Applied Soft Computing 12 (2012) 3526–3538

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

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

A regularity model-based multiobjective estimation of distribution algorithm with reducing redundant cluster operator

Yong Wang[a,b], Jian Xiang[a,b,*], Zixing Cai[a,b]

[a] School of Information Science and Engineering, Central South University, Changsha 410083, PR China
[b] Hunan Engineering Laboratory for Advanced Control and Intelligent Automation, Changsha 410083, PR China

ARTICLE INFO

Article history:
Received 6 July 2011
Received in revised form 24 May 2012
Accepted 6 June 2012
Available online 20 July 2012

Keywords:
Estimation of distribution algorithm
Multiobjective optimization
Modeling
Reducing redundant cluster operator

ABSTRACT

A regularity model-based multiobjective estimation of distribution algorithm (RM-MEDA) has been proposed for solving continuous multiobjective optimization problems with variable linkages. RM-MEDA is a kind of estimation of distribution algorithms and, therefore, modeling plays a critical role. In RM-MEDA, the population is split into several clusters to build the model. Moreover, the fixed number of clusters is recommended in RM-MEDA when solving different kinds of problems. However, based on our experiments, we find that the number of clusters is problem-dependent and has a significant effect on the performance of RM-MEDA. Motivated by the above observation, in this paper we improve the clustering process and propose a *r*educing *r*edundant *c*luster *o*perator (RRCO) to build more precise model during the evolution. By combining RRCO with RM-MEDA, we present an improved version of RM-MEDA, named IRM-MEDA. In this paper, we also construct four additional continuous multiobjective optimization test instances. The experimental results have shown that IRM-MEDA outperforms RM-MEDA in terms of efficiency and effectiveness. In particular, IRM-MEDA performs on average 31.67% faster than RM-MEDA.

1. Introduction

Many optimization problems involve not one but several objectives which should be optimized simultaneously. This kind of problems is considered as multiobjective optimization problems (MOPs). In this paper, we consider the following continuous MOPs:

$$\text{minimize} \quad \vec{y} = \vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) \tag{1}$$

where $\vec{x} = (x_1, \ldots, x_n) \in X \subseteq R^n$ is the decision vector, X is the decision space, $\vec{y} \in Y \subseteq R^m$ is the objective vector, and Y is the objective space.

There are some basic definitions in multiobjective optimization, which are introduced as follows.

Definition 1. Given two decision vectors $\vec{a} = (a_1, \ldots, a_n)$ and $\vec{b} = (b_1, \ldots, b_n)$, if $\forall i \in \{1, \ldots, m\}, f_i(\vec{a}) \leq f_i(\vec{b})$ and $\exists j \in \{1, \ldots, m\}, f_j(\vec{a}) < f_j(\vec{b})$, we say $\vec{a}$ Pareto dominates $\vec{b}$, denoted as $\vec{a} \prec \vec{b}$.

Definition 2. A decision vector $\vec{x} \in X$ is called Pareto optimal solution if there does not exist another decision vector $\vec{x}' \in X$ such that $\vec{x}' \prec \vec{x}$.

Definition 3. The *Pareto set* (*PS*) is the set of all the Pareto optimal solutions:

$$PS = \{\vec{x} \in X | \neg \exists \vec{x}' \in X, \vec{x}' \prec \vec{x}\} \tag{2}$$

The solutions in the *PS* are also called nondominated solutions.

Definition 4. The *Pareto front* (*PF*) is the set of the objective vectors of all the Pareto optimal solutions:

$$PF = \{\vec{f}(\vec{x}) | \vec{x} \in PS\} \tag{3}$$

For MOPs, in most cases, we cannot find a single solution to optimize all the objectives at the same time. Therefore, we have to balance them and find a set of optimal tradeoffs, i.e., *Pareto set* (*PS*) in the decision space and *Pareto front* (*PF*) in the objective space, respectively. Since evolutionary algorithms (EAs) deal with a group of candidate solutions simultaneously, it seems to be natural to use EAs for finding a group of Pareto optimal solutions when solving MOPs. Vector evaluation genetic algorithm (VEGA), introduced by Schaffer [1] in 1980s, is the first actual implementation of EAs to solve MOPs. After that, a considerable number of multiobjective evolutionary algorithms (MOEAs) have been proposed due to increasing interest in solving MOPs by EAs.

The development of MOEAs can be briefly divided into three generations [2,3]. In the first generation of MOEAs, Pareto ranking and fitness sharing are the most common techniques adopted by MOEAs. There are some paradigms in this generation, for

* Corresponding author at: School of Information Science and Engineering, Central South University, Changsha 410083, PR China.
E-mail addresses: ywang@csu.edu.cn (Y. Wang), xiangjiancsu@gmail.com (J. Xiang).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.06.008

example: nondominated sorting genetic algorithm (NSGA), proposed by Srinivas and Deb [4], is based on several layers of classifications of the individuals as suggested by Goldberg [5] and uses crowding distance to maintain the diversity of the population. Niched-Pareto genetic algorithm (NPGA), proposed by Horn et al. [6], employs tournament selection based on Pareto dominance and fitness sharing to keep the diversity. Fonseca and Fleming [7] introduced a multiobjective genetic algorithm (MOGA).

The second generation of MOEAs is characterized by the elitism preservation, which usually stores the nondominated individuals into a predefined archive (also called external population). It is necessary to note that incorporating the elitism into MOEAs can facilitate the convergence of the population. Zitzler and Thiele [8] proposed strength Pareto EA (SPEA), which uses an archive to store the nondominated solutions found so far and adopts clustering to prune the archive if the number of nondominated individuals in the archive exceeds a predefined value. Zitzler et al. [9] also proposed an improved version of SPEA, referred as SPEA2. Compared with SPEA, SPEA2 has the following three properties: (1) a new fitness assignment strategy, (2) a density estimation technique, and (3) a novel archive truncation method. Knowles and Corne [10] presented Pareto archive evolutionary strategy (PAES), which uses (1 + 1)-ES to generate offspring. In PAES, the offspring is compared with the parent and the previously archived nondominated individuals for survival. Moreover, PAES divides the objective space into grids, the aim of which is to maintain the diversity of the population. Inspired by PAES, Corne et al. further developed PESA [11] and PESA-II [12]. Deb et al. [13] proposed an improved version of NSGA, called NSGA-II, by incorporating a fast nondominated sorting approach and a crowding-comparison approach.

In the current research, which belongs to the third generation of MOEAs, some new dominance concepts other than traditional Pareto dominance have been introduced. For instance, Laumanns et al. [14] introduced ε-dominance. Hernández-Díaz et al. [15] proposed an adaptive ε-dominance, which is an improvement of the original ε-dominance [14]. Ben Said et al. [16] proposed r-dominance for interactive evolutionary multi-criteria decision making. Brockoff and Zitzler [17] proposed a local dominance scheme to reduce objective dimensionality. In addition, some researchers combined traditional weight vector based techniques with EAs to deal with MOPs [18–21]. Recently, Zhang and Li [22] proposed a novel MOEA based on decomposition, called MOEA/D, which converts MOPs into a set of scalar optimization subproblems. Moreover, MOEA/D utilizes the neighbor information to produce offspring and optimize the subproblems simultaneously.

Many attempts have also been made to improve the performance of MOEAs by making use of different kinds of EAs as well as swarm intelligence. For example, Coello Coello et al. [23] incorporated Pareto dominance into particle swarm optimization for solving MOPs. Li and Zhang [24] proposed a new version of MOEA/D [22] based on differential evolution. Igel et al. [25] developed a variant of covariance matrix adaptation evolution strategy (CMA-ES) [26] for multiobjective optimization. Ghoseiria and Nadjari [27] presented an algorithm based on multiobjective ant colony optimization to solve the bi-objective shortest path problem. Jamuna and Swarup [28] proposed a multiobjective biogeography based optimization algorithm to design optimal placement of phasor measurement units. Zhang [29] proposed an immune optimization algorithm for dealing with constrained nonlinear multiobjective optimization problems.

Recently, indicator-based MOEAs have also been actively researched in the community of evolutionary multiobjective optimization [30,31].

It can be induced from the Karush–Kuhn–Tucker condition that the *PS* of a continuous MOP is a $(m-1)$-dimensional piecewise continuous manifold in the decision space [32,33], where m is the number of objectives. Thus, for the continuous biobjective optimization problems (i.e., $m = 2$), the *PS* is a piecewise continuous curve; and for the continuous triobjective optimization problems (i.e., $m = 3$), the *PS* is a piecewise continuous 2-D surface.

Based on the above regularity, Zhang et al. [34] proposed a regularity model-based multiobjective estimation of distribution algorithm, referred as RM-MEDA. As a kind of estimation of distribution algorithms (EDAs) [35], RM-MEDA employs the $(m-1)$-dimensional local principal component analysis ($(m-1)$-D local PCA) [36] to build the model of the *PS* in the decision space. The $(m-1)$-D local PCA is a locally linear approach to nonlinear dimension reduction, which can construct local models, each pertaining to a different disjoint region of the data space. In RM-MEDA, firstly, the $(m-1)$-D local PCA divides the population into K (K is a constant integer) disjoint clusters and computes the central point and principal component of each cluster. Afterward, one model is built based on the corresponding central point and principal component for each cluster. The primary aim of modeling in RM-MEDA is to approximate one of the pieces of the *PS* by making use of the solutions in one cluster. Ideally, if the number of clusters K is equal to the number of the pieces of the *PS*, each piece of the *PS* can be approximated by one cluster. In this case, a precise model may be built and the performance of RM-MEDA may be excellent. However, if the number of clusters K is not equal to the number of the pieces of the *PS*; needless to say, the model is not precise.

Since we have no *priori* knowledge about the number of the pieces of the *PS* for a MOP at hand, it is very difficult to determine a reasonable value for K. Moreover, the setting of K is usually problem-dependent. In particular, based on our experiments, this parameter has a significant effect on the performance of RM-MEDA. Since K is fixed to 5 in RM-MEDA, this setting might not be very effective for different kinds of MOPs. In order to overcome the above drawback of RM-MEDA, we design a *r*educing *r*edundant *c*luster *o*perator (RRCO) to enhance the modeling precision of RM-MDEA. By integrating RRCO with RM-MEDA, IRM-MEDA is derived. Extensive experiments have been conducted to compare IRM-MEDA with its predecessor RM-MEDA on a set of biobjective and triobjective test instances with variable linkages (note that variable linkages reflect the interactions among the variables). The experimental results verify that the efficiency and effectiveness of RM-MEDA can be significantly improved by RRCO.

The rest of the paper is organized as follows. Section 2 briefly reviews RM-MEDA. The drawback of modeling in RM-MEDA is discussed in Section 3. Section 4 presents the details of RRCO. IRM-MEDA is described in Section 5. The experimental results are reported in Section 6. Finally, Section 7 concludes this paper.

2. Review of RM-MEDA

2.1. Framework

During the evolution, RM-MEDA maintains:

- a population P_t of N individuals: $P_t = \{\vec{x}_1, \ldots, \vec{x}_N\}$, where t is the generation number;
- their $\vec{f}$-values : $\vec{f}(\vec{x}_1), \ldots, \vec{f}(\vec{x}_N)$.

RM-MEDA is implemented as follows:

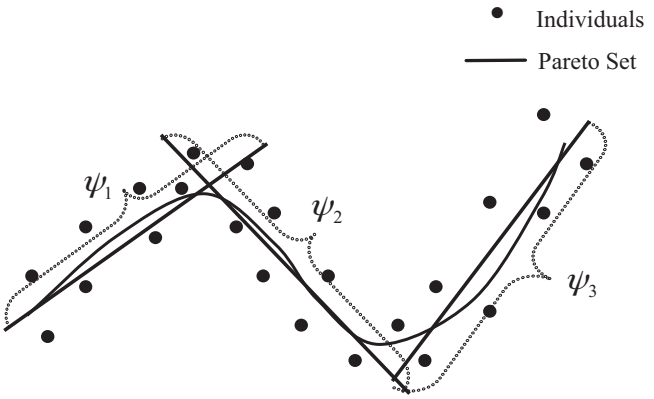

Fig. 1. Illustration of the manifolds of the *PS* for a biobjective optimization problem. The manifolds ψ_1, ψ_2, and ψ_3 are used to approximate the *PS*, $\vec{\zeta}$ is uniformly randomly sampled from $\psi_i (i = 1, \ldots, 3)$, $\vec{\varepsilon}$ is an n-dimensional zero-mean noise vector over $\psi_i (i = 1, \ldots, 3)$, and $\bar{x}_i$ is the center of the ith cluster which corresponds to the ith manifold ψ_i.

Step 1	**Initialization**: Generate an initial population P_0 by randomly sampling N individuals from the decision space S and compute the $\vec{f}$-values of these individuals.
Step 2	**Modeling**: According to the population P_t, build the probability model by the $(m-1)$-D local PCA.
Step 3	**Sampling**: Generate an offspring population Q_t by sampling from the probability model established in Step 2 and compute the $\vec{f}$-value of each individual in Q_t.
Step 4	**Selection**: Select N individuals from P_t and Q_t to construct the population P_{t+1} for the next generation.
Step 5	**Stopping criterion**: If the stopping criterion is satisfied, stop and output the $\vec{f}$-values of the nondominated individuals in the final population, otherwise set $t = t + 1$ and go to Step 2.

It is necessary to note that the selection process in Step 4 is a variant of the nondominated sorting-based selection proposed by Deb et al. [13]. The only difference is that when the number of the selected individuals is larger than N, RM-MEDA removes the redundant individuals with the smallest crowding distances one by one and recalculates the crowding distance after removing one individual. From the above framework, it is obvious that modeling is one of the most important steps of RM-MEDA. The details of the modeling will be explained next.

2.2. Modeling

RM-MEDA exploits the distribution information of the current population in the decision space to build the model. Thereafter, new solutions will be obtained by sampling from the model. Hopefully, the individuals in the population will be uniformly scattered around the *PS* in the decision space as search goes on. RM-MEDA envisages the individuals in the population as independent observations of a random vector $\vec{\xi}$ which can be formulated as follows:

$$\vec{\xi} = \vec{\zeta} + \vec{\varepsilon} \tag{4}$$

where $\vec{\zeta}$ is uniformly distributed over a piecewise continuous $(m-1)$-dimensional manifold, and $\vec{\varepsilon}$ is an n-dimensional zero-mean noise vector.

One of the aims of modeling is to find the principal curve or surface of the population. For the sake of simplicity, suppose that the centroid of $\vec{\xi}$ consists of K manifolds: $\psi_1, \ldots, \psi_K$, each of which is a $(m-1)$-dimensional hyper-rectangle. That is to say, $\vec{\zeta}$ is uniformly distributed over these manifolds. An illustration is given in Fig. 1. Suppose that $\vec{\zeta}$ is uniformly randomly sampled from ψ_i $(i = 1, \ldots, K)$ and $\vec{\varepsilon} \sim N(0, \sigma_i I)$ where I is the $n \times n$ identity matrix and $\sigma_i > 0$ $(i = 1, \ldots, K)$. By doing this, the task of modeling is transformed to estimate ψ_i and σ_i.

RM-MEDA firstly splits the population P_t into K subpopulations $C_1, \ldots, C_K$. Then each subpopulation C_i is utilized to estimate ψ_i and σ_i $(i \in \{1, ..., K\})$. RM-MEDA uses the following procedure to achieve the above purpose:

Step 1	Randomly initialize L_i^{m-1}, $i = 1, \ldots, K$, to be an affine $(m-1)$-dimensional principal subspace containing an individual randomly chosen from the population P_t.
Step 2	Partition the individuals of P_t into K clusters $C_1, \ldots, C_K$:

$$C_i = \{\vec{x} | \vec{x} \in P_t, \text{and } dist(\vec{x}, L_i^{m-1}) \leq dist(\vec{x}, L_k^{m-1}) \text{ for all } k \neq i\} \tag{5}$$

where $dist(x, L_i^{m-1})$ means the Euclidean distance between $\vec{x}$ and its projection in L_i^{m-1}.

Step 3	Update L_i^{m-1}, $i = 1, ..., K$.
Step 4	Repeat Steps 2 and 3 until no change in partition is made.

In the above procedure, L_i^{m-1} is the affine $(m-1)$-dimensional principal subspace of the individuals in C_i. In Step 3, L_i^{m-1} can be updated by the mean and the covariance matrix of the individuals in C_i. The mean of C_i is:

$$\bar{x}_i = \frac{1}{|C_i|} \sum_{\vec{x} \in C_i} \vec{x} \tag{6}$$

and the covariance matrix is:

$$Cov = \frac{1}{|C_i| - 1} \sum_{\vec{x} \in C_i} (\vec{x} - \bar{x}_i)(\vec{x} - \bar{x}_i)^T \tag{7}$$

In addition, the jth principal component $\vec{U}_i^j$ is a unity eigenvector associated with the jth largest eigenvalue of the covariance matrix Cov. Then L_i^{m-1} is updated as follows:

$$\{\vec{x} \in R^n | \vec{x} = \bar{x}_i + \sum_{j=1}^{m-1} \alpha_j \vec{U}_i^j, \alpha_j \in R^n, \quad j = 1, \ldots, m-1\} \tag{8}$$

Next, RM-MEDA uses the above clustering results to build the model. RM-MEDA calculates the range of the projections of the individuals of each cluster C_i in the first $(m-1)$ principal components:

$$l_i^j = \min_{\vec{x} \in C_i} \{(\vec{x} - \bar{x}_i)^T \vec{U}_i^j\} \tag{9}$$

and

$$u_i^j = \max_{\vec{x} \in C_i} \{(\vec{x} - \bar{x}_i)^T \vec{U}_i^j\} \tag{10}$$

where $\bar{x}_i$ is the mean, $\vec{U}_i^j$ is the jth principal component of the covariance matrix Cov of C_i, and $j \in \{1, \ldots, m-1\}$.

Then, each $(m-1)$-D manifold ψ_i is constructed as follows:

$$\psi_i = \{\vec{x} \in R^n | \vec{x} = \bar{x}_i + \sum_{j=1}^{m-1} \beta_j \vec{U}_i^j, l_i^j - 0.25(u_i^j - l_i^j) \leq \beta_j \leq u_i^j + 0.25(u_i^j - l_i^j), \quad j = 1, \ldots, m-1\} \tag{11}$$

It is necessary to note that in order to make a better approximation of the *PS*, RM-MEDA enlarges the range of the projections in each direction $\vec{U}_i^j (j = 1, \ldots, m-1)$ by 50%.

In addition, σ_i is set as follows:

$$\sigma_i = \frac{1}{n - m + 1} \sum_{j=m}^{n} \lambda_i^j \tag{12}$$

where λ_i^j is the jth largest eigenvalue of the covariance matrix Cov of C_i.

Remark 1. As pointed out previously, RM-MEDA is a kind of estimation of distribution algorithms, which exploits the distribution

information of the population to generate the next population. CMA-ES, proposed by Hansen and Ostermeier [26], also incorporates the distribution information of the population into evolution strategy (ES). CMA-ES adopts the following formulation to generate new solutions:

$$\vec{x}_{i,t+1} = \vec{m}_t + \vec{\sigma}_t \times N(\mathbf{0}, \mathbf{C}_t), \quad i = 1, \ldots, \lambda \tag{13}$$

where t is the generation number, $\vec{x}_{i,t+1}$ is the ith offspring, $N(\mathbf{0}, \mathbf{C}_t)$ is a multivariate normal distribution with zero mean and covariance matrix $\mathbf{C}_t$, $\vec{m}_t$ is the mean value of the search distribution, and $\vec{\sigma}_t$ is the overall standard deviation (i.e., the step-size).

Clearly, both Eqs. (4) and (13) contain two terms. However, there are some significant differences between RM-MEDA and CMA-ES. Firstly, in Eq. (4) $\vec{\zeta}$ is uniformly distributed over some manifolds, while in Eq. (13) $\vec{m}_t$ is the mean value of the μ best individuals at generation t. Secondly, when generating the next population, RM-MEDA mainly uses the eigenvectors and eigenvalues of the covariance matrix of the last population, nevertheless, in CME-ES the cumulative evolution path is utilized to update the covariance matrix $\mathbf{C}_t$ and the standard deviation $\vec{\sigma}_t$.

3. The drawback of modeling in RM-MEDA

According to the introduction in Section 2, we can see that modeling is a very important process in RM-MEDA. Moreover, more promising solutions may be generated by sampling from a more precise model. Since the *PS* of the continuous MOPs is a piecewise continuous $(m-1)$-dimensional manifold, RM-MEDA firstly partitions the population P_t into K clusters $C_1, \ldots, C_K$ by making use of the $(m-1)$-D local PCA, with the aim of estimating one manifold ψ_i with one cluster C_i. However, a question which naturally arises is: how to determine the value of K? In RM-MEDA, K is set to a constant integer (i.e., 5). However, needless to say, different shapes of the *PS* may require different number of clusters, and thus, require different values of K.

Suppose that the *PS* of a MOP is shown in Fig. 1. Clearly, in this case if K is set to 3, the ψ_i obtained by modeling can approximate one of the pieces of the *PS* effectively. However, if K is not equal to 3, what will be the corresponding result then? Next, we will discuss the influence of the value of K on the performance.

As shown in Fig. 2, the *PS* is a piecewise continuous 1-D sine curve:

$$x_i = \sin(2\pi x_1), \quad i = 2, 3, \ldots, n, \quad 0 \le x_1 \le 1 \tag{14}$$

where n is the dimension of the decision space. For simplicity, we assume that n is equal to 2. In Fig. 2, the dots denote the individuals which uniformly scatter around the *PS*.

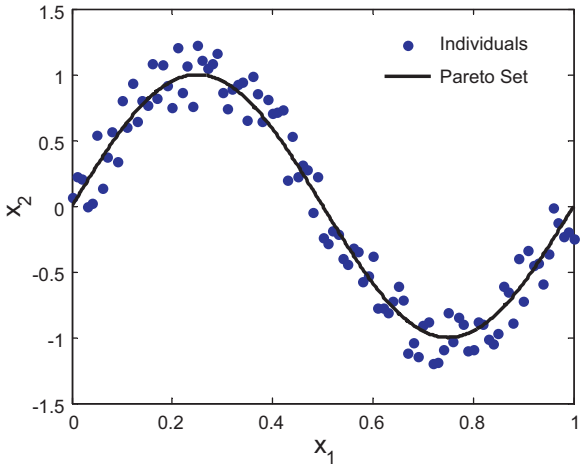

Fig. 2. The *PS* is a piecewise continuous 1-D sine curve, and some individuals are uniformly distributed around the *PS*.

Clearly, in this case, three manifolds ψ_i (i = 1, 2, and 3) can approximate the sine curve effectively. That is, 3 is a suitable value for the number of clusters K. Fig. 3(a) shows the result of clustering by the $(m-1)$-D local PCA with K = 3 and Fig. 3(b) exhibits the ψ_i (i = 1, 2, and 3) computed by Eq. (11). We can see from Fig. 3 that the *PS* can be approximated by ψ_i (i = 1, 2, and 3) very well. However, it should be emphasized that the clustering with the $(m-1)$-D local PCA has some randomness and, therefore, we cannot get the same clustering result if the clustering is applied repeatedly. It is because the $(m-1)$-D local PCA begins with K randomly selected initial points (see Step 1 in Section 2.2) and the clustering result depends on the initial points to a certain degree. When K is set to 3 in our experiments, Fig. 3 shows the typical clustering result.

When K is set to 2 which is smaller than actually required, two typical clustering results are shown in Figs. 4 and 5. From Figs. 4 and 5, we can see that the estimated manifolds ψ_i (i = 1 and 2) cannot approximate the *PS* very well. In particular, as shown in Fig. 5, few better solutions could be sampled from the model built for each cluster. Hence, in this case the performance of the algorithm will significantly degrade. More importantly, maybe the algorithm could not convergence to the *PS* because of the incorrect approximation.

Remark 2. If the number of clusters is smaller than required, the model built will be incorrect, and thus, results in a side influence on the performance of the algorithm.

On the other hand, how about the number of clusters larger than required? Figs. 6–8 show three typical clustering results with K = 5.

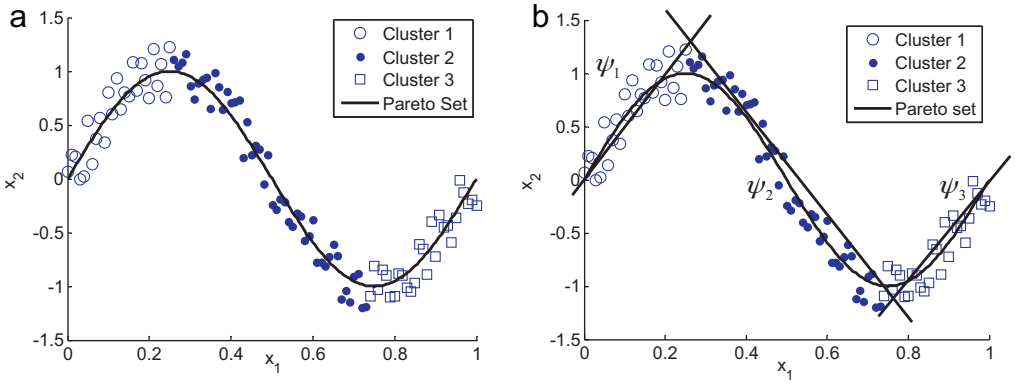

Fig. 3. The results of clustering and modeling with K = 3. (a) The result of clustering. (b) The manifolds ψ_i (i = 1, 2 and 3) built from the clusters.

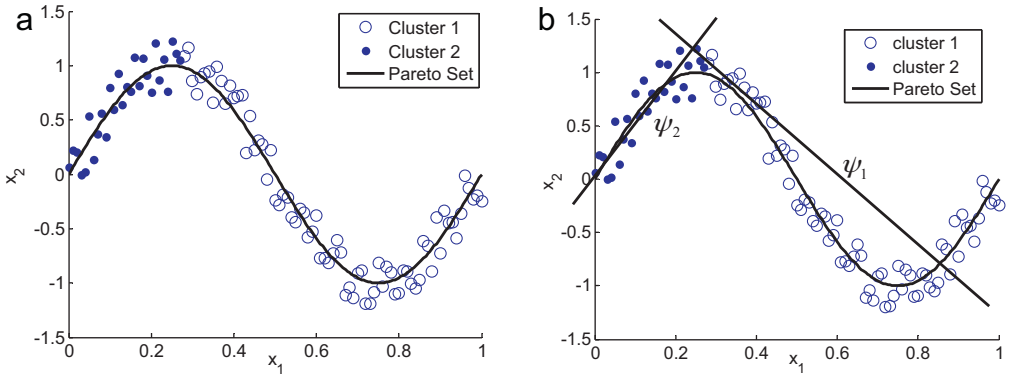

Fig. 4. The results of clustering and modeling with $K=2$. (a) The result of clustering. (b) The manifolds ψ_i ($i=1$ and 2) built from the clusters.

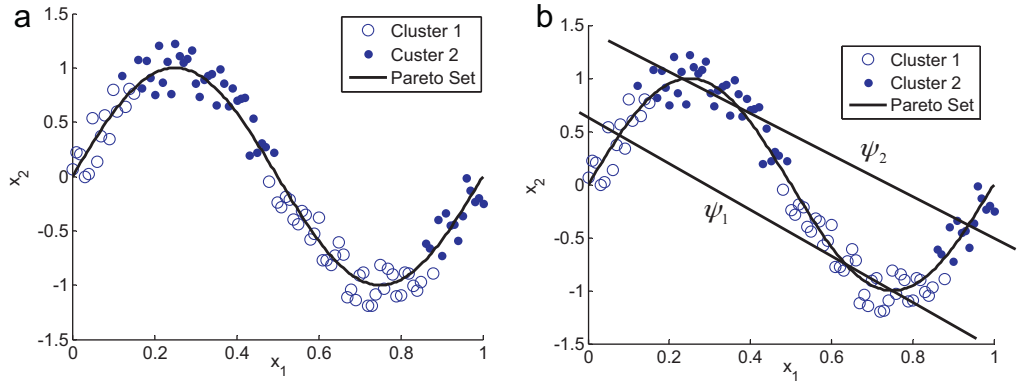

Fig. 5. The results of clustering and modeling with $K=2$. (a) The result of clustering. (b) The manifolds ψ_i ($i=1$ and 2) built from the clusters.

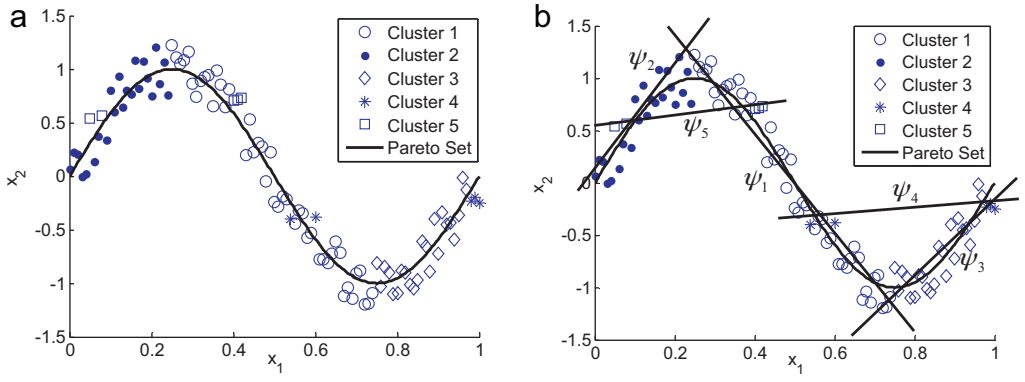

Fig. 6. The results of clustering and modeling with $K=5$. (a) The results of clustering. (b) The models $\psi_i (i=1,\ldots,5)$ built from the clusters.

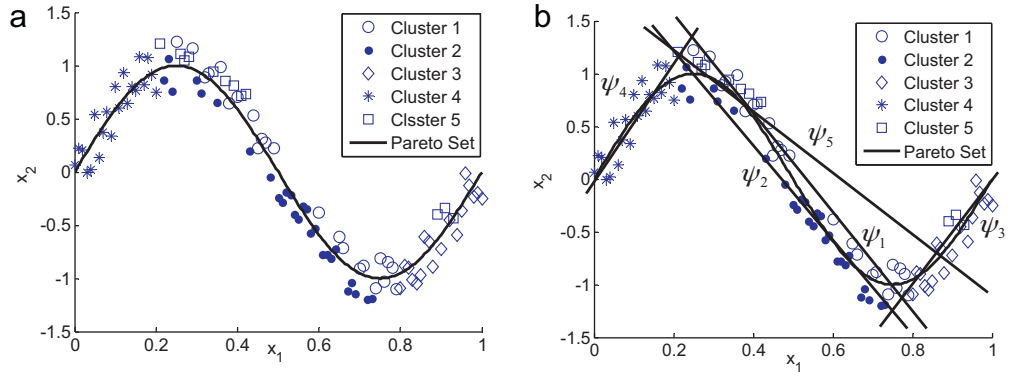

Fig. 7. The results of clustering and modeling with $K=5$. (a) The results of clustering. (b) The models $\psi_i (i=1,\ldots,5)$ built from the clusters.

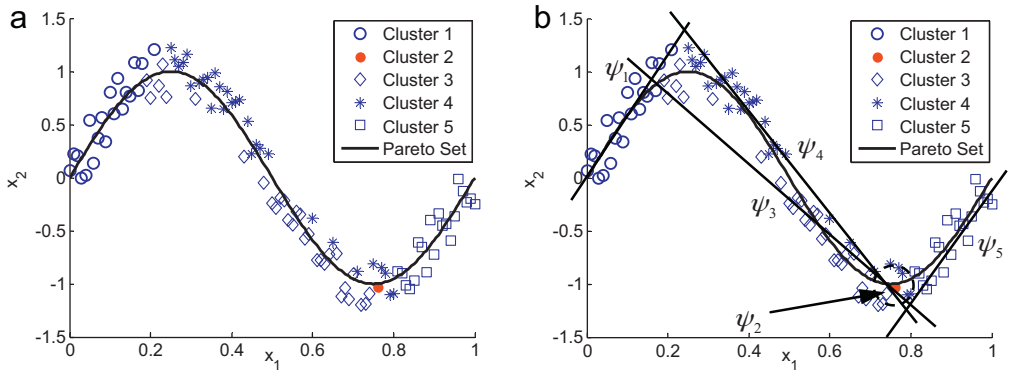

Fig. 8. The results of clustering and modeling with $K = 5$. (a) The results of clustering. (b) The models $\psi_i (i = 1, \ldots, 5)$ built from the clusters.

As shown in these figures, among all the manifolds, three manifolds can be used to estimate the distribution of the *PS* while there are two redundant manifolds. For example, ψ_4 and ψ_5 are two redundant manifolds in Fig. 6. Usually, the points sampled from ψ_4 and ψ_5 are inferior solutions. It is very difficult for these inferior solutions to survive into the next generation. Moreover, the more the inferior individual solutions are generated at each generation, the more the number of function evaluations might be wasted. In addition, from Fig. 7 it is clear that ψ_1, ψ_2, and ψ_5 approximate the same part of the *PS* and, consequently, they can be reduced to just one. As shown in Fig. 8, the second cluster includes only one point. Indeed, it is not an effective cluster and the occurrence of this phenomenon is due to the inappropriate clustering number.

Remark 3. If the number of clusters is larger than required, although the *PS* can be approximated by the model built, there exist some redundant and incorrect manifolds, which have a side effect on the convergence speed and the convergence quality of the algorithm.

Conclusion: According to the above analysis, we can conclude that if the number of clusters used in RM-MEDA is inappropriate, the performance of RM-MEDA will degrade to a certain degree. Particularly, when the number of clusters is smaller than required, the model is always built incorrectly. In this case, RM-MEDA could not converge to the true *PS*. On the other hand, if the number of clusters is larger than required, some redundant manifolds will be generated, which results in the deterioration of the convergence speed and the convergence quality. In general, under the condition that the number of clusters is inappropriate, the performance of RM-MEDA with the number of clusters larger than required is superior to that of RM-MEDA with the number of clusters smaller than required. Maybe this is the reason why in RM-MEDA the number of clusters is set to a slightly big value (i.e., 5), which is larger than required for all the test instances in [34].

4. Reducing redundant cluster operator

Although RM-MEDA usually sets the number of clusters with a big value, as analyzed previously, a big value will also have a side effect on the performance. Moreover, the setting of the number of clusters is usually problem-dependent. In order to overcome the above drawback of RM-MEDA, we propose a *r*educing *r*edundant *c*luster *o*perator (RRCO) in this paper. In the following, the details of RRCO are introduced.

According to the discussion in Section 3, if there exist some redundant clusters, two particular phenomena may occur: (1) the cluster has only one point (as shown in Fig. 8); and (2) some overlapped manifolds approximate the same part of the *PS*, see for example, ψ_1, ψ_2, and ψ_5 in Fig. 7, and ψ_3 and ψ_4 in Fig. 8.

RRCO is designed to address the above two particular phenomena. After implementing the local PCA, we obtain K clusters $C_1, \ldots, C_K$ and K manifolds $\psi_1, \ldots, \psi_K$. Then, RRCO works as follows:

Step 1 Set $A = \{\psi_i, \ldots, \psi_K\}$;
Step 2 **For** $i = 1{:}K$
Step 3 If C_i only contains one individual, then $A = A \backslash \psi_i$;
Step 4 **End For**
Step 5 $L = 0$;
Step 6 **While** $|A| > 0$
Step 7 Choose the first manifold from A, denoted as ψ' and $A = A \backslash \psi'$;
Step 8 **For** $i = 1{:}|A|$
Step 9 If both condition 1 and condition 2 (these two conditions will be explained later) are satisfied for the ith manifold in A and ψ', store this manifold to B;
Step 10 **End For**
Step 11 $A = A \backslash B$;
Step 12 $L = L + 1$;
Step 13 **End While**
Step 14 $K = L$.

During the evolution, we adjust K by RRCO at each generation and use the updated K as the new number of clusters for the next generation.

In RRCO, we use the included angle of two manifolds to identify the overlapped manifolds. When a MOP has m objectives, the included angle is computed between $(m-1)$-D hyper-rectangle. Thus, the included angle is computed between two lines for a biobjective optimization problem, and the included angle is computed between two planes for a triobjective optimization problem. In general, if the included angle of two manifolds is less than a predefined value θ, we consider that these two manifolds overlap with each other. Note, however, that the included angle between parallel manifolds is also very small. As shown in Fig. 9 (this figure is obtained from Fig. 8(b) by eliminating the sine curve, cluster 1,

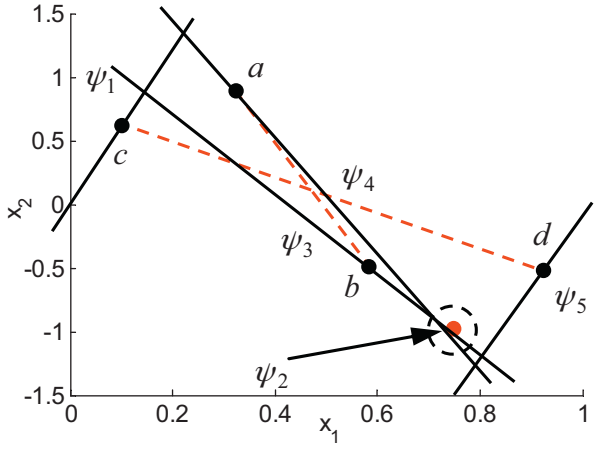

Fig. 9. Illustration of the overlapped manifolds and the parallel manifolds.

cluster 3, cluster 4, and cluster 5), the included angle between the manifolds ψ_3 and ψ_4 which overlap with each other is small, while the included angle between the manifolds ψ_1 and ψ_5 which are parallel with each other is also very small. Therefore, if we only use the included angle of two manifolds to judge whether these two manifolds overlap with each other, some errors may arise.

Next, Fig. 9 is employed as an example to explain how to identify the overlapped manifolds and the parallel manifolds. In Fig. 9, points a, b, c, and d are the central points of the fourth, third, first, and fifth clusters, respectively. The included angle between line segment $\overline{ab}$ and the manifold ψ_3 or the manifold ψ_4 is relatively small, so we consider that ψ_3 and ψ_4 are overlapped manifolds. However, the included angle between line segment $\overline{cd}$ and the manifold ψ_1 or the manifold ψ_5 is relatively large, so we can conclude that ψ_1 and ψ_5 are parallel manifolds. Based on the above analysis, we use the following conditions to distinguish the overlapped manifolds and the parallel manifolds:

Condition 1: $\langle \psi_i, \psi_j \rangle < \theta$

Condition 2: $\min(\langle \overline{CP_{ij}}, \psi_i \rangle, \langle \overline{CP_{ij}}, \psi_j \rangle) < \langle \psi_i, \psi_j \rangle$, where $\overline{CP_{ij}}$ denotes the line segment between CP_i and CP_j, and CP_i and CP_j are the central points of the clusters C_i and C_j, respectively.

In the above two conditions, $\langle \psi_i, \psi_j \rangle$ denotes the included angle between ψ_i and ψ_j, $\langle \overline{CP_{ij}}, \psi_i \rangle$ denotes the included angle between $\overline{CP_{ij}}$ and ψ_i, and $\langle \overline{CP_{ij}}, \psi_j \rangle$ denotes the included angle between $\overline{CP_{ij}}$ and ψ_j. In this paper, θ is a threshold value and is set to $(3/180)\pi$. If both condition 1 and condition 2 are satisfied, the manifolds ψ_i and ψ_j are considered overlapped, while if condition 1 is satisfied and condition 2 is unsatisfied, the manifolds ψ_i and ψ_j are considered parallel. The method to compute the included angle between arbitrary dimension hyper-planes in the Euclidean space is given in Appendix A.

5. IRM-MEDA

By combining RRCO with RM-MEDA, we present an improved version of RM-MEDA, referred as IRM-MEDA. The framework of IRM-MEDA is the same as that of RM-MEDA except that IRM-MEDA uses RRCO to update K. In RM-MEDA, K is fixed during the evolution, whereas IRM-MEDA dynamically adjusts K by extracting some information from the result of clustering at each generation.

IRM-MEDA performs as follows:

Step 1 **Initialization**: Randomly generate an initial population $P_0 = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_N\}$ and initialize K (i.e., the number of clusters).

Step 2 **Modeling**: Use the local PCA to build the model.

Step 3 Modify the value of K by RRCO.

Step 4 **Sampling**: Generate the offspring population Q_t by sampling from the model built.

Step 5 **Selection**: Select N individuals from P_t and Q_t to construct the next population P_{t+1}.

Step 6 **Stopping criterion**: If the stopping criterion is satisfied, stop and output the $\vec{f}$-values of the nondominated individuals of the final population, otherwise go to Step 2.

The procedures of initialization, modeling, sampling, and selection in IRM-MEDA are exactly the same as in RM-MEDA. It is necessary to note that during the sampling, firstly the volume of each manifold ψ_i is computed, and then the probability that a new solution is generated around ψ_i is proportional to the volume of ψ_i. When generating a new solution, the parameter β_j in Eq. (11) is randomly chosen between $l_i^j - 0.25(u_i^j - l_i^j)$ and $u_i^j + 0.25(u_i^j - l_i^j)$. The interested reader can refer to [34] for details.

6. Experimental study

6.1. Test instances

In this paper, we use ten test instances (F1–F10) to verify the effectiveness of IRM-MEDA. The main information of these ten test instances is summarized in Table 1. Test instances F1–F6 are taken from [34]. Since the primary purpose of RM-MEDA is to approximate the *PS* of MOPs by taking advantage of the $(m-1)$-D local PCA, we construct four additional test instances (i.e., F7–F10) with various *PS* structures to further compare RM-MEDA with IRM-MEDA. It is necessary to note that the *PS* structures of F7–F10 are more complex than that of the other six test instances and, therefore, F7–F10 pose a great challenge for both RM-MEDA and IRM-MEDA. The *PS* of these ten test instances has been introduced in Table 1.

In terms of the way of variable linkages, these test instances can be divided into two categories: test instances with linear variable linkages (i.e., F1–F3) and test instances with nonlinear variable linkages (i.e., F4–F10).

Remark 4. Test instances F1, F4, F7, and F8 are variants of ZDT1 [37] and test instances F2 and F5 are variants of ZDT2 [37]. According to the Theorem 1 in [38], F1, F2, F4, F5, F7, and F8 in this paper have the *PS* $(x_1, x_2^*, x_3^*, \ldots, x_n^*)$, where $x_2^*, x_3^*, \ldots, x_n^*$ are the solutions of $g(\vec{x})$, respectively, and x_1 can take any value in [0,1]. Test instances F3 and F6 are variants of DTLZ2 [39]. According to the explanation in [39], F3 and F6 in this paper have the *PS* $(x_1, x_2, x_3^*, \ldots, x_n^*)$, where $x_3^*, \ldots, x_n^*$ are the solutions of $g(\vec{x})$, respectively, and x_1 and x_2 can take any value in [0,1]. Test instances F9 and F10 are variants of test instances F2 and F9 in [24], respectively. According to the Theorem 1 on F2 and F9 in [24], F9 in this paper has the *PS* $(x_1, x_2^*, x_3^*, \ldots, x_n^*)$, where $x_2^*, x_3^*, \ldots, x_n^*$ are equal to $\sin(2\pi x_1)$, respectively, and x_1 can take any value in [0,1], and F10 in this paper has the *PS* $(x_1, x_2^*, x_3^*, \ldots, x_n^*)$, where $x_2^*, x_3^*, \ldots, x_n^*$ are equal to $\cos(2\pi x_1)$, respectively, and x_1 can take any value in [0,1].

6.2. Performance indicators

In this paper, two performance indicators are adopted to compare RM-MEDA and IRM-MEDA.

The first performance indicator measures the convergence speed of an algorithm. When computing the convergence speed of an algorithm, we need to design a stopping criterion. Once the stopping criterion is satisfied, the algorithm is considered convergent and the procedure should halt. Since the purpose of the solution of a MOP is to find the *PS*, the image of which in the objective space is called *PF*, the stopping criterion is based on the reasonable approximation of true *PF*. Let P^* be a set of points uniformly distributed on the true *PF*, and P a set of points which are the image of the nondominated individuals of the population in the objective space. In this paper, firstly the hyper-volume (HV) values [8,40] are calculated for P^* and P, respectively. Afterward, we follow the approach introduced by Nebro et al. [41] to determinate the stopping criterion. In [41], once the HV value of P attains or surpasses the 98% of the HV value of P^*, i.e.,

$$\frac{HV(P)}{HV(P^*)} \geq 98\% \tag{15}$$

we consider that a reasonable approximation the true *PF* has been obtained. Under this condition, the algorithm terminates and the number of function evaluations (FES) is recorded. In Eq. (15), $HV(P)$ denotes the hyper-volume surrounded by P and a fixed reference point, and $HV(P^*)$ denotes the hyper-volume surrounded by P^* and a fixed reference point. We use the mean and standard derivation of FES among 20 independent runs as the first performance indicator

Table 1
The main information of test instances.

Instance	Variables	Objectives	Characteristics
F1	$[0,1]^n$	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-\sqrt{f_1(\vec{x})/g(\vec{x})}]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n}(x_i-x_1)^2\right)/(n-1)$	Convex *PF* Linear linkage $n=50$ The *PS* is a line segment: $x_n=\cdots=x_2=x_1$, $0\le x_1\le 1$
F2	$[0,1]^n$	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-(f_1(\vec{x})/g(\vec{x}))^2]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n}(x_i-x_1)^2\right)/(n-1)$	Concave *PF* Linear linkage $n=50$ The *PS* is a line segment: $x_n=\cdots=x_2=x_1$, $0\le x_1\le 1$
F3	$[0,1]^n$	$f_1(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\cos\left(\frac{\pi}{2}x_2\right)(1+g(\vec{x}))$ $f_2(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right)(1+g(\vec{x}))$ $f_3(\vec{x}) = \sin\left(\frac{\pi}{2}x_1\right)(1+g(\vec{x}))$ $g(\vec{x}) = \sum_{i=3}^{n}(x_i-x_1)^2$	Concave *PF* Linear linkage 3 objectives $n=30$ The *PS* is a 2-D rectangle: $x_n=\ldots=x_3=x_1$, $0\le x_1, x_2\le 1$
F4	$[0,1]^n$	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-\sqrt{f_1(\vec{x})/g(\vec{x})}]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n}(x_i^2-x_1)^2\right)/(n-1)$	Convex *PF* Nonlinear linkage $n=50$ The *PS* is a bounded continuous conic: $x_n=\cdots=x_2=\sqrt{x_1}$, $0\le x_1\le 1$
F5	$[0,1]^n$	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-(f_1(\vec{x})/g(\vec{x}))^2]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n}(x_i^2-x_1)^2\right)/(n-1)$	Concave *PF* Nonlinear linkage $n=50$ The *PS* is a bounded continuous conic: $x_n=\cdots=x_2=\sqrt{x_1}$, $0\le x_1\le 1$
F6	$[0,1]^n$	$f_1(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\cos\left(\frac{\pi}{2}x_2\right)(1+g(\vec{x}))$ $f_2(\vec{x}) = \cos\left(\frac{\pi}{2}x_1\right)\sin\left(\frac{\pi}{2}x_2\right)(1+g(\vec{x}))$ $f_3(\vec{x}) = \sin\left(\frac{\pi}{2}x_1\right)(1+g(\vec{x}))$ $g(\vec{x}) = \sum_{i=3}^{n}({x_i}^2-x_1)^2$	Concave *PF* Nonlinear linkage 3 objectives $n=30$ The *PS* is a 2-D surface: $x_n=\cdots=x_3=\sqrt{x_1}$, $0\le x_1$, $x_2\le 1$
F7	$[0,1]\times[-1,1]^{n-1}$	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-\sqrt{f_1(\vec{x})/g(\vec{x})}]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n}(x_i-\sin(2\pi x_1))^2\right)/(n-1)$	Convex *PF* Nonlinear linkage $n=30$ The *PS* is a period continuous sine curve: $x_n=\cdots=x_2=\sin(2\pi x_1)$, $0\le x_1\le 1$
F8	$[0,1]\times[-1,1]^{n-1}$	$f_1(\vec{x}) = x_1$ $f_2(\vec{x}) = g(\vec{x})[1-\sqrt{f_1(\vec{x})/g(\vec{x})}]$ $g(\vec{x}) = 1+9\left(\sum_{i=2}^{n}(x_i-\cos(2\pi x_1))^2\right)/(n-1)$	Convex *PF* Nonlinear linkage $n=30$ The *PS* is a period continuous cosine curve: $x_n=\cdots=x_2=\cos(2\pi x_1)$, $0\le x_1\le 1$

Table 1 (*Continued*)

Instance	Variables	Objectives	Characteristics
F9	$[0,1]\times[-1,1]^{n-1}$	$f_1(\vec{x}) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - \sin(2\pi x_1))^2$ $f_2(\vec{x}) = 1 - \sqrt{x_1} + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - \sin(2\pi x_1))^2$ where $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$	Convex *PF* Nonlinear linkage $n = 30$ The *PS* is a period continuous sine curve: $x_n = \cdots = x_2 = \sin(2\pi x_1)$, $0 \le x_1 \le 1$
F10	$[0,1]\times[-1,1]^{n-1}$	$f_1(x) = x_1 + \frac{2}{\lvert J_1\rvert}\sum_{j\in J_1}(x_j - \cos(2\pi x_1))^2$ $f_2(x) = 1 - x_1^2 + \frac{2}{\lvert J_2\rvert}\sum_{j\in J_2}(x_j - \cos(2\pi x_1))^2$ where $J_1 = \{j \mid j \text{ is odd and } 2 \le j \le n\}$ and $J_2 = \{j \mid j \text{ is even and } 2 \le j \le n\}$	Concave *PF* Nonlinear linkage $n = 30$ The *PS* is a period continuous cosine curve: $x_n = \cdots = x_2 = \cos(2\pi x_1)$, $0 \le x_1 \le 1$

to measure the convergence speed of an algorithm on each test instance. In our experiments, the size of P^* is set to 300 and 800 for the biobjective optimization problems and for the triobjective optimization problems, respectively.

The second performance indicator measures the convergence quality of an algorithm. The inverted generational distance (IGD) is employed for this purpose, which is one of the most commonly used performance indicators in MOEAs and can be described as follows [34]:

$$D(P^*, P) = \frac{\sum_{v\in P^*} d(v, P)}{|P^*|} \tag{16}$$

where $d(v, P)$ denotes the minimum Euclidean distance between v and the points in P. IGD indicator computes the average distance from all members in the true *PF* to their nearest points in P. This performance indicator can measure both the diversity and the convergence of the population. $D(P^*, P) = 0$ indicates that all the points in P are on the true *PF* and cover the true *PF* uniformly.

6.3. Experimental results

6.3.1. Convergence speed

Firstly, we compare the convergence speed of IRM-MEDA and RM-MEDA in terms of the first performance indicator. The maximal number of FES is set to 30,000 for F1, F2, F4, F5 and F8, to 50,000 for F7, F9, and F10, and to 80,000 for F3 and F6. For test instances with two objectives, the population size is set to 100, and for test instances with three objectives, the population size is set to 200. 20 independent runs have been performed for each test instance, and the average and standard deviation of the number of FES to satisfy Eq. (15) in each run have been recorded. Table 2 summarizes the experimental results. Wilcoxon's rank sum test at a 0.05 significance level is conducted on the experimental results. In addition, we also use the *acceleration rate* (*AR*) to compare the convergence speed, which is defined as follows:

$$AR = \frac{AVE_{RM-MEDA} - AVE_{IRM-MEDA}}{AVE_{RM-MEDA}} \tag{17}$$

where $AVE_{RM\text{-}MEDA}$ and $AVE_{IRM\text{-}MEDA}$ denote the average number of FES provided by RM-MEDA and IRM-MEDA in Table 2, respectively.

As shown in Table 2, IRM-MEDA performs significantly better than RM-MEDA on all the test instances according to the Wilcoxon's rank sum test. Furthermore, IRM-MEDA saves 20–50% FES compared with RM-MEDA, and the mean *AR* is 31.67%.

The above discussion demonstrates that RRCO has the capability to significantly accelerate the convergence of RM-MEDA.

6.3.2. Convergence quality

As mentioned above, the IGD indicator is applied to measure both the convergence and diversity of the population. For each test instance, the same number of FES has been performed for both IRM-MEDA and RM-MEDA. Based on the results in Table 2, the maximal number of FES for each test instance is set about to the mean number of FES required by IRM-MEDA to achieve the 98% of the HV

Table 2
Experimental results of IRM-MEDA and RM-MEDA over 20 independent runs for ten test instances in terms of the convergence speed. "Mean FES" and "Std Dev" indicate the average and standard deviation of FES obtained in 20 runs, respectively. "*AR*" denotes the acceleration rate. Wilcoxon's rank sum test at a 0.05 significance level is performed between IRM-MEDA and RM-MEDA.

Instance	IRM-MEDA Mean FES ± Std Dev ($\times 10^3$)	RM-MEDA Mean FES ± Std Dev ($\times 10^3$)	*AR*
F1	7.9833 ± 1.0371 +	11.047 ± 1.0743	27.73%
F2	10.103 ± 1.3720 +	17.788 ± 3.5596	43.20%
F3	24.613 ± 2.9569 +	41.367 ± 12.291	40.50%
F4	13.152 ± 2.6990 +	19.589 ± 4.2625	32.86%
F5	18.607 ± 2.9081 +	28.677 ± 3.3935	35.12%
F6	45.007 ± 7.4701 +	62.053 ± 13.359	27.47%
F7	26.297 ± 3.7591 +	34.450 ± 3.3414	23.67%
F8	15.870 ± 2.7343 +	21.117 ± 2.4598	24.85%
F9	27.613 ± 4.3017 +	40.597 ± 6.0242	31.98%
F10	26.093 ± 4.4161 +	36.893 ± 5.5490	29.27%
+	10		
−	0	Mean	31.67%
≈	0		

"+", "−", and "≈" denote the performance of IRM-MEDA is better than, worse than, and similar to that of RM-MEDA, respectively.

Table 3
Experimental results of IRM-MEDA and RM-MEDA over 20 independent runs for ten test instances in terms of the IGD indicator. "Mean IGD" and "Std Dev" indicate the average and standard deviation of the IGD indicator obtained in 20 runs, respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between IRM-MEDA and RM-MEDA.

Instance	The number of FES	IRM-MEDA	RM-MEDA
		Mean IGD ± Std Dev ($\times 10^{-3}$)	Mean IGD ± Std Dev ($\times 10^{-3}$)
F1	10,000	5.8516 ± 1.3174 +	10.790 ± 3.0714
F2	10,000	7.7836 ± 3.4711 +	17.583 ± 10.817
F3	25,000	47.910 ± 2.4211 +	51.048 ± 1.8803
F4	15,000	12.578 ± 6.7505 +	24.630 ± 14.534
F5	20,000	8.0418 ± 1.5723 +	28.764 ± 17.663
F6	45,000	59.412 ± 2.8380 +	66.266 ± 3.4071
F7	30,000	10.698 ± 5.7848 +	24.848 ± 11.772
F8	18,000	8.1101 ± 3.7198 +	33.083 ± 21.231
F9	30,000	12.772 ± 8.7324 +	28.554 ± 16.414
F10	25,000	7.7968 ± 2.7730 +	28.982 ± 18.337
+	10		
−	0		
≈	0		

"+", "−", and "≈" denote the performance of IRM-MEDA is better than, worse than, and similar to that of RM-MEDA, respectively.

value of the true *PF*. We have performed 20 independent runs for IRM-MEDA and RM-MEDA, and obtained the mean and standard deviation of the IGD indicator as shown in Table 3. In order to ensure the results with statistical confidence, Wilcoxon's rank sum test at a 0.05 significance level has also been performed.

Table 3 demonstrates that IRM-MEDA provides evidently lower IGD values than RM-MEDA for all the test instances. Moreover, the statistical test indicates that IRM-MEDA outperforms RM-MEDA on all the test instances. Therefore, we can conclude that RRCO is able to significantly improve the convergence quality of RM-MEDA.

Figs. 10–13 exhibit the typical nondominated fronts and all the nondominated fronts of 20 independent runs derived from IRM-MEDA and RM-MEDA for two representative test instances.

The *PS* of F5 is a bounded continuous conic curve. In this case, the model built by RM-MEDA might not be reasonable due to the redundant clusters as analyzed in Section 3, which has a negative influence on the convergence speed and the convergence quality of the population. As shown in Figs. 10 and 11, the differences of the nondominated fronts provided by IRM-MEDA and RM-MEDA on F5 are distinct.

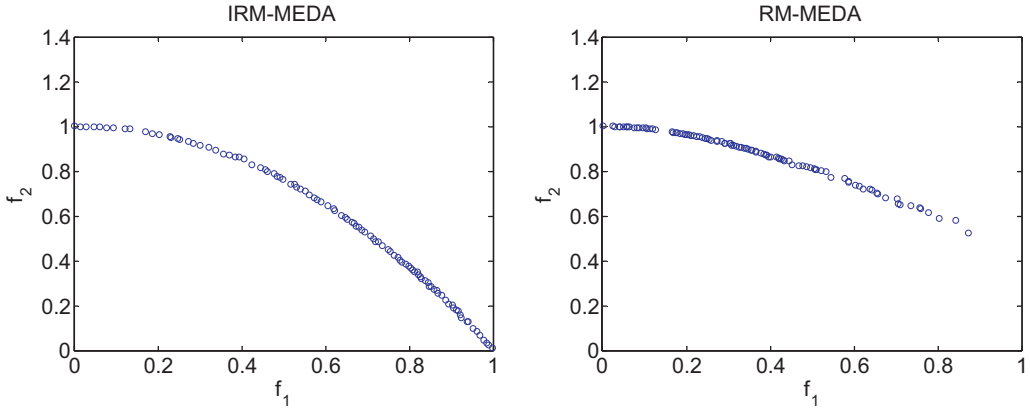

Fig. 10. Plots of the nondominated individuals of the final populations in a typical run obtained by IRM-MEDA and RM-MEDA in the objective space on F5.

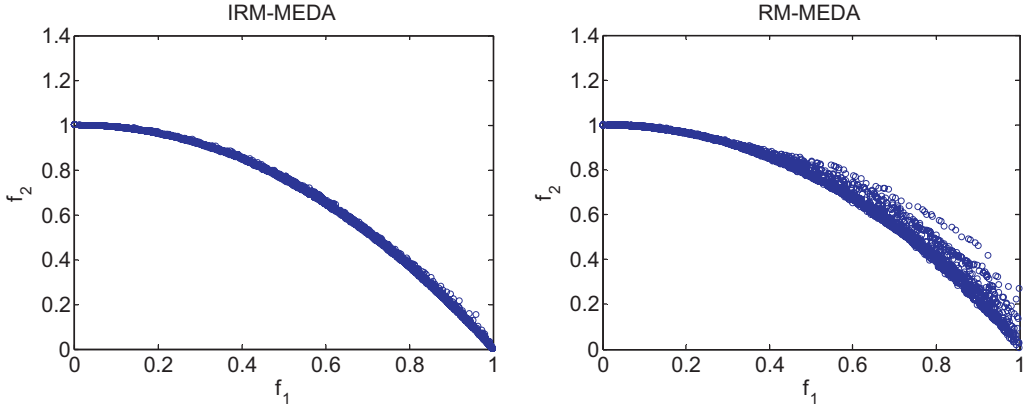

Fig. 11. Plots of the nondominated individuals of all the final populations in 20 runs obtained by IRM-MEDA and RM-MEDA in the objective space on F5.

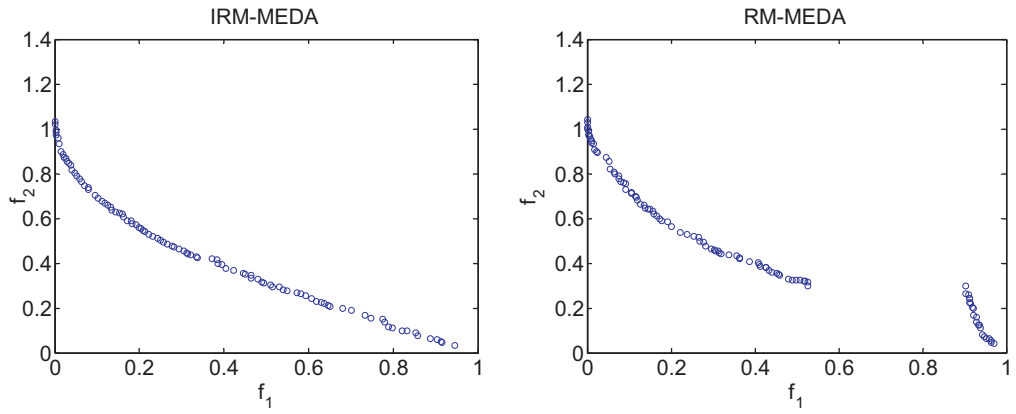

Fig. 12. Plots of the nondominated individuals of the final populations in a typical run obtained by IRM-MEDA and RM-MEDA in the objective space on F7.

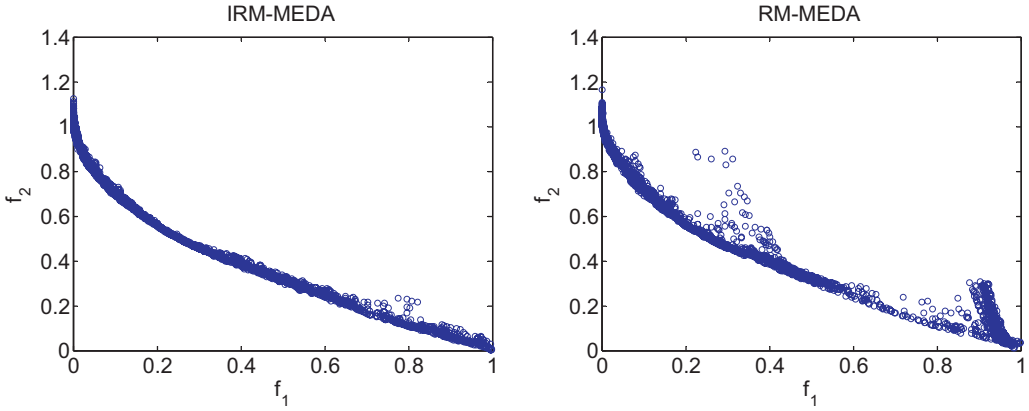

Fig. 13. Plots of the nondominated individuals of all the final populations in 20 runs obtained by IRM-MEDA and RM-MEDA in the objective space on F7.

The *PS* of F7 is a period continuous sine curve. For this kind of *PS*, it is very likely for an algorithm to build a wrong model which greatly deviates from the *PS* of the test instance. Clearly, new trial solutions sampled from the wrong model are often inferior points. These inferior solutions are hard to survive to the next generation and may greatly affect the convergence speed of the population because a lot of FES used to evaluate these inferior solutions will be wasted. Moreover, the redundant clusters will further deteriorate the performance. It can be observed from Figs. 12 and 13 that RM-MEDA cannot converge to the *PF* very well and easily misses some parts of the *PF*.

Based on our observations, when the procedure of IRM-MEDA terminates in each run, the number of clusters (i.e., the parameter K) achieved by IRM-MEDA is 1, 1, 1, 2, 2, 3, 3, 2, 3, and 2 for test instances F1–F10, respectively. The above number of clusters for each test instance is consistent with the actual situation of each test instance.

6.4. Comparison with MOEA/D-DE

In this subsection, the performance of IRM-MEDA is compared with that of another state-of-the-art MOEA, named MOEA/D-DE [24] over the ten test functions F1–F10 in this paper. The two performance indicators introduced in Section 6.2 are used to compare IRM-MEDA and MOEA/D-DE. For IRM-MEDA and MOEA/D-DE, 20 independent runs are implemented on each test instance. We use the same parameter settings for MOEA/D-DE as in the original paper. The experimental results are summarized in Tables 4 and 5. Wilcoxon's rank sum test at a 0.05 significance level is conducted on the experimental results.

Table 4
Experimental results of IRM-MEDA and MOEA/D-DE over 20 independent runs for ten test instances in terms of the convergence speed. "Mean FES" and "Std Dev" indicate the average and standard deviation of FES obtained in 20 runs, respectively. "*AR*" denotes the acceleration rate. Wilcoxon's rank sum test at a 0.05 significance level is performed between IRM-MEDA and MOEA/D-DE.

Instance	IRM-MEDA Mean FES ± Std Dev ($\times 10^3$)	MOEA/D-DE Mean FES ± Std Dev ($\times 10^3$)
F1	7.9833 ± 1.0371 +	16.700 ± 2.2863
F2	10.103 ± 1.3720 +	17.605 ± 2.3234
F3	24.613 ± 2.9569 +	26.437 ± 2.2251
F4	13.152 ± 2.6990 +	20.105 ± 3.7705
F5	18.607 ± 2.9081 +	23.295 ± 2.6045
F6	45.007 ± 7.4701 +	70.030 ± 11.693
F7	26.297 ± 3.7591 −	18.325 ± 4.6936
F8	15.870 ± 2.7343 +	18.745 ± 2.2684
F9	27.613 ± 4.3017 −	22.000 ± 3.5551
F10	26.093 ± 4.4161 +	27.436 ± 4.1715
+	8	
−	2	
≈	0	

"+", "−", and "≈" denote the performance of IRM-MEDA is better than, worse than, and similar to that of MOEA/D-DE, respectively.

From Tables 4 and 5, it is clear that overall IRM-MEDA significantly outperforms MOEA/D-DE. In terms of the convergence speed, IRM-MEDA is faster than MOEA/D-DE on eight test instances, and MOEA/D-DE converges faster than IRM-MEDA only on two test instances. In addition, with respect to the IGD indicator, IRM-MEDA surpasses MOEA/D-DE on seven test functions, and MOEA/D-DE beats IRM-MEDA only on two test functions.

Table 5
Experimental results of IRM-MEDA and MOEA/D-DE over 20 independent runs for ten test instances in terms of the IGD indicator. "Mean IGD" and "Std Dev" indicate the average and standard deviation of the IGD indicator obtained in 20 runs, respectively. Wilcoxon's rank sum test at a 0.05 significance level is performed between IRM-MEDA and MOEA/D-DE.

Instance	The number of FES	IRM-MEDA Mean IGD ± Std Dev ($\times 10^{-3}$)	MOEA/D-DE Mean IGD ± Std Dev($\times 10^{-3}$)
F1	10,000	5.8516 ± 1.3174 +	91.062 ± 32.837
F2	10,000	7.7836 ± 3.4711 +	152.46 ± 48.325
F3	25,000	47.910 ± 2.4211 +	52.073 ± 2.1590
F4	15,000	12.578 ± 6.7505 +	86.311 ± 65.878
F5	20,000	8.0418 ± 1.5723 +	57.084 ± 104.30
F6	45,000	59.412 ± 2.8380 +	85.681 ± 21.473
F7	30,000	10.698 ± 5.7848 −	5.4767 ± 0.4249
F8	18,000	8.1101 ± 3.7198 +	21.963 ± 18.813
F9	30,000	12.772 ± 8.7324 −	6.9172 ± 5.4512
F10	25,000	7.7968 ± 2.7730 ≈	8.8354 ± 10.469
+	7		
−	2		
≈	1		

"+", "−", and "≈" denote the performance of IRM-MEDA is better than, worse than, and similar to that of MOEA/D-DE, respectively.

Table 6
Experimental results of IRM-MEDA with varying θ. "Mean IGD" and "Std Dev" indicate the average and standard deviation of the IGD indicator obtained in 20 runs for ten test instances. The best result for each test instance among the compared algorithms is highlighted in **boldface**.

Instance	$\theta = \frac{1}{180}\pi$ Mean IGD ± Std Dev ($\times 10^{-3}$)	$\theta = \frac{2}{180}\pi$ Mean IGD ± Std Dev ($\times 10^{-3}$)	$\theta = \frac{3}{180}\pi$ Mean IGD ± Std Dev ($\times 10^{-3}$)	$\theta = \frac{5}{180}\pi$ Mean IGD ± Std Dev ($\times 10^{-3}$)	$\theta = \frac{7}{180}\pi$ Mean IGD ± Std Dev ($\times 10^{-3}$)
F1	6.01 ± 1.18	6.01 ± 1.49	5.85 ± 1.32	5.84 ± 1.24	**5.05 ± 2.40**
F2	7.49 ± 3.35	**7.48 ± 3.70**	7.78 ± 3.47	8.06 ± 5.16	8.01 ± 4.68
F3	55.5 ± 1.96	48.5 ± 1.90	47.9 ± 2.42	48.7 ± 2.66	**46.0 ± 12.2**
F4	17.8 ± 8.17	17.4 ± 10.3	**12.6 ± 6.75**	15.5 ± 12.3	14.7 ± 10.4
F5	12.1 ± 15.9	9.06 ± 2.67	**8.04 ± 1.57**	8.49 ± 2.16	10.1 ± 7.72
F6	60.0 ± 3.10	60.4 ± 2.33	**59.4 ± 2.84**	60.5 ± 2.96	64.5 ± 3.72
F7	11.6 ± 5.21	11.4 ± 9.93	**10.7 ± 5.78**	13.7 ± 9.10	18.1 ± 15.7
F8	12.5 ± 11.9	9.98 ± 6.86	**8.11 ± 3.72**	25.2 ± 20.2	35.9 ± 22.2
F9	19.5 ± 24.6	**12.2 ± 9.12**	12.8 ± 8.73	18.7 ± 23.3	14.2 ± 19.7
F10	20.3 ± 30.1	7.83 ± 3.44	**7.80 ± 2.77**	34.8 ± 44.8	33.2 ± 38.5

6.5. Sensitivity to the parameter θ in RRCO

In RRCO, an additional parameter θ is used. To study the sensitivity of the parameter θ, we test five different values: $(1/180)\pi$, $(2/180)\pi$, $(3/180)\pi$, $(5/180)\pi$, and $(7/180)\pi$. For each value of θ, IRM-MEDA performs 20 runs independently for all the test instances and the IGD indicator is used to assess the performance.

Table 6 demonstrates that IRM-MEDA with $\theta = (3/180)\pi$ has the best overall performance. When θ is set to a relatively smaller value (i.e., $(1/180)\pi$ and $(2/180)\pi$), the performance of the algorithm suffers from slight degeneration for F1, F3, F4, F5, F6, F7, F8, and F10, compared with IRM-MEDA with $\theta = (3/180)\pi$. In addition, we also observe that when dealing with F7, F8, F9, and F10 which have more complicated *PS*, the performance degradation of IRM-MEDA with $(5/180)\pi$ occurs, compared with IRM-MEDA with $\theta = (3/180)\pi$. In the case of $\theta = (7/180)\pi$, the algorithm exhibits the worst performance for F6, F7, and F8.

From the above discussion, $\theta = (3/180)\pi$ is recommended for RRCO.

7. Conclusion

RM-MEDA [34] is a recently proposed approach for solving MOPs with variable linkages. In order to approximate the *PS* of MOPs, the $(m-1)$-D local PCA is adopted to build the model of the population in RM-MEDA. In this paper, we analyze the drawback of the clustering process in the modeling suggested by RM-MEDA. Based on our analysis, the number of clusters is problem-dependent and has a significant effect on the performance of RM-MEDA. However, a fixed number of clusters is recommended in [34] for different kinds of problems.

In this paper, we propose an improved version of RM-MEDA, namely IRM-MEDA, by incorporating a *r*educing *r*edundant *c*luster *o*perator (RRCO) to dynamically modify the number of clusters during the evolution. RRCO can adaptively decrease the redundant clusters. The experimental results suggest that the performance of IRM-MEDA is significantly better than that of RM-MEDA in terms of the convergence speed and the convergence quality.

It is worth noting that in this paper we only deal with the number of clusters K larger than required since usually we initialize the number of clusters K with a slightly big value when solving MOPs. How to cope with the case that the number of clusters K is smaller than required is one of our future work. In addition, we will apply IRM-MEDA to solve some real-world MOPs in the future. When applying IRM-MEDA to solve real-world MOPs, firstly we need to elaborate the encoding operator. Moreover, some domain-specific knowledge should be extracted and integrated into IRM-MEDA for increasing efficiency and effectiveness.

The source code of IRM-MEDA is written in MATLAB and can be obtained from the first author upon request.

Acknowledgments

The authors sincerely thank the anonymous reviewers for their constructive and helpful comments and suggestions.

This research was supported in part by the National Natural Science Foundation of China under Grant 60805027, 61175064 and 90820302, and in part by the Research Fund for the Doctoral Program of Higher Education under Grant 200805330005.

Appendix A.

When dealing with MOPs with (m+1) objectives, after implementing the local PCA, the m-dimensional manifolds $\psi_i \in R^{m\times n}$, $i = 1, 2, \ldots, K$ (n is the dimension of decision space) are obtained and satisfy $\psi_i \psi_i{}^T = E$. According to the theorem of high dimension Euclidean geometry [42,43], the included angle θ between ψ_i and ψ_j ($i, j \in \{1, 2, \ldots K\}$ and $i \neq j$) can be computed as follows:

Step 1 Compute the singular values of $\psi_i \psi_j^T$.
Step 2 **If** all the m singular values are equal to 1
Step 3 $\theta = 0$;
Step 4 **Elseif** less than m singular values are equal to 1 and the remaining singular values are equal to 0
Step 5 $\theta = \pi/2$;
Step 6 **Else**
Step 7 $\theta = \arccos(\eta)$, where η is the largest singular value and less than 1.
Step 8 **End If**

References

[1] J.D. Schaffer, Multiple objective optimization with vector evaluated genetic algorithms, in: Proceedings of the First International Conference on Genetic Algorithms, Lawrence Erlbaum, Hillsdale, NJ, 1985, pp. 93–100.
[2] C.A. Coello Coello, Evolutionary multi-objective optimization: a historical view of the field, IEEE Computational Intelligence Magazine 1 (1) (2006) 28–36.
[3] M. Gong, L. Jiao, D. Yang, W. Ma, Research on evolutionary multi-objective optimization algorithms, Journal of Software 20 (2) (2009) 271–289 (in Chinese with English abstract).
[4] N. Srinivas, K. Deb, Multiobjective optimization using nondominated sorting in genetic algorithms, Evolutionary Computation 2 (3) (1994) 221–248.
[5] D.E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley Publishing Company, Reading, MA, 1989.
[6] J. Horn, N. Nafpliotis, D.E. Goldberg, A Niched Pareto genetic algorithm for multiobjective optimization, in: Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, IEEE Service Center, Piscataway, NJ, 1994, pp. 82–87.
[7] C.M. Fonseca, P.J. Fleming, Genetic algorithms for multiobjective optimization: formulation, discussion and generalization, in: Proceedings of the Fifth International Conference on Genetic Algorithms, Morgan Kaufmann Publishers, San Mateo, CA, 1993, pp. 416–423.
[8] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.
[9] E. Zitzler, M. Laumannns, L. Thiele, SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization, in: Proceeding of the EUROGEN 2001—Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problem, 2001, pp. 95–100.
[10] J.D. Knowles, D.W. Corne, Approximating the nondominated front using the Pareto archived evolution strategy, Evolutionary Computation 8 (2) (2000) 149–172.
[11] D.W. Corne, J.D. Knowles, M.J. Oates, The Pareto envelope-based selection algorithm for multiobjective optimization, in: Proceedings of the Parallel Problem Solving from Nature, Paris, France, 2000, pp. 839–848.
[12] D.W. Corne, J.D. Knowles, M.J. Oates, PESA-II: region-based selection in evolutionary multiobjective optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO' 2001), San Francisco, California, 2001, pp. 283–290.
[13] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.
[14] M. Laumanns, L. Thiele, K. Deb, E. Zitzler, Combining convergence and diversity in evolutionary multi-objective optimization, Evolutionary Computation 10 (3) (2002) 263–282.
[15] A.G. Hernández-Díaz, L.V. Santana-Quintero, C.A. Coello Coello, J. Molina, Pareto-adaptive epsilon-dominance, Evolutionary Computation 15 (4) (2007) 493–517.
[16] L. Ben Said, S. Bechikh, K. Ghedira, The r-dominance: a new dominance relation for interactive evolutionary multicriteria decision making, IEEE Transactions on Evolutionary Computation 14 (5) (2010) 801–818.
[17] D. Brockoff, E. Zitzler, Are all objective necessary on dimensionality reduction in evolutionary multi-objective optimization, in: Proceeding of Parallel Problem Solving from Nature, PPSN IX, LNCS, Springer-Verlag, Berlin, 2006, pp. 533–542.
[18] H. Ishibuchi, T. Murata, Multi-objective genetic local search algorithm and its application to flowshop scheduling, IEEE Transactions on Systems, Man and Cybernetics 28 (3) (1998) 392–403.
[19] H. Ishibuchi, T. Yoshida, T. Murata, Balance between genetic search and local search in memetic algorithm for multiobjective permutation flowshop scheduling, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 204–223.
[20] Y.W. Leung, Y. Wang, Multiobjective programming using uniform design and genetic algorithm, IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 30 (3) (2000) 293–304.
[21] A. Jaszkiewicz, On the performance of multiple-objective genetic local search on the 0/1 knapsack problem – a comparative experiment, IEEE Transactions on Evolutionary Computation 6 (4) (2002) 402–412.
[22] Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.
[23] C.A. Coello Coello, G.T. Pulido, M.S. Lechuga, Handling multiple objectives with particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (3) (2004) 256–279.
[24] H. Li, Q. Zhang, Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II, IEEE Transactions on Evolutionary Computation 12 (2) (2009) 284–302.
[25] C. Igel, N. Hansen, S. Roth, Covariance matrix adaptation for multi-objective optimization, Evolutionary Computation 15 (1) (2007) 1–28.
[26] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evolutionary Computation 9 (2) (2001) 159–195.
[27] K. Ghoseiria, B. Nadjari, An ant colony optimization algorithm for the bi-objective shortest path problem, Applied Soft Computing 10 (4) (2010) 1237–1246.
[28] K. Jamuna, K.S. Swarup, Multi-objective biogeography based optimization for optimal PMU placement, Applied Soft Computing 12 (5) (2012) 1457–1620.
[29] Z. Zhang, Immune optimization algorithm for constrained nonlinear multiobjective optimization problems, Applied Soft Computing 7 (3) (2007) 840–857.
[30] E. Zitzler, D. Brockhoff, L. Thiele, The hypervolume indicator revisited: on the design of Pareto-compliant indicators via weighted integration Proceeding of Conference on Evolutionary Multi-Criterion Optimization (EMO 2007), vol. 4403, Lecture Notes in Computer Science, Berlin, 2007, pp. 862–876.
[31] D. Brockhoff, T. Friedrich, F. Neumann, Analyzing hypervolume indicator based algorithms Proceeding of Parallel Problem Solving From Nature (PPSN X), vol. 5199, Springer, Lecture Notes in Computer Science, 2008, pp. 651–660.
[32] K. Miettinen, in: M.A. Norwell (Ed.), Nonlinear Multiobjective Optimization, vol. 12, Kluwer, Kluwer's International Series in Operations Research & Management Science, 1999.
[33] M. Ehrgott, Multicriteria Optimization, vol. 491, Springer, Lecture Notes in Economics and Mathematical Systems, 2005.
[34] Q. Zhang, A. Zhou, Y. Jin, RM-MEDA: a regularity model based multiobjective estimation of distribution algorithm, IEEE Transactions on Evolutionary Computation 12 (1) (2008) 41–63.
[35] P. Larrañaga, J.A. Lozano, Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation, Kluwer, Norwell, MA, 2001.
[36] N. Kambhatla, T.K. Leen, Dimension reduction by local principal component analysis, Neural Computation 9 (7) (1997) 1493–1516.
[37] E. Zitzler, K. Deb, L. Thiele, Comparison of multiobjective evolutionary algorithms: empirical results, Evolutionary Computation 8 (2) (2000) 173–195.
[38] K. Deb, Multi-objective genetic algorithms: problem difficulties and construction of test problems, Evolutionary Computation 7 (3) (1999) 205–230.
[39] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multiobjective optimization, in: Evolutionary Multiobjective Optimization, Theoretical Advances and Applications, Springer, New York, 2005, pp. 105–145.
[40] J.R. Charnetski, R.M. Soland, Multiple-attribute decision making with partial information: the comparative hypervolume criterion, Naval Research Logistics Quarterly 25 (2) (1978) 279–288.
[41] A.J. Nebro, J.J. Durillo, C.A. Coello Coello, F. Luna, E. Alba, A study of convergence speed in multi-objective metaheuristics Proceeding of Parallel Problem Solving from Nature, PPSN X, vol. 5199, Lecture Notes in Computer Science, 2008, pp. 763–772.
[42] Q. Li, On the angle between two dimensional planes in the n-dimensional Euclidean space, Journal of Mathematics 6 (4) (1986) 407–410 (in Chinese with English abstract).
[43] H. Lu, High Dimension Euclidean Geometry, http://www.gwjhx.com/.