Learning to Accelerate Evolutionary Search for Large-Scale Multiobjective Optimization

Songbai Liu, Jun Li, Qiuzhen Lin, *Member*, *IEEE*, Ye Tian, and Kay Chen Tan, *Fellow*, *IEEE*

***Abstract*—Most existing evolutionary search strategies are not so efficient when directly handling the decision space of large-scale multiobjective optimization problems (LMOPs). To enhance the efficiency of tackling LMOPs, this paper proposes an accelerated evolutionary search (AES) strategy. Its main idea is to learn a gradient-descent-like direction vector for each solution via the specially trained feedforward neural network, which may be the learnt possibly fastest convergent direction to reproduce new solutions efficiently. To be specific, a multilayer perceptron with only one hidden layer is constructed, in which the number of neurons in the input and output layers is equal to the dimension of the decision space. Then, to get appropriate training data for the model, the current population is divided into two subsets based on the non-dominated sorting, and each poor solution in one subset with worse convergence will be paired to an elitist solution in another subset with the minimum angle to it, which is considered most likely to guide it with rapid convergence. Next, this multilayer perceptron is updated via backpropagation with gradient descent by using the above elaborately prepared dataset. At last, an accelerated large-scale multiobjective evolutionary algorithm is designed by using AES as reproduction operator. Experimental studies validate the effectiveness of the proposed AES when handling the search space of LMOPs with dimensionality ranging from 1000 to 10000. When compared with six state-of-the-art evolutionary algorithms, the experimental results also show the better efficiency and performance of the proposed optimizer in solving various LMOPs.**

***Index Terms*—Large-scale Multiobjective Optimization, Accelerated Evolutionary Search, Multilayer Perceptron.**

I. Introduction

MAMY real-world optimization problems, such as training and architecture search of deep neural networks [1], capacitated vehicle routing [2], complex community detection [3], etc., often involve multiple mutually conflicting objective functions, which share a search space with many correlated variables. In this paper, when the dimensionality of the search space exceeds one thousand in the above cases, they are called large-scale multiobjective optimization problems (LMOPs). Since evolutionary algorithms (EAs) using the population can simultaneously find a set of tradeoff solutions that are equally optimal when considering all mutually conflicting objectives, the research of designing effective large-scale multiobjective EAs (LMOEAs) to handle LMOPs have attracted increasing attention in recent years [4]. Specifically, the search space of an LMOP expands exponentially with its increasing variable dimensionality, which is referred as the curse of dimensionality. To cope with such a large-scale decision space, the used optimizer should have a very efficient search capability [5]. However, the efficiency of existing search strategies in traditional EAs (e.g., polynomial mutation [6], simulated binary crossover (SBX) [7], particle swarm optimization (PSO) [8], and differential evolution (DE) [9]) deteriorates rapidly when using them to solve LMOPs. Consequently, the population converges slowly during the evolutionary process. Moreover, three recent surveys on evolutionary large-scale multiobjective optimization have pointed out that the key factor of developing an effective LMOEA is to enhance the search efficiency for reproducing high-quality solutions in the large-scale search space [10]-[12]. Several specially designed tricks, like divide and conquer manners (DCM) and dimension reduction techniques (DRT), have been used to customize the search strategies in some existing LMOEAs [13]-[25], which are respectively elaborated below.

In DCM-based LMOEAs [13]-[18], the target LMOP is first divided into multiple exclusive low-dimensional subproblems via a variable grouping method and then an existing solver is selected to separately optimize each subproblem using a subpopulation [13]. Specifically, to enhance the search efficiency, only the variables associated to the corresponding subproblem will change with the evolutionary search, while the remaining variables keep fixed. Thus, the key design principle in this kind of LMOEAs is to find a suitable method for the decomposition of the target LMOP. In other words, they endeavor to tailor an appropriate variable grouping method to divide the original search space properly. For examples, according to the contribution analysis of variables in MOEA/DVA [14], the variables in the target LMOP are divided into three different groups (i.e., diversity-related, convergence-related, and mixed groups of variables). Similarly, in LMEA [15], the variables are only classified into diversity-related and convergence-

Manuscript received on xx. xx. 2021, revised on xx. xx. 2021, and accepted on xx. xx. 2022. This work is partially supported by the National Natural Science Foundation of China (NSFC) under Grant No. U21A20512, Grant No. 61876162 and Grant No. 61876110, and in part by the Research Grants Council of the Hong Kong SAR under Grant No. PolyU11202418, Grant No. PolyU11211521 and Grant No. PolyU11209219, and in part by the Shenzhen Scientific Research and Development Funding Program under Grant JCYJ20190808164211203. (Corresponding authors: *Qiuzhen Lin* and *Kay Chen Tan*)

S.B. Liu is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, 518060, China, and with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR (e-mail: sbliu2-c@my.cityu.edu.hk).

J. Li and Q.Z. Lin are with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China.

Y. Tian is with the Key Laboratory of Intelligent Computing and Signal Processing of the Ministry of Education, Institutes of Physical Science and Information Technology, Anhui University, Hefei 230601, China.

K. C. Tan is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR (e-mail: kctan@polyu.edu.hk).

related groups based on a variable clustering method. Moreover, other methods based on dynamic random grouping [16], importance-based grouping [17], and graph-based differential grouping [18] are also designed. Since the performance of such LMOEAs depends heavily on the adopted grouping methods, these optimizers suffer from two major limitations: 1) the search will be misled once interacting variables are grouped irrelevantly, and 2) a large amount of computation resource is required for dividing variables into proper groups. Therefore, they are only suitable for solving those LMOPs with separable variables or few interacted variables [17].

Regarding DRT-based LMOEAs, they endeavor to get a dimension-reduced subspace of the target LMOP to shrink the search space, followed by using existing search strategies to effectively handle this learnt subspace [10]. There are two common methods in existing LMOEAs to find such a subspace: 1) find a weighted variable space via problem transformation [19]-[21]; and 2) learn a representative space based on machine learning models [22]-[26]. Concretely, for the problem transformation of the target LMOP, its original large-scale search space is transformed to a small-scale search space based on variable grouping in WOF [19] and GLEA [20], and based on the guidance of bi-directional reference vectors in LSMOF [21] and LMOEA-DS [22]. By this way, the original variables are updated according to the change of their associated weight variables (i.e., the small-scale weighted variable space is their actual search space), which can reproduce offspring effectively. However, as the original variables cannot be changed independently, some optimal original spaces may not be reachable, which lets these LMOEAs based on problem transformation easily get local convergence [20]. In addition, to learn a small-scale representative space of the target LMOP, random embedding is adopted in ReMO [23], principal component analysis is used in PCA-MOEA [24], unsupervised neural networks are modeled in MOEA/PSL [25], and pattern mining model is built in PM-MOEA [26]. Although these LMOEAs show excellent performance in solving sparse LMOPs [27], they still face challenges to solve regular and complex LMOPs, as searching the learnt representative space may also lose certain important information about the original search space. Besides, their performance is sensitive to the effectiveness of the learning model adopted [10].

Given above, both DCM-based and DRT-based LMOEAs improve their search efficiency for tackling LMOPs via specially designed tricks to get a simplified or reduced small-scale search space, followed by processing it with the traditional search strategies. Such operations are more likely to make the population easily fall into local optima, as the search is inadequate without fully exploring the original large-scale space [28]. In this context, some researchers try to implicitly enhance the diversity of new designed search strategies for effectively handling the original large-scale decision space directly. For examples, new genetic operators are designed in SparseEA [27] to address LMOPs with many sparse variables; competitive swarm optimizers (CSOs) with a stepped-up position update strategy and a comprehensive competition strategy are proposed in LMOCSO [29] and CCSO [30], respectively; a hypervolume-based dual local search strategy is reported in DLS-MOEA [31]; an adaptive search strategy with diverse directions is introduced in DGEA [32]; and new search strategies driven by generative adversarial network model are proposed in GMOEA [33] and GAN-LMEF [34]. Despite the fact that these LMOEAs can improve their search capabilities to some extent, they still suffer from a low computational efficiency due to the possible slowness of convergence speed [10]. Thus, they are often computationally expensive (in terms of the cost of function evaluations) to obtain a set of near-optimal solutions.

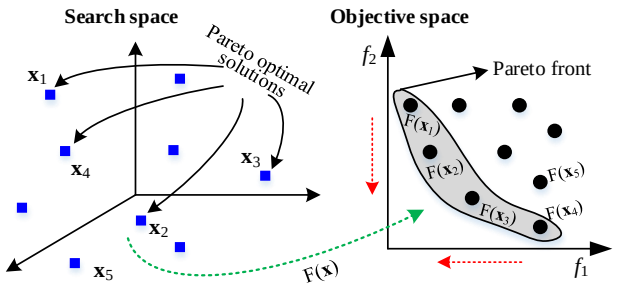


Fig. 1 Illustration of the PS and PF for a 2-objective LMOP.

To enhance the convergence speed, some studies in the domain of evolutionary multi and many-objective optimization have focused on designing the online *innovization* progress operators to help the generated offspring solutions further advance along the learned promising directions [35]-[38]. Specifically, in these *innovized* operators, a machine learning model is trained online to learn the directional improvements of solutions in the variable space, which consists of three modules: collect training solutions from the earlier generations; train the adopted model to learn the patterns on the prepared solutions; and use the well-trained model to repair part of the newly generated offspring. For example, in [37]-[38], the multi-layer full-connected artificial neural network (ANN) model is online trained on an archive of the previous populations from the past few generations, followed by using this model to repair half of the offspring generated by traditional crossover and mutation operators at a preset interval of generations. Recently, an enhanced *innovized* operator with a random forest model is proposed in [39] to improve the repair efficiency in solving multi and many-objective optimization problems. Obviously, learning such an ANN online will bring a huge extra computation cost for the original EAs, especially in solving LMOPs. Moreover, the acceleration of convergence caused by repairing only some of the newly generated offspring is still not enough to address the challenge of optimizing LMOPs. Inspired by this, to effectively speed up the convergence when searching directly in the large-scale decision space, this paper proposes an accelerated evolutionary search (AES) strategy driven by a feedforward artificial neural network, also known as multilayer perceptron (MLP) [40], mainly including the following three contributions.

1) We specially train a three-layer MLP model with only one hidden layer in this paper. In this MLP, the number of neurons in the input and output layers is equal to the

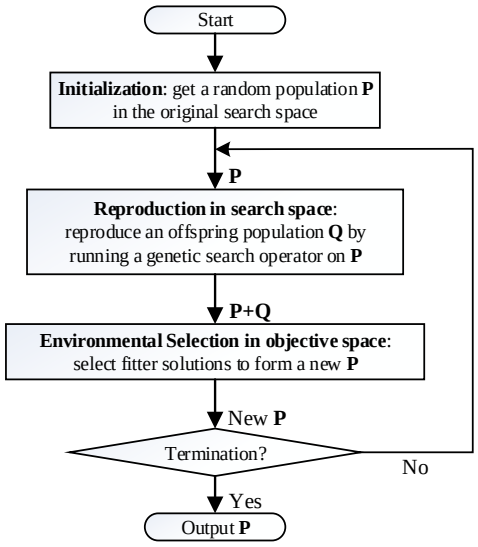


Fig. 2 Illustration of the general framework of an LMOEA.

dimension of the original search space. To get appropriate training data, we label a poor solution in the current population with an elite that is considered most likely to guide it with rapid convergence. In this way, this poor solution and its labelled elite are the input and target output for training the MLP, respectively.

2) We design a new evolutionary search strategy, i.e., AES, based on the above trained MLP model. In AES, each solution can learn the possible fastest convergent direction vector at its current position via the well-trained MLP to guide its evolutionary search, so as to accelerate the population's convergence speed.
3) We develop an accelerated LMOEA framework, termed ALMOEA, in which the offspring is reproduced based on the proposed AES, which can significantly improve the computational efficiency in solving various LMOPs.

The rest of this paper is organized as follows. At first, section II briefly introduces the preliminaries related to evolutionary large-scale multiobjective optimization (ELMO), the basis of multilayer perceptron, and our motivations of this work. Then, Section III provides the details of our proposed method. After that, Section IV presents some experimental studies to validate the effectiveness of the proposed method. Finally, Section V summarizes the main contributions of this paper and gives some important directions to enrich this work.

II. Preliminaries and Motivations

In this section, the preliminaries related to ELMO, including some basic concepts involved in LMOPs and the general framework of LMOEAs, are first introduced in Section II.A. Then, some backgrounds related to multilayer perceptron are briefly introduced in Section II.B. Finally, our motivations to develop a new AES strategy are elaborated in Section II.C.

A. Evolutionary Large-Scale Multiobjective Optimization

Without loss of generality, an unconstrained LMOP can be mathematically modeled as

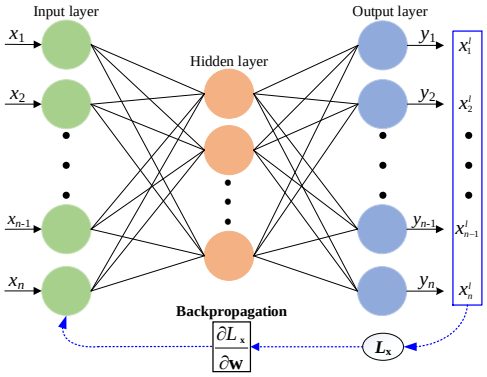


Fig. 3 Illustration of the architecture of the MLP constructed in this paper.

$$\text{Minimize } \mathbf{F}(\mathbf{x}) \in \mathbf{Y},\ \mathbf{x} \in \Omega, \tag{1}$$

where $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}),..., f_m(\mathbf{x}))$ and $\mathbf{x} = (x_1,\ldots,x_n)$, respectively, denote m (often conflicting) objective functions in the objective space $\mathbf{Y}$ and n (often interacting) decision variables in the search space Ω ($m \geq 2$ and $n \geq 1000$). Since the m objectives in (1) are often mutually conflicting to some extent, it is impossible to find a single solution that minimizes all objectives simultaneously. Thus, the goal of optimizing an LMOP is to get a set of Pareto optimal solutions (PS) in its search space, and the mapping of PS in the objective space is called Pareto front (PF), as shown in Fig. 1, which involves the following two basic definitions.

Definition 1 (Pareto dominance and non-dominance): For two solutions $\mathbf{x}, \mathbf{y} \in \Omega$, we say $\mathbf{x}$ dominates $\mathbf{y}$ iff $f_i(\mathbf{x}) \leq f_i(\mathbf{y})$ for $\forall i \in \{1, 2, \ldots, m\}$ and $f_j(\mathbf{x}) < f_j(\mathbf{y})$ for $\exists\, j \in \{1, 2, \ldots, m\}$. $\mathbf{x}$ and $\mathbf{y}$ are said non-dominated with each other, when either one of $\mathbf{x}$ and $\mathbf{y}$ cannot dominate another one. For example, in Fig. 1, $\mathbf{x}_3$ dominates $\mathbf{x}_5$, while $\mathbf{x}_2$ and $\mathbf{x}_3$ are mutually non-dominated.

Definition 2 (Pareto optimality): We say $\mathbf{x} \in \Omega$ is a Pareto optimal solution iff there does not exist $\forall \mathbf{y} \in \Omega$ that dominates it. Clearly, solutions $\mathbf{x}_1$, $\mathbf{x}_2$, $\mathbf{x}_3$, and $\mathbf{x}_4$ are the Pareto optimal solutions in Fig. 1, and their objective vectors form the PF.

Therefore, when designing an LMOEA to solve an LMOP, we aim to find a final population that can closely approximate and evenly cover the entire PF as much as possible. As introduced in Section I, some existing LMOEAs try to speed up the evolutionary search in a simplified small-scale space by using traditional evolutionary strategies, while some of them attempt to design new search strategies to efficiently explore in the original large-scale space. Then, they preserve some superior solutions with better convergence and diversity by customizing their environmental selection strategies in the objective space, including Pareto-based [41]-[42], indicator-based [43]-[44], decomposition-based [45]-[46], and clustering-based [47]-[48] selection strategies. Thus, all the existing LMOEAs share the general framework similar to multiobjective EAs (MOEAs), which includes three main procedures, i.e., initialization, reproduction, and environmental selection, as shown in Fig. 2.

B. Multilayer Perceptron

Neural network is a mathematical model to simulate human

brain for acquiring knowledge through a learning process, which has become a very important learning tool in the field of artificial intelligence. Structurally, a neural network is composed by many neurons in the form of layers. One of the simplest neural networks is feedforward neural network, also known as multilayer perceptron (MLP), in which each layer of neurons receives inputs from the previous layers. Mathematically, each neuron receives its inputs by a summing junction to compute a weighted linear combination, followed by further modifying this combination result with a nonlinear activation function to get the output. As shown in Fig. 3, an MLP has an input layer and an output layer to connect with the outside world. Besides, an MLP usually has one or more layers of hidden neurons to extract important features contained in the input data [49]. In the constructed MLP of this paper, the input and output layers have the same number of neurons, which equals to the number of variables (n) in the LMOP to be solved. Besides, when introducing a machine learning model into an EA, it is necessary to consider whether the cost of learning this model is affordable and worth the benefits, not only referring to function evaluations, but also considering the overhead imposed by the model. Keeping this in mind, only one hidden layer with K neurons is considered in this MLP for the consideration of computational overhead ($K << n$ and K = 10 is suggested in our experiment studies considering from the tradeoff between the time complexity and the final performance). The sigmoid function $\delta(x) = 1/(1+e^{-x})$ is used as the activation function for the MLP, which implies we need to normalize the training solutions before learning the MLP.

Basically, the learning or training of an MLP is a process to iteratively update its parameters (i.e., the weights and biases) based on the backpropagation with gradient descent [50]. Specifically, the steepest descent direction can be calculated by the loss versus the present parameters, followed by iteratively modifying the parameters along this gradient descent direction to reduce the loss. Given the training data set $D = \{(\mathbf{x}_i, \mathbf{x}_i^l)\}_{i=1}^M$ with M input-output examples, the learning goal is to update the parameters of the MLP so that the actual output $\mathbf{y}_i$ corresponding to input $\mathbf{x}_i$ is close enough to its labeled target output $\mathbf{x}_i^l$ for all $i = \{1, 2, \ldots, M\}$ in a statistical sense. In this work, we use the mean-square error (MSE) as the loss L_x to be minimized for evaluation. Moreover, the training of an MLP via backpropagation can be implemented in two basic modes: (1) the sequential mode that updates the parameters on an example-by-example basis; and (2) the batch mode that adjusts the parameters on an epoch-by-epoch basis, where each epoch consists of the entire set of training examples. In this paper, the sequential mode is used to train the MLP.

C. Motivations

As discussed in Section I, the existing LMOEAs based on DCM or DRT have some limitations when solving LMOPs, in which the original large-scale search space cannot be fully explored. Thus, directly handling the original large-scale search space of the target LMOP via designing a new search strategy with high efficiency may be a more attractive and promising scheme to improve the performance of LMOEAs.

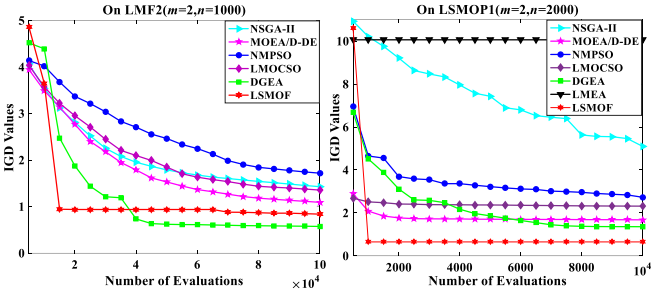


Fig. 4 Convergence profiles of the mean IGD values obtained by seven different EA optimizers on LSMOP1 and LMF2.

However, the search efficiency of most existing search strategies, such as SBX, DE, PSO, CSO, etc., is not satisfactory, which often require an expensive computational budgets to well solve an LMOP [18].

For example, we compared the performance of LSMOF with NSGA-II [41], NMPSO [51], LMOCSO [29], DGEA [32], and MOEA/D-DE [52] on one 2-objective test LMOP (LMF2 [20] with n = 1000). Except for LSMOF, the other five algorithms adopt different evolutionary search strategies to directly handle the large-scale decision space of LMF2. Fig. 4 illustrates their convergence profiles based on the mean inverted generation distance (IGD) [53] values obtained during the evolutionary process with setting the maximum function evaluations $FE_{\max}=10^5$. As can be seen from Fig. 4, although LSMOF can quickly obtain a relatively good IGD value on LMF2 with very little computation resource (FE < 20000), its performance has not been improved since then. In contrast, DGEA is less computationally efficient than LSMOP in solving LMF2 in the early stage of evolution, but it can achieve better IGD values than LSMOF when FE > 40000. In addition, the other four competitors are not only less efficient than LSMOF in solving LMF2, but also perform worse than LSMOF in the final IGD values when $FE_{\max} = 10^5$. Moreover, we also compared the performance of these algorithms (including LMEA) in solving another 2-objective test LMOP (LSMOP1 [54] with n = 2000) when having relatively little computation resource (i.e., $FE_{\max} = 10^4$). According to their convergence profiles with the mean IGD values plotted in Fig 3, LMEA performs obviously worst in this case because all of its computation resource is spent on variable analysis rather than the optimization process. Besides, although LSMOF gets the smallest IGD value, it still quickly stops converging and wastes most of the computation resource. For the remaining five optimizers that search directly in the large-scale decision space of LSMOP1, their performance is poor in both the final IGD values and the performance improvement rates during the evolutionary process, which validates that their computational efficiency is not so satisfactory.

To address the above issues, this paper proposes an AES strategy, aiming to directly search in the large-scale decision space of the target LMOP with a high efficiency. To achieve this, we use an MLP to learn the possible fastest convergent direction vector for each solution, the search of which will be guided based on this learnt direction vector (referred to gradient-descent-like direction vector (GDV) in this paper). In this way, the population's convergence speed can be

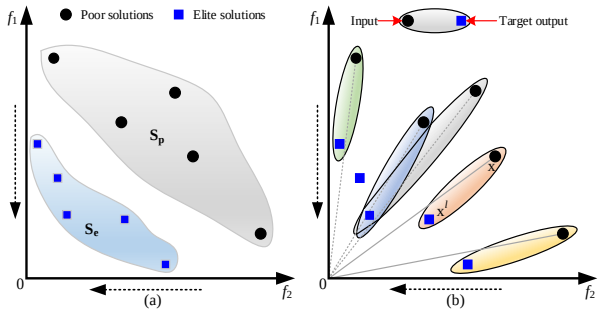


Fig. 5 Illustration of getting the training data via (a) Division of the current population, and (b) labeling the target output of an input solution.

accelerated and the computational efficiency of the correspondingly proposed LMOEA framework, i.e., ALMOEA, can be further improved.

III. The Proposed Methods

In this section, the newly proposed AES strategy is introduced in detail, which can accelerate the convergence speed of the evolutionary population in the correspondingly developed optimizer ALMOEA. As an MLP is constructed in ALMOEA to learn the GDV of each solution for customizing AES, the learning of this MLP is first described in Section III.A. Then, the details of using AES to reproduce offspring are provided in Section III.B, followed by elaborating the general framework of ALMOEA in Section III.C.

A. Learning the GDV of a Solution via MLP

As introduced in Section II.B, to train the MLP, we need to first get a set of training data $D=\{(\mathbf{x}_i,\mathbf{x}_i^I)\}_{i=1}^M$, where each input solution $\mathbf{x}$ has a label $\mathbf{x}^I$ as its target output and M is the number of samples. Under the supervision of $\mathbf{x}^I$, knowledge is acquired by the MLP through a supervised learning process. Here, we expect that the MLP can be used to learn the GDV of an input solution $\mathbf{x}$, which can guide $\mathbf{x}$ with rapid convergence. Thus, each input $\mathbf{x}$ is paired with another solution vector that is most likely to guide $\mathbf{x}$ with rapid convergence for training the MLP. To achieve this, we divide the current population $\mathbf{P}$ with N solutions into two subsets $\mathbf{S_p}$ and $\mathbf{S_e}$, where the solutions in $\mathbf{S_e}$ perform better than those in $\mathbf{S_p}$ in terms of convergence for the target LMOP. Specifically, similar to the environmental selection proposed in NSGA-II, the solutions in $\mathbf{P}$ are first assigned into multiple fronts based on their domination relationships, followed by sorting the solutions in each front according to their crowding distances. Then, the solutions in $\mathbf{P}$ are replaced by the solutions from the first to the last front in sequence. After that, we can add the first $N/2$ solutions of $\mathbf{P}$ to $\mathbf{S_e}$ and the rest to $\mathbf{S_p}$, as a simple example shown in Fig. 5(a). In this way, each solution $\mathbf{x}\in\mathbf{S_p}$ can be the input to train the MLP, and a solution $\mathbf{x}^I\in\mathbf{S_e}$ labelled as the target output of $\mathbf{x}$ can be obtained as

$$\mathbf{x}^I=\mathbf{y}:\underset{\mathbf{y}\in\mathbf{S_e}}{\operatorname{argmin}}\ \theta(\mathbf{x},\mathbf{y}), \tag{2}$$

where $\theta(\mathbf{x},\mathbf{y})$ indicates the acute angle value between $\mathbf{x}$ and $\mathbf{y}$, which is computed as follows:

Algorithm 1 Training

Input: the current MLP model and the current population $\mathbf{P}$
Output: the trained MLP
1: normalize the objective values of all solutions in $\mathbf{P}$ by (4)
2: normalize the variable values of all solutions in $\mathbf{P}$ by (7)
3: divide $\mathbf{P}$ into two subsets $\mathbf{S_p}$ and $\mathbf{S_e}$
4: **for** each $\mathbf{x}\in\mathbf{S_p}$ **do**
5: get its label solution $\mathbf{x}^I\in\mathbf{S_e}$ by (2) and readjust by (6)
6: compute the loss $\boldsymbol{L}_\mathbf{x}$ of MLP according to $\mathbf{x}$ and $\mathbf{x}^I$
7: update the parameters of MLP based on $\partial L_\mathbf{x}/\partial\mathbf{w}$
8: **end for**
9: **return** the trained MLP

$$\theta(\mathbf{x},\mathbf{y})=\arccos\left|\frac{\mathbf{F'(x)}\cdot\mathbf{F'(y)}}{\|\mathbf{F'(x)}\|\times\|\mathbf{F'(y)}\|}\right|, \tag{3}$$

where $\mathbf{F'(x)}=(f_1'(\mathbf{x}),...,f_m'(\mathbf{x}))$ is the normalized vector of the m objective values for $\mathbf{x}$, and $f_i'(\mathbf{x})$ can be computed as,

$$f_i'(\mathbf{x})=\frac{f_i(x)-f_i^{\min}}{f_i^{\max}-f_i^{\min}},i=1,2,...,m, \tag{4}$$

Here, $f_i^{\min}$ and $f_i^{\max}$ are, respectively, the minimum and maximum values of the ith objective for all solutions in $\mathbf{P}$. Fig. 5(b) is a simple example to show this labeling process. In this way, the trained MLP can then be used to learn the GDV at the current position of each solution (also including the solutions in $\mathbf{S_e}$) that is most likely to guide it with rapid convergence. Note that more than half of solutions in $\mathbf{P}$ may be mutually non-dominated in the later evolution, so that only $N/2$ non-dominated solutions with smaller crowding distances are preserved into $\mathbf{S_e}$, which may lead to the situation that some non-dominated solutions in $\mathbf{S_p}$ may not be less convergent than their labelled solutions in $\mathbf{S_e}$ according to (2). Under this case, the angle values between each pair of ($\mathbf{x}$, $\mathbf{x}^I$) are likely to be so small, thus we believe a scalar fitness indicator can be used to compare them more precisely. Here, we compute the distance $EdI(\mathbf{x})$ from each solution $\mathbf{x}$ to the ideal point, as follows:

$$EdI(\mathbf{x})=\sqrt{\|\mathbf{F'(x)}\|\times\|\mathbf{F'(x)}\|} \tag{5}$$

Accordingly, we readjust the input and its target output based on the following way:

$$\begin{cases}\text{input}=\mathbf{x},\text{target}=\mathbf{x}^I & \text{if } EdI(\mathbf{x})\le EdI(\mathbf{x}^I)\\ \text{input}=\mathbf{x}^I,\text{target}=\mathbf{x} & \text{otherwise}\end{cases} \tag{6}$$

Algorithms 1 presents the general training process for the MLP, which requires two inputs: the current MLP model and the current population $\mathbf{P}$. Particularly, the procedures in lines 1-3 are the preprocessing of training data, including the normalization of objective values for all solutions in line 1, the division of $\mathbf{P}$ in line 3, and the normalization of variable values for all solutions in line 2. Here, the ith variable x_i of solution $\mathbf{x}$ is normalized as follows:

$$x_i'=\frac{x_i-lb_i}{ub_i-lb_i},i=1,2,...,n, \tag{7}$$

where $\mathbf{lb}$ = (lb_1, …, lb_n) and $\mathbf{ub}$ = (ub_1, …, ub_n) are the corresponding lower bound and upper bound of the n-dimensional search space for the target LMOP. When training the MLP, only the normalized variable values are considered for each input solution $\mathbf{x}$ and its target output $\mathbf{x}^I$ obtained in line 5. For the training loop in lines 4-8, we first compute the loss $\boldsymbol{L}_\mathbf{x}$ of

Algorithm 2 Reproduction
Input: the trained MLP model and the parent population **P**
Output: the offspring population **Q**
1: initialize the offspring population $\mathbf{Q} = \varnothing$
2: **for** each $\mathbf{x} \in \mathbf{P}$ **do**
3: compute the $\mathbf{x}^{\mathbf{gdv}}$ by inputting $\mathbf{x}$ to the MLP
4: select two different random solutions ($\mathbf{x}^{d_1}$, $\mathbf{x}^{d_2}$) $\in \mathbf{P}$
5: get offspring $\mathbf{x}^{\mathbf{new}}$ by AES on $\mathbf{x}$, $\mathbf{x}^{\mathbf{gdv}}$, $\mathbf{x}^{d_1}$, $\mathbf{x}^{d_2}$ with (8)
6: update $\mathbf{x}^{\mathbf{new}}$ by the polynomial mutation
7: add $\mathbf{x}^{\mathbf{new}}$ into the offspring population **Q**
8: **end for**
9: **return Q**

Algorithm 3 General Framework of ALMOEA
Input: the target LMOP with *m* objectives and *n* variables, the maximum number of function evaluations ($FE_{\max}$)
Output: the final population **P**
1: initialize a population **P** with *N* random solutions of LMOP
2: initialize an MLP with random parameters, and $FE = 0$
3: **while** $FE \leq FE_{\max}$ **do**
4: MLP $\leftarrow$ Training (MLP, **P**) //**Algorithm 1**
5: **Q** $\leftarrow$ Reproduction (MLP, **P**) //**Algorithm 2**
6: **P** $\leftarrow$ Existing environmental selection (**P**, **Q**)
7: $FE \leftarrow FE + N$
8: **end while**
9: **return P**

the MLP according to $\mathbf{x}$ and $\mathbf{x}^I$ in line 6, and then we update the parameters (denoted by $\mathbf{w}$) based on the backpropagation via gradient descent (denoted by $\partial L_{\mathbf{x}} / \partial \mathbf{w}$) in line 7. Finally, the trained MLP is outputted in line 9, which will be used in the proposed AES strategy to learn the GDV of a solution. Clearly, different from the *innovization* progress operators introduced in Section I, we train the MLP based on the current population with only one epoch at each generation, because it's a process of cumulative learning as the population continues to evolve. In this way, we can learn the MLP without introducing computational overhead and ease overfitting to some extent.

B. Accelerated Evolutionary search for Reproduction

In an LMOEA, searching in the decision space of the target LMOP is the driving force for the evolutionary population to converge toward the optima, because it is possible to reproduce high-quality solutions by efficient search. However, as discussed in Section II.C, the existing evolutionary search strategies show poor efficiency in a large-scale decision space, which often leads to a slow convergence speed of the population. The reason behind this is that the offspring generated by the existing search strategies is generally based on the recombination of parent solutions, which is not always the evolutionary direction of the fastest convergence and shows a low search efficiency in the large-scale decision space. Thus, we propose a new AES strategy to enhance the efficiency of searching the large-scale decision space and speed up the convergence of the evolutionary population. Specifically, starting from the current position of a solution $\mathbf{x}$, a new solution $\mathbf{x}^{\mathbf{new}}$ can be obtained by AES with the following way:

$$\mathbf{x}^{\mathbf{new}} = \mathbf{x} + r_1(\mathbf{x} - \mathbf{x}^{\mathbf{gdv}}) + r_2(\mathbf{x}^{d_1} - \mathbf{x}^{d_2}), \quad (8)$$

where $\mathbf{x}^{\mathbf{gdv}}$ is the learned GDV of $\mathbf{x}$, which can be computed by inputting $\mathbf{x}$ into the trained MLP obtained in **Algorithm 1**. Besides, $\mathbf{x}^{d_1}$ and $\mathbf{x}^{d_2}$ are two randomly selected solutions from the current population **P** and $\mathbf{x}^{d_1} \neq \mathbf{x}^{d_2} \neq \mathbf{x}$; r_1 and r_2 are both random numbers ranging from 0 to 1. Clearly, the AES starting from $\mathbf{x}$ defined in (8) consists of two search components: one is a self-acceleration component driven by the negative GDV of $\mathbf{x}$ (i.e., $\mathbf{x} - \mathbf{x}^{\mathbf{gdv}}$), and the other one is a differential-regulation component guided by other solutions (i.e., $\mathbf{x}^{d_1} - \mathbf{x}^{d_2}$). In this way, the convergence to the optima can be accelerated by searching along the learned negative GDV in AES. Meanwhile, the differential-regulation with other solutions can avoid the AES falling into local optima to some extent. Consequently, the reproduction efficiency can be improved by using AES to generate offspring, which is completely different from the ANN-assisted *innovized* reproduction [37]-[38] that repairs some of the offspring solutions generated by the traditional evolutionary operators.

Algorithm 2 presents the general process of reproducing an offspring population **Q** with the AES in (8), which requires two inputs: the trained MLP and the parent population **P**. In particular, **Q** is first initialized as an empty-set in line 1, followed by preserving the *N* newly generated solutions after executing the procedures in lines 2-8. Concretely, for each solution $\mathbf{x} \in \mathbf{P}$ in line 2, we can first compute its GDV ($\mathbf{x}^{\mathbf{gdv}}$) by inputting $\mathbf{x}$ into the well-trained MLP in line 3, and randomly select other two different solutions $\mathbf{x}^{d_1}$ and $\mathbf{x}^{d_2}$ from **P** in line 4. Then, we can run the AES of (8) based on $\mathbf{x}$, $\mathbf{x}^{\mathbf{gdv}}$, $\mathbf{x}^{d_1}$ and $\mathbf{x}^{d_2}$ to reproduce a new offspring solution $\mathbf{x}^{\mathbf{new}}$ in line 5, followed by adding it into **Q** in line 7. Please note that we also use the polynomial mutation [6] to update the newly generated $\mathbf{x}^{\mathbf{new}}$ in line 6 for further avoiding trapped in local optima. At last, the offspring population **Q** with *N* new solutions is outputted in line 9. Compared with the reproduction in DCM-based and DRT-based LMOEAs, our AES-based reproduction not only shows the advantage of accelerating the convergence speed of the evolutionary population, but also has the ability to effectively explore the whole decision space of the target LMOP.

C. The Framework of ALMOEA

This section presents the general framework of the accelerated LMOEA (ALMOEA) proposed in this paper. The pseudocode and flowchart of ALMOEA are shown in **Algorithm 3** and Fig. 6, respectively. Similar to the general framework of LMOEAs as shown in Fig. 2, ALMOEA also has three general procedures, i.e., initialization (lines 1-2), reproduction (line 5), and environmental selection (line 6). In particular, since the reproduction via AES in ALMOEA requires an MLP model to learn the GDV of each solution for guiding it with rapid convergence, a random MLP model is first constructed in line 2, followed by online training this model in line 4 at each generation. Besides, all existing environmental selection strategies can be embedded into ALMOEA in line 6, to select *N* fitter solutions from the combined population of parent and offspring, i.e., the $(\mu+\mu)$ selection way. Thus, in this paper, three most used environmental selection strategies (i.e., Pareto-based, indicator-based, and decomposition-based strategies) are embedded into ALMOEA to study its performance. Concretely, the environmental selection strategies respectively proposed in NSGA-II [41], SMS-EMOA [55], and MOEA/D

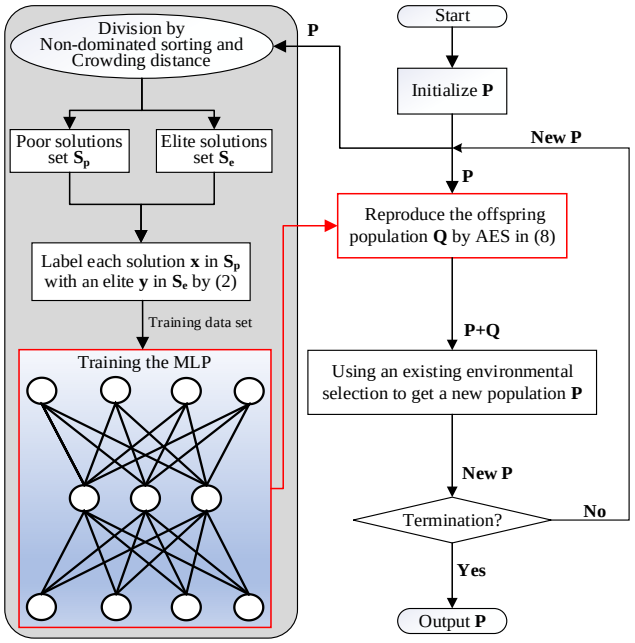


Fig. 6 Illustration of the framework for the proposed ALMOEA.

[46] are considered in ALMOEA, correspondingly yielding its three different versions, termed ANSGA-II, ASMS-EMOA, and AMOEA/D, respectively. Please note that the population tailoring is conducted in SMS-EMOA and MOEA/D once a new offspring is generated, i.e., the iterative $(\mu+1)$ selection way. To meet our requirements, we remodify the selection strategies from these two optimizers to the $(\mu+\mu)$ selection way. Furthermore, the ALMOEA terminates once the function evaluation counter (FE) reaches the preset maximum number of function evaluations ($FE_{\max}$) when iteratively running the procedures in lines 3-8. At last, the final population **P** obtained by ALMOEA is outputted in line 9 to be the approximate optimal solutions set of the target LMOP.

Obviously, at each generation, the computation complexity of ALMOEA is dominated by its three main procedures, that is, the training of MLP in line 4, the reproduction in line 5, and the environmental selection in line 6. Specifically, training the MLP in **Algorithm 1** requires a time complexity of $O(mN^2 + NnK)$, where m, N, n, and K represent the number of objectives, the population size, the number of variables, and the number of neurons in the hidden layer of MLP, respectively. Besides, the worst time complexity of reproducing the offspring population **Q** by **Algorithm 2** is $O(NnK)$. At last, the worst time complexity of running environmental selection depends on the specific existing strategy embedded in line 6. For example, the environmental selection proposed in NSGA-II requires a time complexity of $O(mN^2)$. Thus, the overall worst time complexity of ANSGA-II in one generation is $O(mN^2 + NnK)$, which is comparable to that of NSGA-II if the value of K is significantly smaller than n. Therefore, it is reasonable to set K = 10 in this paper when considering the computation complexity of ALMOEA. The detailed runtime of ALMOEA is analyzed in our experimental studies by comparing with its competitors.

IV. Experimental Studies

To empirically validate the effectiveness of the proposed AES in handing the large-scale search space, we first compare and analyze the performance of ANSGA-II, ASMS-EMOA, and AMOEA/D with their corresponding original algorithms (i.e., NSGA-II, SMS-EMOA, and MOEA/D) in solving a set of different LMOP benchmarks. Then, to further prove the higher computational efficiency of the ALMOEA framework, we make a performance comparison between AMOEA/D and six competitive existing LMOEAs, which cover all the three different types of LMOEAs introduced in Section I. Finally, an ablation study of ALMOEA is conducted to demonstrate the effectiveness of the main components in AES.

A. Benchmark Problems and Performance Metrics

In this paper, two benchmark suites, namely LSMOP [54] and LMF [20], are selected to assess the performance of all considered LMOEAs. For each test problem in LSMOP and LMF, the number of decision variables is set as n = {1000, 3000, 5000, 10000}. In addition, the number of objectives for each test problem in LSMOP is set to m = {2, 3}, while only m = 2 is considered for all LMF test problems. Please note that all the other parameters involved in these two test suites are set as suggested in their original references, which also provide the detailed mathematical definitions of each test problems.

To quantitatively assess the performance of an LMOEA in solving the target LMOP, the well-known inverted generational distance (IGD) [53] and hypervolume (HV) [56], which can concurrently reflect the convergence and diversity of the final population, are selected as the indicators in our experimental studies. Besides, a number of evenly distributed sample points are required in the calculation of the IGD, which should be extracted from the true PF of the target LMOP. Specifically, suppose that **S** is the set of sample points from the true PF and **P** is the final population with only non-dominated solutions obtained by the corresponding LMOEA. Then, the IGD between **P** and **S** can be calculated as follows:

$$\mathrm{IGD}(\mathbf{P},\mathbf{S}) = \frac{1}{|\mathbf{S}|}\sum_{\mathbf{y}\in\mathbf{S}}\left[\min_{\mathbf{x}\in\mathbf{P}}\|\mathbf{F}(\mathbf{x})-\mathbf{y}\|\right], \tag{9}$$

where $|\mathbf{S}|$ denotes the number of points in **S**, and we set $|\mathbf{S}|$ = 10000 in our experiments. Besides, we compute the HV of **S** via the method proposed in the PlatEMO [57], which only requires to specify the nadir point of the true PF. Here, the smaller (or larger) value of IGD (or HV) for **S** indicates the solutions in **P** are more approximate to **S** and more evenly distributed along **S**, which also shows the better performance of the corresponding LMOEA in solving the LMOP.

B. The Compared Algorithms and Parameter Settings

In the experimental studies, we consider three MOEAs (i.e., NSGA-II, SMS-EMOA, and MOEA/D) and seven competitive LMOEAs (i.e., WOF, MOEA/DVA, GLEA, LMOCSO, LSMOF, MOEA/PSL, and DGEA) for performance comparison with our proposed AMLOEA. To make a fair comparison, the corresponding parameters in these considered competitors are all set as suggested in their original references, which also can be found in the PlatEMO and Table S1 of the supple-

TABLE I
THE AVERAGE IGD RESULTS OBTAINED BY THREE VERSIONS OF ALMOEA AND THEIR ORIGINAL VERSIONS ON 3-OBJECTIVE LSMOP1-9

Problem	n	NSGA-II	ANSGA-II	SMS-EMOA	ASMS-EMOA	MOEA/D-DE	AMOEA/D
LSMOP1	1000	9.712E+0(4.41E-1)-	**5.221E-1(4.14E-2)**	5.251E+0(1.77E-1)-	**4.918E-1(2.75E-2)**	1.725E+0(9.46E-2)-	**4.800E-1(2.83E-2)**
	3000	1.088E+1(4.41E-1)-	**5.329E-1(4.17E-2)**	7.972E+0(2.07E-1)-	**5.025E-1(3.49E-2)**	1.786E+0(7.52E-2)-	**4.877E-1(2.77E-2)**
	5000	1.113E+1(3.13E-1)-	**5.527E-1(5.25E-2)**	8.862E+0(1.22E-1)-	**4.921E-1(3.89E-2)**	1.816E+0(1.21E-1)-	**4.942E-1(2.15E-2)**
	10000	1.172E+1(4.77E-1)-	**5.363E-1(2.11E-2)**	9.748E+0(1.04E-1)-	**4.970E-1(3.81E-2)**	1.812E+0(1.10E-1)-	**4.966E-1(2.43E-2)**
LSMOP2	1000	5.721E-2(1.96E-3)-	**4.817E-2(1.60E-3)**	4.262E-2(1.69E-4)-	**3.841E-2(6.42E-4)**	5.204E-2(3.51E-4)-	**3.662E-2(5.72E-4)**
	3000	4.962E-2(2.54E-3)-	**4.376E-2(1.11E-3)**	**3.451E-2(8.41E-5)+**	3.645E-2(5.24E-4)	4.626E-2(1.48E-4)-	**3.208E-2(1.75E-4)**
	5000	4.817E-2(2.33E-3)-	**4.349E-2(1.44E-3)**	**3.338E-2(1.07E-4)+**	3.628E-2(5.57E-4)	4.553E-2(1.37E-4)-	**3.132E-2(1.25E-4)**
	10000	4.624E-2(1.55E-3)-	**4.254E-2(1.33E-3)**	**3.262E-2(1.08E-4)+**	3.595E-2(4.30E-4)	4.516E-2(1.15E-4)-	**3.086E-2(1.81E-4)**
LSMOP3	1000	3.669E+1(1.87E+1)-	**9.574E-1(2.44E-1)**	1.411E+1(1.07E+0)-	**3.981E+0(4.72E+0)**	9.783E+0(3.84E-1)-	**8.596E-1(4.87E-3)**
	3000	5.866E+1(2.79E+1)-	**9.974E-1(2.68E-1)**	1.734E+1(7.97E-1)-	**2.059E+0(3.09E+0)**	1.056E+1(2.55E-1)-	**8.644E-1(4.97E-2)**
	5000	6.562E+1(3.60E+1)-	**1.050E+0(3.37E-1)**	1.813E+1(5.50E-1)-	**1.161E+0(5.83E-1)**	1.064E+1(2.43E-1)-	**8.994E-1(1.14E-1)**
	10000	6.890E+1(3.65E+1)-	**9.878E-1(3.72E-1)**	1.885E+1(4.66E-1)-	**3.748E+0(4.38E+0)**	1.075E+1(2.99E-1)-	**8.886E-1(8.37E-2)**
LSMOP4	1000	1.260E-1(2.99E-3)-	**8.242E-2(2.71E-3)**	1.018E-1(4.62E-4)-	**6.464E-2(1.31E-3)**	1.042E-1(1.48E-3)-	**6.852E-2(3.33E-3)**
	3000	6.679E-2(2.78E-3)-	**5.289E-2(2.22E-3)**	5.226E-2(1.52E-4)-	**4.254E-2(5.94E-4)**	5.912E-2(3.45E-4)-	**4.287E-2(9.86E-4)**
	5000	5.626E-2(2.41E-3)-	**4.735E-2(1.42E-3)**	4.208E-2(1.24E-4)-	**3.915E-2(7.26E-4)**	5.116E-2(2.05E-4)-	**3.724E-2(4.66E-4)**
	10000	4.981E-2(1.99E-3)-	**4.396E-2(1.11E-3)**	**3.547E-2(1.27E-4)+**	3.690E-2(7.65E-4)	4.664E-2(1.31E-4)-	**3.305E-2(1.18E-4)**
LSMOP5	1000	1.810E+1(6.24E-1)-	**5.507E-1(4.19E-2)**	1.373E+1(5.71E-1)-	**5.412E-1(4.65E-4)**	4.067E+0(3.10E-1)-	**5.424E-1(5.05E-4)**
	3000	1.970E+1(4.90E-1)-	**5.409E-1(1.12E-4)**	1.719E+1(4.85E-1)-	**5.472E-1(1.94E-2)**	3.985E+0(2.57E-1)-	**5.434E-1(3.89E-3)**
	5000	2.030E+1(5.65E-1)-	**5.422E-1(4.08E-3)**	1.821E+1(3.26E-1)-	**5.425E-1(3.87E-3)**	4.048E+0(2.41E-1)-	**5.426E-1(8.18E-4)**
	10000	2.060E+1(3.76E-1)-	**5.437E-1(1.13E-2)**	1.921E+1(2.33E-1)-	**5.430E-1(6.36E-3)**	4.225E+0(3.83E-1)-	**5.442E-1(4.54E-3)**
LSMOP6	1000	1.908E+4(4.34E+3)-	**1.338E+0(1.14E-1)**	4.912E+3(7.15E+2)-	**1.361E+0(1.31E-1)**	6.076E+2(3.24E+2)-	**1.332E+0(1.13E-1)**
	3000	4.295E+4(1.24E+4)-	**1.463E+0(1.69E-1)**	1.053E+4(1.10E+3)-	**1.421E+0(1.65E-1)**	9.206E+2(4.29E+2)-	**1.346E+0(8.03E-2)**
	5000	5.080E+4(1.09E+4)-	**1.388E+0(1.35E-1)**	1.297E+4(6.93E+2)-	**1.411E+0(1.45E-1)**	9.291E+2(4.89E+2)-	**1.376E+0(1.20E-1)**
	10000	5.631E+4(1.10E+4)-	**1.476E+0(1.80E-1)**	1.600E+4(3.96E+2)-	**1.474E+0(1.53E-1)**	1.038E+3(5.73E+2)-	**1.336E+0(2.03E-2)**
LSMOP7	1000	1.775E+2(3.92E+2)-	**8.645E-1(2.56E-2)**	1.846E+1(1.76E+1)-	**1.295E+0(1.02E-1)**	1.035E+0(7.76E-2)-	**8.646E-1(3.86E-2)**
	3000	6.463E+1(1.14E+2)-	**8.611E-1(3.42E-2)**	1.941E+1(1.81E+1)-	**1.332E+0(1.08E-2)**	9.838E-1(3.40E-4)-	**8.509E-1(2.90E-2)**
	5000	7.810E+1(6.65E+1)-	**8.620E-1(3.42E-2)**	2.517E+1(2.13E+1)-	**1.287E+0(1.39E-1)**	9.538E-1(6.44E-2)-	**8.578E-1(4.37E-2)**
	10000	6.575E+1(5.81E+1)-	**8.658E-1(3.57E-2)**	1.947E+1(1.62E+1)-	**1.225E+0(1.92E-1)**	9.564E-1(2.23E-3)-	**8.852E-1(5.09E-2)**
LSMOP8	1000	8.917E-1(7.79E-2)-	**2.267E-1(3.18E-2)**	9.228E-1(3.17E-2)-	**2.646E-1(6.65E-2)**	5.196E-1(3.89E-3)-	**2.096E-1(1.31E-2)**
	3000	8.596E-1(6.45E-2)-	**2.249E-1(2.01E-2)**	9.230E-1(3.72E-2)-	**2.408E-1(2.73E-2)**	5.168E-1(4.47E-3)-	**1.873E-1(3.89E-2)**
	5000	8.540E-1(7.83E-2)-	**2.027E-1(4.93E-2)**	9.019E-1(5.39E-2)-	**2.347E-1(3.94E-2)**	5.165E-1(4.10E-3)-	**1.919E-1(3.96E-2)**
	10000	8.546E-1(8.51E-2)-	**2.213E-1(3.62E-2)**	8.897E-1(4.91E-2)-	**2.605E-1(9.15E-2)**	5.157E-1(3.81E-3)-	**2.033E-1(1.57E-2)**
LSMOP9	1000	4.021E+1(1.93E+0)-	**1.539E+0(4.55E-6)**	3.215E+1(2.27E+0)-	**1.539E+0(4.55E-6)**	2.268E+1(1.21E+0)-	**1.539E+0(1.49E-6)**
	3000	7.671E+1(2.48E+0)-	**1.538E+0(6.83E-6)**	6.106E+1(2.17E+0)-	**1.525E+0(6.65E-2)**	2.760E+1(1.01E+0)-	**1.538E+0(6.83E-6)**
	5000	9.386E+1(2.60E+0)-	**1.540E+0(7.48E-3)**	7.534E+1(2.33E+0)-	**1.534E+0(1.70E-2)**	2.911E+1(1.40E+0)-	**1.538E+0(4.66E-4)**
	10000	1.122E+2(1.49E+0)-	**1.538E+0(5.69E-5)**	9.441E+1(1.73E+0)-	**1.491E+0(2.10E-1)**	2.952E+1(1.12E+0)-	**1.538E+0(1.89E-5)**
+/=/−		**0/0/36**	——	**4/0/32**	——	**0/0/36**	——

mentary file. In our proposed ALMOEA, the number of neurons in the hidden layer of the MLP is set as K = 10. For training the MLP in each generation, the learning rate is set to 0.1, the number of epochs is set to only 1, and no momentum term is considered in gradient descent.

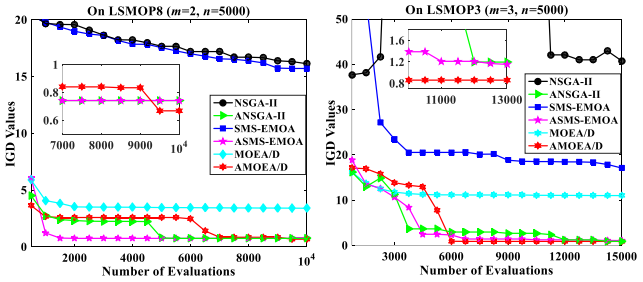


Fig. 7. The convergence profiles of three ALMOEA algorithms and their original versions on 2-objective LSMOP8 and 3-objective LSMOP3 with 5000 decision variables.

Moreover, we set the population size N = 100 and N = 153 in solving the 2-objective LMOPs and 3-objective LMOPs, respectively. Furthermore, to get a statistically sound comparison, AMLOEA and its competitors all run 20 times independently on each test LMOP. On each run, we first consider the case of allocating a relatively small computational budget with the maximum number of function evaluations $FE_{\max}$ =100N, which may be more realistic and more indicative to show whether an LMOEA is computationally efficient in solving LMOPs. Furthermore, we consider allocating a relatively high computational budget ($FE_{\max}$=1000N) to validate the performance of all selected LMOEAs.

C. *Effectiveness of the AES strategy*

Since the main difference of ALMOEA with other existing LMOEAs is the proposed AES strategy used in ALMOEA for offspring reproduction, the effectiveness of the AES strategy is first studied here by making pairwise performance comparisons between three different versions of ALMOEA (namely ANSGA-II, ASMS-EMOA, and AMOEA/D) and their corresponding original versions (namely NSGA-II, SMS-EMOA, and MOEA/D-DE [52] with the Tchebycheff decomposition method). All these six algorithms are used to solve LSMOP1-LSMOP9 problems with m = {2, 3} and n = {1000, 3000, 5000, 10000} in the case of $FE_{\max}$=100N. Besides, Table S2 and Table I record the average IGD results of these six algorithms on 2- and 3-objective LSMOP problems, respectively (Table S2 is provided in the supplementary file of this paper due to page limitation). In addition, in Table S2 and Table I, the three symbols "+", "−", and "=" represent that the performance of the corresponding competitor according to its IGD result is, respectively, better than, worse than, and similar to that of ALMOEA in solving the selected LMOPs (the same meaning in the other tables hereafter). Besides, the corresponding average HV results are also provided in Table S3 of the supplementary file. To ensure a statistically sound conclusion, the

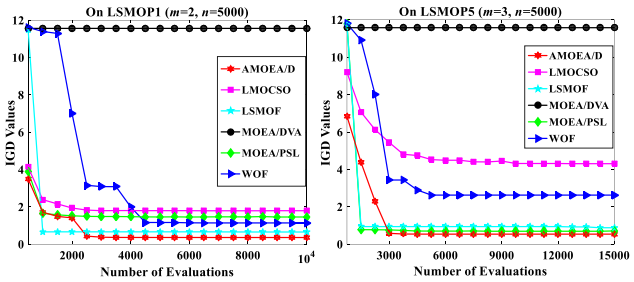


Fig. 8. The convergence profiles of AMOEA/D and its five competitors on 2-objective LSMOP1 and 3-objective LSMOP5 with n = 5000.

marking of these three symbols depends on the result of the Wilcoxon rank-sum test with a 0.05 significance level, and thus they can show statistically significant differences of the pairwise comparison between ALMOEA and its competitors.

As observed from the pairwise comparisons in both Table S2 and Table I, it is clear that the performance of three ALMOEA optimizers is significantly improved when compared with their corresponding original versions in solving most of these LSMOP benchmarks, as only SMS-EMOA can outperform ASMS-EMOA in 4 out of 36 test problems in the case of m = 3. To be specific, the difference between NSGA-II and ANSGA-II (or between SMS-EMOA and ASMS-EMOA) is that NSGA-II (or SMS-EMOA) adopts SBX to reproduce offspring of an LSMOP in its large-scale search space while ANSGA-II (or ASMS-EMOA) uses AES to do that. Thus, the significant performance improvement of ANSGA-II (or ASMS-EMOA) validates that AES is more efficient than SBX in handling the large-scale search spaces of these LSMOP problems. Similarly, by comparing the performance of AMOEA/D and MOEA/D-DE, it is also reasonable to conclude that AES shows obviously higher efficiency than DE when handling the large-scale decision space.

Moreover, to illustrate the evolutionary process of population in the above optimizers, their convergence profiles based on IGD values in solving LSMOP8 with (m = 2, n = 5000) and LSMOP3 with (m = 3, n = 5000) are depicted in Fig. 7. Obviously, we can find that the lowest point of the evolutionary curves obtained by the three versions of ALMOEA on these two LSMOP problems are much lower than that of their three competitors, which proves once again that AES can significantly improve the performance of LMOEAs in solving LMOPs. In addition, we can also find that the descending speed of the evolutionary curves for the above three ALMOEA optimizers is much faster than that of their corresponding original version in the early and middle stages, which demonstrates that the AES strategy can accelerate the convergence speed of the population to a certain extent, so as to improve the computational efficiency of ALMOEA, especially when the computation resource is limited (e.g., $FE_{\max}$ = 100N in this case). In AES, every solution will learn the possible fastest convergent direction vector via a well-trained MLP to guide its search for reproducing high-quality offspring. In this way, in the early and middle stages of evolution, the populations in ANSGA-II, AMOEA/D, and ASMS-EMOA can fast converge to a relative promising state via AES using a few function evaluations.

In this experimental study, in addition to verifying the effectiveness of AES, we can also find some meaningful observations, as described below. At first, in Fig. 7, the final average IGD values obtained by MOEA/D-DE is smaller than those obtained by NSGA-II and SMS-EMOA, which indicates that DE is more suitable than SBX in handling the search space of LSMOP3 and LSMOP8 problems. Secondly, when comparing the performance of these three ALMOEA optimizers, although the descending speed of evolutionary curve obtained by AMOEA/D is slower than that obtained by the other two ALMOEAs in the early stage, AMOEA/D can eventually get a smaller IGD value than them, as shown in Fig. 7. Therefore, it can be inferred that the performance of ALMOEA is also slightly sensitive to the used environmental selection strategy. Specifically, the selection strategy in AMOEA/D is to guarantee the diversity of the population first according to a set of uniformly distributed reference vectors, and then to maintain the convergence based on the aggregation fitness. On the contrary, in ANSGA-II (or ASMS-EMOA), the selection strategy is to ensure the population's convergence according to the domination relationship first, and then to maintain the diversity based on the crowding distance (or hypervolume indicator). Finally, the true PFs of LSMOP3 and LSMOP8 are both regular that can be matched well by a set of uniformly distributed reference vectors used in AMOEA/D. Thus, AMOEA/D is more suitable than other two ALMOEAs to solve LMOPs with regular PFs.

D. Computational Efficiency of ALMOEA

As discussed above, AES is significantly better than the traditional search strategies (e.g., SBX and DE), in terms of computational efficiency, to handle the large-scale search space of LMOPs. In this Section, to further validate the efficiency of the ALMOEA framework, AMOEA/D is selected to compare with six state-of-the-art LMOEAs on the LSMOP problems and a set of newly proposed LMF problems in the case of $FE_{\max}$=100N.

1) Comparison Results on LSMOP Problems

Here, AMOEA/D is compared with five competitive LMOEAs, namely WOF, MOEA/DVA, LMOCSO, LSMOF, and MOEA/PSL, in solving LSMOP1 to LSMOP9 problems with m = (2, 3) and n ranging from 1000 to 10000. The detailed average IGD results obtained by AMOEA/D and its five competitors on these LSMOP problems are presented in Table S4 of the supplementary file, in which the best-performing result of each case is highlighted in bold. As observed from Table S4, AMOEA/D obtains 47 best results out of 72 cases, LSMOF gets 24 best results, and LMOCSO gains only one best result, while MOEA/DVA, WOF, and MOEA/PSL doesn't obtain any best result. Thus, it's reasonable to conclude that AMOEA/D performs better than its five LMOEA competitors in solving most of these LSMOP problems. To be specific, AMOEA/D performs best mainly on LSMOP1, LSMOP2, LSMOP4, and LSMOP8 problems. Based on the specific statistic results given in the last row of Table S4, only LSMOF is comparable with AMOEA/D, as it performs better

Table II
THE IGD RESULTS OBTAINED BY AMOEA/D AND ITS FOUR COMPETITORS ON 2-OBJECTIVE LMF BENCHMARKS IN THE CASE OF $FE_{MAX} = 10^4$

Problem	n	GLEA	LSMOF	DGEA	MOEA/PSL	AMOEA/D
LMF1	1000	1.293E+1(1.82E+0)-	4.450E+1(7.92E+1)-	2.258E+1(4.85E+0)-	2.374E+2(1.76E+2)-	**7.568E-1(5.01E-3)**
	3000	1.582E+1(9.18E-1)-	1.730E+3(1.93E+3)-	2.045E+1(6.29E+0)-	4.492E+3(8.89E+3)-	**7.487E-1(2.45E-2)**
	5000	1.563E+1(6.22E-1)-	1.110E+4(1.43E+4)-	1.886E+1(3.83E+0)-	1.064E+4(1.31E+4)-	**9.693E-1(7.83E-1)**
	10000	1.504E+1(5.52E-1)-	3.434E+4(2.94E+4)-	1.495E+1(3.61E+0)-	8.242E+4(3.68E+4)-	**7.947E-1(1.58E-1)**
LMF2	1000	4.569E+0(1.12E+0)-	1.232E+0(7.25E-1)-	3.314E+0(1.37E+0)-	**4.787E-1(3.33E-2)+**	6.025E-1(1.13E-6)
	3000	7.967E+0(1.32E+0)-	1.279E+0(7.13E-1)-	3.239E+0(1.79E+0)-	**4.845E-1(3.81E-2)+**	6.015E-1(3.67E-3)
	5000	8.849E+0(1.24E+0)-	1.141E+0(3.50E-1)-	3.929E+0(1.01E+0)-	9.745E-1(7.23E-1)-	**6.029E-1(1.94E-3)**
	10000	8.882E+0(9.04E-1)-	1.159E+0(5.71E-1)-	4.465E+0(6.38E-1)-	6.979E-1(2.25E-1)-	**6.270E-1(1.13E-1)**
LMF3	1000	3.511E+1(1.25E+1)-	2.272E+2(1.07E+2)-	6.567E+2(4.75E+2)-	7.672E-1(3.50E-4)-	**7.598E-1(1.33E-1)**
	3000	8.278E+1(2.42E+1)-	2.573E+3(1.01E+3)-	1.589E+4(1.43E+4)-	7.274E-1(3.65E-5)=	**7.273E-1(6.46E-2)**
	5000	1.189E+2(4.23E+1)-	1.323E+4(7.81E+3)-	7.649E+4(5.98E+4)-	1.081E+2(2.31E+2)-	**8.549E-1(6.35E-1)**
	10000	2.548E+2(6.92E+1)-	6.852E+4(4.52E+4)-	3.467E+5(3.72E+5)-	2.748E+1(8.47E+1)-	**2.229E+0(2.88E+0)**
LMF4	1000	1.186E+4(4.51E+3)-	7.162E+0(8.11E+0)-	3.997E+1(1.05E+1)-	2.351E+1(2.80E+1)-	**4.159E-1(5.57E-2)**
	3000	2.415E+4(6.41E+3)-	8.886E+1(5.42E+1)-	4.795E+1(1.31E+1)-	4.791E+2(4.92E+2)-	**4.311E-1(3.70E-2)**
	5000	2.629E+4(5.60E+3)-	1.809E+2(1.48E+2)-	4.891E+1(1.41E+1)-	1.272E+3(1.05E+3)-	**3.904E+1(1.71E+2)**
	10000	3.040E+4(8.27E+3)-	2.358E+3(2.59E+3)-	5.032E+1(1.63E+1)-	4.128E+3(2.85E+3)-	**4.862E-1(2.91E-1)**
LMF5	1000	1.237E+1(7.83E-1)-	1.451E+0(9.11E-2)+	2.122E+0(5.00E-1)-	**9.911E-1(9.10E-1)+**	1.814E+0(5.42E+0)
	3000	1.508E+1(4.77E-1)-	1.723E+0(2.41E-2)-	2.320E+0(2.53E-1)-	8.193E+1(9.45E+1)-	**6.002E-1(8.83E-3)**
	5000	1.612E+1(6.59E-1)-	1.805E+0(2.71E-2)-	1.896E+0(4.47E-1)-	1.206E+2(1.71E+2)-	**1.581E+0(4.37E+0)**
	10000	1.682E+1(6.96E-1)-	1.892E+0(2.43E-2)-	2.034E+0(4.27E-1)-	4.236E+2(6.68E+2)-	**6.369E-1(1.43E-1)**
LMF6	1000	7.187E+0(2.36E+0)-	2.338E+0(3.49E+0)-	2.420E+1(3.17E+1)-	8.972E-1(1.15E-1)-	**7.108E-1(1.08E-2)**
	3000	6.122E+0(1.29E+0)-	2.480E+1(4.19E+1)-	4.676E+2(5.28E+2)-	9.316E-1(1.18E-1)-	**7.210E-1(6.07E-2)**
	5000	6.390E+0(3.33E+0)-	1.216E+2(2.68E+2)-	1.317E+3(1.76E+3)-	9.490E-1(1.27E-1)-	**7.400E-1(1.32E-1)**
	10000	7.233E+0(4.12E+0)-	9.933E+2(1.39E+3)-	1.051E+4(1.56E+4)-	1.040E+0(6.74E-2)-	**7.217E-1(6.27E-2)**
LMF7	1000	9.475E+3(5.25E+3)-	2.955E+1(6.01E+1)-	6.345E+2(4.20E+2)-	6.076E+1(5.13E+1)-	**1.216E+0(2.09E+0)**
	3000	1.886E+4(3.53E+3)-	3.559E+2(3.73E+2)-	7.548E+3(4.39E+3)-	6.974E+2(4.66E+2)-	**4.710E+1(2.04E+2)**
	5000	2.148E+4(2.56E+3)-	**1.172E+3(7.10E+2)+**	2.591E+4(2.05E+4)-	2.646E+3(1.58E+3)+	7.989E+3(1.61E+4)
	10000	2.566E+4(2.98E+3)-	1.936E+4(3.81E+4)-	9.328E+4(5.29E+4)-	8.188E+3(3.65E+3)-	**7.667E+3(2.07E+4)**
LMF8	1000	1.872E+3(8.26E+2)-	3.888E+0(2.91E+0)-	9.313E+1(4.70E+1)-	8.334E+0(1.44E+1)-	**9.523E-1(6.28E-3)**
	3000	8.357E+3(4.15E+3)-	1.611E+1(1.26E+1)-	3.009E+2(4.48E+2)-	1.198E+1(1.77E+1)-	**9.571E-1(2.02E-2)**
	5000	1.191E+4(5.61E+3)-	1.164E+2(1.13E+2)-	1.403E+3(1.86E+3)-	1.548E+1(1.99E+1)-	**1.807E+0(3.80E+0)**
	10000	1.553E+4(6.86E+3)-	5.036E+2(4.95E+2)-	1.085E+4(1.82E+4)-	9.447E+0(1.50E+1)-	**9.616E-1(1.33E-2)**
+/=/-		0/0/32	2/0/30	0/0/32	4/1/27	——

than AMOEA/D in 22 out of 72 LSMOP problems, while other four LMOEA competitors are almost impossible to perform better than AMOEA/D on any LSMOP problem. Due to page limitation, the corresponding average HV results on these LSMOP problems are also presented in Table S5 of the supplementary file.

After comparing the performance of AMOEA/D with its five competitors, we also depict their convergence profiles on 2-objective LSMOP1 and 3-objective LSMOP5 with 5000 decision variables in Fig. 8 to reflect the evolutionary process of their populations. As shown in Fig. 8, the evolutionary curve obtained by MOEA/DVA keep unchanged with an extremely poor performance during the whole process. The reason behind this is that the computational budgets (i.e., $FE_{\max}$=100N) in MOEA/DVA are all used to do variable analysis without actual optimization of the target LSMOP with many variables ($n \geq 1000$ in this work). In this context, MOEA/DVA can only get an initial random population when allocating such a few computational budgets. In addition, the evolutionary curve obtained by LMOCSO is basically very flat, so that the final obtained IGD performance is also very poor on these two LSMOP problems. Similar to the proposed AES, the CSO-based strategy proposed in LMOCSO also searches directly in the large-scale decision space of the target LSMOP for reproduction. However, the performance of LMOCSO is significantly worse than that of AMOEA/D, which validates that the search efficiency of CSO is inferior to AES in directly handling the large-scale search space. Finally, the large-scale search space of the target LSMOP in WOF, LSMOF, and

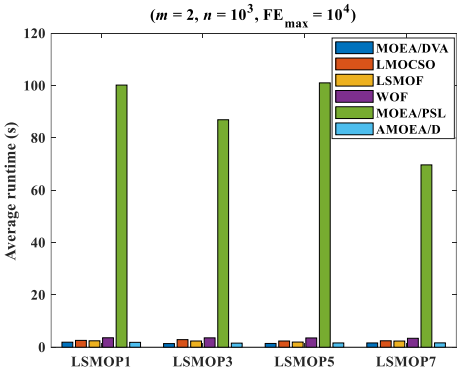


Fig. 9 The average real runtime of AMOEA/D and its five competitors on 2-objective LSMOP1 LSMOP3, LSMOP5, and LSMOP7 problems.

MOEA/PSL are all reduced by specific tricks as introduced in Section I, aiming to improving their computational efficiency by searching in the reduced small-scale space. As shown in Fig. 8, it is clear that their obtained evolutionary curves decrease faster in the early stage of evolution than other LMOEAs, but they quickly fall into stagnation without any change of the following obtained IGD values. Thus, although searching in the dimension-reduced subspace can speed up the convergence toward the optima to some extends, they may easily fall into local optima as some potential optimal space of the target LMOP cannot be fully explored. In contrast, the proposed framework ALMOEA can avoid the above shortcomings by directly searching in the original large-scale decision space with AES to accelerate the convergence speed of the population.

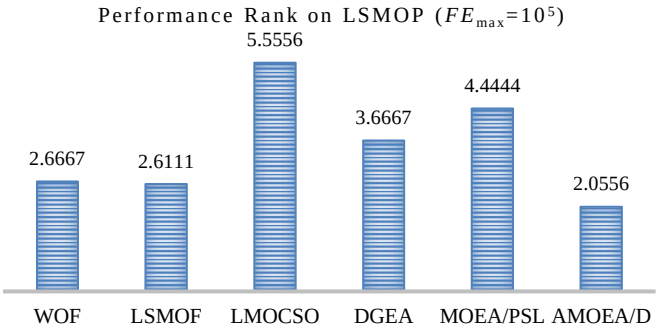


Fig. 10. Illustration of the average performance ranks over nine 2-objective LSMOPs with n = 1000 and FE_{max} = 10^5.

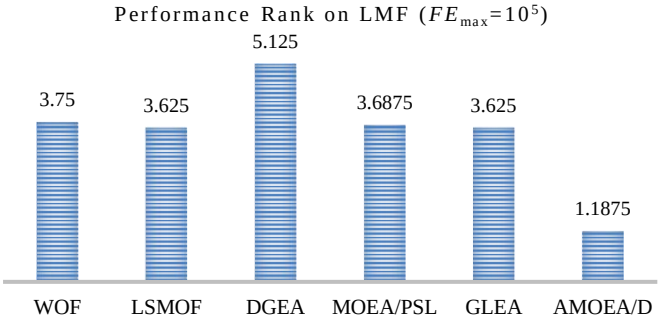


Fig. 11. Illustration of the average performance ranks over eight 2-objective LMFs with n = 1000 and FE_{max} = 10^5.

To further evaluate the actual runtime of AMOEA/D and its five competitors, their average running times (in seconds: s) from 20 runs are plotted in Fig. 9 on the 2-objective LSMOP1, LSMOP3, LSMOP5, and LSMOP7 problems with 1000 variables. Observing from Fig. 9, AMOEA/D shows the similar runtime to MOEA/DVA with the fastest speed on these four LSMOPs, which takes less time than that of WOF, LSMOF, and LMOCSO. Considering MOEA/PSL, it shows the longest running time and is incomparable with other algorithms. The reason behind this is that MOEA/PSL needs to online train an autoencoder model to learn the Pareto subspace at each generation with adaptively changing numbers of hidden layer neurons and many epochs, which requires a worst time complexity $O(mN^2 + Nn^2E)$, where E indicates the number of epochs and E = 10 in this study. Therefore, although training a machine learning model may not cost function evaluations, it is still necessary to consider whether the other cost of learning this model is affordable and worth the benefits in terms of the overhead imposed by the model (e.g., the cost of runtime and memory). Promisingly, we only train a very simple MLP model (e.g., K = 10) in AMOEA/D with less computation overhead (e.g., E = 1), but can improve its performance significantly in handling the large-scale search space. All the solvers were run on a personal computer having an Intel (R) Core (TM) i7-8700 CPU, 3.70GHz (processor), and 32 GB (RAM).

2) Comparison Results on LMF Problems

To further highlight the higher efficiency of ALMOEA, the newly proposed test suite LMF is selected here to be solved. Since MOEA/DVA spends all the computation resource on variable analysis, resulting in low efficiency, we do not consider it in this comparative study. Besides, we have validated that AES is more efficient than CSO in solving LSMOPs, so we replace LMOCSO with DGEA in this experiment. In addition, GLEA is included here to replace WOF for performance comparison with AMOEA/D in solving LMF1 to LMF8 with m = 2 and n ranging from 1000 to 10000, as it is specially tailored for LMF with a self-supervised weighted optimization framework. The detailed average IGD results obtained by AMOEA/D and its competitors are presented in Table II, where AMOEA/D obtains 28 best results in total 32 problems. Thus, we can conclude that AMOEA/D is more efficient than these four competitors in handling the large-scale search space of these LMFs, even though some of them simplify the search space with dimensionality reduction techniques (i.e., GLEA, LSMOF, and MOEA/PSL) or adaptively guide the search with diverse directions in the original search space (i.e., DGEA). Specifically, the LMOPs designed in LMF pose great challenges to existing LMOEAs due to the more complex features, such as mixed function formulation, mixed variable linkages, and unbalanced contribution of variables to the objective function. Thus, the high efficiency of AMOEA/D on these LMFs shows the effectiveness of our proposed AES in handling the large-scale search space of these LMFs. It is worth noting that the IGD results obtained by AMOEA/D on LMF4 and LMF7 problems are relatively worse. It is attributed to the fact that both LMF4 and LMF7 are too troublesome to be solved, which have convex Pareto front, partially separable decision variables, and multimodal landscapes. Even so, the computational efficiency of AMOEA/D is still significantly improved by AES when compared with its four competitors in solving LMFs.

3) More Discussions

To visually show and support the above discussions, Fig. S1 and Fig. S2 provided in the supplementary file plot the final populations obtained by AMOEA/D and its six competitors with the median IGD values from 20 runs in solving the 2-objective LSMOP and LMF problems with n = 3000. From these figures, we can see that the population of AMOEA/D converges faster than that of its competitors in solving most LSMOP and LMF problems, as it can find solutions closer to the true PF of these problems. However, we can also find that the non-dominated solutions obtained by AMOEA/D are more likely to fall into local optima on most multimodal problems, such as LSMOP3, LSMOP6-7, LMF3, LMF5-6, and LMF8 problems. To further verify this observation, we record the proportion of non-dominated solutions in the current population and the survival rate of the newly generated offspring solutions in each generation, respectively, as shown in Fig. S3 to Fig. S6 provided in the supplementary file. From these figures, we can learn two observations about AMOEA/D in solving these LSMOP and LMF problems: i) a small proportion of non-dominated solutions on most cases except for LSMOP2 and LSMOP4 that are easier to solve the problems with higher dimensions [20]; ii) a low survival rate of newly generated solutions on all cases in the middle and later stages of evolution. Therefore, in **Algorithm 1**, although the current population **P** can be divided into two subsets $\mathbf{S_p}$ and $\mathbf{S_e}$ with

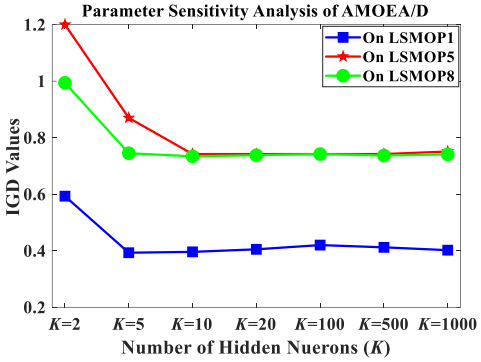


Fig. 12 Parameter sensitivity analysis of K in AMOEA/D

different degree of convergence by non-dominated sorting method on most cases, the acceleration effect of the MLP trained by such datasets on evolutionary search is obviously weakened in the middle and late stages (low survival rate of newly generated solutions). The reasons behind this may be multifaceted, e.g., the trained MLP may be overfitting in the middle and later stages or the evolutionary population may fall into local optima, which requires more comprehensive analysis in our future work.

E. Performance Study with More Function Evaluations

In Section IV. D, the high efficiency of ALMOEA is validate by solving LSMOP and LMF problems in the case of $FE_{\max}$=100N. Here, a relatively sufficient computation resource (i.e., $FE_{\max}$=10^5) is allocated to study the performance of ALMOEA in solving 2-objective LSMOP and LMF problems with n = 1000. In particular, AMOEA/D is selected to compare with WOF, LSMOF, LMOCSO, DGEA, GLEA, and MOEA/PSL on these benchmarks. The average IGD results on LSMOP and LMF are presented in Table S6 and Table S7 of the supplementary file, respectively. It can be observed that AMOEA/D still shows certain advantages in solving these two suites of LMOPs, especially on LMF problems. Besides, to quantify how well each optimizer performs on the LSMOP and LMF problems, AMOEA/D and its competitors are ranked by the Friedman's test [58] in Fig. 10 and Fig. 11, where the lower the rank value, the better the performance of the optimizer on the corresponding test suite. In these two figures, AMOEA/D obtains the best performance ranks in both cases, as it obtains the rank scores of 2.0556 on LSMOP problems and 1.1875 on LMF problems, respectively, which are much lower than that obtained by any of its competitors. Therefore, it is reasonable to conclude that AMOEA/D also shows significantly better performance than its LMOEA competitors on these LSMOP and LMF problems with $FE_{\max}$=10^5.

Nevertheless, we can also find that the advantages of AMOEA/D are much weaker than those reflected in the case of $FE_{\max}$ = 100N, and even there is no improvement in the performance of solving some LSMOP and LMF problems with additional 900N function evaluations (e.g., on LSMOP3, LSMOP5, LSMOP8, LMF1-4, and LMF9). Thus, AMOEA/D is obviously stuck in local optima at the early stage of evolution when solving these problems in the case of $FE_{\max}$ = 10^5. In this context, we redefine the AES strategy in (8) as:

$$\mathbf{x}^{new}=\begin{cases}\mathbf{x}+r_1(\mathbf{x}-\mathbf{x}^{gdv})+r_2(\mathbf{x}^{d_1}-\mathbf{x}^{d_2}) & \text{if } FE<0.1FE_{\max}\\ \mathbf{x}+r_1(\mathbf{x}^{e}-\mathbf{x})+r_2(\mathbf{x}^{d_1}-\mathbf{x}^{d_2}) & \text{otherwise}\end{cases}, \tag{10}$$

where $\mathbf{x}^{e}$ is a random solution selected from $\mathbf{S_e}$. With such a simple change, the performance of AMOEA/D in solving those LSMOP and LMF problems is greatly improved, as the AMOEA/D-E shown in Table S8 of the supplementary file. Therefore, designing some hybrid or adaptive multi-stage search strategies based on the proposed AES is a promising direction to improve the performance of ALMOEA, which will be further explored in our future work.

F. Ablation Study of AES in ALMOEA

In this section, the ablation study of AES is conducted to investigate the effectiveness of its two main components (i.e., the self-acceleration component and differential-regulation component) in AES, as described in Section III.B. We propose three different ablated versions of AMOEA/D, named AMOEA/D1 to AMOEA/D3, respective. Specifically, the differential-regulation term in (8) is removed from the AES of AMOEA/D1, the acceleration term is not included in the AES of AMOEA/D2, and a random solution selected from $\mathbf{S_e}$ is used to replace the GDV for guiding the AES of AMOEA/D3. Then, AMOEA/D is compared with its three variants, respectively, in solving LSMOP1 to LSMOP9 problems with m = (2, 3) and n ranging from 1000 to 10000 ($FE_{\max}$ = 100N). The IGD results obtained by AMOEA/D and its three ablated versions on these LSMOP problems are presented in Table S9 of the supplementary file, where only AMOEA/D1 performs better than AMOEA/D in 1 out of 72 test cases. Intuitively, although AMOEA/D1 performs worse than AMOEA/D in solving these LSMOPs, it shows significantly better performance when compared to AMOEA/D2 and AMOEA/D3. Thus, the self-acceleration component based on the learned GDV of each solution can significantly improve the searching efficiency of AES when handling the large-scale search space. Nevertheless, AMOEA/D1 without the differential-regulation term may easily fall into local optima on these LSMOPs, especially on LSMOP3, LSMOP6, and LSMOP7 with multi-modal landscapes. Therefore, both these two components in AES are indispensable in the evolutionary process, which can cooperatively accelerate the population's convergence speed and avoid falling into local optima. Moreover, although a random solution selected from $\mathbf{S_e}$ can guide the solutions in $\mathbf{S_p}$ to search in an appropriate direction, the solutions in $\mathbf{S_e}$ may blind-guided with insufficient convergence pressure, resulting in low search efficiency. Thus, the GDV learned by the trained MLP can not only guide the search of the poor solutions in its likely fastest convergence direction, but also guide the efficient search of the elite solutions.

G. Parameter Sensitivity Analysis of ALMOEA

In this paper, we use an MLP to learn the GDV of each solution for driving its rapid convergence, and a very important parameter in this MLP is the number of neurons (i.e., K) in its hidden layer, which may affect the architecture and learning ability of this MLP model. Here, we conduct the sensitivity analysis of K in affecting the performance of

AMOEA/D. Specifically, seven different values of K, ranging from 2 to 1000, are considered in AMOEA/D for solving 2-objective LSMOP1, LSMOP5, and LSMOP8 with 1000 variables ($FE_{max}=10^4$). As shown in Fig. 12, AMOEA/D is least effective in solving these three LSMOPs when K is set to 2. The reason behind this may be that the MLP model becomes relatively simple when $K = 2$, and this model is insufficient to learn the GDV of each solution, which leads to low efficiency of AES. In addition, AMOEA/D obtains the similar performance on these three LSMOPs when $K \geq 20$. Therefore, a larger value of K does not necessarily improve the performance of the algorithm, as a large amount of data are required to train a complex MLP model. Furthermore, Fig. S7 of the supplementary file plots the real runtime in solving these nine LSMOPs with different values of K, including $K = \{10, 100, 300, 500, 1000\}$. Clearly, the running time of AMOEA/D is determined by the value of K in its MLP. When K is too large (e.g., K>100 here), online training of the MLP will bring very expensive computational overhead to the algorithm, but such an MLP may not improve its ability to learn the GDV of each solution. As discussed in Section IV.D.3), there are various reasons behind it, which are worthy of further analysis in our future work. In general, the performance of an MLP strongly depends on the training data prepared. It's hard to ensure that the online data (solutions) obtained during the evolutionary process serves the purpose for which we want the MLP to learn. Besides, training this MLP itself is an optimization problem, which may incur various issues, such as selection of optimizer, type of training, overfitting, hyperparameter setting, etc. These issues are not the focus of our consideration in this work, and our motivation is to train a simple MLP to learn the GDV of each solution, followed by guiding it search in the relatively promising direction with high efficiency. Consequently, considering both the performance and computation complexity of AMOEA/D, we suggest not to set the value of K too small ($K < 5$) or too large (K>100) when solving these LSMOPs with $n = 1000$.

V. Conclusions and Future Work

In this paper, an accelerated LMOEA framework has been proposed to solve various LMOPs. To accelerate the convergence speed and improve the computational efficiency, ALMOEA adopts a novel evolutionary search strategy called AES based on an MLP for offspring reproduction. In AES, each solution learns the GDV via a well-trained MLP and then the evolutionary search along these direction vectors can reproduce high-quality offspring more efficiently. In the MLP, the number of neurons in the input and output layers is equal to the dimension of the search space. To obtain appropriate training data for MLP, each poor solution in the current population is labeled with an elite that is considered most likely to guide it with rapid convergence. Then, this neural network is updated via backpropagation with gradient descent on the above specially prepared training data. When embedding three classic MOEAs (i.e., NSGA-II, SMS-EMOA and MOEA/D) into the ALMOEA framework, their convergence speed and performance can be significantly enhanced when compared to the original MOEAs. In addition, when compared to seven competitive LMOEAs, ALMOEA also shows its superiority on various test suites, including LSMOP and LMF. At last, the effectiveness of the main components in AES (i.e., the self-acceleration component and the differential-regulation component) in offspring reproduction is also experimentally validated in this paper.

In our future work, the neural network model and its inner construction will be further studied, aiming to construct a more effective model to solve difficult LMOPs with irregular PS and multi-modal landscape. Since our AES strategy can accelerate the convergence speed by directly searching in the original decision space, it may be more suitable for real-life LMOPs with limited computational budgets, which will be considered as part of our future work.

References

[1] X. Zhou, A. K. Qin, M. Gong, and K. C. Tan. "A Survey on Evolutionary Construction of Deep Neural Networks," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 5, pp. 894-912, 2021.

[2] Y. Mei, X. Li, X. Yao, "Cooperative Coevolution with Route Distance Grouping for Large-Scale Capacitated Arc Routing Problems," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 3, pp. 435-449, 2012.

[3] X. Zhang, K. Zhou, H. Pan, L. Zhang, X. Zeng, and Y. Jin, "A Network Reduction-Based Multiobjective Evolutionary Algorithm for Community Detection in Large-Scale Complex Networks," *IEEE Transactions on Cybernetics*, vol. 50, no. 2, pp. 703-716, 2020.

[4] L. M. Antonio and C. A. C. Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," *IEEE Congress on Evolutionary Computation*, pp. 2758-2765, 2013.

[5] 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, 2015.

[6] P. K. Lehre, X. Yao, "On the Impact of Mutation-Selection Balance on the Runtime of Evolutionary Algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 2, pp. 225-241, 2012.

[7] K. Deb and 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*, vol. 18, no. 4, pp. 577-601, 2014.

[8] Y. D. Valle, et al., "Particle Swarm Optimization: Basic Concepts, Variants and Applications in Power Systems," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 2, pp. 171-195, 2008.

[9] S. Das and P. N. Suganthan, "Differential Evolution: A Survey of the State-of-the-Art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4-31, 2011.

[10] Y. Tian, et al., "Evolutionary Large-Scale Multi-Objective Optimization: A Survey," *ACM Computing Surveys*, vol. 54, no. 8, pp. 1-34, 2021.

[11] M. N. Omidvar, X. Li, X. Yao, "A review of population-based metaheuristics for large-scale black-box global optimization: Part B," *IEEE Transactions on Evolutionary Computation*, in press, 2021.

[12] W. Hong, P. Yang and K. Tang, "Evolutionary Computation for Large-scale Multi-objective Optimization: A Decade of Progresses," Int. J. Autom. Comput. vol. 18, no. 2, pp. 155-169, 2021.

[13] X. Yao, Q. Zhao, D. Gong, and S. Zhu, "Solution of Large-scale Many-objective Optimization Problems Based on Dimension Reduction and Solving Knowledge Guided Evolutionary Algorithm," *IEEE Transactions on Evolutionary Computation*, in press, 2021.

[14] X. Ma, et al., "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, 2016.

[15] 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, 2018.

[16] A. Song, Q. Yang, W. Chen, and J. Zhang, "A random-based dynamic grouping strategy for large scale multi-objective optimization," *IEEE Congress on Evolutionary Computation (CEC)*, pp. 468-475, 2016.

[17] 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*, in press, 2021.

[18] B. Cao, J. Zhao, Z. Lv, and X. Liu, "A Distributed Parallel Cooperative Coevolutionary Multiobjective Evolutionary Algorithm for Large-Scale Optimization," *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 2030-2038, 2017.

[19] 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, 2018.

[20] 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*, in press, 2021.

[21] C. He, et al., "Accelerating Large-Scale Multiobjective Optimization via Problem Reformulation," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 6, pp. 949-961, 2019.

[22] 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, 2021.

[23] H. Qian and Y. Yu, "Solving High-Dimensional Multi-Objective Optimization Problems with Low Effective Dimensions," in *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI'17)*, 2017.

[24] R. Liu, R. Ren, J. Liu, and J. Liu, "A clustering and dimensionality reduction based evolutionary algorithm for large-scale multiobjective problems," *Applied Soft Computing*, vol. 89, 106120, pp. 1-18, 2020.

[25] 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, 2021.

[26] 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*, in press, 2020.

[27] Y. Tian, X. Zhang, C. Wang, and Y. Jin, "An Evolutionary Algorithm for Large-Scale Sparse Multiobjective Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 380-393, 2020.

[28] Li. 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*, in press, 2021.

[29] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient Large-Scale Multiobjective Optimization Based on a Competitive Swarm Optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696-3708, 2020.

[30] 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, in press, 2021.

[31] 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, 2019.

[32] C. He, R. Cheng, and D. Yazdani, "Adaptive Offspring Generation for Evolutionary Large-Scale Multiobjective Optimization," *IEEE Transactions on Systems*, Man, and Cybernetics: Systems, in press, 2020.

[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, 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*, in press, 2021.

[35] A. Ghosh, E. Goodman, K. Deb, R. Averill, and A. Diaz, "A large-scale bi-objective optimization of solid rocket motors using innovization," in IEEE Congress on Evolutionary Computation (CEC), pp. 1–8, 2020.

[36] A. Gaur and K. Deb, "Effect of size and order of variables in rules for multi-objective repair-based innovization procedure," in 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 2177–2184.

[37] S. Mittal, D. K. Saxena, K. Deb, and E. D. Goodman, "A learning-based innovized progress operator for faster convergence in evolutionary multi-objective optimization," ACM Trans. Evol. Learn. Optim., vol. 2, no. 1, 2021. Available: https://doi.org/10.1145/3474059.

[38] S. Mittal, D. K. Saxena, and K. Deb, "Learning-based multiobjective optimization through ANN–assisted online innovization," in Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion, ser. GECCO 20. New York, NY, USA: Association for Computing Machinery, pp. 171–172, 2020.

[39] S. Mittal, D. K. Saxena, K. Deb, E. D. Goodman, "Enhanced Innovized Progress Operator for Evolutionary Multi- and Many-objective Optimization," *IEEE Transactions on Evolutionary Computation*, in press, 2021, doi: 10.1109/TEVC.2021.3131952.

[40] S. K. Pal, and S. Mitra, "Multilayer perceptron, fuzzy sets, and classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, pp. 683-697, 1992.

[41] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182-197, 2002.

[42] M. Li, S. Yang, and X. Liu, "Pareto or Non-Pareto: Bi-Criterion Evolution in Multiobjective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 5, pp. 645-665, 2016.

[43] S. Jiang, J. Zhang, Y. Ong, A. N. Zhang, and P. S. Tan, "A Simple and Fast Hypervolume Indicator-Based Multiobjective Evolutionary Algorithm," *IEEE Transactions on Cybernetics*, vol. 45, no. 10, pp. 2202-2213, 2015.

[44] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: An Improved Two-Archive Algorithm for Many-Objective Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 524-541, 2015.

[45] K. Li, K. Deb, Q. Zhang, and S. Kwong, "An Evolutionary Many-Objective Optimization Algorithm Based on Dominance and Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 5, pp. 694-716, 2015.

[46] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712-731, 2007.

[47] S. Liu, Y. Yu, Q. Lin, and K. C. Tan, "An Adaptive Clustering-based Evolutionary Algorithm for Many-objective Optimization Problems," *Information Sciences*, vol. 537, pp. 261-283, 2020.

[48] S. Liu, J. Zheng, Q. Lin, and K. C. Tan, "Evolutionary Multi and Many-objective Optimization via Clustering for Environmental Selection," *Information Sciences*, vol. 578, pp. 930-949, 2021.

[49] R. J. Preen, S. W. Wilson, and L. Bull, "Autoencoding with a Classifier System," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 6, pp. 1079-1090, 2021.

[50] X. Yu, M. O. Efe, and O. Kaynak, "A General Backpropagation Algorithm for Feedforward Neural Networks Learning," *IEEE Transactions on Neural Networks*, vol. 13, no. 1, pp. 251-254, 2002.

[51] Q. Lin, et al., "Particle Swarm Optimization with a Balanceable Fitness Estimation for Many-Objective Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 1, pp. 32-46, 2018.

[52] H. Li and Q. Zhang, "Multiobjective Optimization Problems with Complicated Pareto Sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284-302, 2009.

[53] P. A. N. Bosman and D. Thierens, "The balance between proximity and diversity in multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 174-188, 2003

[54] 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, 2017.

[55] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjec-

tive Selection Based on Dominated Hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.
[56] L. While, L. Bradstreet, L. Barone, "A Fast Way of Calculating Exact Hypervolumes," IEEE Transactions on Evolutionary Computation, vol. 16, no. 1, pp. 86-95, 2012.
[57] 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, pp. 73–87, 2017.
[58] 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*, in press, 2020.

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 worked for ShenZhen TVT Digital Technology Co., Ltd as a software engineer from 2013 to 2015, and he worked for Shenzhen University as a research assistance from 2018 to 2019.

He is currently a PhD student in Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include nature-inspired computation, evolutionary transfer optimization, and evolutionary large-scale optimization.

Jun Li received the B.S. degree from Tianjin University of Commerce, Tianjin, China, in 2020. He is currently a Master student in the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen, China.

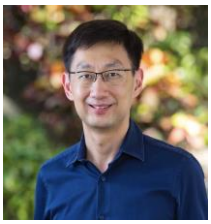
His current research interests include evolutionary multiobjective optimization, evolutionary large-scale multiobjective optimization, and their applications.

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. His current research interests include artificial immune system, multi-objective optimization, and dynamic system.

Ye Tian received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University, Hefei, China, in 2012, 2015, and 2018, respectively.

He is currently an Associate Professor with the Institutes of Physical Science and Information Technology, Anhui University, Hefei, China. His research interests include evolutionary computation and its applications. He was the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation outstanding paper awards, 2020 IEEE Computational Intelligence Magazine outstanding paper award, and 2022 IEEE Computational Intelligence Society outstanding Ph.D. dissertation award.



Kay Chen Tan (Fellow, IEEE) received the B.Eng. degree (First Class Hons.) and the Ph.D. degree from the University of Glasgow, U.K., in 1994 and 1997, respectively. He is currently a Chair Professor (Computational Intelligence) of the Department of Computing, the Hong Kong Polytechnic University. He has published over 300 refereed articles and seven books.

Prof. Tan is currently the Vice-President (Publications) of IEEE Computational Intelligence Society, USA. He served as the Editor-in-Chief of the IEEE Computational Intelligence Magazine from 2010 to 2013 and the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION from 2015 to 2020. He currently serves as an Editorial Board Member for more than ten journals. Prof. Tan is an IEEE Distinguished Lecturer Program (DLP) Speaker and the Chief Co-Editor of Springer Book Series on Machine Learning: Foundations, Methodologies, and Applications.