Learning-Guided Cross-Sampling for Large-Scale Evolutionary Multi-Objective Optimization*

Haofan Wang[a,1], Li Chen[a,*], Xingxing Hao[a,*], Rong Qu[b], Wei Zhou[a], Dekui Wang[a] and Wei Liu[c]

[a]School of Information Science and Technology, Northwest University, Xi'an 710127, China
[b]School of Computer Science, University of Nottingham, Nottingham, NG8 1BB, UK
[c]Leiden Institude of Advanced Computer Science, Leiden University, Leiden, 2333 CA, Netherlands

ARTICLE INFO

Keywords:
Evolutionary algorithm
Large-scale multi-objective optimization
Learning-guided
Cross-sampling
Two-level

ABSTRACT

When tackling large-scale multi-objective problems (LSMOPs), the computational budget could be wasted by traditional offspring generators that explore the search space in a nearly directionless manner, impairing the efficiency of many existing algorithms. To address this issue, this paper proposes a novel two-level large-scale multi-objective evolutionary algorithm called LMOEA-LGCS that incorporates neural network (NN) learning-guided cross-sampling for offspring generation in the first level and a layered competitive swarm optimizer in the second level. Specifically, in the first level, two NNs are trained online to learn promising vertical and horizontal search directions, respectively, against the Pareto Set, and then a batch of candidate solutions are sampled on the learned directions. The merit of learning two explicit search directions is to devote the employed NNs to concentrating on separate or even conflicting targets, i.e., the convergence and diversity of the population, thus achieving a good trade-off between them. In this way, the algorithm can thus explore adaptively towards more promising search directions that have the potential to facilitate the convergence of the population while maintaining a good diversity. In the second level, the layered competitive swarm optimizer is employed to perform a deeper optimization of the solutions generated in the first level across the entire search space to increase their diversity further. Comparisons with six state-of-the-art algorithms on three LSMOP benchmarks, i.e., the LSMOP, UF, and IMF, with 2-12 objectives and 500-8000 decision variables, and the real-world problem TREE demonstrate the advantages of the proposed algorithm.

1. Introduction

Multi-objective optimization (MOO) algorithms play a crucial role in solving multi-objective optimization problems (MOPs) in various domains like medicine [1], transportation [2], logistics [3], and industrial production [4, 5], where multiple conflicting objectives always need to be optimized simultaneously. However, problems in real scenarios could be high-dimensional, namely, numerous variables are involved in the objective function that is used to evaluate the performance of systems or methods [6–8]. The goal of MOO is to find a set of Pareto optimal solutions as improving one objective may harm others.

Recent decades have seen flourishing research that leverages evolutionary algorithms to achieve such goals due to their population-based property and implicit parallelism [9, 10]. Among them, increasing attention has been paid to the development of large-scale multi-objective evolutionary optimization algorithms (LMOEAs) that aim to solve large-scale multi-objective optimization problems (LSMOPs) [11–16]. Existing LMOEA can be broadly classified into three categories.

The first category follows a divide-and-conquer manner, where decision variables are grouped into sub-problems and optimized via sharing and cooperation. For instance, MOEA/DVA [17] divides variables into distance and diverse ones via variable analysis, transforming difficult LSMOPs into simpler low-dimensional and easy-to-solve MOPs. LMEA [18] uses an angle-based clustering to separate convergence-related and diversity-related variables, and then a convergence and diversity optimization strategy are employed to optimize them, respectively. LSMOEA/D [19] proposes a decomposition-based adaptive local decision variable analysis method where an adaptive scaling strategy is used to optimize the grouped decision variables, which can balance the convergence and diversity of the population. Other approaches work alike, but some optimize sub-problems in parallel [20] and some sequentially [21]. This type of LMOEA is capable of trading off the population's convergence and diversity well. However, they usually require a large number of evaluations to reasonably analyze the decision variables, which could be a burden when computational budgets are limited.

The second category encompasses problem transformation-based LMOEAs. These algorithms transform the original problem into equivalent sub-problems to simplify their complexity. For example, WOF [22] integrates multiple objective functions into a single objective function via

*This work was supported by the National Natural Science Foundation of China under Grant(No. 62106199), National key research and development projects(No. 2020YFC1523301), the Xi 'an major scientific and technological achievements transformation and industrialization projects (No. 20GXSF0005), Key Research and Development Program of Shaanxi Province(No.2020KW-068), and the General Project of Education Department of Shaanxi Provincial Government under Grant(No.22JK058)

*Corresponding author
haofanwang@stumail.nwu.edu.cn (H. Wang); chenli@nwu.edu.cn (L. Chen); xingxing.hao@nwu.edu.cn (X. Hao); rong.qu@nottingham.ac.uk (R. Qu); mczhouwei12@gmail.com (W. Zhou); Dekui_wang@126.com (D. Wang); w.liu@liacs.leidenuniv.nl (W. Liu)
ORCID(s): 0009-0009-4768-6699 (H. Wang)

[1]This is the first author footnote

weighted sum, which is then optimized using traditional single-objective optimization algorithms. To alleviate the impact of different weights in WOF, WOF-MMOPSO-RDG [23] employs a dynamic grouping strategy [24]. LSMOF [25] proposes a problem reformulation framework in which the decision space is reconstructed by the weight variables, and the objective space is reduced by the indicator function. Due to the reformulated problems having lower dimensionality than the originals, quasi-optimal solutions can be efficiently obtained and subsequently refined by existing LMOEAs. Inspired by LSMOF, LMOEA-DS [26] establishes search directions in the decision space and then directly samples solutions on them to generate candidates. Benefiting from the guidance of search directions, LMOEA-DS exhibits remarkable ability to speed up the convergence of the population. However, as the population evolves, such directions that are bounded by the margins of decision space could limit the flexibility of sampling solutions, thus degrading its effectiveness at the later stage [27].

Different from the above categories that either divide original problems into sub-problems or transform them into simpler ones to reduce the difficulty of LSMOPs, the third category focuses on designing more efficient search strategies or operators to solve LSMOPs directly. For instance, FDV [28] introduces fuzzy concepts to solve LSMOPs by fuzzifying decision variables. DGEA [29] proposes a preselection strategy to select a balanced parent population, which is then used to construct two types of direction vectors for generating promising offspring with good convergence and diversity, respectively. LMOCSO [30], a variant of the traditional competitive swarm optimizer (CSO) [31], specifically designs a new particle updating strategy that updates the particle's position in a two-stage manner to improve its search efficiency. Due to the good performance and extensibility, it is embedded as a module in many LMOEAs[28, 32–35].

Recently, learning-guided LMOEAs have gained increasing interest. For example, ATLMOEA [32] utilizes a neural network (NN) to predict potentially directional improvement information, which then guides a Differential Evolution (DE) operator to search in promising directions. It exhibits remarkable ability in exploration, but performance would degrade as the number of variables increases due to the underutilization of the learned directions. NN-CSO [33] embeds NN into CSO to not only guide the evolution of the loser particles but also to evolve the winner particles. It is shown to improve the performance of CSOs significantly. Lately, ALMOEA [36] extends the learning-based mechanism to existing LMOEAs, such as the MOEA/D, NSGA-II, etc., to further enhance their performance. It is encouraging to see that incorporating learning mechanisms such as a simple NN into the existing LMOEAs could further promote their performance.

Although many LMOEAs proposed in the past decades demonstrate good performance in solving LSMOPs, there remain limitations and areas for improvement. For example, LMOEAs that fall into the divide-and-conquer category are sensitive to the grouping strategy. Moreover, these algorithms would consume significant computational budgets in variable grouping, which will degrade their performance as the problem's dimensionality grows [37]. Besides, problem transformation-based LMOEAs may encounter information loss during transformation, which may make them blind to certain areas of the decision space, thus leading to local optima. Moreover, decomposing and merging solution sets for large-scale instances are also computationally expensive [38]. At last, the improved LMOEAs that employ traditional offspring generators may exhibit random behaviors during search due to the inherent mechanisms of their generators. Therefore, it is imperative to search in the right directions within a limited number of evaluations when solving LSMOPs.

To address these challenges, this paper proposes a novel large-scale multi-objective evolutionary algorithm, termed as LMOEA-LGCS, that incorporates a learning-guided cross-sampling mechanism for offspring generation in the first level and a layered CSO for solution refinement in the second level. The workflow of LMOEA-LGCS is similar to the ATLMOEA [32] but with the following dissimilarities. (1) Two groups of search directions, i.e., the vertical and horizontal search directions against the Pareto set (PS) are learned in LMOEA-LGCS, while ATLMOEA only learns the vertical search directions according to the definitions in this paper. (2) Solutions predicted by NNs in LMOEA-LGCS are used to establish promising search directions, but in ATLMOEA, they are used as components of a DE [39] operator. (3) The learning-guided cross-sampling and the layered CSO are performed sequentially without extra level or stage control parameters, which are opposite in ATLMOEA. The advantages of LMOEA-LGCS will be elaborated in the following contexts. The main contributions of this paper can be summarized as follows:

1) Unlike most learning-based LMOEAs, two NNs are trained in LMOEA-LGCS to learn promising vertical and horizontal search directions, respectively, against the PS in each generation, aiming at trading off the exploration and exploitation of the search space. The merit of training two NNs is that it can avoid interference between learning to improve convergence and learning to improve diversity, which is always the case if a sole NN is used to accomplish these two tasks.

2) To sufficiently utilize the learned directions, LMOEA-LGCS introduces a cross-sampling framework. After the vertical and horizontal search directions are learned, a number of solutions are then sampled on them. The vertical and horizontal search directions are very likely perpendicular or crossed with each other. The sampled solutions can thus complementally cover the search area as much as possible and maintain a balance between convergence and diversity. In addition, the knowledge learned in different directions during the evolution can be transferred between each other via the selected training dataset at each iteration, which can further benefit the evolution.

3)To further refine the solutions obtained from cross-sampling, the layered CSO is employed as a deep optimizer

to exploit the population in the entire space. It differs from the explicitly two-stage paradigm like [32]. In LMOEA-LGCS, the layered CSO is conducted sequentially after cross-sampling in each generation. Thus, there is no need to design level or stage switching strategies, which usually introduce extra control parameters. Compared with six state-of-the-art LMOEAs, the experimental results demonstrate the effectiveness of LMOEA–LGCS not only on LSMOPs but also on large-scale Many-objective optimization problems.

The subsequent sections of this paper are organized as follows. Section 2 provides preliminaries related to LSMOPs, NNs, CSO, and our motivations. Section 3 elaborates on the main framework and implementations of LMOEA-LGCS. Section 4 analyses the parameter settings and presents experimental results. Section 5 provides more discussions on evaluation indicators and practical implementation of a real-world optimization problem. Finally, Section 6 concludes this paper and suggests potential directions for further research.

2. Preliminaries and Motivations

2.1. Large-Scale Multi-Objective Optimization Problems

Without loss of generality, a LSMOP can be mathematically defined as follows [40]:

$$\begin{cases} \min F(X) = \left(f_1(X), f_2(X), \cdots, f_m(X)\right) \\ subject\ to\ X \in \Omega \end{cases} \tag{1}$$

where $X = (x_1, x_2, \cdots, x_n)$ is an n-dimensional decision variable vector in the decision space Ω. Normally, the value of n is equal to or greater than 100 [17, 18, 28]. The $F(X)$ consists of m conflicting objectives, where m is equal to or greater than 2. Due to the inherent conflicts among these objectives, it is not achievable to simultaneously obtain optimal solutions for all objectives. Instead, the goal is to find a set of non-dominated solutions known as the PS, in which the solutions are equally good or incomparable with each other, and the mapping of PS to the objective space via $F(X)$ is called the Pareto front (PF). In this paper, we test the designed algorithm on LSMOPs, UFs, and IMFs benchmarks with m ranging from 2 to 12 and n ranging from 500 to 8000, which are considerably large scale.

2.2. Neural Networks

NNs are computational models inspired by the functioning of biological nervous systems [41]. Generally, an NN consists of interconnected neurons in three main components: the input layer, hidden layers, and output layer, and each layer is connected by neurons. Each neuron receives input data to calculate a weighted linear combination and then uses a nonlinear activation function to modify and output the result. In this paper, the number of neurons in the input layer and output layer is set to be identical to the dimension of the corresponding LSMOPs [36]. To balance the computational cost and the desired accuracy, a hidden layer with 10 neurons is used. The effect of the number of neurons in the hidden layer on the performance of the algorithm will be discussed in Section 4. The mean square error (MSE) is used as the loss function [42], and the Sigmoid activation function is employed [43].

The training process of NNs is shown as Algorithm 1, in which $S_{inferior}$ and $S_{superior}$ are selected training datasets, and the former represents the input, and the latter is the desired output, respectively. For each solution x in the input dataset, its expected output y is randomly selected from the output dataset (Step 2). Then, the MSE loss between the inference of the current NN (y') and the expected output (y) is calculated and used to update NN's parameters (Steps 3-5).

Algorithm 1 Training of NN

Input: $S_{superior}$ (the output data), $S_{inferior}$ (the input data)
Output: the trained NN
1: **for** each solution $x \in S_{inferior}$ **do**
2: randomly choose an expected output $y \in S_{superior}$ /* y is the ground truth */
3: input x into the current NN and get its inference $y^{'}$ /* $y^{'}$ is the predicted value */
4: compute the MSE loss L between $y^{'}$ and y
5: update the parameters of NN based on $\partial L/\partial W$
6: **end for**
7: **return** the trained NN

2.3. Competitive Swarm Optimizer

The CSO [31] is an optimization algorithm for solving multi-objective optimization problems. It is derived from particle swarm optimization (PSO) but works very differently from PSO. Every particle in CSO can be the global best, and the personal best depends on the pairwise competition, and the loser will update its position by learning from the winner. Benefiting from this flat competition relationship among particles, CSO exhibits good ability to maintain the population's diversity but at the cost of slow convergence [32]. In recent developments, LMOCSO [30] uses a new strategy to evaluate the competitive relationship between particles. Specifically, it calculates particles' fitness based on the shift-based density estimation (SDE) [44], and then the loser will update its velocity ($\vec{v}_l$) and position ($\vec{x}_l$) by learning from the winner as follows,

$$\begin{aligned} \vec{v}_l(t+1) &= r_0\vec{v}_l(t) + r_1\left(\vec{x}_w(t) - \vec{x}_l(t)\right) \\ \vec{x}_l(t+1) &= \vec{x}_l(t) + \vec{v}_l(t+1) + r_0\left(\vec{v}_l(t+1) - \vec{v}_l(t)\right) \end{aligned} \tag{2}$$

where $\vec{v}_l(t)$ and $\vec{x}_l(t)$ are the velocity and position of the loser particle in the tth generation, respectively, and $\vec{x}_w(t)$ indicates the position of the winner particle in the tth generation. r_0 and r_1 are random values between 0 and 1. By (2), the loser particle can sufficiently learn from the winner, which ensures a good diversity of the population. However, it still suffers from slow convergence that would greatly degrade its performance on solving LSMOPs with increasing dimensionality [30, 34, 45]. Therefore, to fully exploit particles' potential and ensure a good diversity of the population, [32] divides particles into four layers where

particles in high layers are considered to have higher quality than those in lower layers. Then, particles in low layers can learn from any of them from any higher layers in a CSO-based manner, which can intuitively promote better diversity than those variants that only divide particles into two categories.

Thus, in this paper, we employ the layered CSO from [32] as an auxiliary optimizer of the designed learning-guided cross-sampling to ensure the population's diversity during the search.

2.4. Motivations

With the increased objective and decision variable sizes, generating promising solutions becomes more difficult and directionless for LMOEAs in solving LSMOPs. Among existing LMOEAs, the search direction-based methodologies, such as LMOEA-DS [26] and ATLMOEA [32], have shown advantages in solving LSMOPs. Benefiting from the guiding solutions generated by sampling on search directions defined by chosen representative individuals in the current population, LMOEA-DS exhibits good potential in generating promising offspring with good convergence and diversity. To better guide the evolution in promising search directions, unlike LMOEA-DS, ATLMOEA [32] uses an NN to learn the potentially directional improvement directions, which is dedicated to accelerating the convergence speed. However, the sampling in LMOEA-DS is limited to a few search directions that are established by the chosen individuals, which may not match the PS shape of the solving problems well and result in performance degradation on complex LSMOPs [26]. Moreover, although ATLMOEA [32] employs an NN to learn the potential improvement directions, the predicted promising solutions are only used to assist a DE operator in generating offspring, which is an underutilization. In addition, only the vertical potential improvement direction, as opposed to our proposed algorithm, is learned in ATLMOEA, which may be easily misled by non-convergent training samples. Consequently, its performance is compromised when tackling problems with dimensions larger than 1000.

To address these challenges, in LMOEA-LGCS, at first, two NNs are used to learn two categories of search directions, i.e., the horizontal and the vertical search directions, which are intentionally parallel and perpendicular to the PS, respectively. Cross-sampling is then performed on the learned directions, ensuring that the generated solutions can consistently converge to the PS and maintain a good diversity as well. Compared with LMOEA-DS and ATLMOEA, our method tends to capture the PS shape of the solving problems more adaptively by sampling on two categories of directions. Besides, the predicted solutions of NNs are utilized to establish search directions rather than as candidates merely or as assistance to genetic operators. By this, more promising offspring can be probably produced once correct directions are established. The experimental results in Sections 4 and 5 show that our algorithm is competitive and time-efficient though two NNs are involved.

Moreover, many LMOEA algorithms, such as WOF [22] and DGEA [29], involve two evolutionary levels where one focuses on convergence and the other focuses on diversity to ensure good performance. This has been experimentally proved a useful mechanism in solving LSMOPs. Therefore, we also incorporate a second level in our algorithm. Specifically, in the second level, the layered CSO [32] is utilized to continue optimizing solutions obtained from the first level, namely, the learning-guided cross-sampling, allowing for a more comprehensive exploration of the search space. Note that the layered CSO is conducted sequentially after the cross-sampling in each generation instead of performing as a separate stage as in [22] and [29]. This avoids bringing in additional control parameters.

3. The Proposed Methods

3.1. Main Architecture

Algorithm 2 presents the main framework of LMOEA-LGCS. Firstly, a set of weight vectors, V, is generated uniformly across the objective space [46], and the population is randomly initialized to P (Step 1). Then, the LMOEA-LGCS starts its main evolution loop, which consists of two main levels. In the first level (Steps 3-7), $N/2$ individuals are randomly selected from population P as the initial population Q_v, and another $N/2$ individuals are also randomly selected from population P as the initial population Q_h. Q_v and Q_h are used as the training dataset of NNs that learn horizontal and vertical search directions, respectively (Step 3). The purpose of randomly selecting half of the population is to avoid homogenization of training data and, on the other hand, promote knowledge exchange among learned search directions. Then, vertical search directions are established by the $Learning\_Vertical\_Direction$ procedure, which takes Q_v as input and outputs starting points V_{sp} and directions D_v for vertical sampling at Step 4. Finally, Ns_1 solutions are sampled on vertical search directions that are decided by D_v and V_{sp} at Step 5. Similarly, Ns_2 solutions are sampled on horizontal search directions learned by the $Learning\_horizontal\_Direction$ in Steps 6 and 7. The next subsection provides a detailed description of vertical and horizontal direction learning and the cross-sampling method.

After obtaining $Q_v^{'}$ and $Q_h^{'}$, N solutions are selected from the combination of $Q_v^{'}$, $Q_h^{'}$, and P based on dominance relationship and the crowded environment selection strategy [11] (Step 8), forming population Off as the result of the first level. Solutions in Off are those with high quality that are generated by cross-sampling at the current generation or from the last generation. To remedy the diversity of the population, the second level employs the layered CSO [32] to further optimize them (Step 9) and generate the new offspring population Q. Specifically, in the $Layered\_Competitive\_Swarm\_Optimizer$, the sampled solutions in Off are divided into four equal layers based on reference vector-guided sorting and their SDE fitness[44], after which the solutions in low layers are considered to have worse performance than those in higher layers. Then, the competitive strategy of LMOCSO is adopted to let the solutions in low layers learn from those in higher layers.

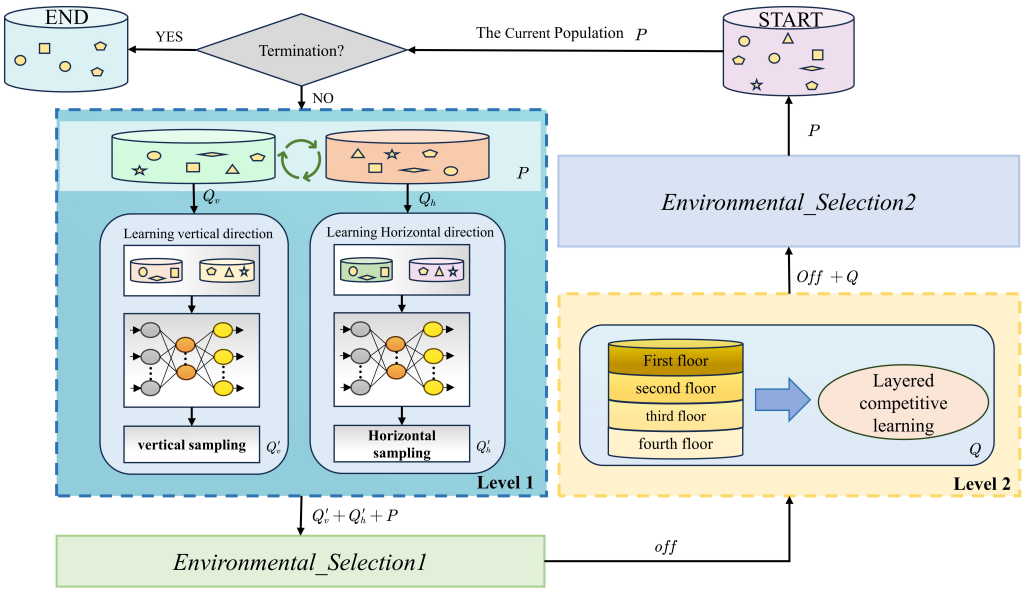

Fig. 1: Overall flowchart of LMOEA-LGCS.

Finally, the reference vector-guided environmental selection proposed in [47] is adopted to select the next generation P from the combination of Off and Q (Step 10). The evolution loop ends until it reaches the computational budget, i.e., the maximum number of function evaluations. Fig. 1 provides the overall flowchart of the proposed LMOEA-LGCS, and the components of Algorithm 2 will be elaborated in the following subsections.

Algorithm 2 Main Framework of the Proposed LMOEA-LGCS.

Input: N (population size), Ns_1 (Number of vertical samples), Ns_2 (Number of Horizontal samples)
Output: P (final population)

1: Generate vector V in a uniform manner and initialize population P
2: **while** termination criterion is not fulfilled **do**
3: Select $N/2$ individuals from population P forming two subpopulations Q_v and Q_h, respectively
4: $(V_{sp}, D_v) = Learning\_Vertical\_Direction(Q_v)$
5: $Q_v^{'} = Direction\_Guided\_Sampling(D_v, V_{sp}, Ns_1)$ // Vertical sampling
6: $(H_{sp}, D_h) = Learning\_Horizontal\_Direction(Q_h)$
7: $Q_h^{'} = Direction\_Guided\_Sampling(D_h, H_{sp}, Ns_2)$ // Horizontal sampling
8: $Off = Environmental\_Selection1(Q_h^{'}, Q_v^{'}, P)$
9: $Q = Layered\_Competitive\_Swarm\_Optimizer(Off, V)$
10: $P = Environmental\_Selection2(Off, Q, V)$
11: **end while**
12: **return** P

3.2. Neural Network-guided Direction Learning

3.2.1. *Learning Vertical Search Directions*

Algorithm 3 outlines the process of learning vertical search directions using a NN. Note that the purpose of learning here is different from that in [32]. Instead of learning the directional solutions for guiding the DE operator, we learn the convergence directions and the diversity directions of the population. Initially, the decision variable values of the solutions in Q_v are normalized using (3) (Step 1), where x_i represents the ith variable of the solution x, and U_i and L_i represents the upper and lower bounds of x_i, respectively.

$$x_i' = \frac{x_i - L_i}{U_i - L_i}, i = 1, 2, \cdots, n \tag{3}$$

Then solutions in Q_v are sorted by non-dominated sorting [48] and equally divided into two sub-populations $S\_superior$ and $S\_inferior$ based on their ranks (Step 2), where $S\_superior$ contains the first half solutions that are closer to the PS and $S\_inferior$ contains the rest half. That is, individuals in $S\_superior$ are better than those in $S\_inferior$.

Afterwards, V_{NN}, namely, the vertical direction learning model, is trained by using the $S\_inferior$ and $S\_superior$ as the input and expected output, respectively (Step 3). Since solutions in $S\_superior$ are better convergent than those in $S\_inferior$, V_{NN} is expected to learn potentially directional improvement information that can guide the population to evolve in promising directions. Therefore, once the V_{NN} is trained, the normalized population Q_v is input to V_{NN} to predict a new potential superior population Q_v' (Step 4), which should have better prospects than Q_v. Next, non-dominated sorting is performed on Q_v and Q_v', respectively, and their first fronts are picked out as Q_{vf} and Q_{vf}', respectively (Step 5), solutions in which will be utilized as the start and end

points of the promising search directions, respectively. That is, the predicted solutions are not directly treated as candidates but used to construct search directions. The reason is that searching in promising directions is more preferential than searching for promising solutions. Intuitively, a promising search direction can promote higher probabilities for finding potential candidates than a sole solution.

To establish a vertical search direction, a start point from Q_{vf} and an end point from Q'_{vf} should be paired somehow. To this end, at first, solutions in Q_{vf} and Q'_{vf} are respectively sorted in descending order according to their objective values, $f_i, f_j, \ldots (1 \leq i < j \leq m)$ (Step 6). Namely, they are firstly sorted by f_i value, then those with the same f_i values are further sorted by f_j value, where $i < j$, etc. This is to avoid intersects of established search directions, thus reducing the probability of duplicated samplings. Note that Q_{vf} and Q'_{vf} hardly contain the same number of solutions, so we first establish $mm = min(|Q_{vf}|, |Q'_{vf}|)$ number of search directions (Steps 7-13), where $|\bullet|$ gives the size of $\bullet$ and $min(\bullet, \circ)$ returns the smaller one between $\bullet$ and $\circ$. Then, the unpaired solutions in Q_{vf} (or Q'_{vf}) will be paired with randomly selected ones from Q'_{vf} (or Q_{vf}), making the total number of search directions to $nn = max(|Q_{vf}|, |Q'_{vf}|)$ (Steps 14-19), where max returns the larger one of its two inputs. The constructed search direction can be represented as (4)

$$d_v^i = {Q_{vf}^i}' - Q_{vf}^i, i = 1, 2, \cdots, nn \tag{4}$$

where ${Q_{vf}^i}'$ and Q_{vf}^i represent the ith solutions in ordered Q'_{vf} and Q_{vf}, respectively. Once the search directions are established, solutions in Q'_{vf} are used as the start points of vertical sampling (step 18), namely, we sample along the established search directions that emit from Q'_{vf}. This is because the solutions in Q'_{vf} have greater potential to promote successful samples. Fig. 2(a) shows examples of vertical search directions, in which X_1, X_2, and X_3 are V_{NN}'s input and X'_1, X'_2, and X'_3 are its selected output, i.e., solutions in Q'_{vf}. The red arrows d_v^1, d_v^2, and d_v^3 represent three search directions that emit from X'_1, X'_2, and X'_3, respectively. It can be seen from Fig. 2(a) that the vertical search directions intend to intersect the PS vertically to provide improvement information of convergence for evolution. Different from most learning-based LMOEAs [32, 36], the trained V_{NN} is used to learn promising search directions instead of just generating offspring. Most likely, a promising search direction can provide more valuable guidance to the evolution than a promising offspring.

3.2.2. Learning Horizontal Search Directions

Algorithm 4 presents the pseudocode of learning horizontal search directions using Q_h. The data preprocessing, training of horizontal search directions learning model H_{NN}, and prediction of potential superior populations follow the same process as Algorithm 3. However, in the construction

Algorithm 3 Learning Vertical Search Directions.

Input: Q_v (The randomly selected sub-populations)
Output: D_v (Set of vertical search directions), V_{sp} (Set of start points of vertical search directions)

1: Normalize decision variable values of solutions in Q_v
2: Divide Q_v into subsets $S\_superior$ and $S\_inferior$ via non-dominated sorting
3: $V_{NN} = Training(S\_superior, S\_inferior)$
4: $Q'_v = V_{NN}(Q_v)$
5: Perform non-dominated sorting on Q_v and Q'_v, and pick out their first front as Q_{vf} and Q'_{vf}, respectively
6: Sort Q_{vf} and Q'_{vf} based on objective values, respectively
7: Initialize the set for storing vertical search directions $D_{v1} = \emptyset, D_{v2} = \emptyset, D_v = \emptyset$
8: $mm = min(|Q_{vf}|, |Q'_{vf}|)$
9: $nn = max(|Q_{vf}|, |Q'_{vf}|)$
10: **for** $i = 1$ to mm **do**
11: Get d_v^i by (4)
12: $D_{v1} = D_{v1} \cup \{d_v^i\}$
13: **end for**
14: T_1 =Randomly select $nn - mm$ individuals from Q'_{vf} if $|Q_{vf}| > |Q'_{vf}|$, otherwise select from Q_{vf}
15: T_2 = Unpaired individuals in Q_{vf} or Q'_{vf}
16: **for** $i = 1$ to $nn - mm$ **do**
17: Calculate $d_v^i = T_1^i - T_2^i$
18: $D_{v2} = D_{v2} \cup \{d_v^i\}$
19: **end for**
20: $V_{sp} = Q'_{vf} \cup T_1$ if $|Q_{vf}| > |Q'_{vf}|$, otherwise Q'_{vf} //Start points of vertical search directions
21: $D_v = D_{v1} \cup D_{v2}$
22: **return** D_v, V_{sp}

procedure of horizontal search directions, it is important to note that Step 3 in Algorithm 4 only requires non-dominated sorting for the predicted population Q'_h, and its first front is picked out as Q'_{hf}. Then, solutions in Q'_{hf} are randomly paired and stored in pairs in Q_p (step 4), and the midpoint of each pair is calculated as the start point of two opposite search directions. By doing this, it is expected to expand the distribution of solutions to margins to enhance the population's diversity. The construction of the horizontal search directions is as (5)

$$\begin{cases} h_p^i = \left(Q_p^{i,1} + Q_p^{i,2}\right)/2 \\ d_+^i = Q_p^{i,1} - h_p^i \\ d_-^i = Q_p^{i,2} - h_p^i \end{cases}, i = 1, 2, \cdots, nn \tag{5}$$

where $Q_p^{i,1}$ and $Q_p^{i,2}$ represent solutions in the ith pair in Q'_{hf}. h_{sp}^i represents the midpoint of $Q_p^{i,1}$ and $Q_p^{i,2}$, which is also the start point of two search directions d_+^i and d_-^i. nn represents the number of pairs in Q'_{hf}. Since each pair of solutions in Q'_{hf} can derive two search directions, a total

of $2nn$ search directions and nn start points are produced and returned by Algorithm 4 (steps 5-12). Fig. 2(b) shows an example of horizontal search directions, in which C_1 is the midpoint of the paired individuals X'_1 and X'_2, and the red arrows indicate the two established search directions d^1_+ and d^1_-. From Fig. 2(b), we can see that the purpose of horizontal search directions is to intersect the PS horizontally to capture its shape as much as possible. In addition, since the horizontal search directions are established by using the predicted solutions that are potentially near the PS of the solved problem, they can be dynamically adjusted as the shape of the PS changes, thus improving the robustness of LMOEA-LGCS in solving different LSMOPs.

Algorithm 4 Learning Horizontal Search Directions.

Input: Q_h (The randomly selected sub-populations)
Output: D_h (Set of horizontal search directions), H_{sp} (Set of start points of horizontal search directions)

1: Train H_{NN} as Algorithm 3 (Step 1-3) does
2: $Q'_h = H_{NN}(Q_h)$
3: Perform non-dominated sorting on Q'_h, and pick out its first front as Q'_{hf}
4: Randomly pair solutions in Q'_{hf} and store those pairs in Q_p
5: $nn = |Q_p|$
6: Initialize the set for storing horizontal search directions $D_h = \emptyset$ and the set for storing starting points $H_{sp} = \emptyset$
7: **for** $i = 1$ to nn **do**
8: Calculate start points h^i_{sp} and search directions d^i_+ and d^i_- using (5) for each ${Q_p}^i$
9: $D_h = D_h \cup \{d^i_+\} \cup \{d^i_-\}$
10: $H_{sp} = H_{sp} \cup \{h^i_{sp}\}$
11: **end for**
12: **return** D_h, H_{sp}

3.3. Learning-guided Cross-sampling

After obtaining the vertical and horizontal search directions, offspring are generated by sampling along them. Algorithm 5 presents the sampling method adopted in this paper. Note that the procedures of sampling on horizontal and vertical search directions are similar, but their directions and starting points of sampling are different. At first, Q_s that stores sampled solutions is initialized to empty (Step 1). Then, for every search direction in D, Ns solutions are sampled by (6) (Steps 2-9).

$$Q^j_i = ||U - L|| * \lambda * \frac{D_i}{||D_i||} + Sp_i, j \in 1, 2, \cdots, Ns \quad (6)$$

where U and L represent the upper and lower bounds of the decision space, respectively, and $||U - L||$ is the length of the decision space. λ is a random number that controls the sampling range, which is a random value in [0, 0.6]. The choice of λ values will be discussed in Section 4. D_i and Sp_i denote the ith sampling direction and start point for sampling, respectively. In case the sampled solution Q^j_i is infeasible, i.e., some variables exceed their ranges, a repair is conducted at Step 6 to resolve this. Fig. 2(c) shows an example that contains both vertical and horizontal search directions in a 3-dimensional decision space, where the blue surface represents the real PS, the green arrow d^i_v and red arrows d^i_+ and d^i_- represent the vertical search directions and the horizontal search directions, respectively. The cross-sampling procedure is to sample solutions on these search directions with the expectation that some of them may intersect the PS or approximate the PS like the yellow stars in Fig. 2(c).

It is important to note that although the sampling range is controlled by parameter λ, the sampled solutions may still be infeasible, i.e., variables in them exceed their allowed ranges. For instance, in Fig. 2(d), the black triangle is an infeasible solution whose x_1 exceeds its lower bound. To solve this, the procedure shown in (7) is used to repair infeasible sampled solutions.

$$Q^{j'}_i = \max\left\{\min\left\{Q^j_i, U\right\}, L\right\} \quad (7)$$

where $Q^{j'}_i$ is the repaired version of Q^j_i, *max* and *min* here returns element-wise maximum and minimum values. The repair procedure in (7) is similar to pulling the variables that exceed their ranges back as the yellow triangle in Fig. 2(d).

Note that the learned vertical and horizontal search directions are very likely perpendicular or crossed with each other, so the sampling on them is also crossed like grids. In addition, since the training sets of H_{NN} and V_{NN} are randomly selected from the current population, there are chances that the solutions used in the vertical search direction construction are generated by the horizontal sampling and vice versa. Thus, the knowledge learned by both directional sampling may be shared with and complementary to each other, which can promote the evolution of the population. That is the reason for calling it cross-sampling. The benefits of cross-sampling will be discussed in the experimental part.

Algorithm 5 Directions Guided Samplings.

Input: D (Set of search directions), S_p (Set of start points), Ns (Number of samples on each search direction), L (Lower bound of the decision space), U (Upper bound of decision space), λ (Parameter that controls the sampling range)
Output: Q_s (Sampling results)

1: $Q_s = \emptyset$
2: **for** $i = 1$ to $|D'|$ **do**
3: **for** $j = 1$ to Ns **do**
4: λ =Randomly generate a number in $(0, 0.6]$
5: Get Q^j_i by (6)
6: Get repaired solution $Q^{j'}_i$ using (7)
7: $Q_s = Q_s \cup \{Q^{j'}_i\}$
8: **end for**
9: **end for**
10: **return** Q_s

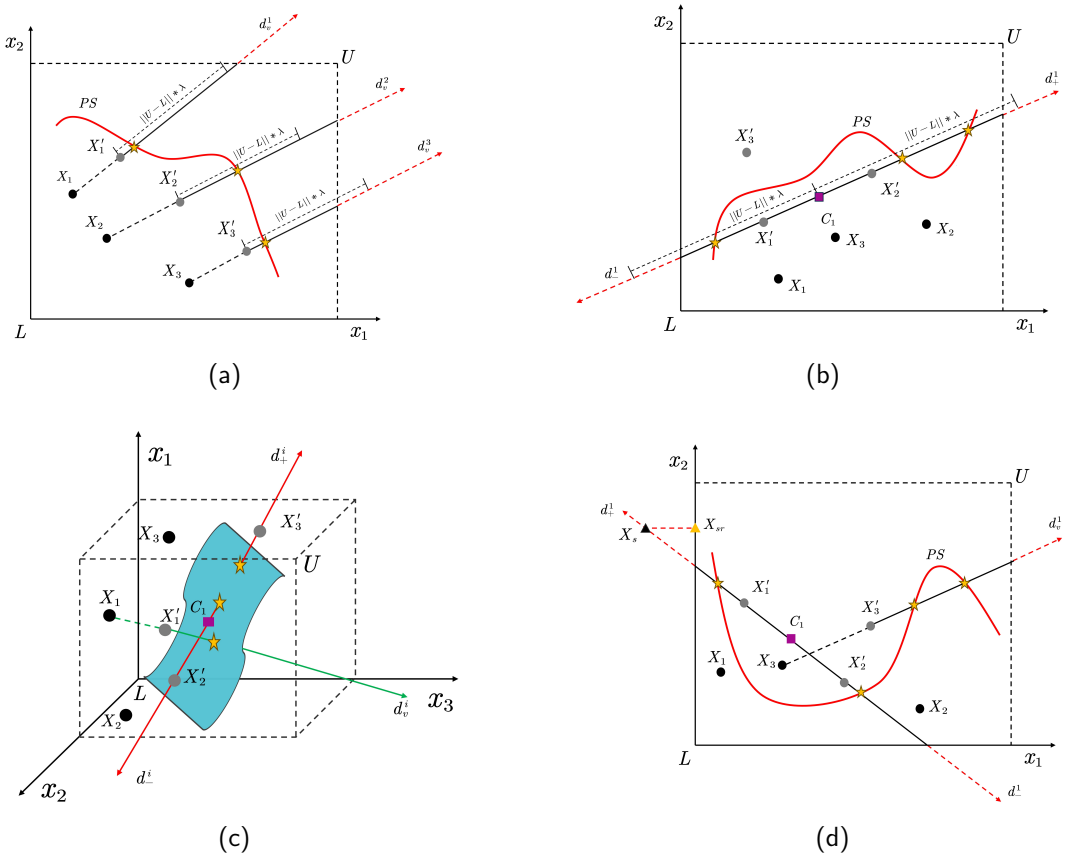

Fig. 2: (a) and (b) are examples of vertical and horizontal search directions, respectively. (c) shows an example of both vertical and horizontal sampling. (d) shows the repair operation for infeasible solutions.

3.4. Layered CSO

To improve the population's diversity, the layered CSO [32] is adopted in the second level to deeply optimize the sampled solutions Off from the first level. Firstly, the objective values of the sampled solutions are normalized as (8)

$$f_i'(x) = \frac{f_i(x) - z_i^*}{z_i^{nadir} - z_i^*}, i = 1, 2, ..., m \tag{8}$$

where z_i^* is the ideal point in the objective space, which is set to the minimal value of the ith objective from all particles, and z_i^{nadir} is the nadir point which is determined as that in [49]. m is the number of objectives. Then, particles are divided into four layers based on reference vector-guided sorting and their SDE fitness[44], after which particles in the high layers are considered to outperform those in the low layers. Then, two particles from different layers are randomly paired, and the one with the lower layer is updated by learning from the one with the higher layer via (2) until all particles are involved. Finally, the reference vector-guided environmental selection [47] is used to select the next generation from the combination of Off and Q by using the uniformly generated weight vector set V, forming the next population P.

4. Experimental Studies

4.1. Benchmarks and Performance Metrics

Three widely used LSMOP benchmarks, i.e., UF [50], IMF [51], and LSMOP [52], are used to evaluate the performance of the proposed LMOEA-LGCS. The number of decision variables, i.e., the dimension, is set to 500, 1000, 2000, 5000, and 8000, while the number of objectives is set to 2, 3, 5, 7, 9, and 12 for LSMOP1-9, and 2 and 3 for UF1-10 and IMF1-10. The inverted generational distance (IGD) [53], the modified inverted generational distanced (IGD+) [54], the coverage over the Pareto front (CPF) [55], and the hypervolume (HV) [56] are used to assess algorithms' performance. To calculate the IGD and IGD+, 10,000 reference points are uniformly sampled from the true PF. A smaller IGD and IGD+ value indicates a better performance [57]. For HV calculation, the reference points (1.1, 1.1, ...) are used, and a larger HV value indicates a better performance, so as the CPF. The maximum number of evaluations is set to 100,000 [58], and the population size N for all algorithms and experiments is set to 300. The other parameters of compared algorithms are set to recommended values in their original articles, in case they are not provided in Table 1.

Table 1
Specific parameter settings of compared algorithms

FDV	
Fuzzy evolution rate	0.8
Step acceleration	0.4
Internal optimization algorithm	LMOCSO
WOF	
Number of groups	4
Groups	ordered
Number of evaluations for original problem	1000
Number of evaluations for transformed problem	500
Internal optimization algorithm	NSGA-II
Crossover probability	0.9
DGEA	
Operation of the environmental selection	RVEA
Number of reference vectors for offspring generation	10
LMOEA-DS	
Cluster number	10
The number of random sampling along each guiding direction	30
Threshold of environment selection	2/3*N
Crossover probability	0.9
LMOCSO	
Penalty parameter α in APD	2
Crossover probability	1.0
ATLMOEA	
Number of neurons	10
Threshold of switching the optimizer ε	0.1
LMOEA-LGCS	
Number of neurons	10
The number of sampling along the horizontal guiding direction	15
The number of sampling along the vertical guiding direction	30
Seach directions range parameter λ	0.6

All algorithms are independently run 20 times on each instance to obtain statistical results. The Wilcoxon rank-sum test [59] with a significance level of 0.05 and the Friedman test that contains Tukey-Kramer post-hoc tests are used to assess the significance of comparisons. In the following context, symbols "+", "-", and "=" indicate the number of instances that the compared algorithm performs significantly better than, significantly worse than, and equally to LMOEA-LGCS, respectively, under the Wilcoxon rank-sum test unless otherwise stated. All experiments are conducted on the PlatEMO [60] in a personal computer with Windows 10 operating system, Intel(R) Core(TM) i7-7700K CPU, and 64 GB RAM.

4.2. Analysis of Parameter Settings

In the proposed LMOEA-LGCS, Ns_1 and Ns_2 control the number of vertical and horizontal samplings, respectively, λ controls the sampling range in each search direction, and the number of neurons Ne affects the complexity of NNs. They may have an impact on the algorithm's performance. Thus, we perform sensitivity analyses on Ns_1, Ns_2, λ, and Ne in this subsection.

4.2.1. Effects of Parameters Ns_1 and Ns_2

In the proposed LMOEA-LGCS, parameters Ns_1 and Ns_2 determine the number of vertical and horizontal samplings, respectively, which may affect LMOEA-LGCS's performance. To obtain an appropriate combination of Ns_1 and Ns_2 values, we examine their settings as 5, 10, 15, 20, 25, 30, 35, and 40 on 2- and 3-objective LSMOP1-9 with 500 and 1000 dimensions. The statistical IGD results are shown in Tables S1-S4 of the Supplementary Material, in which digits in brackets of the first row denote the values of Ns_1 and Ns_2, respectively. Specifically, Tables S1 and S2 present the IGD results of LMOEA-LGCS by varying the value of Ns_1 from 5 to 40 while keeping Ns_2 as 15. The overall trend suggests that [20, 35] is a proper range for Ns_1. Similarly, Tables S3 and S4 of the Supplementary Material present the IGD results of LMOEA-LGCS by varying the value of Ns_2 from 5 to 40 while keeping Ns_1 as 30. From these results, it can be seen that for Ns_2, [10, 30] should be a proper range. Therefore, in this paper, we set ($Ns_1 = 30, Ns_2 = 15$) as a compromise.

4.2.2. Effects of Parameter λ

According to (6), parameter λ controls the range of both vertical and horizontal samplings. A large λ may increase the number of infeasible samples, which will lead to diversity degradation. On the contrary, if it is too small, the learned directions may not be fully exploited. Therefore, we examine the value of λ in [0.1, 4.0]. The experiments are conducted on LSMOP1, 2, 4, 6, and 8 with 2- and 3-objective and 500 and 1000 dimensions. The final results are shown in Figs. S1-S2 and Tables S5 and S6 of the Supplementary Material. From Figs. S1 and S2 of the Supplementary Material, it can be observed that when λ is set to 0.5 and 0.6, there is a significant decrease in IGD values for 2-objective LSMOP1 and LSMOP8, while other instances are insensitive to λ. In addition, from Tables S5 and S6 of the Supplementary Material, it can be seen that the overall trend is the larger the λ, the more the LMOEA-LGCS's performance degrades. The main reason is that a large sampling range will increase the probability of generating infeasible solutions, which, although can be repaired by (7), will compromise the performance of LMOEA-LGCS. Therefore, we set the upper bound of λ to 0.6 as a compromise in experiments.

4.2.3. Effects of the Neuron Number

To evaluate the effects of the neuron numbers Ne in V_{NN} and H_{NN} on the overall performance of LMOEA-LGCS, we examine their values as 1, 5, 10, 15, 20, and 25, on LSMOP1-9 with 2 and 3 objectives and 500 and 1000 dimensions. The statistical IGD results are shown in Table S7 of the Supplementary Material, from which we can see that the more neurons in V_{NN} and H_{NN}, the better the LMOEA-LGCS performs. It is reasonable that more neurons can capture more complex features and thus can provide more accurate predictions, but more training samples and computational resources are needed to train them. Given that finding or tuning an optimal neural model is not our primary target, we set the neuron numbers Ne in V_{NN} and H_{NN} to 10 as a comprise to balance the efficiency and effectiveness. Researchers interested in designing more efficient learning models may refer to Table S7.

4.2.4. Effects of the Population Size

To verify the effect of different population sizes on the algorithm's performance, we examine the performance of all algorithms with population size $N = 100$, 300, and 500 on 2- and 3- objective LSMOP1-9 with 500 dimensions.

The statistical IGD+ results are given in Tables S14-S16 of the Supplementary Material, from which we can see that the population size does have effects on the performance of the algorithms. Generally, for an algorithm, the larger the population size the better the metric values it can achieve. However, it has almost no effect on comparisons among different algorithms. The proposed LMOEA-LGCS exhibits superiority throughout all these different population sizes.

4.3. The Effectiveness of Dual Training Datasets

To avoid the V_{NN} and H_{NN} from learning homogeneous information, we use two randomly selected sub-populations from the current population to train them, respectively. A variant of LMOEA-LGCS termed LMOEA-LGCS-One that utilizes the same sub-population, i.e., the same training dataset to train V_{NN} and H_{NN} is compared against LMOEA-LGCS on 3-objective LSMOP1-9 with dimensions ranging from 500 to 8000. The averaged IGD values over 20 independent runs are given in Table S8 of the Supplementary Material, from which it can be seen that LMOEA-LGCS-One is significantly worse than LMOEA-LGCS on 35 out of 45 instances. This indicates that training V_{NN} and H_{NN} using different datasets has a positive impact on enhancing the algorithm's performance.

4.4. The Effectiveness of Cross-sampling

To validate the effectiveness of the cross-sampling strategy, i.e., the synergy of vertical and horizontal sampling, we compare LMOEA-LGCS with its two variants, namely, LMOEA-LGCS-H that only performs horizontal sampling and LMOEA-LGCS-V that only performs vertical sampling. All other parameters of these two variants are consistent with LMOEA-LGCS. Table S9 of the Supplementary Material presents statistical IGD results on 3-objective LSMOP1-9 with dimensions ranging from 500 to 8000. It can be observed that LMOEA-LGCS outperforms both variants in terms of IGD values on 44 out of 45 instances, which indicates the necessity of the cross-sampling strategy for better performance. Moreover, from Table S9 of the Supplementary Material, we can also see that LMOEA-LGCS-V outperforms LMOEA-LGCS-H in most instances. This outcome is consistent with our original intentions that are shown in Fig. 2 (c), i.e., the purposes of vertical and horizontal search directions are to intersect the PS vertically and horizontally, thus providing potential improvement information for enhancing convergence and diversity, respectively. Hence, LMOEA-LGCS-V could achieve better convergence than LMOEA-LGCS-H. However, it is the synergy of vertical and horizontal sampling that promotes the overall performance of LMOEA-LGCS.

4.5. The Effectiveness of the Two-level Paradigm

To verify the effectiveness of the two-level paradigm, i.e., cross-sampling in the first level and layered CSO in the second level, we compare LMOEA-LGCS with its two variants, namely, LMOEA-LGCS-L1 and LMOEA-LGCS-L2 that only use the first level and second level, respectively. Table S10 of the Supplementary Material presents the statistical results on 3-objective LSMOP1-9 with dimensions ranging from 500 to 8000. From these results we can see that the LMOEA-LGCS outperforms LMOEA-LGCS-L1 and LMOEA-LGCS-L2 on all 45 test problems. Meanwhile, LMOEA-LGCS-L1 gets better results than LMOEA-LGCS-L2 on LSMOP 2, 3, 4, 6, 7, 8, and 9, which indicates the cruciality of the learning-guided cross-sampling. In short, these results indicate the necessity of the two-level paradigm in enhancing the overall performance of LMOEA-LGCS.

4.6. The Effectiveness of Learning

To analyze the impact of leaning on cross-sampling, we compare LMOEA-LGCS with its other variant, LMOEA-CS, that establishes search directions without learning, i.e., it treats the superior solutions of the current population as the expected outputs of the inferior solutions without actually using NNs, and other procedures are identical with LMOEA-LGCS. Table S11 of the Supplementary Material presents the statistical results on 3-objective LSMOP1-9 with 500 to 8000 dimensions. It can be seen that LMOEA-LGCS outperforms LMOEA-CS on 35 out of 45 instances, which demonstrates the importance of NN learning on cross-sampling and LMOEA-LGCS's performance.

4.7. Comparisons with State-of-the-Arts

In this part, we present comparisons between LMOEA-LGCS and other six state-of-the-art LMOEAs, i.e., FDV [28], WOF-NSGA-II [22], DGEA-RVEA [29], LMOEA-DS [26], LMOCSO [30], and ATLMOEA [32] on the three widely-used benchmarks.

4.7.1. Comparisons on LSMOP

In this subsection, we compare LMOEA-LGCS with other six state-of-the-art LMOEAs on 2-, 3-, 5-, 7-, 9-, and 12-objective LSMOP1-9 with up to 8000 dimensions. Table 2 presents the statistical IGD results of 20 independent experiments on 2-objective LSMOP1-9 with dimensions ranging from 500 to 8000. Detailed IGD results on 3-, 5-, 7-, 9-, and 12-objective LSMOP1-9, and HV and IGD+ results on 2- to 12-objective LSMOP1-9 are attached in the Supplementary Material.

From Table 2 it can be observed that LMOEA-LGCS outperforms the other six LMOEAs on most 2-objective LSMOP instances, except that it performs worse than FDV and WOF-NSGA-II on some of the 2-objective LSMOP3, 5, and 7 instances. However, LMOEA-LGCS still outperforms FDV and WOF-NSGA-II overall. Specifically, it performs significantly better than, significantly worse than, and equally to FDV and WOF-NSGA-II on 4/40/1 and 4/30/11 instances, respectively, which demonstrates the superiority of LMOEA-LGCS over FDV and WOF-NSGA-II. In addition, the statistical HV results in Table S18 and the IGD+ results in Tables S29-S38 of the Supplementary Material also illustrate the similar phenomenon. The above comparisons demonstrate that the proposed LMOEA-LGCS is very competitive with the compared algorithms.

To demonstrate the superiority of LMOEA-LGCS visually, Fig. 3 shows the final solutions and IGD convergence

Table 2
Statistical results (mean and standard deviation of IGD metric) on 2-objective LSMOP1-9 with 500, 1000, 2000, 5000, and 8000 dimensions. The best result in each row is highlighted in gray.

Problem	M	D	FDV	WOF-NSGA-II	DGEA-RVEA	LMOEA-DS	LMOCSO	ATLMOEA	LMOEA-LGCS
LSMOP1	2	500	1.7343e-1 (2.18e-2) -	2.4065e-1 (9.77e-2) -	5.7389e-1 (3.01e-1) -	3.0102e-1 (1.28e-2) -	7.5832e-1 (9.25e-2) -	4.0993e-1 (1.02e-1) -	6.7165e-2 (3.09e-2)
	2	1000	3.1026e-1 (4.78e-2) -	2.8297e-1 (1.00e-1) -	4.7435e-1 (2.56e-1) -	3.2207e-1 (8.75e-3) -	1.4039e+0 (1.19e-1) -	4.7483e-1 (1.37e-1) -	5.6769e-2 (2.34e-2)
	2	2000	4.7447e-1 (1.46e-1) -	3.3000e-1 (1.22e-1) -	5.9513e-1 (2.68e-1) -	3.3339e-1 (1.13e-2) -	1.8996e+0 (1.34e-1) -	5.0142e-1 (1.15e-1) -	5.5463e-2 (2.14e-2)
	2	5000	5.8795e-1 (1.45e-1) -	2.9548e-1 (1.42e-1) -	5.7944e-1 (2.85e-1) -	3.3764e-1 (6.48e-3) -	2.0993e+0 (1.28e-1) -	5.0318e-1 (1.10e-1) -	6.1207e-2 (1.92e-2)
	2	8000	6.1289e-1 (1.16e-1) -	2.6849e-1 (1.18e-1) -	6.0417e-1 (2.75e-1) -	3.3959e-1 (8.60e-3) -	2.1137e+0 (9.90e-2) -	5.0954e-1 (1.30e-1) -	5.0883e-2 (1.80e-2)
LSMOP2	2	500	2.3476e-2 (5.67e-3) -	1.4233e-2 (3.41e-3) -	1.7099e-2 (8.12e-3) -	1.2436e-2 (2.25e-4) -	4.1090e-2 (4.17e-4) -	3.1751e-2 (5.47e-3) -	6.0521e-3 (1.88e-3)
	2	1000	1.5873e-2 (3.89e-3) -	9.2048e-3 (2.44e-3) -	9.6374e-3 (1.86e-3) -	7.0981e-3 (1.75e-4) -	2.2415e-2 (3.75e-4) -	2.1548e-2 (1.07e-3) -	3.3162e-3 (6.19e-4)
	2	2000	8.2586e-3 (1.88e-3) -	5.6865e-3 (7.41e-4) -	7.7833e-3 (4.34e-3) -	4.1340e-3 (1.72e-4) -	1.2334e-2 (1.28e-4) -	1.2345e-2 (3.45e-4) -	2.8095e-3 (5.56e-4)
	2	5000	4.4523e-3 (6.84e-4) -	3.4314e-3 (2.95e-4) -	4.8692e-3 (1.80e-3) -	2.4727e-3 (7.92e-5) -	5.7437e-3 (6.44e-5) -	6.0828e-3 (1.42e-4) -	1.8970e-3 (1.63e-4)
	2	8000	3.0694e-3 (3.77e-4) -	2.6063e-3 (1.40e-4) -	4.2090e-3 (1.20e-3) -	1.9046e-3 (4.59e-5) -	4.0834e-3 (5.02e-5) -	4.4147e-3 (8.03e-5) -	1.6589e-3 (5.20e-5)
LSMOP3	2	500	9.8786e-1 (4.31e-2) +	1.4997e+0 (1.40e-1) =	2.0181e+0 (2.14e+0) =	1.5662e+0 (7.54e-4) =	2.7078e+0 (1.98e+0) =	4.0045e+0 (7.52e-1) -	1.5381e+0 (8.30e-2)
	2	1000	1.4506e+0 (7.57e-2) +	1.5103e+0 (1.38e-1) =	3.3600e+0 (6.59e+0) -	1.5735e+0 (3.58e-4) =	7.3681e+0 (4.37e+0) -	6.0789e+0 (5.81e-1) -	1.5584e+0 (4.14e-2)
	2	2000	2.1376e+0 (1.21e-1) -	1.4648e+0 (2.02e-1) =	2.5041e+0 (2.40e+0) -	1.5767e+0 (3.42e-4) -	1.3185e+1 (5.36e+0) -	7.0780e+0 (1.34e+0) -	1.5300e+0 (7.70e-2)
	2	5000	4.0818e+0 (1.24e-1) -	1.4747e+0 (1.67e-1) =	4.5957e+0 (4.26e+0) -	1.5789e+0 (2.85e-4) =	1.8631e+1 (3.54e+0) -	9.2865e+0 (2.69e-1) -	1.5386e+0 (9.87e-2)
	2	8000	5.7092e+0 (1.52e-1) -	1.5152e+0 (1.43e-1) +	5.2175e+0 (4.57e+0) -	1.5792e+0 (4.08e-4) -	2.0674e+1 (4.40e+0) -	9.3931e+0 (2.00e+0) -	1.5270e+0 (1.00e-1)
LSMOP4	2	500	3.9261e-2 (2.41e-3) -	4.2986e-2 (3.97e-3) -	4.9998e-2 (5.63e-3) -	4.3470e-2 (1.39e-3) -	8.3496e-2 (1.42e-3) -	5.2525e-2 (2.45e-3) -	3.1501e-2 (7.76e-3)
	2	1000	2.5979e-2 (1.10e-3) -	2.4408e-2 (3.59e-3) -	2.6582e-2 (2.59e-3) -	2.3346e-2 (9.81e-4) -	5.0318e-2 (4.26e-4) -	3.0256e-2 (9.48e-4) -	1.5104e-2 (3.97e-3)
	2	2000	1.4438e-2 (5.20e-4) -	1.3899e-2 (1.55e-3) -	1.4458e-2 (1.39e-3) -	1.2587e-2 (3.27e-4) -	2.8734e-2 (8.79e-5) -	1.7153e-2 (5.23e-4) -	9.1255e-3 (2.18e-3)
	2	5000	6.4758e-3 (2.74e-4) -	6.9428e-3 (8.32e-4) -	6.8180e-3 (6.27e-4) -	5.6826e-3 (1.74e-4) -	1.3834e-2 (3.79e-5) -	8.2414e-3 (2.63e-4) -	4.6020e-3 (5.68e-4)
	2	8000	4.3403e-3 (1.09e-4) -	4.8837e-3 (5.27e-4) -	5.0553e-3 (7.37e-4) -	3.9323e-3 (1.13e-4) -	9.8517e-3 (2.00e-5) -	5.9866e-3 (2.30e-4) -	3.6571e-3 (3.66e-4)
LSMOP5	2	500	2.9724e-1 (2.12e-1) +	5.0089e-1 (9.88e-2) +	5.7069e+0 (1.45e+0) -	7.4105e-1 (1.18e-4) –	1.6046e+0 (2.68e-1) -	9.1698e-1 (2.33e-1) -	6.6356e-1 (1.70e-1)
	2	1000	9.8518e-1 (2.97e-1) -	6.8227e-1 (9.30e-2) =	6.2644e+0 (9.05e-1) -	7.3139e-1 (5.17e-3) -	3.7770e+0 (3.58e-1) -	9.6264e-1 (2.23e-1) -	6.1689e-1 (1.97e-1)
	2	2000	1.5274e+0 (5.96e-1) -	6.9118e-1 (1.05e-1) =	6.2777e+0 (1.35e+0) -	7.2801e-1 (7.71e-3) =	4.9631e+0 (5.76e-1) -	1.0125e+0 (2.39e-1) -	7.1931e-1 (7.06e-2)
	2	5000	2.0992e+0 (4.14e-1) -	6.5460e-1 (1.30e-1) =	6.5211e+0 (1.07e+0) -	7.3617e-1 (1.21e-3) -	5.3345e+0 (5.60e-1) -	1.1392e+0 (3.54e-1) -	6.4815e-1 (1.63e-1)
	2	8000	1.9810e+0 (5.55e-1) -	5.9356e-1 (1.74e-1) =	6.0342e+0 (1.51e+0) -	7.3288e-1 (7.81e-4) -	5.2664e+0 (4.57e-1) -	1.1264e+0 (2.76e-1) -	6.3977e-1 (1.89e-1)
LSMOP6	2	500	7.7598e-1 (2.31e-3) -	4.5338e-1 (1.53e-1) -	6.7365e-1 (1.63e-1) -	3.1091e-1 (3.14e-2) -	7.7785e-1 (2.31e-2) -	7.4541e-1 (4.90e-2) -	2.2995e-1 (7.96e-2)
	2	1000	7.5977e-1 (1.21e-4) -	4.2563e-1 (1.50e-1) -	6.8377e-1 (1.41e-1) -	3.1184e-1 (9.83e-4) -	7.6850e-1 (4.58e-3) -	7.5551e-1 (1.99e-2) -	2.1297e-1 (7.30e-2)
	2	2000	7.5062e-1 (2.91e-5) -	4.3501e-1 (1.41e-1) -	6.3814e-1 (1.82e-1) -	3.0869e-1 (2.36e-4) -	7.5176e-1 (1.92e-2) -	7.4259e-1 (2.43e-2) -	1.9370e-1 (4.52e-2)
	2	5000	7.4534e-1 (2.24e-6) -	4.3316e-1 (1.90e-1) -	6.6894e-1 (1.59e-1) -	3.0700e-1 (1.34e-4) -	7.4753e-1 (2.03e-4) -	7.4536e-1 (2.77e-5) -	1.8236e-1 (3.34e-2)
	2	8000	7.4409e-1 (6.01e-7) -	3.9026e-1 (1.56e-1) -	6.6432e-1 (1.66e-1) -	3.0691e-1 (1.55e-4) -	7.6039e-1 (1.10e-1) -	7.4410e-1 (1.45e-5) -	1.7483e-1 (3.33e-2)
LSMOP7	2	500	2.4357e+0 (2.53e+0) -	1.4933e+0 (4.59e-2) +	5.8486e+3 (1.17e+3) -	1.5051e+0 (1.12e-3) =	1.5311e+2 (5.81e+1) -	1.5336e+0 (3.68e-2) -	1.5023e+0 (2.10e-2)
	2	1000	8.4463e+0 (3.10e+1) =	1.4861e+0 (8.13e-2) +	6.1710e+3 (1.29e+3) -	1.5116e+0 (4.51e-4) +	1.9290e+3 (6.52e+2) -	2.4518e+0 (4.13e+0) -	1.5154e+0 (2.94e-3)
	2	2000	1.5247e+0 (6.46e-3) -	1.4774e+0 (9.75e-2) =	6.8549e+3 (1.66e+3) -	1.5147e+0 (2.08e-4) =	4.2553e+3 (9.91e+2) -	1.5258e+0 (4.34e-3) -	1.5020e+0 (6.53e-2)
	2	5000	1.2980e+2 (3.02e+2) -	1.4861e+0 (7.88e-2) =	8.0426e+3 (2.41e+3) -	1.5165e+0 (3.36e-4) =	6.3196e+3 (1.21e+3) -	1.7937e+0 (1.19e+0) -	1.4907e+0 (8.30e-2)
	2	8000	7.0119e+2 (1.31e+3) -	1.4907e+0 (6.96e-2) =	6.7244e+3 (1.98e+3) -	1.5168e+0 (2.00e-4) =	6.9149e+3 (1.91e+3) -	1.4927e+1 (4.52e+1) -	1.4987e+0 (8.56e-2)
LSMOP8	2	500	6.5179e-2 (8.11e-3) +	2.9175e-1 (1.55e-1) -	3.7444e+0 (1.32e+0) -	7.4209e-1 (2.28e-16) -	1.2158e+0 (2.02e-1) -	7.2374e-1 (5.25e-2) -	1.7053e-1 (2.41e-2)
	2	1000	2.3554e-1 (3.27e-2) -	4.0623e-1 (2.40e-1) -	4.4161e+0 (1.19e+0) -	7.4209e-1 (2.28e-16) -	2.8131e+0 (3.44e-1) -	7.8885e-1 (6.97e-2) -	1.7783e-1 (2.11e-2)
	2	2000	7.3040e-1 (3.99e-1) -	4.5923e-1 (2.50e-1) -	4.3171e+0 (1.42e+0) -	7.4209e-1 (2.28e-16) -	3.6422e+0 (4.09e-1) -	8.0270e-1 (8.86e-2) -	1.7424e-1 (2.26e-2)
	2	5000	1.0686e+0 (3.78e-1) -	3.9972e-1 (2.22e-1) -	4.8364e+0 (1.01e+0) -	7.4209e-1 (2.28e-16) -	4.1442e+0 (5.03e-1) -	8.1857e-1 (9.33e-2) -	1.9233e-1 (2.90e-2)
	2	8000	1.0784e+0 (4.73e-1) -	2.7995e-1 (1.77e-1) -	4.9856e+0 (1.08e+0) -	7.4209e-1 (2.28e-16) -	4.2059e+0 (3.26e-1) -	8.3073e-1 (1.06e-1) -	1.7098e-1 (2.13e-2)
LSMOP9	2	500	1.3169e-1 (2.09e-2) -	5.7687e-1 (3.08e-1) -	7.3588e+0 (1.52e+0) -	6.2162e-1 (1.51e-1) -	7.7023e-1 (2.03e-1) -	5.5924e-1 (7.62e-3) -	5.5001e-2 (1.87e-2)
	2	1000	1.5174e-1 (8.45e-2) -	4.8606e-1 (2.84e-1) -	1.1990e+1 (3.10e+0) -	6.1970e-1 (3.40e-2) -	3.1201e+0 (9.11e-1) -	5.2345e-1 (3.60e-3) -	3.4666e-2 (1.15e-2)
	2	2000	1.3141e+0 (8.36e-1) -	4.5863e-1 (3.04e-1) -	1.2708e+1 (3.01e+0) -	5.6238e-1 (1.23e-1) -	6.4859e+0 (1.85e+0) -	5.3779e-1 (1.33e-1) -	2.8888e-2 (8.85e-3)
	2	5000	4.0359e+0 (1.67e+0) -	3.6409e-1 (3.66e-1) -	1.3541e+1 (1.65e+0) -	5.7577e-1 (1.13e-1) -	9.2853e+0 (1.83e+0) -	6.6732e-1 (3.58e-1) -	2.1942e-2 (9.43e-3)
	2	8000	5.6058e+0 (2.55e+0) -	3.3805e-1 (3.02e-1) -	1.4038e+1 (3.29e+0) -	5.8259e-1 (5.79e-2) -	9.1977e+0 (2.53e+0) -	8.3119e-1 (4.66e-1) -	1.9904e-2 (7.54e-3)
+/-/=			4/40/1	4/30/11	0/44/1	1/36/8	0/44/1	0/45/0	

curves of all compared algorithms on 3-objective LSMOP8 with 500 dimensions under 100,000 evaluation budgets. The shaded region represents the true PF, and the green dots represent the final solutions obtained by the algorithms. From Fig. 3(a) to (g), it can be seen that LMOEA-LGCS can converge to the PF uniformly while others exhibit either weak convergence or poor diversity. In addition, we also draw the final solutions of 2- and 3-objective LSMOP9 with 8000 dimensions, which has irregular PF and is considered a complex instance [52], as Figs. S6 and S7 of the Supplementary Material, respectively. From Fig. S6, we can see that LMOEA-LGCS converges better than other compared algorithms and distributes solutions well along the PF. Fig. S7 shows that none of the compared algorithms can converge to all disconnected parts of the PFs of the 3-objective LSMOP9. However, LMOEA-LGCS exhibits the best convergence and diversity among them. This is probably because they fail to search in the right directions, leading to a waste of computational resources and a lack of a proper balance between convergence and diversity of the population. Furthermore, by comparing the IGD and HV convergence curves in Fig. 3(h) and Fig. S3 of the Supplementary Material, respectively, it can be seen that LMOEA-LGCS converges the fastest compared to other algorithms. The possible reasons are that the learning-guided cross-sampling can find promising search directions that can provide valuable improvement information for the evolution of the population and can well trade off the exploration and exploitation by incorporating with the layered CSO.

Furthermore, Table 3 summarizes the statistical IGD results (detailed data can be found in the Supplementary Material) of all compared algorithms on 3-, 5-, 7-, 9- and 12-objective LSMOP1-9 with up to 8000 dimensions. From Table 3, we can see that LMOEA-LGCS still has overall better performance than the other six state-of-the-art in these large-scale many-objective problems. Moreover, we summarize the statistical HV results on all test instances in Table S12 and the IGD+ results in Tables S63 and S64 using Friedman test and Wilcoxon rank-sum test, respectively, of the Supplementary Material, which also demonstrates the superiority of LMOEA-LGCS. Moreover, to quantitatively compare the diversity of all algorithms, we calculate the CPF of their final solutions on 500-dimensional LSMOP1-9 with 2, 3, 5, 7, 9, and 12 objectives as Table S17 of the Supplementary Material. As can be seen that the proposed LMOEA-LGCS exhibits better performance in diversity than other competitors.

Additionally, Fig. S5 of the Supplementary Material presents the boxplots of the IGD values of all compared algorithms on 12-objective LSMOP1-9 with 8000 dimensions. For clearer visualization, the IGD values are processed with logarithm. It can be seen that LMOEA-LGCS achieves the best median results on seven test functions, i.e., LSMOP1, 3, 5, 6, 7, 8, and 9, and gains competitive results on LSMOP2 and 4. Overall, compared with other algorithms, LMOEA-LGCS exhibits relatively robust performance.

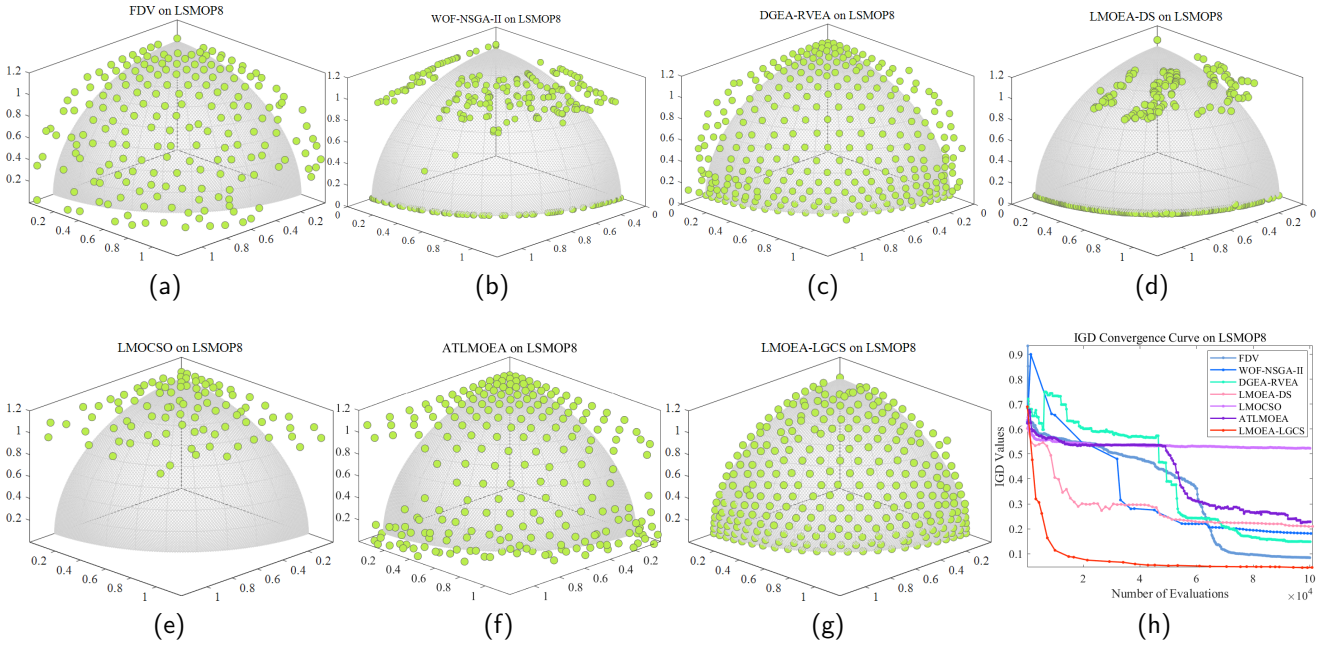

Fig. 3: (a) - (g) Final solutions obtained by all compared algorithms on 3-objective LSMOP8 with 500 dimensions. (h) IGD convergence curves on 3-objective LSMOP8 with 500 dimensions.

To further evaluate the efficiency of LMOEA-LGCS, the average runtime over 20 independent runs of all compared algorithms on 2-objective LSMOP1-9 with 500 variables under a 100,000 evaluation budget is shown in Fig. S4 of the Supplementary Material. We can see that the average runtime of LMOEA-LGCS is medium among all compared algorithms. In addition, what is interesting is that it consumes less average runtime than ATLMOEA, which is also learning-based, in all tested instances, and spends nearly the same time with LMOEA-DS, which is purely directed sampling-based. This demonstrates that the training of two NNs, i.e., the V_{NN} and H_{NN}, does not bring much computational overhead to LMOEA-LGCS but promotes its overall performance.

Additionally, to verify the effect of using the time consumption as the termination condition on comparisons of algorithms, we set the time budget to 40 seconds for all algorithms and keep other parameters untouched. Comparisons in terms of IGD+ on 500-dimensional 2-objective LSMOPs are given in Table S13 of the Supplementary Material. As can be seen that the performance of LMOEA-LGCS is still better than that of other six state-of-the-art LMOEAs.

4.7.2. Comparisons on UF and IMF

For a more comprehensive understanding of LMOEA-LGCS' performance, we also test it on the benchmarks UF1-10 [50] and IMF1-10 [51]. The number of dimensions of instances in both UFs and IMFs is set to 500, 1000, 2000, 5000, and 8000, while the number of objectives is set to 2 and 3. The computational budget and parameter settings in HV, IGD, and IGD+ metrics are identical to those in the LSMOP benchmark. The statistical HV, IGD, and IGD+ results on UFs are shown in Tables S55, S56, S59 and S61

Table 3
The summary of statistical IGD results on 500-, 1000-, 2000-, 5000-, and 8000-dimensional 3-, 5-, 7-, 9- and 12-objective LSMOP1-9. The number before the left backslash represents the number of instances in all 45 instances that LMOEA-LGCS performs significantly better than (+), significantly worse than (-), and equally to (=) the compared algorithm, respectively.

	LMOEA-LGCS	FDV	WOF-NSGA-II	DGEA-RVEA	LMOEA-DS	LMOCSO	ATLMOEA
3	+	41/45	42/45	45/45	43/45	45/45	45/45
	=	3/45	3/45	0/45	2/45	0/45	0/45
	-	1/45	0/45	0/45	0/45	0/45	0/45
5	+	35/45	45/45	38/45	43/45	45/45	37/45
	=	2/45	0/45	1/45	2/45	0/45	1/45
	-	8/45	0/45	6/45	0/45	0/45	7/45
7	+	35/45	43/45	42/45	45/45	40/45	34/45
	=	4/45	1/45	3/45	0/45	1/45	10/45
	-	6/45	1/45	0/45	0/45	4/45	1/45
9	+	35/45	45/45	37/45	43/45	45/45	35/45
	=	0/45	0/45	2/45	1/45	0/45	0/45
	-	10/45	0/45	6/45	1/45	0/45	10/45
12	+	35/45	45/45	37/45	43/45	45/45	35/45
	=	1/45	0/45	8/45	0/45	0/45	3/45
	-	9/45	0/45	0/45	2/45	0/45	7/45

of the Supplementary Material, respectively. From these tables we can see that LMOEA-LGCS performs significantly better than FDV, WOF-NSGA-II, DGEA-RVEA, LMOEA-DS, LMOCSO, and ATLMOEA on 42/50, 34/50, 50/50, 34/50, 50/50, and 49/50, respectively, in terms of IGD metric, and on 36/50, 28/50, 40/50, 25/50, 40/50, and 40/50 instances, respectively, in terms of HV metric, and on 37/50, 1/50, 50/50, 9/50, 48/50, and 35/50 instances, respectively, in terms of IGD+ metric with Friedman test, and on 41/50, 30/50, 50/50, 31/50, 50/50, and 49/50 instances, respectively, in terms of IGD+ metric with Wilcoxon rank-sum test. Although WOF-NSGA-II shows competitiveness with LMOEA-LGCS on UF benchmark in terms of the Friedman test, the summarized

results in Tables S63 and S64 of the Supplementary Material show that LMOEA-LGCS has overall superiority on these three benchmarks.

Similarly, Tables S57, S58, S60 and S62 of the Supplementary Material that present statistical IGD, HV, and IGD+ results on IMFs, respectively, also show that LMOEA-LGCS has better overall performance than compared algorithms.Specifically, LMOEA-LGCS performs significantly better than FDV, WOF-NSGA-II, DGEA-RVEA, LMOEA-DS, LMOCSO, and ATLMOEA on 49/50, 31/50, 50/50, 30/50, 50/50, 45/50, instances, respectively, in terms of IGD metric, and on 47/50, 32/50, 47/50, 35/50, 47/50, 47/50 instances, respectively, in terms of HV metric, and on 41/50, 7/50, 47/50, 20/50, 48/50, and 35/50, instances respectively, in terms of IGD+ metric with Friedman test, and on 50/50, 39/50, 50/50, 36/50, 50/50, and 50/50 instances, respectively, in terms of IGD+ metric with Wilcoxon rank-sum test. All these results demonstrate the superiority of LMOEA-LGCS on IMF benchmarks.

Thus, through the above experimental results, we can that the proposed LMOEA-LGCS is very competitive in solving LSMOPs.

5. More Discussions

In this subsection, we will discuss how reference points in IGD and HV, and different computational budgets affect the comparisons, and an application on real cases.

5.1. Analysis of Reference Points in IGD and HV

To verify how different reference points in HV affect the comparisons, we reevaluate all compared algorithms on 2-, 3-, 5-, 7-, 9-, and 12-objective LSMOP1-9 with 500 and 1000 dimensions with reference points as (1.5, 1.5, ...) and (2.0, 2.0, ...). The statistical results are shown in Tables S39-S42 of the Supplementary Material, from which we can see that different reference points in HV calculation have little influence on the overall comparisons. LMOEA-LGCS still shows competitive performance compared with other algorithms among all these different settings.

Similarly, to verify the effect of the number of reference points in IGD on algorithm's performance, we reevaluate all algorithms on LSMOP1-9 with 2-12 objectives and 500 and 1000 dimensions with $100,000$ and $1,000,000$ reference points in IGD calculation. The statistical results are shown in Tables S43-S46 of the Supplementary Material. It can be seen that there is no significant impact on performance comparisons when various numbers of reference points are used in IGD calculation. The LMOEA-LGCS is also shown to be competitive with other algorithms among all these settings.

5.2. Analyses of the Computational Budget

To verify how different amounts of computational budget can affect the comparisons, we compare all algorithms on 2- and 3-objective LSMOP1-9, UF1-10, and IMF1-10 with 500, 1000, and 3000 dimensions under 10,000 and 1,000,000 function evaluations. The detailed IGD results with 10,000 and 1,000,000 function evaluations are shown in Tables S47-S52 of the Supplementary Material, and Tables S53 and S54 of the Supplementary Material present summaries of these results. Interestingly, LMOEA-DS exhibits competitive performance under 10,000 function evaluations, but when it increases to 1,000,000, it is the FDV that shows competitive performance. These indicate that LMOEA-DS has relatively fast convergence at early stages but may stagnate in late stages, while FDV is the opposite. Actually, Fig. 3(h) tells the same conclusions. However, although LMOEA-DS and FDV perform competitively under 10,000 and 1,000,000 function evaluations, respectively, LMOEA-LGCS still has overall better performance than others under both computational budgets.

5.3. Applications on Real Cases

The Ratio error (RE) estimation of the voltage transformers (VTs) is important in modern power delivery systems [61]. In [61], the time-varying RE estimation (TREE) problems are formulated as LSMOPs, where three types of benchmarks are constructed considering the involved data, i.e., TREE1, TREE2, and TREE3 are type 1 that only involve primary voltage values, TREE4 and TREE5 are type 2 that involve both primary and secondary voltage values, and TREE6 is type 3 that involves both voltage and phase angle values. Among them, TREE1-5 and TREE6 consist of 2 and 3 objectives, respectively.

In experiments, the parameters of all algorithms are consistent with the above. The experimental results are shown in Table S65 of the Supplementary Material, from which we can see that LMOEA-LGCS outperforms all compared algorithms except LMOEA-DS. However, it is very competitive with LMOEA-DS. Specifically, LMOEA-LGCS is good at tackling the type 1 case but shows inferior performance on instances of the other two types that involve more variety of variables compared with LMOEA-DS. This indicates that LMOEA-LGCS may still lack the power to handle interacted decision variables.

Additionally, to verify the effect of different reference points of HV on the comparisons of different algorithms, we additionally calculate HV values with reference points as (1.5,1.5, ...) and (2.0, 2.0, ...) for all TREE instances. Results are provided in Tables S66 and S67 of the Supplementary Material, respectively. As can be seen from Table S65, HV results by using the reference point (1.1,1.1, ...), and Tables S66 and S67 that for a specific algorithm and an instance, the larger the reference point, the larger the HV values. This is consistent with the intuition since larger reference points mean farther distances to the obtained solutions. However, different reference points have no significant effect on comparisons of different algorithms.

6. Conclusions

In this paper, we proposed a learning-guided cross-sampling methodology and a two-level evolution paradigm for large-scale multi/many-objective optimization. The learning procedure learns two NNs, where one learns the vertical

search directions, and the other learns the horizontal search directions. Then, in the first level, solutions are sampled along the learned search directions to enhance both the convergence and diversity of the population. In the second level, a layered CSO is employed to optimize the sampled solutions further, aiming to preserve good diversity.

To validate the effectiveness of the designed algorithm, extensive comparisons were conducted on three widely-used LSMOP benchmarks, i.e., the LSMOP, IMF, and UF, with up to 8000 dimensions and 12 objectives, and a real case TREE. The statistical results demonstrate that the proposed algorithm exhibits overall advantages in solving not only multi-objective problems but also many-objective problems on a considerably large scale. Additionally, ablation experiments on verifying various parameter settings, the effectiveness of dual training datasets, cross-sampling, two-level paradigm, and NN learning were conducted. Moreover, we also analyzed the effect of reference points in IGD and HV metrics and the settings of computational budget on LMOEA-LGCS's performance. All those experimental results revealed that the synergy of the designed components in LMOEA-LGCS makes it competitive, and its performance is relatively robust regardless of the variety of computational budget and evaluation metrics.

Although we employed a layered CSO as a second-level optimizer to further optimize the sampled solutions in LMOEA-LGCS, the learning-guided cross-sampling approach is extendable to existing LMOEAs. Therefore, in future works, we will focus on embedding the proposed learning-guided cross-sampling into existing search frameworks. Moreover, to catch up with the fast development of Deep Learning (DL), it is urgent to take advantage of either mature DL technologies or emerging ones like the Large Language Model (LLM) [62] to promote the practicability of swarm-based methodologies.

References

[1] J. Liang, Z. Hu, Z. Li, K. Qiao, W. Guo, Multi-objective optimization based network control principles for identifying personalized drug targets with cancer (2023). arXiv:2306.13349.

[2] T. Guo, Y. Mei, K. Tang, W. Du, A knee-guided evolutionary algorithm for multi-objective air traffic flow management, IEEE Transactions on Evolutionary Computation (2023). doi:10.1109/TEVC.2023.3281810.

[3] X. Zhang, K. Zhou, H. Pan, L. Zhang, X. Zeng, Y. Jin, A network reduction-based multiobjective evolutionary algorithm for community detection in large-scale complex networks, IEEE Transactions on Cybernetics 50 (2) (2020) 703–716. doi:10.1109/TCYB.2018.2871673.

[4] S. Wang, L. Wang, An estimation of distribution algorithm-based memetic algorithm for the distributed assembly permutation flowshop scheduling problem, IEEE Transactions on Systems, Man, and Cybernetics: Systems 46 (1) (2016) 139–149. doi:10.1109/TSMC.2015.2416127.

[5] Y. Mei, X. Li, X. Yao, Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems, IEEE Transactions on Evolutionary Computation 18 (3) (2014) 435–449. doi:10.1109/TEVC.2013.2281503.

[6] L. Li, C. He, R. Cheng, H. Li, L. Pan, Y. Jin, A fast sampling based evolutionary algorithm for million-dimensional multiobjective optimization, Swarm and Evolutionary Computation 75 (2022) 101181. doi:https://doi.org/10.1016/j.swevo.2022.101181.

[7] C. He, R. Cheng, Population sizing of evolutionary large-scale multi-objective optimization, in: Evolutionary Multi-Criterion Optimization, Springer International Publishing, Cham, 2021, pp. 41–52.

[8] C. Huang, L. Li, C. He, R. Cheng, X. Yao, Adaptive multiobjective evolutionary algorithm for large-scale transformer ratio error estimation, Memetic Computing 14 (2) (2022) 237–251. doi:10.1007/s12293-022-00368-7.

[9] Y. Tian, L. Si, X. Zhang, R. Cheng, C. He, K. Tan, Y. Jin, Evolutionary large-scale multi-objective optimization: A survey, ACM Computing Surveys 54 (06 2021). doi:10.1145/3470971.

[10] Z. Zhan, L. Shi, K. C. Tan, J. Zhang, A survey on evolutionary computation for complex continuous optimization, Artificial Intelligence Review 55 (2021) 59 – 110.

[11] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197. doi:10.1109/4235.996017.

[12] T. O. Olowu, H. Jafari, M. Moghaddami, A. I. Sarwat, Multiphysics and multiobjective design optimization of high-frequency transformers for solid-state transformer applications, IEEE Transactions on Industry Applications 57 (1) (2021) 1014–1023. doi:10.1109/TIA.2020.3035129.

[13] Y. Li, Z. Ni, T. Zhao, M. Yu, Y. Liu, L. Wu, Y. Zhao, Coordinated scheduling for improving uncertain wind power adsorption in electric vehicles—wind integrated power systems by multiobjective optimization approach, IEEE Transactions on Industry Applications 56 (3) (2020) 2238–2250. doi:10.1109/TIA.2020.2976909.

[14] A. Rodríguez-Molina, M. G. Villarreal-Cervantes, E. Mezura-Montes, M. Aldape-Pérez, Adaptive controller tuning method based on online multiobjective optimization: A case study of the four-bar mechanism, IEEE Transactions on Cybernetics 51 (3) (2021) 1272–1285. doi:10.1109/TCYB.2019.2903491.

[15] J. Wang, Y. Zhou, Y. Wang, J. Zhang, C. L. P. Chen, Z. Zheng, Multiobjective vehicle routing problems with simultaneous delivery and pickup and time windows: Formulation, Instances, and Algorithms, IEEE Transactions on Cybernetics 46 (3) (2016) 582–594. doi:10.1109/TCYB.2015.2409837.

[16] C. Juang, T. B. Bui, Reinforcement neural fuzzy surrogate-assisted multiobjective evolutionary fuzzy systems with robot learning control application, IEEE Transactions on Fuzzy Systems 28 (3) (2020) 434–446. doi:10.1109/TFUZZ.2019.2907513.

[17] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, M. Gong, A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables, IEEE Transactions on Evolutionary Computation 20 (2) (2016) 275–298. doi:10.1109/TEVC.2015.2455812.

[18] X. Zhang, Y. Tian, R. Cheng, Y. Jin, A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization, IEEE Transactions on Evolutionary Computation 22 (1) (2018) 97–112. doi:10.1109/TEVC.2016.2600642.

[19] L. Ma, M. Huang, S. Yang, R. Wang, X. Wang, An adaptive localized decision variable analysis approach to large-scale multiobjective and many-objective optimization, IEEE Transactions on Cybernetics 52 (7) (2022) 6684–6696. doi:10.1109/TCYB.2020.3041212.

[20] H. Chen, X. Zhu, W. Pedrycz, S. Yin, G. Wu, H. Yan, PEA: Parallel evolutionary algorithm by separating convergence and diversity for large-scale multi-objective optimization, in: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), 2018, pp. 223–232. doi:10.1109/ICDCS.2018.00031.

[21] H. Chen, R. Cheng, J. Wen, H. Li, J. Weng, Solving large-scale many-objective optimization problems by covariance matrix adaptation evolution strategy with scalable small subpopulations, Information Sciences 509 (2020) 457–469. doi:https://doi.org/10.1016/j.ins.2018.10.007.

[22] H. Zille, H. Ishibuchi, S. Mostaghim, Y. Nojima, A framework for large-scale multiobjective optimization based on problem transformation, IEEE Transactions on Evolutionary Computation 22 (2) (2018) 260–275. doi:10.1109/TEVC.2017.2704782.

[23] R. Liu, J. Liu, Y. Li, J. Liu, A random dynamic grouping based weight optimization framework for large-scale multi-objective optimization problems, Swarm and Evolutionary Computation 55 (2020) 100684. doi:https://doi.org/10.1016/j.swevo.2020.100684.

[24] A. Song, Q. Yang, W. Chen, J. Zhang, A random-based dynamic grouping strategy for large scale multi-objective optimization, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 468–475. doi:10.1109/CEC.2016.7743831.

[25] C. He, L. Li, Y. Tian, X. Zhang, R. Cheng, Y. Jin, X. Yao, Accelerating large-scale multiobjective optimization via problem reformulation, IEEE Transactions on Evolutionary Computation 23 (6) (2019) 949–961. doi:10.1109/TEVC.2019.2896002.

[26] S. Qin, C. Sun, Y. Jin, Y. Tan, J. Fieldsend, Large-scale evolutionary multiobjective optimization assisted by directed sampling, IEEE Transactions on Evolutionary Computation 25 (4) (2021) 724–738. doi:10.1109/TEVC.2021.3063606.

[27] W. Liu, L. Chen, X. Hao, W. Zhou, X. Cao, F. Xie, Offspring regeneration method based on bi-level sampling for large-scale evolutionary multi-objective optimization, Swarm and Evolutionary Computation 75 (2022) 101152. doi:https://doi.org/10.1016/j.swevo.2022.101152.

[28] X. Yang, J. Zou, S. Yang, J. Zheng, Y. Liu, A fuzzy decision variables framework for large-scale multiobjective optimization, IEEE Transactions on Evolutionary Computation 27 (3) (2023) 445–459. doi:10.1109/TEVC.2021.3118593.

[29] C. He, R. Cheng, D. Yazdani, Adaptive offspring generation for evolutionary large-scale multiobjective optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52 (2) (2022) 786–798. doi:10.1109/TSMC.2020.3003926.

[30] Y. Tian, X. Zheng, X. Zhang, Y. Jin, Efficient large-scale multiobjective optimization based on a competitive swarm optimizer, IEEE Transactions on Cybernetics 50 (8) (2020) 3696–3708. doi:10.1109/TCYB.2019.2906383.

[31] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Transactions on Cybernetics 45 (2) (2015) 191–204. doi:10.1109/TCYB.2014.2322602.

[32] Q. Lin, J. Li, S. Liu, L. Ma, J. Li, J. Chen, An adaptive two-stage evolutionary algorithm for large-scale continuous multi-objective optimization, Swarm and Evolutionary Computation 77 (2023) 101235. doi:10.1016/j.swevo.2023.101235.

[33] L. Li, Y. Li, Q. Lin, S. Liu, J. Zhou, Z. Ming, C. A. C. Coello, Neural net-enhanced competitive swarm optimizer for large-scale multiobjective optimization, IEEE Transactions on Cybernetics PP (2023) 1–14. doi:10.1109/TCYB.2023.3287596.

[34] S. Liu, Q. Lin, Q. Li, K. C. Tan, A comprehensive competitive swarm optimizer for large-scale multiobjective optimization, IEEE Transactions on Systems, Man, and Cybernetics: Systems 52 (9) (2022) 5829–5842. doi:10.1109/TSMC.2021.3131312.

[35] X. Wang, K. Zhang, J. Wang, Y. Jin, An enhanced competitive swarm optimizer with strongly convex sparse operator for large-scale multiobjective optimization, IEEE Transactions on Evolutionary Computation 26 (5) (2022) 859–871. doi:10.1109/TEVC.2021.3111209.

[36] S. Liu, J. Li, Q. Lin, Y. Tian, K. C. Tan, Learning to accelerate evolutionary search for large-scale multiobjective optimization, IEEE Transactions on Evolutionary Computation 27 (1) (2023) 67–81. doi:10.1109/TEVC.2022.3155593.

[37] K. Li, S. Kwong, J. Cao, M. Li, J. Zheng, R. Shen, Achieving balance between proximity and diversity in multi-objective evolutionary algorithm, Information Sciences 182 (1) (2012) 220–242. doi:https://doi.org/10.1016/j.ins.2011.08.027.

[38] Y. Sun, M. Kirley, S. K. Halgamuge, A recursive decomposition method for large scale continuous optimization, IEEE Transactions on Evolutionary Computation 22 (5) (2018) 647–661. doi:10.1109/TEVC.2017.2778089.

[39] R. Storn, K. Price, Differential Evolution – A simple and efficient heuristic for global optimization over continuous spaces, Journal of Global Optimization 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.

[40] S. Liu, M. Jiang, Q. Lin, K. C. Tan, Evolutionary large-scale multiobjective optimization via self-guided problem transformation, in: 2022 IEEE Congress on Evolutionary Computation (CEC), 2022, pp. 1–8. doi:10.1109/CEC55065.2022.9870259.

[41] N. Jin, D. Liu, Z. Pang, T. Huang, Wavelet basis function neural networks, in: 2007 International Joint Conference on Neural Networks, 2007, pp. 500–505. doi:10.1109/IJCNN.2007.4371007.

[42] B. G. Bodmann, P. K. Singh, Burst erasures and the mean-square error for cyclic parseval frames, IEEE Transactions on Information Theory 57 (7) (2011) 4622–4635. doi:10.1109/TIT.2011.2146150.

[43] J. Han, C. Moraga, The influence of the sigmoid function parameters on the speed of backpropagation learning, in: International workshop on artificial neural networks, Springer, 1995, pp. 195–201.

[44] M. Li, S. Yang, X. Liu, Shift-based density estimation for Pareto-based algorithms in many-objective optimization, IEEE Transactions on Evolutionary Computation 18 (3) (2014) 348–365. doi:10.1109/TEVC.2013.2262178.

[45] S. Qi, J. Zou, S. Yang, Y. Jin, J. Zheng, X. Yang, A self-exploratory competitive swarm optimization algorithm for large-scale multiobjective optimization, Information Sciences 609 (2022) 1601–1620. doi:https://doi.org/10.1016/j.ins.2022.07.110.

[46] Y. Tian, X. Xiang, X. Zhang, R. Cheng, Y. Jin, Sampling reference points on the Pareto fronts of benchmark multi-objective optimization problems, in: 2018 IEEE Congress on Evolutionary Computation (CEC), 2018, pp. 1–6. doi:10.1109/CEC.2018.8477730.

[47] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, A reference vector guided evolutionary algorithm for many-objective optimization, IEEE Transactions on Evolutionary Computation 20 (5) (2016) 773–791. doi:10.1109/TEVC.2016.2519378.

[48] S. Das, P. N. Suganthan, Differential evolution: A survey of the state-of-the-art, IEEE Transactions on Evolutionary Computation 15 (1) (2011) 4–31. doi:10.1109/TEVC.2010.2059031.

[49] S. Liu, Q. Lin, K. C. Tan, M. Gong, C. A. Coello Coello, A fuzzy decomposition-based multi/many-objective evolutionary algorithm, IEEE Transactions on Cybernetics 52 (5) (2022) 3495–3509. doi:10.1109/TCYB.2020.3008697.

[50] Q. Zhang, A. Zhou, S. Zhao, P. N. Suganthan, W. Liu, S. Tiwari, Multiobjective optimization test instances for the cec 2009 special session and competition, 2009.

[51] R. Cheng, Y. Jin, K. Narukawa, B. Sendhoff, A multiobjective evolutionary algorithm using gaussian process-based inverse modeling, IEEE Transactions on Evolutionary Computation 19 (6) (2015) 838–856. doi:10.1109/TEVC.2015.2395073.

[52] R. Cheng, Y. Jin, M. Olhofer, B. sendhoff, Test problems for large-scale multiobjective and many-objective optimization, IEEE Transactions on Cybernetics 47 (12) (2017) 4108–4121. doi:10.1109/TCYB.2016.2600577.

[53] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, E. Tsang, Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion, in: 2006 IEEE International Conference on Evolutionary Computation, 2006, pp. 892–899. doi:10.1109/CEC.2006.1688406.

[54] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, in: Evolutionary Multi-Criterion Optimization: 8th International Conference, EMO 2015, Guimarães, Portugal, March 29–April 1, 2015. Proceedings, Part II 8, Springer, 2015, pp. 110–125.

[55] Y. Tian, R. Cheng, X. Zhang, M. Li, Y. Jin, Diversity assessment of multi-objective evolutionary algorithms: Performance metric and benchmark problems [research frontier], IEEE Computational Intelligence Magazine 14 (3) (2019) 61–74. doi:10.1109/MCI.2019.2919398.

[56] L. While, L. Bradstreet, L. Barone, A fast way of calculating exact hypervolumes, IEEE Transactions on Evolutionary Computation 16 (1) (2012) 86–95. doi:10.1109/TEVC.2010.2077298.

[57] P. Czyżżak, A. Jaszkiewicz, Pareto simulated annealing—a metaheuristic technique for multiple-objective combinatorial optimization, Journal of Multi-criteria Decision Analysis 7 (1998) 34–47.

[58] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 577–601. doi:10.1109/TEVC.2013.2281535.

[59] J. Derrac, S. García, D. Molina, F. Herrera, A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms, Swarm Evol. Comput. 1 (2011) 3–18.

[60] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum], IEEE Computational Intelligence Magazine 12 (4) (2017) 73–87. doi:10.1109/MCI.2017.2742868.

[61] C. He, R. Cheng, C. Zhang, Y. Tian, Q. Chen, X. Yao, Evolutionary large-scale multiobjective optimization for ratio error estimation of voltage transformers, IEEE Transactions on Evolutionary Computation 24 (5) (2020) 868–881. doi:10.1109/TEVC.2020.2967501.

[62] F. Liu, X. Lin, Z. Wang, S. Yao, X. Tong, M. Yuan, Q. Zhang, Large language model for multi-objective evolutionary optimization (2023). arXiv:2310.12541.