Neural Net-Enhanced Competitive Swarm Optimizer for Large-scale Multi-objective Optimization

Lingjie Li, Yongfeng Li, Qiuzhen Lin, *Member, IEEE*, Songbai Liu, Junwei Zhou, Zhong Ming, and Carlos A. Coello Coello, *Fellow, IEEE*

***Abstract*—The competitive swarm optimizer (CSO) classifies swarm particles into loser and winner particles and then uses the winner particles to efficiently guide the search of the loser particles. This approach has very promising performance in solving large-scale multi-objective optimization problems (LMOPs). However, most studies of CSOs ignore the evolution of the winner particles, although their quality is very important for the final optimization performance. Aiming to fill this research gap, this paper proposes a new neural net-enhanced CSO for solving LMOPs, called NN-CSO, which not only guides the loser particles via the original CSO strategy, but also applies our trained neural network (NN) model to evolve winner particles. First, the swarm particles are classified into winner and loser particles by the pairwise competition. Then, the loser particles and winner particles are respectively treated as the input and desired output to train the NN model, which tries to learn promising evolutionary dynamics by driving the loser particles toward the winners. Finally, when model training is complete, the winner particles are evolved by the well-trained NN model, while the loser particles are still guided by the winner particles to maintain the search pattern of CSOs. To evaluate the performance of our designed NN-CSO, several LMOPs with up to 10 objectives and 1000 decision variables are adopted, and the experimental results show that our designed NN model can significantly improve the performance of CSOs and shows some advantages over several state-of-the-art large-scale multi-objective evolutionary algorithms as well as over model-based evolutionary algorithms.**

***Index Terms*—Competitive swarm optimizer, large-scale optimization, multi-objective optimization, neural network.**

I. Introduction

MULTI- objective optimization problems (MOPs) usually contain several conflicting objectives that need to be optimized simultaneously [1], as defined by

$$\begin{aligned} &\text{minimize} \quad F(x) = (f_1(x), \ldots, f_m(x)), \\ &\text{subject to} \quad x \in \Omega \end{aligned} \tag{1}$$

Manuscript received xx. xx. 2023; revised xx. xx. 2023; accepted xx. xx. 2023. This work was supported by the National Natural Science Foundation of China (NSFC) under Grants 61836005 and 62272315; in part by the Guangdong Regional Joint Foundation Key Project under Grant 2022B1515120076; in part by Natural Science Foundation of Guangdong Province under Grant No. 2023A1515011238, and in part by the Shenzhen Science and Technology Program under Grants JCYJ20220531101411027 and JCYJ20190808164211203. Carlos A. Coello Coello gratefully acknowledges support from CONACyT grant no. 2016-01-1920 (Investigación en Fronteras de la Ciencia 2016). (Corresponding Author: *Qiuzhen Lin* and *Zhong Ming*)

L. Li, Y. Li, Q. Lin, S. Liu, and Z. Ming are affiliated with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (email: qiuzhlin@szu.edu.cn).

J. Zhou is affiliated with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, Wuhan, China.

C.A. Coello Coello is affiliated with the Department of Computer Science, CINVESTAV-IPN (Evolutionary Computation Group), Mexico City, 07300, MEXICO. He is also with the Faculty of Excellence of the School of Engineering and Sciences, Tecnologico de Monterrey, Monterrey, N.L., MEXICO.

where $x = (x_1, ..., x_n)$ denotes the n-dimensional decision vector of a solution from the search space Ω and $F(x)$ defines m objective functions. Due to the conflicts that often arise in different objectives, there is not a single optimal solution, but a set of equally optimal solutions termed the Pareto-optimal set (PS) for solving MOPs [2]. The mapping of PS onto the objective space is termed the Pareto-optimal front (PF) [2]. In particular, the problem in Equation (1) is called a large-scale multi-objective optimization problem (LMOP) when the number of decision variables n is no less than 100 [3]. During the past few decades, a number of multi-objective evolutionary algorithms (MOEAs) have been proposed with very effective performance for solving MOPs [4]–[6]. However, experimental results show that most of the existing MOEAs are not efficient when solving LMOPs with a large number of decision variables, due to their weak search abilities [7]. To better solve LMOPs, a number of large-scale MOEAs (LMOEAs) have been designed and most of them can be roughly divided into three categories [3], which are introduced sequentially as follows.

The first kind of LMOEAs applies different variable grouping strategies to divide the decision variables into several groups and then optimizes each group collaboratively [8]–[13]. For example, interdependence variable analysis and control variable analysis methods were designed in MOEA/DVA [8] to classify the decision variables as position variables, distance variables and mixed variables, which are then used to fine-tune the convergence and diversity of the population. To divide the decision variables more generically, an angle-based clustering method was proposed in LMEA [9] to group the decision variables as convergence-related and diversity-related variables. Then, these two different types of variables are optimized alternately by using different search strategies. In addition, a reformulated decision variable analysis was proposed in LERD [10], which reformulates decision variable analysis process into an MOP with binary decision variables and then optimizes each group of decision variables. An adaptive decision variable analysis method was designed in LSMOEA/D [11], which can balance the convergence and diversity by adjusting a dynamic control parameter. In contrast to the above approaches that follow similar classification principles to consider both convergence and diversity, a variable importance-based mechanism was designed in LVIDE [12], which quantifies the importance of variables to the target problem and then allocates more computational resources to variables with higher importance. Similarly, a multipopulation algorithm based on the contribution objective of decision variables was presented in DVCOEA [13], which groups the decision variables according to their contribution objectives

and then optimizes them independently in each subpopulation. However, such LMOEAs highly depend on the adopted grouping methods and suffer from high computational costs when conducting variable analysis [14].

The second category of LMOEAs converts LMOPs into small-scale problems by using problem transformation or dimensionality reduction techniques [15]–[18]. Then, traditional MOEAs can be directly used to search for the optimal solution in a smaller search space. For example, a weighted optimization framework was designed in WOF [15], which optimizes the weight vector for the best objective value of each solution. Then, an efficient framework via problem reformulation was proposed in LSMOF [16], which uses the direction vectors and weight variables to reduce the dimensionality of the target problem. A directed sampling strategy was adopted in LMOEA-DS [17] to generate solutions, which provide promising search directions in the decision space, which speeds up convergence. A pattern mining approach was proposed in PM-MOEA [18] to detect the maximum and minimum candidate sets of the nonzero variables of PS, aiming to limit the dimensionality when producing offspring. However, after problem transformation or dimensionality reduction, some original optimal spaces are not attainable, which may cause these LMOEAs fall into local optima. [19].

Different from the two above types of LMOEAs that reduce the difficulty or the dimensionality of the target LMOP, the most recent LMOEAs contain more efficient search strategies or operators to directly solve LMOPs, and can provide stronger search capabilities [20]–[25]. Particularly, the competitive swarm optimizer (CSO), which is as an improved variant of the particle swarm optimizer, is one of the most representative methods of this type of LMOEAs, which owes its competitiveness in solving LMOPs to its efficient search capability and inherent adaptability [26]. Some related works of CSOs for solving LMOPs will be introduced in detail in Section II-B. Unfortunately, although empirical results have shown that CSO approaches are considered as prominent methods for solving LMOPs [26]–[31], they still suffer from a major challenge, i.e., most existing studies of CSOs pay more attention to the way in which the loser particles learn from the winner particles, but ignore the evolution of the winner particles. In fact, the quality of the winner particles directly determines the effectiveness of CSO-based learning which has a negative effect on the convergence speed of the whole population, especially when solving problems with high-dimensionality in both the decision space and objective space [3].

Some recent studies that combine traditional evolutionary algorithms with efficient machine learning models have attracted much attention due to their promising search abilities in the large-scale decision space. These approaches are often termed model-based evolutionary algorithms (MBEAs) [32]. For example, generative adversarial networks (GANs) were applied in GMOEA [33] and GAN-LMEF [34] as reproduction operators to generate more promising offspring solutions. Two unsupervised neural networks, i.e., a restricted Boltzmann machine and a denoising autoencoder, were used in MOEA/PSL [35] to reduce the dimensionality of the search space. A single layer denoising autoencoder was designed in [36] to reuse the structured knowledge captured from previously optimized solutions, aiming to enhance the evolutionary search. A feedforward neural network was implemented in AMOEA/D [14] to accelerate the evolutionary search.

Although these MBEAs have shown superior advantages due to their fast convergence and efficient exploitation, their performance heavily depends on both the qualities of the training samples and the adopted models [37], [38]. In this regard, the CSO has a natural advantage and can be perfectly coupled with the model training process, as its swarm is divided into a winner particle set with good performance and a loser particle set with poor performance by pairwise competition during iteration. These particles can be directly used as ***real*** and ***fake*** samples for model training. Inspired by this, it is reasonable to believe that the CSO and a neural network model can effectively cooperate with each other due to their corresponding characteristics and advantages. Specifically, the samples for model training can be easily obtained by using the pairwise competition of the CSO, and a neural network model with good training samples can learn the promising evolutionary dynamics, which has a high potential to speed up the convergence of the CSO and generates more promising offspring particles for CSO.

Based on the previous discussion, this paper proposes a new neural net-enhanced CSO, called NN-CSO, for solving LMOPs. The main contributions of this paper are as follows:

- A three-layer neural net (NN) model with one hidden layer is designed to learn the promising evolutionary dynamics for CSO. More specifically, the loser and winner particles obtained from the pairwise competition are treated as the input and desired output samples for training our NN model. In this way, the evolutionary dynamics driving the losers towards the winners can be learned with this model.
- A new NN-enhanced CSO is presented, which not only evolves the loser particles via the original CSO learning strategy, but also applies the above well-trained NN model to guide the evolution of winner particles.
- The empirical results indicate that our proposed NN-CSO can significantly improve the performance of the original CSOs, and has a superior performance with respect to several advanced LMOEAs and to several state-of-the-art (SOTA) MBEAs when solving four different benchmark suites of LMOPs.

The rest of this paper is organized as follows. Some basic information about the neural network model, as well as a brief review of the CSO for solving LMOPs and the motivation for this work are introduced in Section II. The details of the proposed method are introduced in Section III. Our experimental results and their corresponding discussion are provided in Section IV. The limitations of our work are described in Section V. Finally, our conclusions and some possible paths for future research are provided in Section VI.

II. Preliminary

A. Neural Network Model

In 1986, *Rumelhart* and *Meclelland et al.* proposed a new learning procedure based on the back-propagating error for

TABLE I
SUMMARY OF MAIN ABBREVIATIONS

Abbreviations	Definition
CSO	competitive swarm optimizer
LMOEA	large-scale MOEA
LMOP	large-scale multi-objective optimization problem
MBEAs	model-based evolutionary algorithm
MOEA	multi-objective evolutionary algorithm
NN	neural network
PF	Pareto-optimal front
PS	Pareto-optimal set

TABLE II
SUMMARY OF MAIN NOTATIONS

Symbols	Definition
E, E_{min}	the current error and predefined desired minimum error
T, T_{max}	the current training times and maximum training times
K	the number of neurons in the hidden layer
P	the current population
W, W'	winner particles and trained winner particles
L, L'	loser particles and updated loser particles
$NN_{trained}$	the well-trained neural net
N, M, D	the population size, the number of objectives, the number of decision variables

networks of neuron-like units [39], known as back-propagation neural networks (BPNNs), which can repeatedly adjust the weights of the difference between the actual output vectors and the desired output vectors of the network. The main purpose of training model is to find a set of weights to ensure that the actual output vectors produced by the network are the same as (or infinitely close to) the desired output vectors.

Generally, BPNN model training consists of two stages, including the ***Feed-Forward*** process and the ***Back-Propagation*** process [39]. In the first stage, the input samples are fed into the input layer, and then transmitted to the output layer after being processed layer by layer in several hidden layers. After that, the actual output vectors generated by the BPNN model are compared with the desired output vectors. Thus, the current error E of the model is calculated by a loss function and then is compared with the predefined desired minimum error E_{min}. If and only if E is no longer larger than E_{min}, the model training would be finished. Otherwise, model training enters the second stage (i.e., the ***Back-propagation*** process), which applies E to repeatedly adjust the related parameters of the neural net (i.e., the weight vectors and biases) via gradient descent. For a better understanding of the model training process, its general flow diagram is plotted in Fig. 1.

When considering the constructed neural net (NN) model of this paper, we adopted a simple three-layer NN model that contains only one hidden layer. The reason behind this is that when implementing a machine learning model into a traditional EA, we need to consider not only the performance of the algorithm but also the additional cost of model training. Particularly, the numbers of neurons in the input and output layers are equal to the number of decision variables of the target problem, and the number of neurons (K) in the hidden layer is suggested to be set in the range of [5,10] for computational efficiency. Note that the parameter sensitivity analysis of K is discussed in **Section IV-G2**. In addition, the sigmoid function $f(x) = \frac{2}{1+e^{-2x}} - 1$ is adopted as an active function for the NN model and the mean-square error is used as the loss

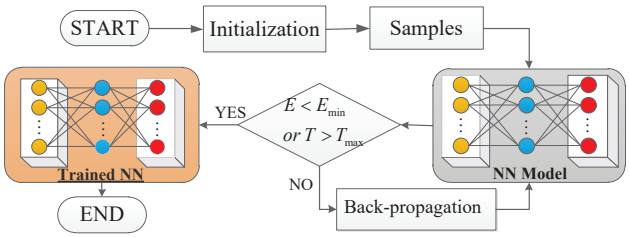


Fig. 1. The general flow diagram of the NN model training including the sampling process and the back-propagation process.

function to compute the current error rate of the NN model.

B. Related Work of Competitive Swarm Optimizer for LMOPs

In 2015, *Cheng* and *Jin* originally proposed an improved variant of a particle swarm optimizer, called the competitive swarm optimizer (CSO) [26], for solving LMOPs, which introduces the concept of a competition mechanism between particles within a single swarm [26]. Different from traditional PSOs [40]–[42] that apply the global best and personal best particles to guide the search direction of the swarm, CSO introduces a pairwise competition mechanism to guide swarm evolution. For a more intuitive understanding of the CSO, the schematic of the pairwise competition is plotted in Fig. A.1 of the Supplementary Material. Specifically, the swarm is first divided into two groups according to some specific metrics (e.g., the fitness function values and the Pareto-dominant relationship), including the winner particle set and the loser particle set. Then, the velocity and position of the loser particles in CSO [26] are updated by learning from their corresponding winner particles, formulated as follows:

$$\begin{aligned} V_l(t+1) = r_1 V_l(t) + r_2 \left(X_w(t) - X_l(t)\right) \\ +\varphi r_3 \left(\bar{X}(t) - X_l(t)\right), \end{aligned} \tag{2}$$

$$X_l(t+1) = X_l(t) + V_l(t+1), \tag{3}$$

where r_1, r_2 and r_3 are three random real-valued vectors ranging from (0, 1), $\bar{X}(t)$ is the average position of the relevant particles at the t^{th} iteration, and φ is a control parameter.

The empirical results have verified the effectiveness of CSOs in solving single-objective large scale optimization problems (LSOPs) [27], [28], [43], such as the two-phase based learning swarm optimizer (TPLSO) [28], the population entropy based swarm optimizer (DCSO) [27], and the level-based learning swarm optimizer [43]. Recently, CSOs have been successfully extended to solve LMOPs [29]–[31]. For instance, to improve the search efficiency, a two-stage strategy was suggested in LMOCSO [30] to update the position of the particles, which first pre-updates the position of each particle based on its previous velocity, and then updates the position of each pre-updated particle by learning from a leader particle. A tri-competition mechanism was proposed in S-ECSO [29], which can achieve a good trade-off between exploration and exploitation. A strong convex sparse operator was also implemented in S-ECSO, aiming to generate sparse particles

TABLE III
SUMMARY OF THE NOVELTY, ADVANTAGES, DISADVANTAGES AND THE WAY OF EVOLVING WINNER PARTICLES (W) OF EXISTING CSOS.

Algorithms	Year	Novelty	Advantages	Disadvantages	The evolution of W
CSO [26]	2015	the first work to present CSO	suitable for LMOPs	low convergence speed	None
DCSO [27]	2016	a population entropy based method with a two-stage evolutionary process	fast convergence speed	difficult to exactly divide two stages	None
LMOCSO [30]	2020	a novel position update strategy	efficient search capability	cannot properly solve multi-modal LMOPs	Mutation operator
TPLSO [28]	2021	a two-phase learning strategy	fast convergence speed	cannot guarantee diversity	None
CCSO [31]	2022	a comprehensive competitive learning strategy	good search capability	high computational cost	Mutation operator
S-ECSO [29]	2022	a tri-competition mechanism with a strongly convex sparse operator	effective for sparse LMOPs	converges prematurely	None

during the position update process. Moreover, a comprehensive competitive learning (CCL) strategy was proposed in CCSO [31], which uses three competition mechanisms, including environmental, cognitive and social competitions, to guide the particle search. Table III summarizes the novelty, advantages, disadvantages and the evolution of winner particles of the CSOs mentioned above.

As shown in the last column of Table III, most existing studies of CSOs ignore the evolution of winner particles or just apply a simple polynomial-based mutation (PM) operator [44] to evolve them. However, high-quality winner particles can better guide the evolutionary search process and speed up the convergence of the entire population, especially when solving the problems with both high-dimensional objective space and decision space. Therefore, when designing a CSO for solving LMOPs, we should pay attention not only to the evolution of loser particles, but also to the method of evolving the winner particles. Obviously, the study of the evolution of winner particles in CSOs deserves further study.

C. Our Motivations

Here, we explain the motivations of this work, as follows:

1) As summarized in the last column of Table III, most existing CSO-based works only focus on the way of guiding the evolution of the loser particles, while ignoring the evolution of the winner particles. In fact, the quality of the winner particles greatly affects the performance of CSO, as it determines the evolutionary directions. Inspired by this, this paper not only uses the original CSO search strategy to guide the evolution of the loser particles, but also attempts to design an effective method for evolving the winner particles.
2) NN models can extract useful information from training samples to guide the evolutionary process. Specifically, we expect to learn promising evolutionary dynamics to drive the input towards the desired output, by using the trained NN models. In this regard, CSO shows a natural advantage because the loser particles and winner particles obtained by pairwise competition can be directly used as samples for model training. Therefore, it is a natural idea to combine CSO and NN models.
3) CSO and NN models have natural complementary advantages. On the one hand, following the principle of pairwise competition, CSO has a strong exploration ability, but it also presents slow convergence speed to a certain extent [31]. On the other hand, NN models used in many existing MBEAs [14], [33], [34] have shown that they can help to speed up convergence while presenting an efficient exploitation ability.

Based on the above analyses, this paper designs an NN-enhanced CSO for solving LMOPs. Specifically, the loser particles and winner particles classified in CSO can be respectively used as the input and desired output samples to train the NN model. Then, the well-trained NN model is used as a reproduction operator to evolve the winner particles, which can generate more promising offspring particles for CSO. In this way, the performance and robustness of CSO in solving LMOPs can be significantly enhanced.

Algorithm 1 The framework of NN-CSO

Input: *MaxFes* (termination criterion), *N* (swarm size).
Output: *P* (the final optimal swarm).
1: *P*←Initialize a particle swarm of size *N*;
2: *NN*←Initialize the network model;
3: **while** *MaxFes* is not reached **do**
4: *W*, *L*←**Samples** (*P*);
5: $NN_{Trained}$ ←**Network Training** (*W*, *L*);
6: $W' \leftarrow NN_{Trained}$ (*W*);
7: L' ←**CSO** (*W*, *L*);
8: *P*← Environmental Selection ($L' \cup W' \cup P$);
9: **end while**

III. OUR PROPOSED METHOD

A. The Complete Framework of the Proposed NN-CSO

The pseudo-code of the framework of NN-enhanced CSO is given in **Algorithm 1** and is called NN-CSO, where *MaxFes* represents the (pre-defined) maximum number of evaluations and N is the swarm size. First, the initialization process is run in Lines 1 to 2, where the particle swarm P and the NN model are initialized separately. After that, the algorithm enters the main evolutionary loop. Specifically, the sampling process is first performed in Line 4, which is used to construct the input samples (i.e., the loser particles, denoted as ***L***) and the desired output samples (i.e., the winner particles, abbreviated as ***W***) for the NN model. More details of ***Samples*** are described in **Algorithm 2**. Then, the network training process is performed in Line 5 by using the inputs ***W*** and ***L***. The details of ***Network Training*** are explained in **Algorithm 3**. After model training is completed, two offspring reproduction strategies, including an NN-assisted learning strategy and a CSO-based learning strategy, are performed in Lines 6-7, respectively. More details of ***Offspring Reproduction*** are introduced in **Section III-C**. At the end of each iteration, the environmental selection is

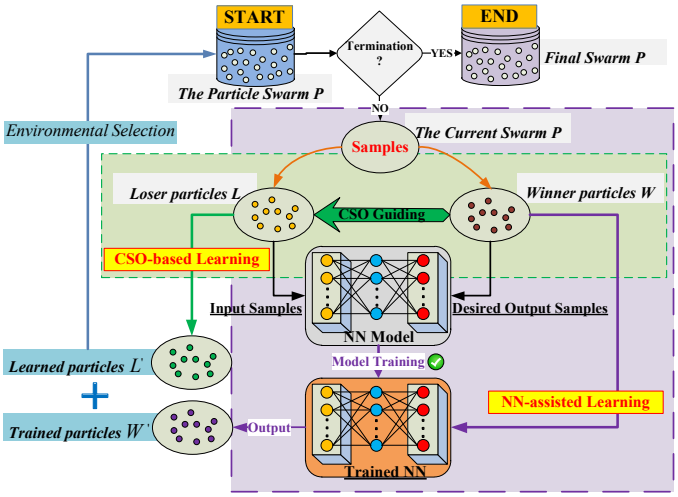


Fig. 2. The outline of NN-CSO including the sampling process, model training, NN-assisted learning, CSO-based learning, and environmental selection.

performed in Line 8. Without loss of generality, the adopted CSO-based learning strategy and environmental selection can be alternatively selected from the existing CSOs [28]–[30]. Therefore, our proposed NN-enhanced CSO framework is extensible and versatile, as it can be embedded into most existing CSOs. When the termination condition is reached, the main evolutionary loop terminates. Finally, all optimal particles in P are reported as the final optimal solution set. The outline of the proposed NN-CSO is illustrated in Fig. 2 for an intuitive observation.

B. NN Model Training

Regarding the structure of our NN model, the simplest three-layer network model with only one hidden layer is adopted in this paper. The same numbers of neurons are set in the input and output layers according to the number of decision variables in the target LMOP. In addition, when designing an MBEA, we should consider the cost and overhead generated by model training. Therefore, only one hidden layer with K neurons is adopted in our model for computational efficiency, where K is suggested to be 5 according to our empirical studies presented in **Section IV-G2**. In general, a complete model training consists of two main steps, as follows:

1) *Sampling:* The sampling process is first performed to construct the input samples and the desired output samples for training the NN model. **Algorithm 2** presents the process of collecting samples for training the NN model with one input: the current particle swarm P. Particularly, the procedure of pairwise competition of CSO is performed in Lines 3-16, and the Pareto-dominant relationship is adopted to split the swarm into two subsets, including the winner particle set $\boldsymbol{W}$ and the loser particle set $\boldsymbol{L}$. That is, half of the particles with good convergence that win the pairwise competition are added into $\boldsymbol{W}$ and the another half is divided into $\boldsymbol{L}$. As most solutions in P may be mutually non-dominated at later iterations, the Pareto-dominant relationship may be invalid. Thus, we further adopt the shift-density-estimate (SDE) method [45] to compare each pair of solutions when they are non-dominated,

Algorithm 2 Samples(P)

Input: P (the current particle swarm), N (the swarm size).
Output: W (the **winner** samples), L (the **loser** samples) .

1: Initialize $W \leftarrow \varnothing$, $L \leftarrow \varnothing$;
2: **while** $W < N/2$ **do**
3: randomly select two particles x_1 and x_2 from P;
4: **if** (x_1 dominates x_2) **then**
5: add x_1 into W and x_2 into L;
6: **else if** (x_1 is dominated by x_2) **then**
7: add x_1 into L and x_2 into W;
8: **else**
9: calculate the fitness values using (4);
10: **if** ($f_{SDE}(x_1) > f_{SDE}(x_2)$) **then**
11: add x_1 into W and x_2 into L;
12: **else**
13: add x_1 into L and x_2 into W;
14: **end if**
15: **end if**
16: **end while**

as this method can effectively reflect the diversity situation of particles. Here, the formulation of SDE is provided as follows:

$$f_{\mathrm{SDE}}(x_i) = \min_{x_j \in P \wedge i \neq j} \sqrt{\sum_{m=1}^{M} \left(\max\left\{0, f_m(x_j) - f_m(x_i)\right\}\right)^2}, \tag{4}$$

where $f_m(x_i)$ denotes the m^{th} objective value of particle x_i, and M is the number of objectives. Finally, the samples for model training are prepared, where the particles in $\boldsymbol{L}$ and $\boldsymbol{W}$ are treated as the input and desired output, respectively.

2) *Model Training:* When the above sampling is completed, the algorithm enters the model training stage. **Algorithm 3** presents the general training process of the NN model. At first, some related parameters (i.e., the biases and the weight vectors) of the NN model are initialized in Line 1. After that, the training procedure is performed in Lines 2-10. The current error rate E is computed in Line 4. Then, the weight vectors and biases of each neuron are updated by using gradient descent, denoted as $\partial E/\partial(w, v)$. Note that the training loop terminates when the current error rate E of the network is no longer larger than the predefined desired minimum error rate E_{min} or when the training time T reaches the maximum allowable training time T_{max}. Finally, the properly trained NN model, denoted as $NN_{Trained}$, can learn the promising evolutionary search direction that approximates the optima. Thus, the final NN model can be used as a reproduction operator to evolve the winner particles, aiming to generate high-quality solutions for guiding the CSO search.

C. NN-Assisted Offspring Reproduction

After the model training is finished, the process of offspring reproduction is performed. In this article, two reproduction strategies are used for offspring generation, including an NN-assisted learning strategy and a CSO-based learning strategy. Fig. 3 provides a diagram of the two proposed offspring reproduction strategies. Specifically, the NN-assisted learning

Algorithm 3 Network Training (W, L)

Input: W (the **winner** samples), L (the **loser** samples), E_{min} (the minimum error) and T_{max} (the maximum training times).

Output: $NN_{Trained}$ (the trained network model).

```
initialize the biases and weight vectors for NN;
for all samples in W and L do
    //Feed-Forward process:
    calculate the current error rate E of NN;
    if (E>E_min or T<T_max) then
        //Back-propagation process:
        adjust the parameters using ∂E/∂(w, v);
        T=T+1;
    end if
end for
```

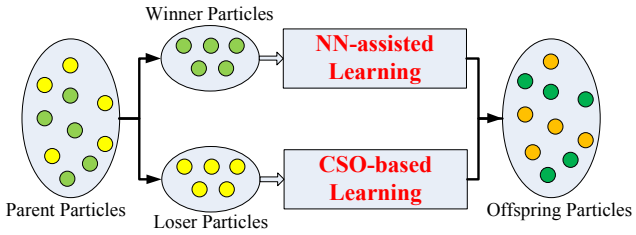


Fig. 3. Two offspring reproduction strategies in NN-CSO: NN-assisted learning for winner particles and CSO-based learning for loser particles.

strategy is designed for evolving the winner particles, which applies the well-trained NN model obtained in **Algorithm 3** to guide the evolution of the winner particles. In this way, promising offspring can be generated by learning from the properly trained NN model that can provide promising evolutionary search direction for CSO. On the other hand, the CSO-based learning strategy is adopted for the evolution of the loser particles, where the velocity and position of the loser particles are updated by learning from the winner particles, which maintain the original search pattern of CSO. As mentioned above, our designed model can be embedded into most existing CSOs [28]–[30], [43]. In addition, experimental results provided in **Section IV-D** show that our designed NN model is scalable for different CSOs and can significantly improve the performance of the original CSOs in solving a variety of LMOPs.

D. Computational Complexity of NN-CSO

As presented in **Algorithm 1**, our proposed NN-CSO consists of five main components, including the sampling process, the training process of the neural network, the evolution of the winner particles, the evolution of the loser particles, and the environmental selection. Specifically, the sampling process in **Algorithm 2** has a time complexity of $O(MN^2)$, where M and N denote the number of objectives and the population size, respectively. In addition, the training of the neural network in **Algorithm 3** has a time complexity of $O(NDEK)$, where D, E and K represent the number of decision variables, the epochs for training the neural network and the number of neurons in the hidden layer. Furthermore, the time complexity of the evolution of the winner particles is $O(NDK)$, and the evolution of the loser particles has a time complexity of $O(ND)$. Finally, the time complexity of environmental selection depends on the specific selection strategy used in the original CSO. Taking LMOCSO [30] as an example, its environmental selection has a time complexity of $O(MN^2)$. In summary, the overall time complexity of NN-CSO for one generation is $O(MN^2 + NDEK)$.

IV. Experimental Studies

A. Experimental Arrangement

To empirically examine the performance of NN-CSO, three types of experimental comparisons are conducted in this paper. First, the effectiveness of our designed NN model for CSO is validated by embedding the designed NN model into three well-known CSOs (i.e., LMOCSO [30], TPLSO [28] and S-ECSO [29]). Second, the performance of NN-CSO is compared with respect to five advanced LMOEAs, including MOEA/DVA [8], LSMOF [16], DGEA [20], LMEA [9] and LSMOEA/D [11], respectively on solving several LMOPs. Third, we conducted experimental comparisons between NN-CSO and two SOTA MBEAs (i.e., MOEA/PSL [35] and AMOEA/D [14]). The experiments are organized as follows:

- The effectiveness of the proposed NN model is evaluated by embedding it into three well-known CSOs. Please note that the related experiments are provided in **Section IV-D**.
- The superiority of the proposed NN-CSO over five advanced LMOEAs in solving several LMOPs is validated. Please note that the related experimental comparisons and analysis can be found in **Section IV-E**.
- The comparisons between our proposed method and two SOTA MBEAs are provided in **Section IV-F**.
- Some further discussions are provided in **Section IV-G**, including visualizations of the final solution sets, parametric analysis, execution efficiency, ablation analysis, and performance study on super-large-scale problems.

The proposed method and all the compared algorithms except AMOEA/D are implemented in the PlatEMO framework using MATLAB [46], while AMOEA/D is implemented in the jMetal framework [47] which uses JAVA. All the algorithms are run on a personal computer with an Intel Core i7-6700 CPU, 3.40 GHZ (processor), and 20 GB of RAM.

B. Test Problems and Performance Measures

Benchmark problems: In our experimental studies, four test suites (LSMOP1-LSMOP9 [48], UF1-UF10 [49], DTLZ1-DTLZ7 [50], and WFG1-WFG9 [51]) are used to validate the performance of our proposed algorithm. Regarding the numbers of objectives (M), we adopt M = {2, 3, 5, 8, 10} for LSMOP1-LSMOP9, DTLZ1-DTLZ7 and WFG1-WFG9, M = 2 for UF1-UF7 and M = 3 for UF8-UF10 in our experiments. Regarding the number of decision variables (D), we adopt D= {100, 200, 500, 1000} for all problems.

Performance measures: To accurately reflect the quality of the final solution set obtained by each compared algorithm, the inverted generational distance (IGD) [52] is adopted as our performance measure. A smaller IGD value indicates a better

performance of the algorithm. The set of reference points required for calculating the IGD values is evenly sampled from the true PF of each test problem with a size close to 10000 points. Note that all the problems are tested for 20 independent runs, and the mean and standard deviation of the IGD values are recorded. In addition, in order to ensure a statistically sound conclusion, the Wilcoxon rank sum test with a 0.05 significance level and the Wilcoxon signed-rank test using the platform ***KEEL*** [53] are also used in the experimental analysis. Note that the symbols "+", "-", and "=" in tables respectively indicate that the compared algorithms are *significantly better than*, *worse than* and *similar to* our proposed approach.

C. Parameters Settings for the Compared Algorithms

For a fair comparison, some unique parameters of each compared algorithm (i.e., search strategies and corresponding parameter settings) are set as recommended in their original references, as summarized in Table A.1 of the Supplementary Material. In NN-CSO, the number of neurons in the hidden layer K is set to 5 according to our parametric analysis, which is discussed in **Section IV-G2**. For training the neural network models, at each iteration, the learning rate α is set to 0.01, the number of training epochs ($epochs$) is set to 20, and the desired minimum error (E_{min}) is set to 0.01. In addition, the population size N and the termination condition $MaxFes$ are set to 300 and 100000 for all test problems, respectively.

D. Results of Embedding the NN Model into Three CSOs

To verify the effectiveness of our designed NN model for CSOs, three NN-enhanced CSO variants are presented by embedding the proposed NN model into three well-known CSOs (S-ECSO [29], LMOCSO [30], and TPLSO [28]), abbreviated as NN-SECSO, NN-LMOCSO, and NN-TPLSO, respectively. In particular, TPLSO is extended for solving the target LMOPs by using the SDE method [45] to replace the objective value as the performance measure. Then, the experimental comparisons between these original CSOs and their corresponding NN-enhanced CSOs are conducted on four test suites of LMOPs (i.e., LSMOP, UF, DTLZ and WFG test problems). In addition, the convergence performance of each compared algorithm is further investigated.

1) Comparisons on UF Test Problems

The average IGD results and standard deviations obtained by each compared algorithm on 40 UF test problems with 2-3 objectives and 100 to 1000 decision variables are provided in Table A. 2 of the Supplementary Material. Specifically, the numbers of test problems that are better solved by the proposed NN model are 30, 32 and 22 out of 40 UF test problems, respectively for S-ECSO vs NN-SECSO, LMOCSO vs NN-LMOCSO, and TPLSO vs NN-TPLSO, whereas only 8, 0, and 4 cases experienced deterioration with the proposed NN model. Therefore, the experimental results in Table A.2 have validated the effectiveness of our proposed NN model, as it can obviously improve the performance of the original CSOs in solving most of the UF test problems.

2) Comparisons on LSMOP Test Problems

Due to page limitations, the average IGD results and standard deviations after 20 independent runs of the six above algorithms on 72 LSMOP test problems with 2-objective and 3-objective and 100 to 1000 decision variables are summarized in Table A.3 of the Supplementary Material. Note that the better mean results for each comparison between the original CSOs and their corresponding NN-enhanced CSOs are marked in **boldface**. As observed from Table A.3, the effectiveness of the proposed NN model is validated, as these NN-enhanced CSOs clearly enhance the performance of the original CSOs in solving most of the test problems adopted. More specifically, the numbers of test problems in which performance was improved by our NN model are 55, 68 and 61, respectively, for S-ECSO vs NN-SECSO, LMOCSO vs NN-LMOCSO, and TPLSO vs NN-TPLSO, whereas only in 4, 3, and 0 cases experienced deterioration with our NN model. Therefore, the superior performance of these NN-enhanced CSOs over their corresponding original CSOs has shown that our NN model can significantly improve the performance of CSOs in solving LSMOP test problems.

3) Comparisons on the DTLZ Test Problems

The average IGD results and standard deviations of the compared algorithms on 84 DTLZ test problems with 5 to 10 objectives and 100 to 1000 decision variables are summarized in Table A.4 of the Supplementary Material. In Table A.4, the NN-enhanced CSOs (i.e., NN-SECSO, NN-LMOCSO, and NN-TPLSO) perform better than their corresponding original CSOs (i.e., S-ECSO, LMOCSO, and TPLSO) in 58, 75 and 67 out of 84 DTLZ test problems, respectively, and are only inferior in 26, 6 and 7 cases, respectively. Therefore, the advantages of the NN-enhanced CSOs over their corresponding original CSOs in solving most of DTLZ test problems directly confirm the effectiveness of our proposed NN model.

4) Comparisons on the WFG Test Problems

The average IGD results and standard deviations obtained by each compared algorithm on 108 WFG test problems are displayed in Table A.5 of the Supplementary Material. According to the results presented in Table A.5, the effectiveness of our proposed NN-enhanced CSO is validated, as NN-SECSO, NN-LMOCSO, and NN-TPLSO obtain better results in 92, 77 and 107 out of 108 instances from the WFG test suite, respectively, whereas their corresponding original CSOs (S-ECSO, LMOCSO, and TPLSO) only obtain better results in 16, 5, and 0 cases, respectively.

In summary, the experimental results of the above pairwise comparisons (i.e., S-ECSO vs. NN-SECSO, LMOCSO vs. NN-LMOCSO, and TPLSO vs. NN-TPLSO) show that the three NN-enhanced CSO variants have significant advantages over their corresponding original CSOs in solving all four suites of LMOPs. The reason behind this is that the aforementioned traditional CSOs focus only on the evolution of the loser particles and ignore the evolution of the winner particles, which largely limits the capability of CSOs to solve different types of LMOPs, especially for the target problems with many objectives and a large number of decision variables.

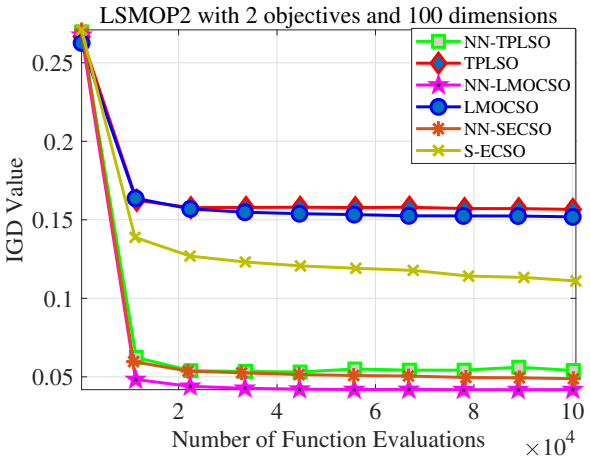


Fig. 4. The convergence profiles of the six compared algorithms on LSMOP2 with 2 objectives and 100 decision variables.

5) Further Discussions on the Convergence Performance

Here, the convergence profiles of all the compared algorithms are plotted in Fig. 4 and Fig. A.2 of the Supplementary Material, for solving the following problems: the 2-objective LSMOP2 with 100 decision variables, the 3-objective LSMOP1 with 100 decision variables, the 3-objective LSMOP3 with 500 and 1000 decision variables, the 2-objective LSMOP8 with 500 and 1000 decision variables, and the 2-objective LSMOP9 with 500 and 1000 decision variables, respectively. Please note that these LSMOP test problems have different characteristics, hence the superior performance of our NN-enhanced CSOs with respect to their original versions can be verified in a comprehensive manner.

From Fig. 4 and Fig. A.2, we can derive some meaningful observations. First, NN-SECSO, NN-LMOCSO, and NN-TPLSO show a faster convergence speed than their corresponding original CSOs (i.e., S-ECSO, LMOCSO, and TPLSO), which confirms that our proposed NN model can accelerate the search process of the original CSOs. Second, NN-SECSO, NN-LMOCSO, and NN-TPLSO can achieve a promising IGD level at an early evolutionary stage, whereas S-ECSO, LMOCSO, and TPLSO fail to achieve an acceptable IGD level and get trapped in local optima in most cases. This indicates that our proposed NN model has an efficient exploitation ability which can help traditional CSOs to avoid falling into local optima. In particular, as shown in Figs. A.2 (b)-(e), although the IGD values of NN-SECSO are the same as those of S-ECSO, NN-SECSO exhibits a faster convergence than S-ECSO. In summary, we can conclude that the proposed NN model not only can speed up convergence of traditional CSOs, but that it also helps them to avoid falling into local optima. Therefore, the NN-enhanced CSOs show obvious advantages in solving a variety of LMOPs when compared to their corresponding original CSOs.

E. Comparisons with respect to Five Advanced LMOEAs

In this part, NN-LMOCSO is adopted as our representative algorithm for performance comparison, but it is abbreviated as NN-CSO for ease of description. To verify the effectiveness of NN-CSO in solving several LMOPs (i.e., the LSMOP, UF, DTLZ and WFG test problems), five competitive LMOEAs, including LSMOF [16], DGEA [20], LMEA [9], LSMOEA/D [11], and MOEA/DVA [8], are adopted here for comparison.

1) Comparisons on the UF Problems

In this subsection, NN-CSO is compared to LSMOF, DGEA, LMEA, LSMOEA/D and MOEA/DVA on UF test problems with 2-objective and 3-objective and 100 to 1000 decision variables. Due to page limitations, the average IGD results from 20 independent runs of NN-CSO and the five compared LMOEAs on the UF test problems are summarized in Table A.6 of the Supplementary Material, respectively. Compared to the five competitive LMOEAs previously indicated, our proposed NN-CSO achieved the best performance in 15 out of 40 of the UF test problems, respectively. More specifically, NN-CSO outperformed its five competitors in 17, 40, 30, 40 and 40 out of the 40 UF problems adopted, and it was worse in 13, 0, 10, 0 and 0 cases, respectively. Therefore, it can be concluded that NN-CSO performs much better than the other LMOEAs adopted in our comparative study, for most of the UF test problems.

Similarly, based on the value of D (dimensionality of the problem), the separate statistical results of these five LMOEAs versus our proposed NN-CSO in solving the UF test problems are summarized in Table IV. Clearly, NN-CSO is significantly better than DGEA, LSMOEA/D and MOEA/DVA on all the UF test problems. When compared to LSMOF, DGEA, LSMOEA/D and MOEA/DVA, NN-CSO is never outperformed by any of these competitors in each of the dimensionalities adopted for these test problems. For LMEA, NN-CSO shows obvious advantages in most cases except for the UF test problems with $D = 100$. Therefore, when compared to five advanced LMOEAs, our proposed NN-CSO shows superior performance in solving UF1 to UF10 with 2-objective and 3-objective and 100 to 1000 decision variables.

2) Comparisons on the LSMOP, DTLZ and WFG Problems

Here, NN-CSO is compared with respect to LSMOF, DGEA, LMEA, LSMOEA/D and MOEA/DVA on 180 LSMOP, 140 DTLZ and 180 WFG test problems. The average IGD results from 20 independent runs of NN-CSO and these five compared LMOEAs on the LSMOP, DTLZ and WFG test problems with 100 to 1000 decision variables (D) and 2 to 10 objectives (M) are presented in Table A.7, Table A.8 and Table A.9 of the Supplementary Material, respectively. NN-CSO obtained the best performance in most cases when compared to its five competitors on the LSMOP, DTLZ and WFG test problems, since NN-CSO performed best in 127 out of 180 LSMOP test problems, 69 out of 140 DTLZ test problems and 66 out of 180 WFG test problems, respectively. According to the summarized results collected in the last rows of these three tables, NN-CSO is better than LSMOF, DGEA, LMEA, LSMOEA/D and MOEA/DVA in 118, 106, 170, 162 and 159 out of 180 LSMOP problems, respectively. In addition, NN-CSO is better than LSMOF, DGEA, LMEA, LSMOEA/D and MOEA/DVA in 87, 101, 108, 139 and 121 out of 140 DTLZ problems, respectively, and better than LSMOF, DGEA, LMEA, LSMOEA/D and MOEA/DVA in 99, 107, 125, 152 and 167 out of 180 WFG problems, respectively. Therefore,

TABLE IV
SUMMARY OF THE COMPARISON OF RESULTS BETWEEN NN-CSO AND ITS COMPETITORS BASED ON IGD VALUES

NN-CSO vs LMOEAs	D	Test Benchmark Problems			
		on LSMOP +/-/=	on UF +/-/=	on DTLZ +/-/=	on WFG +/-/=
LSMOF	100	2/18/7	3/4/3	11/22/2	17/22/6
	200	7/31/7	4/4/2	9/21/5	15/26/4
	500	7/28/10	4/4/2	8/23/4	15/23/7
	1000	4/26/15	2/5/3	7/21/7	11/26/6
	ALL	**20/118/42**	**13/17/10**	**35/87/18**	**58/99/23**
DGEA	100	0/19/8	0/10/0	1/22/12	5/30/10
	200	0/25/20	0/10/0	0/27/8	4/31/10
	500	3/28/14	0/10/0	0/24/11	7/29/9
	1000	4/27/14	0/10/0	0/24/11	15/17/13
	ALL	**8/106/66**	**0/40/0**	**1/101/38**	**31/107/42**
LMEA	100	4/18/5	10/0/0	31/3/1	29/7/9
	200	0/45/0	0/10/0	0/35/0	10/32/3
	500	0/45/0	0/10/0	0/35/0	2/43/0
	1000	0/44/1	0/10/0	0/35/0	2/43/0
	ALL	**4/170/6**	**10/30/0**	**31/108/1**	**43/125/12**
LSMOEA/D	100	2/21/4	0/10/0	0/35/0	3/41/1
	200	1/41/3	0/10/0	0/35/0	3/41/1
	500	1/41/3	0/10/0	0/35/0	4/37/4
	1000	0/41/4	0/10/0	0/34/1	6/33/6
	ALL	**4/162/14**	**0/40/0**	**0/139/1**	**16/152/12**
MOEA/DVA	100	0/26/1	0/10/0	0/35/0	2/43/0
	200	0/44/1	0/10/0	0/35/0	2/42/1
	500	6/36/3	0/10/0	4/28/3	4/40/1
	1000	9/35/1	0/10/0	11/23/1	3/42/0
	ALL	**15/159/6**	**0/40/0**	**15/121/4**	**11/167/2**

the experimental results in Tables A.7 to A. 9 show that NN-CSO performs significantly better than its five competitors on most of the LSMOP, DTLZ and WFG test problems.

Furthermore, based on the value of D, the statistical results of these five LMOEAs versus NN-CSO on solving the LSMOP, DTLZ and WFG test problems are summarized in Table IV. From these statistical results, NN-CSO is clearly better than its five competitors on the LSMOP, DTLZ and WFG test suites with varying dimensionality. Specifically, when solving the DLTZ and WFG test problems, NN-CSO outperforms LSMOF, DGEA, LMEA, LSMOEA/D and MOEA/DVA in most cases, while NN-CSO only shows slight disadvantages with respect to LMEA on the DTLZ and WFG test problems with $D = 100$. When solving the LSMOP test problems, NN-CSO outperforms its competitors in most cases. Therefore, we can conclude that our proposed NN-CSO shows significant advantages in solving most of the LSMOP, DTLZ and WFG test problems when compared with five advanced LMOEAs.

3) Comparisons of the Overall Performance

To clearly show the overall performance of each algorithm when solving the LMOPs with different dimensionalities, all the compared algorithms are ranked by the Friedman test in the ***KEEL*** platform. Moreover, to visualize the ranking scores of each compared algorithm, Fig. 5 shows the ranking scores of all algorithms based on their corresponding dimensionality. More specifically, NN-CSO achieves the best performance in all the dimensionalities adopted, as its scores are 1.9, 1.6, 1.5 and 1.7 for the test problems with D = 100, 200, 500 and 1000, respectively, which are much lower than those of the other compared algorithms. As shown in Fig. 5, compared to its five competitors, NN-CSO shows an obvious advantage for test problems with D = 100, 200 and 500. Note that, when

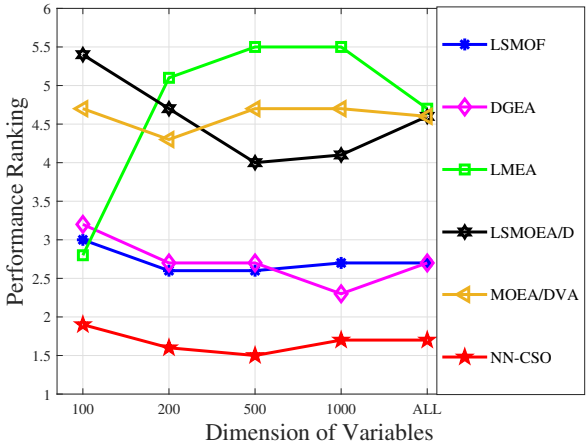


Fig. 5. Illustration of the average performance ranks over all test problems for different dimensionalities (D).

D = 1000, NN-CSO is slightly better than DGEA, as the average ranking scores of NN-CSO and DGEA are 1.8 and 2.2, respectively. Finally, NN-CSO has the lowest average ranking score of 1.7 for all problems for all dimensionalities, as seen in the last column "ALL" of Fig. 5. This value is much lower than that of the other competitors.

Furthermore, Fig. A.3 of the Supplementary Material shows the ranking scores of each test suite for all algorithms. According to Fig. A.3, NN-CSO achieved the best performance on all test suites, as it obtained the ranking scores of 2.1, 1.8, 1.4 and 1.5 on the WFG, UF, DTLZ and LSMOP test suites, respectively. Specifically, NN-CSO shows clear advantages on the WFG, DTLZ and LSMOP test suites, as the ranking scores of NN-CSO on the three above test suites are much lower than those of LSMOF, DGEA, LMEA, LSMOEA/D and MOEA/DVA. Considering the UF test suites, NN-CSO is much better than DGEA, LMEA, LSMOEA/D and MOEA/DVA, but is slightly better than LSMOF, since the average ranking scores of NN-CSO and LSMOF are 1.8 and 1.9, respectively. To conclude, as can be seen from the last column labelled as "ALL", NN-CSO had the lowest average ranking score (1.7) for all the test suites adopted, while the average ranking scores of these five LMOEAs were 2.7, 2.8, 4.6, 4.7 and 4.5, respectively.

4) Observations and Discussions

In summary, based on the above experimental results, the following observations and conclusions can be drawn.

First, three LMOEAs based on decision variable analysis (i.e., MOEA/DVA, LMEA and LSMOEA/D), perform poorly in all the adopted test problems. The main reason for this poor performance might be that they require a large number of evaluations to perform variable analysis, and in our experiments, we allocated a low number of evaluations for optimizing the target LMOPs. As a result, it is difficult for them to approximate the optimal solutions with limited computational resources. Therefore, it is critical to balance the resource allocation between variable analysis and evolution when solving LMOPs.

Second, the performance of DGEA on all the test problems

adopted with different characteristics are similar and poor. The main reason for the poor performance of DGEA is due to the inefficiency of its search operator. Therefore, when designing a search operator for solving LMOPs, we should not only consider the exploitation capability but also consider the exploration capability of the search operator.

Third, LSMOF, which is based on problem transformation performs relatively well in all dimensionalities due to its fast convergence speed based on a reduction of the dimensionality of the target problem. Nevertheless, LSMOF could not achieve the best performance, because it may get trapped in local optima to some extent. The reason behind this is that the problem transformation method adopted in LSMOF narrows down the search space, so LSMOF fails to fully explore the original search space to obtain promising solutions.

Overall, our proposed NN-CSO performs much better than the five above competitive LMOEAs when solving several LMOPs with different dimensionalities. This validates that the traditional CSOs enhanced with our NN model are more advantageous. In addition, experimental comparisons with two latest relevant works (i.e., PMMOEA [18] and CCSO [31]) are conducted to further verify the superiority of our proposed method. The experimental results on four adopted LMOPs are presented in Table A. 10 of the Supplementary Material, which further verifies that NN-CSO shows clear advantages over PMMOEA and CCSO in most of the adopted cases.

F. Comparisons with Two SOTA Large-scale MBEAs

As our proposed NN-CSO is a model-based evolutionary algorithm (MBEA), two SOTA large-scale MBEAs (i.e., MOEA/PSL [35] and AMOEA/D [14]) are also adopted in our experiments for performance comparison. Due to page limitations, the IGD results obtained by each compared algorithm for solving each type of benchmark problem are provided in Tables A.11 to A.14 of the Supplementary Material. Table III summarizes the final statistical comparison results for solving all the test problems. As shown in the last column of Table III, NN-CSO outperforms MOEA/PSL and AMOEA/D in 301 and 306 out of 540 cases, respectively, while it is only outperformed in 123 and 168 cases. In addition, the overall performance of each compared algorithm on solving all test problems is evaluated by using the Friedman test, where the performance scores for NN-CSO, MOEA/PSL and AMOEA/D are 1.6, 2.3 and 2.1, respectively. That is, the performance score of NN-CSO for solving all test problems is much lower than that of the two compared MBEAs. In summary, these empirical results further verify the superiority of NN-CSO over these two SOTA MBEAs in most of the adopted test cases.

G. Further Discussion

1) Visualizations of the Final Solution Sets

To visually show the final solutions' distribution in objective space, Fig. 6 and Figs. A.4 and A.5 of the Supplementary Material depict the final solution sets obtained by our proposed NN-CSO and its seven compared algorithms on WFG8 with 2 objectives and 1000 decision variables, DTLZ5 with 5 objectives and 1000 decision variables, LSMOP4 with 2 objectives and 500 decision variables, and UF4 with 2 objectives and 1000 decision variables, respectively. As shown in these figures, the final solution sets obtained by our proposed NN-CSO show better diversity distribution than the other approaches. Moreover, the final solution sets obtained by NN-CSO can approximate the true PF more closely than the others. In Summary, NN-CSO shows significant superiority over the other compared algorithms, which confirms the efficiency of our proposed NN model in handing a variety of LMOPs.

TABLE V
SUMMARY OF THE FINAL STATISTICAL COMPARISON RESULTS OF THREE MBEAS ON SOLVING ALL TEST PROBLEMS BASED ON IGD VALUES.

NN-CSO vs MBEAs	Test problems				
	on LSMOP + / - / =	on UF + / - / =	on DTLZ + / - / =	on WFG + / - / =	in total + / - / =
MOEA/PSL	19 / **96** / 65	11 / **23** / 6	41 / **86** / 13	52 / **96** / 32	123 / **301** / 116
AMOEA/D	73 / **84** / 23	1 / **38** / 1	44 / **85** / 11	50 / **99** / 31	168 / **306** / 66

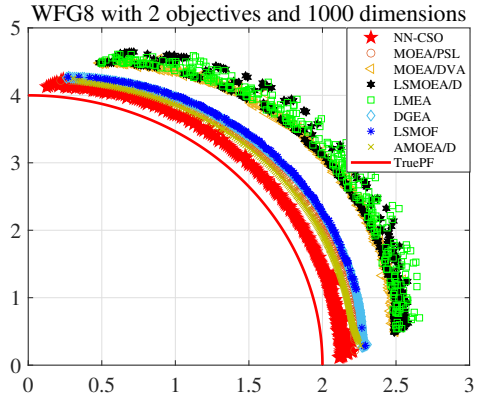


Fig. 6. The final population sets obtained by the eight compared algorithms on WFG8 with 2 objectives and 1000 decision variables.

2) Parameter Sensitivity Analysis of our NN Model

In this section, the sensitivity to the number of neurons K in the hidden layer, which is a very important parameter in NN-CSO that affects the architecture and learning ability of the NN model, is analyzed. Specifically, four different values of K (i.e., K=$\{2, 5, 10, 15\}$ are considered for solving 2-objective LSMOP1 with 100 and 1000 decision variables.

As observed from the average IGD values and running times of NN-CSO with different values of K in Fig. 7 and Fig. A.6 of the Supplementary Material, NN-CSO with K = 2 shows the worst performance in solving these two LMOPs, which indicates that a simple structure of the NN model is insufficient for learning the evolutionary direction from the loser particles towards the winner particles. Moreover, NN-CSO shows a similar performance in the cases of K = $\{5, 10, 15\}$. However, the time cost of NN-CSO increases dramatically as the value of K increases. In other words, if the number of neurons (K) in the hidden layer is too large, the structure of the NN model will become complicated. Unfortunately, training such a complex NN model is very time consuming, and it fails to significantly improve the performance of the algorithm. Consequently, by considering both the performance and training time cost of NN-CSO, we suggest that the value of K should be set between 5 and 10, and we adopt K= 5 in this paper.

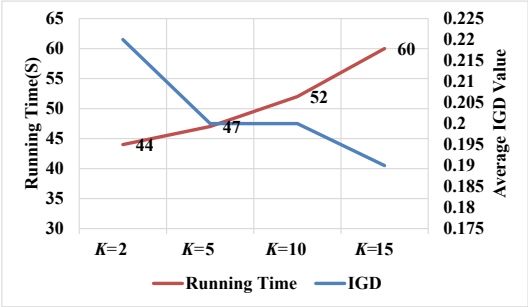


Fig. 7. Average IGD values and running time of NN-CSO with different values of K in solving 2-objective LSMOP1 with 100 decision variables.

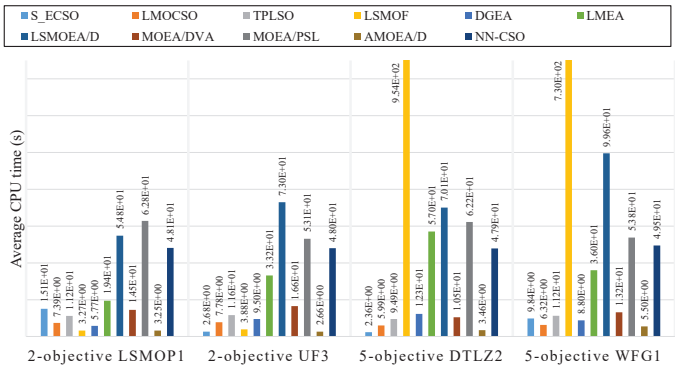


Fig. 8. Average CPU times of all the compared algorithms on some LMOPs.

3) Comparisons in terms of Execution Efficiency

The training time is also an important criterion for evaluating the efficiency of an algorithm, especially for an MBEA that combines the conventional MOEA with a neural network model. Therefore, we further investigate the execution efficiency of NN-CSO and the other algorithms compared in our experimental study. The average CPU running time of each compared algorithm from 20 runs is plotted in Fig. 8 and Fig. A.7 on the 2-objective LSMOP1 and UF3 with 100 and 1000 decision variables, and the 5-objective DTLZ2 and WFG1 with 100 and 1000 decision variables, respectively.

Some conclusions can be learned from Fig. 8 and Fig. A.7. First, compared to the three original CSO variants considered here (i.e., S-ECSO, LMOCSO and TPLSO), the CPU running times of NN-CSO are approximately three or four times longer than those of the original CSOs. That is, our proposed NN-enhanced CSO can significantly improve the performance of traditional CSOs with only a small additional time cost. Second, considering NN-CSO and two MBEAs (i.e., MOEA/PSL and AMOEA/D), MOEA/PSL shows the worst execution efficiency due to its complex network structure, and AMOEA/D has the lowest time cost, because the network model in AMOEA/D is only trained once per iteration. Nevertheless, such rough network model training fails to obtain a network model with high precision, which may degrade the optimization performance of the algorithm to some extent. Overall, problem transformation based methods, such as LSMOF, have the longest running time among all competitors. The reason behind this is that LSMOF consumes considerable time tracking PS of LMOPs. In summary, NN-CSO not only shows promising performance in solving several LMOPs, but also has good execution efficiency.

4) Ablation Analysis of NN-CSO

To validate the effectiveness of each component contained in NN-CSO, five variants of NN-CSO are implemented in our ablation experiments. Specifically, the arrangements of the ablation experiments are as follows:

- To study the effectiveness of the evolutionary mechanism of NN-CSO, two variants (i.e., NN-CSO-I and NN-CSO-II) are designed, where NN-CSO-I applies the NN-assisted learning strategy to evolve winner particles and loser particles, and NN-CSO-II directly uses the NN-assisted learning strategy to evolve the entire population.
- To investigate whether the performance improvement is attributed to the evolution of winner particles via the NN-assisted learning strategy of NN-CSO or due to the use of the network model, two variants (i.e., NN-CSO-III and NN-CSO-IV) are designed, where NN-CSO-III only performs the mutation operation rather than the NN-assisted learning strategy on the winner particles. Conversely, NN-CSO-IV trains the NN model by using random samples rather than those obtained through the pairwise competition of CSO.
- To verify the superiority of our designed NN model in NN-CSO, a variant (i.e., AES-CSO) is designed, which adopts the AES model proposed in AMOEA/D [14] to replace our designed NN model.

To verify the superiority of NN-CSO over its five variants, without loss of generality, several different test problems with different characteristics are adopted. Table VI provides the average IGD results for NN-CSO and its five variants when solving a number of different benchmark problems (i.e., 5-objective LSMOP1 and LSMOP9, 2-objective UF3 and 3-objective UF9, 5-objective DTLZ1 and DTLZ2, 5-objective WFG2 and WFG8) with 100 and 1000 decision variables, which exhibit distinct characteristics and PF shapes.

As observed from Table VI, NN-CSO outperforms these five variants in most cases. Specifically, the superiority of NN-CSO over NN-CSO-I and NN-CSO-II shows the effectiveness of our evolutionary mechanism, in which the winner particles are evolved by the NN-assisted learning strategy, while the loser particles are updated by learning from the winner particles. Similarly, NN-CSO shows obvious advantages over NN-CSO-III and NN-CSO-IV, which validates that the performance improvement is due to the use of an NN-assisted learning strategy to evolve the winner particles. Finally, NN-CSO outperforms AES-CSO in most cases, which validates the effectiveness of our designed NN model. In Summary, these ablation experiments further verify the effectiveness of each component contained in NN-CSO.

5) Performance Study on Super-large-scale Problems

In this part, we further explore the performance of our proposed NN-CSO and other compared algorithms in solving super-large-scale problems [54], i.e., problems with more than 5000 decision variables. Table A.15 of the Supplementary Material provides the IGD results for all the compared algorithms when solving the same benchmark problems as those from Table VI with 5000 and 10000 decision variables. Obviously,

TABLE VI
THE IGD COMPARISON RESULTS OF NN-CSO AND ITS FIVE VARIANTS WHEN SOLVING SOME TEST PROBLEMS WITH 100, 1000 DECISION VARIABLES.

Problem	M	D	NN-CSO-I	NN-CSO-II	NN-CSO-III	NN-CSO-IV	AES-CSO	NN-CSO
LSMOP1	5	100	8.6111e-1 (3.33e-2) -	8.5873e-1 (3.21e-2) -	**4.7380e-1 (3.45e-2) +**	5.8532e-1 (6.48e-2) -	6.4028e-1 (5.64e-2) -	5.4001e-1 (4.78e-2)
LSMOP1	5	1000	1.0216e+0 (2.79e-2) -	1.0019e+0 (1.61e-2) -	1.4141e+0 (1.59e-1) -	9.0578e-1 (1.08e-1) =	9.4360e-1 (6.43e-2) -	**8.7563e-1 (5.58e-2)**
LSMOP9	5	100	1.5749e+0 (3.52e-1) -	1.4012e+0 (2.39e-1) -	9.4260e-1 (2.15e-1) =	9.9477e-1 (2.22e-1) -	**8.8636e-1 (1.03e-1) =**	9.2390e-1 (7.70e-2)
LSMOP9	5	1000	1.1914e+0 (3.27e-1) -	1.1563e+0 (4.32e-2) -	7.9858e+1 (5.83e+1) -	1.1741e+0 (2.07e-1) -	**8.9144e-1 (1.51e-1) =**	9.3546e-1 (2.02e-1)
UF3	2	100	4.9611e-1 (1.84e-2) -	4.7689e-1 (2.00e-2) -	2.3434e-1 (7.81e-3) -	1.8467e-1 (5.15e-3) =	**1.7537e-1 (3.88e-3) =**	1.8399e-1 (7.77e-3)
UF3	2	1000	4.6358e-1 (2.05e-2) -	4.6738e-1 (2.73e-2) -	3.2107e-1 (3.19e-3) -	1.2537e-1 (1.14e-3) =	1.2576e-1 (8.81e-4) =	**1.2473e-1 (7.05e-4)**
UF9	3	100	1.1675e+0 (1.16e-1) -	1.3319e+0 (1.34e-1) -	5.0372e-1 (4.06e-2) =	5.4028e-1 (1.66e-2) -	**4.6222e-1 (3.67e-2) +**	5.0989e-1 (4.59e-3)
UF9	3	1000	1.6891e+0 (1.79e-1) -	1.5143e+0 (1.52e-1) -	7.6722e-1 (1.35e-2) -	7.2356e-1 (4.29e-2) -	6.7230e-1 (9.01e-2) =	**6.6051e-1 (1.66e-2)**
DTLZ1	5	100	1.2498e+3 (2.88e+2) -	1.5054e+3 (3.15e+2) -	3.1214e+2 (4.18e+1) -	2.9975e+2 (7.00e+1) =	3.9461e+2 (3.16e+2) =	**1.6925e+2 (1.38e+2)**
DTLZ1	5	1000	1.3669e+4 (2.30e+3) -	1.4800e+4 (1.52e+3) -	8.3434e+3 (7.54e+2) -	3.0638e+3 (4.04e+2) -	3.4173e+3 (2.04e+3) -	**2.2002e+2 (2.22e+2)**
DTLZ2	5	100	3.1059e+0 (2.86e-1) -	3.1025e+0 (2.49e-1) -	**4.3847e-1 (2.35e-2) +**	6.2749e-1 (3.03e-2) -	6.8261e-1 (3.31e-2) -	4.9801e-1 (4.04e-2)
DTLZ2	5	1000	3.8766e+1 (5.80e+0) -	4.3752e+1 (4.08e+0) -	4.3532e+0 (5.67e-1) -	3.1633e+0 (2.21e-1) -	4.6113e+0 (6.05e-1) -	**8.2565e-1 (3.53e-2)**
WFG2	5	100	9.9013e-1 (8.15e-2) -	9.5370e-1 (7.83e-2) -	5.2601e-1 (2.05e-2) =	5.6887e-1 (4.27e-2) -	5.6467e-1 (2.88e-2) -	**5.0634e-1 (1.48e-2)**
WFG2	5	1000	1.0276e+0 (8.92e-2) -	1.0557e+0 (4.74e-2) -	7.6442e-1 (2.33e-2) -	**6.3079e-1 (3.32e-2) +**	6.6784e-1 (2.48e-2) +	7.2980e-1 (2.70e-2)
WFG8	5	100	1.7562e+0 (2.87e-2) -	1.7962e+0 (4.11e-2) -	1.1548e+0 (1.59e-2) =	1.2181e+0 (2.23e-2) -	1.2151e+0 (1.93e-2) -	**1.1426e+0 (4.87e-3)**
WFG8	5	1000	1.6040e+0 (6.82e-2) -	1.7252e+0 (3.64e-2) -	1.3328e+0 (4.49e-2) =	1.2953e+0 (5.90e-3) =	**1.2133e+0 (1.28e-2) +**	1.3015e+0 (1.09e-2)
+/-/=			0/16/0	0/16/0	2/9/5	1/10/5	3/7/6	

the performance results summarized in Table A.15 further validate that NN-CSO still shows significant superiority over other compared algorithms when solving the problems with super-large-scale dimensionality.

In addition, we also conducted a preliminary study of the performance of LMOEAs [8], [9], [11], [16], [20] and MBEAs [14], [35] in solving the problems in large-scale many-objective environments, i.e., the dimensionality of the objective space is also large-scale. From the IGD results provided in Table A.16 of the Supplementary Material, it can be easily observed that solving the problems in a large-scale many-objective environment is a very large challenge for the existing algorithms, because even after a relatively large number of iterations, their IGD values remain very large, which means that they failed to converge in the large-scale objective space. The reason behind this may be the limited performance of the existing search operators, the environmental selection methods, etc., under a large-scale many-objective environment. Therefore, how to efficiently address problems with super-large-scale many-objective optimization problems is still an open issue that deserves further study.

V. LIMITATIONS

The experimental results presented above validate that the NN model can effectively enhance traditional CSOs in solving a variety of LMOPs and that our proposed NN-CSO has clear advantages over several state-of-the-art LMOEAs and MBEAs. However, our method still suffers from some limitations. First, as analyzed in Section IV-G3, when compared to traditional LMOEAs, our method is somewhat time-consuming as it fails to yield the best results in terms of execution efficiency. Second, as learned from Section IV-G5, the search capability of NN-CSO is not sufficient when solving super-large-scale many-objective problems.

VI. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed an NN-enhanced CSO for solving a variety of LMOPs, called NN-CSO. Specifically, CSO assists the training of the NN model because the samples used for model training can be easily obtained by the pairwise competition of CSO. Then, a well-trained NN model can learn promising evolutionary dynamics that can drive loser particles towards winner particles, consequently boosting the convergence speed of traditional CSOs. Therefore, the combination of CSO and an NN model enables effective and complementary cooperation. The extensive experimental results verified the validity of NN-CSO in solving four different LMOP benchmark suites, showing that the designed NN model can significantly improve the performance of conventional CSOs and the proposed NN-CSO has significant advantages over several state-of-the-art LMOEAs and MBEAs.

This study validated that an evolutionary algorithm combined with a machine learning model can be considered a promising technique for solving complex MOPs. As part of our future research, we plan to adopt a combination of an appropriate machine learning model and evolutionary algorithm according to the characteristics of the target problem. In addition, an emerging distributed GPU-accelerated library [55] is believed to improve the execution efficiency of algorithms.

REFERENCES

[1] Z. Liu, H. Wang, and Y. Jin, "Performance indicator-based adaptive model selection for offline data-driven multiobjective evolutionary optimization," *IEEE Transactions on Cybernetics*, early access, doi: 10.1109/TCYB.2022.3170344.

[2] S. Liu, Q. Lin, J. Li, and K. C. Tan, "A survey on learnable evolutionary algorithms for scalable multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, early access, doi: 10.1109/TEVC.2023.3250350.

[3] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. C. Tan, and Y. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1–34, June. 2021.

[4] K. Li and R. Chen, "Batched data-driven evolutionary multiobjective optimization based on manifold interpolation," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1, pp. 126–140, February. 2023.

[5] L. Li, Q. Lin, Z. Ming, K.-C. Wong, M. Gong, and C. A. C. Coello, "An immune-inspired resource allocation strategy for many-objective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, early access, doi: 10.1109/TSMC.2022.3221466.

[6] Y. Guo, G. Chen, M. Jiang, D. Gong, and J. Liang, "A knowledge guided transfer strategy for evolutionary dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, early access, doi: 10.1109/TEVC.2022.3222844.

[7] H. Wang, L. Jiao, R. Shang, S. He, and F. Liu, "A memetic optimization strategy based on dimension reduction in decision space," *Evolutionary Computation*, vol. 23, no. 1, pp. 69–100, March. 2015.

[8] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, April. 2016.

[9] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 97–112, February. 2018.

[10] C. He, R. Cheng, L. Li, K. C. Tan, and Y. Jin, "Large-scale multiobjective optimization via reformulated decision variable analysis," *IEEE Transactions on Evolutionary Computation*, early access, doi: 10.1109/TEVC.2022.3213006.

[11] L. Ma, M. Huang, S. Yang, R. Wang, and X. Wang, "An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6684–6696, July. 2022.

[12] S. Liu, Q. Lin, Y. Tian, and K. C. Tan, "A variable importance-based differential evolution for large-scale multiobjective optimization," *IEEE Transactions on Cybernetics*, vol. 52, no. 12, pp. 13 048–13 062, December. 2022.

[13] Y. Xu, C. Xu, H. Zhang, L. Huang, Y. Liu, Y. Nojima, and X. Zeng, "A multi-population multi-objective evolutionary algorithm based on the contribution of decision variables to objectives for large-scale multi/many-objective optimization," *IEEE Transactions on Cybernetics*, early access, doi: 10.1109/TCYB.2022.3180214.

[14] S. Liu, J. Li, Q. Lin, Y. Tian, and K. C. Tan, "Learning to accelerate evolutionary search for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 1, pp. 67–81, February. 2023.

[15] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multiobjective optimization based on problem transformation," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 260–275, April. 2018.

[16] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, and X. Yao, "Accelerating large-scale multiobjective optimization via problem reformulation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 949–961, December. 2019.

[17] S. Qin, C. Sun, Y. Jin, Y. Tan, and J. Fieldsend, "Large-scale evolutionary multiobjective optimization assisted by directed sampling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 724–738, August. 2021.

[18] Y. Tian, C. Lu, X. Zhang, F. Cheng, and Y. Jin, "A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6784–6797, July. 2022.

[19] S. Liu, Q. Lin, K.-C. Wong, Q. Li, and K. C. Tan, "Evolutionary large-scale multiobjective optimization: Benchmarks and algorithms," *IEEE Transactions on Evolutionary Computation*, early access, doi: 10.1109/TEVC.2021.3099487.

[20] C. He, R. Cheng, and D. Yazdani, "Adaptive offspring generation for evolutionary large-scale multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 2, pp. 1–13, February. 2022.

[21] W. Hong, K. Tang, A. Zhou, H. Ishibuchi, and X. Yao, "A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 525–537, June. 2019.

[22] Z. Li, Q. Zhang, X. Lin, and H.-L. Zhen, "Fast covariance matrix adaptation for large-scale black-box optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 5, pp. 2073–2083, May. 2020.

[23] 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 Transactions on Cybernetics*, vol. 47, no. 9, pp. 2896–2910, September. 2017.

[24] Z.-J. Wang, Z.-H. Zhan, S. Kwong, H. Jin, and J. Zhang, "Adaptive granularity learning distributed particle swarm optimization for large-scale optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, pp. 1175–1188, March. 2021.

[25] K. Zhang, C. Shen, and G. G. Yen, "Multipopulation-based differential evolution for large-scale many-objective optimization," *IEEE Transactions on Cybernetics*, early access, doi: 10.1109/TCYB.2022.3178929.

[26] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, February. 2015.

[27] W.-X. Zhang, W.-N. Chen, and J. Zhang, "A dynamic competitive swarm optimizer based-on entropy for large scale optimization," in *2016 Eighth International Conference on Advanced Computational Intelligence (ICACI)*, 2016, pp. 365–371.

[28] R. Lan, Y. Zhu, H. Lu, Z. Liu, and X. Luo, "A two-phase learning-based swarm optimizer for large-scale optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 12, pp. 1–10, December. 2021.

[29] X. Wang, K. Zhang, J. Wang, and Y. Jin, "An enhanced competitive swarm optimizer with strongly convex sparse operator for large-scale multi-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, October. 2022.

[30] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multi-objective optimization based on a competitive swarm optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, August. 2020.

[31] S. Liu, Q. Lin, Q. Li, and K. C. Tan, "A comprehensive competitive swarm optimizer for large-scale multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 9, pp. 1–14, September. 2022.

[32] J. Zhang, Z.-h. Zhan, Y. Lin, N. Chen, Y.-j. Gong, J.-h. Zhong, H. S. Chung, Y. Li, and Y.-h. Shi, "Evolutionary computation meets machine learning: A survey," *IEEE Computational Intelligence Magazine*, vol. 6, no. 4, pp. 68–75, November. 2011.

[33] C. He, S. Huang, R. Cheng, K. C. Tan, and Y. Jin, "Evolutionary multiobjective optimization driven by generative adversarial networks (GANs)," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3129–3142, June. 2021.

[34] Z. Wang, H. Hong, K. Ye, G.-E. Zhang, M. Jiang, and K. C. Tan, "Manifold interpolation for large-scale multiobjective optimization via generative adversarial networks," *IEEE Transactions on Neural Networks and Learning Systems*, early access, doi: 10.1109/TNNLS.2021.3113158.

[35] Y. Tian, C. Lu, X. Zhang, K. C. Tan, and Y. Jin, "Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3115–3128, June. 2021.

[36] L. Feng, Y.-S. Ong, S. Jiang, and A. Gupta, "Autoencoding evolutionary search with learning across heterogeneous problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 760–772, October. 2017.

[37] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, August. 2013.

[38] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, April. 2018.

[39] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Readings in Cognitive Science*, vol. 323, no. 6088, pp. 399–421, 1988.

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

[41] Q. Yang, W.-N. Chen, T. Gu, H. Zhang, H. Yuan, S. Kwong, and J. Zhang, "A distributed swarm optimizer with adaptive communication for large-scale optimization," *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3393–3408, July. 2020.

[42] L. Li, Y. Li, Q. Lin, Z. Ming, and C. A. C. Coello, "A convergence and diversity guided leader selection strategy for many-objective particle swarm optimization," *Engineering Applications of Artificial Intelligence*, vol. 115, p. 105249, 2022.

[43] Q. Yang, W.-N. Chen, J. D. Deng, Y. Li, T. Gu, and J. Zhang, "A level-based learning swarm optimizer for large-scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 578–594, August. 2018.

[44] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Computer Science and Informatics*, vol. 26, pp. 30–45, 1996.

[45] M. Li, S. Yang, and X. Liu, "Shift-based density estimation for Pareto-based algorithms in many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 348–365, June. 2014.

[46] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, November. 2017.

[47] A. J. Nebro, J. J. Durillo, and M. Vergne, "Redesigning the JMetal multi-objective optimization framework," in *Proceedings of the Companion*

Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation, 2015, p. 1093–1100.

[48] R. Cheng, Y. Jin, M. Olhofer, and B. sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Transactions on Cybernetics*, vol. 47, no. 12, pp. 4108–4121, December. 2017.

[49] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari *et al.*, "Multiobjective optimization test instances for the CEC 2009 special session and competition," *University of Essex, Colchester, UK and Nanyang technological University, Singapore, special session on performance assessment of multi-objective optimization algorithms, technical report*, vol. 264, pp. 1–30, 2008.

[50] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proceedings of the 2002 Congress on Evolutionary Computation*, 2002, pp. 825–830.

[51] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, October. 2006.

[52] W. Chen, H. Ishibuchi, and K. Shang, "Fast greedy subset selection from large candidate solution sets in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 4, pp. 750–764, 2022.

[53] Alcalá-FdezJ., SánchezL., GarcíaS., J. J. Del, VenturaS., M. Garrellj., OteroJ., RomeroC., BacarditJ., and M. Rivasv., "KEEL: a software tool to assess evolutionary algorithms for data mining problems," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, no. 3, pp. 307–318, May. 2008.

[54] Y. Tian, Y. Feng, X. Zhang, and C. Sun, "A fast clustering based evolutionary algorithm for super-large-scale sparse multi-objective optimization," *IEEE/CAA Journal of Automatica Sinica*, early access, doi: 10.1109/JAS.2022.105437.

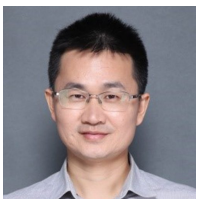
[55] B. Huang, R. Cheng, Y. Jin, and K. C. Tan, "EvoX: A distributed GPU-accelerated library towards scalable evolutionary computation," *arXiv preprint arXiv:2301.12457*, 2023.

Lingjie Li received his B.S. degree from Shandong Technology and Business University, Yantai, China in 2017, and his M.S. degree and Ph.D. degree from the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China in 2020 and 2023, respectively.

He is currently an assistant researcher in Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ). He focuses on research in the area of evolutionary computation, including algorithm researches on multi/many-objective optimization, large-scale optimization, and application researches on feature selection, cloud/edge computing, and remote sensing hyperspectral images.

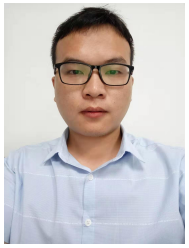
Yongfeng Li received the B.S. degree from Dongguan University of Technology, Dongguan, China in 2020.

He is currently pursuing the Ph.D. degree in College of Computer Science and Software Engineering, Shenzhen University. His current research interests include evolutionary large-scale optimization and machine learning.



Qiuzhen Lin (Member IEEE) received the B.S. degree from Zhaoqing University and the M.S. degree from Shenzhen University, China, in 2007 and 2010, respectively. He received the Ph.D. degree from Department of Electronic Engineering, City University of Hong Kong, Kowloon, Hong Kong, in 2014.

He is currently an associate professor in College of Computer Science and Software Engineering, Shenzhen University. He has published over sixty research papers since 2008. He is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE. His current research interests include artificial immune system, multi-objective optimization, and dynamic system.



Songbai Liu (Member IEEE) received the B.S. degree from Changsha University and the M.S. degree from Shenzhen University, China, in 2012 and 2018, respectively. He received the Ph.D. degree from Department of Computer Sciences, City University of Hong Kong, in 2022.

He is currently an assistant professor in College of Computer Science and Software Engineering, Shenzhen University. His research interests include evolutionary algorithms + machine learning, evolutionary large-scale optimization, and their applications.

Junwei Zhou received the Ph.D. degree in the department of electrical engineering from the City University of Hong Kong. Since then, he has been with Polytechnic University of Turin, Italy, as a visiting scholar, with City University of Hong Kong, Hong Kong, China, as a researcher.

He is currently an Associate Professor with the School of Computer Science and Artificial Intelligence, Wuhan University of Technology, China, a Post-doctoral Researcher with the Pennsylvania State University, USA. His research interests include computer vision, information security and distributed source coding. He has served as PC member of top conferences such as EMNLP 2023, Coling 2022, ACL 2022, and ISSRE 2022.

Zhong Ming received the the Ph.D. degree in Computer Science and Technology from Sun Yat-sen University, Guangzhou, China, in 2003.

He is currently the Executive Director of the Graduate School of Shenzhen University, and a Professor with the National Engineering Laboratory for Big Data System Computing Technology and the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China. His research interests include software engineering and artificial intelligence. He has published more than 200 refereed international conference and journal papers (including 40+ ACM/IEEE Transactions papers). He was the recipient of the ACM TiiS 2016 Best Paper Award and some other best paper awards.

Carlos A. Coello Coello (Fellow, IEEE) received the Ph.D. degree in computer science from Tulane University, New Orleans, LA, USA, in 1996.

He is a Professor (CINVESTAV-3F Researcher) with the Department of Computer Science of CINVESTAV-IPN, Mexico City, Mexico. He has authored and coauthored over 450 technical papers and book chapters. He has also coauthored the book Evolutionary Algorithms for Solving Multi-Objective Problems (Second Edition, Springer, 2007). His publications currently report over 57 600 citations in Google Scholar (his H-index is 95). His research interests include evolutionary multiobjective optimization and constraint-handling techniques for evolutionary algorithms.

Dr. Coello Coello was a recipient of the 2007 National Research Award from the Mexican Academy of Sciences in the area of Exact Sciences, the 2013 IEEE Kiyo Tomiyasu Award, and the 2012 National Medal of Science and Arts in the area of Physical, Mathematical and Natural Sciences. He is currently the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He is a member of the Association for Computing Machinery and the Mexican Academy of Science.