Applied Soft Computing Journal 89 (2020) 106120

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

A clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems

Ruochen Liu *, Rui Ren, Jin Liu, Jing Liu

Key Lab of Intelligent Perception and Image Understanding of Ministry of Education, International Center of Intelligent Perception and Computation, Xidian University, Xi'an, 710071, China

ARTICLE INFO

Article history:
Received 21 January 2019
Received in revised form 14 January 2020
Accepted 18 January 2020
Available online 23 January 2020

Keywords:
Large-scale multi-objective problems
Cooperative coevolution
Decision variable clustering
Dimensionality reduction

ABSTRACT

When solving multi-objective problems (MOPs) with a large number of variables, analysis of the linkage between decision variables is maybe useful for avoiding "the curse of dimensionality". In this work, a clustering and dimensionality reduction based evolutionary algorithm for large-scale multi-objective problems is suggested, which focuses on clustering decision variables into two categories and then utilizes a dimensionality reduction approach to get a lower dimensional representation for those variables that affect the convergence of the evolution. The interdependence analysis is carried out next aiming to decompose the convergence variables into a number of subcomponents that are easier to be tackled. The algorithm presented in this article is promising on a series of test functions, and the outcome of these experiments reveal that our suggested algorithm is able to prominently enhance the performance; meanwhile it can save computing costs to a large extent compared with some latest evolutionary algorithms (EAs). In addition, the proposed algorithm can be extended to solve MOPs with dimensions up to 5000, with a good performance obtained.

1. Introduction

In the academic sector of evolutionary computation, earlier interest is mainly drawn to single objective optimization. With the appearance of many complicated problems in real world, more and more researchers have devoted themselves to solving this kind of optimization problems, which usually have no less than two conflict objectives and are referred to multi-objective problems (MOPs). It is inspiring that lots of suggested evolutionary algorithms have been successfully used to settle MOPs, such as NSGA-II [1] and MOEA/D [2] and NSGA-III [3].

It has been proved that these multi-objective evolutionary algorithms (MOEAs) have the ability to obtain all-right performance on some test problems, and further have been used to tackle with some practical issues [4–8]. However, in the real world, there still exist many problems that need to be solved which contain a mass of decision variables [9–12]. The effectiveness and efficacy of existing optimization approaches will deteriorate with the increasing of the number of dimensions due to "the curse of dimensionality". With the more and more extensive applications with high dimensionality occurring, some special multi-objective evolutionary algorithms need to be designed.

A popular means to solve this type of problems is to apply a divide-and-conquer tactics, which was first proposed in [13]. It aims to decompose all variables into a set of smaller groups that are easier to be settled. Cooperative coevolution optimizes each sub-problem independently after decomposition is achieved.

In the cooperative coevolution (CC), a pivotal step is how to choose decomposition strategy. As CC optimizes each sub-problem one by one, a satisfying decomposition strategy is supposed to place interacting variables in one group and independent variables in different groups at the same time. From the view of our point, the current grouping approaches fall into two categories: one is fixed grouping strategies [14–16] and the other is dynamic grouping categories [17,18]. For single-objective optimization problems [19–23], dependency relationship between variables are simple, so this kind of problems is relatively easy to solve. But different from [24–27], in multi-objective problems, there are dependencies among variables and among multiple objectives, which makes it tougher to get perfect solutions.

When it comes to solve a multi-objective problem, linkage between decision variables needs to be taken into account. Several works have been done on this topic, among them grouping-based methods are fast-growing. These methods divide all variables into a set of smaller sub-groups in search space to deal with them one by one. Many-objective evolutionary algorithm for large scale variables (LMEA) [28] and decision variable analysis based evolutionary algorithm (MOEA/DVA) [29] are two typical algorithms for it. They both categorize the variables in decision space into diversity related clusters and convergence related groups, which

* Corresponding author.
E-mail address: ruochenliu@xidian.edu.cn (R. Liu).

https://doi.org/10.1016/j.asoc.2020.106120
1568-4946/

control the distribution on the final front of final population and how close they approach to the true Pareto Front (PF).

In addition to the linkage between variables, another point that should not be ignored in multi-objective optimization problems is the mapping relationship between decision variables and objective functions. Wang et al. put forward dimensional reduction based memetic optimization strategy (DRMOS) in [30]. The unbalancing mapping relationship means objective values are influenced by variables differently. DRMOS aims to reduce the dimensionality of search space by recognizing the mapping relationship to evolve the population. In [31], the nonlinear correlation information entropy is used to measure the mapping relationship between decision variables and objective functions because of its effectiveness in describing no matter linear or nonlinear correlation, which improves the computational efficiency of efficient decomposed search strategy.

In this work, a clustering and dimensionality reduction based evolutionary algorithm for multi-objective problems (MOPs) with large-scale variables is suggested. Firstly, we conduct a clustering strategy to separate all variables in decision space into two clusters, named diversity related variables and convergence related variables. Then principal components analysis (PCA) [32] is utilized to get a lower representation for convergence related variables, which will next be decomposed into a set of subgroups by the interdependence analysis procedure. In the end, every subcomponent is optimized cooperatively to evolve the population. The main contributions can be given as follows:

(1) A decision variable clustering strategy can separate all variables into two groups: diversity related variables as well as convergence related variables, which can group variables more accurately and benefits the whole evolution. The approach uses the angles between the convergence direction and the sampled solutions as features and applies k-means approach to group these features. A smaller angle means this variable has more contribution to convergence and a bigger angle means it contributes more to diversity. Thus all variables can be grouped either convergence variables or diversity variables.
(2) PCA is used to get a lower representation for the original convergence variables. It represents the most crucial information of the data set in a new orthogonal coordinate system. PCA generates principal components (PCs) by linearly combining original variables. In this way, PCs have no correlations to each other. They also maintain the maximum variation of the original data. The majority information of original data set is maintained under such representation. This contributes to smaller computation costs and more satisfactory results.
(3) We have conducted empirical evaluations on several test suites to make comparisons with some MOEAs that have the potential to solve large-scale MOPs to assess the property of our suggested evolutionary algorithm. The final experimental outcome shows that our algorithm can solve large-scale optimization problems that have as many as 5000 decision variables, and satisfactory results using acceptable computational cost are obtained.

In the rest of the article, firstly we retrospect correlated concepts in Section 2. Some details of the primary procedure of the suggested algorithm are introduced in Section 3. Section 4 elaborate experiments as well as results to empirically evaluate the performance. Section 5 comes to conclusions and points out our future work.

2. Background

Several elementary definitions in the academic sector of multi-objective optimization are firstly presented in the following part. Then, we concisely recall the existing MOEAs for MOPs. After that, we recall some related works for large-scale optimization. Finally, knowledge about PCA is elaborated.

2.1. Multi-objective optimization

In general, we can formulate an MOP as

$$\begin{aligned} \min \quad & \mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \\ & \textit{subject to} \quad \mathbf{x} \in \Omega \end{aligned} \tag{1}$$

where $\mathbf{F}(\mathbf{x})$ is a m dimensional objective function, it includes m real-valued continuous objectives $(f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x}))$, $\mathbf{x} \in \mathbf{R}^D$ is a vector that contains D dimensional variables in search space.

In fact, there are numerous optimal solutions that cannot dominate each other in multi-objective optimization. In general, we can define the dominance relationship between two solutions for a minimum MOP as follows:

$$\forall i \in \{1, \ldots, M\} : F_i(\mathbf{x}) \leq F_i(\mathbf{y}) \wedge \exists j \in \{1, \ldots, M\} : F_j(\mathbf{x}) < F_j(\mathbf{y}) \tag{2}$$

If the abovementioned inequation is satisfied, we can state $\mathbf{x}$ dominates $\mathbf{y}$, which can be given in the form of $\mathbf{x} \prec \mathbf{y}$. Pareto optimal refers to those solutions that are not dominated by any another solution.

Generally speaking, it is tough to get all Pareto optimal solutions. Thus, we have three aims to achieve: (1) enhance convergent ability of the algorithm to make that all final solutions come near to the true Pareto front (PF); (2) improve diversity of the algorithm to make that the objective vectors spread diversely on the PF on condition that the convergence speed is satisfactory; (3) use computing resource as little as possible on condition that convergence and diversity are both guaranteed. During the past couple of years, a lot of attention has been attracted to the area of multi-objective optimization and many researchers have devoted themselves to this field. Therefore, a host of outstanding MOEAs [33,34] have been proposed. Generally, we can classify MOEAs into three categories: (1) dominance-based MOEAs [1, 3]; (2) decomposition-based MOEAs [2,35]; (3) indicator-based MOEAs [36–38].

In the first category, namely dominance-based MOEAs, the most prevalent one is NSGA-II [1]. It determines the dominance relationship among all solutions, then it applies a non-dominated sorting strategy to sort all solutions. To make sure the diversity of the population, an elitist preserving approach and a crowding distance mechanism are put into use. During the evolutionary process, once N offspring are generated using genetic algorithm operators, where N is the size of population, the offspring are combined with the parents, then the aforementioned three mechanisms are applied to choose N individuals from the combined population to carry through next evolution. Based on the fundamental algorithm, many variants were raised, such as [3, 35].

The most famous one in the field of the decomposition-based MOEAs is MOEA/D [2]. Its core idea is the use of decomposition strategy, by which it can transform an MOP into a series of single-objective optimization problems. It achieves this goal by attributing each objective a weight. There are three significant decomposition methods, for example, we can transform an MOP into N single-objective problems using the Tchebycheff method. After this step, N individuals are produced and co-evolved. Following this, a crowd of researchers have proposed many other new versions of this type of MOEAs, including [33,36].

As for indicator-based MOEAs, they generally adopt a performance indicator to guide evolution. Hypervolume (HV) [33] is the most commonly used indicator because it can measure convergence performance and diversity performance simultaneously. [36] is an instance of indicator-based MOEA.

Even though these above mentioned MOEAs are promising to solve MOPs already, they are all efficient and valid only in relatively low-dimensional space. Once they are applied to solve multi-objective problems with high dimensional decision variables, their performance would sharply decrease. The primary reason is the exponentially increases in search space when the dimension grows. On the other hand, it is easy for solutions to trap into local areas because lots of local PFs exist in high-dimensional space. Hence, it is necessary to design particular MOEAs to deal with large-scale MOPs with effect.

2.2. Large-scale multi-objective optimization

We have noticed that there are numerous MOPs that contain thousands of decision variables in our real-world applications, but the existed multi-objective evolutionary algorithms cannot solve these problems perfectly. As a matter of fact, extensibility in variable space is an issue which has not been researched deeply in the area of MOEAs. Experience shows that as the quantity of variables in search space increases, the effectiveness of most MOEAs decreases dramatically [39,40].

In nature, coevolution occurs between interacting species or groups. In cooperative algorithms, individuals who cooperate well with others will be rewarded, while those who do not perform well together will be punished. Antonio and Coello firstly utilized the framework of cooperative coevolution (CC) in evolutionary algorithms aiming to solve large-scale MOPs in [13]. It applies a divide-and-conquer method to randomly decompose a lot of variables in search space into several smaller subcomponents with the same size. Then the subcomponents evolve collaboratively using a separate evolutionary algorithm. Following this achievement, many other cooperative coevolutionary methods have been proposed. Majority of them aim at solving large scale global optimization problems [17].

While for MOPs, we must consider multiple objectives at the same time, making it much more complicated when using the abovementioned CC framework. Lately, an MOEA has been put forward by Ma et al. for settling large-scale multi-objective problems, known as MOEA/DVA [29]. It firstly uses an approach called decision variable analysis to separate all decision variables into three categories, which can be achieved by perturbing the values of certain variables and then checking the dominance relationship between solutions. Base on the dominance relationship, a decision variable can be classified into one of three groups. More specifically, if the generated solutions do not dominate and are not dominated by any other solutions after variable perturbation, this variable is called distance variable. If the generated solutions have dominance relationship with each other after perturbation, this variable is called position variable. Otherwise, the variable is called mixed variable. After the decision variable grouping step is achieved, MOEA/DVA conducts an interdependence analysis to decompose high dimensional variables to several independent subcomponents with lower number of variables and then optimizes each subcomponent separately. Once the solutions are optimized to reach the Pareto front, all decision variables are optimized together to make sure the final solutions are distributed uniformly on the front. One problem in MOEA/DVA is that it treats all mixed variables as diversity related variables simply. However, in fact, these variables still need to be further divided into either diversity related groups or convergence related groups because their contribution to convergence and diversity varies. Another novel evolutionary algorithm adopting decision variable clustering approach is presented in [28]. It is specially tailored for dealing with many objective optimization problems with high dimensional decision space. It firstly clusters all decision variables into two classes, where one class influences convergence performance and remain variables is correlated with diversity. Then during optimization process, two strategies are adopted separately for the two classes.

Both MOEA/DVA [29] and LMEA [28] apply grouping-based strategy to partition decision variables into some different categories and then deal with them separately. The grouping strategy they use is an idea called fixed grouping approach. There are dynamic grouping strategies as well. In [41], firstly, a decomposition pool that contains different group sizes is designed. Then, one of the group sizes is chosen probabilistically from this pool. Based on the selected grouping size, the whole dimension is divided into many groups with different sizes. This method can save many function evaluations.

2.3. Principal component analysis (PCA)

PCA was put forward by Karl Pearson [42] and has become one important statistical method from then on. It has been applied in many aspects including dimensionality reduction, feature elimination, multivariate data analysis, image recognition, data visualization and machine learning tasks. In PCA, data is transformed to a new coordinate system determined by data itself. In the transformation of coordinate system, the direction of maximum variance is determined as the direction of orthogonal coordinate axis, on account of the maximum variance of data comprising the most crucial information of data. The direction which contains the maximum variance of original data is taken as the new coordinate axis. The next new coordinate axis chooses the direction which has the second largest variance and at the same time is orthogonal to the first coordinate axis. Repeating the process for the number of the feature dimension of the original data, we notice that the first several coordinate axes contain most of the variances, and the latter ones contain almost zero variances. Based on this knowledge, the remain coordinate axes can be ignored, and the preceding several coordinate axes with absolutely no partial variance are just retained so as to realize the dimension reduction of data features.

PCA was firstly put into use to reduce high dimensional space in [43]. In higher dimensional analysis cases, the more explanatory variables are, the greater the possibility of over-fitting. Therefore, principal component analysis can mitigate the influence of over-fitting, particularly when there exists strong relationship between variables.

3. The proposed algorithm

In this section, firstly we design a novel clustering and dimensionality reduction based evolutionary algorithm for MOPs with numerous decision variables, referred to as PCA-MOEA. As aforementioned, we can first adopt a clustering approach to classify all decision variables into two categories, one correlated with convergence and the other correlated with diversity. As a result, we can handle with each group specifically. Firstly, the diversity variables are initialized by the uniformly sampling technique [44] and convergence variables are initialized randomly. After that, we use PCA to obtain a lower representation for the convergence variables because the number of them is still large. In this step, we can control the percent of variance, which decides the number of decision variables after processing. Then interdependence analysis is carried out to detect the interdependence relationship among convergence variables that are in low dimension space.

Variables in the same subgroup interact with each other while there are no interactions between two different subgroups. By this step, variables with small number can be optimized easily. In the last, the cooperative coevolution frame is employed to evolve the population. We present the primary steps of PCA-MOEA in Algorithm 1.

Several crucial components of PCA-MOEA are further illustrated in the coming parts.

Algorithm 1 PCA-MOEA

Input:
the size of population N, FE_{max} (the maximal number of function evaluation), dimension n
Output:
final population **pop**, corresponding objective vectors **Val**

Set $FE = 0$. Adopt Algorithm 2 to cluster decision variables to generate [*Diver*, *Conver*];
pop(:, *Diver*) ← Employ uniformly sampling technique [44] to initialize all diverse variables;
pop(:, *Conver*) ← Initialize the convergence variables (CVs) randomly;
Evaluate the individuals,**Val** = **F**(*pop*) , **Old Val** = **Val**, $FE = FE + N$;
[**pop,Val**] ← Apply PCA and Algorithm 3 to get the low-dimensional representation low*CV* of the large-scale Conver to form new pop;
[*Subcomponents*, **pop, Val**] ← divide the low*CV* and evolve population [**pop, Val**];
Threshold ← 0.01, Bound← 1;
while *threshold* ≤ *BoundandFE*$_{max}$ > *FE* **do**
  for $j = 1 : size(Subcomponents)$ **do**
    [**pop,Val**] ← *SubcomponentOptimizer* (**pop, Val**, *Subcomponent*[*j*])
  end for
  utility ← *CalculateUtility*(***Val,Old Val***);
end while

3.1. Variable clustering

We illustrate decision variable clustering approach in Algorithm 2. Suppose given a two-objective minimization problem, each individual has five variables, x_1, x_2, x_3, x_4 and x_5. Firstly, we choose two candidate solutions from the whole population randomly, and then perform ten perturbation operations on each variable of the selected solutions. After that, we normalize the perturbed generated solutions. In order to adapt to the normalized solution of each group, we generate line L. We also require the normal line of hyperplane $f_1 + \cdots + f_M = 1$, which represents the convergence direction. Following this we can get the angle between convergence direction and the normal line of hyperplane each fitted line L. In this case, we choose two candidate solutions for clustering and each variable is bound up with two angles, these variables with larger angles make more contribution to diversity, while variables with smaller angles make more contribution to convergence. We need to note that the more candidate solutions we select for clustering decision variables, the more angles bound up with each variable are generated. As a result, we can get more accurate measurements.

In the end, we adopt *k-means* clustering approach to separate all variables in search space into two groups considering the properties of each variable. Variables with smaller angle averages are grouped in convergent correlation clustering, while the others are grouped in the diversity-related cluster.

We should notice that the technique we adopt here can gain more precise results. Specifically speaking, these variables that are identified as mixed variables in MOEA/DVA no longer

Algorithm 2 Decision Variable Clustering

Input: *pop*, number of selected candidate solutions*SelNum*, number of perturbation*PerNum*
Output: *[Diver, Conver]*

for $i = 1 : n$ **do**
  C ← Choose SelNum solutions from pop randomly;
  for $j = 1 : SelNum$ **do**
    Make perturbation for the *i*-th variable of C[*j*] for *PerNum* times to bring a generation SP and then normalize it;
    Generate a fitting line L for SP in objective space;
    Angle[*i*][*j*] ← the angle between generated L and normal line of hyperplane;
    MSE[*i*][*j*] ← the mean square error of the fitting;
  end for
end for
CV ← $\{mean(MSE[i]) < 1e-2 | i = 1, \ldots, D\}$;
[*C*1, *C*2] ← apply *k-means* to group all variables into two categories adopting *Angle* as feature;
if $CV \bigcap C1 \neq \emptyset$ and $CV \bigcap C2 \neq \emptyset$ **then**
  $CV \leftarrow CV \bigcap C$, *C* is either *C1* or *C2* depending on which of the average of *Angle* is smaller;
end if
DV ← $\{j \notin CV | j = 1, \ldots, D\}$

exist, and they are classified either diversity-related variables or convergence-related variables. Besides, some diversity-related variables may also be considered as convergence-related variables since they make more contribution to convergence.

Another one point that we should pay attention to is, as the case stands, certain decision variable may change its class in different regions. The more solutions we select to be interfered with, the more areas we will probably sample, however, the global consistency of the final solutions still cannot be guaranteed.

3.2. The detailed introduction of PCA

Suppose we get a data set $x = \{x^{(1)}, x^{(2)}, \ldots, x^{(m)}\}$, the first step we need to adopt is preprocessing, that is to say mean normalization. We primarily calculate the average value of each feature and afterwards replace each of $x^{(i)}$ with $x^{(i)} - \mu$, aiming to make each feature have precise zero mean value. We can use the following formula to obtain a $N \times 1$ column vector μ

$$\mu = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{3}$$

To guarantee that different features are processed on the same scale, the next step is to re-scale coordinate because different attributes have different scales, so that each coordinate has a unit variance

$$\sigma_j^2 = \frac{1}{m} \sum (x_j^i)^2 \tag{4}$$

After that we replace each $x^{(i)}$ with x^i/σ_j. Once the preprocessing step is achieved, we calculate the covariance matrix that is symbolized by Σ, defined as follows:

$$\Sigma = \frac{1}{m} \sum_{i=1}^{m} (x^i)(x^i)^T \tag{5}$$

The covariance matrix is symmetric whose size is $N \times N$. In order to obtain it, we need to calculate the eigenvectors of the matrix Σ. We can use singular value decomposition [45] to achieve this.

We can gain eigenvalues as well as eigenvalue matrices by SVD, where 'U' is an *N* square matrix with a vector column

$\{u^{(1)}, \ldots, u^{(m)}\}$. 'S' is a diagonal matrix, which only has the vector element $\{s^{(1)}, \ldots, s^{(m)}\}$ in the diagonal line whose dimension also is N.

We can construct a new matrix $U_{reduce} \in \mathbf{R}^{n\times k}$ by selecting the first k column of U to reduce dimension. We demand to build the relationship between W and U_{reduce} to get a vector called 'W', denoted by $W = (U_{reduce})^T \times X$ where $W \in \mathbf{R}^{n\times 1}$.

After the eigenvectors of the covariance matrix are obtained, we rank them in descending order of eigenvalues. Small eigenvalues mean that their components are less important, so we can ignore them without losing important information. Finally, we get a feature matrix, which is formed by using the eigenvalues we select from the list of eigenvectors in the columns.

We can formulate the average squared error ASE by Eq. (6), where $x^{(i)}$ is the original data, x^i_{proj} is projection data which is in the low-dimensional subspace.

$$ASE = \frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x^i_{proj}\|^2 \tag{6}$$

The total variation is shown as follows:

$$\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2 \tag{7}$$

A representative method to determine k is to select the minimum value that can maintain 99% variance:

$$\frac{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)} - x^i_{proj}\|^2}{\frac{1}{m}\sum_{i=1}^{m} \|x^{(i)}\|^2} \leq 0.01 \tag{8}$$

3.3. Interdependence analysis

In this section, we will detect the interaction between the convergence-related variables, because the amount of them is relatively large and they influence the convergence speed to the true PF. According to these relationships, these variables would be further decomposed into a set of sub-groups. In this case, variables interact with each other are in the same subgroup, and there is no interaction between each of the two subgroups. Those variables in the same subgroup are also called interacting variables, and each of them cannot be optimized independently since there exists interactions among them. Algorithm 3 indicates the specifics of the interaction detection for variables correlated with convergence, and the skill we apply is proposed in [29]. Many single-objective large-scale optimization [19,25] have also adopted similar approaches for variable interaction analysis.

We define the interactions between two variables as: suppose we have an MOP $f = min(f_1, f_2, \ldots, f_m)$, if there exist x, a_1, a_2, b_1, b_1, and among $f_1, f_2, \ldots, f_m$, at least one f_k, $1 \leq k \leq m$, can make $f_k(x)|_{x_i=a_2, x_j=b_1} < f_k(x)|_{(x_i=a_1, x_j=b_1)}$ and $f_k(x)|_{x_i=a_2, x_j=b_2} > f_k(x)|_{x_i=a_1, x_j=b_2}$ be met simultaneously. Under this circumstance, variables x_i and x_j are considered to be interacted with each other, where $f_k(x)|_{x_i=a_2, x_j=b_1} = f_k(x_1, \ldots, x_{i-1}, a_2, \ldots, x_{j-1}, b_1, \ldots, x_n)$. In Algorithm 3, we firstly initialize an empty subset of interaction variable groups and then assign convergence-related variables to disparate subgroups according to the pairwise interactions between variables. More specifically, two variables will be assigned into the same subgroup on condition that a variable interacts with at least one current variable in subCV; otherwise, this variable will be assigned to a new subgroup. We can infer that there are two extreme cases: in one situation, there is only one subgroup, which signifies that the convergence variables are completely inseparable, in another case, if the decision variables are completely separable, the number of subgroups would be $|\ CV\ |$.

Algorithm 3 Interdependence Analysis

Input: pop, Conver, number of chosen solutions CorNum
Output: subCVs

set *subCVs* as ∅;
for all the $v \in CV$ **do**
  set *CorSet* as ∅;
  for all the *Group* ∈ *subCVs* **do**
    for all the $u \in Group$ **do**
      flag ← **False**;
      for i = 1 : *CorNum* **do**
        Choose an individual p from pop randomly;
        if v has interaction with u in individual p **then**
          flag ← **True**;
          CorSet = {*Group*} *cupCorSet*;
          Break;
        end if
      end for
      if *flag* is **True then**
        Break;
      end if
    end for
  end for
  if *CorSet* = ∅ **then**
    subCVs = *subCVs* ∪ {{v}};
  else
    subCVs = *subCVs*/*CorSet*;
    Group ← all variables in *CorSet* and v;
    subCVs = *subCVs* ∪ {*Group*};
  end if
end for

3.4. Subcomponent optimization

Just as proposed in MOEA/DVA [29], Algorithm 4 introduces subcomponent optimization in detail. When optimize one sub-MOP, the information of its neighborhood is meaningful. Here we use Euclidean distances between diversity variables to define neighborhood relations. If the diversity variables of the ith sub-MOP is close to the that of the jth sub-MOP, then we say ith sub-MOP and jth sub-MOP are neighbors. Since the values of diversity-related variables are constant for each sub-MOP earlier, now only the offspring population of the ith sub-MOP is used to renew the current solution. Differential evolution [46] is used to generate new solutions in this step.

Algorithm 4 Subcomponent Optimization

Input: *pop*, *Obj*, indexes of convergence variables in the identical subcomponent *indexes*.
Output: *pop*, corresponding objective vectors *Val*

for $i = 1 : N$ **do**
  The p-th and q-th individual are randomly chosen from the adjacent individuals. Adopt differential evolution (DE) strategy [46] to form new variables $y' \leftarrow$ pop(i, *indexes*) + F ∗ (*pop*(*p*,indexes)-pop(*q*, indexes)) later execute mutational operation on y';
  The value is replaced by the randomly generated value within the boundary when a component of y' is out of boundary. Then make evaluations for the new population, set *FE*=*FE*+1;
  if $\sum_{j=1}^{m} f_j(y) < \sum_{j=1}^{m} Val(i, j)$ **then**
    let *pop*(*i*, :) =y and *Val*(*i*, :) =F(y);
  end if
end for

Table 1
Average and standard deviation of IGD obtained by the compared algorithms on problems with 30 variables.

Problems	PCA-MOEA	MOEA/DVA		LMEA		NSGA-III		MOEA/D	
UF1	3.6156E−03 (8.3932E−05)	4.1434E−03 (8.2635E−05)	≈	5.9665E−03 (3.4239E−03)	−	7.5532E−03 (5.3728E−03)	−	5.9238E−03 (2.3132E−01)	−
UF2	3.9060E−03 (5.3910E−5)	4.2738E−03 (8.3613E−4)	≈	7.3550E−03 (2.4186E−04)	−	8.2805E−03 (4.3291E−03)	−	6.2931E−03 (7.3813E−04)	−
UF3	1.1219E−01 (9.2592E−02)	2.2714E−02 (7.2599E−03)	+	7.8110E−02 (7.5420E−03)	+	4.3821E−02 (2.3748E−02)	+	**8.2734E−03** (6.2831E−03)	+
UF4	**1.3683E−02** (4.0392E−04)	3.5067E−02 (1.0070E−03)	−	3.2457E−02 (3.2281E−03)	−	4.3279E−02 (6.4132E−03)	−	4.5277E−02 (5.3288E−03)	−
UF5	**2.5660E−02** (3.2931E−02)	3.2592E−02 (4.6786E−03)	≈	1.2013E−01 (6.4271E−03)	−	4.2841E−02 (8.3749E−02)	−	4.2885E−02 (4.2173E−02)	−
UF6	6.2344E−01 (3.4904E−02)	**5.6134E−02** (1.3729E−02)	+	1.2823E−01 (4.3980E−03)	+	3.2118E−01) (4.8871E−02)		3.2813E−01 (3.2382E−01)	≈
UF7	5.5550E−02 (4.0288E−04)	**3.7667E−03** (4.6437E−05)	+	6.0919E−02 (8.7121E−04)	≈	6.7283E−03 (4.2142E−03)	+	5.2318E−03 (5.3821E−03)	+
UF8	**4.7767E−02** (3.2915E−03)	5.7788E−02 (1.1960E−02)	−	9.9038E−02 (7.5730E−03)	−	7.3877E−02 (3.2375E−02)	−	6.4738E−02 (3.2732E−02)	−
UF9	4.4875E−02 (3.0980E−03)	1.2333E−01) (1.6254E−01)		1.3572E−01 (3.4729E−02)	−	**5.3729E−02** (2.3841E−02)	≈	3.2831E−01 (4.7694E−02)	−
UF10	**1.0134E−01** (2.1218E−02)	1.0352E−01 (3.3009E−03)	≈	4.6055E−01 (6.4729E−02)	−	4.3728E−01 (7.2713E−01)	−	3.8716E−01 (3.2371E−02)	−
WFG1	9.2360E−01 (3.2132E−02)	2.1730E+00 (1.4480E−02)	−	1.5998E+00 (3.2194E−03)	−	1.323E+00 (6.4182E−03)	−	**3.2166E−01** (6.2713E−02)	≈
WFG2	3.3146E−01 (9.2194E−02)	2.2200E−01 (3.4970E−02)	≈	2.7077E−01 (5.2188E−02)	≈	3.2931E−01 (3.8321E−02)	≈	**4.2318E−02** (1.2131E−02)	−
WFG3	**4.4629E−02** (4.1213E−02)	7.7500E−02 (1.2320E−02)	≈	5.9578E−02 (6.3823E−02)	−	3.2774E−01 (2.3238E−02)	−	5.2886E−01 (2.3813E−02)	−
WFG4	4.5311E−01 (1.2102E−02)	2.2543E−01 (1.6949E−03)	+	**2.1377E−01** (6.3831E−03)	+	3.8692E−01 (2.3828E−02)	≈	3.8125E−01 (3.2842E−02)	≈
WFG5	3.9820E−01 (8.2913E−03)	**2.1210E−01** (5.5937E−03)	+	2.3260E−01 (3.2881E−03)	+	3.0281E−01 (1.8792E−02)	≈	2.9789E−01 (3.2813E−02)	≈
WFG6	**2.0512E−01** (3.2019E−03)	2.1900E−01 (1.9370E−03)	≈	2.1807E−01 (8.5283E−03)	≈	3.2981E−01 (3.2933E−03)	−	2.7688E−01 (2.3891E−02)	−
WFG7	5.0831E−01 (8.2812E−03)	**2.1500E−01** (1.1140E−03)	+	2.2136E−01 (8.4731E−03)	+	3.0218E−01 (5.8079E−03)	+	3.2831E−01 (4.2813E−03)	+
WFG8	3.1863E−01 (9.3201E−03)	2.9030E−01 (7.6020E−03)	≈	**2.5522E−01** (4.2383E−03)	+	3.2813E−01 (5.2739E−02)	≈	3.0084E−01 (4.3823E−02)	≈
WFG9	**2.2627E−01** (8.2938E−03)	2.4660E−01 (1.8960E−02)	≈	2.3309E−01 (2.3841E−03)	≈	2.9743E−01 (1.0836E−02)	−	2.7649E−02 (7.3723E−03)	−
+/−/≈		6/4/9		6/9/4		3/10/6		4/10/5	

4. Experiments and results

For the sake of demonstrating the performance of PCA-MOEA, several widely used suits of benchmark problems are experimentally tested in this part, i.e. the DTLZ test suites [47], the ZDT test suites [48], the complicated UF1–UF10 problems [49] and WFG test suites [50]. All the simulations were run on a personal computer with Windows 7 systems configured as Intel (R) Core (TM) i3 CPU M 350 and 4G RAM.

4.1. Performance measurements

Inverted generational distance (IGD) [50] is adopted in the following experimental studies, as it is an index that can assess convergence and uniformity performance simultaneously. Let $\mathbf{P}^*$ be plenty of evenly distributed solutions on the true PF, and $\mathbf{P}$ is the approximate Pareto front. $d(\nu, \mathbf{P})$ expresses the minimum Euclidean distance form ν in $\mathbf{P}^*$ to solution $\mathbf{P}$. Then we can define the IGD value from $\mathbf{P}^*$ to $\mathbf{P}$ as

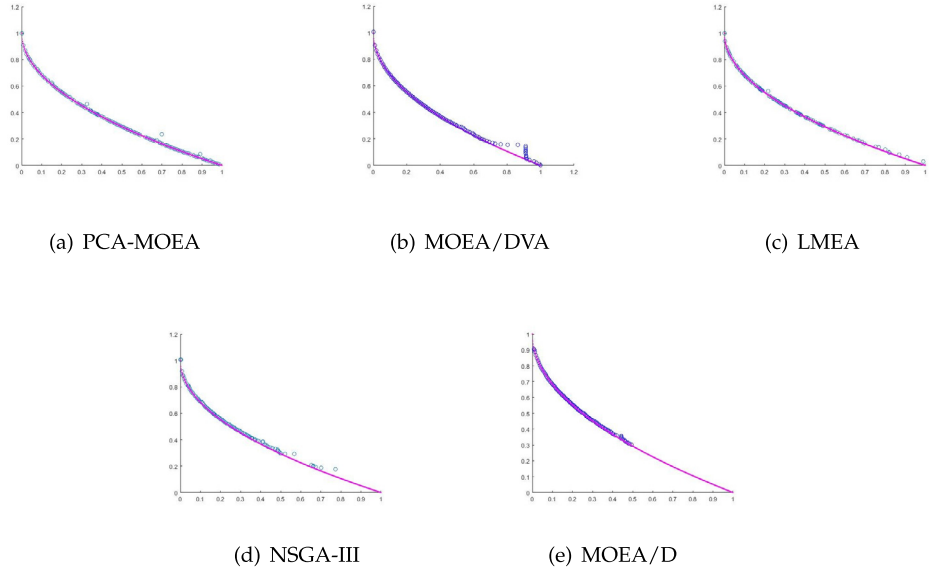
$$IGD(\mathbf{P}^*, \mathbf{P}) = \frac{\sum_{\nu \in \mathbf{P}^*} d(\nu, \mathbf{P})}{|\mathbf{P}^*|}$$

For two-objective test problems, we set the number of solutions in $\mathbf{P}^*$ as 500 and for three-objective instances that number is set as 2500 in this work. Each experiment runs 30 times independently. And the best results are highlighted on the basis of the average IGD.

4.2. Experimental result and analysis

4.2.1. Results of problems with 30 variables

In this part, we apply all algorithms on test suits with low-dimensional decision variables, i.e., UF1–UF9 with 30 variables and WFG1–WFG9 test problems having 24 variables. The reason why we use these test problems is that we want to make complete comparisons with MOEA/DVA [29]. The details of the mathematical descriptions as well as their ideal PFs can be referred in [46] and [51]. For 2-objective functions, the size of population is set as 100 while for 3-objective test instances, it is set as 150, which is the same as in [29]. For fair comparisons, some general parameters used in compared algorithms are stated here. The distribution indexes of SBX as well as polynomial mutations were set at 15. Besides, we set the mutation probability as 1/n, where n refers to the quantity of variables in decision space. Crossover rate (CR) and scaling factor (F) of differential evolution are set as 1 and 0.5, respectively, as recommended in [52]. In MOEA/D, we set the neighbor subproblems (T) size as $0.1N$ as



(a) PCA-MOEA (b) MOEA/DVA (c) LMEA

(d) NSGA-III (e) MOEA/D

Fig. 1. The PF obtained by five algorithms on UF2 function with 30 decision variables.

usual, where N is the size of population, and another parameter–parent selection probability, is set as 0.9. These parameters are fixed in the following parts of experiments. As in [29], the maximal number of function evaluations is set as 300 000 for UF test suites and 100 000 for WFG test suites respectively. In this section, to make a fair comparison, we set the same number of function evaluations as in [29]. But, it needs to be pointed out that number only needs to be set as 100 000 for UF test suites and 10 000 for WFG problems in the proposed algorithm, because in this case of setting we already have satisfactory results, which will be proven in later section. The code of LMEA can be found in http://www.soft-computing.de/jin-pub_year.html. The code of MOEA/D and NSGA-III can be found in PlatEMO [53]. And MOEA/DVA is implemented by ourselves in Matlab according to [29].

Table 1 presents the statistical results of the five compared algorithms in terms of IGD over 30 runs. In the table, the best results of the five algorithms are highlighted. IGD of the five compared algorithms are also analyzed using the Wilcoxon signed-rank test [41]. "+", "−" and "≈" in Table 1 denote the performance of the compared algorithms is better than, worse than, or similar to that of the proposed algorithm, respectively.

From the statistical results, we can see that PCA-MOEA has better performance on UF1–UF2, UF4–UF5, UF8, UF10, WFG3, WFG6 as well as WFG9 problems, while MOEA/DVA performs best on UF6–UF7, WFG5 and WFG7, LMEA get the best results on WFG4 and WFG8 problems. As for UF3, WFG1 and WFG2, MOEA/D has more satisfactory results over other algorithms. The proposed PCA-MOEA significantly outperforms the LMEA, NSGA-III and MOEA/D according to the Wilcoxon signed-rank test and performs similarly to MOEA/DVA. The success of PCA-MOEA and MOEA/DVA come from two facets. On one hand, they all decompose the convergence related variables into a series of smaller sub-components, which can reduce the optimization difficulty. On the other hand, in the last stage of the whole evolution process, all variables are optimized together to get a uniformly distributed population. We can reach the conclusion that if the MOP is decomposable, we can use the structure of the problem to simplify the optimization.

For most UF test problems, all variables related to convergence are independent for each objective function while for most WFG test suites; the mapping from PS to PF has a great deviation. We kept eighty percent of variance so the second parameter of PCA is set as 80%. After dimensionality reduction, the number of convergence-related variables turns to 18 and 10 separately, while the original number is 29 and 20. This difference is not very significant, as such problems are still small-scale MOPs, however, by doing this, and we indeed reduce computational costs because less number of function evaluations is needed.

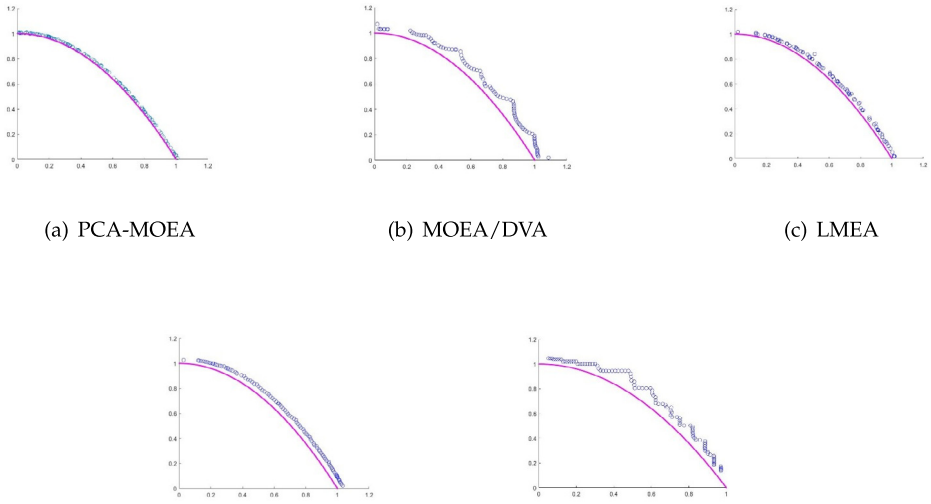
Figs. 1 and 2 shows the obtained PF by five algorithms on UF2 and UF4 test functions. We can find clearly that the final populations gotten by MOEA/DVA, MOEA/D and NSGA-III do not spread over the whole true Pareto front on UF2 problem, even though their convergence ability is satisfactory. LMEA has better convergence, but its diversity is less desirable than PCA-MOEA. For UF4 problem, LMEA and NSGA-III have a not too bad diversity performance, but neither of them converges to the true PF, in addition, MOEA/DVA and MOEA/D do not obtain good results. However, our PCA-MOEA not only has great convergence ability, its diversity performance is also excellent for the two problems.

4.2.2. Results of test problem with 200 variables

MOEA/DVA dealt with MOPs with 200 variables to demonstrate its effectiveness. In this section of experiments, we conduct all algorithms on test functions with 200 decision variables. To make fair comparisons, we apply test function ZDT4, DTLZ1, DTLZ3, UF1–UF6 and UF10, as in [29]. The population size is 100 for ZDT4 and UF1–UF7 problems, while for DTLZ1, DTLZ3 and UF8–UF10 problems, it is set as 150. Table 2 shows the comparison results. For ZDT4, DTLZ and UF1–UF2 test functions, the maximum number of function evaluations is set as 1 200 000 and for UF3–UF6 and UF10 problems, it is set as 3 000 000 for MOEA/DVA, LMEA and MOEA/D. When applying MOEA/DVA, a smaller number is needed in different cases.

The statistical results of IGD over 30 runs of the compared algorithms are described in Table 2. In the table, the results of the best performing algorithm, i.e., the smallest mean IGD values are highlighted. The Wilcoxon signed-rank test [41] is conducted to compare the significance of difference between PCA/MOEA and the compared algorithms. "+", "−" and "≈" indicate that the performance of PCA/MOEA is better than, worse than, or similar to that of the algorithm under comparison, respectively.

We can observe that on most problems, the newly presented algorithm is superior to MOEA/DVA, LMEA and MOEA/D. Not only does it have a smaller IGD value, which means the better convergence and diversity ability, but also the number of function



(a) PCA-MOEA (b) MOEA/DVA (c) LMEA

(d) NSGA-III (e) MOEA/D

Fig. 2. The PF obtained by five algorithms on UF4 function with 30 decision variables.

Table 2
Average and standard deviation of IGD obtained by the compared algorithms on problems with 200 variables.

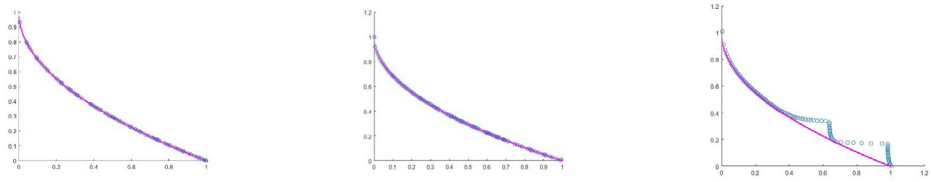
Problems	PCA-MOEA (90%)	PCA-MOEA (10%)	MOEA/DVA		LMEA		MOEA/D	
ZDT4	**3.8682E−03** (3.9829E−05)	3.9283E−03 (4.3872E−04)	3.9231E−03 (3.5337E−05)	≈	2.1664E−02 (6.4812E−03)	−	9.4592E−01 (2.7429E−02)	−
DTLZ1	4.1248E−02 (5.3819E−04)	**1.6330E−02** (3.2938E−04)	2.2652E−02 (1.1502E−04)	−	8.1169E−02 (4.1739E−03)	−	3.5817E+02 (6.3812E+01)	−
DTLZ3	**3.1999E−02** (3.2911E−04)	4.2469E−02 (8.2839E−04)	5.8425E−02 (1.7293E−04)	−	2.8812E−01 (4.3719E−03)	−	5.3798E+02 (3.2189E+02)	−
UF1	3.9391E−03 (5.3297E−04)	**2.7804E−03** (2.1937E−04)	4.0108E−03 (8.1582E−05)	−	8.8718E−03 (2.3184E−02)	−	8.3810E−02 (5.2739E−02)	−
UF2	3.6308E−03 (8.3239E−05)	**2.9190E−03** (8.3726E−05)	4.0657E−03 (2.1424E−05)	−	1.3663E−02 (4.3812E−03)	−	3.4829E−02 (6.4822E−02)	−
UF3	5.2287E−02 (6.4132E−05)	3.6361E−02 (9.4273E−05)	**3.9059E−03** (5.0902E−05)	+	2.7957E−02 (4.2831E−04)	≈	2.3193E−02 (2.3139E−02)	≈
UF4	1.1352E−02 (8.4223E−05)	**1.0076E−02** (2.1837E−05)	3.2392E−02 (2.8345E−04)	−	3.1062E−02 (6.3719E−02)	−	1.2931E−01 (2.3139E−02)	−
UF5	3.1679E−02 (4.2813E−03)	**3.0466E−02** (6.8372E−03)	3.2378E−02 (5.2747E−03)	≈	9.6097E−02 (8.2713E−03)	−	1.3822E−01 (3.2819E−01)	−
UF6	**1.4097E−02** (8.2831E−04)	3.6369E−02 (4.1832E−03)	1.8064E−02 (2.5301E−03)	−	3.5603E−02 (3.2881E−03)	−	4.3829E−02 (5.3723E−02)	−
UF10	**6.8384E−02** (2.3928E−02)	1.0426E−01 (9.4737E−01)	2.3715E−01 (4.8283E−01)	−	4.1564E−01 (6.3811E−02)	−	1.3562E+00 (2.3913E−01)	−
+/−/≈			1/7/2		0/9/1		0/9/1	

evaluations is smaller. We just need 400 000 function evaluations for the first five test instances and 1 000 000 for the last five instances to get ideal results when the parameter in PCA is set as 95%, the number in parentheses under the title of PCA-MOEA. When it comes to 10%, the needed number of function evaluations is much smaller. In the procedure, it can be seen that the number of convergence-related variables becomes 77 and 5 separately from the original 200 for 2-objective UF problems. The maximal number of function evaluations for the first five problems is set as 1 200 000 and for the other, it is set as 3 000 000 when all the other algorithms work.

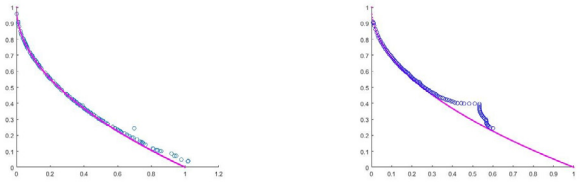
It should be noted that the number of convergence-related variables after dimensionality reduction has correlation with the number of the population size, which is no more than the initial population size.

Figs. 3 and 4 present the final population obtained by compared algorithms on UF2 and UF4 test functions, respectively. Clearly, the PF get by PCA-MOEA is better than the others. Both the convergence ability and diversity ability are more excellent. And the result obtained by PCA-MOEA when the parameter is set as 10% is slightly better than the other cases where the parameter is set as 95% in Fig. 3. In Fig. 4, the difference between the first two figures is smaller, which means that the parameter setting has a little influence in this case.

Fig. 5 shows the evolution curves of IGD values generated by the compared algorithms on four functions with 200 variables. From these figures, we can observe that PCA-MOEA converges faster and can converge at lower IGD values than the other algorithms. The reason is that the dimensionality reduction method reduces the difficulty of optimization and saves a lot of function evaluations. Furthermore, the independent optimization of each sub-component can accelerate the convergence rate.

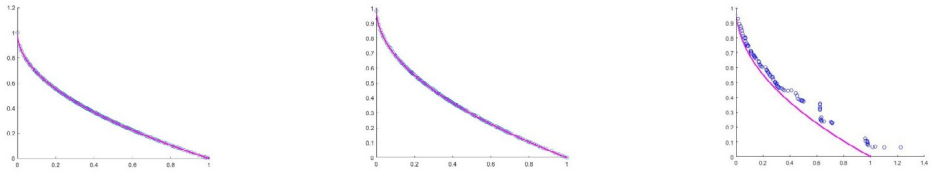


(a) PCA-MOEA(95%) (b) PCA-MOEA(10%) (c) MOEA/DVA



(d) LMEA (e) MOEA/D

Fig. 3. The PF obtained by five algorithms on UF2 function with 200 decision variables.



(a) PCA-MOEA(95%) (b) PCA-MOEA(10%) (c) MOEA/DVA

(d) LMEA (e) MOEA/D

Fig. 4. The PF obtained by five algorithms on ZDT4 function with 200 decision variables.

4.2.3. Results of test problem with 1000 variables

MOEA/DVA is promising for solving multi-objective problems with 200 variables in decision space, however, when the dimensionality increases to 1000, it sharply loses its efficiency, which is mainly on account of the large expenses of function evaluations in the procedure of decision variable analysis and interdependence analysis. Thus it is urgently needed to get the number of dimensionality reduced. In this part, we will make comparisons on UF test suites and WFG test suites with 1000 variables. These two test suites are adopted in [54].

In [54], a random-based dynamic grouping approach is proposed to solve MOPs with numerous decision variables, it decomposes the whole dimension into a lot of groups with the same size, each containing some variables. In this approach, the group size and components in each group are all dynamic. So in this part, we also add MOEA/D-RDG developed in [54] as a comparative algorithm. MOEA/D-RDG is implemented by ourselves in Matlab according to [54].

In these experiments, for fair comparison, we set population size as 300 for 2-objective functions and 600 for 3-objective functions. The population size and the adopted test problems stay the same with [54]. We compare PCA-MOEA with MOEA/DVA, LMEA, MOEA/D-RGD and MOEA/D. The statistical results of IGD over 30 runs of the compared algorithm can be found in Table 3.

In the table, the results of the best performing algorithm are highlighted. The Wilcoxon signed-rank test [41] is conducted to compare the significance of difference between PCA/MOEA and the compared algorithms. We can obviously see from the above table that on 1000 dimensional MOPs, PCA-MOEA has much better performance than MOEA/DVA, LMEA and MOEA/D on UF test problems. PCA-MOEA beats MOEA-RGD down on 8 UF functions but does not perform much well on WFG test problems. MOEA/RDG obtains 6 best results out of 9 problems. In this case, we kept ten percent of variance in these experiments hence the second parameter of PCA is set as 10%.

Another point our algorithm being better than other algorithms is that it saves much more computational costs. On 2-objective UF1–UF7 functions, our algorithm only needs 6.0E+5 evaluations to get satisfactory results, and 1.0E+6 for 3-objective UF8–UF10 functions, and for 3-objective WFG1–WFG9 functions,

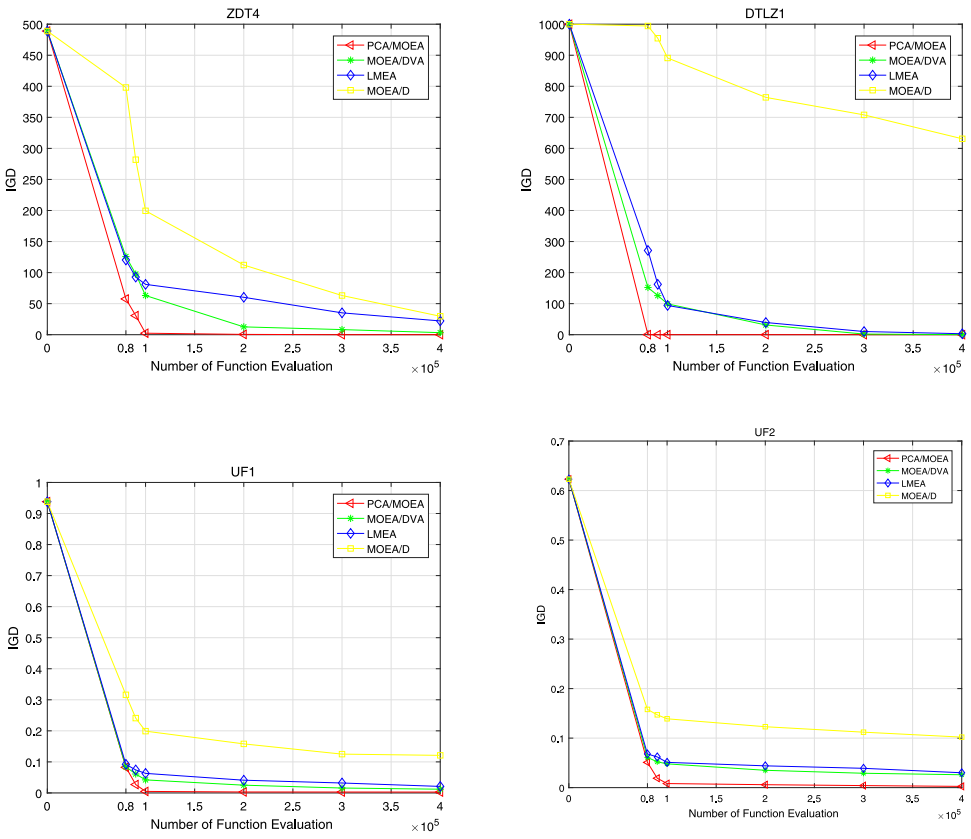


Fig. 5. The evolution curves of IGD values generated by the compared algorithms on four functions with 200 variables.

the number is 7.0E+5. While for MOEA/DVA, LMEA, MOEA/RDG, MOEA/D, they all need 1.0E+7 evaluations to get the final PF, which requires plenty of time.

4.3. Results of test problem with 5000 variables

In the former section, it has been demonstrated that for test functions with 30 to 1000 variables in decision space, our suggested algorithm is promising to solve them. To validate the scalability of the algorithm, in this part, the new algorithm is conducted on test suites with up to 5000 variables. We set the population size as 300 and the maximal function evaluations as 2 000 000. Due to the large dimensionality, we chose to set the percent of variance in PCA as 10%. The results are presented below.

From in Table 4, it can be observed that although the dimensionality is larger, the value of IGD is even less than the above mentioned results, which means that we can also get the satisfactory solutions in this case. And we can also find that after dimensionality reduction, the low representative dimension for different UF problems all become 21 for 5000 decision variables and 17 for 2000 decision variables, respectively. A problem with 21 or 17 convergence-related variables is much easier to deal with. During evolution, the interdependence between variables does not change, namely, and for UF test problems, there just exists sparse variable interactions. We should pay attention to the fact that the number of function evaluations is 2 000 000, which is not much larger compared with former experiments. This obviously demonstrates that our strategy can save computational costs.

For further investigation, Fig. 6 plots the average values of IGD over 20 runs of PCA-MOEA on UF1, UF4 and UF8 problems with various number of variables. As we can find obviously, the IGD

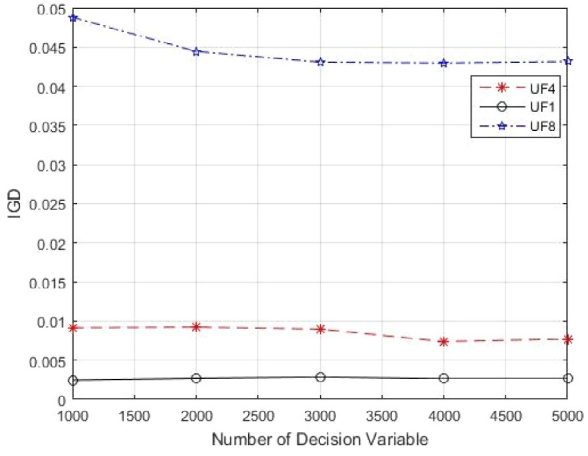


Fig. 6. IGD metric value of PCA-MOEA on UF1, UF4 and UF8 test problems with different number of decision variables, averaging over 20 runs.

value does not deteriorate as the number of variables increasing from 1000 to 5000. Under some circumstances, the value even gets smaller, which shows that the algorithm has good scalability.

It is worth noting that other algorithms cannot solve such large-scale problems due to too high computational costs. The efficiency of our algorithm can be validated effectively.

4.4. Experimental study on the effectiveness of PCA

Here, we present a table and analyze the components of each algorithm in the following parts.

Table 3
Average and standard deviation of IGD obtained by the compared algorithms on problems with 1000 variables.

Problems	PCA-MOEA	MOEA/DVA		LMEA		MOEA/D-RDG		MOEA/D	
UF1	**1.4917E−03** (3.9433E−03)	9.4342E−03 (4.2341E−03)	−	4.3791E−03 (2.4810E−03)	−	4.6088E−03 (8.7010E−04)	−	2.0250E−01 (4.7538E−02)	−
UF2	**1.8824E−03** (9.8941E−03)	6.3209E−03 (3.2052E−03)	−	2.0764E−02 (6.4822E−03)	−	5.4708E−03 (2.5791E−03)	−	9.0383E−02 (4.2934E−02)	−
UF3	3.6242E−02 (4.3213E−03)	2.0931E−02 (3.2395E−03)	≈	7.4728E−02 (5.3829E−03)	+	**2.6605E−03** (6.5686E−04)	+	9.1349E−02 (4.2983E−02)	≈
UF4	**7.2061E−03** (2.4920E−02)	4.0328E−02 (4.3291E−04)	−	7.6210E−03 (4.3711E−04)	≈	1.0796E−01 (1.2308E−02)	−	8.3742E−02 (4.2938E−02)	−
UF5	**2.5029E−02** 3.9130E−04	4.2980E−01 8.3712E−03	−	3.6433E−01 (4.8792E−03)	−	1.0984E−01 (1.3889E−01)	−	9.4829E−01 (7.4872E−02)	−
UF6	**1.0547E−02** (4.2932E−03)	5.3920E−02 (1.4391E−03)	−	4.6511E−02 (2.6549E−03)	−	2.4528E−02 (9.5862E−04)	−	7.4763E−01 (4.3872E−02)	−
UF7	3.2817E−02 (2.3873E−03)	8.4240E−02 (8.3092E−03)	−	5.7547E−02 (5.7671E−03)	−	**5.3030E−03** (7.2415E−04)	+	7.3729E−01 (4.3710E−02)	−
UF8	**2.8438E−02** (7.6791E−02)	5.3920E−01 (4.3239E−02)	−	7.6535E−01 (1.3431E−02)	−	1.6902E−01 (1.1688E−01)	−	4.2931E−01 (9.3861E−02)	−
UF9	2.3988E−02 (3.2938E−02)	4.2989E−01 (4.3942E−02)	−	**3.6541E−03** (5.7641E−03)	+	2.6720E−02 (1.6021E−02)	−	6.2094E−01 (3.2339E−02)	−
UF10	9.0738E−02 (3.2187E−03)	7.1352E+00 (7.8921E−02)	−	2.0582E+00 (8.7735E−03)	−	2.5661E+00 (2.0235E−01)	−	9.3719E−01 (3.2188E−02)	−
WFG1	9.0033E−01 (3.2032E−03)	2.3981E+00 (3.4894E−02)	−	2.5068E+00 (4.7641E−03)	−	9.0189E−01 (1.2965E−02)	≈	3.4792E+00 (3.3019E−02)	−
WFG2	2.9691E−01 (8.2712E−01)	5.3903E−01 (3.4740E−02)	−	3.0864E−01 (7.5432E−02)	≈	**1.2485E−01** (2.4283E−02)	+	1.2817E+00 (4.8392E−02)	−
WFG3	**1.7017E−02** (3.2091E−02)	5.3910E+00 (4.2139E−02)	−	7.9849E−01 (3.5312E−03)	−	1.5543E+00 (1.7457E−02)	−	2.4729E+00 (1.2934E−03)	−
WFG4	5.0524E−01 (3.2934E−02)	8.4912E−01 (3.4929E−03)	−	**8.7512E−02** (5.6412E−03)	+	8.8967E−02 (1.4603E−03)	+	7.8371E−01 (4.3910E−02)	−
WFG5	4.3920E−01 (9.4883E−03)	5.3291E−01 (4.9024E−03)	−	3.6511E−01 (2.7759E−03)	≈	**1.1670E−01** (2.2352E−04)	+	1.6791E+00 (3.2938E−03)	−
WFG6	2.4909E−01 (1.3249E−03)	**4.3294E−02** (3.4898E−03)	+	6.3712E−02 (4.2810E−03)	+	8.8139E−02 (5.1434E−04)	+	8.3742E−01 (1.2938E−02)	−
WFG7	7.3981E−01 (7.9301E−03)	4.3920E−01 (9.4232E−03)	≈	9.8273E−02 (2.3719E−04)	+	**8.7095E−02** (1.1844E−04)	+	1.4923E+00 (9.4728E−03)	−
WFG8	9.1930E−01 (7.3889E−03)	4.4903E−01 (5.3291E−03)	≈	5.2718E−01 (7.4390E−03)	≈	**1.0856E−01** (3.1555E−03)	+	8.3872E−01 (3.0984r-3)	−
WFG9	8.2839E−01 (3.1283E−03)	5.3931E−01 (3.2939E−03)	≈	5.3719E−01 (5.3182E−03)	≈	**9.0481E−02** (1.0489E−03)	+	7.3763E−01 (9.3848E−02)	≈
+/−/		1/14/4		5/9/5		9/9/1		0/17/2	

Table 4
Average and standard deviation of IGD obtained the algorithms on problems with 2000 and 5000 variables.

Problems	2000-variable	5000-variable
UF1	2.6929e−3(5.3811e−4)	2.6471e−3(3.2473e−4)
UF2	2.7500e−3(7.5819e−4)	1.4911e−3(8.3728e−4)
UF3	6.9666e−2(4.2819e−4)	3.5940e−2(5.2938e−5)
UF4	9.2341e−3(6.5829e−4)	7.7227e−3(2.3948e−4)
UF5	2.1823e−2(2.3180e−4)	2.2412e−2(4.2938e−4)
UF6	3.4261e−2(7.5918e−5)	3.2074e−2(4.9847e−5)
UF7	3.8879e−2(3.1823e−3)	3.3842e−2(3.2938e−3)
UF8	4.4441e−2(9.7281e−2)	4.4154e−2(4.3984e−3)
UF9	3.7932e−2(3.2810e−3)	3.1010e−2(2.9381e−3)
UF10	9.1263e−2(3.1890e−3)	1.0446e−1(5.4421e−3)

In order to verify the effectiveness of PCA, experiments with and without PCA are carried out under the same algorithm framework. We choose the UF1 test problem and run the above two algorithms on 30, 100, 200, 300, 400, 500 and 1000 dimensions of UF1 respectively. As mentioned earlier, the introduction of PCA will greatly save resources, which also means less time. We compare the time required for the two kinds of experiments when the IGD is around 0.003 in Table 6.

Table 5
The components of each algorithm.

Algorithm	Clustering approach	Dimensionality reduction technique	Optimization method
PCA-MOEA	k-means	PCA	Cooperative coevolution
PCA-MOEA without PCA	k-means	No	Cooperative coevolution
MOEA/DVA	Decision variable analysis	No	Cooperative coevolution and MOEA/D
LMEA	k-means	No	Convergence optimization and diversity optimization
MOEA/D	No	No	MOEA/D

Fig. 7 is an image representation. From Table 5 and Fig. 7, we can see that the use of PCA significantly enhances the effectiveness of the algorithm and achieves excellent results in a very short time. The higher the dimension is, the more obvious the superiority is. In conclusion, the proposed PCA-MOEA

Table 6
Comparisons of time consumed with PCA and without PCA.

Dimension	30	100	200	300	400	500	1000
Time consumed with PCA	0.57 s	0.76 s	0.98 s	1.17 s	1.37 s	1.72 s	3.06 s
Time consumed without PCA	5.43 s	22 s	42 s	137 s	239 s	412 s	2520 s

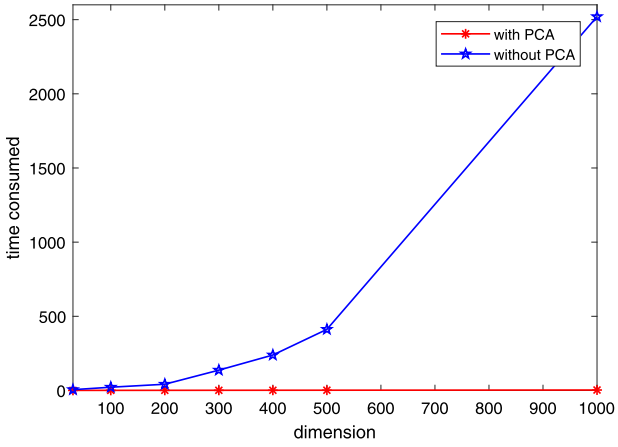


Fig. 7. Time Consumed by algorithms with PCA and without PCA.

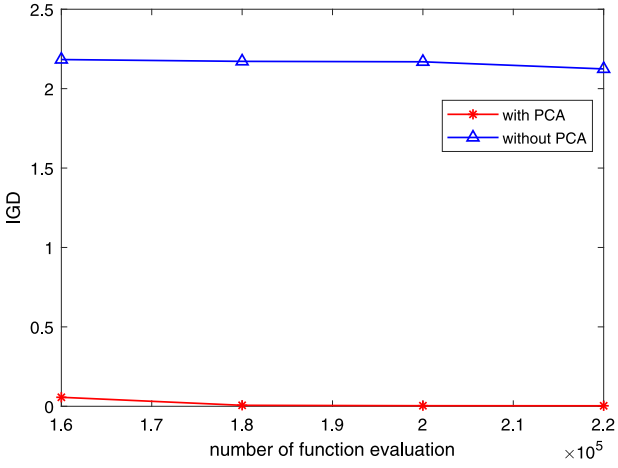


Fig. 8. Convergence Results by Algorithms with PCA and without PCA.

algorithm can reduce the computation time of solving large-scale multi-objective optimization problems and has good performance. The main reason is that the original large-scale multi-objective optimization problem transforms a low-dimensional optimization problem after the dimensionality reduction, and the search complexity decreases significantly. Therefore, PCA-MOEA is very competitive.

When solving UF1 with 1000 decision variables, PCA-MOEA only needs 200 000 function evaluations to obtain excellent solutions, while in PCA-MOEA without PCA, it is still in the stage of interdependence analysis, in which a large number of function evaluations are consumed because of the large number of variables to obtain the correlation relationship between variables, so it has not entered the optimization stage yet. Fig. 8 shows the convergence results.

4.5. Sensitivity analysis of parameter

This section is designed to analyze the impact of the parameter-percent of variance setting in PCA. We conduct our

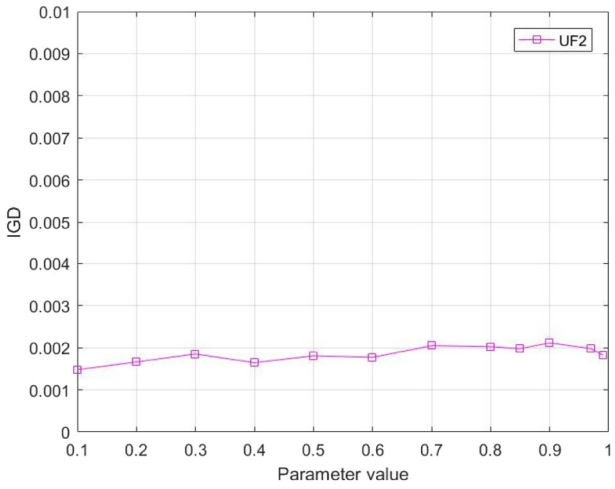


Fig. 9. IGD metric value of PCA-MOEA on UF2 problem with different parameter value, averaging over 20 runs.

algorithm on UF2 problem in a set of different conditions. The number of variables is set as 5000. The results after dimensionality reduction are displayed in Table 7. Fig. 9 also shows the IGD metric values with different parameter setting. It indicates that the final results do not have a great variation as the parameter changes. In this example, a smaller setting can get a more ideal result. On the other aspect, the number of variables after reduction gets bigger as the parameter value increases and thus a larger number of function evaluations is demanded to accomplish the optimization process. It can be concluded that a smaller parameter value has the ability to save computational costs on the condition that the final results are desirable. It should be noted that the most proper value is correlated with several other parameters, such as population size, the selected test problem. When we deal with problems having more than 1000 decision variables, we can choose a smaller parameter to see if the results are satisfactory, if not, then we can try a larger parameter.

4.6. Applying PCA-MOEA to wind turbine placement

Wind power is an increasingly important source of growth for the global renewable energy market. Wind Turbine Placement Optimization is the process of determining the layout of a wind turbine in a wind farm. This maximizes the cost-effectiveness of installation of wind power plants, thereby increasing their competitiveness in the renewable energy market. In this paper, we consider three combinations [55] to test the performance of our algorithm, that is, the following objective combinations: (1) maximizing the power output (yield) and minimizing the area of the convex hull; (2) maximizing the power output (yield) and minimizing the total distance of the minimum spanning tree; (3) a combination of power, convex hull area and minimum spanning tree.

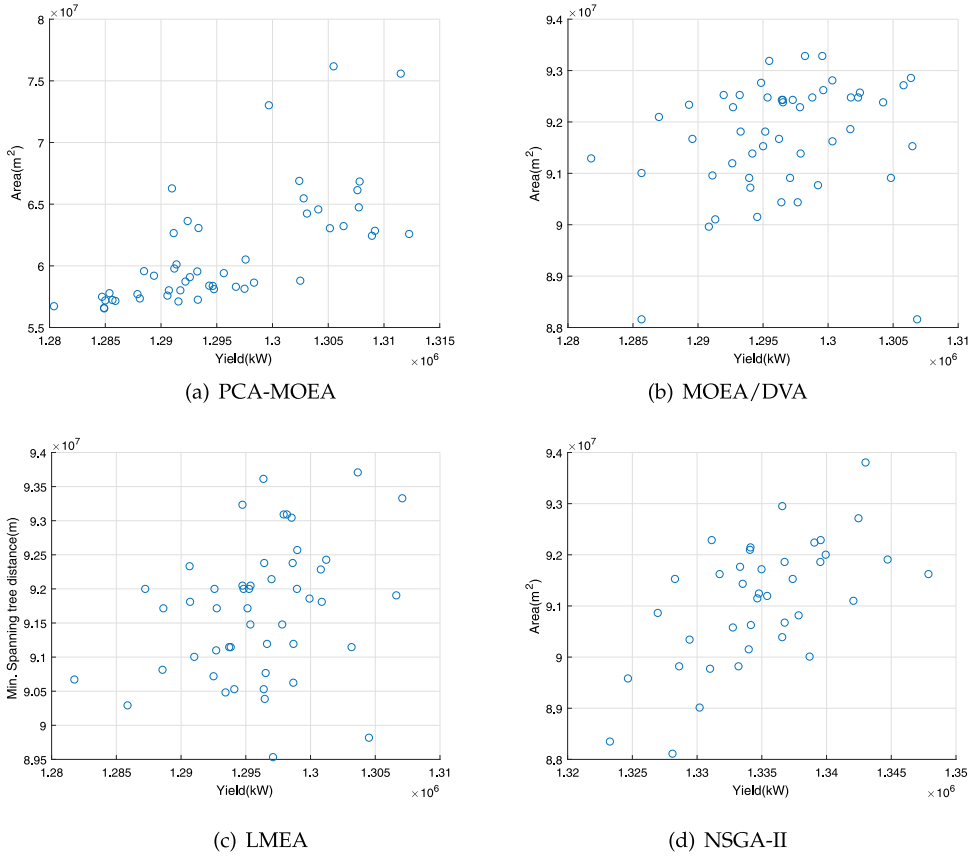
The Euclidean minimum spanning tree is used to calculate the minimum cable length required to connect all wind turbines in a particular wind farm structure. The calculation method is to first construct a complete map on the set of points representing

Table 7
The variation of IGD value with parameters.

Parameter	10%	20%	30%	40%	50%	60%
Reduced dimension	21	43	67	93	120	149
IGD	1.4801e−3	1.6628e−3	1.8537e−3	1.6453e−3	1.8097e−3	1.7721e−3
Function evaluation	5e5	7e5	1e6	1.4e6	1.8e6	2e6
Parameter	70%	80%	85%	90%	97%	99%
Reduced dimension	180	215	234	254	285	294
IGD	2.0497e−3	2.0265e−3	1.9794e−3	2.1146e−3	1.9870e−3	1.8273e−3
Function evaluation	2.8e6	3e6	3.5e6	4.5e6	5e6	5.4e6

Table 8
Average and standard deviation value of relative HV by PCA-MOEA, MOEA/DVA, and LMEA for Wind Turbine Placement Optimization with 200 turbines.

Combination		PCA-MOEA	MOEA-DVA	LMEA	NSGA-II
1	Scenario 1	**0.9999(0.0325)**	0.9360(0.0245)	0.5145(0.0962)	0.9652(0.0325)
	Scenario 2	**0.9969(0.0523)**	0.9522(0.0265)	0.8831(0.0652)	0.9437(0.0259)
2	Scenario 1	**0.9977(0.0298)**	0.9896(0.0685)	0.9936(0.0631)	0.9885(0.0325)
	Scenario 2	**0.9865(0.0256)**	0.9830(0.0365)	0.9825(0.0360)	0.9856(0.0532)
3	Scenario 1	**0.9922(0.0092)**	0.9788(0.0101)	0.9290(0.0206)	0.9652(0.0355)
	Scenario 2	**0.9882(0.0156)**	0.9527(0.0139)	0.9489(0.0130)	0.9685(0.0326)



(a) PCA-MOEA (b) MOEA/DVA

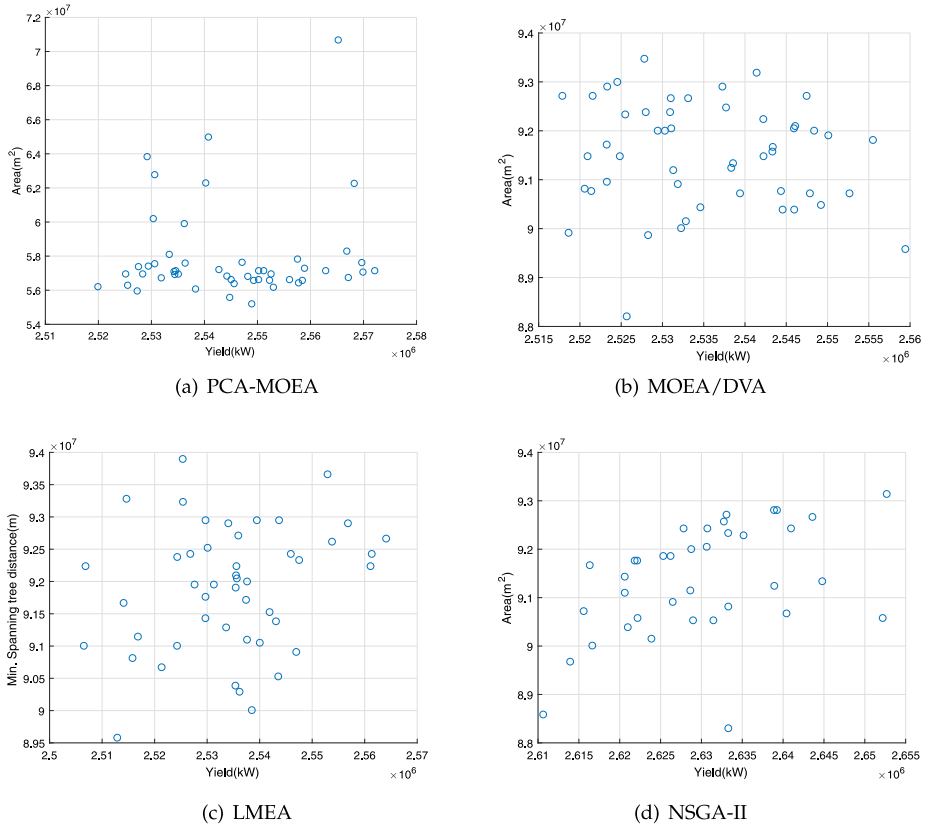
(c) LMEA (d) NSGA-II

Fig. 10. The obtained fitness value by the four algorithms with 200 turbines in *Scenario 1* when maximizing the power output (yield) and minimizing the area of the convex hull.

the wind turbine, these point sets being given by the Euclidean distance between any pair of turbines. The minimum spanning tree for this graph is calculated and used as a target for cable length costs.

In our experiments, the cost of a convex hull is defined as the area enclosed by a set of points that make up a convex hull. This value is the minimum land area required for wind farm layout. We calculate convex hull using Graham's scan algorithm. And the introduction to calculating energy output can be found in [55]. The code of these objective combinations can be found in https://github.com/d9w/WindFLO.

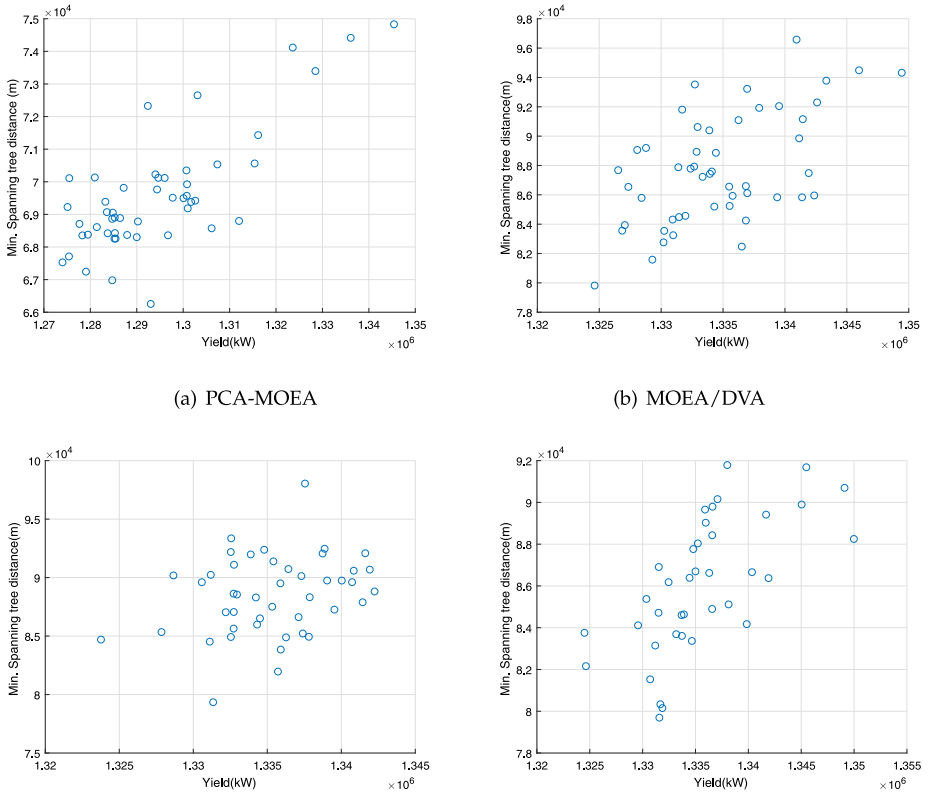
In order to compare the performance of our algorithm, we choose MOEA/DVA, LMEA and NSGA-II as comparison algorithms. And NSGA-II is implemented by ourselves in Matlab according to [1]. And for initialization, we generates individuals that meet the constraints. And if the positions of turbines do not meet constraints, they are deleted and generate new solutions in their domain that satisfy constraints. And we choose 200 turbines scenario. *Scenario 1* and *Scenario 2* are different scenarios and the latter is rather complex. The population size is 50, which is the same as [55]. For all the experiment, the maximum number of function evaluations is set as 1 000 000, and all the tests are



(a) PCA-MOEA (b) MOEA/DVA

(c) LMEA (d) NSGA-II

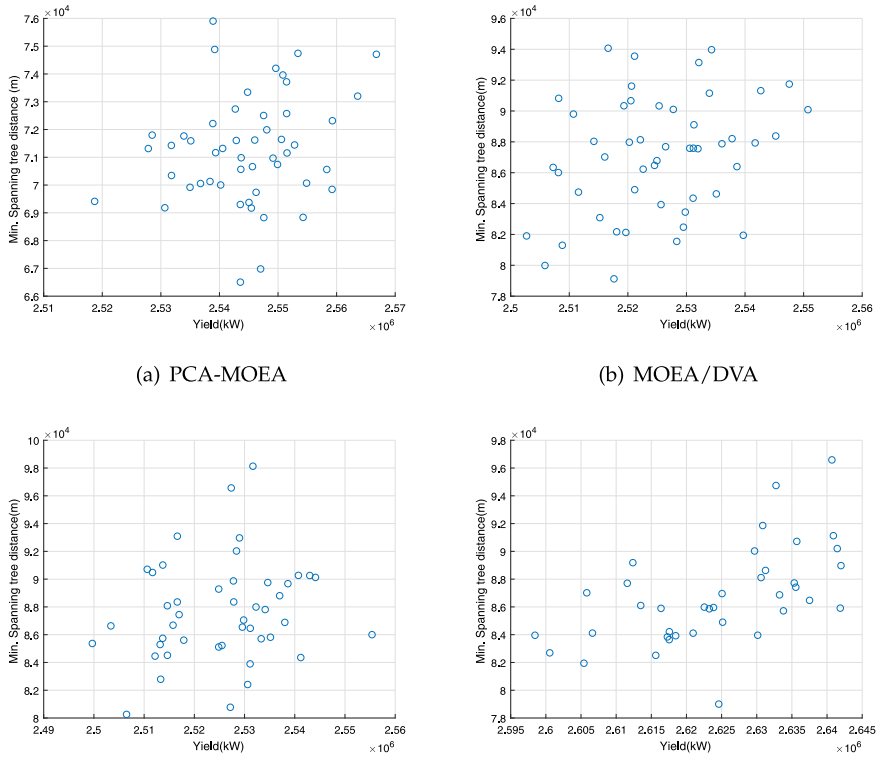
Fig. 11. The obtained fitness value by the four algorithms with 200 turbines in *Scenario 2* when maximizing the power output (yield) and minimizing the area of the convex hull.



(a) PCA-MOEA (b) MOEA/DVA

(c) LMEA (d) NSGA-II

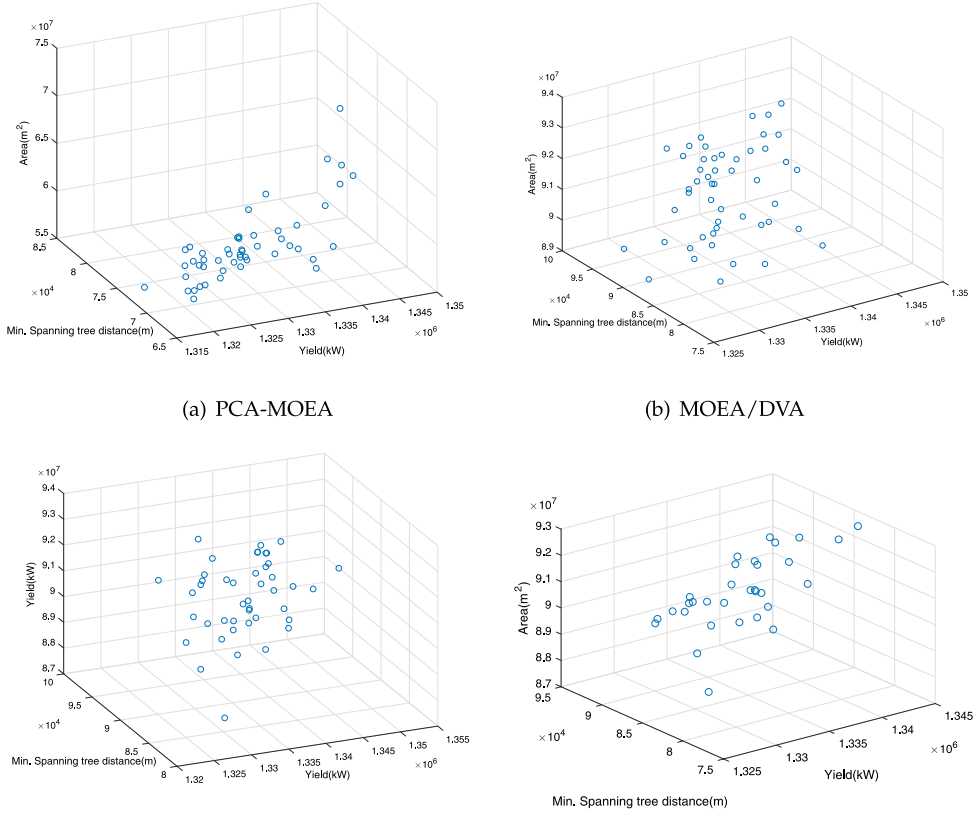
Fig. 12. The obtained fitness value by the four algorithms with 200 turbines in *Scenario 1* when maximizing the power output (yield) and minimizing the total distance of the minimum spanning tree.



(a) PCA-MOEA (b) MOEA/DVA

(c) LMEA (d) NSGA-II

Fig. 13. The obtained fitness value by the four algorithms with 200 turbines in *Scenario 2* when maximizing the power output (yield) and minimizing the total distance of the minimum spanning tree.



(a) PCA-MOEA (b) MOEA/DVA

(c) LMEA (d) NSGA-II

Fig. 14. The obtained fitness value by the four algorithms with 200 turbines in *Scenario 1* when combining power, convex hull area and minimum spanning tree.

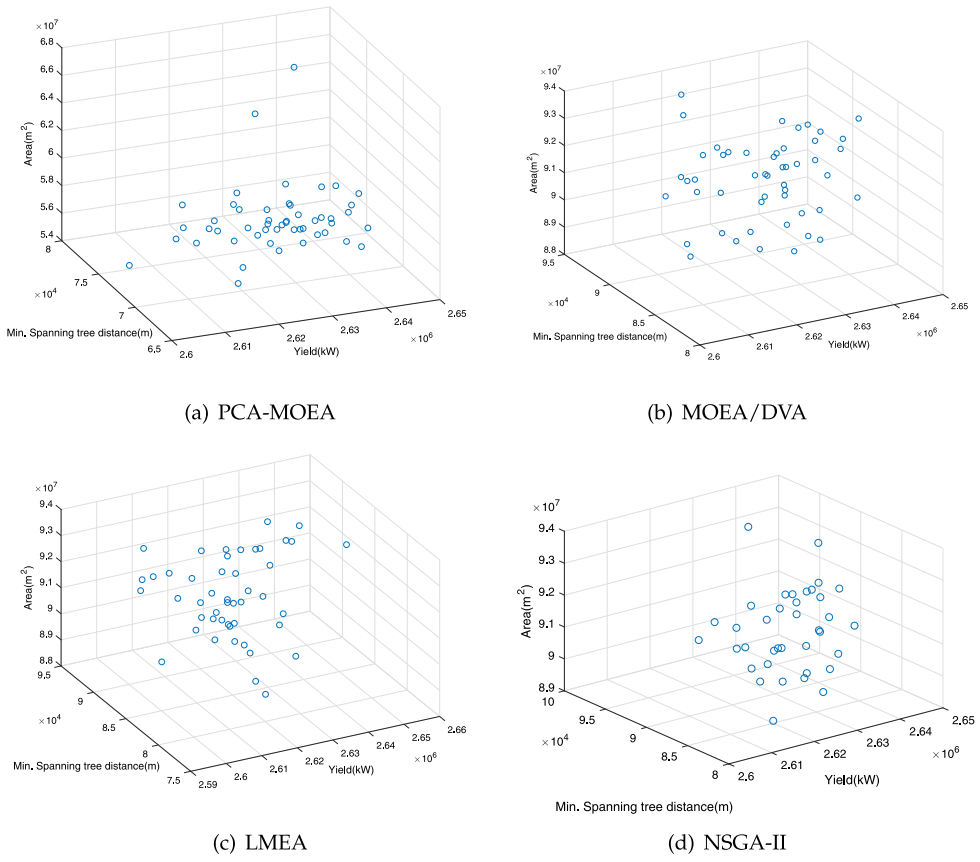


Fig. 15. The obtained fitness value by the four algorithms with 200 turbines in *Scenario 2* when combining power, convex hull area and minimum spanning tree.

run for 20 times independently. The obtained objective value will be normalized into the following ranges: area 10 km^2–100 km^2, yield 1 000 000 kW–1 440 000 kW, spanning tree length 60 km–150 km.

Table 8 lists the relative HV values obtained by the three algorithm. and the reference point for HV calculation is set to $(10 * 1.1, 1\,440\,000 * 1.1, 150 * 1.1)$. The relative HV is the ratio of the HV of the obtained objective value to the HV of the lowest point of the normalized region.

In the table, the results of the best performing algorithm are highlighted. From the results in Table 8, we can see that the proposed algorithm can deal with wind turbine placement optimization better than MOEA-DVA and LMEA. And in more complex scenarios, our algorithm has more obvious advantages. And we plot the result of the three algorithms from Figs. 10 to 15.

From Figs. 10 and 11, that is plotting the objective combination 1: maximizing the power output (yield) and minimizing the area of the convex hull, we can see that PCA-MOEA can obtain similar yield while greatly saving the area of the convex hull when comparing with MOEA/DVA and LMEA. For example, in *Scenario 1*, when the yield is $1.3 * 10^6$ kW, the area PCA-MOEA needs is just about $7.4 * 10^7$ m^2, but MOEA/DVA and LMEA need $9.08 * 10^7$ m^2 and $9.18 * 10^7$ m^2 respectively.

For the objective combination 2: maximizing the power output (yield) and minimizing the total distance of the minimum spanning tree, we can see from Figs. 12 and 13 that PCA-MOEA can use less cable length to achieve higher productivity when comparing with MOEA/DVA and LMEA. And in *Scenario 2*, when the yield is $2.54 * 10^6$ kW, the cable length PCA-MOEA needs is just about $7.09 * 10^4$ m but MOEA/DVA and LMEA need $8.2 * 10^4$ m and $8.42 * 10^4$ m respectively.

As for three-objective problems, i.e. the objective combination 3, we can see from the Figs. 14 and 15 that the proposed algorithm has smaller area and minimum spanning tree while obtaining higher productivity when compared with the other two algorithms, which means that our algorithm can achieve high productivity and save some costs at the same time. So our algorithm can deal with the Wind Turbine Placement Optimization problems.

5. Conclusion

This article suggests a principal component analysis based multi-objective evolutionary algorithm for multi-objective problems with numerous decision variables, in which a clustering method is also applied. We conduct our algorithm on some benchmark functions with low dimensionality, in the above mentioned experiment, i.e., UF test suits and WFG test suites with 30 variables. We can get the ideal results with less computational costs by comparing our algorithm with MOEA/DVA. Besides, we have also applied our technique on several test problems with 200 variables and 1000 variables, which validate the effectiveness and efficiency of our algorithm. Finally, we made a try to solve problems with up to 5000 variables. It is inspiring that the proposed approach can also tackle with such problems well, obtain solutions that both convergence and diversity are satisfactory.

Although PCA/MOEA is promising in solving large scale optimization problems, there are still some problems that need to be solved in the years ahead. One of its drawbacks is that it consumes too much computing resources in the variable grouping stage. How to deal with decision variables in a more effective way is a valuable research direction. Another interesting direction on the parameter setting is PCA.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.asoc.2020.106120.

CRediT authorship contribution statement

Ruochen Liu: Conceptualization, Methodology, Software. **Rui Ren:** Software, Formal analysis, Writing - original draft. **Jin Liu:** Software, Formal analysis, Investigation, Writing - review & editing. **Jing Liu:** Data curation, Investigation.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Nos. 61876141 and 61373111), and the Provincial Natural Science Foundation of Shaanxi of China (No. 2019JZ-26).

References

[1] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[2] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[3] K. Deb, H. Jain, An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints, IEEE Trans. Evol. Comput. 18 (4) (2014) 577–601.
[4] C. Liu, J. Liu, Z. Jiang, A multiobjective evolutionary algorithm based on similarity for community detection from signed social networks, IEEE Trans. Cybern. 44 (12) (2014) 2274–2287.
[5] H. Zhu, Y. Shi, Brain storm optimization algorithm for full area coverage of wireless sensor networks, in: 2016 Eighth International Conference on Advanced Computational Intelligence (ICACI), 2016, pp. 14–20.
[6] Y. Guo, D. Liu, M. Chen, Y. Liu, An energy-efficient coverage optimization method for the wireless sensor networks based on multi-objective quantum-inspired cultural algorithm, in: Advances in Neural Networks – ISNN 2013, 2013, pp. 343–349.
[7] Y. Guo, D. Liu, Multi-population cooperative particle swarm cultural algorithms, in: 2011 Seventh International Conference on Natural Computation, Vol. 3, 2011, pp. 1351–1355.
[8] Y. Guo, P. Zhang, J. Cheng, C. Wang, D. Gong, Interval multi-objective quantum-inspired cultural algorithms, Neural Comput. Appl. 30 (3) (2018) 709–722.
[9] L.M. Antonio, C.A.C. Coello, Use of cooperative coevolution for solving large scale multiobjective optimization problems, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 2758–2765.
[10] M. Guo, C. Wang, Multi-objective quantum cultural algorithm and its application in the wireless sensor networks' energy-efficient coverage optimization, in: Intelligent Data Engineering and Automated Learning – IDEAL 2013, 2013, pp. 161–167.
[11] R. Scheffermann, M. Bender, A. Cardeneo, Robust solutions for vehicle routing problems via evolutionary multiobjective optimization, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 1605–1612.
[12] Y. Guo, J. Cheng, S. Luo, D. Gong, Y. Xue, Robust dynamic multi-objective vehicle routing optimization method, IEEE/ACM Trans. Comput. Biol. Bioinform. 15 (6) (2018) 1891–1903.
[13] M.A. Potter, K.A.D. Jong, A cooperative coevolutionary approach to function optimization, Third Parallel Prob. Solving Form Nat. 866 (1994) 249–257.
[14] X. Li, Y. Mei, X. Yao, M.N. Omidvar, Cooperative co-evolution with differential grouping for large scale optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 378–393.
[15] W. Chen, T. Weise, Z. Yang, K. Tang, Large-scale global optimization using cooperative coevolution with variable interaction learning, in: International Conference on Parallel Problem Solving from Nature, 2010, pp. 300–309.
[16] Y. Sun, M. Kirley, S.K. Halgamuge, Extended differential grouping for large scale global optimization with direct and indirect variable interactions, 50 (Anno 26) (2) (2015) 313–320.
[17] M.N. Omidvar, X. Li, X. Yao, Cooperative co-evolution with delta grouping for large scale non-separable function optimization, in: Evolutionary Computation, 2011, pp. 1–8.
[18] Z. Yang, K. Tang, X. Yao, Multilevel cooperative coevolution for large scale optimization, in: Evolutionary Computation, 2008, pp. 1663–1670.
[19] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybern. 45 (2) (2015) 191.
[20] X. Li, X. Yao, X. Li, X. Yao, Cooperatively coevolving particle swarms for large scale optimization, IEEE Trans. Evol. Comput. 16 (2) (2012) 210–224.
[21] K.H. Hajikolaei, Z. Pirmoradi, G.H. Cheng, G.G. Wang, Decomposition for large-scale global optimization based on quantified variable correlations uncovered by metamodelling, Eng. Optim. 47 (4) (2015) 429–452.
[22] J. Ghorpadeaher, V.A. Metre, Clustering multidimensional data with pso based algorithm, Comput. Sci. (2014).
[23] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inform. Sci. 291 (6) (2015) 43–60.
[24] W.N. Chen, J. Zhang, H.S.H. Chung, W.L. Zhong, W.G. Wu, Y.H. Shi, A novel set-based particle swarm optimization method for discrete optimization problems, IEEE Trans. Evol. Comput. 14 (2) (2010) 278–300.
[25] W.N. Chen, J. Zhang, Y. Lin, N. Chen, Z.H. Zhan, S.H. Chung, Y. Li, Y.H. Shi, Particle swarm optimization with an aging leader and challengers, IEEE Trans. Evol. Comput. 17 (2) (2013) 241–258.
[26] W.J. Yu, M. Shen, W.N. Chen, Z.H. Zhan, Y.J. Gong, Y. Lin, O. Liu, J. Zhang, Differential evolution with two-level parameter adaptation, IEEE Trans. Cybern. 44 (7) (2014) 1080–1099.
[27] K.P. Wong, Z.Y. Dong, Differential evolution, an alternative approach to evolutionary algorithm, in: International Conference on Intelligent Systems Application to Power Systems, 2005.
[28] X. Zhang, Y. Tian, R. Cheng, Y. Jin, A decision variable clustering based evolutionary algorithm for large-scale many-objective optimization, IEEE Trans. Evol. Comput. 22 (1) (2018) 97–112.
[29] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, M. Gong, A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables, IEEE Trans. Evol. Comput. 20 (2) (2016) 275–298.
[30] H. Wang, L. Jiao, R. Shang, S. He, F. Liu, A memetic optimization strategy based on dimension reduction in decision space, Evol. Comput. 23 (1) (2015) 69–100.
[31] H. Wang, Y. Jin, Efficient nonlinear correlation detection for decomposed search in evolutionary multi-objective optimization, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 649–656.
[32] J. Shlens, A tutorial on principal component analysis, in: Systems Neurobiology Laboratory, Salk Institute for Biological Studies, 2005.
[33] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, J. Wu, Moea/d with adaptive weight adjustment, Evol. Comput. 22 (2) (2014) 231–264.
[34] Z. He, G.G. Yen, Diversity improvement in decomposition-based multi-objective evolutionary algorithm for many-objective optimization problems, in: IEEE International Conference on Systems, Man and Cybernetics, 2014, pp. 2409–2414.
[35] H.L. Liu, F. Gu, Q. Zhang, Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems, IEEE Trans. Evol. Comput. 18 (3) (2014) 450–455.
[36] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, Lecture Notes in Comput. Sci. 3242 (2004) 832–842.
[37] N. Beumea, M. Emmerich, Sms-emoa: Multiobjective selection based on dominated hypervolume, European J. Oper. Res. 181 (3) (2007) 1653–1669.
[38] C.A.R.g. Villalobos, C.A.C. Coello, A New Multi-Objective Evolutionary Algorithm Based on a Performance Assessment Indicator, 20 (1) (2012) 16–37.
[39] J.J. Durillo, A.J. Nebro, C.A.C. Coello, F. Luna, E. Alba, A comparative study of the effect of parameter scalability in multi-objective metaheuristics, in: Evolutionary Computation, 2008, pp. 1893–1900.
[40] G.R. Zavala, A.J. Nebro, F. Luna, C.A.C. Coello, A survey of multi-objective metaheuristics applied to structural optimization, Struct. Multidiscip. Optim. 49 (4) (2013) 537–558.
[41] D.A. Wolfe, M. Hollander, Nonparametric statistical methods, in: Biostatistics and Microbiology: A Survival Manual, 2009.
[42] F.R.S. Karlpearson, Liii. On lines and planes of closest fit to systems of points in space, Phil. Mag. 2 (11) (1901) 559–572.
[43] R. Frisch, Correlation and Scatter in Statistical Variables, Sosialkonomisk Institutt Universitetet i Oslo, 1951.
[44] K.T. Fang, D.K.J. Lin, Uniform experimental designs and their applications in industry, Handbook of Statist. 22 (03) (2003) 131–170.
[45] R. Bro, E. Acar, T.G. Kolda, Resolving the sign ambiguity in the singular value decomposition, J. Chemom. 22 (2) (2008) 135–140.
[46] R. Storn, K. Price, Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (4) (1997) 341–359.
[47] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable multi-objective optimization test problems, in: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, 2002, pp. 825–830.
[48] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: Empirical results, Evol. Comput. 8 (2) (2000) 173–195.

[49] Q. Zhang, A. Zhou, S. Zhao, P.N. Suganthan, W. Liu, S. Tiwari, Multiobjective Optimization Test Instances for the CEC 2009 Special Session and Competition, special Session on Performance Assessment of Multi-Objective Optimization Algorithms, Technical Report, 264, University of Essex, Colchester, UK and Nanyang technological University, Singapore, 2008.

[50] S. Huband, P. Hingston, L. Barone, R.L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477–506.

[51] Q. Zhang, W. Liu, H. Li, The performance of a new version of Moea/d, in: Evolutionary Computation, 2009. CEC '09. IEEE Congress on, 2009, pp. 203–208.

[52] H. Li, Q. Zhang, Optimization problems with complicated pareto sets, Moea/d and Nsga-ii, IEEE Trans. Evol. Comput. 13 (2) (2009) 284–302.

[53] Y. Tian, R. Cheng, X. Zhang, Y. Jin, Platemo: A matlab platform for evolutionary multi-objective optimization [educational forum], IEEE Comput. Intell. Mag. 12 (4) (2017) 73–87.

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

[55] R. Tran, J. Wu, C. Denison, T. Ackling, M. Wagner, F. Neumann, Fast and effective multi-objective optimisation of wind turbine placement, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, ACM, 2013, pp. 1381–1388, http://dx.doi.org/10.1145/2463372.2463541, URL http://doi.acm.org/10.1145/2463372.2463541.