Solving Large-Scale Multiobjective Optimization Problems With Sparse Optimal Solutions via Unsupervised Neural Networks

Ye Tian, Chang Lu, Xingyi Zhang, *Senior Member, IEEE*, Kay Chen Tan, *Fellow, IEEE*, and Yaochu Jin, *Fellow, IEEE*

***Abstract*—Due to the curse of dimensionality of search space, it is extremely difficult for evolutionary algorithms to approximate the optimal solutions of large-scale multiobjective optimization problems (LMOPs) by using a limited budget of evaluations. If the Pareto-optimal subspace is approximated during the evolutionary process, the search space can be reduced and the difficulty encountered by evolutionary algorithms can be highly alleviated. Following the above idea, this article proposes an evolutionary algorithm to solve sparse LMOPs by learning the Pareto-optimal subspace. The proposed algorithm uses two unsupervised neural networks, a restricted Boltzmann machine, and a denoising autoencoder to learn a sparse distribution and a compact representation of the decision variables, where the combination of the learnt sparse distribution and compact representation is regarded as an approximation of the Pareto-optimal subspace. The genetic operators are conducted in the learnt subspace, and the resultant offspring solutions then can be mapped back to the original search space by the two neural networks. According to the experimental results on eight benchmark problems and eight real-world problems, the proposed algorithm can effectively solve sparse LMOPs with 10 000 decision variables by only 100 000 evaluations.**

Manuscript received September 19, 2019; revised January 14, 2020; accepted March 7, 2020. This work was supported in part by the Key Project of Science and Technology Innovation 2030 supported by the Ministry of Science and Technology of China under Grant 2018AAA0100105, in part by the National Natural Science Foundation of China under Grant 61672033, Grant 61822301, Grant 61876123, Grant 61906001, and Grant U1804262, in part by the Hong Kong Scholars Program under Grant XJ2019035, in part by the Anhui Provincial Natural Science Foundation under Grant 1808085J06 and Grant 1908085QF271, in part by the State Key Laboratory of Synthetical Automation for Process Industries under Grant PAL-N201805, in part by the Research Grants Council of the Hong Kong Special Administrative Region, China, under Grant CityU11202418 and Grant CityU11209219, and in part by the Royal Society International Exchanges Program under Grant IEC\NSFC\170279. This article was recommended by Associate Editor M. Zhang. *(Corresponding author: Xingyi Zhang.)*

Ye Tian is with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Institutes of Physical Science and Information Technology, Anhui University, Hefei 230601, China, and also with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: field910921@gmail.com).

Chang Lu and Xingyi Zhang are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230601, China (e-mail: lucy_lc@outlook.com; xyzhanghust@gmail.com).

Kay Chen Tan is with the Department of Computer Science, City University of Hong Kong, Hong Kong (e-mail: kaytan@cityu.edu.hk).

Yaochu Jin is with the Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K. (e-mail: yaochu.jin@surrey.ac.uk).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/TCYB.2020.2979930

***Index Terms*—Denoising autoencoder (DAE), large-scale multiobjective optimization, Pareto-optimal subspace, restricted Boltzmann machine (RBM), sparse Pareto-optimal solutions.**

I. Introduction

In the era of big data, there exists plenty of complicated data in many research fields and real-world applications, which raises a variety of optimization problems having multiple objectives and a large number of decision variables [1]–[3]. These large-scale multiobjective optimization problems (LMOPs) present a huge search space that grows exponentially with the number of decision variables, posing stiff challenges for evolutionary algorithms to efficiently approximate the Pareto-optimal solutions [4]. To address the inevitable "curse of dimensionality," some multiobjective evolutionary algorithms (MOEAs) have been tailored for LMOPs, which are mainly based on the following two ideas.

The first idea for solving LMOPs is decision variable decomposition, which adopts the divide-and-conquer strategy that divides the decision variables into several groups randomly or heuristically, and optimizes each group of decision variables separately. For example, NSCCGA [5] relates each decision variable to a subpopulation, then optimizes each subpopulation by NSGA-II [6]. MOEA/DVA [7] divides the decision variables into position variables, distance variables, and mixed variables, and further divides the distance variables according to their interactions on the objective functions. MOEA/DVA first optimizes each group of distance variables until the population converges, then fine-tunes all the decision variables for better diversity. LMEA [8] clusters the decision variables into convergence-related variables and diversity-related variables, and iteratively optimizes each type of variables by different strategies.

The second idea for solving LMOPs is problem transformation, which aims to convert the original LMOP into a small-scale problem, so that it can be handled by general optimizers. Different from traditional decomposition-based MOEAs (e.g., those based on hierarchical decomposition [9] or Minkowski distance [10]) using a set of weights to transfer multiobjective optimization into single-objective optimization, WOF [11] uses a set of weights to alter the decision variables, where each weight is related to multiple decision variables and the number of weights is much smaller than the number of decision variables. As a result, a small-scale problem can be established by considering the weights as variables to be optimized. LSMOF [12] defines two reference directions on a solution in search space, and searches for better solutions by moving the solutions along with the reference directions. In other words, better solutions can be found by optimizing only two weights, where each weight determines the location of the solution in a reference direction.

Despite the promising performance of these MOEAs on some LMOPs, most of them are shown to be of low efficiency or effectiveness [13]. For the decision variable decomposition-based MOEAs, a large number of function evaluations are required for detecting the interactions between decision variables, and the detecting results are probably inaccurate on functions with complicated landscapes [8]. The problem transformation-based MOEAs are more vulnerable to getting trapped in local optimums due to the loss of diversified search directions [12], though they can quickly find well-converged solutions by optimizing a small-scale problem.

According to the Karush–Kuhn–Tucker condition [14], the Pareto-optimal solutions of an LMOP constitute an $(M-1)$-dimensional piecewise continuous manifold, where M is the number of objectives and usually much smaller than the number of decision variables. That is, all the Pareto-optimal solution can fill an $(M-1)$-dimensional Pareto-optimal subspace, which accounts for a tiny proportion of the original D-dimensional search space since $M \ll D$. Therefore, the original search space is reducible if some quasioptimal solutions have been found, and the difficulty of LMOPs can be highly alleviated. In fact, this regularity property has been essential for many estimations of distribution algorithms [15], [16]. Nevertheless, these algorithms still encounter difficulties in solving LMOPs [13], [16], since the interactions between decision variables are so complex that it is difficult to learn the accurate Pareto-optimal subspace.

In this article, we propose a Pareto-optimal subspace learning-based evolutionary algorithm for solving the LMOPs whose Pareto-optimal solutions are sparse, that is, most decision variables of the Pareto-optimal solutions are zero. Such LMOPs widely exist in many real-world applications [17]–[19], but there does not exist any MOEA tailored for them so far. Specifically, the proposed algorithm, called MOEA/PSL, includes the following two main contributions.

1) The restricted Boltzmann machine (RBM) [20] and denoising autoencoder (DAE) [21] are adopted in the proposed MOEA/PSL to learn the Pareto-optimal subspace. At the beginning of each generation, the decision variables of the nondominated solutions are used to train the two neural networks, where the RBM is used to learn a sparse distribution of the decision variables and the DAE is used to learn a compact representation. The combination of the sparse distribution and compact representation is regarded as an approximation of the Pareto-optimal subspace, where genetic operators are conducted in the learnt subspace instead of the original search space. Therefore, the original search space is highly reduced.
2) A parameter adaptation strategy is designed to automatically determine the parameters in Pareto-optimal subspace learning. On the one hand, the size of hidden layers of the two neural networks is estimated according to the sparsity of the nondominated solutions. On the other hand, the ratio of offspring solutions generated in the learnt subspace is dynamically adjusted according to the number of successful offspring solutions generated at the previous generations. In this way, the proposed algorithm can adapt to different sparse LMOPs without any predefined parameter.

The remainder of this article is organized as follows. In Section II, an introduction to the sparse LMOPs in real-world applications is given, and existing Pareto-optimal subspace learning approaches are reviewed. Section III presents the proposed MOEA/PSL in detail. In Section IV, the experimental results are presented and analyzed. Finally, conclusions are drawn and future work is outlined in Section V.

II. Related Work

A. Sparse LMOPs in Real-World Applications

As listed in Table I, sparse LMOPs widely exist in many fields, including machine learning, network science, software engineering, signal processing, data mining, economics, and so on. As presented in the table, these sparse LMOPs contain 10–10^5 binary or real variables. The binary variable-based LMOPs aim to select or delete a small number of elements from a large candidate set for optimizing specific objectives. For example, the goals of feature selection [17] and instance selection [24] are to select a few features and instances from the training set for the minimum classification error, respectively. Besides, the real variable-based LMOPs aim to find the optimal values of a set of parameters, such as the weights in neural network training [37] and the original signal in sparse signal reconstruction [18].

A significant feature of these problems is that their Pareto-optimal solutions are sparse. Most of these problems consider the sparsity of the solution as an objective to be optimized (e.g., the number of selected features in the feature selection, the model complexity in the neural network training, and the sparsity of the signal in sparse signal reconstruction), hence the Pareto-optimal solutions are sparse. Although some other problems do not explicitly optimize the sparsity of the solution, the Pareto-optimal solutions are also very sparse due to their objectives. For instance, the community detection problem [38] aims to identify the community centers from all the nodes, where the number of community centers is much less than the number of nodes; the power grid fault diagnosis problem [39] aims to detect all the faulty sections, which are usually much less than the normal sections.

Although the above problems have been tackled by some MOEAs as listed in Table I, these MOEAs are restricted to the objective functions and data structure of a specific problem, which cannot be used to solve different sparse LMOPs in a black-box manner. Therefore, it is necessary to develop a generic MOEAs for solving sparse LMOPs. Besides, existing large-scale MOEAs are inefficient for solving sparse LMOPs

TABLE I
SEVENTEEN SPARSE LMOPs IN REAL-WORLD APPLICATIONS

Field	Problem	Objectives	Type and length of variables	Meaning of variables
Machine Learning	Feature selection [22]	Minimize the error of the model Minimize the number of selected features	Binary 13–617	Whether each feature is selected
	Discretization-based feature selection [23]	Minimize the error of the model Maximize the diversity of selected features	Integer 2308–12600	Cut point of each feature
	Instance selection [24]	Minimize the error of the model Minimize the number of selected instances	Binary 80–1728	Whether each instance is selected
	Neural network training [13]	Minimize the error of the model Minimize the complexity of the model	Real 641–10041	Weights of the neural network
	Neural architecture search [25]	Minimize the error of the model Minimize the complexity of the model	Binary 4096	Architecture of the neural network
	Adversarial attack on neural networks [26]	Maximize the mislead rates Minimize the degree of pixel change	Real 784–120000	Change value of each pixel
	Ensemble learning [27]	Minimize the error of the ensemble model Minimize the number of selected models	Binary 100	Whether each model is selected
Network Science	Community detection [28]	Maximize the intra-link density Minimize the inter-link density	Binary 12–6927	Whether each node is selected as center
	Critical node detection [29]	Minimize the pairwise connectivity Minimize the number of deleted nodes	Binary 235–5000	Whether each node is deleted
	Influence maximization [30]	Maximize the influence Minimize the cost	Binary 5254–15233	Whether each node is selected as seed
Software Engineering	Software product configuration [31]	Minimize the number of deleted features Minimize the number of unused features Minimize the number of known defects Minimize the sum of costs	Binary 544–62482	Whether each feature is deleted
Signal Processing	Sparse signal reconstruction [18]	Minimize the reconstruction error Minimize the sparsity of the signal	Real 512–10240	Reconstructed signal
Data Mining	Pattern mining [32]	Maximize the frequency Maximize the completeness	Binary 1000–5000	Whether each item is selected
Economics	Portfolio optimization [33]	Maximize the expected return Minimize the risk	Real 31–1318	Portfolio of the instruments
Others	Facility location [34]	Minimize the facility construction costs Minimize the distance to demand points	Binary 2500–105000	Whether each facility is selected
	Multi-objective knapsack [35]	Maximize the profit of each knapsack	Binary 250–750	Whether each item is selected
	Power grid fault diagnosis [36]	Minimize the difference between actual and expected states Minimize the difference between observed and actual states	Binary 28-107	Whether each section is faulty

due to the computationally expensive objectives. For example, large-scale MOEAs can well solve an LMOP with 5000 decision variables by using 200 000 000 function evaluations [8], but it is impractical to train a deep neural network for so many times for solving the neural architecture search problem [25]. Considering the sparse nature of Pareto-optimal solutions, the Pareto-optimal subspace can be approximated by ignoring the dimensions where the decision variables in Pareto-optimal solutions are zero. Then, MOEAs can search for better solutions in the learnt subspace instead of the original search space. In other words, the original search space is drastically reduced.

Following this idea, this article proposes a Pareto-optimal subspace learning-based MOEA for solving sparse LMOPs, which will be applied to eight real-world problems selected from Table I. In the next section, the Pareto-optimal subspace learning approaches in existing MOEAs are reviewed.

B. Existing Pareto-Optimal Subspace Learning Approaches

The Pareto-optimal subspace learning approaches in existing MOEAs are mainly based on classical machine-learning techniques. In [40], two genetic algorithms were proposed for solving MOPs with low effective dimensions. The low effective dimensions indicate that the objective functions are only related to a small proportion of the decision variables, hence the Pareto-optimal subspace can be easily learned by ignoring many useless dimensions. The algorithms in [40] adopt random embedding to ignore decision variables randomly, which is theoretically verified to keep the Pareto-optimal subspace unchanged. On the other hand, some MOEAs adopt principal component analysis (PCA) to learn the Pareto-optimal subspace directly. Based on the condition that the Pareto-optimal solutions lie on an $(M-1)$-dimensional manifold [14], a regularity model-based multiobjective estimation of the distribution algorithm was proposed in [15]. This MOEA uses an $(M-1)$-dimensional local PCA to partition the population into several clusters, and generates offspring solutions around each cluster centroid. Sun *et al.* [41] suggested a two-stage MOEA with Pareto-optimal subspace learning, called MaOEA-IT. In the first stage, the algorithm searches for some well-converged solutions by considering only the population convergence. In the second stage, it uses these solutions to learn the Pareto-optimal subspace via PCA, and generates offspring solutions in the learnt subspace for enhancing the population diversity.

Although the idea of Pareto-optimal subspace learning has been successfully adopted in a few MOEAs, they are not suited for solving sparse LMOPs due to the following reasons. First,

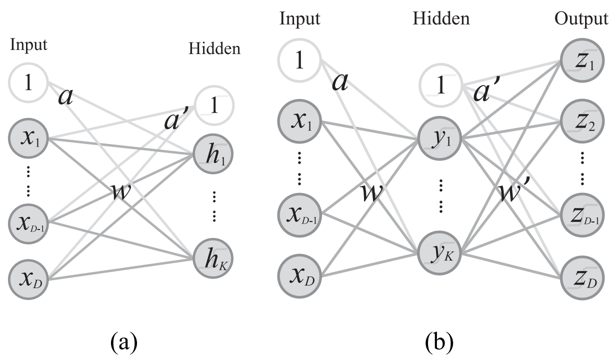

Fig. 1. General structures of (a) RBM and (b) AE.

these approaches learn the Pareto-optimal subspace by linearly reducing the decision variables, whereas the interactions between the decision variables of many real-world sparse LMOPs are nonlinear. Second, these approaches are tailored for specific types of optimization problems, which do not consider the sparse nature of LMOPs. Third, these approaches can only handle continuous decision variables, which are unsuitable for many sparse LMOPs with binary variables.

As reported in [42], a hybrid representation of the solution is effective for solving sparse LMOPs, where each solution is represented by a binary vector denoting the mask and a real vector denoting the decision variables. Following this idea, the proposed MOEA/PSL adopts RBM and DAE to learn the Pareto-optimal subspace from the solutions with hybrid representation. More specifically, RBM is adopted to learn a sparse distribution from the binary vectors due to its ability of learning the probability distribution of the input obeying a binomial distribution, and DAE is adopted to learn a compact representation from the real vectors due to its ability of learning the compact representation of the input in continuous space. By adopting both RBM and DAE, the proposed MOEA/PSL can control the sparsity of solutions and learn a low-dimensional Pareto-optimal subspace, thus highly improving the efficiency in finding sparse and well-converged solutions. In the experiments in Section IV-E, the utilization of both RBM and DAE will be verified by comparing them with a single RBM, DAE, and some other techniques.

In the next section, some basic concepts about RBM and DAE are introduced.

C. Restricted Boltzmann Machine and Denoising Autoencoder

RBM [20] is the building block of the deep belief network, which can reduce the dimensionality of input data in an unsupervised manner. As can be seen from Fig. 1(a), a typical RBM consists of an input layer and a hidden layer, where the nodes in the two layers are binary variables obeying a binomial distribution [43]. Given a vector $\mathbf{x}$ as the input, the value of each node h_j in the hidden layer is set to 1 with a probability

$$p\big(h_j = 1|\mathbf{x}\big) = \sigma\big(a_j + \Sigma_i x_i w_{ij}\big) \tag{1}$$

where a_j is the bias, w_{ij} is the weight, and $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. In practice, the value of h_j is sampled by comparing $p(h_j = 1|\mathbf{x})$ with a uniformly distributed random value within [0, 1]. Similarly, the reconstructed value of each node x_i in the input layer is set to 1 with a probability

$$p\big(x'_i = 1|\mathbf{h}\big) = \sigma\Big(a'_j + \Sigma_j h_j w_{ij}\Big). \tag{2}$$

The goal of training RBM is to minimize the reconstruction error (i.e., the difference between the reconstructed vector $\mathbf{x}'$ and the original input $\mathbf{x}$) by finding the optimal values of $\mathbf{a}$, $\mathbf{a}'$, and $\mathbf{w}$, which can be achieved by the contrastive divergence algorithm [44].

Autoencoder (AE) [45] is the building block of stacked AE, which is also used for dimensionality reduction. Different from RBM, AE is a three-layer network working in continuous space. As shown in Fig. 1(b), the value of each node y_j in the hidden layer can be calculated by

$$y_j = \sigma\big(a_j + \Sigma_i x_i w_{ij}\big) \tag{3}$$

and the value of each node z_i in the output layer can be calculated by

$$z_i = \sigma\Big(a'_i + \Sigma_j y_j w'_{ji}\Big). \tag{4}$$

The reconstruction error of AE is the difference between the output $\mathbf{z}$ and the input $\mathbf{x}$, hence AE can be trained by the same way as feedforward neural network [46]. As for DAE [21], it is a popular variant of AE that enforces the robustness by adding noise to the input. In this article, each input $\mathbf{x}$ for DAE is perturbed by randomly setting the elements of $\mathbf{x}$ to zero.

Although the above neural networks have been adopted in the estimation of the distribution algorithm [47] and the evolutionary multitasking algorithm [48], they have not been used in Pareto-optimal subspace learning for solving LMOPs before. In this article, RBM and DAE are adopted to learn a sparse distribution and a compact representation of the decision variables, respectively, where the combination of the learnt sparse distribution and compact representation is regarded as an approximation of the Pareto-optimal subspace. In the next section, the procedure of the proposed algorithm is elaborated.

III. Proposed Algorithm

A. Framework of MOEA/PSL

The general framework of MOEA/PSL is presented in Fig. 2 and Algorithm 1. To begin with, N solutions are initialized by the Latin hypercube sampling method [49] to form the initial population P, and the nondominated front number [50] and crowding distance [6] of each solution are calculated. In the main loop, N parents are selected by the binary tournament selection and used to generate N offspring solutions. The offspring set O are then combined with the population P, and N solutions with better nondominated front numbers and crowding distances in $P \bigcup O$ will survive to the next generation. Finally, the parameters ρ and K are adapted according to the new population. As a consequence, the mating selection and environmental selection of MOEA/PSL are the same as those of NSGA-II [6], while MOEA/PSL generates some offspring solutions in a learnt subspace.

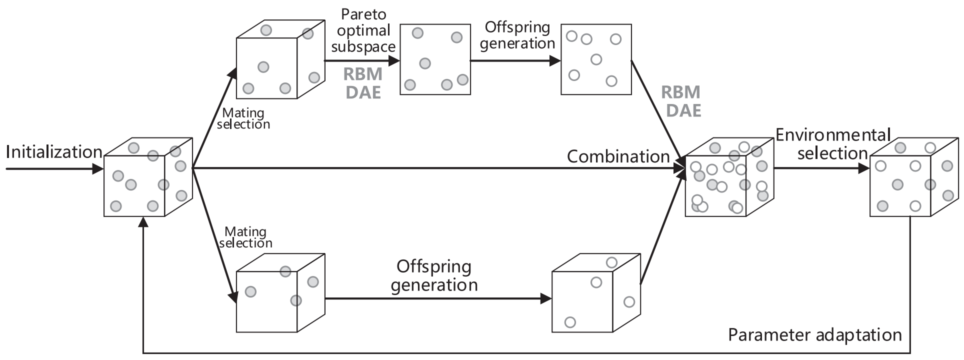

Fig. 2. Procedure of MOEA/PSL.

Algorithm 1: Framework of MOEA/PSL

```
Input: $N$ (population size)
Output: $P$ (final population)
1  $P \leftarrow Initialization(N)$;
2  $[F_1, F_2, \cdots] \leftarrow NondominatedSorting(P)$; //$F_i$ is the
       set of solutions in the $i$-th non-dominated
       front
3  $CrowdDis \leftarrow CrowdingDistance(F_1, F_2, \cdots)$;
4  $\rho \leftarrow 0.5$; //Ratio of offspring solutions
       generated in the Pareto optimal subspace
5  $K \leftarrow N$; //Size of the hidden layers
6  while termination criterion not fulfilled do
7      $P' \leftarrow$ Select $N$ parents via binary tournament selection
       according to the non-dominated front number and
       $CrowdDis$ of solutions in $P$;
8      $O \leftarrow Variation(P, P', \rho, K)$;
9      $P \leftarrow P \bigcup O$;
10     Delete duplicated solutions from $P$;
11     $[F_1, F_2, \cdots] \leftarrow NondominatedSorting(P)$;
       $CrowdDis \leftarrow CrowdingDistance(F_1, F_2, \cdots)$;
12     $k \leftarrow$ Minimum value s.t.$|F_1 \bigcup \cdots \bigcup F_i| \geq N$;
13     Delete $|F_1 \bigcup \cdots \bigcup F_k| - N$ solutions from $F_k$ with the
       smallest $CrowdDis$;
14     $P \leftarrow F_1 \bigcup \cdots \bigcup F_k$;
15     $[\rho, K] \leftarrow ParameterAdaptation(P, \rho)$;
16 return $P$;
```

In the following two sections, the core components of MOEA/PSL are described in detail.

B. Pareto-Optimal Subspace Learning and Offspring Generation in MOEA/PSL

The nondominated solutions in the population are adopted as an approximation of the Pareto-optimal solutions for learning the Pareto-optimal subspace. To enable RBM and DAE to learn the sparse distribution and compact representation, respectively, each solution **x** is represented by a binary vector **xb** and a real vector **xr**, and each decision variable of **x** is obtained by

$$x_i = xb_i \times xr_i \tag{5}$$

where xb_i indicates whether the ith decision variable is zero, and xr_i indicates the real value of the ith decision variable. If the problem is with binary variables, the real vector **xr** is always set to a vector of ones. In short, each solution is represented by **xb** and **xr** instead of **x**, where the length of all the three vectors is equal to the number of decision variables. Before generating offspring solutions, the binary vectors of all the nondominated solutions are used to train an RBM via the contrastive divergence algorithm, and the real vectors of all the nondominated solutions are used to train a DAE via gradient descent. As shown in Fig. 3, the real vector and binary vector of each solution can be reduced by the representation of the hidden layers of the two neural networks, and the reduced vectors can also be recovered to normal vectors. In other words, each solution can be mapped between the original search space and the Pareto-optimal subspace, where offspring solutions are generated in the Pareto-optimal subspace and evaluated by the objective functions in the original search space.

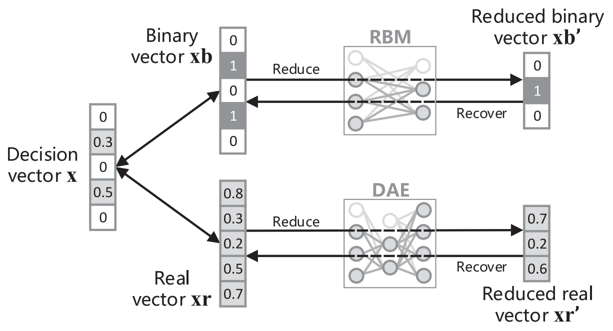

Fig. 3. Reduction and recover of solutions in MOEA/PSL. Note that the variables **xb** and **xr** are stored in each solution, while **x** is a temporary variable for calculating objective functions.

Afterward, two parents are randomly picked up from the mating pool each time, whose binary vectors are used to generate the binary vectors of two offspring solutions by single-point crossover and bitwise mutation, and the real vectors are used to generate the real vectors of two offspring solutions by simulated binary crossover [51] and polynomial mutation [52]. It is worth noting that a parameter ρ is used to determine whether each offspring solution is generated in the Pareto-optimal subspace or the original search space. Specifically, the parameter ρ is compared with a random value within $[0, 1]$. If ρ is larger than the random value, the binary vectors and real vectors of the parents are reduced by (1) and (3), respectively, and the offspring solutions are generated in the Pareto-optimal subspace and then

Algorithm 2: *Variation*(P, P', ρ, K)

```
Input: $P$ (current population), $P'$ (mating pool), $\rho$ (ratio of
       offspring solutions generated in the Pareto optimal
       subspace), $K$ (size of hidden layers)
Output: $O$ (offspring set)
//Model training
1  $NP \leftarrow$ All the non-dominated solutions in $P$;
2  $NPB \leftarrow$ Binary vectors of the solutions in $NP$;
3  $NPR \leftarrow$ Real vectors of the solutions in $NP$;
4  Train an RBM with $K$ hidden neurons based on $NPB$;
5  if the decision variables are real numbers then
6      Train a DAE with $K$ hidden neurons based on $NPR$;
//Offspring generation
7  $OB \leftarrow \emptyset$; //Binary vectors of offspring
     solutions
8  $OR \leftarrow \emptyset$; //Real vectors of offspring solutions
9  while $NPB \neq \emptyset$ do
10     $[\mathbf{xb}, \mathbf{yb}] \leftarrow$ Randomly select two vectors from $NPB$;
11     $NPB \leftarrow NPB \setminus \{\mathbf{xb}, \mathbf{yb}\}$;
12     if the decision variables are real numbers then
13         $[\mathbf{xr}, \mathbf{yr}] \leftarrow$ Select two vectors from $NPR$ that have the
           same locations as $\mathbf{xb}$ and $\mathbf{yb}$ in $NPB$;
14         $NPR \leftarrow NPR \setminus \{\mathbf{xr}, \mathbf{yr}\}$;
15     if $\rho > rand()$ then
16         $[\mathbf{xb}', \mathbf{yb}'] \leftarrow$ Reduce $\mathbf{xb}$ and $\mathbf{yb}$ by (1);
17         $[\mathbf{pb}', \mathbf{qb}'] \leftarrow$ Perform single-point crossover and
           bitwise mutation on $\mathbf{xb}'$ and $\mathbf{yb}'$;
18         $[\mathbf{pb}, \mathbf{qb}] \leftarrow$ Recover $\mathbf{pb}'$ and $\mathbf{qb}'$ by (2);
19         $OB \leftarrow OB \bigcup \{\mathbf{pb}, \mathbf{qb}\}$;
20         if the decision variables are real numbers then
21             $[\mathbf{xr}', \mathbf{yr}'] \leftarrow$ Reduce $\mathbf{xr}$ and $\mathbf{yr}$ by (3);
22             $[\mathbf{pr}', \mathbf{qr}'] \leftarrow$ Perform simulated binary crossover
               and polynomial mutation on $\mathbf{xr}'$ and $\mathbf{yr}'$;
23             $[\mathbf{pr}, \mathbf{qr}] \leftarrow$ Recover $\mathbf{pr}'$ and $\mathbf{qr}'$ by (4);
24             $OR \leftarrow OR \bigcup \{\mathbf{pr}, \mathbf{qr}\}$;
25     else
26         $[\mathbf{pb}, \mathbf{qb}] \leftarrow$ Perform single-point crossover and bitwise
           mutation on $\mathbf{xb}$ and $\mathbf{yb}$;
27         $OB \leftarrow OB \bigcup \{\mathbf{pb}, \mathbf{qb}\}$;
28         if the decision variables are real numbers then
29             $[\mathbf{pr}, \mathbf{qr}] \leftarrow$ Perform simulated binary crossover
               and polynomial mutation on $\mathbf{xr}$ and $\mathbf{yr}$;
30             $OR \leftarrow OR \bigcup \{\mathbf{pr}, \mathbf{qr}\}$;
31 $O \leftarrow$ Generate offspring solutions by (5) based on $OB$ and $OR$;
32 return $O$;
```

recovered by (2) and (4); otherwise, the offspring solutions are generated in the original search space without using RBM or DAE. Algorithm 2 summarizes the procedure of the offspring generation strategy.

C. Parameter Adaptation Strategy in MOEA/PSL

There are two parameters related to the offspring generation in MOEA/PSL, that is, the ratio of offspring solutions generated in the Pareto-optimal subspace ρ and the size of the hidden layers K. Intuitively, the parameter ρ should be dynamically adjusted to better balance exploration and exploitation, and the parameter K should decrease with the convergence of the population. To this end, a parameter adaptation strategy is designed to automatically determine the values of ρ and K at each generation. Specifically, the parameter ρ is iteratively updated by

Algorithm 3: *ParameterAdaptation*(P, ρ)

```
Input: $P$ (current population), $\rho$ (ratio of offspring solutions
       generated in the Pareto optimal subspace)
Output: $\rho$ (ratio of offspring solutions generated in the Pareto
        optimal subspace), $K$ (size of hidden layers)
//Update the parameter $\rho$
1  $s_{1,t} \leftarrow$ Number of successful offspring solutions generated in
   the Pareto optimal subspace at the current generation;
2  $s_{2,t} \leftarrow$ Number of successful offspring solutions generated in
   the original search space at the current generation;
3  Update $\rho$ by (6);
//Determine the parameter $K$
4  $NP \leftarrow$ All the non-dominated solutions in $P$;
5  Calculate $\mathbf{dec}$ by (7);
6  Calculate $K$ by (8);
7  return $\rho$ and $K$;
```

$$\rho_{t+1} = 0.5 \times \left(\rho_t + \frac{s_{1,t}}{s_{1,t} + s_{2,t}}\right) \tag{6}$$

where ρ_t denotes the value of ρ at the t-th generation and $\rho_0 = 0.5$, $s_{1,t}$ denotes the number of successful offspring solutions generated in the Pareto-optimal subspace at the t-th generation, and $s_{2,t}$ denotes the number of successful offspring solutions generated in the original search space at the t-th generation. A successful offspring solution means that it survives to the next generation, hence the ratio $[(s_{1,t})/(s_{1,t} + s_{2,t})]$ reflects the effectiveness of generating offspring solutions in the Pareto-optimal subspace. In short, the parameter ρ is updated according to the number of successful offspring solutions generated at the previous generations, where a higher ratio of successful offspring solutions generated in the Pareto-optimal subspace results in a larger ρ, and vice versa.

In contrast, the parameter K is determined according to the sparsity of the nondominated solutions in the current population. Let **dec** be a binary vector denoting whether each decision variable should be nonzero, the probability that dec_i is set to 1 is estimated according to the nondominated solution set NP

$$p(\text{dec}_i = 1|NP) = \frac{1}{|NP|} \sum_{\mathbf{x} \in NP} |\text{sign}(x_i)| \tag{7}$$

where $|\text{sign}(x_i)|$ is equal to 0 if $x_i = 0$ and 1 otherwise. Then, the value of dec_i is sampled by comparing the probability with a uniformly distributed random value within [0, 1]. Since the zero elements in **dec** (i.e., the sparse part of the decision variables) can be ignored, the size of the hidden layers is set to the number of nonzero elements in **dec**, that is

$$K = \sum \text{dec}_i. \tag{8}$$

The pseudocode of the parameter adaptation strategy in MOEA/PSL is given in Algorithm 3.

D. Computational Complexity of MOEA/PSL

Each generation of MOEA/PSL consists of four main steps, that is, Pareto-optimal subspace learning, offspring generation, environmental selection, and parameter adaptation. The time complexity of training neural networks in the Pareto-optimal subspace learning is $O(NEDK)$, where N, E, D, and K denote the population size, the number of epochs for training, the number of decision variables, and the hidden layer size, respectively. The time complexity of generating offspring solutions is $O(NDK)$. The environmental selection is the same as that of NSGA-II, which has a time complexity of $O(MN^2)$ [6], where M denotes the number of objectives. The time complexity of parameter adaptation is $O(ND)$. To summarize, the total computational complexity of MOEA/PSL in one generation is $O(MN^2 + NEDK)$.

IV. Empirical Studies

To empirically investigate the performance of MOEA/PSL, it is first compared with four representative MOEAs on eight benchmark problems with sparse Pareto-optimal solutions. Then, MOEA/PSL is tested on eight real-world problems from various fields. Afterward, the effectiveness of the two neural networks in the Pareto-optimal subspace learning is verified. Finally, the effectiveness of the parameter adaptation strategy in MOEA/PSL is verified. All the experiments are conducted on PlatEMO [53].

A. Comparative Algorithms

Four state-of-the-art MOEAs are selected as baselines in the experiments, namely, LMEA [8], WOF-SMPSO [11], MaOEA-IT [41], and SparseEA [42]. LMEA is a divide-and-conquer MOEA tailored for LMOPs, which divides the decision variables into convergence-related variables and diversity-related variables and optimizes them separately. WOF-SMPSO refers to the WOF-based speed-constrained multiobjective particle swarm optimization (PSO) algorithm, which is efficient for LMOPs due to the fast convergence speed of PSO and a problem transformation strategy. MaOEA-IT uses PCA to learn the Pareto-optimal subspace according to a set of well-converged solutions. Besides, SparseEA is currently the only MOEA considering the sparse nature of problems, but it encounters difficulties when dealing with a large number of decision variables due to the absence of Pareto-optimal subspace learning.

Population Size and the Number of Function Evaluations: The population size and the number of function evaluations of all the compared MOEAs are set to the same for fair comparisons. For benchmark problems, the population size is set to 100 and the number of function evaluations is set to $100 \times D$, where D denotes the number of decision variables. Due to the computationally expensive objectives of real-world problems, the population size is set to 50 as suggested in [42]; besides, the number of function evaluations is set to 2.0×10^4, 1.0×10^5, and 2.0×10^5 for problems with approximately 1000, 5000, and 10 000 real variables, and set to 1.0×10^4, 5.0×10^4, and 1.0×10^5 for problems with approximately 1000, 5000, and 10 000 binary variables.

Genetic Operators: In LMEA, MaOEA-IT, SparseEA, and MOEA/PSL, the single-point crossover and bitwise mutation are employed for solving problems with binary variables, and the simulated binary crossover [51] and polynomial mutation [52] are employed for solving problems with real variables; the probability of crossover is set to 1, the probability of mutation is set to $1/D$, and the distribution index of both crossover and mutation is set to 20. In WOF-SMPSO, the PSO operator [57] and the polynomial mutation are employed for solving all the problems; when handling binary variables, it optimizes the same number of real variables within [0, 1] and rounds the variables before calculating objective values.

Other Parameters: The other parameters in all the MOEAs are tuned for a relatively good performance. For LMEA, the number of selected solutions for variable clustering is set to 2, the number of perturbations on each solution for variable clustering is set to 4, and the number of selected solutions for variable interaction analysis is set to 5. For WOF-SMPSO, the number of groups is set to 4, the number of evaluations for the original problem is set to 1000, the number of evaluations for the transformed problem is set to 500, the number of chosen solutions for weight optimization is set to 3, and the fraction of evaluations for weight optimization is set to 0.5. For MaOEA-IT, the number of evaluations for dynamic weight aggregation is set to $50 \times D$, and the number of evaluations for reference lines mapping is set to D. For MOEA/PSL, the number of epochs for training neural networks is set to 10.

B. Test Problems

The sparse multiobjective test suite [42] is adopted to test the performance of the compared MOEAs, which contains eight benchmark problems SMOP1–SMOP8 with the scalable number of decision variables. These problems are characterized by multimodality, deception, epistasis, and low intrinsic dimensionality, posing various difficulties for MOEAs to obtain a set of sparse solutions. In the experiments, the number of objectives of these problems is set to 2, the number of decision variables is set to 1000, 5000, and 10 000, and the sparsity of Pareto-optimal solutions is set to 0.1.

Moreover, eight sparse LMOPs in real-world applications are established as test problems, including feature selection, instance selection, neural network training, community detection, critical node detection, sparse signal reconstruction, pattern mining, and portfolio optimization. The mathematical definitions of these applications can be found in Supplementary Material I. As shown in Table II, three datasets are used in each application, which result in three sparse LMOPs with approximately 1000, 5000, and 10 000 decision variables.

For each MOEA on each test problem, 30 independent runs are performed to obtain statistical results, where the IGD indicator [58] with 10 000 reference points is used to measure the results on benchmark problems. Since the Pareto fronts of real-world problems are unknown, the HV indicator [59] with a reference point $(1, 1)$ is used to measure the results on real-world problems. Besides, the Wilcoxon rank-sum test with a significance level of 0.05 is adopted to perform statistical analysis, where the symbols "+," "−," and "$\approx$" indicate that the result by another MOEA is significantly better, significantly worse, and statistically similar to that obtained by the proposed MOEA/PSL, respectively.

TABLE II
DATASETS OF EIGHT SPARSE LMOPS IN REAL-WORLD APPLICATIONS

Feature selection problem	Type of variables	No. of variables	Dataset	No. of samples	No. of features	No. of classes
FS1		800	Gse72526[1]	61	800	4
FS2	Binary	5966	Prostate GE[2]	102	5966	2
FS3		10000	Arcene[2]	200	10000	2
Instance selection problem	Type of variables	No. of variables	Dataset	No. of samples	No. of features	No. of classes
IS1		862	Fouclass[3]	862	3	2
IS2	Binary	4177	Abalone[4]	4177	9	2
IS3		11055	phishing[3]	11055	68	2
Neural network training problem	Type of variables	No. of variables	Dataset	No. of samples	No. of features	No. of classes
NN1		1181	Spambase[4]	4597	57	2
NN2	Real	5161	Semeion Handwritten Digit[4]	1593	256	10
NN3		10041	Madelon[4]	2600	500	2
Community detection problem	Type of variables	No. of variables	Dataset	No. of nodes	No. of edges	
CD1		1133	Email[5]	1133	5451	
CD2	Binary	4039	Facebook [54]	4039	88234	
CD3		9885	Duke14 [54]	9885	506437	
Critical node detection problem	Type of variables	No. of variables	Dataset	No. of nodes	No. of edges	
CN1		1176	Power Network [54]	1176	8688	
CN2	Binary	4039	Facebook [54]	4039	88234	
CN3		9885	Duke14 [54]	9885	506437	
Sparse signal reconstruction problem	Type of variables	No. of variables	Dataset	Length of signal	Length of received signal	Sparsity of signal
SR1		1024	Synthetic [55]	1024	480	260
SR2	Real	5120	Synthetic [55]	5120	2400	1300
SR3		10240	Synthetic [55]	10240	4800	2600
Pattern mining problem	Type of variables	No. of variables	Dataset	No. of transactions	No. of items	Avg. length of transactions
PM1		1000	Synthetic [56]	10000	1000	500
PM2	Binary	5000	Synthetic [56]	50000	5000	2500
PM3		10000	Synthetic [56]	100000	10000	5000
Portfolio optimization problem	Type of variables	No. of variables	Dataset	No. of instruments	Length of each instrument	
PO1		1000	EURCHF[6]	1000	50	
PO2	Real	5000	EURCHF[6]	5000	50	
PO3		10000	EURCHF[6]	10000	50	

1. https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi 2. http://featureselection.asu.edu/datasets.php
3. https://www.csie.ntu.edu.tw/%7ecjlin/libsvmtools/datasets/binary.html 4. https://archive.ics.uci.edu/ml
5. http://deim.urv.cat/%7ealexandre.arenas/data/welcome.htm 6. https://www.metatrader5.com/en

C. Results on Benchmark Problems

Table III lists the IGD values obtained by the five compared MOEAs on SMOP1–SMOP8 with 1000, 5000, and 10 000 decision variables. In general, the proposed MOEA/PSL performs the best on 19 out of 24 test instances, SparseEA performs the best on five test instances, while LMEA, WOF-SMPSO, and MaOEA-IT do not obtain any best result. As a consequence, the experimental results of MOEA/PSL and SparseEA verify the importance of considering sparsity in solving sparse problems, and the superiority of MOEA/PSL over SparseEA verifies the effectiveness of Pareto-optimal subspace learning in solving LMOPs.

For further observation, Fig. 4 plots the nondominated solution sets with median IGD values among 30 runs obtained by the five compared MOEAs on SMOP1, SMOP5, and SMOP8 with 10 000 decision variables. For SMOP1 with a mostly unimodal landscape, the proposed MOEA/PSL converges better than SparseEA and significantly outperforms LMEA, WOF-SMPSO, and MaOEA-IT. For SMOP8 with complex variable interactions, the difference between MOEA/PSL and SparseEA becomes larger. This is because SparseEA does not consider any variable interaction when generating offspring solutions, whereas the proposed MOEA/PSL can learn the variable interactions by RBM and DAE. As for SMOP5 with a unimodal landscape, both MOEA/PSL and SparseEA have satisfactory convergence performance, since the nonsparse region of the Pareto-optimal solutions of SMOP5 is unfixed and quite easy to be detected. Moreover, according to the convergence profiles of IGD values obtained by the five MOEAs shown in Fig. 5, it is obvious that MOEA/PSL converges faster than the other MOEAs during the entire evolutionary process. To summarize, the proposed MOEA/PSL can effectively solve sparse LMOPs with various landscapes.

D. Results on Real-World Problems

Table IV shows the HV values obtained by the five compared MOEAs on eight sparse LMOPs in the real-world applications with approximately 1000, 5000, and 10 000 decision variables. It can be seen that the proposed MOEA/PSL obtains the best overall performance on these real-world problems, having achieved the best performance on 19 out of 24 test instances. Besides, SparseEA and WOF-SMOPSO obtain 4 and 1 best results, respectively. Furthermore, Fig. 6 depicts the nondominated solution sets with median HV values among 30 runs obtained by the five compared MOEAs on FS3, NN3, and PO3 with approximately 10 000 decision variables. For the feature selection problem, the solutions obtained by MOEA/PSL are slightly better than those obtained by SparseEA and WOF-SMPSO, while LMEA and

TABLE III
IGD VALUES OBTAINED BY LMEA, WOF-SMPSO, MAOEA-IT, SPARSEEA, AND THE PROPOSED MOEA/PSL ON SMOP1–SMOP8, WHERE THE BEST RESULT IN EACH ROW IS HIGHLIGHTED

Problem	Dec	LMEA	WOF-SMPSO	MaOEA-IT	SparseEA	MOEA/PSL
SMOP1	1000	8.2758e-1 (8.25e-3) −	2.3787e-1 (2.38e-2) −	4.7235e-1 (7.75e-2) −	2.8467e-2 (2.12e-17) −	**1.1507e-2 (2.11e-3)**
	5000	8.6219e-1 (3.48e-3) −	1.7494e-1 (2.52e-2) −	7.5928e-1 (3.41e-2) −	3.8535e-2 (1.25e-3) −	**1.3438e-2 (2.33e-3)**
	10000	8.7141e-1 (1.98e-3) −	1.4972e-1 (1.56e-2) −	9.3680e-1 (5.54e-2) −	4.0795e-2 (7.77e-4) −	**1.4308e-2 (4.31e-3)**
SMOP2	1000	1.7590e+0 (7.28e-3) −	3.2865e-1 (1.46e-1) −	1.1721e+0 (5.36e-2) −	6.4140e-2 (4.23e-17) −	**4.2070e-2 (9.70e-3)**
	5000	1.7838e+0 (3.03e-3) −	1.8621e-1 (2.67e-2) −	1.6124e+0 (3.05e-2) −	9.9443e-2 (2.23e-3) −	**4.7483e-2 (7.87e-3)**
	10000	1.7912e+0 (1.56e-3) −	1.8004e-1 (1.13e-2) −	1.7619e+0 (3.55e-2) −	1.0450e-1 (3.13e-3) −	**5.5203e-2 (3.48e-3)**
SMOP3	1000	2.1679e+0 (1.02e-2) −	7.0334e-1 (2.76e-3) −	1.5729e+0 (5.55e-2) −	3.2951e-2 (2.12e-17) −	**1.2598e-2 (3.06e-3)**
	5000	2.1960e+0 (4.31e-3) −	7.0172e-1 (1.12e-3) −	2.0555e+0 (3.81e-2) −	4.2581e-2 (1.40e-3) −	**1.2200e-2 (8.68e-4)**
	10000	2.2020e+0 (2.46e-3) −	7.0123e-1 (6.11e-4) −	2.1973e+0 (1.82e-2) −	4.5652e-2 (6.62e-4) −	**1.5615e-2 (7.36e-4)**
SMOP4	1000	8.8105e-1 (3.87e-3) −	6.0319e-2 (7.03e-2) −	5.9578e-1 (5.38e-2) −	**4.5184e-3 (8.82e-19) +**	4.7175e-3 (2.47e-4)
	5000	8.9139e-1 (1.81e-3) −	9.4086e-3 (4.94e-3) −	8.0334e-1 (1.80e-2) −	4.7526e-3 (2.36e-4) ≈	**4.7211e-3 (1.14e-4)**
	10000	8.9378e-1 (1.75e-3) −	1.0315e-2 (1.04e-2) −	8.6391e-1 (2.41e-2) −	5.0535e-3 (3.72e-4) −	**4.5711e-3 (1.11e-4)**
SMOP5	1000	6.0578e-1 (5.70e-3) −	3.5623e-1 (1.80e-3) −	4.4576e-1 (1.72e-2) −	**5.8769e-3 (2.84e-4) +**	7.4279e-3 (4.60e-4)
	5000	6.2194e-1 (3.01e-3) −	3.4946e-1 (5.59e-4) −	5.4720e-1 (1.43e-2) −	**5.8794e-3 (2.35e-4) +**	6.8328e-3 (2.47e-4)
	10000	6.2439e-1 (1.40e-3) −	3.4855e-1 (2.53e-4) −	6.2980e-1 (1.63e-2) −	**6.0049e-3 (1.39e-4) +**	6.6151e-3 (1.72e-4)
SMOP6	1000	2.4791e-1 (2.41e-3) −	8.0357e-2 (1.53e-3) −	1.6209e-1 (1.04e-2) −	**7.0559e-3 (3.22e-4) +**	7.4349e-3 (5.34e-4)
	5000	2.5654e-1 (1.49e-3) −	3.8829e-2 (1.99e-2) −	2.4433e-1 (8.07e-3) −	7.5168e-3 (2.35e-4) −	**6.9494e-3 (3.12e-4)**
	10000	2.5786e-1 (7.80e-4) −	2.0055e-2 (1.07e-2) −	2.9218e-1 (8.71e-3) −	7.5046e-3 (9.62e-5) −	**6.8096e-3 (3.51e-4)**
SMOP7	1000	1.4609e+0 (1.58e-2) −	9.6058e-2 (8.53e-3) −	8.7400e-1 (8.53e-2) −	8.8565e-2 (7.96e-3) −	**5.5240e-2 (3.62e-2)**
	5000	1.5238e+0 (8.84e-3) −	8.3179e-2 (4.20e-3) −	1.2876e+0 (4.95e-2) −	1.2660e-1 (4.33e-3) −	**4.9372e-2 (8.25e-3)**
	10000	1.5406e+0 (4.11e-3) −	7.6538e-2 (5.80e-3) −	1.5120e+0 (9.26e-2) −	1.4665e-1 (2.94e-2) −	**5.0672e-2 (7.09e-3)**
SMOP8	1000	3.2484e+0 (1.31e-2) −	5.6267e+0 (2.23e-2) −	2.4728e+0 (6.67e-2) −	2.4697e-1 (1.49e-2) −	**2.2823e-1 (4.59e-2)**
	5000	3.2995e+0 (4.10e-3) −	5.3577e-1 (4.01e-3) −	2.9442e+0 (5.61e-2) −	3.2577e-1 (9.22e-3) −	**2.5555e-1 (3.72e-2)**
	10000	3.3093e+0 (3.72e-3) −	5.3185e-1 (3.71e-3) −	3.1224e+0 (5.42e-2) −	3.4426e-1 (4.90e-3) −	**2.5127e-1 (7.19e-3)**
+/ − / ≈		0/24/0	0/24/0	0/24/0	5/18/1	

'+', '−' and '≈' indicate that the result is significantly better, significantly worse, and statistically similar to that obtained by the proposed MOEA/PSL, respectively.

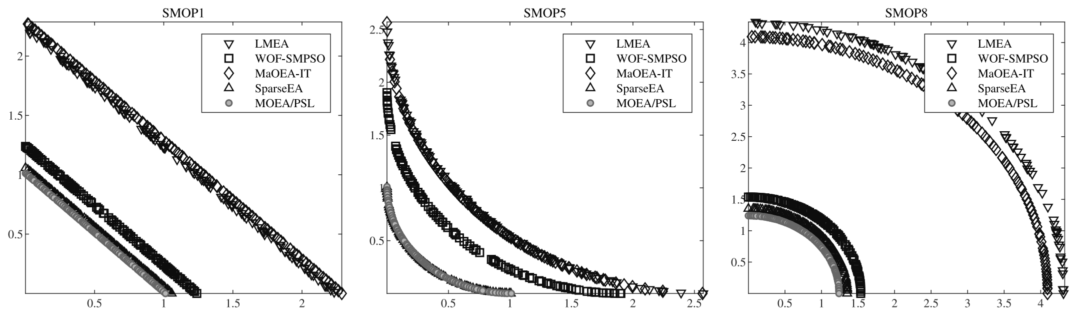

Fig. 4. Nondominated solution sets with median IGD obtained by LMEA, WOF-SMPSO, MaOEA-IT, SparseEA, and the proposed MOEA/PSL on SMOP1, SMOP5, and SMOP8 with 10 000 decision variables.

MaOEA-IT can only find a single solution with bad convergence. For the neural network training problem, the solutions obtained by MOEA/PSL have good convergence and diversity, the solutions obtained by LMEA and WOF-SMPSO are not well converged, and MaOEA-IT and SparseEA can only find a single solution. As for the portfolio optimization problem, the solutions obtained by MOEA/PSL have significantly better convergence than those obtained by SparseEA, and the solutions obtained by LMEA, WOF-SMPSO, and MaOEA-IT shrink to the upper left corner of the objective space. As a consequence, these real-world problems are quite challenging for existing MOEAs, while the proposed MOEA/PSL is more promising for solving these problems than existing MOEAs.

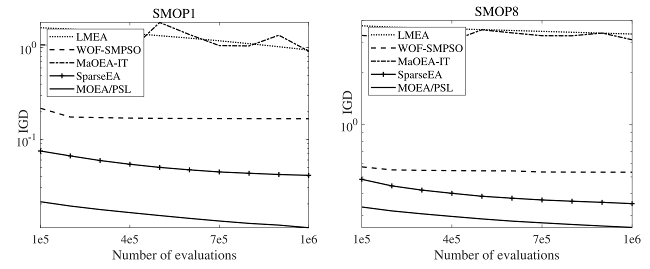

Fig. 5. Convergence profiles of IGD values obtained by LMEA, WOF-SMPSO, MaOEA-IT, SparseEA, and the proposed MOEA/PSL on SMOP1 and SMOP8 with 10 000 decision variables.

In addition, the runtimes consumed by the five compared MOEAs on the eight benchmark problems and eight real-world problems with approximately 10 000 decision variables are listed in Table V. It can be found that the runtime of MOEA/PSL is not significantly higher than the other MOEAs, though MOEA/PSL should train two neural networks at each generation. This is because the other MOEAs also have complex operations (e.g., variable interaction analysis in LMEA, Pareto-optimal subspace

TABLE IV
HV VALUES OBTAINED BY LMEA, WOF-SMPSO, MAOEA-IT, SPARSEEA, AND THE PROPOSED MOEA/PSL ON FS1–FS3, IS1–IS3, NN1–NN3, CD1–CD3, CN1–CN3, SR1–SR3, PM1–PM3, AND PO1–PO3, WHERE THE BEST RESULT IN EACH ROW IS HIGHLIGHTED

Problem	Dec	LMEA	WOF-SMPSO	MaOEA-IT	SparseEA	MOEA/PSL
FS1	800	2.6893e-1 (0.00e+0) −	8.8087e-1 (5.65e-16) −	2.8697e-1 (0.00e+0) −	9.8467e-1 (4.52e-16) −	9.8478e-1 (3.39e-16)
FS2	5966	4.8819e-1 (2.26e-16) −	9.5529e-1 (0.00e+0) −	6.2641e-1 (3.39e-16) −	9.8215e-1 (4.52e-16) −	9.8719e-1 (5.04e-3)
FS3	10000	3.3145e-1 (5.65e-17) −	8.9075e-1 (4.52e-16) −	3.7800e-1 (1.13e-16) −	9.7262e-1 (7.90e-16) ≈	9.7443e-1 (7.60e-3)
IS1	862	5.9388e-1 (1.38e-2) −	8.7186e-1 (4.59e-3) −	5.7149e-1 (1.85e-2) −	8.4077e-1 (1.07e-2) −	8.9922e-1 (1.78e-2)
IS2	4177	4.6320e-1 (1.59e-2) −	7.4703e-1 (3.19e-3) −	4.7467e-1 (7.97e-3) −	7.0593e-1 (9.11e-3) −	8.0424e-1 (2.42e-2)
IS3	11055	5.3065e-1 (3.04e-3) −	9.6268e-1 (1.35e-3) ≈	6.7505e-1 (8.71e-3) −	9.5401e-1 (7.10e-4) −	9.6778e-1 (7.31e-3)
NN1	1181	3.3298e-1 (1.53e-2) −	4.7043e-1 (1.31e-2) −	3.1841e-1 (9.69e-2) −	9.0530e-1 (1.73e-2) −	9.2951e-1 (3.99e-3)
NN2	5161	2.9111e-1 (1.36e-2) −	6.7505e-1 (7.18e-2) −	3.3409e-1 (1.01e-1) −	9.1165e-1 (3.07e-2) −	9.6078e-1 (2.41e-2)
NN3	10041	2.6363e-1 (1.49e-2) −	3.4689e-1 (2.69e-2) −	2.6016e-1 (5.04e-2) −	5.8179e-1 (4.53e-2) −	6.6586e-1 (1.48e-2)
CD1	1133	9.9357e-2 (2.10e-2) −	6.9976e-1 (1.70e-3) −	3.7719e-1 (8.57e-3) −	5.0006e-1 (0.00e+0) −	7.0540e-1 (1.13e-16)
CD2	4039	8.7164e-2 (1.20e-2) −	6.7661e-1 (1.65e-3) −	4.0942e-1 (4.68e-3) −	5.4546e-1 (3.42e-3) −	6.9660e-1 (3.55e-3)
CD3	9885	3.1492e-1 (1.28e-3) ≈	5.8850e-1 (2.02e-3) ≈	3.7066e-1 (2.00e-3) −	5.3839e-1 (1.07e-3) −	5.8784e-1 (8.28e-4)
CN1	1176	7.0620e-1 (1.77e-2) −	8.6239e-1 (8.82e-3) −	5.9259e-1 (1.40e-2) −	9.4455e-1 (2.72e-3) −	9.7206e-1 (2.12e-3)
CN2	4039	5.6216e-1 (3.26e-2) −	7.9818e-1 (9.42e-3) −	5.5264e-1 (1.73e-2) −	8.6601e-1 (1.22e-2) −	9.2460e-1 (5.36e-3)
CN3	9885	4.4994e-1 (2.57e-3) −	7.1083e-1 (2.28e-3) −	2.5489e-1 (6.69e-2) −	7.1141e-1 (1.28e-3) −	7.2132e-1 (4.28e-4)
SR1	1024	1.5091e-1 (1.23e-2) −	1.1669e-1 (6.35e-3) −	1.0888e-1 (6.59e-3) −	2.6363e-1 (1.62e-2) −	3.2374e-1 (1.91e-2)
SR2	5120	1.2704e-1 (7.51e-3) −	9.8539e-2 (2.16e-3) −	9.4736e-2 (2.44e-3) −	2.0594e-1 (6.87e-3) −	2.8826e-1 (6.91e-3)
SR3	10240	1.2392e-1 (6.12e-3) −	9.5335e-2 (1.73e-3) −	9.4564e-2 (2.88e-3) −	2.0541e-1 (8.50e-3) −	2.7922e-1 (6.67e-3)
PM1	1000	8.2645e-3 (3.53e-18) −	1.0368e-1 (2.01e-3) −	1.4405e-2 (1.08e-2) −	1.5721e-1 (1.71e-3) +	1.3620e-1 (2.52e-3)
PM2	5000	8.2645e-3 (3.53e-18) −	9.4358e-2 (7.33e-4) +	1.2648e-2 (3.11e-3) −	1.1577e-1 (8.65e-4) +	8.5430e-2 (1.95e-2)
PM3	10000	8.2645e-3 (1.81e-18) −	9.2739e-2 (3.67e-4) −	9.2836e-3 (1.17e-3) −	1.1023e-1 (5.13e-4) +	9.5855e-2 (3.18e-4)
PO1	1000	9.1432e-2 (9.57e-5) −	9.2197e-2 (2.90e-4) −	9.1671e-2 (7.46e-5) −	1.2368e-1 (1.63e-3) ≈	1.2367e-1 (5.84e-4)
PO2	5000	9.1120e-2 (2.82e-5) −	9.1244e-2 (5.42e-5) −	9.1120e-2 (1.53e-5) −	1.2380e-1 (1.78e-3) ≈	1.2412e-1 (9.30e-4)
PO3	10000	9.1082e-2 (3.97e-5) −	9.1097e-2 (3.77e-5) −	9.1041e-2 (1.06e-5) −	1.2429e-1 (1.48e-3) ≈	1.2481e-1 (2.42e-4)
+/−/≈		0/23/1	1/21/2	0/24/0	3/17/4	

'+', '−' and '≈' indicate that the result is significantly better, significantly worse, and statistically similar to that obtained by the proposed MOEA/PSL, respectively.

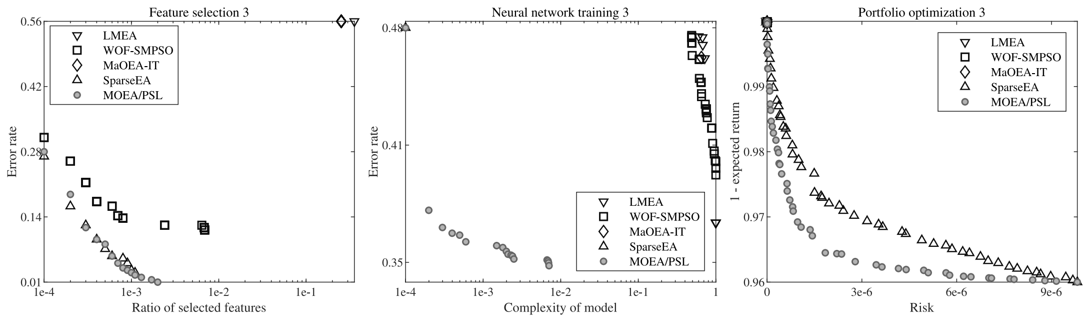

Fig. 6. Nondominated solution sets with median HV obtained by LMEA, WOF-SMPSO, MaOEA-IT, SparseEA, and the proposed MOEA/PSL on FS3, NN3, and PO3 with approximately 10 000 decision variables.

TABLE V
RUNTIMES (IN SECOND) OF LMEA, WOF-SMPSO, MAOEA-IT, SPARSEEA, AND THE PROPOSED MOEA/PSL ON BENCHMARK PROBLEMS AND REAL-WORLD PROBLEMS

Problem	Dec	LMEA	WOF-SMPSO	MaOEA-IT	SparseEA	MOEA/PSL
SMOP1–SMOP8 (average)	10000	1.1042e+4	5.6467e+3	8.2906e+3	1.0391e+4	6.8868e+3
FS3	10000	2.4979e+4	2.8925e+3	6.1546e+3	5.4577e+2	4.3347e+2
IS3	11055	2.1538e+5	5.7815e+5	1.3433e+5	4.2568e+4	2.2611e+5
NN3	10041	4.3060e+4	9.3179e+3	4.4536e+3	4.8048e+3	3.5246e+3
CD3	9885	1.2685e+5	4.1383e+4	1.3893e+5	7.5853e+4	6.1153e+4
CN3	9885	5.7187e+4	2.7179e+4	1.2290e+4	3.5482e+4	2.6756e+4
SR3	10240	3.7550e+4	9.2712e+3	7.5098e+3	5.2169e+3	2.6068e+3
PM3	10000	6.4473e+4	9.7101e+4	6.3456e+3	1.5784e+5	1.1052e+5
PO3	10000	8.1801e+3	1.5218e+3	1.9847e+3	2.2545e+3	2.2262e+3

learning in MaOEA-IT, and population initialization in SparseEA), while the neural networks in MOEA/PSL are relatively small and easy to be trained. In short, the proposed MOEA/PSL has competitive efficiency to the other MOEAs.

E. Effectiveness of the Two Neural Networks in MOEA/PSL

The proposed MOEA/PSL learns the Pareto-optimal subspace by both RBM and DAE, while a single neural network can also be used to achieve this goal. In this case, each solution does not need to be represented by a binary vector and a real vector, and the decision variables can directly be reduced. To verify the effectiveness of using both RBM and DAE, the proposed MOEA/PSL is compared with its variants using a single technique, including random embedding [60], PCA [61], RBM, and DAE. It is worth noting that although there exist many other machine-learning techniques able to learn a subspace, the vectors reduced by most of them are not recoverable, hence they cannot be used in MOEA/PSL.

TABLE VI
IGD VALUES OBTAINED BY MOEA/PSL AND MOEA/PSL WITH A SINGLE TECHNIQUE ON SMOP1–SMOP8, WHERE THE BEST RESULT IN EACH ROW IS HIGHLIGHTED

Problem	MOEA/PSL (only random embedding)	MOEA/PSL (only PCA)	MOEA/PSL (only RBM)	MOEA/PSL (only DAE)	MOEA/PSL
SMOP1	9.5389e-2 (5.56e-3) −	3.1069e-1 (5.95e-3) −	1.3810e-1 (1.32e-2) −	3.4657e-1 (8.12e-3) −	1.1507e-2 (2.11e-3)
SMOP2	3.6321e-1 (3.41e-2) −	9.5175e-1 (2.24e-2) −	1.5020e-1 (4.21e-2) −	9.7563e-1 (2.62e-2) −	4.2070e-2 (9.70e-3)
SMOP3	1.1895e+0 (3.61e-2) −	1.1837e+0 (3.90e-2) −	2.2785e+0 (2.05e-2) −	1.2607e+0 (2.87e-2) −	1.2598e-2 (3.06e-3)
SMOP4	6.1402e-2 (1.01e-2) −	4.2656e-1 (1.59e-2) −	5.9898e-2 (6.25e-3) −	4.3297e-1 (1.29e-2) −	4.7175e-3 (2.47e-4)
SMOP5	3.9132e-1 (3.36e-3) −	4.0154e-1 (8.31e-4) −	1.2366e-1 (1.77e-2) −	4.1233e-1 (1.30e-3) −	7.4279e-3 (4.60e-4)
SMOP6	6.5513e-2 (3.66e-3) −	9.0714e-2 (3.72e-3) −	7.3576e-2 (5.39e-3) −	9.0152e-2 (3.29e-3) −	7.4349e-3 (5.34e-4)
SMOP7	2.4670e-1 (2.04e-2) −	2.1980e-1 (1.02e-2) −	3.5022e-1 (6.44e-2) −	2.2567e-1 (5.81e-3) −	5.5240e-2 (3.62e-2)
SMOP8	1.1996e+0 (8.21e-2) −	1.2495e+0 (4.09e-2) −	3.7650e-1 (9.77e-3) −	1.2564e+0 (1.95e-2) −	2.2823e-1 (4.59e-2)
$+/-/\approx$	0/8/0	0/8/0	0/8/0	0/8/0	

'−' indicates that the result is significantly worse than that obtained by MOEA/PSL.

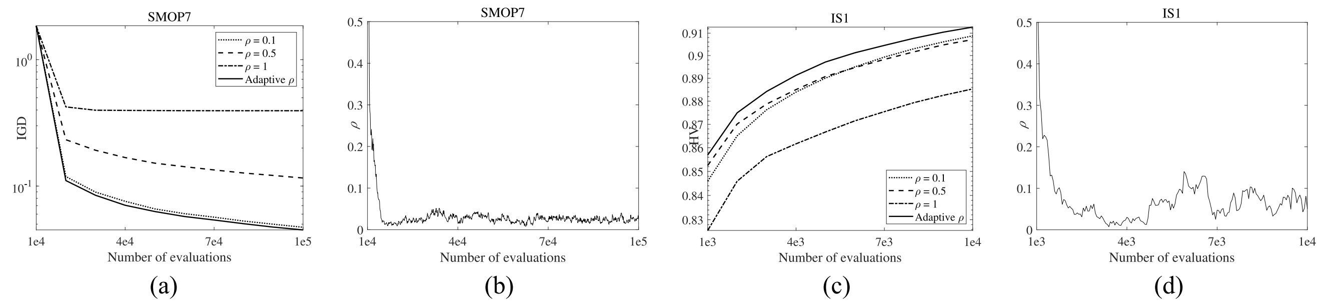

Fig. 7. Convergence profiles of IGD and HV values obtained by MOEA/PSL with adaptive and fixed ρ on SMOP7 and IS1. (a) IGD values on SMOP7. (b) Variation of ρ on SMOP7. (c) HV values on IS1. (d) Variation of ρ on IS1.

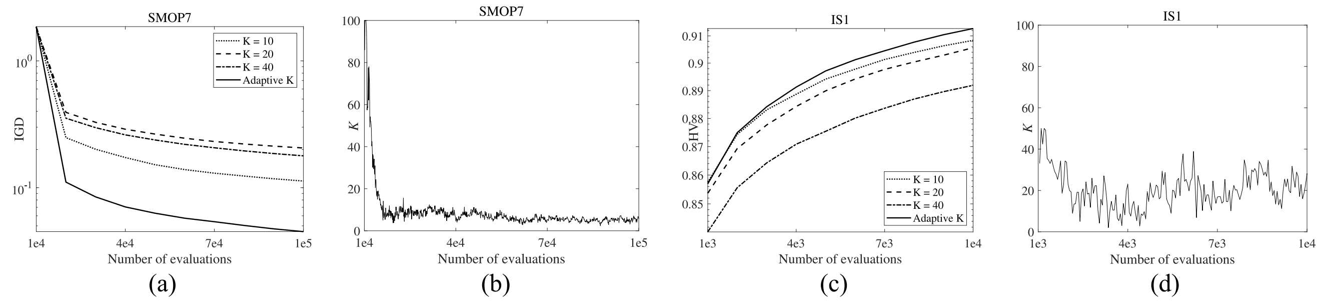

Fig. 8. Convergence profiles of IGD and HV values obtained by MOEA/PSL with adaptive and fixed K on SMOP7 and IS1. (a) IGD values on SMOP7. (b) Variation of K on SMOP7. (c) HV values on IS1. (d) Variation of K on IS1.

Table VI lists the IGD values obtained by MOEA/PSL and its four variants on SMOP1–SMOP8 with 1000 decision variables. Obviously, the original MOEA/PSL exhibits significantly better performance than those with a single technique on all the test problems. The superiority of using both RBM and DAE is mainly attributed to the consideration of sparsity, where the RBM is used to learn a sparsity distribution and the DAE is used to learn a compact representation. In short, it is necessary to learn the Pareto-optimal subspace and consider the sparsity simultaneously in solving sparse LMOPs.

F. Effectiveness of the Parameter Adaptation Strategy in MOEA/PSL

There are two parameters related to the employment of RBM and DAE in MOEA/PSL, that is, the ratio of offspring solutions generated in the Pareto-optimal subspace ρ and the size of the hidden layers K, both of which are adapted during the evolution of MOEA/PSL. To verify the effectiveness of the parameter adaptation strategy, MOEA/PSL is compared with some of its variants using a fixed ρ or K.

Fig. 7 plots the convergence profiles of IGD and HV values obtained by MOEA/PSL with adaptive and fixed ρ on SMOP7 with 1000 real variables and the instance selection problem with 862 binary variables. As shown in the figures, the MOEA/PSL with adaptive ρ converges faster than the MOEA/PSL with $\rho = 0.1$, $\rho = 0.5$, and $\rho = 1$. Besides, MOEA/PSL assigns different values to ρ at different generations on different problems. Therefore, it is unreasonable to set ρ to a fixed value; in contrast, adapting ρ during the evolution of MOEA/PSL can lead to a relatively good performance on different problems. Fig. 8 plots the convergence profiles of IGD and HV values obtained by MOEA/PSL with adaptive and fixed K on SMOP7 and the instance selection problem.

Similarly, the MOEA/PSL with adaptive K outperforms the MOEA/PSL with $K = 10$, $K = 20$, and $K = 40$. To summarize, the parameter adaptation strategy can not only make MOEA/PSL parameterless but also improve the performance in solving sparse LMOPs.

V. Conclusion

To address the limitations of existing MOEAs in solving LMOPs with sparse Pareto-optimal solutions, this article has proposed a Pareto-optimal subspace learning-based MOEA by using RBM and DAE. In each generation, an RBM is trained to learn a sparse distribution of the decision variables, and a DAE is trained to learn a compact representation. In this way, the decision variables can be reduced by the representation of the hidden layers of RBM and DAE. Moreover, a parameter adaptation strategy has been developed to determine the ratio of offspring solutions generated in the Pareto-optimal subspace and the size of the hidden layers.

In the experiments, the proposed MOEA has been compared with four state-of-the-art MOEAs on eight benchmark problems and eight real-world problems taken from various fields. The experimental results have demonstrated that the proposed MOEA is more effective than the compared MOEAs in solving sparse LMOPs with 1000–10 000 decision variables.

This article has revealed the necessity of Pareto-optimal subspace learning in solving sparse LMOPs. Although the proposed MOEA has successfully achieved this goal by using RBM and DAE, it may become impractical when dealing with the sparse problems having millions of decision variables (e.g., deep neural network training [62]), since the population is not large enough for training the neural networks, and the training process will be very time consuming. Therefore, it is desirable to incorporate the proposed Pareto-optimal subspace learning approach into other more effective MOEAs to solve super-large-scale problems.

References

[1] J. Liu, M. Gong, Q. Miao, X. Wang, and H. Li, "Structure learning for deep neural networks based on multiobjective optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 6, pp. 2450–2463, Jun. 2018.

[2] C. Cao, C. Li, Q. Yang, Y. Liu, and T. Qu, "A novel multi-objective programming model of relief distribution for sustainable disaster supply chain in large-scale natural disasters," *J. Clean. Prod.*, vol. 174, pp. 1422–1435, Feb. 2018.

[3] S. Mouassa and T. Bouktir, "Multi-objective ant lion optimization algorithm to solve large-scale multi-objective optimal reactive power dispatch problem," *Int. J. Comput. Math. Elect. Electron. Eng.*, vol. 38, no. 1, pp. 304–324, 2019.

[4] L. M. Antonio and C. A. Coello Coello, "Use of cooperative coevolution for solving large scale multiobjective optimization problems," in *Proc. IEEE Congr. Evol. Comput.*, 2013, pp. 2758–2765.

[5] A. W. Iorio and X. Li, "A cooperative coevolutionary multiobjective algorithm using non-dominated sorting," in *Proc. Genet. Evol. Comput. Conf.*, 2004, pp. 537–548.

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

[7] X. Ma *et al.*, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 275–298, Apr. 2016.

[8] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 22, no. 1, pp. 97–112, Feb. 2018.

[9] H. Xu, W. Zeng, D. Zhang, and X. Zeng, "MOEA/HD: A multiobjective evolutionary algorithm based on hierarchical decomposition," *IEEE Trans. Cybern.*, vol. 49, no. 2, pp. 517–526, Feb. 2019.

[10] H. Xu, W. Zeng, X. Zeng, and G. G. Yen, "An evolutionary algorithm based on Minkowski distance for many-objective optimization," *IEEE Trans. Cybern.*, vol. 49, no. 11, pp. 3968–3979, Nov. 2019.

[11] H. Zille, H. Ishibuchi, S. Mostaghim, and Y. Nojima, "A framework for large-scale multi-objective optimization based on problem transformation," *IEEE Trans. Evol. Comput.*, vol. 22, no. 2, pp. 260–275, Apr. 2018.

[12] C. He *et al.*, "Accelerating large-scale multi-objective optimization via problem reformulation," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 949–961, Dec. 2019.

[13] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multi-objective optimization based on a competitive swarm optimizer," *IEEE Trans. Cybern.*, early access, doi: 10.1109/TCYB.2019.2906383.

[14] M. Dellnitz, O. Schütze, and T. Hestermeyer, "Covering Pareto sets by multilevel subdivision techniques," *J. Optim. Theory Appl.*, vol. 124, no. 1, pp. 113–136, 2005.

[15] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Trans. Evol. Comput.*, vol. 12, no. 1, pp. 41–63, Feb. 2008.

[16] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff, "A multiobjective evolutionary algorithm using Gaussian process based inverse modeling," *IEEE Trans. Evol. Comput.*, vol. 19, no. 6, pp. 838–856, Dec. 2015.

[17] T. M. Hamdani, J.-M. Won, A. M. Alimi, and F. Karray, "Multi-objective feature selection with NSGA-II," in *Proc. Int. Conf. Adapt. Nat. Comput. Algorithms*, 2007, pp. 240–247.

[18] H. Li, Q. Zhang, J. Deng, and Z.-B. Xu, "A preference-based multiobjective evolutionary approach for sparse optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 29, no. 5, pp. 1716–1731, May 2018.

[19] Y. Tian, S. Yang, L. Zhang, F. Duan, and X. Zhang, "A surrogate-assisted multi-objective evolutionary algorithm for large-scale task-oriented pattern mining," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 2, pp. 106–116, Apr. 2019.

[20] A. Fischer and C. Igel, "An introduction to restricted Boltzmann machines," in *Proc. Iberoamerican Congr. Pattern Recognit.*, 2012, pp. 14–36.

[21] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. 25th Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[22] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.

[23] B. Tran, B. Xue, and M. Zhang, "A new representation in PSO for discretization-based feature selection," *IEEE Trans. Cybern.*, vol. 48, no. 6, pp. 1733–1746, Jun. 2018.

[24] N. Verbiest, J. Derrac, C. Cornelis, S. García, and F. Herrera, "Evolutionary wrapper approaches for training set selection as preprocessing mechanism for support vector machines: Experimental evaluation and support vector analysis," *Appl. Soft Comput.*, vol. 38, pp. 10–22, Jan. 2016.

[25] Z. Lu *et al.*, "NSGA-NET: A multi-objective genetic algorithm for neural architecture search," 2018. [Online]. Available: arXiv:1810.03522.

[26] X. Liu, Y. Luo, X. Zhang, and Q. Zhu, "A black-box attack on neural networks based on swarm evolutionary algorithm," 2019. [Online]. Available: arXiv:1901.09892.

[27] C. Qian, Y. Yu, and Z.-H. Zhou, "Pareto ensemble pruning," in *Proc. 29th Conf. Artif. Intell. (AAAI)*, 2015, pp. 463–484.

[28] Y. Tian, S. Yang, and X. Zhang, "An evolutionary multiobjective optimization based fuzzy method for overlapping community detection," *IEEE Trans. Fuzzy Syst.*, early access, doi: 10.1109/TFUZZ.2019.2945241.

[29] M. Lalou, M. A. Tahraoui, and H. Kheddouci, "The critical node detection problem in networks: A survey," *Comput. Sci. Rev.*, vol. 28, pp. 92–117, May 2018.

[30] J. Yang and J. Liu, "Influence maximization-cost minimization in social networks based on a multiobjective discrete particle swarm optimization algorithm," *IEEE Access*, vol. 6, pp. 2320–2329, 2018.

[31] Y. Xiang, Y. Zhou, Z. Zheng, and M. Li, "Configuring software product lines by combining many-objective optimization and SAT solvers," *ACM Trans. Softw. Eng. Methodol.*, vol. 26, no. 4, p. 14, 2018.

[32] X. Zhang, F. Duan, L. Zhang, F. Cheng, Y. Jin, and K. Tang, "Pattern recommendation in task-oriented applications: A multi-objective perspective [application notes]," *IEEE Comput. Intell. Mag.*, vol. 12, no. 3, pp. 43–53, Aug. 2017.

[33] K. Lwin, R. Qu, and G. Kendall, "A learning-guided multi-objective evolutionary algorithm for constrained portfolio optimization," *Appl. Soft Comput.*, vol. 24, pp. 757–772, Nov. 2014.

[34] A. Rahmani and S. MirHassani, "A hybrid firefly-genetic algorithm for the capacitated facility location problem," *Inf. Sci.*, vol. 283, pp. 70–78, Nov. 2014.

[35] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 257–271, Nov. 1999.

[36] J. Zhao, Y. Xu, F. Luo, Z. Dong, and Y. Peng, "Power system fault diagnosis based on history driven differential evolution and stochastic time domain simulation," *Inf. Sci.*, vol. 275, pp. 13–29, Aug. 2014.

[37] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 38, no. 3, pp. 397–415, May 2008.

[38] L. Zhang, H. Pan, Y. Su, X. Zhang, and Y. Niu, "A mixed representation-based multiobjective evolutionary algorithm for overlapping community detection," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2703–2716, Sep. 2017.

[39] S. Wang and D. Zhao, "Power grid fault diagnosis based on immune clonal constrained multi-objective optimization method," *Power Syst. Technol.*, vol. 41, no. 12, pp. 4061–4068, 2017.

[40] H. Qian and Y. Yu, "Solving high-dimensional multi-objective optimization problems with low effective dimensions," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 875–881.

[41] Y. Sun, B. Xue, M. Zhang, and G. G. Yen, "A new two-stage evolutionary algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 748–761, Oct. 2019.

[42] Y. Tian, X. Zhang, C. Wang, and Y. Jin, "An evolutionary algorithm for large-scale sparse multi-objective optimization problems," *IEEE Trans. Evol. Comput.*, early access, doi: 10.1109/TEVC.2019.2918140.

[43] L. V. D. Maaten and E. Postma, "Dimensionality reduction: A comparative review," Tilburg Centre Creative Comput., Tilburg Univ., Tilburg, The Netherlands, Rep. TR 2009-005, 2009.

[44] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.

[45] J. Zhai, S. Zhang, J. Chen, and Q. He, "Autoencoder and its various variants," in *Proc. IEEE Int. Conf. Syst. Man Cybern.*, 2018, pp. 415–419.

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

[47] H. Tang, V. A. Shim, K. C. Tan, and J. Y. Chia, "Restricted Boltzmann machine based algorithm for multi-objective optimization," in *Proc. IEEE Congr. Evol. Comput.*, 2010, pp. 1–8.

[48] L. Feng *et al.*, "Evolutionary multitasking via explicit autoencoding," *IEEE Trans. Cybern.*, vol. 49, no. 9, pp. 3457–3470, Sep. 2019.

[49] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, 2000.

[50] X. Zhang, Y. Tian, R. Cheng, and Y. Jin, "An efficient approach to nondominated sorting for evolutionary multi-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 2, pp. 201–213, Apr. 2015.

[51] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Syst.*, vol. 9, no. 4, pp. 115–148, 1995.

[52] K. Deb and M. Goyal, "A combined genetic adaptive search (GeneAS) for engineering design," *Comput. Sci. Informat.*, vol. 26, no. 4, pp. 30–45, 1996.

[53] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB platform for evolutionary multi-objective optimization," *IEEE Comput. Intell. Mag.*, vol. 12, no. 4, pp. 73–87, Nov. 2017.

[54] R. A. Rossi and N. K. Ahmed, "The network data repository with interactive graph analytics and visualization," in *Proc. 29th Conf. Artif. Intell. (AAAI)*, 2015, pp. 4292–4293. [Online]. Available: http://networkrepository.com

[55] J. Liang, M. Gong, H. Li, C. Yue, and B. Qu, "Problem definitions and evaluation criteria for the cec special session on evolutionary algorithms for sparse optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, Rep. 2018001, 2018.

[56] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. Int. Conf. Very Large Data Bases*, 1994, pp. 487–499.

[57] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. Coello Coello, F. Luna, and E. Alba, "SMPSO: A new PSO-based metaheuristic for multi-objective optimization," in *Proc. IEEE Symp. Comput. Intell. Multi Crit. Decis. Making*, 2009, pp. 66–73.

[58] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.

[59] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 29–38, Feb. 2006.

[60] Z. Wang, M. Zoghi, F. Hutter, D. Matheson, and N. D. Freitas, "Bayesian optimization in high dimensions via random embeddings," in *Proc. 23rd Int. Joint Conf. Artif. Intell.*, 2013, pp. 1778–1784.

[61] P. Baldi and K. Hornik, "Neural networks and principal component analysis: Learning from examples without local minima," *Neural Netw.*, vol. 2, no. 1, pp. 53–58, 1989.

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

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 and is also a Postdoctoral Research Fellow with the Department of Computer Science, City University of Hong Kong, Hong Kong. His current research interests include multiobjective optimization methods and their applications.

Dr. Tian was a recipient of the 2018 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award and the 2020 *IEEE Computational Intelligence Magazine* Outstanding Paper Award.

Chang Lu received the B.Sc. degree from Anhui University, Hefei, China, in 2017, where he is currently pursuing the M.Sc. degree with the School of Computer Science and Technology.

His current research interests include multiobjective optimization and machine learning.

Xingyi Zhang (Senior Member, IEEE) received the B.Sc. degree from Fuyang Normal College, Fuyang, China, in 2003, and the M.Sc. and Ph.D. degrees from the Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2009, respectively.

He is currently a Professor with the School of Computer Science and Technology, Anhui University, Hefei, China. His current research interests include unconventional models and algorithms of computation, multiobjective optimization, and membrane computing.

Prof. Zhang was a recipient of the 2018 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award and the 2020 *IEEE Computational Intelligence Magazine* Outstanding Paper Award.

Kay Chen Tan (Fellow, IEEE) received the B.Eng. degree (First Class Hons.) in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, U.K., in 1994 and 1997, respectively.

He is a Full Professor with the Department of Computer Science, City University of Hong Kong, Hong Kong. He has published over 200 refereed articles and six books.

Prof. Tan is the Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. He was the Editor-in-Chief of the *IEEE Computational Intelligence Magazine* from 2010 to 2013, and currently serves as an editorial board member of more than ten journals. He was an Elected Member of the IEEE CIS AdCom from 2017 to 2019.

Yaochu Jin (Fellow, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996, respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Bochum, Germany, in 2001.

He is currently a Distinguished Chair Professor of computational intelligence with the Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. He was a Finland Distinguished Professor and a Changjiang Distinguished Visiting Professor. He has (co)authored over 300 peer-reviewed journal and conference papers and been granted eight patents on evolutionary optimization.

Dr. Jin was a recipient of the 2014 and 2016 *IEEE Computational Intelligence Magazine* Outstanding Paper Award, the 2018 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award, and the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology. He has been named a Highly Cited Researcher for 2019 by the Web of Science Group. He is the Co-Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and *Complex & Intelligent Systems*. He is also an Associate Editor or an Editorial Board Member of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, the IEEE TRANSACTIONS ON CYBERNETICS, the IEEE TRANSACTIONS ON NANOBIOSCIENCE, *Evolutionary Computation*, *BioSystems*, *Soft Computing*, and *Natural Computing*. He was an IEEE Distinguished Lecturer from 2017 to 2019.