Complex & Intelligent Systems (2023) 9:1127–1142
https://doi.org/10.1007/s40747-021-00553-0

ORIGINAL ARTICLE

Improved SparseEA for sparse large-scale multi-objective optimization problems

Yajie Zhang[1] · Ye Tian[2] · Xingyi Zhang[1]

Received: 14 July 2021 / Accepted: 21 September 2021 / Published online: 9 October 2021

Abstract
Sparse large-scale multi-objective optimization problems (LSMOPs) widely exist in real-world applications, which have the properties of involving a large number of decision variables and sparse Pareto optimal solutions, i.e., most decision variables of these solutions are zero. In recent years, sparse LSMOPs have attracted increasing attentions in the evolutionary computation community. However, all the recently tailored algorithms for sparse LSMOPs put the sparsity detection and maintenance in the first place, where the nonzero variables can hardly be optimized sufficiently within a limited budget of function evaluations. To address this issue, this paper proposes to enhance the connection between real variables and binary variables within the two-layer encoding scheme with the assistance of variable grouping techniques. In this way, more efforts can be devoted to the real part of nonzero variables, achieving the balance between sparsity maintenance and variable optimization. According to the experimental results on eight benchmark problems and three real-world applications, the proposed algorithm is superior over existing state-of-the-art evolutionary algorithms for sparse LSMOPs.

Keywords Large-scale multi-objective optimization · Sparse pareto optimal solutions · Evolutionary algorithm · Real-world applications

Introduction

Large-scale multi-objective optimization problems, which widely exist in scientific and engineering areas, refer to the problems involving a large number of decision variables and multiple conflicting objectives. For example, the optimization of a vehicle routing problem (VRP) usually consists of hundreds of customers [1], and the number of decision variables in the ratio error estimation of voltage transformers (TREE) can vary from hundreds to even millions [2]. Evolutionary algorithms (EAs), as a population-based optimization method, are capable of obtaining a set of trade-off solutions in a single run and less vulnerable to trap into local optimums. A variety of EAs showing promising performance on different kinds of multi-objective optimization problems (MOPs) have been proposed during the past two decades [3–5], however, their performance usually degenerate when they are adopted to tackle LSMOPs. One reason is that the search space expands exponentially with the increasing of the number of decision variables, which is known as the curse of dimensionality [6]. To solve this dilemma, various approaches have been tailored for solving LSMOPs during the past ten years [7], which can be roughly categorized into four types. They are decision variable grouping [8], decision variable analysis [9,10], problem reformulation [11,12], and special offspring generation strategy based EAs [13,14,16,17].

In the field of LSMOPs, there exists a special kind of optimization problem becoming increasingly important in real-world applications and scientific researches, which is

This paper is an expanded version of our conference paper submitted to EMO2021.

✉ Xingyi Zhang
xyzhanghust@gmail.com

Yajie Zhang
yjzhang17719490727@163.com

Ye Tian
field910921@gmail.com

1 Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui University, Hefei 230601, China

2 Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, Institutes of Physical Science and Information Technology, Anhui University, Hefei 230601, China

known as sparse LSMOPs. The Pareto optimal solutions of such kind of problem are sparse, i.e., most decision variables of the solutions are zero. For example, the portfolio optimization problem is to maximize the expected return and minimize the potential risks, the invested products usually account for only a small portion of all candidate products [19]. In the neural network training problem, to minimize the complexity and error of a network model, many weights should be set to zero [18]. When existing approaches customized for general LSMOPs are employed to solve sparse LSMOPs, few of them can obtain a satisfactory solution set within a limited budget of function evaluations, since they do not consider the sparse nature of Pareto optimal solutions when evolving the population, thus converging slowly on sparse LSMOPs [20].

To fill the gap mentioned above, several multi-objective optimization evolutionary algorithms (MOEAs) have been tailored for sparse LSMOPs in recent years. In SparseEA [21], aiming to maintain the sparsity of generated solutions, a novel population initialization strategy and genetic operators have been proposed. In MOEA/PSL [22], two unsupervised neural networks are used to learn a sparse distribution and a compact representation of the decision variables, thus achieving the approximation of Pareto optimal subspace. In PM-MOEA [25], pattern mining techniques are utilized to mine the sparse distribution of Pareto optimal solutions and thus considerably reduce the search space. In MDR-SAEA [26], the authors propose to use feature selection approach to achieve dimensionality reduction, and apply Kriging-assisted evolutionary algorithms to solve the expensive sparse LSMOPs. In MP-MMEA [27], a multi-population MOEA guiding the search behavior of populations via adaptively updated vectors is proposed to deal with sparse large-scale multi-modal MOPs, where the guiding vectors can not only accelerate convergence in the huge search space, but also differentiate the search direction of each population.

It is necessary to note that all the algorithms enumerated above adopt a two-layer encoding scheme, i.e., each decision variable x_i is represented by $x_i = mask_i \times dec_i$, where i ranges from 1 to D, and D is the number of decision variables. One main purpose of adopting such encoding strategy is to facilitate the detection of the positions of nonzero variables. It can be found that the algorithms in [22,25,26] attempt to find the sparse distribution of decision variables firstly via different dimensionality reduction techniques, and put the optimization of nonzero real variables in the second place. However, paying too much attention to sparsity detection may hinder the optimization of nonzero variables. Figure 1 shows the parallel coordinates plot of the decision variables of solutions obtained by SparseEA, MOEA/PSL, and PM-MOEA on SMOP1 and SMOP3 with 1000 decision variables, where the sparsity of these two problems is set to 0.1, i.e., the last 900 decision variables in the Pareto optimal solutions are zero. It can be found that, for SMOP1, even though PM-MOEA detects the positions of nonzero variables more precisely than the other two algorithms, it does not obtain the best IGD value among the three algorithms as expected, since MOEA/PSL optimizes the nonzero real variables better. For SMOP3, when all the three algorithms detect the positions of nonzero variables precisely, PM-MOEA obtains the best IGD value as it optimizes the key variables which affect the function fitness more sufficiently.

Based on the above observations, it can be found that the optimization of nonzero variables is as important as the detection of sparsity. In this paper, we propose to enhance the connection between *mask* and *dec* with the assistance of variable grouping techniques. We do not put sparsity maintenance in the first place anymore, on the contrary, we hope to optimize key variables more sufficiently without sacrificing the effect of sparsity maintenance. In this way, the balance between sparsity maintenance and variables optimization can be better achieved. The main contributions of this paper are summarized as follows:

- A new algorithm equipping customized genetic operators for sparse LSMOPs is proposed in this paper, in which the connection between real variables and binary variables is enhanced with the assistance of variable grouping techniques. Thus, the real variables are ensured to be optimized as long as the corresponding binary variables are flipped, improving the efficiency of producing sparse Pareto optimal solutions.
- Based on the performance evaluation and empirical study results of the conference version of this paper [20], a more comprehensive comparison study is conducted to reveal the merits and drawbacks of the existing state-of-the-art MOEAs and the newly proposed algorithm on benchmark and real-world LSMOPs with sparse Pareto optimal solutions.

The remainder of this paper is organized as follows. We first introduce the existing MOEAs for general LSMOPs and sparse LSMOPs, and then elaborate on our proposed MOEA for sparse LSMOPs. Next, we present the experimental studies on eight benchmark problems and three real-world applications. Finally, we draw the conclusions and outline some future research directions.

Related work

As mentioned in the last section, existing MOEAs for general LSMOPs can be roughly categorized into four different groups. They are based on decision variable grouping [8], decision variable analysis [9,10], problem reformu-

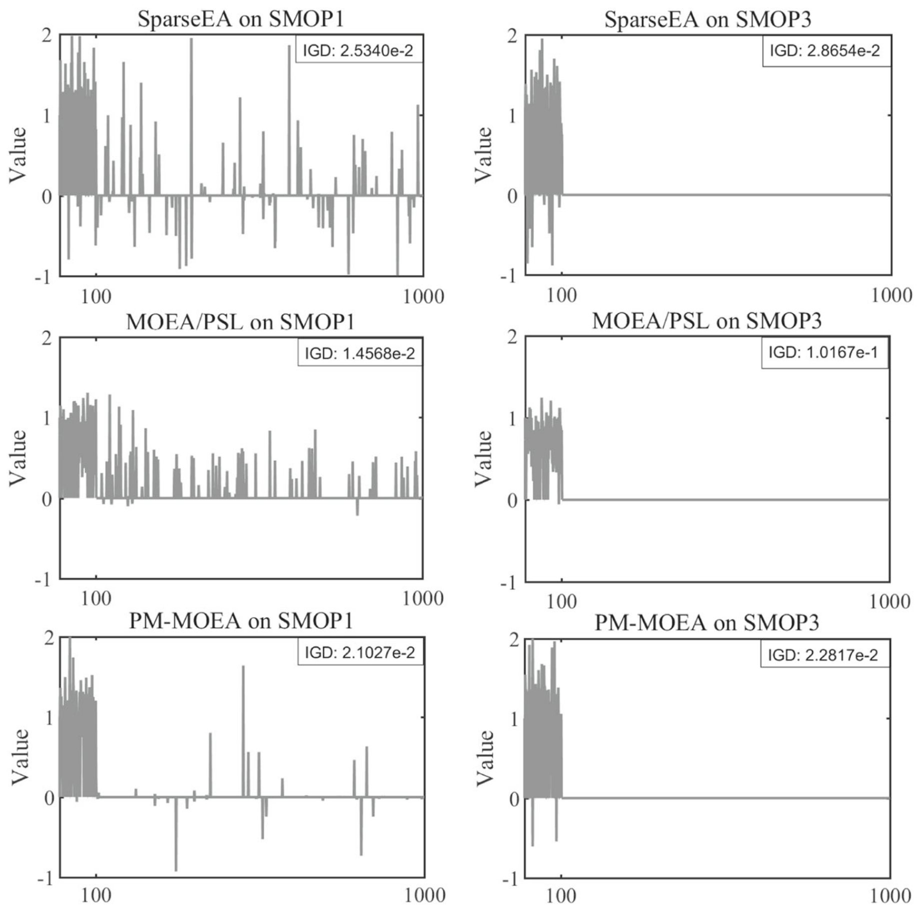

Fig. 1 Parallel coordinates plot of the decision variables of solutions obtained by SparseEA, MOEA/PSL, and PM-MOEA on SMOP1 and SMOP3 with 1000 variables, the sparsity is set to 0.1. The value in the top right corner of each sub-figure represents the median IGD indicator obtained by the corresponding algorithm over 30 runs

lation [11,12], and special offspring generation strategies [13,14,16,17].

The main idea of the algorithms based on decision variable grouping is to divide decision variables into different groups that can be optimized via independent sub-populations in a divide-and-conquer manner. However, their performance can be greatly affected by the adopted variable grouping techniques. For example, random grouping [28] is employed in the third-generation cooperative co-evolutionary differential evolution algorithm (CCGDE3) [8] considering that dividing variables into random groups provides better results than applying a deterministic division scheme when dealing with nonseparable functions. Besides, other variable grouping techniques (e.g., ordered grouping [29], linear grouping [30] and differential grouping [31]) have also shown effectiveness in solving specific LSMOPs.

MOEA/DVA [9] and LMEA [10] are two well-known algorithms belonging to the category based on decision variable analysis. The key component of MOEA/DVA consists of control property analysis and variable linkage analysis, in which the former divides the decision variables into position related variables, distance related variables, and mixed variables, while the latter further divides distance-related variables into smaller subgroups of interacting variables. Afterwards, variables in each subgroup are optimized independently through a differential evolution based optimizer. In contrast to MOEA/DVA treating mixed variables as diversity-related variables, LMEA clusters a decision variable as either convergence-related variable or diversity-related variable precisely. Subsequently, convergence optimization strategy and diversity optimization strategy are employed to optimize the corresponding variables alternately.

For the methods based on problem reformulation, two representative algorithms are WOF [11] and LSMOF [12]. WOF divides decision variables into different groups and assigns a weight variable to each group, thus the dimensionality of problems can be greatly reduced by altering variables in the same group at the same time. On the other hand, LSMOF defines a set of reference directions in the decision space and associates them with a number of weight variables to reformulate the original problem into a low-dimensional single-objective optimization problem. After obtaining enough quash-optimal solutions near the Pareto set, LSMOF spreads such solutions over the approximated Pareto set evenly via an embedded differential evolution algorithm.

The last category employs special offspring generation strategies. To improve the search efficiency, LMOCSO [14] suggests a new reproduction operator based on the competitive swarm optimizer [15], and an acceleration term is added to the position update mechanism to accelerate the convergence speed. Instead of optimizing decision variables directly, LCSA [16] evolves a population of coefficient vectors, by taking advantage of the inherent knowledge of the population, offsprings can thus be obtained by a linear combination of existing individuals. GLMO [17] embeds variable grouping into mutation operators to improve the quality of generated offsprings, and three new mutation operators are presented, they are Linked Polynomial Mutation, Grouped Polynomial Mutation and Grouped and Linked Polynomial Mutation. DGEA [13] generates promising solutions via constructing direction vectors in the decision space. Specifically, in each iteration, two kinds of direction vectors related to convergence and diversity are constructed adaptively, and offsprings are then produced along each direction vector through sampling the built Gaussian distribution.

Despite that, the above delicate approaches work well on general LSMOPs, their performance usually degenerates when they are applied to solve sparse LSMOPs. One main reason is that few of them consider the sparse nature of Pareto optimal solutions when evolving the population, thus converging slowly on sparse LSMOPs. To fit this gap, several MOEAs customized for sparse LSMOPs have been proposed, which make fully use of the sparsity of problems to speed up the convergence to Pareto optimal sets. In this paper, we divide them into two different categories according to whether the dimensionality reduction techniques are used.

As for the sparse MOEAs without dimensionality reduction techniques, SparseEA [21] has a similar framework to NSGA-II, while the novelties lie in its population initialization strategy and genetic operators, which ensure the sparsity of generated individuals. Specifically, in the population initialization strategy, SparseEA first calculates the fitness scores for each decision variable based on non-dominated sorting [3], and then generates the initial population based on the obtained scores. As for the genetic operators, SparseEA flips zero or nonzero binary variables with the same probability on the basis of fitness scores, however, for the real part of decision variables, SparseEA simply executes conventional genetic operators. Recently, A multi-population evolutionary algorithm, termed MP-MMEA, has been proposed for solving sparse large-scale multi-modal multi-objective optimization problems (MMOPs), MP-MMEA adopts adaptively adjusted guiding vectors to improve both the convergence and diversity of each population, in which the guiding vectors can not only lead the sub-populations to evolve towards sparse Pareto sets efficiently, but also diversify the search direction of each subpopulation in the decision space.

As for the sparse MOEAs based on dimensionality reduction techniques, MOEA/PSL [22] adopts the restricted Boltzmann machine (RBM) [23] and denoising autoencoder (DAE) [24] to learn the sparse distribution and compact representation of decision variables, and regards the combination of the learnt sparse distribution and compact representation as an approximation of the Pareto optimal subspace. Subsequently, genetic operators are conducted in the reduced subspace instead of the original search space, in this way, the huge search space is highly reduced. Similarly, PM-MOEA [25] utilizes data mining techniques to mine the maximum and minimum candidate sets of the nonzero variables in Pareto optimal solutions, and then executes genetic operators on the dimensions determined by the maximum and minimum candidate sets, therefore, the high-dimensional decision space can also be greatly reduced. To address the curse of dimensionality encountered in sparse LSMOPs with expensive functions, MDR-SAEA [26] executes non-dominated sorting based feature selection and mask evolving based feature selection within a multi-stage framework to reduce the search space, and then performs surrogate-assisted optimization for the dimension-reduced problems.

Proposed MOEA for sparse LSMOPs

Up to now, existing MOEAs tailored for sparse LSMOPs put sparsity maintenance in the first place, where the real part of nonzero variables can hardly be optimized sufficiently within a limited budget of function evaluations. Therefore, in this paper, an improved version of SparseEA, termed SparseEA2, is proposed, in which the connection between real variables and binary variables is enhanced with the assistance of variable grouping techniques. Thus ensuring that the real part of nonzero variables can attract more attentions to be optimized more sufficiently, without sacrificing the effect of sparsity maintenance. In this section, we will first introduce SparseEA, and then elaborate on our proposed SparseEA2 specifically.

SparseEA

Figure 2 shows the procedure of SparseEA, which is very similar to NSGA-II [3]. The mating pool selection and environmental selection of SparseEA are the same as the counterparts of NSGA-II, while the novelties lie in its population initialization strategy and genetic operators.

Algorithm 1 presents the population initialization strategy of SparseEA, which consists of two steps, i.e., calculating the fitness of decision variables and generating the initial population. In the first step, the real vector Dec is set to a uniformly randomly generated $D \times D$ matrix or a $D \times D$ matrix of ones according to the types of decision variables, and the binary vector $Mask$ is set to a $D \times D$ identity matrix. Here, we note that Dec denotes the decision variables and $Mask$ denotes the mask. Thereafter, a population Q with D solutions is generated by multiplying Dec by $Mask$. Then,

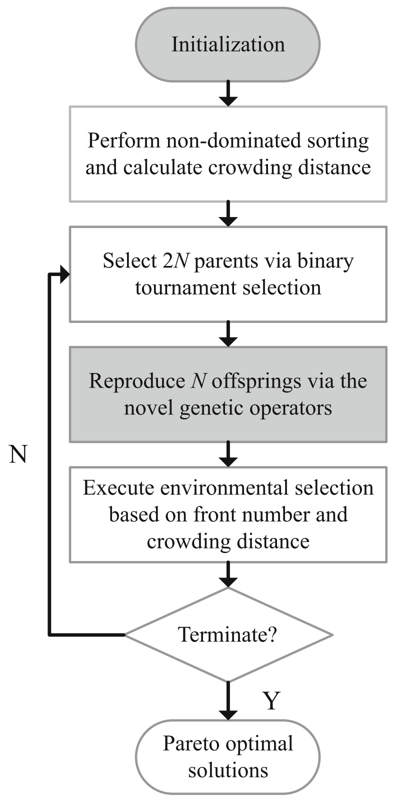
Initialization
Perform non-dominated sorting and calculate crowding distance
Select 2N parents via binary tournament selection
Reproduce N offsprings via the novel genetic operators
Execute environmental selection based on front number and crowding distance
Terminate?
N
Y
Pareto optimal solutions

Fig. 2 Procedure of SparseEA

Algorithm 1: Initialization strategy of SparseEA

```
Input: $N$ (population size)
Output: $P$ (initial population), $Fit$ (fitness scores of decision variables)
1  //Calculating the fitness of decision variables
2  $D \leftarrow$ Number of decision variables;
3  if the decision variables are real numbers then
4      $Dec \leftarrow$ Uniformly randomly generate the decision variables of $D$ solutions;
5  else if the decision variables are binary numbers then
6      $Dec \leftarrow D \times D$ matrix of ones;
7  $Mask \leftarrow D \times D$ identity matrix;
8  $Q \leftarrow$ Generate a population by $Dec$ and $Mask$;
9  $[F_1, F_2, ...] \leftarrow$ Perform non-dominated sorting on $Q$;
10 for $i = 1$ to $D$ do
11     $Fit_i \leftarrow$ Assign the non-dominated front number of the $i$-th solution in $Q$ to the fitness of the $i$-th decision variable;
12 //Generating the initial population
13 if the decision variables are real numbers then
14     $Dec \leftarrow$ Uniformly randomly generate the decision variables of $N$ solutions;
15 else if the decision variables are binary numbers then
16     $Dec \leftarrow N \times D$ matrix of ones;
17 $Mask \leftarrow N \times D$ matrix of zeros;
18 for $i = 1$ to $N$ do
19     for $j = 1$ to $rand() \times D$ do
20         $[m, n] \leftarrow$ Randomly select two decision variables;
21         if $Fit_m \le Fit_n$ then
22             Set the $m$-th element in the $i$-th binary vector in $Mask$ to 1;
23         else
24             Set the $n$-th element in the $i$-th binary vector in $Mask$ to 1;
25 $P \leftarrow$ Generate a population by $Dec$ and $Mask$;
26 return $P$ and $Fit$;
```

non-dominated sorting is executed on Q and the front number of the i-th individual is regarded as the fitness of the i-th decision variable. The fitness of each decision variable can be used to measure its contribution to the objective value, i.e., a smaller fitness of a decision variable indicates a lower probability that the decision variable should be set to zero. In the second step, Dec is first set to a $N \times D$ matrix in the same way in the first step, and $Mask$ is set to a $N \times D$ matrix of zeros. Then, for each solution, a random number of decision variables are set to 1 according to their fitness. Finally, the initial population P with N solutions is generated via multiplying Dec by $Mask$.

The other key component of SparseEA, i.e., genetic operator, is presented in Algorithm 2, which is composed of generating the *mask* of offsprings and generating the *dec* of offsprings. To be specific, two parents **p** and **q** are randomly selected from P' to generate an offspring **o** in each turn. Aiming to generate the binary vector *mask* of **o**, a uniformly distributed random number in [0, 1] is firstly generated, if $rand()$ is smaller than 0.5, two decision variables from the nonzero elements in $\mathbf{p}.mask \cap \overline{\mathbf{q}.mask}$ are randomly selected, and the element with bigger fitness is set to 0. Otherwise, two decision variables from the nonzero elements in $\overline{\mathbf{p}.mask} \cap \mathbf{q}.mask$ are randomly selected, and the element with smaller fitness is set to 1. Afterwards, $\mathbf{o}.mask$ is mutated by either the following two operations with the same probability: selecting two elements from the nonzero elements in $\mathbf{o}.mask$, and setting the one with bigger fitness to 0; or selecting two elements from the nonzero elements in $\overline{\mathbf{o}.mask}$, and setting the one with a smaller fitness to 1. In short, SparseEA flips one zero element or nonzero element in the binary vector with the same probability, and the element to be flipped is decided based on the fitness of each decision variable. Subsequently, simulated binary crossover [32] and polynomial mutation [33] are performed to generate the real vector *dec* of offspring **o**, and if the decision variables are binary numbers, *dec* is simply set to a vector of ones. For more details about SparseEA, the readers are referred to [21].

SparseEA2

We see that in the genetic operators of SparseEA, most of the efforts are put forward to the generation of *mask*, while few

Algorithm 2: Genetic operator of SparseEA

```
    Input: $P'$ (parent individuals), $Fit$ (fitness of decision
           variables)
    Output: $O$ (offspring individuals)
1   $O \leftarrow$ Null;
2   while $P'$ is not empty do
3       [$\mathbf{p}$, $\mathbf{q}$] $\leftarrow$ Randomly select two parents from $P'$ and
        remove them from $P'$;
4       //Generating the $mask$ of offspring $\mathbf{o}$
5       $\mathbf{o}.mask \leftarrow \mathbf{p}.mask$;
6       //Crossover of $mask$
7       if $rand() < 0.5$ then
8           Randomly select two decision variables from the
            nonzero elements in $\mathbf{p}.mask \cap \overline{\mathbf{q}.mask}$;
9           Set the element with bigger fitness in $\mathbf{o}.mask$ to 0;
10      else
11          Randomly select two decision variables from the
            nonzero elements in $\overline{\mathbf{p}.mask} \cap \mathbf{q}.mask$;
12          Set the element with smaller fitness in $\mathbf{o}.mask$ to 1;
13      //Mutation of $mask$
14      if $rand() < 0.5$ then
15          Randomly select two decision variables from the
            nonzero elements in $\mathbf{o}.mask$;
16          Set the element with bigger fitness in $\mathbf{o}.mask$ to 0;
17      else
18          Randomly select two decision variables from the
            nonzero elements in $\overline{\mathbf{o}.mask}$;
19          Set the element with smaller fitness in $\mathbf{o}.mask$ to 1;
20      //Generating the $dec$ of offspring $\mathbf{o}$
21      if the decision variables are real numbers then
22          $\mathbf{o}.dec \leftarrow$ Perform simulated binary crossover and
            polynomial mutation based on $\mathbf{p}.dec$ and $\mathbf{q}.dec$;
23      else
24          $\mathbf{o}.dec \leftarrow$ Vector of ones;
25      $O \leftarrow O \cap \{\mathbf{o}\}$;
26  return $O$;
```

attentions have been paid to the generation of dec, i.e., simply performing simulated binary crossover and polynomial mutation based on $\mathbf{p}.dec$ and $\mathbf{q}.dec$. Out of this consideration, we have the following two concerns:

1 On the one hand, when the position of the flipped binary variable and the positions where crossover and mutation take place in the real vector are not consistent, the real part of the key nonzero variables may not be optimized at all, thus the population can hardly converge to the sparse Pareto optimal set within a limit budget of function evaluations.
2 On the other hand, since only one binary variable is flipped each time, and the key nonzero variables only account for a small proportion in all decision variables. Executing genetic operators on each real variable with the same probability may be a kind of waste.

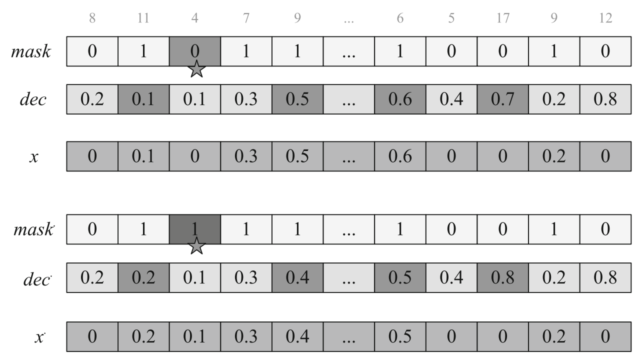
8 11 4 7 9 ... 6 5 17 9 12
mask 0 1 0 1 1 ... 1 0 0 1 0
dec 0.2 0.1 0.1 0.3 0.5 ... 0.6 0.4 0.7 0.2 0.8
x 0 0.1 0 0.3 0.5 ... 0.6 0 0 0.2 0
mask' 0 1 1 1 1 ... 1 0 0 1 0
dec' 0.2 0.2 0.1 0.3 0.4 ... 0.5 0.4 0.8 0.2 0.8
x' 0 0.2 0.1 0.3 0.4 ... 0.5 0 0 0.2 0

Fig. 3 An example of the mutation process of SparseEA, in which x and $x^{\cdot}$ are the parent individual and offspring individual, $mask$, dec, $mask^{\cdot}$, and $dec^{\cdot}$ are the binary vectors and real vectors of x and $x^{\cdot}$, respectively. The light gray numbers above the table represent the fitness of each decision variable, and the highlighted cells denote the positions where mutation takes place

To explain our concerns clearly, Fig. 3 presents the mutation process of SparseEA. We know that when the uniformly distributed random number is not smaller than 0.5, SpareseEA randomly select two decision variables from the nonzero elements in $\overline{\mathbf{o}.mask}$ and set the one with smaller fitness to 1. Supposing that the randomly selected two decision variables are the first one and the third one, according to the fitness scores of decision variables which keep unchanged during the whole optimization process, the third element which is marked by a star will be flipped to 1, and thus we obtain the binary vector $mask^{\cdot}$ of offspring individual $x^{\cdot}$. As for the real vector dec, each variable has a equal probability of $1/D$ to be mutated, and in this example, we present the mutated real variables in the highlighted cells.

It can be observed that since the positions of elements to be mutated in binary vector $mask$ and real vector dec are not consistent with each other, even though the binary variable with smaller fitness (higher probability that the decision variable should be nonzero in the Pareto optimal solutions) can be selected and flipped, its corresponding real variable keeps unchanged during current reproduction process. That is, for solution x, its key nonzero real variable is not optimized at all in current iteration, even if the position of nonzero variable has been found precisely. Besides, despite of that many real variables in dec have been mutated, for the variables that are not related to the nonzero elements in Pareto optimal solutions, the efforts made for them will be in fact a kind of waste.

To address this dilemma, variable grouping techniques are utilized to enhance the connection between $mask$ and dec, thus ensuring when a binary variable is flipped, its corresponding real variable should be optimized at the same time. Specifically, instead of performing crossover and mutation operators on binary vector first, simulated binary crossover is executed on real variables, then we divide the obtained vari-

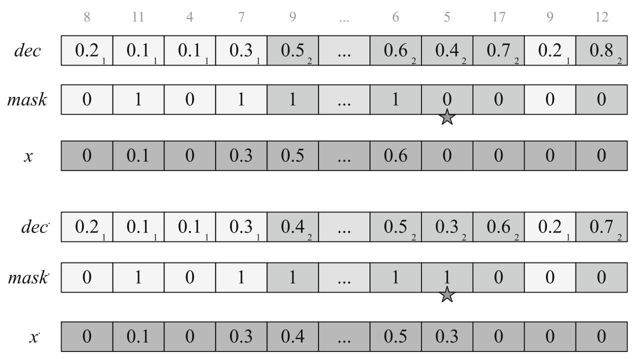

Fig. 4 An example of the mutation process of SparseEA2, in which x and x' are the parent individual and offspring individual, *mask*, *dec*, *mask'*, and *dec'* are the binary vectors and real vectors of x and x', respectively. The light gray numbers above the table represent the fitness of each decision variable, and the number with smaller font size in each cell denotes the grouping index

ables into different groups and randomly select one group to perform mutation operation on the basis of variable grouping techniques in [17]. Figure 4 shows the mutation process of the proposed SparseEA2, we first divide real variables after simulated binary crossover into different groups (two groups in this example) via ordered grouping [29], then one group of variables is randomly selected. Supposing that the second group of variables is selected, after those variables are changed with the same mutation amount, the binary variables having the same positions to the mutated real variables are picked out, and we call the set of those binary variables as $PreMask$. Afterwards, a uniformly distributed random number in [0, 1] is generated, and if the random number is smaller than 0.5, two variables from the nonzero elements in $PreMask$ are randomly selected, and the one with bigger fitness is set to 0. Otherwise, two variables from the nonzero elements in $\overline{PreMask}$ are randomly selected, and the one with smaller fitness is set to 1. Supposing that the variables whose fitness are 5 and 12 are selected, the former one which is marked by a star will be flipped from 0 to 1, as its fitness is smaller.

It can be observed that the connections between *mask* and *dec* can indeed be enhanced through the operations elaborated above. As a result, without sacrificing the effect of sparsity maintenance, as long as one binary variable is flipped, its corresponding real variable should be optimized at the same time. Besides, since the attention is only paid to one group of variables each time, the efforts devoted to the mutation process can also be saved, which is very meaningful when only one binary variable in *mask* is flipped in each iteration. The genetic operators of SparseEA are replaced with the ones elaborated above, and we call the new algorithm as SparseEA2. To validate the effectiveness of SparseEA2, we run SparseEA2 on SMOP1 and SMOP3 with 1000 decision variables. Figure 5 shows the parallel coordinates plot of the decision variables of solutions obtained by SparseEA2, compared to the results as shown in Fig. 1, we see that the first 100 key variables that should be set to nonzero in the Pareto optimal solutions are optimized more sufficiently, and the IGD values attached in the top right corner of each sub-figure are also much smaller than the ones obtained by SparseEA, MOEA/PSL, and PM-MOEA.

Experimental studies on benchmark problems

Four different types of MOEAs tailored for general LSMOPs and three MOEAs customized for sparse LSMOPs are compared with SparseEA2, including CCGDE3 [8], LMEA [10], WOF-SMPSO [11], GLMO [17], SparseEA [21], MOEA/PSL [22], and PM-MOEA [25]. In this section, we investigate the performances of these eight algorithms on eight benchmark problems, i.e., SMOP1-SMOP8 [21], which have various landscape functions, and are very suitable for assessing the performance of existing MOEAs in obtaining sparse Pareto optimal solutions.

Experimental settings for benchmark problems

Algorithms: For CCGDE3, the number of species is set to 2, and random grouping is adopted. For LMEA, the number of selected solutions for decision variables clustering is set to 2, the number of perturbations on each solution is set to 4, and the number of selected solutions for decision variable interaction analysis is set to 5. For WOF-SMPSO, the number of function evaluations for each optimization of the original problem is set to 1000, while the number of evaluations for each optimization of the transformed problem is set to 500, the number of chosen solutions is set to 3, the number of groups is set to 4, and ordered grouping is adopted. For GLMO, the number of groups is set to 4, NSGA-II is adopted as the underlying optimizer and ordered grouping is employed. For PM-MOEA, the population size and the number of generations of its evolutionary pattern mining approach are set to 20 and 10, respectively. For SparseEA2, the number of groups is set to 4, and ordered grouping is used. In LMEA, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2, the simulated binary crossover and polynomial mutation are employed to produce offsprings, the probability of crossover and mutation are set to 1 and $1/D$, where D is the number of decision variables, and the distribution index of both crossover and mutation is set to 20. In CCGDE3, the DE operator and polynomial mutation are used for offspring generation, where the control parameters are set to $CR = 1$, $F = 0.5$, $pm = 1/D$, and $\eta = 20$. In WOF-SMPSO, the PSO operator and polynomial mutation are employed.

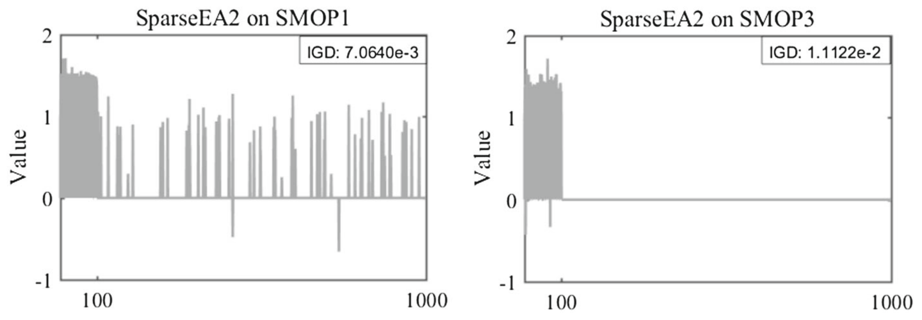

Fig. 5 Parallel coordinates plot of the decision variables of solutions obtained by SparseEA2 on SMOP1 and SMOP3 with 1000 variables, the sparsity is set to 0.1. The value in the top right corner of each sub-figure represents the median IGD obtained by SparseEA2 over 30 runs

Problems: For SMOP1 - SMOP8, the number of objectives is set to 2, the number of decision variables is set to 1000, 2000, and 5000, and the sparsity of Pareto optimal solutions is set to 0.1, which denotes the ratio of nonzero elements in the decision variables.

Stopping criteria and population size: The maximum number of function evaluations is adopted as the stopping criteria, which is set to $100 \times D$ for each MOEA. The population size is set to 100.

Performance metrics: The inverted generational distance (IGD) [36] is adopted to measure each obtained solution set, and roughly 10000 reference points on each Pareto front are sampled to calculate the IGD value. We perform 30 independent runs for each MOEA on each problem, and the Wilcoxon rank-sum test with a significance level of 0.05 is adopted to perform statistical analysis. Here, we note that all the experimental studies in this paper are conducted on the PlatEMO [1] [34]

Experimental results on benchmark problems

Table 1 shows the IGD values obtained by CCGDE3, LMEA, WOF-SMPSO, GLMO, SparseEA, MOEA/PSL, PM-MOEA and the proposed SparseEA2 on SMOP1-SMOP8 with 1000, 2000, and 5000 decision variables over 30 runs. For the 24 benchmark problems, while MOEA/PSL obtains the best results on 4 problems, PM-MOEA and SparseEA2 perform the best on 10 problems, respectively. Based on the Wilcoxon rank-sum test with a significance level of 0.05, compared to SparseEA2, the statistical analysis results of the other seven algorithms are 0/24//0, 0/24/0, 0/23/1, 0/24/0, 1/21/2, 4/17/3, and 9/14/1. Thus, it can be concluded that SparseEA2 exhibits the best performance over the other 7 algorithms on the eight sparse benchmark LSMOPs.

Besides, It is evident that the four MOEAs customized for sparse LSMOPs perform obviously better than the other four MOEAs tailored for general LSMOPs on these 24 benchmark problems. One may doubt that sparse optimization is a special optimization problem, and the algorithms that can solve general LSMOPs should also be able to solve sparse LSMOPs. In fact, this question can be answered from the following three viewpoints. Firstly, existing MOEAs for general LSMOPs mostly generate the initial population in a random manner within the large search space, and the generated initial population is usually far from the sparse optimal Pareto sets. Secondly, without customizing special genetic operators, for each decision variable, existing MOEAs for general LSMOPs usually traverse each legal value with the same probability, which is very inefficient. Thirdly, the computational budget is usually limited, e.g., $100 \times D$ function evaluations for benchmark problems in this paper. Under the above three conditions, existing MOEAs for general LSMOPs can hardly converge to the sparse optimal Pareto fronts within the limited computational budget.

Figure 6 shows the Pareto optimal fronts with median IGD values obtained by the eight compared algorithms on SMOP5 and SMOP8 with 5000 decision variables over 30 runs. For SMOP5, we see that SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 exhibit the best results that have no obvious differences, WOF-SMPSO and GLMO obtain similar results that are worse than the four sparse MOEAs, while LMEA performs worst. For SMOP8, firstly, WOF-SMPSO, GLMO, CCGDE3 and LMEA are significantly outperformed by the other four MOEAs customized for sparse LSMOPs. Secondly, for the four sparse MOEAs, MOEA/PSL and SparseEA2 obtain similar results that are better than the other two algorithms, while SparseEA performs worst within the four sparse MOEAs.

Figure 7 shows the Pareto optimal sets with median IGD values obtained by the eight compared algorithms on SMOP8 with 2000 decision variables over 30 runs. Firstly, we see that the decision variables of solutions obtained by CCGDE3 and LMEA cover the whole search space, which explains why these two algorithms obtain the worst results. Secondly, for WOF-SMPSO and GLMO which exhibit slightly worse results than the remaining four sparse MOEAs, even though the decision variables of solutions obtained by them are very close to zero, there still exist huge differences with the sparse Pareto optimal solution set. Thirdly, for the four sparse MOEAs which perform obviously better than the other four

[1] The source codes of algorithms and problems used in this paper can be found from https://github.com/BIMK/PlatEMO.

Table 1 IGD values obtained by CCGDE3, LMEA, WOF-SMPSO, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 on SMOP1-SMOP8 with 1000, 2000, and 5000 decision variables, where the best result in each row is highlighted. $'+'$, $'-'$ and $'\approx'$ indicate that the result is significantly better, significantly worse, and statistically similar to that obtained by SparseEA2

Problem	*Dec*	CCGDE3	LMEA	WOF-SMPSO	GLMO	SparseEA	MOEA/PSL	PM-MOEA	SparseEA2
SMOP1	1000	8.8126e-1 (8.88e-2) −	1.5866e+0 (1.81e-2) −	6.6985e-2 (1.63e-2) −	7.6410e-2 (1.22e-3) −	2.5340e-2 (2.55e-3) −	1.4568e-2 (3.09e-3) −	2.1027e-2 (2.85e-3) −	**7.0640e-3 (3.76e-4)**
	2000	9.6182e-1 (1.03e-1) −	1.6167e+0 (1.42e-2) −	5.5703e-2 (1.05e-2) −	7.8585e-2 (5.28e-4) −	3.2900e-2 (1.99e-3) −	1.6337e-2 (2.42e-3) −	2.7313e-2 (1.97e-3) −	**1.2479e-2 (1.63e-3)**
	5000	9.5750e-1 (1.07e-1) −	1.6335e+0 (6.47e-3) −	4.5108e-2 (5.22e-3) −	7.9465e-2 (2.63e-4) −	3.8251e-2 (1.37e-3) −	**1.7868e-2 (3.82e-3)** +	3.4758e-2 (1.60e-3) −	3.2757e-2 (1.67e-3)
SMOP2	1000	1.8044e+0 (1.44e-1) −	2.2268e+0 (7.54e-3) −	4.5582e-1 (1.51e-1) −	6.9075e-1 (4.75e-2) −	6.5692e-2 (4.44e-3) −	3.6419e-2 (6.13e-3) −	5.2793e-2 (5.31e-3) −	**1.4395e-2 (3.32e-3)**
	2000	1.8346e+0 (1.27e-1) −	2.2359e+0 (6.60e-3) −	3.4336e-1 (1.36e-1) −	7.4145e-1 (7.58e-2) −	8.5930e-2 (3.34e-3) −	4.4748e-2 (6.05e-3) −	6.5996e-2 (5.05e-3) −	**2.8198e-2 (3.16e-3)**
	5000	1.8289e+0 (1.53e-1) −	2.2484e+0 (3.71e-3) −	1.9892e-1 (1.01e-1) −	7.6820e-1 (6.61e-2) −	1.0162e-1 (2.87e-3) −	**5.1891e-2 (8.05e-3)** +	8.5650e-2 (2.91e-3) −	7.3213e-2 (3.26e-3)
SMOP3	1000	2.0253e+0 (7.87e-2) −	2.5887e+0 (1.15e-2) −	6.6731e-1 (1.74e-3) −	7.0279e-1 (9.48e-4) −	2.8654e-2 (2.72e-3) −	1.0167e-1 (3.26e-1) −	2.2817e-2 (4.59e-3) −	**1.1122e-2 (1.58e-3)**
	2000	2.0124e+0 (6.76e-2) −	2.5990e+0 (5.75e-3) −	6.6337e-1 (1.09e-3) −	7.0224e-1 (3.27e-4) −	3.6493e-2 (2.39e-3) −	**1.3851e-2 (1.96e-3)** +	1.0401e-1 (2.32e-1) −	1.8073e-2 (2.05e-3)
	5000	2.0765e+0 (7.27e-2) −	2.6095e+0 (3.47e-3) −	6.6047e-1 (6.62e-4) −	7.1528e-1 (7.56e-2) −	4.4340e-2 (1.40e-3) −	**2.8029e-2 (2.89e-2)** +	3.4359e-2 (1.44e-3) −	3.2576e-2 (2.10e-3)
SMOP4	1000	9.0041e-1 (9.57e-2) −	1.1039e+0 (5.93e-3) −	1.1345e-1 (6.81e-2) −	2.2769e-1 (5.64e-2) −	4.6729e-3 (2.13e-4) +	5.0270e-3 (4.41e-4) −	**4.0961e-3 (5.66e-5)** +	4.7950e-3 (2.08e-4)
	2000	9.7158e-1 (9.98e-2) −	1.1070e+0 (3.77e-3) −	8.7456e-2 (7.46e-2) −	2.8920e-1 (7.20e-2) −	4.8226e-3 (2.66e-4) ≈	4.9715e-3 (3.94e-4) −	**4.0683e-3 (6.27e-5)** +	4.7413e-3 (2.72e-4)
	5000	9.7264e-1 (1.25e-1) −	1.1117e+0 (2.50e-3) −	3.2693e-2 (5.25e-2) ≈	3.0110e-1 (6.00e-2) −	4.8123e-3 (3.45e-4) ≈	4.8218e-3 (2.30e-4) ≈	**4.0764e-3 (6.09e-5)** +	4.8085e-3 (2.26e-4)
SMOP5	1000	6.3908e-1 (5.33e-2) −	1.0755e+0 (9.10e-3) −	3.4907e-1 (1.41e-3) −	3.4949e-1 (3.63e-4) −	5.8762e-3 (2.84e-4) −	7.0848e-3 (4.88e-4) −	**4.6732e-3 (1.72e-4)** +	5.2556e-3 (2.37e-4)
	2000	6.4593e-1 (4.84e-2) −	1.0913e+0 (1.09e-2) −	3.4819e-1 (9.51e-4) −	3.4945e-1 (3.63e-4) −	6.0142e-3 (3.14e-4) −	7.2627e-3 (4.75e-4) −	**4.6264e-3 (1.06e-4)** +	5.1980e-3 (2.61e-4)
	5000	6.5622e-1 (4.44e-2) −	1.1026e+0 (5.16e-3) −	3.4799e-1 (7.04e-4) −	3.4954e-1 (3.36e-4) −	6.0042e-3 (2.75e-4) −	7.2858e-3 (3.85e-4) −	**4.7953e-3 (1.03e-4)** +	5.1769e-3 (3.49e-4)
SMOP6	1000	2.9715e-1 (2.77e-2) −	4.9404e-1 (3.64e-3) −	2.2532e-2 (4.50e-3) −	1.4484e-2 (6.15e-4) −	7.2663e-3 (3.98e-4) −	7.5967e-3 (6.98e-4) −	**4.8109e-3 (1.61e-4)** +	6.6149e-3 (3.17e-4)
	2000	3.0501e-1 (3.27e-2) −	4.9955e-1 (3.82e-3) −	1.7571e-2 (2.66e-3) −	1.4767e-2 (7.71e-4) −	7.3921e-3 (3.17e-4) −	8.2052e-3 (6.46e-4) −	**4.9399e-3 (1.66e-4)** +	6.8737e-3 (3.77e-4)
	5000	3.3183e-1 (3.59e-2) −	5.0365e-1 (1.97e-3) −	1.3859e-2 (2.88e-3) −	1.4685e-2 (8.57e-4) −	7.9026e-3 (1.94e-4) −	7.8546e-3 (4.24e-4) −	**5.2910e-3 (1.20e-4)** +	6.9830e-3 (2.24e-4)
SMOP7	1000	1.5156e+0 (5.06e-2) −	2.9411e+0 (2.82e-2) −	1.5267e-1 (5.43e-3) −	1.5722e-1 (3.27e-4) −	8.5253e-2 (8.40e-3) −	8.4173e-2 (7.98e-2) −	7.0724e-2 (9.19e-3) −	**1.0709e-2 (2.41e-3)**
	2000	1.6013e+0 (5.98e-2) −	3.0169e+0 (2.07e-2) −	1.4379e-1 (1.79e-2) −	1.5733e-1 (3.10e-4) −	1.0506e-1 (5.59e-3) −	1.0238e-1 (8.59e-2) −	9.0782e-2 (6.93e-3) −	**1.4226e-2 (2.58e-3)**
	5000	1.6656e+0 (8.92e-2) −	3.0650e+0 (1.67e-2) −	1.3188e-1 (2.78e-2) −	1.5751e-1 (2.21e-4) −	1.2721e-1 (4.04e-3) −	9.4797e-2 (7.58e-2) ≈	1.1326e-1 (3.70e-3) −	**6.3160e-2 (4.58e-3)**
SMOP8	1000	3.4557e+0 (6.64e-2) −	3.7067e+0 (7.75e-3) −	6.0281e-1 (1.58e-1) −	7.2638e-1 (1.58e-1) −	2.4367e-1 (1.55e-2) −	2.3849e-1 (4.80e-2) −	**1.9630e-1 (9.98e-3)** ≈	1.9781e-1 (1.45e-2)
	2000	3.3090e+0 (5.68e-2) −	3.7247e+0 (3.83e-3) −	5.4849e-1 (4.38e-2) −	6.4276e-1 (6.03e-2) −	2.8719e-1 (1.19e-2) −	2.5695e-1 (4.96e-2) −	2.3208e-1 (1.22e-2) −	**2.1496e-1 (7.23e-3)**
	5000	3.3183e+0 (4.93e-2) −	3.7371e+0 (2.95e-3) −	5.2674e-1 (4.27e-2) −	5.8223e-1 (1.91e-2) −	3.2861e-1 (8.48e-3) −	2.5661e-1 (4.70e-2) ≈	2.8396e-1 (1.03e-2) −	**2.5084e-1 (5.83e-3)**
+/ − / ≈		0/24/0	0/24/0	0/23/1	0/24/0	1/21/2	4/17/3	9/14/1	

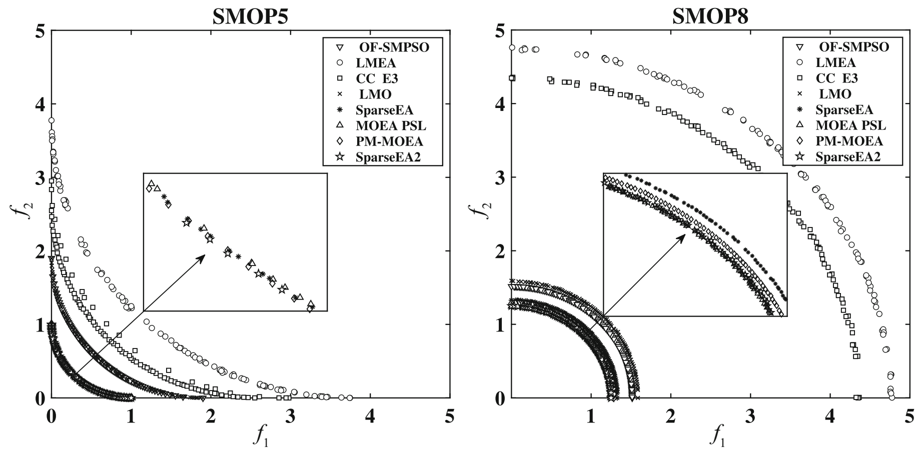

Fig. 6 Pareto optimal fronts with median IGD values obtained by WOF-SMPSO, LMEA, CCGDE3, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 on SMOP5 and SMOP8 with 5000 decision variables over 30 runs

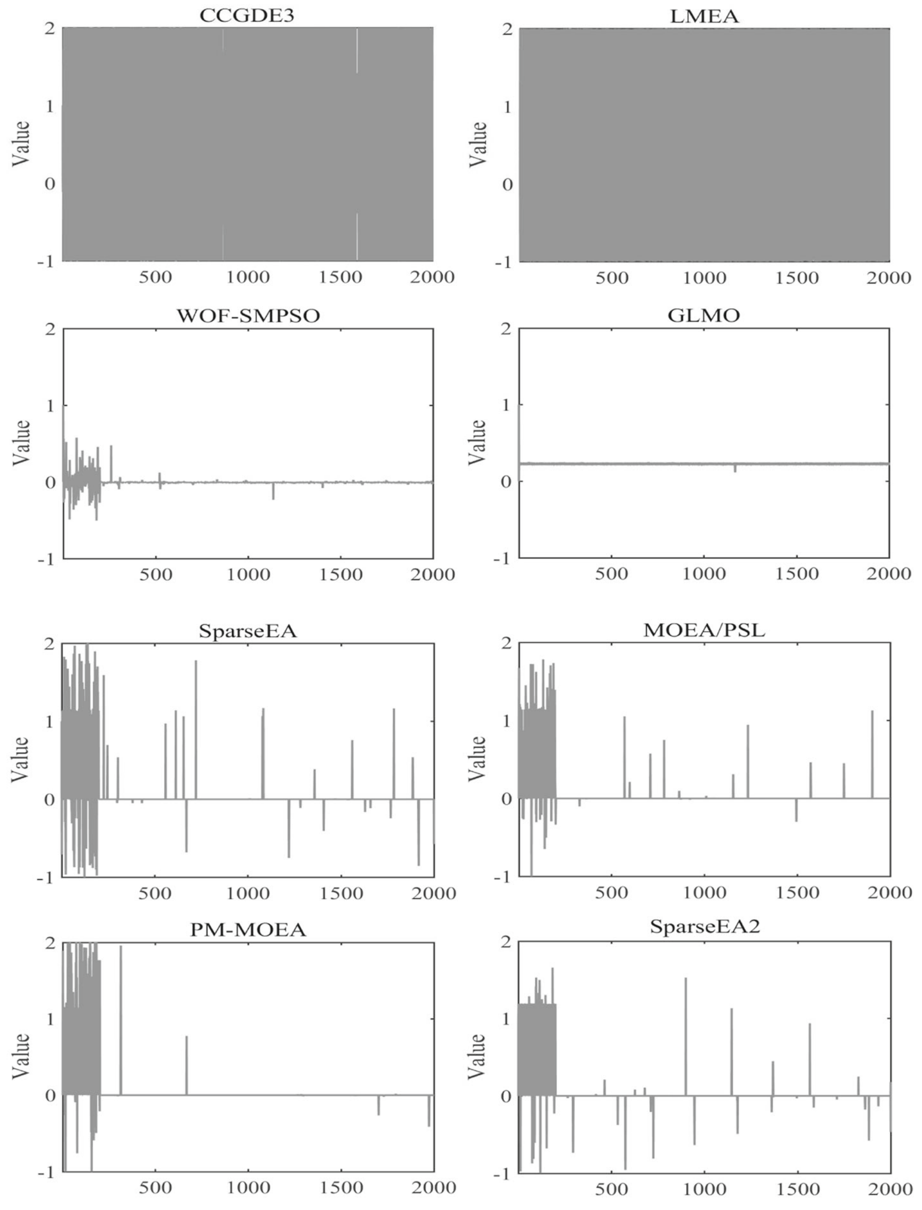

Fig. 7 Pareto optimal sets with median IGD values obtained by CCGDE3, LMEA, WOF-SMPSO, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 on SMOP8 with 2000 decision variables over 30 runs

algorithms, most of the decision variables obtained by them are very sparse. Most importantly, the key nonzero variables which affect the function fitness obtained by SparseEA2 are optimized more sufficiently than the other three sparse MOEAs without sacrificing the effect of sparsity maintenance, which can properly explains why SparseEA2 obtains better results than SparseEA, MOEA/PSL, and PM-MOEA on SMOP8.

Here, it is necessary to analyze why the decision variables of solutions obtained by CCGDE3 and LMEA cover the whole search space. On the one hand, the performance of CCGDE3 and LMEA depend heavily on the precision of decision variables grouping or classification, and they work well on problems with separable variables. However, the landscape functions of SMOPs are complicated, a solution can be Pareto optimal when the first fixed number of decision variables are nonzero and the remaining variables are all zero, besides, there exist strong interactions between these variables. On the other hand, CCGDE3 is nontrivial to select the proper cooperators for executing function evaluations and LMEA consumes too many function evaluations to conduct the decision variable analysis. Under the three conditions previously analyzed, the decision variables obtained by these two algorithms are reasonably far from the sparse optimal sets.

Experimental studies on real-world applications

To further validate the superiority of SparseEA2 over the compared MOEAs in solving sparse LSMOPs, in this section, three real-world applications, namely, the neural network training problem [18], the portfolio optimization problem [19], and the sparse signal reconstruction problem [35] are selected to conduct a deeper experimental study.

Experimental settings for real-world problems

Algorithms: The compared algorithms and the corresponding parameter settings of each algorithm are kept the same as the last section.

Problems: For each real-world applications, three datasets are used, thus, there are in total nine problems empirically tested in this section. Table 2 presents the details of each problem, where NN, PO, and SR denote the neural network training problem, the portfolio optimization problem, and the sparse signal reconstruction problem, respectively.

Stopping criteria and population size: The maximum number of function evaluations is adopted as the stopping criteria, which is set to 2.0×10^4, 4.0×10^4 and 1.0×10^5 for problems with approximately 1000, 2000 and 5000 decision variables. The population size is set to 50.

Table 2 Datasets of three sparse LSMOPs in real-world applications

Neural network training problem	Type of variables	No. of variables	Dataset	No. of samples	No. of features	No. of classes
NN1	Real	1241	Connectionist bench sonar[2]	208	60	2
NN2		2041	Hill valley[2]	606	100	2
NN3		6241	LSVT voice rehabilitation[2]	126	310	2
Portfolio optimization problem	**Type of variables**	**No. of variables**	**Dataset**	**No. of instruments**	**Length of each instrument**	
PO1	Real	500	EURCHF[3]	500	50	
PO2		1000	EURCHF[3]	1000	50	
PO3		5000	EURCHF[3]	5000	50	
Sparse signal reconstruction problem	**Type of variables**	**No. of variables**	**Dateset**	**Length of signal**	**Length of received signal**	**Sparsity of signal**
SR1	Real	1024	Synthetic [35]	1024	480	260
SR2		2048	Synthetic [35]	2048	960	520
SR3		5120	Synthetic [35]	5120	2400	1300

2. https://archive.ics.uci.edu/ml
3. https://www.metatrader5.com/en

Performance metrics: The HV [37] indicator with a reference point (1,1) is employed to measure the results on real-world applications, and the Wilcoxon rank-sum test with a significance level of 0.05 is also adopted to perform the statistical analysis.

Experimental results on real-world problems

Table 3 shows the HV values obtained by CCGDE3, LMEA, WOF-SMPSO, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and the proposed SparseEA2 on the three real-world applications. We see that SparseEA2 exhibits the best results on the nine test instances. Compared to SparseEA2, the statistical analysis results of the other seven algorithms are 0/9/0, 0/9/0, 0/9/0, 0/9/0, 0/8/1, 0/7/2, and 0/8/1. Besides, it can be found that the orders of magnitude of HV values obtained by the sparse MOEAs are mostly e-1, while it is e-2 for the MOEAs tailored for general LSMOPs, which indicates that there exist big differences between general MOEAs and sparse MOEAs in solving real-world applications with sparse optimal solutions. Since the reasons why the algorithms performing well on general LSMOPs are not able to solve sparse LSMOPs have be analyzed in the former section, we do not repeat it anymore.

Figure 8 shows the Pareto optimal fronts with median HV values obtained by the eight compared algorithms on the neural network training problem and the sparse signal reconstruction problem with approximately 5000 decision variables over 30 runs. For the neural network training problem, the four sparse MOEAs, i.e., SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 exhibit obviously better results than the other four algorithms tailored for general LSMOPs. Within the four sparse MOEAs, SparseEA2 obtains the best result. For the sparse signal reconstruction problem, the results obtained by CCGDE3, GLMO, MOEA/PSL, LMEA, and WOF-SMPSO are outperformed by these obtained by the remaining three algorithms. While within the remaining three sparse MOEAs, SparseEA2 obtains the best result, i.e., finding the most sparse signal for the lowest loss.

Figure 9 shows the Pareto optimal sets with median HV values obtained by the eight compared algorithms on the portfolio optimization problem with 5000 decision variables over 30 runs. Firstly, for the four MOEAs tailored for general LSMOPs, the Pareto optimal sets obtained by them are not sparse at all. Secondly, within the four MOEAs customized for sparse LSMOPs, the sparsity of the Pareto optimal set obtained by PM-MOEA is much bigger than that of the other three sparse MOEAs, while SparseEA2 obtains the most sparse Pareto optimal set.

Table 3 HV values obtained by CCGDE3, LMEA, WOF-SMPSO, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 on nine real-world test instances, where the best result in each row is highlighted. $'+'$, $'-'$ and $'\approx'$ indicate that the result is significantly better, significantly worse, and statistically similar to that obtained by SparseEA2, respectively

Problem	*Dec*	CCGDE3	LMEA	WOF-SMPSO	GLMO	SparseEA	MOEA/PSL	PM-MOEA	SparseEA2
NN	1241	8.0550e-2 (9.72e-4) −	3.1123e-1 (1.13e-2)−	3.1473e-1 (1.30e-2) −	3.0896e-1 (1.30e-2) −	8.6351e-1 (1.24e-2) ≈	8.6562e-1 (8.99e-3) ≈	8.5383e-1 (8.54e-3)−	**8.6765e-1 (1.69e-2)**
	2041	6.4116e-2 (3.73e-3) −	2.4372e-1 (7.96e-3)−	2.5261e-1 (1.12e-2) −	2.5511e-1 (1.48e-2) −	7.4686e-1 (1.13e-1) −	8.5969e-1 (7.77e-2) ≈	8.4557e-1 (1.02e-1)≈	**8.8643e-1 (6.64e-2)**
	6241	8.9300e-2 (7.59e-4) −	3.2516e-1 (1.30e-2)−	3.3086e-1 (1.88e-2) −	3.3320e-1 (1.63e-2) −	9.2454e-1 (1.56e-2) −	9.2663e-1 (1.69e-2) −	9.2160e-1 (2.14e-2)−	**9.3538e-1 (2.07e-2)**
PO	500	9.2134e-2 (9.64e-5) −	9.1395e-2 (9.98e-5)−	9.2891e-2 (3.71e-4) −	9.4620e-2 (4.51e-4) −	1.2368e-1 (1.21e-3) −	1.0754e-1 (1.08e-2) −	1.2050e-1 (7.87e-3)−	**1.2421e-1 (1.55e-3)**
	1000	9.1614e-2 (6.47e-5) −	9.1166e-2 (6.77e-5)−	9.2459e-2 (5.97e-4) −	9.3743e-2 (4.74e-4) −	1.2356e-1 (1.73e-3) −	1.0476e-1 (1.28e-2) −	1.2211e-1 (6.13e-3)−	**1.2436e-1 (1.53e-3)**
	5000	9.1134e-2 (1.80e-5) −	9.0960e-2 (1.09e-5)−	9.1914e-2 (2.44e-4) −	9.1840e-2 (9.74e-5) −	1.2368e-1 (1.84e-3) −	1.1126e-1 (1.47e-2) −	1.2074e-1 (6.94e-3)−	**1.2441e-1 (1.59e-3)**
SR	1024	0.0000e+0 (0.00e+0) −	1.0808e-1 (1.84e-2) −	1.9832e-1 (2.36e-2) −	8.3825e-2 (7.49e-3) −	3.3275e-1 (7.42e-3) −	1.0980e-1 (2.88e-2) −	2.8115e-1 (1.60e-2)−	**3.4431e-1 (9.61e-3)**
	2048	0.0000e+0 (0.00e+0) −	1.1481e-1 (1.20e-2) −	2.0572e-1 (1.61e-2) −	8.3693e-2 (5.90e-3) −	3.3762e-1 (6.09e-3) −	8.7570e-2 (3.08e-2) −	2.9087e-1 (8.54e-3)−	**3.4449e-1 (5.21e-3)**
	5120	0.0000e+0 (0.00e+0) −	1.1410e-1 (1.40e-2) −	1.9253e-1 (1.08e-2) −	7.9221e-2 (8.46e-3) −	3.3395e-1 (3.28e-3) −	5.0119e-2 (3.17e-2) −	2.9957e-1 (3.52e-3)−	**3.5382e-1 (3.45e-3)**
$+/-/\approx$		0/9/0	0/9/0	0/9/0	0/9/0	0/8/1	0/7/2	0/8/1	

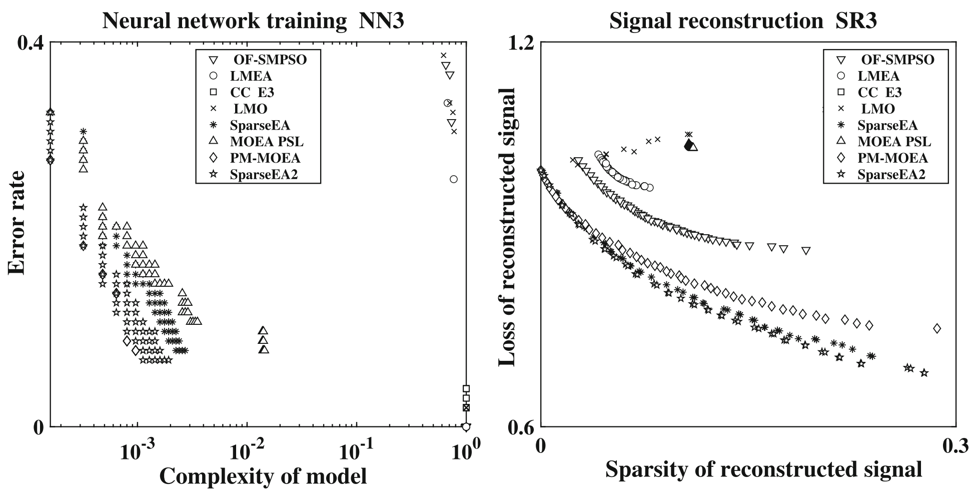

Fig. 8 Pareto optimal fronts with median HV values obtained by WOF-SMPSO, LMEA, CCGDE3, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 on NN3 and SR3 with approximately 5000 decision variables over 30 runs

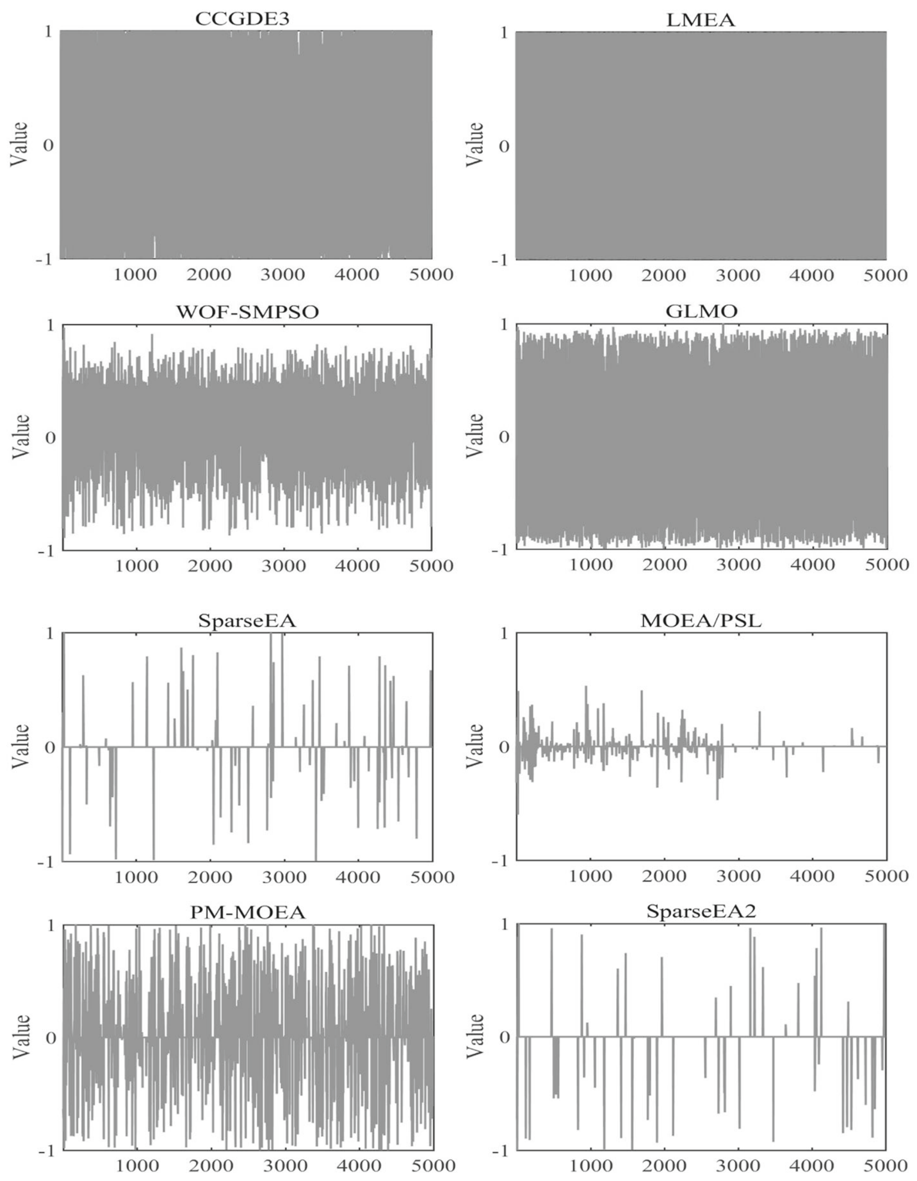

Fig. 9 Pareto optimal sets with median HV values obtained by CCGDE3, LMEA, WOF-SMPSO, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 on the portfolio optimization problem with 5000 decision variables over 30 runs

Table 4 Runtime (in second) of CCGDE3, LMEA, WOF-SMPSO, GLMO, SparseEA, MOEA/PSL, PM-MOEA, and SparseEA2 on SMOP1-SMOP8, neural network training, portfolio optimization, and sparse signal reconstruction. The least runtime in each row is highlighted, the longest runtime in each row is also shown in bold italic, while the number in each bracket denotes the rank of the different algorithm on the same test instance

Problem	*Dec*	CCGDE3	LMEA	WOF-SMPSO	GLMO	SparseEA	MOEA/PSL	PM-MOEA	SparseEA2
SMOP1-SMOP8 (average)	1000	**9.7315E+00 (1)**	***8.6636E+02 (8)***	2.0982E+01 (2)	3.5760E+01 (3)	4.8030E+01 (4)	1.3883E+02 (6)	5.0326E+02 (7)	1.1369E+02 (5)
SMOP1-SMOP8 (average)	2000	**3.4637E+01 (1)**	***6.1369E+03 (8)***	7.6530E+01 (2)	8.0911E+02 (6)	6.8912E+02 (5)	6.1788E+02 (4)	3.0706E+03 (7)	5.2629E+02 (3)
SMOP1-SMOP8 (average)	5000	**3.0074E+02 (1)**	***8.1106E+04 (8)***	4.4916E+02 (2)	1.1453E+03 (3)	2.4434E+03 (4)	2.5077E+03 (5)	1.3427E+04 (7)	5.6937E+03 (6)
NN1	1241	**2.4938E+01 (1)**	1.1892E+02 (7)	2.6319E+01 (2)	4.0221E+01 (3)	4.1513E+01 (4)	9.7244E+01 (6)	7.5029E+01 (5)	***1.5187E+02 (8)***
NN2	2041	1.4463E+02 (2)	5.5843E+02 (7)	**1.4302E+02 (1)**	2.1000E+02 (4)	2.0921E+02 (3)	3.2566E+02 (5)	3.5341E+02 (6)	***9.6244E+02 (8)***
NN3	6241	**2.1296E+02 (1)**	1.1163E+03 (7)	2.4577E+02 (2)	5.1003E+02 (3)	5.4786E+02 (4)	7.7006E+02 (6)	6.7921E+02 (5)	***3.3741E+03 (8)***
PO1	500	4.4806E+00 (3)	2.0947E+01 (5)	1.0888E+01 (4)	**2.4999E+00 (1)**	3.0155E+00 (2)	2.1762E+01 (6)	***1.3521E+02 (8)***	4.3170E+01 (7)
PO2	1000	**1.1487E+01 (1)**	6.1849E+01 (5)	2.7093E+01 (4)	1.5822E+01 (2)	1.6384E+01 (3)	6.6241E+01 (6)	***9.1429E+02 (8)***	6.5572E+02 (7)
PO3	5000	1.7795E+03 (3)	2.2950E+04 (6)	1.9775E+03 (4)	1.6621E+03 (2)	**1.6479E+03 (1)**	2.5723E+03 (5)	***3.4449E+04 (8)***	2.7609E+04 (7)
SR1	1024	**3.3172E+00 (1)**	2.1509E+01 (7)	6.8544E+00 (2)	9.1393E+00 (3)	1.3013E+01 (5)	1.8370E+01 (6)	***5.5813E+01 (8)***	9.2411E+00 (4)
SR2	2048	**3.6111E+01 (1)**	7.8069E+01 (7)	4.7242E+01 (2)	5.3533E+01 (3)	7.2000E+01 (6)	6.4667E+01 (5)	***2.4109E+02 (8)***	6.2651E+01 (4)
SR3	5120	**5.1530E+02 (1)**	1.1244E+03 (6)	6.1277E+02 (2)	7.4665E+02 (4)	1.7618E+03 (7)	7.8948E+02 (5)	***3.0484E+03 (8)***	7.2767E+02 (3)
Average Rank		1.4167	6.7500	2.4167	3.0833	4.0000	5.4167	7.0833	5.8333

Computational efficiency of SparseEA2

Lastly, the computational efficiency of SparseEA2 is compared to the other seven MOEAs. Table 4 lists the runtime (in second) of the eight MOEAs on the benchmark SMOPs and real-world applications. It can be observed that the efficiency of SparseEA2 is worse than the compared algorithms on the neural network training problems and the portfolio optimization problems, competitive to the compared algorithms on benchmark problems with 2000 decision variables and the sparse signal reconstruction problems. Since the initialization strategy of SparseEA2 has a time complexity of O(MD^2), where M and D denotes the number of objectives and decision variables, respectively, the average rank of SparseEA2 is reasonably bigger than that of CCGDE3, WOF-SMPSO, and GLMO. Moreover, compared to SparseEA, the ordered grouping with a time complexity of O(D^2) is additionally performed in each generation, the runtime of SparsEA2 is thus longer than SparseEA on most test problems. While for some problems such as the sparse signal reconstruction problems, SparseEA2 becomes more efficient since its genetic operators generate sparse solutions more efficient, and a sparse solution usually corresponds to cheap objective evaluations. Even though, the average rank of SparsEA2 is still similar to MOEA/PSL, smaller than PM-MOEA and LMEA. In short, the computational efficiency of SparseEA2 is affordable.

Conclusions and future work

In this paper, we have proposed an improved version of SparseEA for solving sparse LSMOPs. The core idea of the proposed algorithm is to enhance the connection between $mask$ and dec with the assistance of variable grouping techniques, thus ensuring that the real part of a decision variable should be optimized at the same time when its binary part is flipped. Besides, since only one binary variable in $mask$ is flipped each time, there is no need to perform mutation operations on each real variable with the same mutation probability. Through enhancing the connection between $mask$ and dec, SparseEA2 can also avoid wasting efforts on the variables that may not be related to the nonzero elements in the Pareto optimal solutions.

In the experimental studies, the proposed algorithm has been compared with four MOEAs tailored for general LSMOPs and three MOEAs customized for sparse LSMOPs on eight benchmark problems as well as three real-world applications with sparse Pareto optimal solutions. The statistical results on those in total 33 test instances have demonstrated the superiority of the proposed algorithm over other MOEAs in solving sparse LSMOPs.

To the best of our knowledge, there are only five MOEAs are specially designed for sparse LSMOPs up to now, where the study on large-scale sparse MOEAs is still in its infancy. Therefore, it deserves more attention in the community of evolutionary computation. For example, we can customize MOEAs for sparse LSMOPs without using the two-layer encoding scheme. Besides, we can adopt more effective environmental selection strategies in SparseEA2 for sparse LSMOPs with many objectives (i.e., many-objective knapsack problems [38]), and we can also combine the proposed algorithm with effective constraint handling techniques for solving sparse constrained LSMOPs (i.e., optimal software product selection problems [39]).

Author contributions Yajie Zhang: Methodology, Writing - original draft Ye Tian: Methodology, Writing - review & editing Xingyi Zhang: Writing - review & editing The results reported in our study do not involve any humans and/or animals

Funding Not applicable

Availability of data and material Not applicable

Declarations

Conflict of interest Not applicable

Code availability Not applicable

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

References

1. Xiao J, Zhang T, Du J, Zhang X (2019) An evolutionary multiobjective route grouping-based heuristic algorithm for large-scale capacitated vehicle routing problems. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2019.2950626
2. He C, Cheng R, Zhang C, Tian Y et al (2020) Evolutionary large-scale multiobjective optimization for ratio error estimation of voltage transformers. IEEE Trans Evol Comput 24(5):868–881
3. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197
4. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11(6):712–731
5. Tian Y, Cheng R, Zhang X, Jin Y (2018) An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility. IEEE Trans Evol Comput 22(4):609–622
6. Wang H, Jiao L, Shang R, He S, Liu F (2015) A memetic optimization strategy based on dimension reduction in decision space. Evol Comput 23(1):69–100
7. Tian Y, Si L, Zhang X, Cheng R, et al (2021) Evolutionary large-scale multi-objective optimization: a survey. ACM Comput Surv
8. Antonio L, Coello Coello C (2013) Use of cooperative coevolution for solving large scale multiobjective optimization problems. In: IEEE Congress on Evolutionary Computation (CEC), pp 2758–2765
9. Ma X, Liu F, Qi Y et al (2016) A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables. IEEE Trans Evol Comput 20(2):275–298
10. Zhang X, Tian Y, Cheng R, Jin Y (2018) A decision variable clustering-based evolutionary algorithm for large-scale many-objective optimization. IEEE Trans Evol Comput 22(99):97–112
11. Zille H, Ishibuchi H, Mostaghim S, Nojima Y (2018) A framework for large-scale multiobjective optimization based on problem transformation. IEEE Trans Evol Comput 22(2):260–275
12. He C, Li L, Tian Y et al (2019) Accelerating large-scale multiobjective optimization via problem reformulation. IEEE Trans Evol Comput 23(6):949–961
13. He C, Cheng R, Yazdani D (2020) Adaptive offspring generation for evolutionary large-scale multiobjective optimization. Syst Man Cybern Syst IEEE Trans. https://doi.org/10.1109/TSMC.2020.3003926
14. Tian Y, Zheng X, Zhang X, Jin Y (2019) Efficient large-scale multiobjective optimization based on a competitive swarm optimizer. IEEE Trans. Cybern 50(8):3696–3708
15. Cheng R, Jin Y (2015) A competitive swarm optimizer for large scale optimization. IEEE Trans Cybern 45(2):191–204
16. Zille H, Mostaghim S (2019) Linear search mechanism for multi- and many-objective optimisation. In: Deb K et al (eds), EMO 2019, LNCS, Springer, Cham 11411:399–410
17. Zille H, Ishibuchi H, Mostaghim S, Nojima Y (2016) Mutation operators based on variable grouping for multi-objective large-scale optimization. In: IEEE Symposium Series on Computational Intelligence (SSCI), Greece. https://doi.org/10.1109/SSCI.2016.7850214
18. Jin Y, Sendhoff B (2008) Pareto-based multiobjective machine learning: an overview and case studies. IEEE Trans Syst Man Cybern Syst 38(3):397–415
19. Lwin K, Qu R, Kendall G (2014) A learning-guided multi-objective evolutionary algorithm for constrained portfolio optimization. Appl Soft Comput J 24:757–772
20. Zhang Y, Tian Y, Zhang X (2021) A comparsion study of evolutionary algorithms on large-scale sparse multi-objective optimization problems. In: Ishibuchi H et al (eds), EMO 2021, LNCS, Springer, 12654:424–437
21. Tian Y, Zhang X, Wang C, Jin Y (2020) An evolutionary algorithm for large-scale sparse multiobjective optimization problems. IEEE Trans Evol Comput 24(2):380–393
22. Tian Y, Lu C, Zhang X et al (2020) Solving large-scale multiobjective optimization problems with sparse optimal solutions via unsupervised neural networks. IEEE Trans Cybern 51(6):3115–3128

23. Fischer A, Igel C (2012) An introduction to restricted Boltzmann machines. In: the Iberoamerican congress on pattern recognition, Springer, 14–36
24. Vincent P, Larochelle H, Bengio Y, et al (2008) Extracting and composing robust features with denoising autoencoders. In: The 25th International Conference on Machine Learning, ACM, pp 1096–1103
25. Tian Y, Lu C, Zhang X, et al (2020) A pattern mining based evolutionary algorithm for large-scale sparse multi-objective optimization problems. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2020.3041325
26. Tan Z, Wang H, Liu S (2021) Multi-stage dimension reduction for expensive sparse multi-objective optimization problems. Neurocomputing pp 159–174
27. Tian Y, Liu R, Zhang X et al (2020) A multi-population evolutionary algorithm for solving large-scale multi-model multi-objective optimization problems. IEEE Trans Evol Comput. https://doi.org/10.1109/TEVC.2020.3044711
28. Omidvar M, Li X, Yang Z, Yao X (2010) Cooperative co-evolution for large scale optimization through more frequent random grouping. In: IEEE Congress on Evolutionary Computation (CEC), pp 1–8. https://doi.org/10.1109/CEC.2010.5586127
29. Chen W, Weise T, Yang Z, Tang K (2010) Large-scale global optimization using cooperative coevolution with variable interaction learning. In: Schaefer, Robert, Cotta, Carlos, et al (eds.) PPSN. LNCS, 6239:300-309. Springer, Heidelberg
30. Aelst S, Wang X, Zamar R, Zhu R (2006) Linear grouping using orthogonal regression. Comput Stat Data Anal 50(5):1287–1312
31. Omidvar M, Yang M, Mei Y, Yao X (2017) DG2: a faster and more accurate differential grouping for large-scale black-box optimization. IEEE Trans Evol Comput 21(6):929–942
32. Deb K, Agrawal R (1995) Simulated binary crossover for continuous search space. Complex Syst 9(4):115–148
33. Deb K, Goyal M (1996) A combined genetic adaptive search (GeneAS) for engineering design. Comput Sci Inf 26(4):30–45
34. Tian Y, Cheng R, Zhang X, Jin Y (2017) PlatEMO: a MATLAB platform for evolutionary multi-objective optimization. IEEE Comput Intell Mag 12(4):73–87
35. Liang J, Gong M, Li H, Yue C, and Qu B (2018) 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
36. Bosman P, Thierens D (2003) The balance between proximity and diversity in multiobjective evolutionary algorithms. IEEE Trans Evol Comput 7(2):174–188
37. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3(4):257–271
38. Ishibuchi H, Akedo N, Nojima Y (2015) Behavior of multiobjective evolutionary algorithms on many-objective knapsack problems. IEEE Trans Evol Comput 19(2):264–283
39. Xiang Y, Yang X, Zhou Y, Huang H (2020) Enhancing decomposition-based algorithms by estimation of distribution for constrained optimal software product selection. IEEE Trans Evol Comput 24(2):245–259