A Fast Clustering Based Evolutionary Algorithm for Super-Large-Scale Sparse Multi-Objective Optimization

Ye Tian, Yuandong Feng, Xingyi Zhang, *Senior Member, IEEE*, and Changyin Sun

***Abstract*—During the last three decades, evolutionary algorithms (EAs) have shown superiority in solving complex optimization problems, especially those with multiple objectives and non-differentiable landscapes. However, due to the stochastic search strategies, the performance of most EAs deteriorates drastically when handling a large number of decision variables. To tackle the curse of dimensionality, this work proposes an efficient EA for solving super-large-scale multi-objective optimization problems with sparse optimal solutions. The proposed algorithm estimates the sparse distribution of optimal solutions by optimizing a binary vector for each solution, and provides a fast clustering method to highly reduce the dimensionality of the search space. More importantly, all the operations related to the decision variables only contain several matrix calculations, which can be directly accelerated by GPUs. While existing EAs are capable of handling fewer than 10 000 real variables, the proposed algorithm is verified to be effective in handling 1 000 000 real variables. Furthermore, since the proposed algorithm handles the large number of variables via accelerated matrix calculations, its runtime can be reduced to less than 10% of the runtime of existing EAs.**

***Index Terms*—Evolutionary computation, fast clustering, sparse multi-objective optimization, super-large-scale optimization.**

I. Introduction

MANY scientific and engineering fields such as artificial intelligence [1], data mining [2], software engineering [3], bioinformatics [4], and economics [5] include complex optimization problems with multiple conflicting objectives and a large number of decision variables, which are collectively known as large-scale multi-objective optimization problems (LMOPs). These problems are generally NP-hard with complicated landscapes, and have global optima which are hard to obtain by exact methods; by contrast, multi-objective evolutionary algorithms (MOEAs) can find quasi-optimal solutions for LMOPs in polynomial time [6].

Since the first MOEA was suggested for solving LMOPs in 2013 [7], a number of MOEAs have been proposed to handle the high-dimensional search space using various techniques, including decision variable grouping, decision variable analysis, and decision space reduction. The decision variable grouping based MOEAs randomly divide the decision variables into several groups and optimize each group of decision variables alternately [7], [8], so that the LMOP can be split into small-scale problems and solved easily. Since the random grouping strategy may divide two interacting decision variables into different groups and drive the population into local optima, the decision variable analysis based MOEAs divide the decision variables according to their correlations to the other decision variables and the objective functions [9], [10], which can improve both population diversity and the probability of finding global optima. The decision space reduction based MOEAs facilitate the solving of LMOPs by reducing the dimensions of the decision space, with the assistance of problem transformation [11] and dimensionality reduction [12] techniques.

While conventional MOEAs are effective for problems with less than 100 variables [13], the MOEAs tailored for LMOPs have shown promising performance on problems with 1000 to 10 000 variables [10], [12], [14]. Nevertheless, they are not applicable to the problems with much more variables, which are termed super-large-scale multi-objective optimization problems (SLMOPs) in this work. SLMOPs widely exist in many research fields, such as large-scale feature selection tasks with about 45 000 candidate features [15], deep neural network training tasks with more than 150 000 weights [16], and time-varying ratio error estimation tasks with up to 300 000 variables [14]. For decision variable grouping based MOEAs, if the 300 000 decision variables of an SLMOP are randomly divided into 100 groups, each group will contain 3000 decision variables that still form an LMOP; if the decision variables are divided into many more groups, the convergence speed will highly deteriorate as the large number of

Manuscript received August 19, 2021; revised September 3, 2021; accepted September 30, 2021. This work was supported in part by the National Key Research and Development Program of China (2018AAA0100100), the National Natural Science Foundation of China (61822301, 61876123, 61906001), the Collaborative Innovation Program of Universities in Anhui Province (GXXT-2020-051), the Hong Kong Scholars Program (XJ2019035), and Anhui Provincial Natural Science Foundation (1908085QF271). Recommended by Associate Editor Shangce Gao. *(Corresponding author: Xingyi Zhang.)*

Citation: Y. Tian, Y. D. Feng, X. Y. Zhang, and C. Y. Sun, "A fast clustering based evolutionary algorithm for super-large-scale sparse multi-objective optimization," *IEEE/CAA J. Autom. Sinica*, vol. 10, no. 4, pp. 1048–1063, Apr. 2023.

Y. 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 (e-mail: field910921@gmail.com).

Y. D. Feng is with the School of Computer Science and Technology, Anhui University, Hefei 230601, China (e-mail: yuandongfeng@stu.ahu.edu.cn).

X. Y. Zhang is with the School of Artificial Intelligence, Anhui University, Hefei 230601, China (e-mail: xyzhanghust@gmail.com).

C. Y. Sun is with the School of Automation, Southeast University, Nanjing 210096, China (e-mail: cysun@seu.edu.cn).

Digital Object Identifier 10.1109/JAS.2022.105437

groups need to be optimized in sequence. For decision variable analysis based MOEAs, because at least four function evaluations are required to detect whether two decision variables interact, 1.8×10^{11} function evaluations should be consumed to detect interactions between all the 300 000 decision variables, which is impractical for real-world applications. As for decision space reduction based MOEAs, it is difficult to reduce the 300 000-dimensional decision space using only hundreds of solutions without losing the optimal regions.

More seriously, most operations in existing MOEAs are too complex to be parallelized and accelerated by hardware devices, whose computational complexities increase considerably with the number of decision variables. On the one hand, the reproduction operators (e.g., simulated binary crossover (SBX) [17] and polynomial mutation [18]) used in most MOEAs have multiple options with different probabilities to be selected, where the generation of each decision variable of each offspring is performed in different branches. On the other hand, for decision variable grouping and decision variable analysis based MOEAs, the decision variables in different groups can only be optimized in sequence; while for decision space reduction based MOEAs, it is very time-consuming to calculate the correlations between the massive decision variables for dimensionality reduction.

As a consequence, for an LMOP with more decision variables, existing MOEAs require more function evaluations to approximate the global optima, and the generation of a single solution is also more time-consuming. To improve the efficiency in solving SLMOPs, this paper aims to develop an MOEA for reducing both the number of function evaluations and the complexity of solution generation. We found that the optimal solutions of many SLMOPs contain a large number of zero variables, such as feature selection for selecting a small proportion of candidate features [19] and neural network training for finding a sparse architecture [1]. Hence, we focus on handling such sparse SLMOPs due to their prevalence in real-world applications. Specifically, this paper contains the following contributions:

1) To find sparse optimal solutions more efficiently, a fast clustering method is proposed to divide the large number of decision variables into multiple groups. For all the variables in the same group, a single variable is used to represent them so that they are optimized, hence the search space is highly reduced and the convergence speed can be improved. In this way, sparse optimal solutions can be approximated by using fewer function evaluations.

2) Based on the fast clustering method, an evolutionary algorithm is designed for solving sparse SLMOPs, termed super-large-scale multi-objective evolutionary algorithm (SLMEA). In contrast to complicated operations like decision variable division and decision space reduction, all the operations related to the decision variables in SLMEA are converted into matrix calculations, which can be easily accelerated by GPUs to reduce the complexity of solution generation.

3) To verify the efficiency of the proposed algorithm, it is tested on a variety of benchmark problems and real-world problems with up to one million decision variables. In comparison to several state-of-the-art MOEAs tailored for LMOPs, the proposed algorithm can exhibit significantly better performance on most test instances. Moreover, with the assistance of GPU acceleration, the runtime of the proposed algorithm is less than 10% of the runtime of the other MOEAs.

The rest of this paper is organized as follows. In Section II, existing MOEAs for solving LMOPs and techniques for efficiency improvement are reviewed, which is followed by the motivation of this work. In Section III, the proposed algorithm and fast clustering method are elaborated upon. In Section IV, the experimental results are presented and analyzed. Lastly, conclusions are drawn in Section V.

II. Related Work and Motivation

A. Large-Scale MOEAs

A multi-objective optimization problem is defined as

$$\begin{aligned}&\min_{\boldsymbol{x}}\ \boldsymbol{f}(\boldsymbol{x})=(f_1(\boldsymbol{x}),\ldots,f_M(\boldsymbol{x}))\\&\text{s.t. }\ \boldsymbol{x}=(x_1,\ldots,x_D)\in\Omega\end{aligned}\tag{1}$$

where $\boldsymbol{x}$ denotes a decision vector having D variables and $\boldsymbol{f}$ denotes an objective vector having $M\geq2$ functions. In general, a problem with $D\geq100$ real variables is called an LMOP [20], and a problem with $D\geq10\,000$ real variables is called an SLMOP in this work. Besides, sparse SLMOPs denote those having sparse optimal solutions, i.e., most variables in these solutions are zero. For most metaheuristics including MOEAs, they do not need other information about these problems (e.g., gradients) due to their stochastic search strategies. Accordingly, many more function evaluations should be consumed to solve a problem with more variables, which greatly hinders the search efficiency of metaheuristics [10]. In order to alleviate the curse of dimensionality, decision variable grouping, decision variable analysis, and decision space reduction techniques have been developed to tackle LMOPs [21], [22].

A naive idea for solving LMOPs is to adopt the divide-and-conquer strategy. By randomly dividing the decision variables into a predefined number of groups with equal size, CCGDE3 [7] alternately optimizes each group of decision variables and fixes the rest. Similarly, a dynamic grouping strategy is suggested in MOEA/D-RDG [23] to adjust the group size. However, the random division of decision variables does not consider the interactions between decision variables and thus is likely to trap the population into local optima. Hence, some MOEAs adopt the differential grouping strategy to detect interactions between decision variables, where the variables interacting with each other are assigned to the same group [24], [25]. This way, the global optimal solutions will not be missed when optimizing each group of interacting variables in sequence. While the differential grouping strategy only considers the convergence of the population, the decision variable analysis strategies aim to enhance both convergence and diversity performance. As one of the first deci-

sion variable analysis based MOEAs, MOEA/DVA [9] divides decision variables into position variables, distance variables, and mixed variables by analyzing their control properties. It optimizes position variables via differential grouping until the population converges, and then tunes the distance and mixed variables for diversity enhancement. For more robust search performance, LMEA [10] uses a decision variable clustering strategy to divide the decision variables into convergence-related variables and diversity-related variables, then alternately optimizes the two types of decision variables via different strategies for the improvement of both convergence and diversity. PEA [26] adopts a similar idea to LMEA, where the optimization of convergence-related variables is parallelized. On the other hand, decision space reduction strategies aim to directly reduce the number of decision variables without division. WOF [11] suggests a problem transformation framework to convert an LMOP into a small-scale optimization problem, where the decision variables are divided into multiple groups and a single weight is optimized instead of all the variables in each group; thus the decision space can be naturally reduced. LSMOF [27] suggests a problem reformulation framework, which optimizes only two weights for each solution along two search directions for better quality of the whole population. ReMO [28] adopts the random embedding technique to handle LMOPs with low intrinsic dimensions, where the objectives are affected by only a small proportion of all decision variables. In addition, DLS-MOEA [29] uses a self-evaluation evolution based dual local search mechanism, which generates offspring by Gaussian or Cauchy mutation and evaluates offspring by a meta-model.

In spite of the promising performance of existing large-scale MOEAs, they are ineffective for solving SLMOPs as discussed in Section I. For decision variable grouping and decision variable analysis based MOEAs, they need to optimize many groups of variables in sequence; besides, the function evaluations consumed by differential grouping, decision variable analysis, and decision variable clustering are unaffordable [10]. Although decision space reduction based MOEAs do not require a large number of function evaluations, they are likely to miss global optimal solutions since the population cannot provide sufficient samples to learn an optimal subspace in the high-dimensional decision space [30]. On the other hand, existing MOEAs are also inefficient in solving SLMOPs, since most of the decision variable grouping, decision variable analysis, and decision space reduction techniques require a large number of iterations that cannot be easily parallelized.

B. Sparse MOEAs

Many LMOPs in real-world applications have sparse optimal solutions, such as neural architecture search [31], community detection [32], pattern mining [33], and power grid fault diagnosis [34], hence some MOEAs have been tailored for solving such LMOPs in recent years. As the first generic MOEA for finding sparse optimal solutions, SparseEA [35] suggests a bi-level encoding to find the zero decision variables in optimal solutions efficiently. By taking advantage of the sparse nature of LMOPs, SparseEA can approximate sparse optimal solutions by using fewer function evaluations than other large-scale MOEAs. In MOEA/PSL [12], a restricted Boltzmann machine and a denoising autoencoder are adopted to learn a sparse distribution and a compact representation of the optimal solutions, which enable the algorithm to find large-scale sparse solutions in the learnt subspace. In PM-MOEA [30], an evolutionary pattern mining approach is developed to mine the sparse distribution of the optimal solutions, which can provide different small-scale subspaces to be exploited by the algorithm.

Although the above sparse MOEAs significantly improve efficiency in finding sparse optimal solutions, they are still inefficient in finding sparse optimal solutions of SLMOPs. This is mainly due to the high complexities of the strategies for generating sparse solutions, where the non-dominated sorting based fitness calculation in SparseEA, the neural network based dimensionality reduction in MOEA/PSL, and the evolutionary pattern mining in PM-MOEA hold a time complexity of $O(D^2)$, $O(NDEK)$, and $O(NDN'G')$, respectively, where D is the number of decision variables, N is the population size, E is the number of epochs for training neural networks, K is the hidden layer size of the neural networks, N' is the population size for pattern mining, and G' is the number of generations for pattern mining. In addition, SparseEA requires a space complexity of $O(D^2)$ for fitness calculation, MOEA/PSL requires a space complexity of $O(DK)$ to store neural networks, and PM-MOEA requires a space complexity of $O(DN')$ to store the mined sparse distributions. On the contrary, a much simpler but still effective search strategy is suggested in the proposed algorithm for finding sparse optimal solutions, whose time complexity is as low as $O(ND)$ and space complexity is just $O(D)$. Moreover, the search strategies in SparseEA, MOEA/PSL, and PM-MOEA contain iterated steps while the proposed search strategy consists of only matrix calculations, which can be deployed on GPUs for a further improvement of efficiency.

C. Parallel MOEAs

In order to improve search efficiency, some work has been dedicated to the parallelization of MOEAs in terms of the algorithmic level, iteration level, and solution level [36]. The algorithmic level parallelization employs master-slave models [37], [38] or island models [39], [40] to optimize multiple populations simultaneously, where the populations evolve toward different parts of the Pareto front and share useful information with each other via migration. The iteration level parallelization is performed on the generation and evaluation of solutions in a single population [41], [42], which can easily accelerate existing MOEAs like NSGA-II [43] and MOEA/D [44]. The solution level parallelization is for the evaluation of a single solution, where problem-dependent strategies are developed to accelerate calculation of objective functions [45], [46].

Although these parallel MOEAs are efficient for solving

small-scale optimization problems, they are not applicable to large-scale optimization since the operations related to the decision variables (e.g., reproduction operators) are not parallelized. Moreover, their frameworks are incompatible with existing large-scale MOEAs and sparse MOEAs, which means that these parallel MOEAs are ineffective for solving LMOPs as well as SLMOPs.

D. Motivation of This Work

According to the above analysis, it can be determined that existing large-scale MOEAs and sparse MOEAs are inefficient to solve SLMOPs, and existing parallel MOEAs are ineffective in solving SLMOPs. Besides, it is difficult to take advantage of all three types of MOEAs by combing their strategies. In order to better handle the sparse SLMOPs that widely exist in the real world, the motivation of this work is to develop new techniques for the improvement of both effectiveness and efficiency. New techniques should take much fewer function evaluations compared to existing MOEAs for finding sparse optimal solutions for SLMOPs, and they should be able to be parallelized and accelerated. It is worth noting that although some evolutionary algorithms have already been developed for solving specific problems with millions [16] or even billions of variables [47], the core techniques in these algorithms are heavily customized for specific objective functions and datasets, which cannot be used to solve different SLMOPs.

On the contrary, this work proposes a problem-independent MOEA for solving sparse SLMOPs. By suggesting a fast clustering based search strategy, the proposed algorithm can use a relatively small number of function evaluations to approximate the sparse optimal solutions of SLMOPs. More importantly, the main operations in the proposed algorithm consist of only matrix calculations rather than iterated steps, which can be deployed on GPUs for a significant improvement of efficiency. The detailed procedure of the proposed algorithm is presented in the next section.

III. The Proposed Algorithm

A. General Procedure of SLMEA

The general procedure of the proposed SLMEA is shown in Fig. 1, where the mating selection, offspring generation, and environmental selection are iteratively performed until the termination criterion is satisfied. By following the bi-level encoding in SparseEA, the proposed SLMEA represents each solution $\boldsymbol{x} = (x_1, \ldots, x_D) \in \Omega$ by a real vector $\boldsymbol{r} = (r_1, \ldots, r_D) \in \Omega$ and a binary vector $\boldsymbol{b} = (b_1, \ldots, b_D) \in \{0,1\}^D$, where each decision variable x_i is determined by $x_i = r_i \times b_i$. In this way, the optimization of the real vector $\boldsymbol{r}$ can find the optimal decision vectors and the optimization of the binary vector $\boldsymbol{b}$ can determine the real variables that should be optimized. Owing to the multiplication of real variables r_i and binary variables b_i, all the r_i whose corresponding $b_i = 0$ do not need to be optimized, and thus the decision space Ω can be highly reduced. While the optimization of the binary vector $\boldsymbol{b}$ additionally introduces a D-dimensional binary search space, a fast clustering method is suggested for dimensionality reduction.

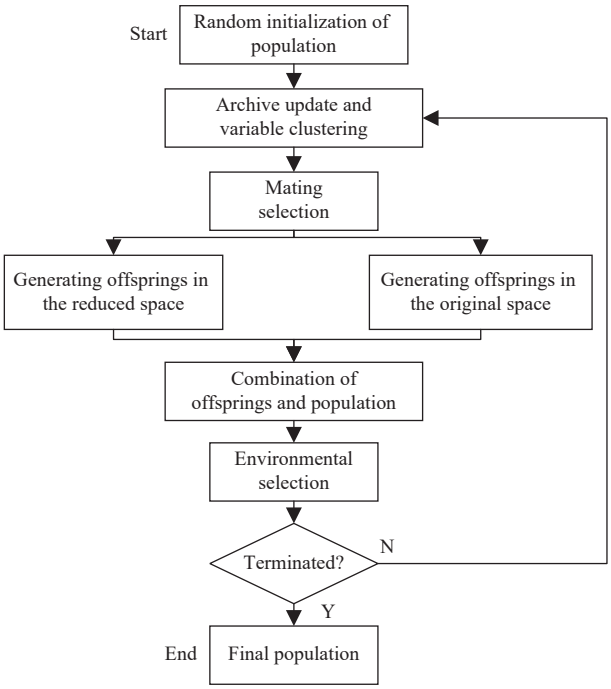

Fig. 1. Procedure of SLMEA, where the core contributions are shown in dark color.

As depicted in Fig. 1, the core contributions of SLMEA lie in two aspects, i.e., updating an archive for binary variable clustering, and generating offspring in the reduced search space.

The pseudocode of the main procedure of SLMEA is given in Algorithm 1. To begin with, a population P with size N and an empty archive A are initialized (Lines 1 and 2), and the initial values of the parameters K and ρ are set to 5 and 0.5 (Lines 3 and 4), respectively. To initialize solutions with different sparsity, for each solution $\boldsymbol{x}$ in the initial population, SLMEA sets its real vector $\boldsymbol{r}$ to a random vector and sets its binary vector $\boldsymbol{b}$ to a vector of zeros. Then, $\lceil rand \times D \rceil$ variables in the binary vector are randomly selected and flipped to one, where *rand* denotes a random number within $[0,1]$. In each generation, SLMEA first updates the archive (Line 6) and uses it to divide the decision variables into several sets *Group* for dimensionality reduction (Line 7). Then, the algorithm selects $2N$ parents from the population for generating N offsprings, where each parent is selected via binary tournament selection (Line 8). After generating the offsprings (Line 9), the parameters K and ρ are automatically updated (Line 10) and the offsprings are combined with the population (Line 11), among which N solutions survive to the next generation (Lines 12–16). For simplicity, the mating selection and environmental selection adopt the same selection criteria to NSGA-II [43], where solutions with smaller non-dominated front numbers and larger crowding distance values are preferred. To handle problems with more than three objectives, the mating selection and environmental selection in many-objective evolutionary algorithms [48] can also be adopted. The core components of SLMEA are elaborated in the next three subsections, including decision variables clustering, offspring generation, and parameter adaptation.

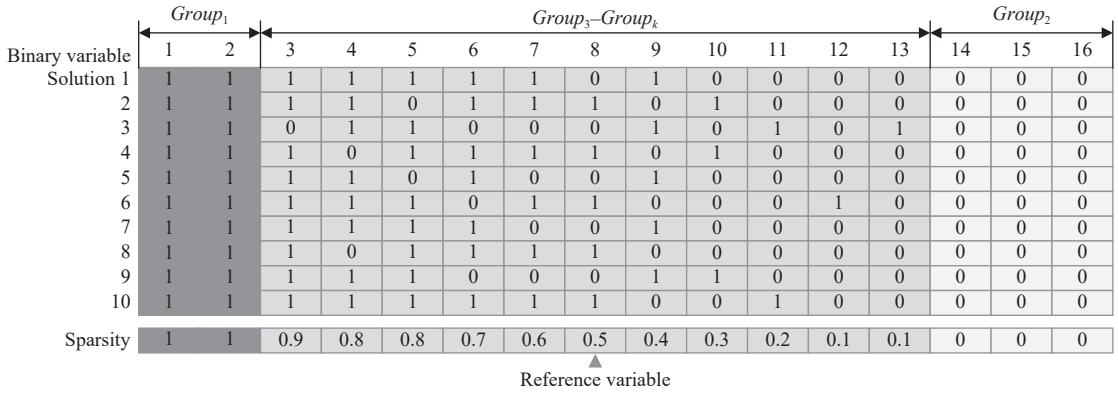

Fig. 2. Illustration of the fast clustering method in SLMEA.

Algorithm 1 Procedure of SLMEA

Input: N (population size), N_A (archive size)
Output: P (final population)
1 $P \leftarrow Initialization(N)$; //Initial population
2 $A \leftarrow \emptyset$; //Initial archive
3 $K \leftarrow 5$; //Initial number of groups
4 $\rho \leftarrow 0.5$; //Initial parameter
5 while termination criterion is not fulfilled **do**
6 $A \leftarrow UpdateArchive(A,P,N_A)$; //Algorithm 2
7 $Group \leftarrow Clustering(A,K)$; //Algorithm 3
8 $P' \leftarrow$ Select $2N$ parents via binary tournament selection according to the non-dominated front number and crowding distance of solutions in P;
9 $O \leftarrow Variation(P',Group,\rho)$; //Algorithm 4
10 $[K,\rho] \leftarrow ParaAdapt(O,K,\rho)$; //Algorithm 7
11 $P \leftarrow P \cup O$;
12 $[F_1,F_2,\ldots] \leftarrow$ Perform non-dominated sorting on P;
13 $CrowdDis \leftarrow$ Calculate the crowding distance for each solution in $F_1,F_2,\ldots$;
14 $l \leftarrow$ Minimum value s.t. $|F_1 \cup \cdots \cup F_l| \geq N$;
15 Delete $|F_1 \cup \cdots \cup F_l| - N$ solutions from F_l with the smallest $CrowdDis$;
16 $P \leftarrow F_1 \cup \cdots \cup F_l$;
17 return P.

B. Fast Clustering Based Dimensionality Reduction

In order to divide the decision variables for dimensionality reduction, the proposed SLMEA updates an archive A to store useful information about the relations between variables. As shown in Algorithm 2, the archive is first combined with the current population P, and only the non-dominated solutions remain. If the number of non-dominated solutions exceeds the archive size N_A, N_A solutions with larger crowding distance values are remained. Then, the binary vectors of the solutions in the archive are used to divide the decision variables into K groups. As illustrated in Fig. 2, the sparsity of each variable is calculated as its mean value among the binary vectors of all solutions in the archive

$$Sparsity_i = \frac{1}{|A|}\sum_{j=1}^{|A|} B_{ji} \tag{2}$$

where B_{ji} denotes the value of the ith binary variable of the jth solution in A. Afterwards, the variables with a sparsity of 1 are assigned to the first group $Group_1$, the variables with a sparsity of 0 are assigned to the second group $Group_2$, and all the other variables are grouped into $K-2$ groups via a fast clustering method. Specifically, the variable whose sparsity is the closest to 0.5 is selected as the reference variable c, and the similarity between each variable b and c is calculated by

$$Sim_{bc} = \frac{N_{(b=0,c=1)} + N_{(b=1,c=0)}}{N_{(b=0,c=1)} + N_{(b=1,c=0)} + N_{(b=1,c=1)}} \tag{3}$$

where $N_{(b=0,c=1)}$ denotes the number of solutions in A whose bth binary variable is 0 and cth binary variable is 1. Lastly, the variables are sorted according to their similarities to the reference variable, where the first $(D-|Group_1|-|Group_2|)/(K-2)$ variables are assigned to the group $Group_3$, the second $(D-|Group_1|-|Group_2|)/(K-2)$ variables are assigned to the group $Group_4$, and so on. In Section IV-D, the proposed fast clustering method is empirically verified to be more effective than some other clustering methods.

Algorithm 2 $UpdateArchive(A,P,N_A)$

Input: A (current archive), P (current population), N_A (archive size)
Output: A (updated archive)
1 $A \leftarrow A \cup P$;
2 $A \leftarrow$ The non-dominated solutions in A;
3 if $|A| > N_A$ **then**
4 $CrowdDis \leftarrow$ Calculate the crowding distance for each solution in A;
5 Delete $|A| > N_A$ solutions from A with the smallest $CrowdDis$;
6 return A.

The pseudocode of the proposed fast clustering method is given in Algorithm 3. Since most of the operations are related to a large number of binary variables, a binary matrix B is first constituted by the binary vector of all the solutions in the archive (Line 1). Then, all the operations are performed based on B without any iterated steps (Lines 2–9), which can be easily accelerated by GPUs since they are all matrix calculations. Lastly, the decision variables are divided into several groups according to the rank of similarities (Lines 11 and 12), where

the number of iterations K is very small. As a consequence, the proposed clustering method is much faster than conventional clustering methods (e.g., k-means and DBSCAN [49]) that should repeatedly calculate the distances between nodes, which hold very high computational complexities due to the large number of variables. The groups of variables are then used to reduce the search space in offspring generation, which is described in the next subsection.

Algorithm 3 *Clustering*(A, K)

Input: A (current archive), K (number of groups)
Output: *Group* (groups of variables)
1 $B \leftarrow$ A $|A| \times D$ matrix containing the binary vectors of all solutions in A; //D denotes the number of variables
2 $Sparsity \leftarrow \text{sum}(B)$; //sum of each column of B
3 $c \leftarrow \text{argmin}_b |Sparsity_b - 0.5|$;
4 $B' \leftarrow \text{repmat}(B_{\cdot c}, D)$; //Repeat the c-th column of B for D columns
5 $Sim \leftarrow \text{sum}\left(\frac{(1-B)\cdot B' + B\cdot(1-B')}{(1-B)\cdot B' + B\cdot(1-B') + B\cdot B'}\right)$;
6 $rank \leftarrow \text{sort}(Sim)$; //Rank of variables in Sim
7 $Group_1 \leftarrow \{b | Sparsity_b = 1\}$;
8 $Group_2 \leftarrow \{b | Sparsity_b = 0\}$;
9 $rank \leftarrow rank \setminus Group_1 \setminus Group_2$;
10 $l \leftarrow \lceil \frac{|rank|}{K-2} \rceil$;
11 for $i = 3$ to K **do**
12 $Group_i \leftarrow \{rank_{1+(i-3)l}, \ldots, rank_{\min(|rank|,(i-2)l)}\}$;
13 return *Group*.

C. Offspring Generation

The proposed SLMEA generates the real vector of each offspring by using SBX [17] and polynomial mutation [18], while generates the binary vector of each offspring by using uniform crossover and bit-flip mutation. Moreover, both the real vectors and binary vectors of offsprings are generated in the reduced space according to *Group*. To be specific, for the real vector $\boldsymbol{r} = (r_1, \ldots, r_D)$ of each parent, it is shortened to $\boldsymbol{r}^* = (r_1^*, \ldots, r_K^*)$ by representing all the variables in the same group by a single real value, where

$$r_i^* = \frac{1}{|Group_i|} \sum_{j \in Group_i} r_j. \tag{4}$$

Similarly, for the binary vector $\boldsymbol{b} = (b_1, \ldots, b_D)$ of each parent, it is shortened to $\boldsymbol{b}^* = (b_1^*, \ldots, b_K^*)$ by representing all the variables in the same group by a single binary value, where

$$p(b_i^* = 1 | Group_i) = \frac{1}{|Group_i|} \sum_{j \in Group_i} b_j \tag{5}$$

and the value of b_i^* is sampled by comparing the probability p with a uniformly distributed random value within $[0, 1]$. After generating the reduced real vector $\boldsymbol{r}^*$ and binary vector $\boldsymbol{b}^*$ of each offspring, they can be recovered to $\boldsymbol{r}$ and $\boldsymbol{b}$ by

$$\begin{aligned} r_j &= r_i^* \\ b_j &= b_i^* \end{aligned} \tag{6}$$

for all $j \in Group_i$ and $i \in \{1, \ldots, K\}$. Therefore, the search space can be highly reduced from D to the number of groups K.

The pseudocode of generating offspring is presented in Algorithm 4. In order to accelerate the operations via GPUs, a real matrix PR consisting of the real vectors of all the parents (Line 2) and a binary matrix PB consisting of the binary vectors of all the parents (Line 5) are constructed, where the real matrix OR and binary matrix OB of the offsprings are generated based on several matrix calculations on PR and PB (Lines 12–17), respectively. To strike a balance between exploration and exploitation, some offsprings are generated in the reduced search space while others are generated in the original search space. That is, the real matrix and binary matrix of the parents are divided into two parts, where the matrices PR and PB are used to generate offspring in the original search space while the matrices PR^* and PB^* are used to generate offspring in the reduced search space.

Algorithm 4 *Variation*$(P', Group, \rho)$

Input: P' (parent population), *Group* (groups of variables), ρ (ratio of offspring generated in the reduced space)
Output: O (offspring population)
1 $index \leftarrow rand_{1\times|P'|} > \rho$; //Indexes of the elements in a $1 \times |P'|$ random vector that are larger than ρ
2 $PR \leftarrow$ A $|index| \times D$ matrix containing the real vectors of all solutions in P' whose indexes are in *index*;
3 $PR' \leftarrow$ A $(|P'| - |index|) \times D$ matrix containing the real vectors of all solutions in P' whose indexes are not in *index*;
4 $PR^* \leftarrow$ A $(|P'| - |index|) \times |Group|$ matrix of zeros;
5 $PB \leftarrow$ A $|index| \times D$ matrix containing the binary vectors of all solutions in P' whose indexes are in *index*;
6 $PB' \leftarrow$ A $(|P'| - |index|) \times D$ matrix containing the binary vectors of all solutions in P' whose indexes are not in *index*;
7 $PB^* \leftarrow$ A $(|P'| - |index|) \times |Group|$ matrix of zeros;
8 for $i = 1$ to $|Group|$ **do**
9 $PR^*_{\cdot i} \leftarrow \text{mean}(PR'_{\cdot Group_i})$; //$i$th column of PR^* is set to the mean of $Group_i$th columns of PR'
10 $PB^*_{\cdot i} \leftarrow \text{mean}(PB'_{\cdot Group_i})$; //$i$th column of PB^* is set to the mean of $Group_i$th columns of PB'
11 $PB^* \leftarrow PB^* > rand_{(|P'|-|index|)\times|Group|}$; //">" is the element wise logical operator of "greater than"
12 $OR \leftarrow RealOperators(PR)$; //Algorithm 5
13 $OR^* \leftarrow RealOperators(PR^*)$; //Algorithm 5
14 $OR' \leftarrow$ A $(|P'| - |index|) \times D$ matrix of zeros;
15 $OB \leftarrow BinaryOperators(PB)$; //Algorithm 6
16 $OB^* \leftarrow BinaryOperators(PB^*)$; //Algorithm 6
17 $OB' \leftarrow$ A $(|P'| - |index|) \times D$ matrix of zeros;
18 for $i = 1$ to $|Group|$ **do**
19 $OR'_{\cdot Group_i} \leftarrow OR^*_{\cdot i}$; //$Group_i$th columns of OR' are set to the same as ith column of OR^*
20 $OB'_{\cdot Group_i} \leftarrow OB^*_{\cdot i}$; //$Group_i$th columns of OB' are set to the same as ith column of OB^*
21 $O \leftarrow$ Use the real matrix $OR \cup OR'$ and binary matrix $OB \cup OB'$ to generate an offspring population;
22 return O.

It is worth noting that the adopted crossover and mutation operators are performed on a single decision variable, which cannot handle the real and binary matrices directly. In particular, the widely used SBX contains two for-end blocks and three if-else blocks for generating a number of offsprings [17]. To address this issue, we convert all the operations of the crossover and mutation operators into matrix calculations, so that the generation of offspring can be deployed on GPUs. As shown in Algorithm 5, the adopted SBX and polynomial mutation consist of several matrix calculations on the real matrix without any for-end or if-else block, where the crossover probability is set to 1, the mutation probability is set to $1/D$, and the distribution index is set to η. Similarly, the adopted uniform crossover and bit-flip mutation also consist of only matrix calculations as shown in Algorithm 6.

Algorithm 5 *RealOperators*(PR)

Input: PR (real matrix of parents)
Output: OR (real matrix of offsprings)
//Simulated binary crossover
$PR1 \leftarrow$ The upper half of PR; //Set of first parents for generating each offspring
$PR2 \leftarrow$ The lower half of PR; //Set of second parents for generating each offspring
$M \leftarrow$ A random matrix with the same size as $PR1$; //Each element is randomly sampled in $[0,1]$
$R_1 \leftarrow$ A random matrix with the same size as $PR1$;
$R_2 \leftarrow$ A random matrix with the same size as $PR1$;
$T_1 \leftarrow M < 0.5$; //"$<$" is the element-wise logical operator of "less than"
$T_2 \leftarrow \text{sign}(R_1 - 0.5)$; //"sign" is the element-wise signum function
$T_3 \leftarrow R_2 < 0.5$;
$B_1 \leftarrow (2 \times M)^{\frac{1}{\eta+1}}$; //The power operator works on each element of the matrix
$B_2 \leftarrow (2 - 2 \times M)^{-\frac{1}{\eta+1}}$;
$B \leftarrow T_1 \cdot B_1 + (1 - T_1) \cdot B_2$; //The multiplication operator works on each element of the matrix
$B \leftarrow T_2 \cdot B$;
$B \leftarrow (1 - T_3) \cdot B + T_3$;
$OR \leftarrow 0.5[(1 + B) \cdot PR1 + (1 - B) \cdot PR2]$;
//Polynomial mutation
$M \leftarrow$ A random matrix with the same size as OR;
$R \leftarrow$ A random matrix with the same size as OR;
$n \leftarrow$ Number of rows (i.e., number of offsprings) in OR;
$L \leftarrow \text{repmat}(Lower, n)$; //Repeat the lower bounds $Lower$ of the problem for n rows
$U \leftarrow \text{repmat}(Upper, n)$; //Repeat the upper bounds $Upper$ of the problem for n rows
$S \leftarrow R < \frac{1}{D}$;
$T \leftarrow M < 0.5$;
$T_1 \leftarrow S \cdot T$;
$T_2 \leftarrow S \cdot (1 - T)$
$C_1 \leftarrow [2 \cdot M + (1 - 2 \cdot M) \cdot (1 - \frac{OR - L}{U - L})^{\eta+1}]^{\frac{1}{\eta+1}} - 1$;
$C_2 \leftarrow 1 - [2 \cdot (1 - M) + 2 \cdot (M - 0.5) \cdot (1 - \frac{U - OR}{U - L})^{\eta+1}]^{\frac{1}{\eta+1}}$;
$OR \leftarrow OR + (U - L) \cdot (C_1 \cdot T_1 + C_2 \cdot T_2)$;
return OR.

Algorithm 6 *BinaryOperators*(PB)

Input: PB (binary matrix of parents)
Output: OB (binary matrix of offsprings)
//Uniform crossover
$PB1 \leftarrow$ The upper half of PB; //Set of first parents for generating each offspring
$PB2 \leftarrow$ The lower half of PB; //Set of second parents for generating each offspring
$M \leftarrow$ A random matrix with the same size as $PB1$; //Each element is randomly sampled in $[0,1]$
$T \leftarrow M < 0.5$; //"$<$" is the element-wise logical operator of "less than"
$OB \leftarrow PB1 \cdot T + PB2 \cdot (1 - T)$; //The multiplication operator works on each element of the matrix
//Bit-flip mutation
$M \leftarrow$ A random matrix with the same size as OB; //Each element is randomly sampled in $[0,1]$
$T \leftarrow M < \frac{1}{D}$;
$OB \leftarrow OB \cdot (1 - T) + (1 - OB) \cdot T$;
return OB.

D. Parameter Adaptation

The proposed SLMEA adaptively adjusts two parameters K and ρ to control the generation of offspring, where K denotes the number of groups (i.e., dimensions of the reduced search space) and ρ denotes the ratio of offsprings generated in the reduced search space. Intuitively, if many solutions generated in the reduced search space are promising, i.e., the reduced search space benefits the evolution of population, the parameters K and ρ should become larger to take better advantage of the reduction of search space. By contrast, if few solutions generated in the reduced search space are promising, the parameters K and ρ should become smaller. For this purpose, the parameter K is updated by

$$K_{t+1} = K_t \times e^{\frac{1}{K_t}\left(\frac{ns_{1,t}}{s_{1,t}} - \frac{ns_{1,t-1}}{s_{1,t-1}}\right)} \tag{7}$$

where K_t denotes the value of K at the tth generation, $s_{1,t}$ denotes the number of offsprings generated in the reduced search space at the tth generation, and $ns_{1,t}$ denotes the number of non-dominated offsprings generated in the reduced search space at the tth generation. Obviously, a larger $\frac{ns_{1,t}}{s_{1,t}}$ indicates that more solutions generated in the reduced search space are promising, and a positive $\frac{ns_{1,t}}{s_{1,t}} - \frac{ns_{1,t-1}}{s_{1,t-1}}$ indicates that the ratio of promising solutions generated in the reduced search space is increased. Thus, the value of K becomes larger.

On the other hand, the parameter ρ is updated by

$$\rho_{t+1} = 0.5 \times \left(\rho_t + \frac{s_{2,t} \times ns_{1,t}}{s_{2,t} \times ns_{1,t} + s_{1,t} \times ns_{2,t}}\right) \tag{8}$$

where $s_{2,t}$ denotes the number of offsprings generated in the original search space at the tth generation, and $ns_{2,t}$ denotes the number of non-dominated offsprings generated in the original search space at the tth generation. Since a larger $\frac{s_{2,t} \times ns_{1,t}}{s_{2,t} \times ns_{1,t} + s_{1,t} \times ns_{2,t}}$ indicates that the ratio of non-dominated solu-

tions generated in the reduced search space is larger than in the original search space, it is positively related to ρ so that more solutions can be generated in the reduced search space at the next generation. To summarize, the pseudocode of parameter adaptation is given in Algorithm 7.

Algorithm 7 *ParaAdapt*(O, K, ρ)

Input: O (offspring population), K, ρ (parameters)
Output: K, ρ (updated parameters)
1 $s_{1,t} \leftarrow$ The number of solutions in O whose binary vectors are generated in the reduced search space;
2 $s_{2,t} \leftarrow$ The number of solutions in O whose binary vectors are generated in the original search space;
3 $F_1 \leftarrow$ Determine the non-dominated solutions in O;
4 $ns_{1,t} \leftarrow$ The number of solutions in F_1 whose binary vectors are generated in the reduced search space;
5 $ns_{2,t} \leftarrow$ The number of solutions in F_1 whose binary vectors are generated in the original search space;
6 $K \leftarrow$ Update K according to (7);
7 $\rho \leftarrow$ Update ρ according to (8);
8 return K, ρ.

E. Time Complexity of SLMEA

According to Algorithm 1, the mating selection, environmental selection, and archive update of SLMEA use the same strategies to NSGA-II, which hold a time complexity of $O(MN^2)$ [50], where M is the number of objectives and N is the population size. For the fast clustering method, the time complexities of calculating the sparsity of binary variables, calculating similarities between each binary variable and the reference variable, and sorting the similarities are $O(ND)$, $O(ND)$, and $O(D\log D)$, respectively, where D is the number of decision variables. For the generation of offspring, it holds a time complexity of $O(ND)$ since all the operations are performed on the real matrix PR and the binary matrix PB. Considering that $M \ll \log D \ll N$, the total time complexity of SLMEA is $O(ND)$ for one generation, which equals the time complexity of many classical MOEAs such as NSGA-II [43], SPEA2 [51], and MOEA/D [44]. By contrast, the time complexity of existing large-scale MOEAs and sparse MOEAs is up to $O(D^2)$, since they divide the decision variables by detecting the interaction between each two variables [9], [10], or reduce the decision space by learning the relations between variables [12], [52]. Moreover, with the assistance of GPU accelerated matrix calculations, the runtime of SLMEA is much less than existing MOEAs as indicated in the experimental results.

IV. Experimental Results

To verify the performance of the proposed SLMEA on SLMOPs, it is compared to six state-of-the-art MOEAs (i.e., NSGA-II [43], CCGDE3 [7], LMOCSO [20], WOF-SMPSO [11], SparseEA [35], and MOEA/PSL [12]), where NSGA-II is a classical MOEA that holds the same selection strategies to SLMEA; CCGDE3, LMOCSO, and WOF-SMPSO are state-of-the-art MOEAs for solving LMOPs; SparseEA and MOEA/PSL are state-of-the-art MOEAs for solving sparse LMOPs. These MOEAs are tested on eight benchmark problems and three real-world applications with 10 000 to 1 000 000 decision variables, where all the experiments are conducted on PlatEMO [53].

A. Parameter Settings

1) Algorithms: The parameters in the compared MOEAs are set as suggested in their original papers. In CCGDE3, the number of groups is set to 2 and the size of each subpopulation is set to 40. In 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. In SLMEA, the archive size is set to the same as the population size. As for the reproduction operators, NSGA-II, SparseEA, MOEA/PSL, and the proposed SLMEA adopt SBX and polynomial mutation for real variables and uniform crossover and bit-flip mutation for binary variables, where the crossover probability is set to 1, the mutation probability is set to $1/D$ (D is the number of decision variables), and the distribution index η is set to 20. CCGDE3 adopts the differential evolution, where both the learning rate F and crossover rate CR are set to 0.5. LMOCSO adopts the competitive swarm optimizer and WOF-SMPSO adopts the particle swarm optimization, where the polynomial mutation is tailed after the particles are updated. When handling binary variables, CCGDE3, LMOCSO, and WOF-SMPSO optimize the same number of real variables within $[0, 1]$ and round them before calculating the objective values.

2) Benchmark Problems: The eight benchmark problems SMOP1–SMOP8 [35] are characterized by various landscapes with different difficulties, including low intrinsic dimensionality, epistasis, deception, and multi-modality. More importantly, all the optimal solutions of SMOP1–SMOP8 have adjustable sparsity. For all the eight problems, the number of objectives is set to 2, the number of decision variables is varied from 10 000 to 1 000 000, and the sparsity of optimal solutions (i.e., ratio of nonzero variables in each optimal solution) is set to 0.1.

3) Real-World Problems: Three types of sparse SLMOPs are taken from real-world applications, including feature selection [54], pattern mining [2], and neural network training [55]. The feature selection problem aims to select a small proportion of features from a training set for minimizing the classification error and number of selected features, the pattern mining problem aims to select some items from a transaction dataset for maximizing the frequency and occupancy rate of the selected items in the dataset, and the neural network training problem aims to optimize all the weights for minimizing the classification error and network complexity. Detailed definitions are referred to in [35]. For each problem, three datasets are adopted to form three test instances as listed in Table I, where the number of objectives is 2 and the number of decision variables is varied from 9712 to 100 000.

TABLE I
DATASETS OF THREE SPARSE SLMOPS IN REAL-WORLD APPLICATIONS

Feature selection	No. of variables	Dataset	No. of samples	No. of features	No. of classes
FS1	9712	nci9[1]	60	9712	9
FS2	45 151	DGLA_BRA_180[1]	180	45 151	4
FS3	100 000	Synthetic[2]	100	100 000	2
Pattern mining	No. of variables	Dataset	No. of transactions	No. of items	Avg. length of transactions
PM1	10 000	Synthetic [56]	5000	10 000	500
PM2	50 000	Synthetic [56]	25 000	50 000	2500
PM3	100 000	Synthetic [56]	50 000	100 000	5000
Neural network training	No. of variables	Dataset	No. of samples	No. of features	No. of classes
NN1	10 041	Madelon[1]	2600	500	2
NN2	40 041	colon[1]	62	2000	2
NN3	97 281	BASEHOCK[1]	1993	4862	2

1. https://jundongl.github.io/scikit-feature/datasets.html
2. https://scikit-learn.org/stable/

4) Population Size and Termination Criterion: For fairness, all the compared MOEAs evolve a population with the same size for the same number of function evaluations. For solving the benchmark problems, the population size is set to 100 for 10 000 and 100 000 decision variables; since the MOEAs with a population size of 100 will run out of memory for 1 000 000 decision variables, the population size is set to 50 in this case. The maximum number of function evaluations is set to 100 000, 300 000, and 1 000 000 for 10 000, 100 000, and 1 000 000 decision variables, respectively. To solve the real-world problems, the population size is always set to 100, and the maximum number of function evaluations is set to 50 000, 100 000, and 150 000 for approximately 10 000, 50 000, and 100 000 decision variables. It is worth noting that some MOEAs may be extremely time-consuming for problems with 1 000 000 decision variables, hence each MOEA is executed for at most three days, even if the maximum number of function evaluations is not reached.

5) Assessment Criteria: Since the true Pareto front of the benchmark problems are known, the inverted generational distance (IGD) [57] is employed to assess the quality of each population obtained by the compared MOEAs, where 10 000 reference points are sampled on each true Pareto front by using the methods suggested in [58]. On the other hand, the true Pareto front of the real-world problems are unknown, hence the hypervolume (HV) [59] is employed to assess the quality of each population, where the reference point is set to the worst objective values of the problems, i.e., $(1, 1)$. For each MOEA on each problem, the mean and standard deviation of the indicator values over 30 independent runs are recorded. In addition, the Friedman test with Bonferroni correction [60] at a significance level of 0.05 is adopted, where "+", "−", and "≈" indicate that the indicator values obtained by an MOEA are significantly better, significantly worse, and statistically similar to those obtained by the proposed SLMEA on a problem, respectively.

B. Comparisons on Benchmark Problems

The IGD values obtained by seven MOEAs on SMOP1–SMOP8 are listed in Table II. Generally, the proposed SLMEA achieves the best IGD values on 18 out of 24 test instances, which is followed by MOEA/PSL gaining the best IGD values on the remaining six test instances. In terms of the statistical test, the proposed SLMEA significantly outperforms MOEA/PSL on 18 test instances and outperforms the other five MOEAs on all the 24 test instances. In short, the proposed SLMEA exhibits obviously better performance than existing MOEAs for solving benchmark SLMOPs.

For visual observation, Fig. 3 plots the populations with median IGD values obtained by seven MOEAs on SMOP3, SMOP5, and SMOP7 with 1 000 000 decision variables. It can be found that the populations obtained by the proposed SLMEA have the best convergence, which is attributed to the bi-level encoding scheme and the fast clustering based dimensionality reduction method. The populations obtained by MOEA/PSL have the second best convergence, since it also uses the bi-level encoding scheme and a dimensionality reduction method. The populations obtained by WOF-SMPSO have the third best performance, which is mainly due to its problem transformation framework that can quickly converge to a local optimum. Besides, the populations obtained by NSGA-II, CCGDE3, LMOCSO, and SparseEA have the worst convergence, which implies that they are not suitable for solving SLMOPs.

In terms of efficiency, Fig. 4 presents the average runtime consumed by seven MOEAs on SMOP1–SMOP8 with 10 000, 100 000, and 1 000 000 decision variables. It can be clearly observed that the GPU accelerated SLMEA is much more efficient than the other MOEAs, where its superiority becomes more significant with the increase of the number of decision variables. In particular, the runtime of SLMEA is approximately 1/14 of the runtime of WOF-SMPSO, SparseEA, and MOEA/PSL on the problems with 1 000 000 decision variables; in fact, WOF-SMPSO, SparseEA, and MOEA/PSL run for three days before the maximum number of function evaluations is reached. It should be noted that NSGA-II, CCGDE3, and LMOCSO can be accelerated by

TABLE II
IGD VALUES OBTAINED BY NSGA-II, CCGDE3, LMOCSO, WOF-SMPSO, SPARSEEA, MOEA/PSL, AND THE PROPOSED SLMEA ON SMOP1–SMOP8, WHERE THE BEST RESULT IN EACH ROW IS SHOWN IN BOLD

Problem	D	NSGA-II	CCGDE3	LMOCSO	WOF-SMPSO	SparseEA	MOEA/PSL	SLMEA
SMOP1	10 000	8.5255E–1 (2.11E–2)−	1.1139E+0 (4.97E–2)−	7.1073E–1 (1.53E–2)−	2.2035E–1 (3.09E–2)−	3.8454E–1 (1.23E–2)−	2.2836E–2 (2.54E–3)−	**1.7755E–2 (5.56E–3)**
	100 000	1.3335E+0 (4.90E–3)−	1.1403E+0 (1.04E–2)−	7.2615E–1 (1.19E–2)−	3.2690E–1 (1.90E–2)−	6.5167E–1 (7.39E–3)−	**3.0330E–2 (4.64E–3)+**	3.8269E–2 (1.07E–2)
	1 000 000	1.6088E+0 (9.99E–4)−	1.1664E+0 (0.00E+0)−	7.3840E–1 (0.00E+0)−	2.8202E–1 (6.78E–3)−	8.1868E–1 (4.55E–4)−	4.8356E–2 (8.46E–3)−	**3.3165E–2 (4.57E–3)**
SMOP2	10 000	1.6723E+0 (6.57E–3)−	2.1214E+0 (6.69E–2)−	2.0303E+0 (4.06E–3)−	3.4487E–1 (1.34E–1)−	8.1988E–1 (1.14E–2)−	1.0195E–1 (1.39E–3)−	**8.5880E–2 (3.38E–2)**
	100 000	2.0483E+0 (5.26E–3)−	2.1093E+0 (3.35E–3)−	2.0365E+0 (1.24E–2)−	1.0149E+0 (2.34E–1)−	1.0585E+0 (2.90E–3)−	**1.0525E–1 (5.84E–3)+**	1.2244E–1 (1.09E–2)
	1 000 000	2.2175E+0 (1.01E–3)−	2.1489E+0 (0.00E+0)−	2.0289E+0 (0.00E+0)−	1.0516E+0 (1.07E–1)−	1.1651E+0 (0.00E+0)−	**1.2154E–1 (8.55E–3)+**	1.4765E–1 (3.54E–3)
SMOP3	10 000	2.0604E+0 (1.82E–2)−	2.1404E+0 (3.97E–2)−	1.7561E+0 (1.44E–2)−	7.0353E–1 (2.06E–3)−	2.5800E+0 (1.41E–2)−	4.7620E–2 (7.74E–3)−	**3.1226E–2 (1.65E–2)**
	100 000	2.4274E+0 (3.41E–3)−	2.1163E+0 (3.13E–3)−	1.7635E+0 (3.95E–3)−	7.0114E–1 (2.24E–4)−	2.8156E+0 (3.18E–3)−	4.8219E–1 (8.67E–2)−	**3.4858E–2 (1.11E–2)**
	1 000 000	2.5806E+0 (3.95E–4)−	2.1367E+0 (0.00E+0)−	1.7673E+0 (0.00E+0)−	7.0095E–1 (1.02E–6)−	2.9037E+0 (3.47E–4)−	5.1837E–2 (8.83E–3)−	**2.6745E–2 (1.49E–3)**
SMOP4	10 000	8.2280E–1 (3.57E–3)−	1.0483E+0 (4.56E–2)−	1.0378E+0 (1.60E–2)−	2.8248E–2 (3.98E–2)−	2.6116E–1 (4.86E–3)−	**4.7853E–3 (6.10E–5)+**	5.3671E–3 (7.43E–4)
	100 000	1.0069E+0 (3.79E–3)−	1.0606E+0 (1.37E–3)−	1.0321E+0 (1.54E–2)−	3.4081E–1 (5.82E–2)−	3.7765E–1 (2.23E–3)−	**4.9729E–3 (2.32E–4)+**	5.9332E–3 (1.47E–3)
	1 000 000	1.0912E+0 (8.23E–4)−	1.0829E+0 (0.00E+0)−	1.0447E+0 (0.00E+0)−	3.5733E–1 (0.00E+0)−	4.2600E–1 (3.24E–5)−	9.7073E–3 (5.81E–4)−	**9.3905E–3 (4.72E–4)**
SMOP5	10 000	6.0886E–1 (4.25E–3)−	6.8227E–1 (2.28E–2)−	4.5992E–1 (5.44E–4)−	3.5439E–1 (2.99E–3)−	2.2960E–1 (4.32E–3)−	8.3291E–3 (1.94E–4)−	**7.4829E–3 (1.55E–3)**
	100 000	9.1117E–1 (3.27E–3)−	6.8136E–1 (3.42E–3)−	4.6135E–1 (3.70E–4)−	3.6432E–1 (2.40E–3)−	3.8880E–1 (4.04E–3)−	9.4841E–3 (2.90E–4)−	**6.5888E–3 (1.39E–3)**
	1 000 000	1.0934E+0 (9.99E–4)−	7.0471E–1 (0.00E+0)−	4.6270E–1 (0.00E+0)−	9.7012E–2 (0.00E+0)−	4.9411E–1 (4.01E–4)−	1.6188E–2 (3.59E–4)−	**1.5529E–2 (4.70E–3)**
SMOP6	10 000	2.5603E–1 (3.54E–3)−	3.5305E–1 (2.27E–2)−	2.2085E–1 (1.68E–3)−	5.8457E–2 (1.27E–2)−	1.0191E–1 (2.31E–3)−	1.1692E–2 (5.95E–4)−	**7.3531E–3 (1.35E–3)**
	100 000	4.1137E–1 (2.87E–3)−	3.4831E–1 (2.71E–3)−	2.2544E–1 (2.19E–3)−	9.7341E–2 (4.84E–4)−	1.8529E–1 (1.42E–3)−	1.3794E–2 (3.96E–4)−	**5.2252E–3 (7.76E–4)**
	1 000 000	4.9367E–1 (5.62E–4)−	3.6337E–1 (0.00E+0)−	2.3002E–1 (0.00E+0)−	1.9822E–1 (2.80E–3)−	2.3308E–1 (0.00E+0)−	**2.2386E–2 (9.96E–4)+**	2.7621E–2 (3.67E–3)
SMOP7	10 000	1.6126E+0 (8.31E–2)−	1.7815E+0 (5.99E–2)−	8.3812E–1 (6.25E–2)−	8.1614E–2 (8.87E–3)−	9.3992E–1 (5.28E–3)−	1.2639E–1 (1.41E–2)−	**4.8563E–2 (4.25E–2)**
	100 000	2.4672E+0 (4.29E–2)−	1.8551E+0 (2.73E–2)−	8.4147E–1 (2.25E–2)−	2.1584E–1 (1.02E–2)−	1.5216E+0 (1.24E–2)−	2.4075E–1 (6.64E–3)−	**1.0216E–2 (3.69E–3)**
	1 000 000	3.0333E+0 (5.11E–3)−	1.9036E+0 (0.00E+0)−	8.6488E–1 (0.00E+0)−	5.4422E–1 (1.77E–2)−	1.8954E+0 (1.61E–3)−	1.0446E–1 (4.86E–3)−	**5.3954E–2 (6.17E–3)**
SMOP8	10 000	3.0912E+0 (3.35E–2)−	3.6289E+0 (4.72E–2)−	3.0709E+0 (1.05E–1)−	5.7141E–1 (2.35E–2)−	2.1975E+0 (2.09E–2)−	3.4412E–1 (6.35E–2)−	**3.0189E–1 (3.36E–2)**
	100 000	3.5233E+0 (8.99E–3)−	3.6259E+0 (3.90E–3)−	3.1002E+0 (4.92E–2)−	6.4938E–1 (1.31E–1)−	2.7164E+0 (7.09E–3)−	3.8808E–1 (8.60E–2)−	**3.3533E–1 (2.06E–2)**
	1 000 000	3.7057E+0 (9.92E–4)−	3.6743E+0 (0.00E+0)−	3.2062E+0 (0.00E+0)−	4.2363E–1 (0.00E+0)−	2.8873E+0 (4.24E–3)−	4.5537E–1 (7.97E–3)−	**4.5356E–1 (7.22E–3)**
+/−/≈		0/24/0	0/24/0	0/24/0	0/24/0	0/24/0	6/18/0	

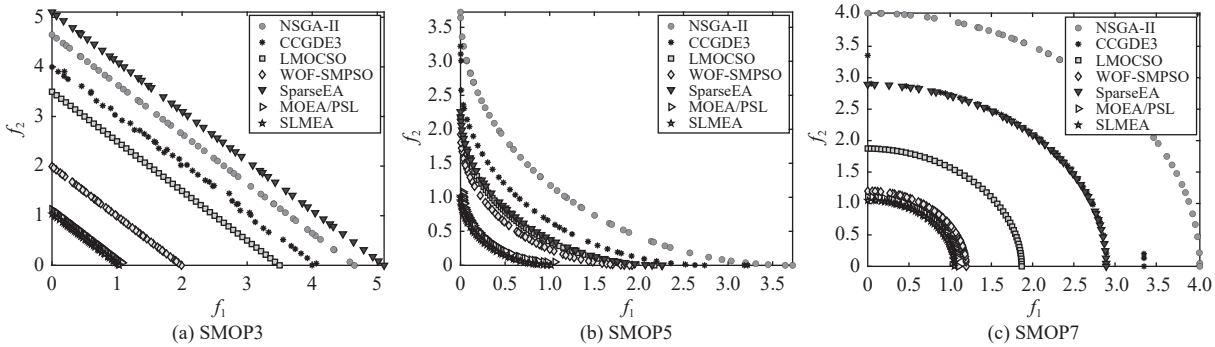

Fig. 3. Populations with median IGD values obtained by NSGA-II, CCGDE3, LMOCSO, WOF-SMPSO, SparseEA, MOEA/PSL, and the proposed SLMEA on SMOP3, SMOP5, and SMOP7 with 1 000 000 decision variables.

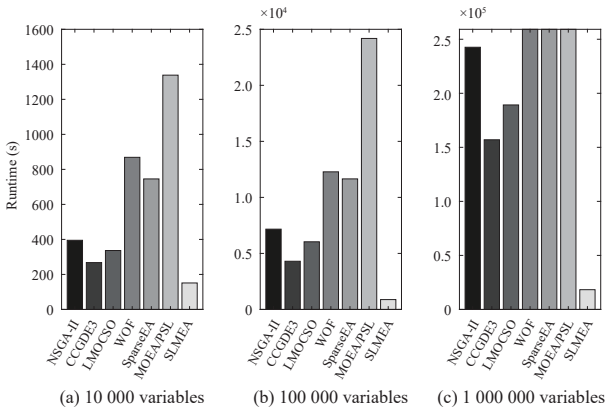

Fig. 4. Average runtime (in second) of NSGA-II, CCGDE3, LMOCSO, WOF-SMPSO, SparseEA, MOEA/PSL, and the proposed SLMEA on SMOP1–SMOP8 with 10 000, 100 000, and 1 000 000 decision variables.

GPUs if the matrix calculation based reproduction operators (i.e., Algorithms 5 and 6) are adopted, which will become as efficient as SLMEA; nevertheless, these MOEAs are ineffective for solving SLMOPs. By contrast, although MOEA/PSL exhibits the second best performance on the benchmark SLMOPs, it cannot be fully accelerated by GPUs since it requires many iterations to train neural networks. As a consequence, only SLMEA can strike a balance between effectiveness and efficiency in solving SLMOPs.

C. Comparisons on Real-World Problems

The HV values obtained by seven MOEAs on the real-world problems FS1–FS3, PM1–PM3, and NN1–NN3 are listed in Table III. It can be seen from the table that the proposed SLMEA obtains the best results on all the nine test instances, and its HV values are significantly better than those obtained by all the other MOEAs. Besides, Table IV presents the sparsity of the solutions obtained by seven MOEAs, where SLMEA obtains the sparsest solutions on six out of nine test instances. Hence, it is confirmed that the proposed SLMEA is effective in finding sparse solutions, and the tested real-world problems are indeed sparse SLMOPs. Furthermore, Fig. 5 plots the populations with median HV values obtained by seven MOEAs on FS3, PM3, and NN3 with approximately 100 000 decision variables. For the feature selection problem, both WOF-SMPSO and SLMEA can obtain a set of diverse solutions, while NSGA-II, CCGDE3, LMOCSO, SparseEA, and MOEA/PSL can only obtain several poorly converged solutions; clearly, the solutions obtained by SLMEA are superior to those obtained by WOF-SMPSO. For the patter mining problem, the solutions obtained by SLMEA also have better convergence and diversity than those obtained by WOF-SMPSO and MOEA/PSL, and the other four MOEAs can only obtain a single solution. As for the neural network training problem, the solutions obtained by SLMEA have lower error rates and much lower network complexities than those obtained by the other MOEAs.

To summarize, SLMEA is more effective than existing MOEAs for solving SLMOPs with sparse optimal solutions. In comparison to large-scale MOEAs such as CCGDE3, LMOCSO, and WOF-SMPSO, the proposed SLMEA adopts a bi-level encoding scheme, which can easily find sparse solutions and directly handle binary decision variables. While SparseEA also adopts a bi-level encoding scheme, the proposed SLMEA suggests a fast clustering method to reduce the dimensionality of the search space, hence the convergence speed can be highly accelerated. Compared to the neural network based dimensionality reduction method in MOEA/PSL, the fast clustering method in SLMEA does not train models or suffer from the lack of training samples, which is more effective in reducing the high-dimensional search spaces of SLMOPs. Moreover, SLMEA is more efficient than existing MOEAs for solving SLMOPs, since all the operations related to the decision variables can be converted into matrix calculations and accelerated by GPUs. On the other hand, regarding the limitations of SLMEA, it will be even less efficient than existing MOEAs if the number of decision variables is small, since the acceleration provided by GPUs becomes insignificant while the time consumed by the communication with GPUs is considerable. Besides, SLMEA is not effective for solving SLMOPs whose optimal solutions are not sparse, since the core components in SLMEA are tailored for generating sparse solutions.

D. Performance Verification of the Components in SLMEA

Lastly, the effectiveness of the core components in SLMEA is verified, including the fast clustering method and parameter adaptation strategies. For this aim, the original SLMEA is compared to several variants on the SMOP8 with 5000 real

TABLE III
HV VALUES OBTAINED BY NSGA-II, CCGDE3, LMOCSO, WOF-SMPSO, SPARSEEA, MOEA/PSL, AND THE PROPOSED SLMEA ON FS1–FS3, PM1–PM3, NN1–NN3, WHERE THE BEST RESULT IN EACH ROW IS SHOWN IN BOLD

Problem	D	NSGA-II	CCGDE3	LMOCSO	WOF-SMPSO	SparseEA	MOEA/PSL	SLMEA
FS1	9712	3.6349E–1 (0.00E+0)−	3.6002E–1 (0.00E+0)−	3.3387E–1 (0.00E+0)−	6.3592E–1 (0.00E+0)−	3.2777E–1 (0.00E+0)−	6.6356E–1 (5.44E–2)−	**8.1791E–1 (0.00E+0)**
FS2	45 151	4.2421E–1 (0.00E+0)−	4.1410E–1 (0.00E+0)−	4.2859E–1 (0.00E+0)−	7.7776E–1 (0.00E+0)−	3.9963E–1 (0.00E+0)−	7.2452E–1 (7.76E–3)−	**8.5856E–1 (0.00E+0)**
FS3	100 000	4.3576E–1 (0.00E+0)−	4.1768E–1 (0.00E+0)−	5.1029E–1 (0.00E+0)−	8.9974E–1 (0.00E+0)−	3.9709E–1 (0.00E+0)−	7.1771E–1 (2.72E–2)−	**9.6359E–1 (0.00E+0)**
PM1	10 000	8.2645E–3 (0.00E+0)−	8.2645E–3 (0.00E+0)−	8.2645E–3 (0.00E+0)−	9.5234E–2 (8.70E–4)−	8.2645E–3 (0.00E+0)−	9.5458E–2 (9.69E–4)−	**1.0272E–1 (1.63E–3)**
PM2	50 000	8.2645E–3 (0.00E+0)−	8.2645E–3 (0.00E+0)−	8.2645E–3 (0.00E+0)−	9.1905E–2 (1.62E–4)−	8.2645E–3 (0.00E+0)−	7.0969E–2 (4.18E–2)−	**9.3869E–2 (2.91E–4)**
PM3	100 000	8.2645E–3 (0.00E+0)−	8.2645E–3 (0.00E+0)−	8.2645E–3 (0.00E+0)−	9.1456E–2 (0.00E+0)−	8.2645E–3 (0.00E+0)−	9.1321E–2 (2.21E–5)−	**9.2860E–2 (0.00E+0)**
NN1	10 041	2.5927E–1 (6.95E–3)−	5.9808E–2 (2.78E–3)−	2.5847E–1 (1.14E–2)−	2.5825E–1 (7.41E–3)−	3.5055E–1 (9.55E–3)−	6.7542E–1 (3.15E–2)−	**7.2218E–1 (1.95E–2)**
NN2	40 041	3.8402E–1 (1.38E–2)−	9.0574E–2 (8.35E–4)−	3.6445E–1 (8.22E–3)−	3.7271E–1 (1.91E–2)−	4.5592E–1 (2.97E–3)−	9.2179E–1 (1.68E–2)−	**9.7708E–1 (9.01E–3)**
NN3	97 281	2.8323E–1 (1.78E–2)−	7.9908E–2 (2.80E–3)−	2.7893E–1 (5.65E–3)−	2.8347E–1 (5.04E–3)−	4.1181E–1 (2.27E–3)−	8.6065E–1 (1.20E–1)−	**9.9743E–1 (0.00E+0)**
+/−/≈		0/9/0	0/9/0	0/9/0	0/9/0	0/9/0	0/9/0	

TABLE IV
SPARSITY (I.E., RATIO OF NONZERO VARIABLES) OF THE SOLUTIONS OBTAINED BY NSGA-II, CCGDE3, LMOCSO, WOF-SMPSO, SPARSEEA, MOEA/PSL, AND THE PROPOSED SLMEA ON FS1–FS3, PM1–PM3, NN1–NN3, WHERE THE BEST RESULT IN EACH ROW IS SHOWN IN BOLD

Problem	D	NSGA-II	CCGDE3	LMOCSO	WOF-SMPSO	SparseEA	MOEA/PSL	SLMEA
FS1	9712	3.6697E–1 (0.00E+0)−	9.9928E–1 (0.00E+0)−	9.6762E–1 (0.00E+0)−	9.8211E–1 (0.00E+0)−	4.4419E–1 (0.00E+0)−	5.6072E–3 (2.28E–3)−	**6.1779E–4 (0.00E+0)**
FS2	45 151	4.4482E–1 (0.00E+0)−	9.9966E–1 (0.00E+0)−	8.7675E–1 (0.00E+0)−	9.9665E–1 (0.00E+0)−	4.7686E–1 (0.00E+0)−	3.1753E–2 (1.15E–2)−	**1.5090E–4 (0.00E+0)**
FS3	100 000	4.5699E–1 (0.00E+0)−	9.9992E–1 (0.00E+0)−	8.7593E–1 (0.00E+0)−	9.9413E–1 (0.00E+0)−	4.8508E–1 (0.00E+0)−	9.7297E–2 (2.48E–2)−	**3.5000E–4 (0.00E+0)**
PM1	10 000	2.8170E–3 (8.88E–5)−	9.4800E–1 (7.11E–4)−	2.0250E–3 (2.87E–4)−	9.5100E–1 (1.09E–2)−	4.9943E–1 (3.80E–3)−	**5.8289E–4 (1.35E–4)+**	1.1664E–3 (2.90E–5)
PM2	50 000	6.9995E–4 (3.63E–5)−	9.3428E–1 (7.64E–4)−	3.4500E–4 (6.61E–5)≈	9.6367E–1 (4.07E–2)−	5.0167E–1 (2.37E–3)−	3.1806E–2 (6.33E–2)≈	**3.0505E–4 (3.01E–5)**
PM3	100 000	4.3010E–4 (2.57E–5)−	9.3054E–1 (0.00E+0)−	2.1500E–4 (3.00E–5)−	9.8856E–1 (0.00E+0)−	4.9981E–1 (7.25E–4)−	**6.0032E–5 (2.71E–6)+**	1.8090E–4 (0.00E+0)
NN1	10 041	7.6070E–1 (2.30E–2)−	9.9980E–1 (9.34E–5)−	8.0566E–1 (1.81E–2)−	8.8686E–1 (1.74E–2)−	5.5913E–1 (1.40E–2)−	**3.2649E–2 (1.62E–2)≈**	6.1730E–2 (4.59E–2)
NN2	40 041	9.8555E–1 (6.65E–3)−	9.9992E–1 (4.33E–5)−	8.1772E–1 (3.06E–3)−	9.8255E–1 (3.95E–3)−	5.9386E–1 (1.20E–2)−	3.2557E–2 (1.81E–2)≈	**1.6776E–2 (1.42E–2)**
NN3	97 281	9.2039E–1 (5.63E–2)−	9.9997E–1 (9.84E–6)−	8.1875E–1 (4.23E–2)−	8.8439E–1 (5.00E–2)−	6.0958E–1 (1.10E–3)−	1.1616E–1 (4.80E–2)−	**1.6315E–2 (8.50E–3)**
+/−/≈		0/9/0	0/9/0	0/8/1	0/9/0	0/9/0	2/4/3	

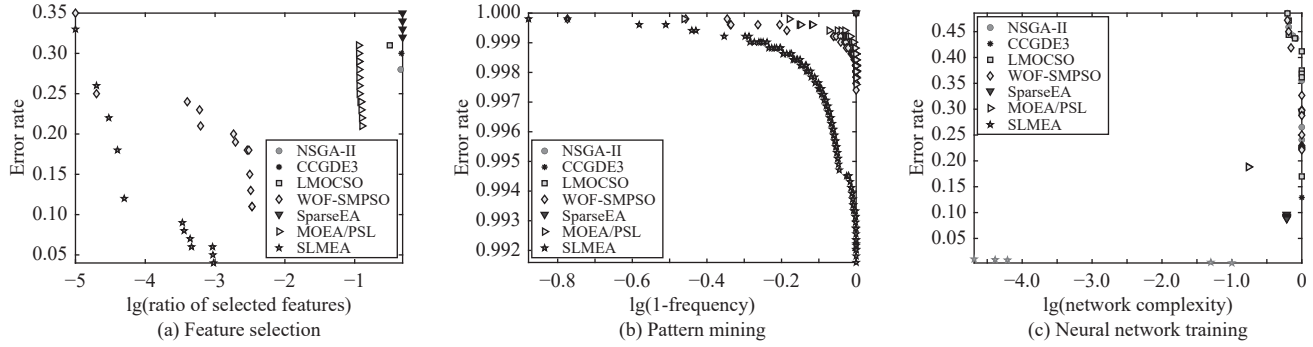

Fig. 5. Populations with median HV values obtained by NSGA-II, CCGDE3, LMOCSO, WOF-SMPSO, SparseEA, MOEA/PSL, and the proposed SLMEA on FS3, PM3, and NN3 with approximately 100 000 decision variables.

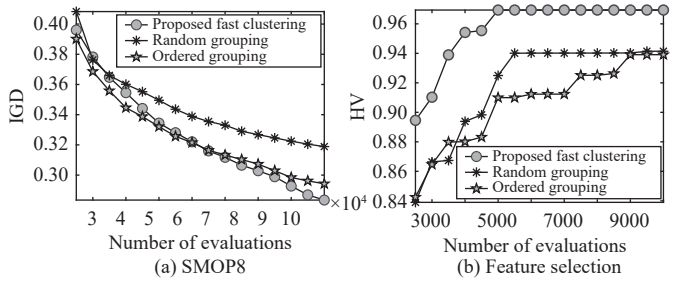

Fig. 6. Convergence profiles of SLMEA on SMOP8 and the feature selection problem, where the proposed fast clustering method, random grouping, and ordered grouping are adopted.

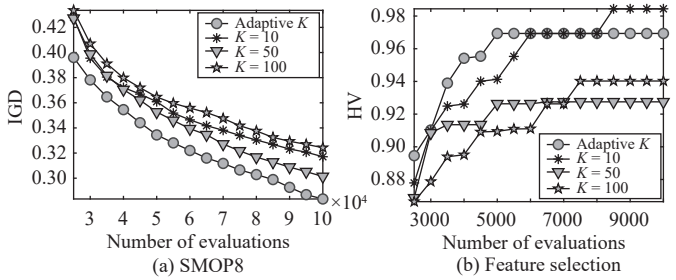

Fig. 7. Convergence profiles of SLMEA on SMOP8 and the feature selection problem, where the parameter K is adaptively updated or fixed to 10, 50, 100.

variables and the feature selection problem with 2000 binary variables (i.e., the dataset used in NN2). Fig. 6 depicts the convergence profiles of SLMEA with three clustering methods, including the proposed fast clustering, the random grouping, and the ordered grouping. The random grouping divides the binary variables into K groups randomly, and the ordered grouping divides the binary variables into K groups according to their sparsity calculated by (2), where both of them are popular clustering methods that do not require a large number of function evaluations to detect interactions between variables [7], [11]. As can be observed from the figure, the original SLMEA has a better convergence speed than the variants based on random grouping and ordered grouping, which verifies the effectiveness of the proposed fast clustering method.

Fig. 7 draws the convergence profiles of SLMEA with different settings of K, where K is adaptively updated by (7) or fixed to 10, 50, 100. It is obvious that the SLMEA with an adaptive K exhibits the best overall performance, having the best convergence speed on SMOP8 and a slightly worse convergence speed than the SLMEA with $K = 10$ on the feature selection problem. Besides, Fig. 8 depicts the convergence profiles of SLMEA with different settings of initial K, where the performance of SLMEA is very similar when using different initial values of K. Moreover, Fig. 9 shows the convergence profiles of SLMEA with different settings of ρ, where ρ is adaptively updated by (6) or fixed to $0.1, 0.5, 1$. It can be determined that the SLMEA with an adaptive ρ outperforms the SLMEA with a fixed ρ on SMOP8, and exhibits competitive performance compared to the SLMEA with $\rho = 0.1$. As a result, the effectiveness of the parameter adaptation strategies in SLMEA can be verified.

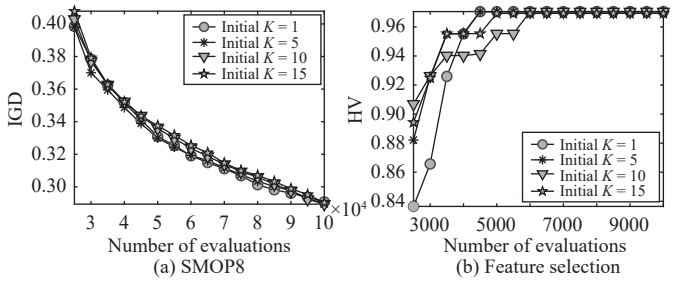

Fig. 8. Convergence profiles of SLMEA on SMOP8 and the feature selection problem, where the initial value of parameter K is set to $1, 5, 10, 15$.

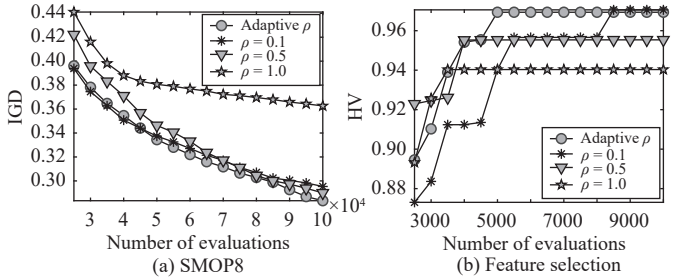

Fig. 9. Convergence profiles of SLMEA on SMOP8 and the feature selection problem, where the parameter ρ is adaptively updated or fixed to 0.1, 0.5, 1.

V. Conclusions

Existing MOEAs have successfully solved various LMOPs with less than 10 000 variables, but they showed low effec-

tiveness and efficiency in solving SLMOPs with hundreds of thousands of variables. Therefore, this paper serves as a first attempt to solve SLMOPs with sparse optimal solutions, where a fast clustering based MOEA has been proposed to address the curse of dimensionality. The proposed MOEA can reduce the high-dimensional search space by grouping the massive decision variables, which has been verified to be effective for solving SLMOPs with up to 1 000 000 variables. More importantly, all the operations related to decision variables can be converted into matrix calculations and accelerated by GPUs, where the runtime of the proposed MOEA has been reduced to less than a tenth of the runtime of existing MOEAs.

In the future, it is desirable to further enhance the proposed MOEA by taking full advantage of the objective functions of specific SLMOPs. In terms of the efficiency, the objective functions can be converted into those consisting of only matrix calculations, such that the procedure of function evaluations is considerably accelerated; furthermore, solution level parallelization strategies [45], [46] can also be adopted to accelerate the evaluation of each solution. In terms of its effectiveness, detailed information about the objective functions (e.g., datasets involved in the functions [61] and gradients of the functions [1]) can be considered in the generation of offspring, which highly improves the convergence of the population.

REFERENCES

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

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

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

[4] Y. Tian, X. C. Su, Y. S. Su, and X. Y. Zhang, "EMODMI: A multi-objective optimization based method to identify disease modules," *IEEE Trans. Emerg. Top. Comput. Intell.*, vol. 5, no. 4, pp. 570–582, Aug. 2021.

[5] J. Branke, B. Scheckenbach, M. Stein, K. Deb, and H. Schmeck, "Portfolio optimization with an envelope-based multi-objective evolutionary algorithm," *Eur. J. Oper. Res.*, vol. 199, no. 3, pp. 684–693, Dec. 2009.

[6] R. Cheng, Y. C. Jin, M. Olhofer, and B. Sendhoff, "Test problems for large-scale multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, vol. 47, no. 12, pp. 4108–4121, Dec. 2017.

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

[8] L. M. Antonio, C. A. Coello Coello, S. G. Brambila, J. F. González, and G. C. Tapia, "Operational decomposition for large scale multi-objective optimization problems," in *Proc. Genetic and Evolutionary Computation Conf. Companion*, Prague, Czech Republic, 2019, pp. 225–226.

[9] X. L. Ma, F. Liu, Y. T. Qi, X. D. Wang, L. L. Li, L. C. Jiao, M. L. Yin, and M. G. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Trans. EComput.*, vol. 20, no. 2, pp. 275–298, Apr. 2016.

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

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

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

[13] Y. C. Hua, Q. Q. Liu, K. R. Hao, and Y. C. Jin, "A survey of evolutionary algorithms for multi-objective optimization problems with irregular Pareto fronts," *IEEE/CAA J. Autom. Sinica*, vol. 8, no. 2, pp. 303–318, Feb. 2021.

[14] C. He, R. Cheng, C. J. Zhang, Y. Tian, Q. Chen, and X. Yao, "Evolutionary large-scale multiobjective optimization for ratio error estimation of voltage transformers," *IEEE Trans. EComput.*, vol. 24, no. 5, pp. 868–881, Oct. 2020.

[15] S. Singh, J. Kubica, S. Larsen, and D. Sorokina, "Parallel large scale feature selection for logistic regression," in *Proc. SIAM Int. Conf. Data Mining*, Sparks, USA, 2009, pp. 1172–1183.

[16] J. Liu, M. G. Gong, Q. G. Miao, X. G. 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.

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

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

[19] Y. Tian, S. S. Yang, X. Y. Zhang, and Y. C. Jin, "Using PlatEMO to solve multi-objective optimization problems in applications: A case study on feature selection," in *Proc. IEEE Congr. Evolutionary Computation, Wellington, New Zealand*, 2019, pp. 1–8.

[20] Y. Tian, X. T. Zheng, X. Y. Zhang, and Y. C. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3696–3708, Aug. 2020.

[21] W. J. Hong, P. Yang, and K. Tang, "Evolutionary computation for large-scale multi-objective optimization: A decade of progresses," *Int. J. Autom. Comput.*, vol. 18, no. 2, pp. 155–169, Apr. 2021.

[22] Y. Tian, L. C. Si, X. Y. Zhang, R. Cheng, C. He, K. C. Tan, and Y. C. Jin, "Evolutionary large-scale multi-objective optimization: A survey," *ACM Comput. Surv.*, vol. 54, no. 8, p. 174, Nov. 2022.

[23] A. Song, Q. Yang, W. N. Chen, and J. Zhang, "A random-based dynamic grouping strategy for large scale multi-objective optimization," in *Proc. IEEE Congr. Evolutionary Computation*, Vancouver, Canada, 2016, pp. 468–475.

[24] F. Sander, H. Zille, and S. Mostaghim, "Transfer strategies from single- to multi-objective grouping mechanisms," in *Proc. Genetic and Evolutionary Computation Conf.*, Kyoto, Japan, 2018, pp. 729–736.

[25] M. H. Li and J. X. Wei, "A cooperative co-evolutionary algorithm for large-scale multi-objective optimization problems," in *Proc. Genetic and Evolutionary Computation Conf. Companion*, Kyoto, Japan, 2018, pp. 1716–1721.

[26] H. K, Chen, X. M. Zhu, W. Pedrycz, S. Yin, G. H. Wu, and H. Yan, "PEA: Parallel evolutionary algorithm by separating convergence and diversity for large-scale multi-objective optimization," in *Proc. 38th IEEE Int. Conf. Distributed Computing Systems*, Vienna, Austria, 2018, pp. 223–232.

[27] C. He, L. H. Li, Y. Tian, X. Y. Zhang, R. Cheng, Y. C. Jin, and X. Yao, "Accelerating large-scale multiobjective optimization via problem reformulation," *IEEE Trans. EComput.*, vol. 23, no. 6, pp. 949–961, Dec. 2019.

[28] H. Qian and Y. Yu, "Solving high-dimensional multi-objective optimization problems with low effective dimensions," in *Proc. 31st AAAI Conf. Artificial Intelligence*, San Francisco, USA, 2017, pp. 875–881.

[29] W. J. Hong, K. Tang, A. M. Zhou, H. Ishibuchi, and X. Yao, "A scalable indicator-based evolutionary algorithm for large-scale multiobjective optimization," *IEEE Trans. EComput.*, vol. 23, no. 3, pp. 525–537, Jun. 2019.

[30] Y. Tian, C. Lu, X. Y. Zhang, F. Cheng, and Y. C. Jin, "A pattern mining-based evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Trans. Cybern.*, 2020. vol. 52, no. 7, pp. 6784–6797, Jul. 2022.

[31] Z. C. Lu, I. Whalen, V. Boddeti, Y. Dhebar, K. Deb, E. Goodman, and W. Banzhaf, "NSGA-NET: A multi-objective genetic algorithm for neural architecture search," arXiv preprint arXiv: 1810.03522, Oct. 2018.

[32] Y. Tian, S. S. Yang, and X. Y. Zhang, "An evolutionary multiobjective optimization based fuzzy method for overlapping community detection," *IEEE Trans. Fuzzy Syst.*, vol. 28, no. 11, pp. 2841–2855, Nov. 2020.

[33] X. Y. Zhang, F. C. Duan, L. Zhang, F. Cheng, Y. C. 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.

[34] J. H. Zhao, Y. Xu, F. J. Luo, Z. Y. Dong, and Y. 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.

[35] Y. Tian, X. Y. Zhang, C. Wang, and Y. C. Jin, "An evolutionary algorithm for large-scale sparse multiobjective optimization problems," *IEEE Trans. EComput.*, vol. 24, no. 2, pp. 380–393, Apr. 2020.

[36] E. G. Talbi, "A unified view of parallel multi-objective evolutionary algorithms," *J. Parallel Distrib. Comput.*, vol. 133, pp. 349–358, Nov. 2019.

[37] W. Q. Ying, S. Y. Chen, B. Wu, Y. H. Xie, and Y. Wu, "Distributed parellel MOEA/D on spark," in *Proc. Int. Conf. Computing Intelligence and Information System*, Nanjing, China, 2017, pp. 18–23.

[38] N. Kantour, S. Bouroubi, and D. Chaabane, "A parallel MOEA with criterion-based selection applied to the knapsack problem," *Appl. Soft Comput.*, vol. 80, pp. 358–373, Jul. 2019.

[39] T. F. Qiu and G. Ju, "A selective migration parallel multi-objective genetic algorithm," in *Proc. Chinese Control and Decision Conf.*, Xuzhou, China, 2010, pp. 463–467.

[40] C. Sanhueza, F. Jiméenez, R. Berretta, and P. Moscato, "PasMoQAP: A parallel asynchronous memetic algorithm for solving the multi-objective quadratic assignment problem," in *Proc. IEEE Congr. Evolutionary Computation*, Donostia, Spain, 2017, pp. 1103–1110.

[41] B. Derbel, A. Liefooghe, G. Marquet, and E. G. Talbi, "A fine-grained message passing MOEA/D," in *Proc. IEEE Congr. Evolutionary Computation*, Sendai, Japan, 2015, pp. 1837–1844.

[42] B. Xu, Y. Zhang, D. W. Gong, and L. Wang, "A parallel multi-objective cooperative co-evolutionary algorithm with changing variables," in *Proc. Genetic and Evolutionary Computation Conf. Companion*, Berlin, Germany, 2017, pp. 1888–1893.

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

[44] Q. F. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[45] T. J. Stanley and T. N. Mudge, "A parallel genetic algorithm for multiobjective microprocessor design," in *Proc. 6th Int. Conf. Genetic Algorithms*, San Francisco, USA, 1995, pp. 597–604.

[46] R. Szmit and A. Barak, "Evolution strategies for a parallel multi-objective genetic algorithm," in *Proc. 2nd Annu. Conf. Genetic and Evolutionary Computation*, Las Vegas, Nevada, USA, 2000, pp. 227–234.

[47] K. Deb and C. Myburgh, "A population-based fast algorithm for a billion-dimensional resource allocation problem with integer variables," *Eur. J. Oper. Res.*, vol. 261, no. 2, pp. 460–474, Sept. 2017.

[48] B. D. Li, J. L. Li, K. Tang, and X. Yao, "Many-objective evolutionary algorithms: A survey," *ACM Comput. Surv.*, vol. 48, no. 1, p. 13, Sept. 2015.

[49] M. Ester, H. P. Kriegel, J. Sander, and X. W. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. Int. Conf. Knowledge Discovery and Data Mining*, Portland, Oregon, USA, 1996, pp. 226–231.

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

[51] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proc. 5th Conf. Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, Athens, Greece, 2001, pp. 95–100.

[52] R. C. Liu, R. Ren, J. Liu, and J. Liu, "A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems," *Appl. Soft Comput.*, vol. 89, p. 106120, Apr. 2020.

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

[54] B. Xue, M. J. 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.

[55] X. Yao, "A review of evolutionary artificial neural networks," *Int. J. Intell. Syst.*, vol. 8, no. 4, pp. 539–567, Jan. 1993.

[56] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proc. 20th Int. Conf. Very Large Data Bases*, San Francisco, USA, 1994, pp. 487–499.

[57] 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. EComput.*, vol. 7, no. 2, pp. 117–132,

Apr. 2003.

[58] Y. Tian, X. S. Xiang, X. Y. Zhang, R. Cheng, and Y. C. Jin, "Sampling reference points on the Pareto fronts of benchmark multi-objective optimization problems," in *Proc. IEEE Congr. Evolutionary Computation*, Rio de Janeiro, Brazil, 2018, pp. 1–6.

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

[60] 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 EComput.*, vol. 1, no. 1, pp. 3–18, Mar. 2011.

[61] X. Xiang, Y. Tian, J. H. Xiao, and X. Y. Zhang, "A clustering-based surrogate-assisted multiobjective evolutionary algorithm for shelter location problem under uncertainty of road networks," *IEEE Trans. Industrial Informatics*, vol. 16, no. 12, pp. 7544–7555, Dec. 2020.

Ye Tian received the B.Sc., M.Sc., and Ph.D. degrees from Anhui University in 2012, 2015, and 2018, respectively. He is currently an Associate Professor with the Institutes of Physical Science and Information Technology, Anhui University, and also a Postdoctoral Research Fellow with the Department of Computing, the Hong Kong Polytechnic University. His current research interests include evolutionary computation and its applications. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award, the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award, and the 2022 IEEE Computational Intelligence Society Outstanding Ph.D. Dissertation Award.

Yuandong Feng received the B.Sc. degree from Hunan Normal University in 2018. He is currently a master student at the School of Computer Science and Technology, Anhui University. His current research interests include large scale multi-objective optimization and its applications.

Xingyi Zhang (Senior Member, IEEE) received the B.Sc. degree from Fuyang Normal College in 2003, and the M.Sc. and Ph.D. degrees from Huazhong University of Science and Technology in 2006 and 2009, respectively. He is currently a Professor with the School of Artificial Intelligence, Anhui University. His current research interests include unconventional models and algorithms of computation, multi-objective optimization, and membrane computing. He is the recipient of the 2018 and 2021 IEEE Transactions on Evolutionary Computation Outstanding Paper Award and the 2020 IEEE Computational Intelligence Magazine Outstanding Paper Award.

Changyin Sun received the B.Sc. degree from Sichuan University in 1996, and the M.Sc. and Ph.D. degrees from Southeast University in 2001 and 2004, respectively. He is currently a Professor with the School of Automation, Southeast University. His current research interests include intelligent control, flight control, pattern recognition, and optimal theory. He is an Associate Editor of *IEEE Transactions on Neural Networks and Learning Systems*.