A Dynamic Knowledge-guided Coevolutionary Algorithm for Large-Scale Sparse Multiobjective Optimization Problems

Yingwei Li, Xiang Feng, and Huiqun Yu

***Abstract*—Large-scale sparse multiobjective optimization problems (SMOPs) exist widely in the real-world applications. To solve such issues, algorithms are often required to possess the ability to handle the high-dimensional decision space and discover the sparse distribution of optimal solutions simultaneously. For this reason, most existing multiobjective optimization evolutionary algorithms (MOEAs) can not obtain the satisfactory results. In this paper, a dynamic knowledge-guided coevolutionary algorithm is proposed for solving large-scale SMOPs, which suggests a cooperative coevolutionary framework tailored for large-scale SMOPs. Specifically, the variable selection is performed initially for the dimension reduction, and two populations are evolved in the original decision space and the reduced decision space, respectively. After the offspring generation, the interaction between these two populations is conducted to share the useful information. In addition, a dynamic update mechanism is suggested to utilize the sparsity knowledge dynamically during the whole evolutionary process. The superiority of the proposed algorithm is demonstrated by applying it to a variety of benchmark test instances and real-world test instances with the comparison of five state-of-the-art MOEAs.**

***Index Terms*—Sparse Pareto optimal solutions, large-scale multiobjective optimization, cooperative coevolution, evolutionary algorithm.**

I. Introduction

A. Background and Significance

Several objectives need to be optimized simultaneously in some optimization problems, which are known as multiobjective optimization problems (MOPs). Since the conflicts among objectives, a single solution can not make all the objectives optimal in the meantime. Instead, a set of solutions should be found to solve MOPs. Hence, the evolutionary algorithms (EAs) are usually employed to tackle MOPs since a set of solutions can be obtained in one loop. The optimal solutions of MOPs are called Pareto set (PS), and the Pareto front (PF) is formed by the corresponding values of PS in the objective space.

For many real-world MOPs, the optimal solutions are sparse. In other words, zero decision variables account for a large proportion in the optimal solutions, and this kind of MOPs is called sparse multiobjective optimization problems (SMOPs). For instance, in the feature selection (FS) problem, we need to get the highest accuracy of classification and choose the fewest features in the meantime [1] [2]. In most cases, a large number of candidate features are given while only a small part of them are useful for the classification, which means the optimal solutions in the FS problem are sparse. Therefore, it can be characterized as a SMOP. In the neural network training problem, both the best performance and the lowest complexity of models are expected [3]. Besides, most of weights are expected to be zero to alleviate overfitting [4]–[6], which indicates that the neural network training problem can also be characterized as a SMOP. In summary, SMOPs exist widely in many fields of science and engineering, such as portfolio optimization [7], pattern mining [8], sparse regression [9], etc. Most of real-world SMOPs have so many decision variables that these SMOPs are large-scale multiobjective optimization problems (LMOPs) in the meantime, termed large-scale SMOPs.

On account of the high-dimensional nature of decision space in the large-scale SMOPs, they are initially treated as LMOPs, and the multiobjective optimization evolutionary algorithms (MOEAs) designed for LMOPs are adopted to tackle large-scale SMOPs, which can be generally classified into three different categories.

In the first category, the decision variables are decomposed into several groups and the corresponding algorithms are designed to optimize them separately. For instance, in MOEA/DVA [10], the decision variables are divided into three categories according to the decision variable analysis method. In LMEA [11], a clustering method is adopted to divide the decision variables into diversity-related ones and convergence-related ones. In the second category, the LMOP is converted into a small-scale optimization problem which can be tackled by the traditional optimization methods. In WOF [12], each so-

This work is supported by the National Natural Science Foundation of China (No.62276097), Key Program of National Natural Science Foundation of China (No.62136003), National Key Research and Development Program of China (No.2020YFB1711700), Special Fund for Information Development of Shanghai Economic and Information Commission (No.XX-XXFZ-02-20-2463) and Scientific Research Program of Shanghai Science and Technology Commission (No.21002411000). *(Corresponding author: Xiang Feng.)*

Yingwei Li, Xiang Feng, and Huiqun Yu are with the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai 200237, and also with the Shanghai Engineering Research Center of Smart Energy, China (y21210042@mail.ecust.edu.cn; xfeng@ecust.edu.cn; yhq@ecust.edu.cn;).

lution is corresponding to a low-dimensional weight vector. In this way, the original high-dimensional optimization problem can be converted into a low-dimensional one. In LSMOF [13], two weight variables are designed for searching better solutions along different directions so that the dimension of decision space can be reduced in this way. In the third category, the novel search strategies [14]–[16] are suggested to generate offsprings. For example, a new particle updating strategy is proposed based on the competitive swarm optimizer [17] in LMOCSO [15], which makes a great improvement in both convergence speed and effectiveness.

However, the above algorithms are mainly tailored for LMOPs, which means the sparse property of Pareto optimal solutions is ignored. For this reason, their performance on large-scale SMOPs is not as decent as expected. With the increasing attention to such issues, large-scale SMOPs have been treated as an independent branch of LMOPs in the recent years, and a series of MOEAs have been proposed for the better performance on such issues. In SparseEA [18], the novel strategies for population initialization and offspring generation are suggested to take advantage of the sparsity of optimal solutions. However, the obtained prior knowledge is static during the evolutionary process, which means the obtained sparse distribution of optimal solutions may not be suitable at the latter stage of evolution. In MOEA/PSL [19], two unsupervised neural networks [20], [21] are trained to learn compact representation and sparse distribution, respectively. However, the computational cost is not satisfactory and the performance may be further improved with the guidance of sparsity knowledge. In summary, on the one hand, the prior sparsity knowledge should be highly-regarded during the whole evolutionary process. On the other hand, it is anticipated to find the sparse distribution of Pareto optimal solutions at an acceptable computational cost.

B. Contributions

To address the above issues, this article aims to make the sparsity prior knowledge utilized throughout the whole evolutionary process. Based on this consideration, a dynamic knowledge-guided coevolutionary algorithm is proposed for solving large-scale SMOPs, which is denoted as DKCA. Generally speaking, the cooperative coevolutionary framework is adopted in the proposed DKCA, and the obtained sparse distribution of Pareto optimal solutions is updated dynamically according to the performance of each decision variable during the whole evolutionary process, which aims to make the obtained solutions closer to the optimal ones. Specifically, the main contributions of the proposed algorithm are summarized as follows.

1) A cooperative coevolutionary framework is proposed to discover the sparse distribution of Pareto optimal solutions and improve the quality of offsprings. Specifically, a decision variable selection method is performed at first to select variables with signific, and the reduced decision space is formed based on the dimension of selected variables. Afterward, two populations are initialized, which are evolved in the original decision space (OP) and the reduced decision space (RP), respectively. Moreover, these two populations cooperate with each other after the offspring generation, which aims to share the valuable information to generate the offsprings with higher quality.

2) With the aim of steering the optimization in the correct direction during the whole evolutionary process, a score update mechanism is designed to adjust the importance of each decision variable dynamically. Specifically, the scores of decision variables are calculated before the evolution of population, which are regarded as the metric to evaluate the significance of each decision variable. If the sparsity of RP has not been changed for several generations, it can be assumed that the current sparsity is appropriate for the optimization problem. Hence, the current nonzero decision variables will have a lower possibility to be set to zero due to the significance they show during the evolutionary process. By this means, the significant decision variables can be selected precisely and the direction of evolution can be adjusted timely.

3) To verify the competitiveness of the proposed DKCA, it is tested on various benchmark SMOPs and real-world SMOPs under the comparison with five state-of-the-art MOEAs. According to the experimental results, the proposed DKCA performs significantly better than other compared algorithms, which shows the superiority of DKCA in solving different kinds of large-scale SMOPs.

The rest of this paper is organized as follows. In Section II, the basic definitions of large-scale SMOPs are introduced, and the existing MOEAs for large-scale SMOPs, existing EAs for high-dimensional FS problems, and the existing coevolutionary algorithms for LMOPs are overviewed briefly. In Section III, the specific procedures of the proposed algorithm are explained in detail. In Section IV, the experimental results of the proposed algorithm are depicted and analyzed. In the end, the conclusions and future work are presented in Section V.

II. Related Work

A. Large-scale SMOPs

Specifically, a MOP without constraint can be defined as follows:

$$\begin{aligned} \min_{\mathbf{x}} \quad & \mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})) \\ \text{s.t.} \quad & \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

Where $\mathbf{x} = (\mathrm{x}_1, \ldots, \mathrm{x}_n)$ is a candidate solution, which consists of n decision variables; $\mathbf{f} : \Omega \rightarrow \mathbb{R}^m$ is the objective function set which consists of m objectives; $\mathbb{R}^m$ and $\Omega \subseteq \mathbb{R}^n$ are the objective space and the decision space, respectively. Considering two solutions $\mathbf{x}_i$ and $\mathbf{x}_j$ in a minimize optimization problem, if $f_k(\mathbf{x}_i) \leq f_k(\mathbf{x}_j)$ for

each $k \in \{1, \ldots, m\}$ and $\mathbf{f}(\mathbf{x}_i) \neq \mathbf{f}(\mathbf{x}_j)$, it is said that $\mathbf{x}_i$ dominates $\mathbf{x}_j$, expressed as $\mathbf{x}_i \leq \mathbf{x}_j$. Moreover, if $\mathbf{x}^*$ is not dominated by any solution, it is called a Pareto optimal solution.

A MOP is called a SMOP on condition that only a small part of decision variables of Pareto optimal solutions are nonzero. In most real-world applications, SMOPs are often accompanied with a large number of decision variables, which are known as large-scale SMOPs. To solve large-scale SMOPs, MOEAs not only need to tackle the curse of dimensionality, but also need to handle the sparsity of Pareto optimal solutions. With the development of the research in this field, several MOEAs have shown their effectiveness in solving such issues, which are reviewed briefly in the next subsection.

B. Existing MOEAs for large-scale SMOPs

Generally speaking, the existing MOEAs for solving large-scale SMOPs can be divided into two categories.

In the first category, the prior knowledge of sparsity is adopted to evaluate the importance of each decision variable. In SparseEA [18], the score of each decision variable is calculated before the evolution, and the optimization is leaded by the comparison of these scores. Moreover, the genetic operators based on the scores are also suggested in order to ensure the sparsity of offsprings. Subsequently, with the aim of paying more attention to optimize the nonzero variables, the SparseEA2 [22] is proposed to improve the connection between binary variables and real variables. More recently, in MDR-SAEA [23], the prior knowledge of sparsity is obtained based on the FS method, and a dimension reduction method with several stages is proposed to handle the large-scale decision variables in SMOPs. In S-ECSO [24], a novel operator is designed based on the optimization of a strongly convex function to generate sparse solutions, and these sparse solutions will be compared with the original ones to decide which one will be reserved. Then, the enhanced competitive swarm optimizer is employed to maintain the balance between exploration and exploitation. For this kind of MOEAs, most of them only obtain the static sparse information before the evolution. Therefore, during the evolutionary process, the changeless sparsity knowledge may not be suitable for the current population any more or even mislead the direction of optimization.

In the second category, more attention is paid to discover the sparse distribution of the Pareto optimal solutions. Based on the basic idea of FS, the pattern mining approach [8] is used in PM-MOEA [25]. Afterward, the minimum candidate sets and maximum candidate sets of the nonzero variables are obtained, which determine the dimension of offspring generation. In MOEA/PSL [19], a denoising autoencoder [21] and a restricted Boltzmann machine [20] are trained to approximate the Pareto optimal subspace, which amis to reduce the decision variables by the representation of adaptively adjustable hidden layers. For this kind of MOEAs, high computational cost is needed for mining the sparse distribution of Pareto optimal solutions. Moreover, due to the lack of prior sparsity knowledge, this kind of MOEAs may have unsatisfactory performance at the early stage of evolution.

C. Existing EAs for high-dimensional FS problems

To some extent, the large-scale SMOPs can be seemed as FS problems, and many algorithms are designed based on this idea [19] [25]. Therefore, the EAs for high-dimensional FS problems are overviewed briefly in this subsection. In general, this kind of algorithms can be divided into three categories according to their different techniques for high-dimensional search space.

In the first category, the novel genetic operators are designed for the high-dimensional decision variables. In SaPSO [26], different candidate solution generation strategies are maintained in the strategy pool, and a self-adaptive mechanism is adopted to calculate the probability and choose the suitable strategy. In SM-MOEA [27], a dimension reduction operator and an individual repairing operator are suggested based on the proposed steering matrix, which aims to guide the direction of evolution. In the second category, the local search strategy is embedded in EAs to maintain the balance between exploration and exploitation. In PSO-LSRG [28], a novel local search method based on k-nearest neighbor is proposed to facilitate the exploitation. In HPSO-LS [29], according to the correlation information of features, the local search method is adopted to lead the search process. In the third category, the cooperative coevolutionary framework is employed. In PSO-EMT [30], two related tasks are established to share the valuable information between each other. In SAEA-PRG [31], the random grouping strategy is adopted to divide the original population into several subpopulations to reduce the dimension of search space. In conclusion, the cooperative cooperation is regarded as a useful way for high-dimensional FS problems. However, to the best of our knowledge, there are rarely algorithms based on this idea for tackling large-scale SMOPs.

D. Existing Coevolutionary Algorithms for LMOPs

In the recent years, CAs have shown the great superiority in solving different kinds of MOPs [32], such as constrained optimization problems [33] [34], multimodal optimization problems [35], dynamic optimization problems [36], many-objective optimization problems [37], and so on. In this section, several CAs tailored for LMOPs are overviewed to show the competitiveness of CAs on LMOPs.

Overall, the basic idea of these MOEAs is to divide the large-scale decision variables into several groups, and each group is optimized alternately while fixing the rest. In DECC-G [38], the grouping strategy and the adaptive weighting strategy are proposed to construct the cooperative coevolutionary framework. In CCGDE3 [39],

the decision variables are decomposed into several groups with equal size randomly. Then, the GDE3 algorithm [40] is employed as the optimizer. In TS [41], two different strategies are considered to combine the interactions of all the objectives. In DPCCMOEA [42], a modified variable analysis method is proposed, which makes the decision variables decomposed into several groups, and each group is optimized by a subpopulation.

In summary, large-scale SMOPs still remains challenging and it is expected for MOEAs to make improvements aiming at the issues mentioned above. Besides, CAs have shown great effectiveness in both FS problems and LMOPs. However, to the best of our knowledge, the basic idea of CAs has not been used in solving large-scale SMOPs yet. Therefore, the performance of CAs on such problems is worth looking forward to. Based on these considerations, this paper suggests a dynamic knowledge-guided coevolutionary algorithm for solving large-scale SMOPs. Under the cooperation of two populations with different scales of decision variables, the obtained knowledge of sparsity can be adjusted dynamically during the whole evolutionary process. In the next section, the details of the proposed algorithm are elaborated.

III. Proposed Algorithm

A. General Framework of DKCA

The proposed DKCA is based on the cooperative coevolution of two populations, and the general framework is shown in Fig. 1 to make it more visually. Moreover, the detailed process of DKCA is exhibited in Algorithm 1. As can be seen, before the main loop, the variable selection is performed at first to select the significant variables and calculate the score of each decision variable. Based on the dimension of selected variables d and the dimension of original variables D, two populations with the same size N but different dimensions of decision variables are initialized, which are denoted as RP and OP, respectively. There are five main steps during the optimization, i.e., mating selection, offspring generation, variable replacement, environmental selection, and score update. In the mating selection, the non-dominated sorting and the calculation of crowding distance [43] are performed on each solution in OP and RP. Afterward, based on the non-dominated front number and crowding distance of each solution, the binary tournament selection is performed in both OP and RP so that $2N$ solutions can be selected as parents, termed $Parent_{OP}$ and $Parent_{RP}$, respectively. In the offspring generation, according to the scores of variables, N offsprings of OP and RP are generated by $Parent_{OP}$ and $Parent_{RP}$ separately. In the variable replacement, since the variables in RP are the results of careful selection, the nonzero variables of individuals in RP are considered more valuable than those in OP. For this reason, the corresponding decision variables of each individual in OP are also set to be nonzero. Subsequently, the environmental selection is conducted on the combination of offsprings and their own parents, which aims to choose N solutions as the next generation. Finally, the scores of decision variables are adjusted dynamically based on their performance during the evolutionary process. In the next two subsections, the major techniques of DKCA are described in detail.

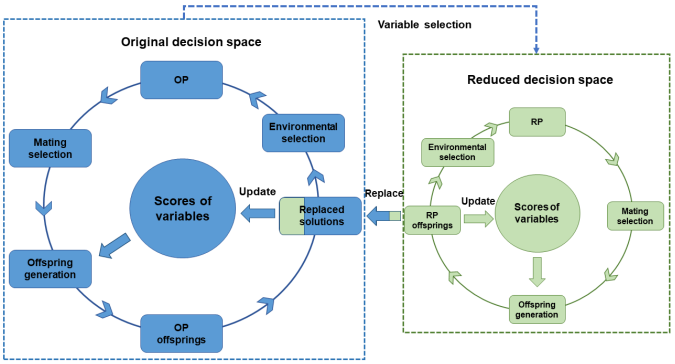

Fig. 1. General framework of the proposed DKCA.

Algorithm 1: General framework of DKCA

Input: N(population size), D(number of original decision variables), r(ratio of selected decision variables), k(number of generations to evaluate the sparsity of population), s(number of cycles in the sampling)
Output: P(final population)
$[d, Score] \leftarrow Variableselection(D, r, s)$;
$[RP, OP] \leftarrow Initialization(D, d, N, Score)$;
while *termination criterion not met* **do**
  `//Mating selection`
  $Parent_{OP} \leftarrow$ Select $2N$ parents via binary tournament selection;
  $Parent_{RP} \leftarrow$ Select $2N$ parents via binary tournament selection;
  `//Offspring generation`
  $Offspring_{OP} \leftarrow Variation(Parent_{OP}, Score)$;
  $Offspring_{RP} \leftarrow Variation(Parent_{RP}, Score)$;
  `//Variable replacement`
  Corresponding elements in $Offspring_{OP}.mask$ are replaced by nonzero ones in $Offspring_{RP}.mask$;
  `//Environmental selection`
  $OP \leftarrow OP \cup Offspring_{OP}$;
  $RP \leftarrow RP \cup Offspring_{RP}$;
  $OP \leftarrow EnvironmentalSelection(OP, N)$;
  $RP \leftarrow EnvironmentalSelection(RP, N)$;
  `//Score update`
  $Score \leftarrow Scoreupdate(RP, N, s, Score, k)$;
return OP;

B. Initialization of Cooperative Populations

The hybrid representation of solutions is adopted in DKCA, where each solution x consists of two components, a binary vector $mask$ and a real vector dec.

$$(x_1, \ldots, x_D) = (dec_1 \times mask_1, \ldots, dec_D \times mask_D) \quad (2)$$

Algorithm 2: $Variableselection(D, r, s)$

Input: D(number of original decision variables), r(ratio of selected decision variables), s(number of cycles in the sampling)
Output: d(number of selected decision variables), $Score$(scores of decision variables)

$Dec \leftarrow D \times D$ random matrix;
$Mask \leftarrow D \times D$ identity matrix;
$Standard \leftarrow 1 \times D$ zero matrix;
$[score_1, \ldots, score_D] \leftarrow 0$;
for $i = 1$ *to* s **do**
  Generate new Dec;
  $Q \leftarrow$ A population whose solutions are produced by each row of Dec and $Mask$ according to (2);
  $[Front_1, \ldots, Front_D, Front_{\text{Standard}}] \leftarrow$ Do non-dominated sorting on $Q \cup Standard$;
  $[score_1, \ldots, score_D] \leftarrow [score_1, \ldots, score_D] + [Front_1, \ldots, Front_D]$;
  $d_i \leftarrow \emptyset$;
  for $j = 1$ *to* D **do**
    if $Front_j \leq Front_{Standard}$ **then**
      $d_i \leftarrow x_j \cup d_i$;
if $|d_1 \cup d_2 \cup \ldots \cup d_s| \leq r \times D$ **then**
  $d \leftarrow |d_1 \cup d_2 \cup \ldots \cup d_s|$
else
  $d \leftarrow |d_1 \cap d_2 \cap \ldots \cap d_s|$
$Score \leftarrow [score_1, \ldots, score_D]$;
return d, $Score$;

Algorithm 3: $Initialization(D, d, N, Score)$

Input: D (number of original decision variables), d(number of selected decision variables), N(population size), $Score$(scores of decision variables)
Output: RP(population in reduced decision space), OP(population in original decision space)

$Dec_{RP} \leftarrow N \times d$ random matrix;
$Mask_{RP} \leftarrow N \times d$ zero matrix;
$Dec_{OP} \leftarrow N \times D$ random matrix;
$Mask_{OP} \leftarrow N \times D$ zero matrix;
for $k = 1$ *to* N **do**
  for $i = 1$ *to* $random() \times d$ **do**
    $[a, b] \leftarrow$ Select two decision variables randomly;
    Corresponding element of the one with smaller score is set to 1 in $Mask_{RP}$;
  for $j = 1$ *to* $random() \times D$ **do**
    $[a, b] \leftarrow$ Select two decision variables randomly;
    Corresponding element of the one with smaller score is set to 1 in $Mask_{OP}$;
$RP \leftarrow$ A population whose solutions are produced by each row of Dec_{RP} and $Mask_{RP}$ according to (2);
$OP \leftarrow$ A population whose solutions are produced by each row of Dec_{OP} and $Mask_{OP}$ according to (2);
return RP, OP;

Where D is the number of decision variables, $mask_i$ and dec_i are the binary vector and real vector of the ith decision variable, respectively. As can be seen in (2), $mask$ can define the sparsity of solutions, while dec can make solutions closer to the PF. In DKCA, the real vectors are optimized by a traditional genetic operator while the binary vectors are guided by the obtained sparsity knowledge dynamically during the evolutionary process.

Before the initialization of two populations, the selection of decision variables is performed to determine the dimension of the decision space in RP. The procedure of variable selection is presented in Algorithm 2. As can be seen, the Dec is initialized as a $D \times D$ matrix with random values, and the $Mask$ is initialized as a $D \times D$ identity matrix. The $Standard$ is a $1 \times D$ zero matrix, which is initialized to evaluate the decision variables in the non-dominated sorting. At the beginning of sampling, the population Q which has only one nonzero variable in each row is generated by Dec and $Mask$ according to (2). As a consequence, every decision variable is represented by a solution in Q. Afterward, the non-dominated front number of each solution in the union of Q and $Standard$ is calculated, which is recorded as the score of the related decision variable. In view of the dominance rule in the non-dominated rank, the smaller score a decision variable has, the lower possibility it will have to be set to zero. Moreover, the decision variables whose non-dominated front number is less than that of $Standard$'s are selected as the significant nonzero ones.

In addition, such sampling operation is repeated s times and a new Dec is initialized in each cycle to prevent the influence of random values. If the proportion of the selected variables is not more than the threshold r, the number of selected variables will be considered insufficient. Therefore, the union of variables in each cycles is adopted, termed $d_1 \cup d_2 \cup \ldots \cup d_s$; otherwise, the intersection of variables in each cycles is adopted, termed $d_1 \cap d_2 \cap \ldots \cap d_s$. As a result, the reduced decision space is formed according to the dimension of the selected decision variables.

The procedure of the initialization strategy is presented in detail in Algorithm 3. Under the cooperative coevolutionary framework, two populations with the same population size N but the different decision variable scales (i.e., D and d) are initialized. Afterward, the comparison of scores is executed between two randomly selected variables, and the one with the smaller score is recorded, whose corresponding element in the $Mask$ is set to 1. Finally, based on the obtained $Mask$ and Dec matrices, RP and OP are initialized according to (2).

To summarize, in the preparing section of DKCA, the score of each decision variable is calculated, based on which the significant variables are selected. RP is

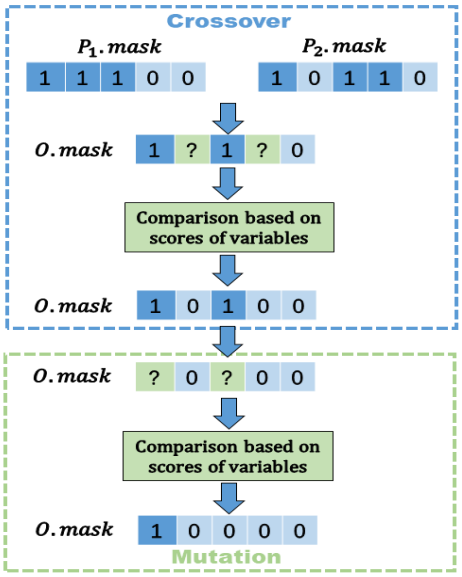

Fig. 2. The basic process of offspring generation in DKCA.

initialized according to the dimension of these selected decision variables, while OP is initialized according to the dimension of original decision variables. In the next subsection, the dynamic update mechanism of decision variable scores are introduced in detail.

C. Dynamic Score Update Mechanism

The method of offspring generation is similar to the one in SparseEA [18], and it is adopted in both RP and OP, which aims to make the optimization guided by scores of decision variables during the evolutionary process. The basic process of offspring generation is shown in Fig. 2. Specifically, P_1 and P_2 are selected randomly as parents to generate the offspring O. For the *mask* vector of O, it is initialized as the same as that of P_1 at first. In the crossover section, the different nonzero variables of P_1 and P_2 are chosen as candidate variables, and two of them are selected randomly to conduct the comparison according to their scores. For the variable with smaller score, the corresponding element in $O.mask$ is set to 1; for the variable with larger score, the corresponding element in $O.mask$ is set to 0. A random value is produced to decide which of the above operations is performed. In the mutation section, nonzero variables in $O.mask$ are selected as candidate variables, and the comparison based on scores of variables is executed in the same way mentioned above. For the *dec* vector of O, the traditional simulated binary crossover [44] and polynomial mutation [45] are adopted to generate offsprings.

To make full use of prior sparsity knowledge, scores are updated dynamically during the evolutionary process. As shown in Algorithm 4, the number of nonzero decision variables in each individual of RP is obtained, termed $[n_1, \ldots, n_N]$. Furthermore, the mode of $[n_1, \ldots, n_N]$ in the i-th generation $mode_i$ is recorded as *spar* to represent the sparsity of the current population. If $mode_i$ is equal to $mode_{i-1}$, fix will be added. Hence, the value of fix means the sparsity of the current population has been constant for fix generations continuously. Once fix is more than the threshold k, individuals whose sparsity is equal to *spar* are selected, termed Ind_{spar}. Afterward, the scores of the nonzero decision variables in Ind_{spar} are changed by

$$Score_{nonzero} = Score_{nonzero} - \lceil \frac{Score_{nonzero}}{s} \rceil \quad (3)$$

Where $Score$ is the score vector of decision variables. $nonzero$ are indices of nonzero variables in the Ind_{spar}. s is the number of cycles in the sampling.

To explain this process more clearly, the illustration of the adopted dynamic score update mechanism is shown in Fig. 3. As can be seen, we assume that the population has five individuals, and each individual has five decision variables. At first, the number of nonzero variables in each row of *mask* matrix is regarded as the sparsity of each individual. Then, the mode of all the individuals' sparsity is taken as the sparsity of the whole population in the current generation. If this sparsity keeps unchanged for k generations, the individuals with this sparsity will be selected and their nonzero decision variables will be regarded as the variables with potential. These variables are supposed to deserve a higher probability of being set

Algorithm 4: $Scoreupdate(RP, N, s, Score, k)$

Input: RP(population in reduced decision space), N(population size), s(number of cycles in the sampling), $Score$(scores of decision variables), k(number of generations to evaluate sparsity of population)

Output: $Score$(updated scores of decision variables)

```
i ← 1;
while fix < k do
    RP_new ← The new generation of RP;
    [n_1, ..., n_N] ← The number of nonzero decision variables in each individual of RP_new;
    mode_i ← mode in [n_1, ..., n_N];
    if i ≥ 2 then
        if mode_i = mode_{i-1} then
            fix ← fix + 1;
        else
            fix ← 1;
    else
        fix ← 1;
    i ← i + 1;
spar ← mode_i;
Ind_spar ← individuals whose sparsity is equal to spar in RP;
Score ← scores of nonzero variables in Ind_spar are recalculated according to (3);
return Score;
```

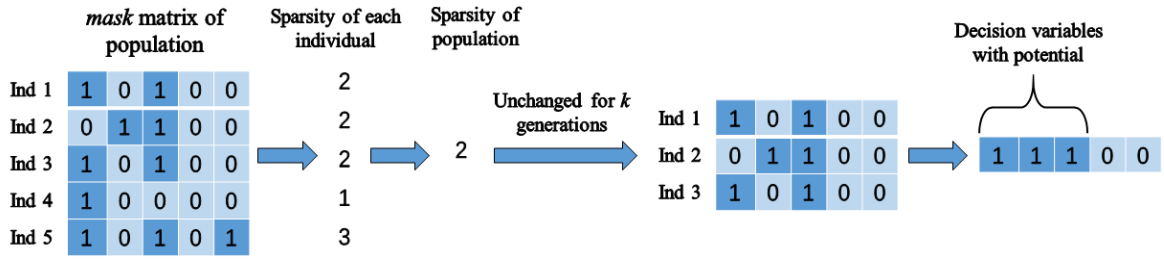

Fig. 3. Illustration of the dynamic score update mechanism used on a population consisting of five individuals with five decision variables.

TABLE I
PARAMETER SETTINGS AND CHARACTERISTICS OF EIGHT BENCHMARK SMOPS AND THREE REAL-WORLD SMOPS.

Benchmark problem	No. of variables	No. of objectives	Sparsity of Pareto optimal solutions	Features of PF	Features of landscape	Separability of objective functions
SMOP1-SMOP8	**100** **500** **1000** **5000**	**2**	0.1	Linear(SMPOP1-3) Convex(SMOP4-6) Concave(SMOP7, 8)	Mixed (SMOP1) Multimodal (SMOP2, 6, 7) Unimodal (SMOP3, 5) Deceptive (SMOP2, 3, 4, 8)	Separable (SMOP1-6) Mixed (SMOP7) Nonseparable (SMOP8)
Neural network training problem	No. of variables	Dataset	Size of hidden layer	No. of samples	No. of features	NO. of classes
NN1	**401**	Climate Model Simulation Crashes[1]	20	540	18	2
NN2	**512**	Statlog (German)[2]	20	1000	24	2
NN3	**1241**	Connectionist Bench (Sonar)[1]	20	208	60	2
Portfolio optimization problem	No. of variables	Dataset	No. of instruments	Length of each instrument		
PO1	**100**	EURCHF[3]	100	50		
PO2	**500**	EURCHF[3]	500	50		
PO3	**1000**	EURCHF[3]	1000	50		
Sparse signal reconstruction problem	No. of variables	Dataset	Length of signal	Length of received signal	Sparsity of signal	
SR1	**128**	Synthetic [46]	128	60	35	
SR2	**512**	Synthetic [46]	512	240	130	
SR3	**1024**	Synthetic [46]	1024	480	260	

[1] http://archive.ics.uci.edu/ml/index.php
[2] https://www.csie.ntu.edu.tw//%7ecjlin/libsvmtools/datasets/binary.html
[3] https://www.metatrader5.com/en

to nonzero values. Therefore, their scores are updated according to (3). By this means, the scores of decision variables are able to be adjusted dynamically based on their performance during the evolutionary process.

In summary, since the adopted offspring generation method in DKCA mainly depends on the scores of decision variables, the accuracy of scores has an important impact on the quality of generated solutions. Therefore, scores of the selected potential variables should be highly-regarded during the whole evolutionary process. Based on this consideration, the dynamic score update mechanism is designed in DKCA, which provides the constantly changeable sparsity knowledge for the evolution.

D. Computational complexity of DKCA

During the optimization, there are five main steps in each generation, which are mating selection, offspring generation, variable replacement, environmental selection, and score update. The computational complexity of each step is $O(N)$, $O(ND)$, $O(Nd)$, $O(MN^2)$, and $O(Ndk)$, respectively. Since two populations with different variable scales are involved in DKCA, some evolutionary operations (i.e., mating selection, offspring selection, and environmental selection) are performed twice in each generation. As a consequence, the overall computational complexity is $O(2N + ND + 2Nd + 2MN^2 + Ndk)$. After simplification, the computational complexity of DKCA is $O(MN^2)$.

IV. Experimental Studies

In order to verify the effectiveness and efficiency of DKCA on solving various large-scale SMOPs, it is tested on a variety of benchmark SMOPs and real-world SMOPs with the comparison of five state-of-the-art MOEAs in this section. All the experiments are conducted on a PC with an Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz, 8 GB RAM, and MATLAB R2021b. Moreover, the codes of the compared algorithms, evaluation metrics, and benchmark problems are provided by PlatEMO [47]. The source code of DKCA is available at https://github.com/smlzxxx/DKCA.

A. Parameter Settings

1) *Problems:* Eight benchmark SMOPs [18] and three real-world SMOPs are adopted to test the performance of DKCA on tackling different kinds of SMOPs. Specifically,

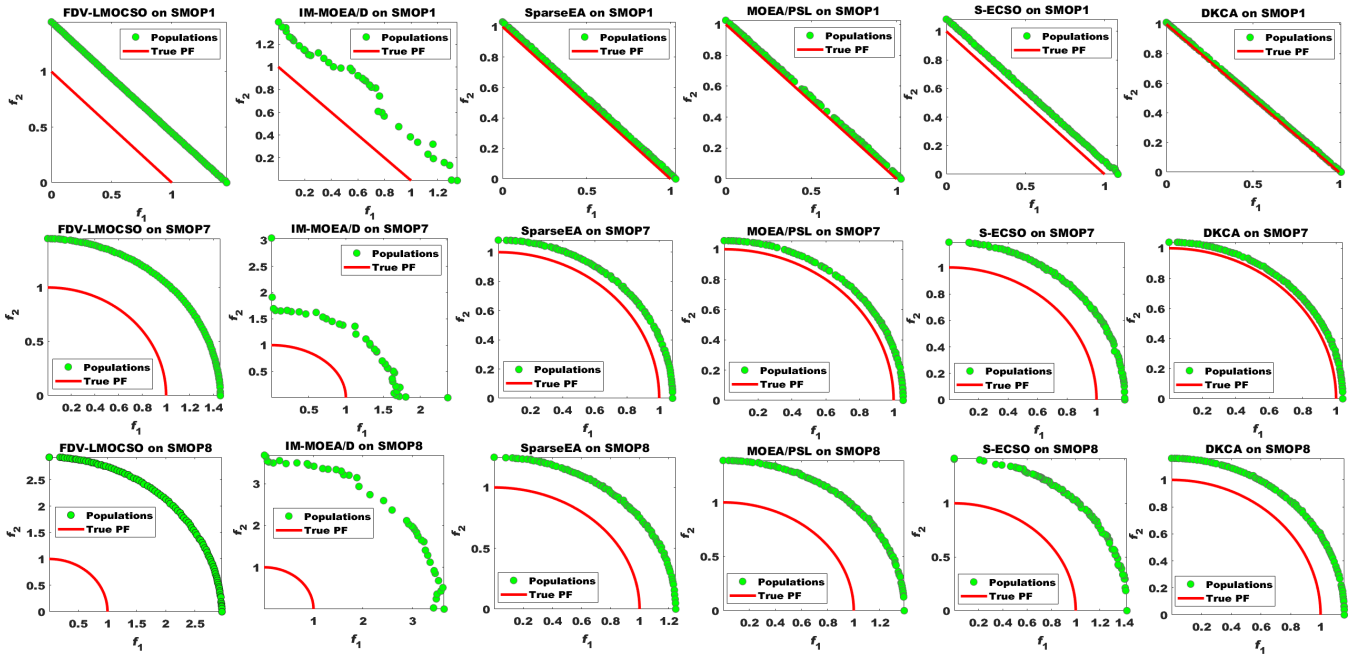

Fig. 4. Non-dominated solution sets with median IGD obtained by FDV-LMOCSO, IM-MOEA/D, SparseEA, MOEA/PSL, S-ECSO, and the proposed DKCA on SMOP1, SMOP7, and SMOP8 with 1000 variables.

SMOP1-SMOP8 are benchmark test problems with various difficulties, while NN1-NN3, PO1-PO3, SR1-SR3 represent the neural network training problem, the portfolio optimization problem, and the sparse signal reconstruction problem with three datasets, respectively. The specific parameter settings and characteristics of the involved SMOPs are shown in Table I.

2) *Performance metrics:* For the benchmark SMOPs, both the IGD indicator [48] and HV values [49] are employed as the evaluation metric. For the real-world SMOPs, since the true Pareto fronts of these SMOPs are unknown, the HV values are calculated to measure the solutions obtained by each algorithm. Specifically, the IGD values are calculated by the 10000 reference points with the uniform distribution on each PF [50], and the HV values are calculated with the reference point (1, 1). Moreover, the Wilcoxon rank-sum test [51] is adopted so that the differences of various algorithms on each test problem can be presented more directly, and the significance level is set to 0.05. In particular, '+', '−', and '=' denote the results of compared algorithms are significantly better, significantly worse, and statistically similar to those of the proposed DKCA, respectively. For each algorithm, the experimental results are obtained by 30 independent runs on each test instance.

3) *Algorithms:* Five state-of-the-art MOEAs (i.e., FDV-LMOCSO [52], IM-MOEA/D [53], SparseEA [18], MOEA/PSL [19], and S-ECSO [24]) are adopted as the compared algorithms, where FDV-LMOCSO and IM-MOEA/D are two state-of-the-art MOEAs tailored for LMOPs based on the dimension reduction and the inverse modeling, respectively. SparseEA, MOEA/PSL, and S-ECSO are three effective MOEAs tailored for large-scale SMOPs recently. All the parameter settings of compared MOEAs are kept the same as the ones in their original literature to keep the superiority. Specifically, For FDV-LMOCSO, the fuzzy evolution rate and the step acceleration are set to 0.8 and 0.4, respectively. For IM-MOEA/D, the number of reference vectors is set to 10. For S-ECSO, the parameter $\lambda_{initial}$ in the strongly convex sparse operator is set to 0.35. For DKCA, the sensitivity analysis of parameters is done based on a series of environments, which is exhibited in supplementary material. According to the experimental results, the number of generations to evaluate sparsity of population k and the ratio of selected decision variables r are set to 4 and 0.3, respectively.

4) *Genetic operators:* For FDV-LMOCSO, the competitive swarm optimizer and the polynomial mutation [45] are used. For IM-MOEA/D, the reproduction based on Gaussian process and the polynomial mutation are adopted. For SparseEA, MOEA/PSL, and DKCA, the simulated binary crossover [44] and the polynomial mutation are employed. Furthermore, the crossover probability, the mutation probability, and the distribution index of both crossover and mutation are set to 1, $1/D$ (D symbolizes the number of decision variables), and 20, respectively.

5) *Population size and terminal condition:* For a fair comparison, the population size is set to the same value on all the algorithms, which is set to 100 for the benchmark test instances while 50 for the real-world test instances. In addition, the number of function evaluations is set to 100 $\times D$ for the benchmark test instances while 20000 for the real-world test instances, where D stands for the number of decision variables.

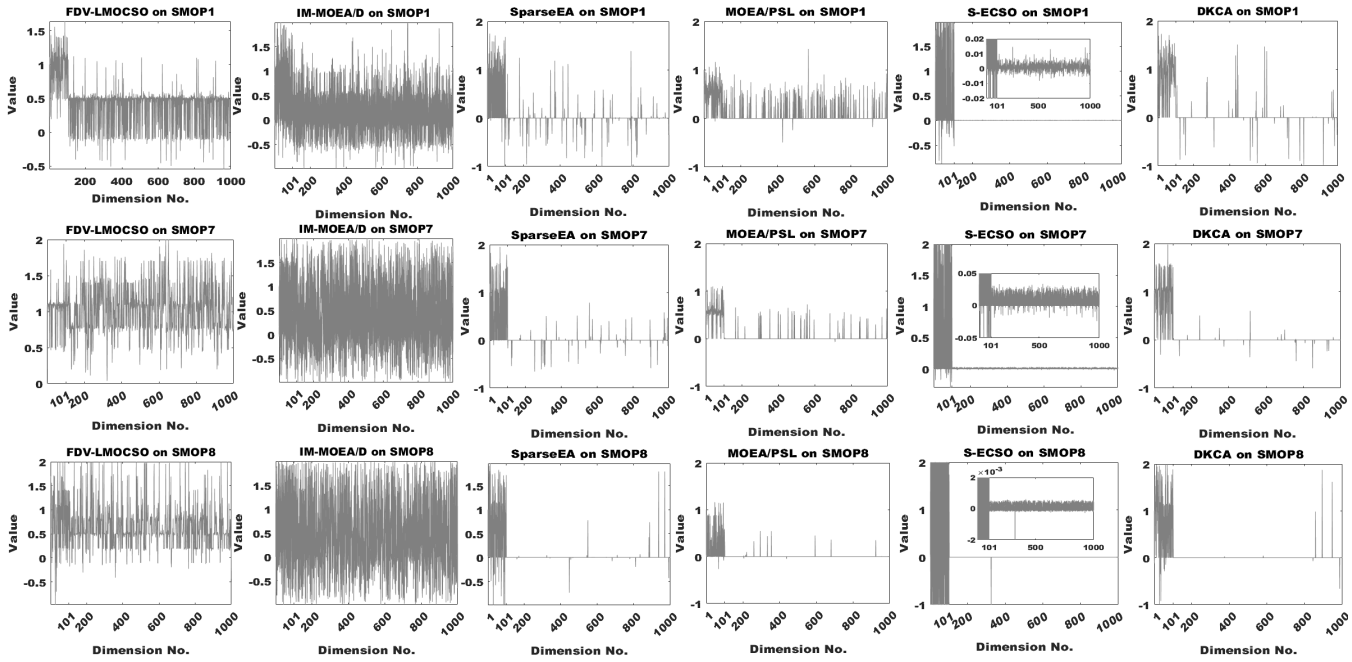

Fig. 5. Parallel coordinates plot of the decision variables of solutions obtained by FDV-LMOCSO, IM-MOEA/D, SparseEA, MOEA/PSL, S-ECSO, and the proposed DKCA on SMOP1, SMOP7, and SMOP8 with 1000 variables.

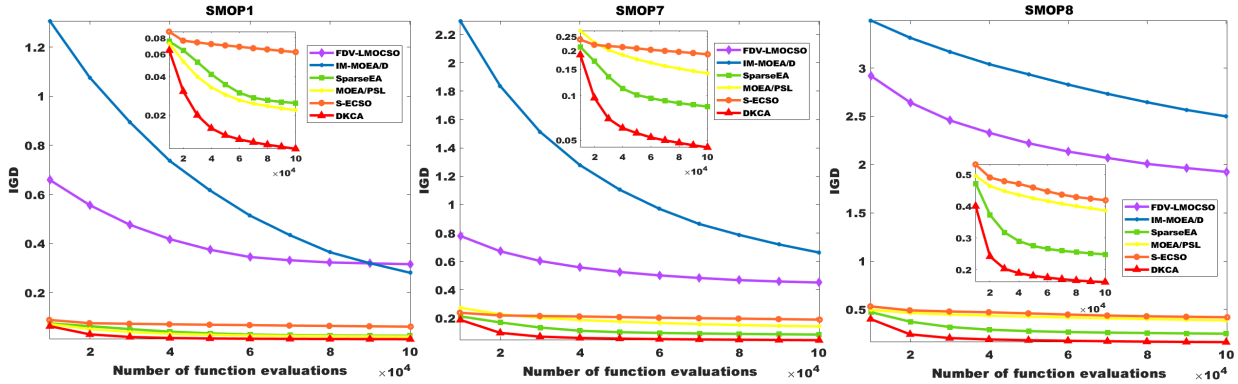

Fig. 6. Convergence profiles of IGD values obtained by the proposed DKCA and other compared algorithms on SMOP1, SMOP7,and SMOP8 with 1000 deicision variables.

B. Experimental Results on Benchmark SMOPs

Each algorithm has 30 independent runs on SMOP1-SMOP8 with 100, 500, 1000, and 5000 decision variables. The IGD values obtained by DKCA and other compared algorithms are presented in Table II, and the non-dominated solution sets with median IGD are plotted in Fig. 4. For the statistical results, the proposed DKCA performs significantly better than FDV-LMOCSO, IM-MOEA/D, SparseEA, MOEA/PSL, and S-ECSO on 32, 32, 20, 32, and 28 test problems, respectively. For the graphical results, it is obviously that the solutions obtained by DKCA have better convergence than those obtained by FDV-LMOCSO, IM-MOEA/D, and S-ECSO, while better diversity than those obtained by SparseEA and MOEA/PSL. In addition, the HV values are also calculated and summarized as summation of '+/-/=' according to the results of Wilcoxon rank-sum test, which are shown in Table III. The specific HV values can be found in supplementary material. As can be seen, the proposed DKCA is also superior to others. The competitive experimental results demonstrate the solutions with good performance in both convergence and diversity can be obtained by DKCA.

To further explore the performance of DKCA, the parallel coordinates plot [54] of the decision variables of solutions obtained by DKCA and compared algorithms on SMOP1, SMOP7, and SMOP8 with 1000 decision variables is presented in Fig. 5, where the first hundred decision variables are nonzero ones while others are zero in the Pareto optimal solutions. As can be seen, for FDV-LMOCSO and IM-MOEA/D, almost all the decision variables of the obtained solutions are nonzero. As For four MOEAs tailored for SMOPs, most decision variables

TABLE II
THE EXPERIMENTAL RESULTS OF DKCA AND OTHER COMPARED ALGORITHMS ON BENCHMARK SMOPS. THE BEST RESULT IN EACH ROW IS HIGHLIGHTED.

Problem	D	FDV-LMOCSO	IM-MOEA/D	SparseEA	MOEA/PSL	S-ECSO	DKCA
SMOP1	100	3.1709e-1 (1.80e-2) -	2.4539e-1 (3.01e-2) -	1.0459e-2 (2.57e-3) -	1.0890e-2 (4.93e-3) -	4.9379e-2 (6.71e-3) -	**5.2623e-3 (4.75e-4)**
	500	3.1974e-1 (1.25e-2) -	2.8831e-1 (2.50e-2) -	1.7974e-2 (3.08e-3) -	2.0096e-2 (2.95e-3) -	6.0289e-2 (3.27e-3) -	**7.1173e-3 (1.46e-3)**
	1000	3.1637e-1 (1.23e-2) -	2.8132e-1 (2.51e-2) -	2.4725e-2 (2.35e-3) -	2.1902e-2 (1.97e-3) -	6.2039e-2 (2.69e-3) -	**1.0067e-2 (1.72e-3)**
	5000	3.2275e-1 (1.03e-2) -	2.9672e-1 (2.75e-2)	3.8852e-2 (1.53e-3) -	4.1037e-2 (9.68e-3) -	6.3242e-2 (1.27e-3) -	**2.5530e-2 (1.57e-3)**
SMOP2	100	1.2798e+0 (8.63e-3) -	1.0255e+0 (1.15e-1) -	3.2469e-2 (8.16e-3) -	2.5371e-2 (1.35e-2) -	1.1291e-1 (1.44e-2) -	**1.2076e-2 (4.59e-3)**
	500	1.2769e+0 (1.72e-3) -	1.1132e+0 (1.09e-1) -	4.9158e-2 (7.71e-3) -	4.5906e-2 (8.38e-3) -	1.2994e-1 (6.54e-3) -	**2.1569e-2 (4.96e-3)**
	1000	1.2757e+0 (1.50e-3) -	1.1005e+0 (9.49e-2) -	6.8712e-2 (6.11e-3) -	5.2404e-2 (4.12e-3) -	1.3412e-1 (4.23e-3) -	**3.2753e-2 (4.36e-3)**
	5000	1.2750e+0 (5.91e-4) -	1.1434e+0 (7.95e-2) -	1.0098e-1 (2.63e-3) -	8.1889e-2 (4.90e-3) -	1.3601e-1 (2.34e-3) -	**7.0076e-2 (3.58e-3)**
SMOP3	100	6.9199e-1 (8.15e-2) -	1.9737e+0 (4.86e-2) -	1.7694e-2 (4.25e-3) -	4.3618e-2 (6.25e-2) -	2.7256e-1 (2.40e-1) -	**5.6425e-3 (1.22e-3)**
	500	6.4390e-1 (3.69e-4) -	2.1088e+0 (4.31e-2) -	2.1272e-2 (3.19e-3) -	8.6141e-2 (2.91e-1) -	6.0005e-2 (3.34e-3) -	**6.4072e-3 (9.65e-4)**
	1000	6.6413e-1 (9.25e-2) -	2.1408e+0 (3.75e-2) -	2.8968e-2 (3.07e-3) -	2.2590e-2 (5.36e-3) -	6.2348e-2 (1.76e-3) -	**9.6210e-3 (1.70e-3)**
	5000	6.5573e-1 (7.08e-2) -	2.1817e+0 (3.90e-2)	4.4046e-2 (1.42e-3) -	3.6143e-2 (2.36e-3) -	6.2673e-2 (9.57e-4) -	**2.5913e-2 (1.30e-3)**
SMOP4	100	6.9129e-1 (1.31e-2) -	5.1051e-1 (6.75e-2) -	4.7851e-3 (2.33e-4) =	4.9544e-3 (3.13e-4) -	**3.9472e-3 (5.92e-5) +**	4.7397e-3 (2.05e-4)
	500	6.9882e-1 (6.21e-3) -	5.7010e-1 (6.19e-2) -	4.7799e-3 (2.35e-4) =	5.1157e-3 (3.53e-4) -	**3.9797e-3 (5.62e-5) +**	4.7423e-3 (1.88e-4)
	1000	6.9826e-1 (7.75e-3) -	5.9295e-1 (4.87e-2) -	4.7827e-3 (2.86e-4) =	5.3438e-3 (4.49e-4) -	**3.9974e-3 (7.11e-5) +**	4.7295e-3 (1.73e-4)
	5000	6.9598e-1 (8.11e-3) -	5.8831e-1 (5.52e-2) -	4.8129e-3 (2.07e-4) =	5.0562e-3 (4.25e-4) -	**3.9718e-3 (6.00e-5) +**	4.7431e-3 (2.08e-4)
SMOP5	100	3.6068e-1 (4.45e-3) -	4.1058e-1 (1.32e-2) -	**6.1761e-3 (4.76e-4) =**	7.5766e-3 (4.19e-4) -	1.2312e-2 (1.01e-3) -	6.7307e-3 (5.82e-4)
	500	3.5864e-1 (2.38e-3) -	4.1993e-1 (8.50e-3) -	**5.9443e-3 (3.12e-4) =**	7.7369e-3 (4.45e-4) -	1.1665e-2 (1.02e-3) -	6.4380e-3 (3.26e-4)
	1000	3.5738e-1 (1.93e-3) -	4.2359e-1 (9.65e-3) -	**6.0587e-3 (3.37e-4) =**	9.2720e-3 (4.02e-4) -	1.1451e-2 (1.26e-3) -	6.3126e-3 (3.29e-4)
	5000	3.5883e-1 (2.75e-3) -	4.2221e-1 (6.51e-3) -	**6.1496e-3 (2.66e-4) +**	8.9697e-3 (2.83e-4) -	9.8041e-3 (1.47e-3) -	6.9279e-3 (2.91e-4)
SMOP6	100	7.7093e-2 (4.37e-3) -	1.1600e-1 (1.31e-2) -	**7.6883e-3 (6.87e-4) +**	8.9111e-3 (6.43e-4) -	1.5104e-2 (2.14e-3) -	8.3560e-3 (5.64e-4)
	500	7.3142e-2 (3.79e-3) -	1.3027e-1 (9.84e-3) -	**7.0910e-3 (4.60e-4) =**	8.8662e-3 (6.53e-4) -	1.1155e-2 (1.73e-3) -	7.6330e-3 (5.85e-4)
	1000	7.3759e-2 (2.83e-3) -	1.3694e-1 (1.13e-2) -	**7.2624e-3 (4.50e-4) =**	1.1244e-2 (7.24e-4) -	9.7811e-3 (1.26e-3) -	7.5729e-3 (4.20e-4)
	5000	7.3407e-2 (4.19e-3) -	1.3990e-1 (1.12e-2) -	**7.8506e-3 (3.25e-4) +**	1.3773e-2 (6.57e-4) -	9.0544e-3 (1.35e-3) -	8.6623e-3 (3.16e-4)
SMOP7	100	4.3162e-1 (4.47e-2) -	5.9981e-1 (4.19e-2) -	4.3870e-2 (9.79e-3) -	3.0372e-2 (1.20e-2) -	1.7108e-1 (1.45e-2) -	**1.3880e-2 (6.19e-3)**
	500	4.4663e-1 (3.16e-2) -	6.5323e-1 (2.92e-2) -	6.3094e-2 (1.03e-2) -	1.2050e-1 (1.12e-1) -	1.8754e-1 (7.47e-3) -	**2.6720e-2 (6.49e-3)**
	1000	4.5108e-1 (2.63e-2) -	6.6242e-1 (2.47e-2) -	8.3934e-2 (9.41e-3) -	1.2597e-1 (1.04e-1) -	1.8940e-1 (6.18e-3) -	**4.3220e-2 (6.50e-3)**
	5000	4.4702e-1 (2.11e-2) -	6.7395e-1 (2.78e-2) -	1.2454e-1 (3.83e-3) -	1.2916e-1 (8.11e-2) -	1.8990e-1 (2.91e-3) -	**9.4569e-2 (6.14e-3)**
SMOP8	100	1.9496e+0 (1.53e-1) -	2.3489e+0 (8.84e-2) -	1.6819e-1 (3.60e-2) -	1.8382e-1 (5.25e-2) -	3.8945e-1 (4.12e-2) -	**1.0971e-1 (2.31e-2)**
	500	1.9001e+0 (1.07e-1) -	2.4470e+0 (8.72e-2) -	2.0961e-1 (1.61e-2) -	2.9091e-1 (4.93e-2) -	4.1706e-1 (1.49e-2) -	**1.3829e-1 (1.56e-2)**
	1000	1.9247e+0 (6.11e-2) -	2.4995e+0 (5.43e-2) -	2.4254e-1 (1.42e-2) -	3.7523e-1 (3.09e-2) -	4.1853e-1 (1.03e-2) -	**1.6077e-1 (1.22e-2)**
	5000	1.9284e+0 (5.98e-2) -	2.5274e+0 (8.12e-2) -	3.2666e-1 (1.02e-2) -	4.7832e-1 (9.99e-3) -	4.2540e-1 (1.14e-2) -	**2.4406e-1 (6.00e-3)**
+/-/=		0/32/0	0/32/0	3/20/9	0/32/0	4/28/0	

TABLE III
THE EXPERIMENTAL RESULTS OF HV VALUES OBTAINED BY DKCA AND OTHER COMPARED ALGORITHMS ON BENCHMARK SMOPS, SUMMARIZED AS SUMMATION OF +/-/=.

D	FDV-LMOCSO	IM-MOEA/D	SparseEA	MOEA/PSL	S-ECSO
100	0/8/0	0/8/0	1/5/2	0/8/0	1/7/0
500	0/8/0	0/8/0	2/5/1	0/7/1	1/7/0
1000	0/8/0	0/8/0	0/5/3	0/8/0	1/7/0
5000	0/8/0	0/8/0	2/5/1	0/8/0	1/7/0

of the obtained solutions are equal to zero due to their sparsity handling mechanisms. It is worth noting that almost all the decision variables of the obtained solutions, except for the first hundred ones, are hovering around zero in S-ECSO, which is mainly due to its different encoding strategy. Furthermore, the proposed DKCA has the best performance among these four MOEAs in terms of the sparsity of obtained solutions. In other words, the sparse distribution of solutions obtained by DKCA is most similar to that of Pareto optimal solutions, which shows the superiority of the proposed DKCA on SMOPs.

Moreover, the convergence profiles of IGD values obtained by the proposed DKCA and other compared algorithms on SMOP1, SMOP7, and SMOP8 with 1000 deicision variables are shown in Fig. 6. It can be observed that DKCA has not only a faster convergence speed but also the better IGD values than other algorithms. As a consequence, it can be confirmed that DKCA is superior to other competitors in terms of both effectiveness and efficiency.

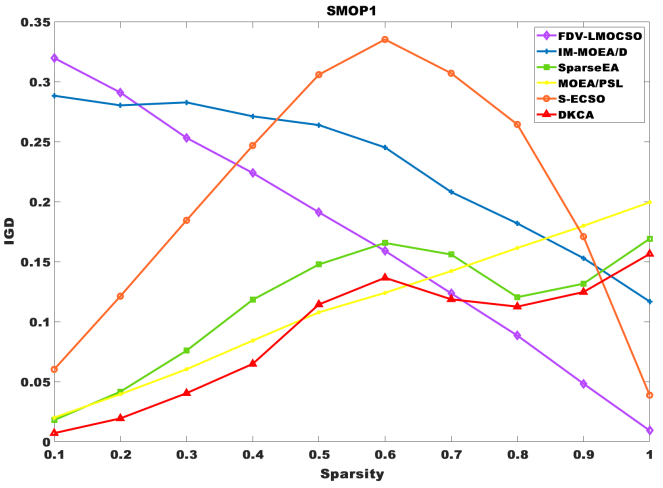

Fig. 7. IGD values obtained by the proposed DKCA and other compared algorithms on SMOP1 with 500 variables, where the sparsity of Pareto optimal solutions is ranged from 0.1 to 1.

In order to investigate the effect of optimal solutions sparsity on the performance of DKCA, the IGD values obtained by DKCA and other compared algorithms on SMOP1 with 500 variables and different levels of sparsity are plotted in Fig. 7. As can be seen, for FDV-LMOCSO and IM-MOEA/D, the performance gets worse

TABLE IV
THE EXPERIMENTAL RESULTS OF DKCA AND OTHER COMPARED ALGORITHMS ON REAL-WORLD SMOPS. THE BEST RESULT IN EACH ROW IS HIGHLIGHTED.

Problem	D	FDV-LMOCSO	IM-MOEA/D	SparseEA	MOEA/PSL	S-ECSO	DKCA
NN1	401	5.3707e-1 (3.62e-2) +	3.3618e-1 (2.19e-2)	9.7063e-1 (1.05e-2) -	9.3985e-1 (2.63e-2) -	9.2153e-1 (2.75e-3) -	**9.7363e-1 (3.59e-3)**
NN2	521	4.2476e-1 (3.04e-2) +	2.9030e-1 (1.70e-2)	8.0833e-1 (3.85e-3) =	7.9422e-1 (1.04e-2) -	7.9756e-1 (1.21e-2) -	**8.0876e-1 (3.09e-3)**
NN3	1241	4.3061e-1 (3.17e-2) +	3.0936e-1 (1.79e-2)	**8.6638e-1 (1.34e-2) =**	8.4424e-1 (1.58e-2) -	8.6570e-1 (2.37e-2) =	8.6251e-1 (9.50e-3)
PO1	100	1.1999e-1 (1.02e-3) -	1.2150e-1 (6.05e-4)	1.2376e-1 (1.52e-3) -	1.2358e-1 (1.32e-3) -	**1.2494e-1 (5.24e-6) =**	1.2451e-1 (6.76e-4)
PO2	500	9.5234e-2 (5.62e-4) -	9.6762e-2 (9.62e-4)	1.2381e-1 (1.28e-3) -	1.0942e-1 (1.11e-2) -	1.2325e-1 (2.39e-3) -	**1.2392e-1 (1.24e-3)**
PO3	1000	9.2277e-2 (1.81e-4) =	9.2218e-2 (1.28e-4)	1.2384e-1 (1.32e-3) =	1.1023e-1 (1.30e-2) -	1.2202e-1 (3.60e-3) -	**1.2389e-1 (1.12e-3)**
SR1	128	3.3452e-1 (3.23e-2) +	3.2339e-1 (2.62e-2)	4.0748e-1 (3.82e-3) -	3.8829e-1 (6.04e-3) -	3.5813e-1 (1.15e-2) -	**4.0765e-1 (2.92e-3)**
SR2	512	2.6959e-1 (2.16e-2) +	2.5508e-1 (1.74e-2)	3.8789e-1 (5.55e-3) -	3.1683e-1 (1.97e-2) -	2.2781e-1 (1.07e-2) -	**3.8903e-1 (7.67e-3)**
SR3	1024	2.3108e-1 (1.41e-2) +	2.1604e-1 (9.92e-3)	3.6036e-1 (1.14e-2) -	2.6195e-1 (2.16e-2) -	1.7164e-1 (7.47e-3) -	**3.6306e-1 (4.93e-3)**
+/-/=		0/9/0	0/9/0	0/6/3	0/9/0	0/7/2	

TABLE V
IGD VALUES OBTAINED BY DKCA', DKCA", AND DKCA ON SMOP2, SMOP7, AND SMOP8 WITH 100, 500, AND 1000 VARIABLES, RESPECTIVELY. THE BEST RESULT IN EACH ROW IS HIGHLIGHTED.

Problem	D	DKCA'	DKCA"	DKCA
SMOP2	100	3.0518e-2 (5.96e-3) -	1.2674e-2 (5.25e-3) -	**1.2076e-2 (4.59e-3)**
	500	5.0090e-2 (6.31e-3) -	2.3046e-2 (5.12e-3) =	**2.1569e-2 (4.96e-3)**
	1000	6.8153e-2 (5.64e-3) -	3.3859e-2 (4.60e-3) -	**3.2753e-2 (4.36e-3)**
SMOP7	100	3.9222e-2 (8.71e-3) -	1.4590e-2 (6.70e-3) -	**1.3880e-2 (6.19e-3)**
	500	6.5352e-2 (9.47e-3) -	2.8192e-2 (7.01e-3) -	**2.6720e-2 (6.49e-3)**
	1000	8.6624e-2 (7.15e-3) -	4.4263e-2 (7.33e-3) -	**4.3220e-2 (6.50e-3)**
SMOP8	100	1.6351e-1 (3.23e-2) -	1.0977e-1 (2.21e-2) =	**1.0971e-1 (2.31e-2)**
	500	2.0675e-1 (1.91e-2) -	1.4585e-1 (2.19e-2) -	**1.3829e-1 (1.56e-2)**
	1000	2.3991e-1 (1.30e-2) -	1.6361e-1 (1.91e-2) =	**1.6077e-1 (1.22e-2)**
+/-/=		0/9/0	0/6/3	

as the sparsity decreases, which is mainly due to the lack of techniques designed for SMOPs. For S-ECSO, it can achieve relatively good performance when the sparsity is very high or very low. However, it has bad performance in other cases. For SparseEA, MOEA/PSL, and DKCA, their performance gets better with the decrease of the sparsity, and DKCA achieves the best performance in most cases among these three MOEAs. To summarize, the proposed DKCA can obtain the best performance when the sparsity of optimal solutions is less than 0.7, which demonstrates its superior effectiveness on SMOPs.

C. Experimental Results on Real-world SMOPs

In this section, the proposed algorithm is tested on three real-world SMOPs with the comparison of other five MOEAs. Table IV shows the HV values obtained by DKCA and other compared algorithms. Note that DKCA gets 8 best results of 10 test instances; both SparseEA and S-ECSO get 1 best result; FDV-LMOCSO, IM-MOEA/D, and MOEA/PSL do not get the best result. Moreover, DKCA has obviously better performance than FDV-LMOCSO, IM-MOEA/D, SparseEA, MOEA/PSL, and S-ECSO on 9, 9, 6, 9 and 7 test problems, respectively.

To show the superiority of DKCA more visually, the populations with median HV values obtained by FDV-LMOCSO, IM-MOEA/D, SparseEA, MOEA/PSL, S-ECSO, and the proposed DKCA on NN2, PO2, and SR2 with approximately 500 decision variables are shown in Fig. 8. As can be seen, for three real-world SMOPs, the solutions obtained by DKCA, SparseEA, and MOEA/PSL are obviously better than those obtained by FDV-LMOCSO, IM-MOEA/D, and S-ECSO in both convergence and diversity. Furthermore, for SparseEA, the obtained solutions are usually not well-converged. For MOEA/PSL, the obtained solutions can not distribute uniformly on the PF. As for DKCA, the obtained solutions are superior to those obtained by other MOEAs in both convergence and diversity. Hence, the proposed DKCA is more promising than the existing MOEAs to tackle the real-world SMOPs.

D. Effectiveness of the Primary Components in DKCA

In this section, DKCA is compared with its two variants DKCA' and DKCA" to verify the effectiveness of its core components. Specifically, in DKCA', the scores of decision variables are still updated dynamically during the evolution, but only one population is evolved, which aims to investigate the effectiveness of the adopted co-evolutionary framework. In DKCA", the coevolutionary framework of OP and RP is adopted with the lack of dynamic score update mechanism. In other words, the scores of decision variables are fixed before the evolution so that the effectiveness of the embedded score update strategy can be investigated.

Table V depicts the IGD values obtained by DKCA', DKCA" and DKCA on SMOP2, SMOP7, and SMOP8 with 100, 500, and 1000 variables, respectively. It is obviously that DKCA outperforms its two variants on all the three SMOPs at the different scales of decision variables. In order to further investigate the effectiveness of the primary components during the whole evolutionary process, the convergence profiles of IGD values obtained by DKCA and its variants on SMOP2, SMOP7, and SMOP8 with 500 variables are shown in Fig. 9. As can be seen, DKCA exhibits significantly better than DKCA', which means the proposed coevolutionary framework brings a huge improvement comparing with the traditional evolutionary strategy with single population. Moreover, DKCA has more promising performance than DKCA", which demonstrates the significant variables can be selected more precisely based on the embedded dynamic score update mechanism.

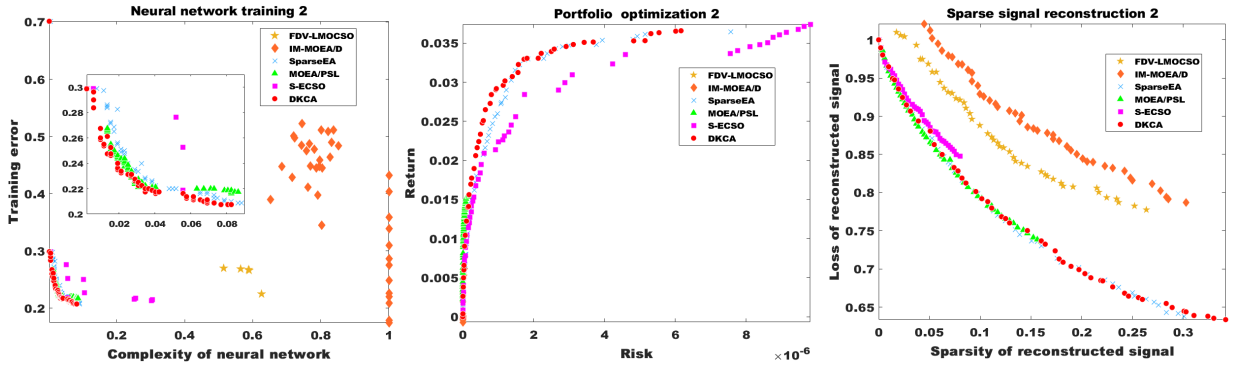

Fig. 8. Populations with median HV values obtained by FDV-LMOCSO, IM-MOEA/D, SparseEA, MOEA/PSL, S-ECSO, and the proposed DKCA on NN2, PO2, and SR2 with approximately 500 decision variables.

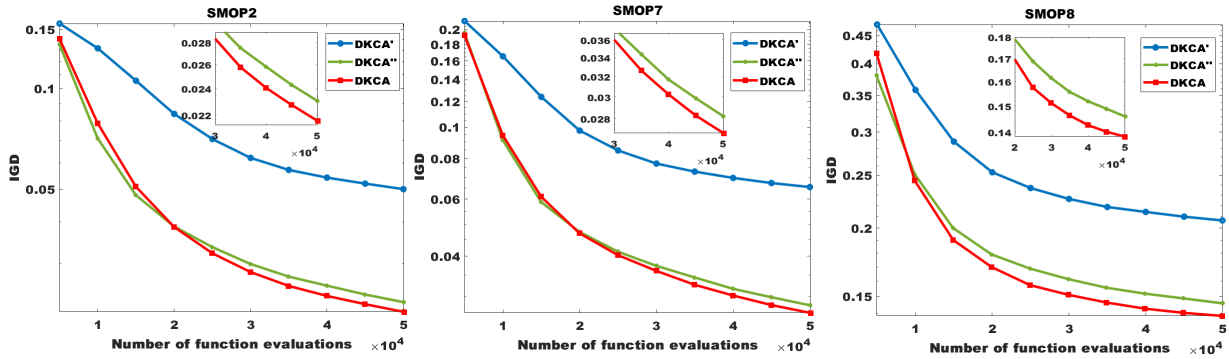

Fig. 9. Convergence profiles of IGD values obtained by DKCA and its variants on SMOP2, SMOP7, and SMOP8 with 500 variables.

TABLE VI
AVERAGE RUNTIMES (IN SECONDS) OF FDV-LMOCSO, IM-MOEA/D, SPARSEEA, MOEA/PSL, S-ECSO, AND DKCA ON EIGHT BENCHMARK SMOPS WITH 1000 DECISION VARIABLES AND THREE REAL-WORLD APPLICATIONS (NN3, PO3, AND SR3). THE LEAST RUNTIME IN EACH ROW IS HIGHLIGHTED.

Problem	D	FDV-LMOCSO	IM-MOEA/D	SparseEA	MOEA/PSL	S-ECSO	DKCA
SMOP1-SMOP8(average)	1000	**2.314e+1**	4..160e+1	7.902e+1	3.354e+2	4.714e+1	1.981e+2
NN3	1241	2.8380e+1	2.8824e+1	7.117e+1	5.035e+1	**1.519e+1**	3.544e+1
PO3	1000	5.3015e+1	**5.1406e+1**	5.159e+1	9.898e+1	5.484e+1	6.810e+1
SR3	1024	3.3694e+1	2.9755e+1	3.711e+1	4.975e+1	**2.211e+1**	3.856e+1

E. Computational efficiency of DKCA

In this section, the computational efficiency of DKCA is tested with the comparison of other five MOEAs, and the average runtimes (in seconds) of FDV-LMOCSO, IM-MOEA/D, SparseEA, MOEA/PSL, S-ECSO, and DKCA on eight benchmark test problems and three real-world test problems (NN3, PO3, and SR3) are listed in Table VI. As can be seen, for MOEAs tailored for large-scale SMOPs, the DKCA shows significantly better computational efficiency than MOEA/PSL on both benchmark test instances and real-world test instances, and its runtime is slightly longer than that of SparseEA and S-ECSO due to the embedded cooperative coevolutionary framework. For MOEAs tailored for LMOPs, they consume less runtime but obtain poorer results since the sparsity of the Pareto optimal solutions is ignored. In summary, the proposed DKCA is able to obtain the significantly better performance with a relatively low time consumption.

V. CONCLUSIONS

The sparsity knowledge of Pareto optimal solutions utilized in the existing MOEAs is usually static or even ignored, which leads to the challenges for solving large-scale SMOPs. Based on this consideration, a dynamic knowledge-guided coevolutionary algorithm is proposed for solving large-scale SMOPs in this paper. To be specific, two populations are initialized and coevolved in the proposed DKCA. One is evolved in the original decision space, while the other is evolved in the reduced decision space, and the cooperation of these two populations are performed after the offspring generation. Moreover, a dynamic score update mechanism is adopted to adjust the scores of decision variables dynamically during the whole evolutionary process. To the best of our knowledge, the

proposed DKCA is the first MOEA to solve large-scale SMOPs with the cooperative coevolutionary framework.

Experimentally, to investigate the effectiveness of the proposed algorithm, DKCA is tested on a series of benchmark SMOPs and real-world SMOPs with the comparison of five state-of-the-art MOEAs. According to the experimental results, the proposed DKCA is obviously superior to other competitors, which shows the great potential of the CAs in tackling large-scale SMOPs.

In this paper, the promising future of the cooperative coevolutionary framework has been shown in solving large-scale SMOPs. Therefore, more MOEAs based on the coevolutionary framework are highly desirable to tackle SMOPs. Besides, optimizers with better performance can be equipped in DKCA to achieve better performance. Finally, more effective strategies can be designed to make use of sparsity knowledge dynamically during the evolutionary process.

References

[1] B. Xue, M. Zhang, and W. N. Browne, "Particle Swarm Optimization for Feature Selection in Classification: A Multi-Objective Approach," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656–1671, 2013.

[2] Y. Jin and B. Sendhoff, "Pareto-based multiobjective machine learning: An overview and case studies," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 3, pp. 397–415, 2008.

[3] H. A. Abbass, "Pareto neuro-evolution: Constructing ensemble of neural networks using multi-objective optimization," in *The 2003 Congress on Evolutionary Computation*, vol. 3, 2003, pp. 2074–2080.

[4] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[5] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by v1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, 1997.

[6] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[7] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, "A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 321–344, 2012.

[8] L. Tang, L. Zhang, P. Luo, and M. Wang, "Incorporating occupancy into frequent pattern mining for high quality pattern recommendation," in *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, 2012, pp. 75–84.

[9] C. Qian, Y. Yu, and Z.-H. Zhou, "Subset selection by pareto optimization," *Advances in neural information processing systems*, vol. 28, 2015.

[10] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, 2015.

[11] 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, 2016.

[12] 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, 2017.

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

[14] H. Hiba, A. A. Bidgoli, A. Ibrahim, and S. Rahnamayan, "Cgde3: An efficient center-based algorithm for solving large-scale multi-objective optimization problems," in *2019 IEEE Congress on Evolutionary Computation*, 2019, pp. 350–358.

[15] 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, 2019.

[16] S. Liu, Q. Lin, Q. Li, and K. C. Tan, "A comprehensive competitive swarm optimizer for large-scale multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[17] R. Cheng and Y. Jin, "A competitive swarm optimizer for large scale optimization," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 191–204, 2014.

[18] 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.

[19] 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.

[20] A. Fischer and C. Igel, "An introduction to restricted boltzmann machines," in *Iberoamerican Congress on Pattern Recognition*, 2012, pp. 14–36.

[21] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, 2008, pp. 1096–1103.

[22] Y. Zhang, Y. Tian, and X. Zhang, "Improved sparseea for sparse large-scale multi-objective optimization problems," *Complex & Intelligent Systems*, pp. 1–16, 2021.

[23] Z. Tan, H. Wang, and S. Liu, "Multi-stage dimension reduction for expensive sparse multi-objective optimization problems," *Neurocomputing*, vol. 440, pp. 159–174, 2021.

[24] X. Wang, K. Zhang, J. Wang, and Y. Jin, "An enhanced competitive swarm optimizer with strongly convex sparse operator for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 859–871, 2021.

[25] Y. Tian, C. Lu, X. Zhang, F. Cheng, and Y. Jin, "A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 6784–6797, 2020.

[26] Y. Xue, B. Xue, and M. Zhang, "Self-adaptive particle swarm optimization for large-scale feature selection in classification," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 5, pp. 1–27, 2019.

[27] F. Cheng, F. Chu, Y. Xu, and L. Zhang, "A steering-matrix-based multiobjective evolutionary algorithm for high-dimensional feature selection," *IEEE transactions on cybernetics*, vol. 52, no. 9, pp. 9695–9708, 2021.

[28] B. Tran, B. Xue, and M. Zhang, "Improved pso for feature selection on high-dimensional datasets," in *Simulated Evolution and Learning: 10th International Conference, SEAL 2014, Dunedin, New Zealand, December 15-18, 2014. Proceedings 10*. Springer, 2014, pp. 503–515.

[29] P. Moradi and M. Gholampour, "A hybrid particle swarm optimization for feature subset selection by integrating a novel local search strategy," *Applied Soft Computing*, vol. 43, pp. 117–130, 2016.

[30] K. Chen, B. Xue, M. Zhang, and F. Zhou, "An evolutionary multitasking-based feature selection method for high-dimensional classification," *IEEE Transactions on Cybernetics*, vol. 52, no. 7, pp. 7172–7186, 2020.

[31] S. Liu, H. Wang, W. Peng, and W. Yao, "A surrogate-assisted evolutionary feature selection algorithm with parallel random grouping for high-dimensional classification," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 1087–1101, 2022.

[32] L. M. Antonio and C. A. C. Coello, "Coevolutionary multiobjective evolutionary algorithms: Survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 6, pp. 851–865, 2017.

[33] Y. Tian, T. Zhang, J. Xiao, X. Zhang, and Y. Jin, "A coevolutionary framework for constrained multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 1, pp. 102–116, 2020.

[34] J. Wang, Y. Li, Q. Zhang, Z. Zhang, and S. Gao, "Cooperative multiobjective evolutionary algorithm with propulsive population for constrained multiobjective optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 52, no. 6, pp. 3476–3491, 2021.

[35] R. Wang, W. Ma, M. Tan, G. Wu, L. Wang, D. Gong, and J. Xiong, "Preference-inspired coevolutionary algorithm with active diversity strategy for multi-objective multi-modal optimization," *Information Sciences*, vol. 546, pp. 1148–1165, 2021.

[36] Z. Yang, Y. Jin, and K. Hao, "A bio-inspired self-learning coevolutionary dynamic multiobjective optimization algorithm for internet of things services," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 675–688, 2018.

[37] X.-F. Liu, Z.-H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, "Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 587–602, 2018.

[38] Z. Yang, K. Tang, and X. Yao, "Large scale evolutionary optimization using cooperative coevolution," *Information Sciences*, vol. 178, no. 15, pp. 2985–2999, 2008.

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

[40] S. Kukkonen and J. Lampinen, "Gde3: The third evolution step of generalized differential evolution," in *2005 IEEE Congress on Evolutionary Computation*, vol. 1, 2005, pp. 443–450.

[41] F. Sander, H. Zille, and S. Mostaghim, "Transfer strategies from single-to multi-objective grouping mechanisms," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 729–736.

[42] 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.

[43] 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.

[44] K. Deb, R. B. Agrawal *et al.*, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 2, pp. 115–148, 1995.

[45] K. Deb and M. Goyal, "A combined genetic adaptive search for engineering design," *Computer Science and Informatics*, vol. 26, no. 4, pp. 30–45, 1996.

[46] 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," *Technical Report, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China, Report# 2018001*, 2018.

[47] Y. Tian, R. Cheng, X. Zhang, and Y. Jin, "PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 73–87, 2017.

[48] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[49] L. While, P. Hingston, L. Barone, and S. Huband, "A faster algorithm for calculating hypervolume," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 29–38, 2006.

[50] Y. Tian, X. Xiang, X. Zhang, R. Cheng, and Y. Jin, "Sampling reference points on the pareto fronts of benchmark multi-objective optimization problems," in *2018 IEEE Congress on Evolutionary Computation*, 2018, pp. 1–6.

[51] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[52] X. Yang, J. Zou, S. Yang, J. Zheng, and Y. Liu, "A fuzzy decision variables framework for large-scale multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, 2021.

[53] L. R. Farias and A. F. Araújo, "Im-moea/d: An inverse modeling multi-objective evolutionary algorithm based on decomposition," in *2021 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2021, pp. 462–467.

[54] M. Li, L. Zhen, and X. Yao, "How to read many-objective solution sets in parallel coordinates," *IEEE Computational Intelligence Magazine*, vol. 12, no. 4, pp. 88–100, 2017.