autoencoding," *IEEE Trans. Evol. Comput.*, vol. 25, no. 3, pp. 567–581, 2021.
[51] L. Feng, W. Zhou, W. Liu, Y. S. Ong, and K. C. Tan, "Solving dynamic multiobjective problem via autoencoding evolutionary search," *IEEE Trans. Cybern.*, vol. 52, no. 5, pp. 2649–2662, 2022.
[52] X. F. Liu *et al.*, "Neural network-based information transfer for dynamic optimization," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 31, no. 5, pp. 1557–1570, 2020.
[53] Z. H. Zhan, Z. J. Wang, H. Jin, and J. Zhang, "Adaptive distributed differential evolution," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4633–4647, Nov. 2020.
[54] A. Ghosh, S. Das, A. K. Das, and L. Gao, "Reusing the past difference vectors in differential evolution—A simple but significant improvement," *IEEE Trans. Cybern.*, vol. 50, no. 11, pp. 4821–4834, Nov. 2020.
[55] X. Xia *et al.*, "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4862–4875, Dec. 2020.
[56] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 20, no. 5, pp. 773–791, 2016.
[57] J. Cohen, Statistical Power Analysis for the Behavioral Sciences. New York, NY, USA: Psychology Press, 1988.
[58] S. Bechikh, A. Chaabani, and L. Ben Said, "An efficient chemical reaction optimization algorithm for multiobjective optimization," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2051–2064, Oct. 2015.
[59] M. Elarbi, S. Bechikh, C. A. Coello Coello, M. Makhlouf, and L. Ben Said, "Approximating complex Pareto fronts with predefined normal-boundary intersection directions," *IEEE Trans. Evol. Comput.*, vol. 24, no. 5, pp. 809–823, 2020.
[60] M. Elarbi, S. Bechikh, A. Gupta, L. Ben Said, and Y. Ong, "A new decomposition-based NSGA-II for many-objective optimization," *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 48, no. 7, pp. 1191–1210, 2018.
[61] X. Zhang *et al.*, "Graph-Based Deep Decomposition for Overlapping Large-Scale Optimization Problems," *IEEE Trans. Syst. Man, Cybern. Syst.*, 2022, DOI: 10.1109/TSMC.2022.3212045.
[62] J. Y. Li, Z. H. Zhan, K. C. Tan, and J. Zhang, "Dual differential grouping: A more general decomposition method for large-scale optimization," *IEEE Trans. Cybern.*, 2022, DOI: 10.1109/TCYB.2022.3158391.
[63] Z. J. Wang, J. R. Jian, Z. H. Zhan, Y. Li, S. Kwong, and J. Zhang, "Gene targeting differential evolution: A simple and efficient method for large scale optimization," *IEEE Trans. Evol. Comput.*, 2022, DOI: 10.1109/TEVC.2022.3185665.
[64] Z. G. Chen, Z. H. Zhan, H. Wang, and J. Zhang, "Distributed individuals for multiple peaks: A novel differential evolution for multimodal optimization problems," *IEEE Trans. Evol. Comput.*, vol. 24, no. 4, pp. 708-719, Aug. 2020.
[65] Y. Jiang, Z. H. Zhan, K. C. Tan, and J. Zhang, "Optimizing niching centers for multimodal optimization," *IEEE Trans. Cybern.*, 2022, DOI: 10.1109/TCYB.2021.3125362.
[66] S. H. Wu, Z. H. Zhan, K. C. Tan, and J. Zhang, "Orthogonal transfer for multitask optimization," *IEEE Trans. Evol. Comput.*, 2022, DOI: 10.1109/TEVC.2022.3160196.
[67] Y. Jiang, Z. H. Zhan, K. C. Tan, and J. Zhang, "A bi-objective knowledge transfer framework for evolutionary many-task optimization," *IEEE Trans. Evol. Comput.*, 2022, DOI: 10.1109/TEVC.2022.3210783.
[68] Y. Q. Wang, J. Y. Li, C. H. Chen, J. Zhang, and Z. H. Zhan, "Scale adaptive fitness evaluation-based particle swarm optimization for hyperparameter and architecture optimization in neural networks and deep learning," *CAAI Trans. Intell. Technol.*, 2022, DOI: 10.1049/cit2.12106.
[69] J. Y. Li, K. J. Du, Z. H. Zhan, H. Wang, and J. Zhang, "Distributed differential evolution with adaptive resource allocation," *IEEE Trans. Cybern.*, 2022, DOI: 10.1109/TCYB.2022.3153964.
[70] Y. Guo, J. Y. Li, and Z. H. Zhan, "Efficient hyperparameter optimization for convolution neural networks in deep learning: A distributed particle swarm optimization approach," *Cybern. Syst.*, vol. 52, no. 1, pp. 36–57, 2020.

Zhi-Hui Zhan (Senior Member, IEEE) received the Bachelor's degree and the Ph. D. degree in Computer Science from the Sun Yat-Sen University, Guangzhou China, in 2007 and 2013, respectively.

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

Dr. Zhan was a recipient of the IEEE Computational Intelligence Society (CIS) Outstanding Early Career Award in 2021, the Outstanding Youth Science Foundation from National Natural Science Foundations of China (NSFC) in 2018, and the Wu Wen-Jun Artificial Intelligence Excellent Youth from the Chinese Association for Artificial Intelligence in 2017. His doctoral dissertation was awarded the IEEE CIS Outstanding Ph. D. Dissertation and the China Computer Federation Outstanding Ph. D. Dissertation. He is one of the World's Top 2% Scientists for both Career-Long Impact and Year Impact in Artificial Intelligence and one of the Highly Cited Chinese Researchers in Computer Science. He is currently the Chair of Membership Development Committee in IEEE Guangzhou Section and the Vice-Chair of IEEE CIS Guangzhou Chapter. He is currently an Associate Editor of the *IEEE Transactions on Evolutionary Computation*, the *Neurocomputing*, the *Memetic Computing*, and the *Machine Intelligence Research*.

Jian-Yu Li (Member, IEEE) received the Bachelor's degree and the Ph. D. degree in Computer Science and Technology from the South China University of Technology, China, in 2018 and 2022, respectively.

His research interests mainly include computational intelligence, data-driven optimization, machine learning including deep learning, and their applications in real-world problems, and in environments of distributed computing and big data. He currently serves as a reviewer of the *IEEE Transactions on Evolutionary Computation* and the *Neurocomputing*.

Sam Kwong (Fellow, IEEE) received the Ph.D. degree from the University of Hagen, Germany, in 1996.

He is currently a Chair Professor with the Department of Computer Science, City University of Hong Kong. His research interests include pattern recognition, evolutionary computations, and video analytics.

Prof. Kwong was elevated to an IEEE Fellow for his contributions to optimization techniques for cybernetics and video coding in 2014. He is the President of the IEEE Systems, Man, and Cybernetics (SMC). He was also appointed as an IEEE Distinguished Lecturer of the IEEE SMC Society in March 2017. He is currently an Associate Editor of the *IEEE Transactions on Evolutionary Computation*.

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

He is currently a Korea Brain Pool Fellow Professor with Hanyang University, South Korea. His current research interests include computational intelligence, cloud computing, operations research, and power electronic circuits. He has published over more than 150 IEEE Transactions papers in his research areas.

Dr. Zhang was a recipient of the Changjiang Chair Professor from the Ministry of Education, China, in 2013, The National Science Fund for Distinguished Young Scholars of China in 2011 and the First-Grade Award in Natural Science Research from the Ministry of Education, China, in 2009. He is currently an Associate Editor of the *IEEE Transactions on Evolutionary Computation* and the *IEEE Transactions on Cybernetics*.