RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm

Qingfu Zhang, Aimin Zhou, Yaochu Jin

2008

Preprint:

This is an accepted article published in IEEE Transactions on Evolutionary Computation. The final authenticated version is available online at: https://doi.org/[DOI not available]

Modelling the Regularity in an Estimation of Distribution Algorithm for Continuous Multiobjective Optimization with Variable Linkages

Qingfu Zhang, Aimin Zhou and Yaochu Jin

***Abstract*— Under mild conditions, it can be induced from the Karush-Kuhn-Tucker condition that the Pareto set, in the decision space, of a continuous multiobjective optimization problem is $(m-1)$-D piecewise continuous, where m is the number of objectives. Based on this regularity property, we propose a Regularity Model based Multiobjective Estimation of Distribution Algorithm (RM-MEDA) for continuous multiobjective optimization problems with variable linkages. At each generation, the proposed algorithm models a promising area in the decision space by a probability distribution whose centroid is a $(m-1)$-D piecewise continuous manifold. The Local PCA algorithm is used for building such a model. New trial solutions are sampled from the model thus built. A non-dominated sorting based selection is used for choosing solutions for the next generation. Systematic experiments have shown that, overall, RM-MEDA outperforms other three state-of-the-art algorithms, namely, GDE3, PCX-NSGA-II and MIDEA, on a set of test instances with variable linkages. We have demonstrated that, compared with GDE3, RM-MEDA is not sensitive to algorithmic parameters, and has good scalability to the number of decision variables in the case of nonlinear variable linkages. A few shortcomings of RM-MEDA have also been identified and discussed in this paper.**

***Index Terms*— Multiobjective optimization, estimation of distribution algorithm, regularity, Karush-Kuhn-Tucker condition, local principal component analysis, variable linkages, sensitivity, scalability.**

I. Introduction

Multiobjective optimization problems (MOP) arise in many engineering areas. Very often, the objectives in a MOP conflict with each other, no single solution can optimize all the objectives at the same time. The Pareto set/front is the set of all the optimal tradeoffs in the decision/objective space. When the preference of a decision maker is unavailable or very difficult to specify mathematically as it is in many applications, the decision maker usually requires an approximation to the Pareto set and/or Pareto front for making their final choice. Such an approximation can be a finite set of Pareto optimal solutions or a mathematical approximation model of the Pareto set/front.

Since the publication of Schaffer's seminal work [1], a number of *evolutionary algorithms* (EA) have been developed for multiobjective optimization problems [2]–[5]. The major advantage of these *multiobjective evolutionary algorithms* (MOEA) over other methods are that they work with a population of candidate solutions and thus can produce a set of Pareto optimal solutions to approximate the Pareto front and/or Pareto set in a single run. The current MOEA research mainly focuses on the following highly related issues:

- **Fitness Assignment and Diversity Maintenance:** Like their counterparts for scalar optimization, most MOEAs employ a selection operator to direct its search into promising areas in the decision space. Since Pareto domination is not a complete ordering, conventional selection operators, which were originally developed for scalar objective optimization, cannot be directly applied to multiobjective optimization. Furthermore, the task of most MOEAs is to produce a set of solutions which are evenly distributed in the Pareto front, a selection operator in MOEAs should not encourage the search to converge to a single point. Therefore, it is not a trivial job to assign a relative fitness value to each individual solution for reflecting its utility in selection in MOEAs. Fitness assignment has been subject to much research over the last two decades [6]–[8]. Some techniques such as fitness sharing and crowding have been frequently used within fitness assignment for maintaining the diversity of search [9], [10].
- **External Population:** It is usually very hard to balance diversity and convergence with a single on-line population in current MOEAs. The on-line population may be unable to store some representative solutions found during the search due to its limited size. To overcome this shortcoming, an external population (archive) is often used in MOEAs for recording nondominated solutions found during the search. Some effort has been made to study how to maintain and utilize such an external population [11], [12].
- **Combination of MOEA and Local Search:** Combinations of EAs and local search heuristics, often called memetic algorithms, have been shown to outperform traditional evolutionary algorithms in a wide variety of scalar objective optimization problems. A few *multiobjective memetic algorithms* (MOMA) have been developed over the past decade. MOMAs need to consider how to evaluate solution quality for their local search operators. In *multiobjective genetic local search* (MOGLS) [13], [14], a scalarizing function with random weights are used as its evaluation function in both its local search and selection. *Memetic Pareto archived evolution strategy*

Q. Zhang and A. Zhou are with Department of Computer Science, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, U.K ({qzhang, azhou}@essex.ac.uk).

Y. Jin is with Honda Research Institute Europe, Carl-Legien-Str. 30, 63073 Offenbach, Germany (Yaochu.Jin@honda.ri.de).

(M-PAES) evaluates solutions by using domination ranks [15].

Surprisingly, not much work has been done on how to generate new solutions in MOEAs. The implementations of most current MOEAs directly adopt traditional genetic recombination operators such as crossover and mutation. The characteristics of MOPs have not well been utilized in generating new solutions in current MOEAs. Very recently, Deb *et al.* have compared the performance of several recombination operators on some test problems with variable linkages [16]. Their experimental results suggest that variable linkages could cause difficulties for MOEAs and recombination operators are crucial to the performance of a MOEA.

It has been observed that under mild smoothness conditions, the Pareto set (in the decision space) of a continuous MOP is a piecewise continuous $(m-1)$-dimensional manifold where m is the number of the objectives. This observation has been used in several mathematical programming methods for approximating the Pareto front or Pareto set [17]–[19]. However, as suggested in [20], such regularity has not been exploited explicitly by most current MOEAs except for those combining local search that take advantage of the regularity implicitly, such as the dynamic weighted aggregation method [21]. The work in this paper will show that reproduction of new trial solutions based on this regularity property can effectively cope with the variable linkages in continuous MOPs.

Estimation of distribution algorithms (EDA) are a new computing paradigm in evolutionary computation [22]. There is no crossover or mutation in EDAs. Instead, they explicitly extract globally statistical information from the selected solutions and build a posterior probability distribution model of promising solutions, based on the extracted information. New solutions are sampled from the model thus built and fully or in part replace the old population. Several EDAs have been developed for continuous MOPs [23]–[25]. However, these EDAs do not take the regularity into consideration in building probability models. Note that probability modelling techniques under regularity have been widely investigated in the area of statistical learning [26], [27], it is very suitable to take the advantage of the regularity in the design of EDAs for a continuous MOP.

As one of the first attempts to capture and utilize the regularity of the Pareto set in the decision space, we have recently proposed using local principal component analysis for extracting regularity patterns of the Pareto set from the previous search. We have studied two naive hybrid MOEAs in which some trial solutions are generated by traditional genetic operators and others by sampling from probability models built based on regularity patterns [28], [29]. Our preliminary experimental results are very encouraging. This paper conducts a further and thorough investigation alone this line. In comparison with our work presented in [28] and [29]. The main contributions of this paper include:

- Based on the regularity property of continuous MOPs, an estimation of distribution algorithm for continuous multiobjective optimization has been developed. Unlike our previous algorithms, it does not use crossover or mutation operators for producing new offspring. And besides, the parameter estimation for model building in this paper is more statistically sound and yet simpler than that used in [28] and [29]. Our experimental studies have shown that the algorithm presented in this paper performs similarly to its predecessors.[1]
- Systematic experiments have been conducted to compare the proposed algorithm with other three state-of-the art algorithms on a set of bi- or tri-objective test instances with linear or nonlinear variable linkages. Only original NSGA-II with SBX [30] was compared on a few test instances in [28] and [29].
- The sensitivity to algorithmic parameters and the scalability to the number of decision variables of the proposed algorithm and Generalized Differential Evolution 3 (GDE3) [31] have been experimentally studied.

The rest of the paper is organized as follows. The next section introduces continuous multiobjective optimization problems, Pareto optimality and the regularity property induced from Karush-Kuhn-Tucker condition. Section III presents the motivation and the details of the proposed algorithm. Section IV briefly describes the three other algorithms used in comparison. Section V presents and analyzes the experimental results. Section VI experimentally studies the sensitivity and scalability, and presents CPU-time costs of the proposed algorithm and GDE3. The final section concludes the paper and outlines future research work.

II. Problem Definition

In this paper, we consider the following *continuous multiobjective optimization problem* (continuous MOP):

$$\begin{aligned} \text{minimize} \quad & \vec{F}(x) = (f_1(x), f_2(x), \ldots, f_m(x))^T \\ \text{subject to} \quad & x \in X \end{aligned} \tag{1}$$

where $X \subset R^n$ is the decision space and $x = (x_1, \ldots, x_n)^T \in R^n$ is the decision variable vector. $\vec{F} : X \to R^m$ consists of m real-valued continuous objective functions $f_i(x)$ $(i = 1, 2, \ldots, m)$. R^m is the objective space. In the case of $m = 2$, this problem is referred to as a continuous bi-objective optimization problem.

Let $a = (a_1, \ldots, a_m)^T$, $b = (b_1, \ldots, b_m)^T \in R^m$ be two vectors, a is said to *dominate* b, denoted by $a \prec b$, if $a_i \leq b_i$ for all $i = 1, \ldots, n$, and $a \neq b$. A point $x^\star \in X$ is called *(globally) Pareto optimal* if there is no $x \in X$ such that $\vec{F}(x) \prec \vec{F}(x^\star)$. The set of all the Pareto optimal points, denoted by PS, is called the *Pareto set*. The set of all the Pareto objective vectors, $PF = \{y \in R^m | y = \vec{F}(x), x \in PS\}$, is called the *Pareto front* [2], [17].

Under certain smoothness assumptions, it can be induced from Karush-Kuhn-Tucker condition that the *PS* of a continuous MOP defines a piecewise continuous $(m-1)$-dimensional manifold in the decision space [17], [19]. Therefore, the *PS* of a continuous bi-objective optimization problem is a piecewise continuous curve in R^n while the *PS* of a continuous MOP with three objectives is a piecewise continuous surface. In fact,

[1]Due to the page limit of this paper, the experimental comparison between this algorithm and its predecessors will be presented in a furthercoming working report, but not in this paper.

the *PS*s of ZDT test problems [6], [20], [32], the widely-used test instances for continuous bi-objective optimization problems in the evolutionary computation community, are line segments in the decision space.

III. Algorithm

A. Basic Idea

EDAs build a probability model for characterizing promising solutions in the search space based on statistical information extracted from the previous search and then sample new trial solutions from the model thus built. A very essential issue is what kind of model one should use for such a task. A good model should be easy to build and sample, and be able to describe promising areas with good fidelity [33], [34].

The population in the decision space in an EA for (1) will hopefully approximate the *PS* and be uniformly scattered around the *PS* as the search goes on. Therefore, we can envisage the points in the population as independent observations of a random vector $\xi \in R^n$ whose centroid is the *PS* of (1). Since the *PS* is an $(m-1)-$dimensional piecewise continuous manifold, ξ can be naturally described by:

$$\xi = \zeta + \varepsilon \tag{2}$$

where ζ is uniformly distributed over a piecewise continuous $(m-1)$-dimensional manifold. ε is an n-dimensional zero-mean noise vector.

Fig.1 illustrates our basic idea.

B. Algorithm Framework

At each generation t, the proposed algorithm for (1), called *Regularity Model based Multiobjective Estimation of Distribution Algorithm* (RM-MEDA) hereafter, maintains:

- a population of N solutions (i. e. points in X):

  $$Pop(t) = \{x^1, x^2, \ldots, x^N\}$$

  and
- their $\vec{F}$-values: $\vec{F}(x^1), \vec{F}(x^2), \ldots, \vec{F}(x^N)$.

The algorithm works as follows:

RM-MEDA

Step 0 Initialization: Set $t := 0$. Generate an initial population $Pop(0)$ and compute the $\vec{F}$-value of each individual solution in $Pop(0)$.

Step 1 Stopping Condition: If stopping condition is met, stop and return the nondominated solutions in $Pop(t)$ and their corresponding $\vec{F}$ vectors. All these $\vec{F}$ vectors constitute an approximation to the *PF*.

Step 2 Modelling: Build the probability model (2) for modelling the distribution of the solutions in $Pop(t)$.

Step 3 Reproduction: Generate a new solution set Q from the model (2). Evaluate the $\vec{F}$-value of each solution in Q.

Step 4 Selection: Select N individuals from $Q \bigcup Pop(t)$ to create $Pop(t+1)$.

Step 5 Set $t := t + 1$ and go to **Step 1**.

In the following, we discuss the implementation of modelling, reproduction and selection in the above algorithm.

C. Modelling

Fitting the model (2) to the points in $Pop(t)$ is highly related to principal curve or surface analysis, which aims at finding a central curve or surface of a set of points in R^n [35]. However, most current algorithms for principal curve or surface analysis are rather expensive due to the intrinsic complexity of their models. Since modelling needs to be conducted at each generation of RM-MEDA, a too complicated model is not affordable in RM-MEDA. On the other hand, a complicated model is not necessary since the actual distribution of the points in $Pop(t)$ could hardly be exactly described by (2).

In our implementation, we assume, for the sake of simplicity, that the centroid of ξ consists of K manifolds $\Psi^1, \ldots, \Psi^K$ (i.e., ζ in (2) is uniformly distributed on these manifolds), each Ψ^j is a $(m-1)$-dimensional hyper-rectangle. Particularly,

- in the case of two objectives (i.e., $m = 2$): Each Ψ^j is a line segment in R^n.
- in the case of three objectives (i.e., $m = 3$): Each Ψ^j is a 2-D rectangle in R^n.

Let A^j denote the event that ζ is from Ψ^j. Then

$$\sum_{j=1}^{K} Prob(A^j) = 1.$$

We make the following assumptions that under the condition that the event A^j happens:

- ζ is uniformly randomly generated over Ψ^j.
- $\varepsilon \sim N(0, \sigma_j I)$, where I is the $n \times n$ identity matrix and $\sigma_j > 0$. In other words, all the components in ε are i. i. d..

The modelling problem, under the above assumptions, becomes estimating Ψ^j, σ_j and $Prob(A^j)$ $(j = 1, 2, \ldots, K)$. To solve it, we need to firstly partition $Pop(t)$ into K disjoint clusters $S^1, \ldots, S^K$ such that the points in S^j can be regarded as being sampled under the condition of A^j. Then we can estimate the parameters.

In this paper, we use *the* $(m-1)$*-dimensional Local Principal Component Analysis* ($(m-1)$-D Local PCA) algorithm [36] for partitioning $Pop(t)$. Let S be a finite subset of R^n, the (sample) mean of S is

$$\bar{x} = \frac{1}{|S|} \sum_{x \in S} x$$

where $|S|$ is the cardinality of S. The (sample) covariance matrix of the points in S is

$$Cov = \frac{1}{|S| - 1} \sum_{x \in S} (x - \bar{x})(x - \bar{x})^T.$$

The i-th principal component U^i is a unity eigenvector associated with the i-th largest eigenvalue of the matrix Cov. Then the affine $(m-1)$-dimensional principal subspace of the points in S is defined as

$$\{x \in R^n | x = \bar{x} + \sum_{i=1}^{m-1} \theta_i U^i, \theta_i \in R, i = 1, 2, \ldots, m-1\}.$$

The partition $S^1, \ldots, S^K$ obtained by the $(m-1)$-D Local PCA algorithm minimizes the following error function:

$$\sum_{j=1}^{K} \sum_{x \in S^j} dist(x, \mathcal{L}_j^{m-1})^2$$

where $\mathcal{L}_j^{m-1}$ is the affine $(m-1)$-dimensional principal subspace of the points in S^j, $dist(x, \mathcal{L}_j^{m-1})$ is the Euclidean distance from x to its projection in $\mathcal{L}_j^{m-1}$.

The $(m-1)$-D Local PCA [36] partitions $Pop(t) = \{x^1, \ldots, x^N\}$ into $S^1, \ldots, S^K$ in the following iterative way:

Step 1 Randomly initialize $\mathcal{L}_i^{m-1}$ to be an affine $(m-1)$-dimensional space containing a point randomly chosen from $Pop(t)$.

Step 2 Partition the points in $Pop(t)$ into K disjoint clusters $S^1, \ldots, S^K$:

$$\begin{aligned} S^j \quad = \{x \mid \quad & x \in Pop(t), \text{ and } dist(x, \mathcal{L}_j^{m-1}) \\ & \leq dist(x, \mathcal{L}_k^{m-1}) \text{ for all } k \neq i\}. \end{aligned}$$

Step 3 Set $\mathcal{L}_j^{m-1}$ $j = 1, \ldots, K$ to be the affine $(m-1)$-dimensional principal subspace of the points in S^j.

Step 4 Iterate Steps 2 and 3 until no change in partition is made.

The Local PCA algorithm is more suitable for partitioning $Pop(t)$ for building the model (2) than the widely used K-means clustering method in which a cluster centroid is a point [27], since we assume that the centroid of each cluster should be a $(m-1)$-D hyper-rectangle instead of a point.

Schematically, modelling (i.e., estimating Ψ_k and σ_k) in RM-MEDA works as follows:

Modelling by the $(m-1)$-D Local PCA

Step 2.1 Partition $Pop(t)$ into K disjoint clusters $S^1, \ldots, S^K$ by the $(m-1)$-D Local PCA algorithm.

Step 2.2 For each cluster S^j. Let $\bar{x}^j$ be its mean and U_i^j be its i-th principal component. Compute

$$a_i^j = \min_{x \in S^j} (x - \bar{x}^j)^T U_i^j \tag{3}$$

and

$$b_i^j = \max_{x \in S^j} (x - \bar{x}^j)^T U_i^j \tag{4}$$

for $i = 1, 2, \ldots, m-1$. Then set

$$\begin{aligned} \Psi^j = \{x \in R^n | x = \bar{x}^j + \textstyle\sum_{i=1}^{m-1} \alpha_i U_i^j, \\ a_i^j - 0.25(b_i^j - a_i^j) \leq \alpha_i \leq b_i^j + 0.25(b_i^j - a_i^j), \\ i = 1, \ldots, m-1.\}. \end{aligned}$$

Let λ_i^j be the $i-$th largest eigenvalue of the covariance matrix of the points in S^j, set

$$\sigma_j = \frac{1}{n-m} \sum_{i=m}^{n} \lambda_i^j. \tag{5}$$

We would like to make the following comments on the above modelling method:

- The smallest hyper-rectangle, containing the projections of all the points of S^j in the affine $(m-1)$-dimensional principal subspace of S^j, is

$$\begin{aligned} \Phi^j \quad = \quad & \{x \in R^n | x = \bar{x} + \sum_{i=1}^{m-1} \alpha_i U_i^j, \\ & a_i^j \leq \alpha_i \leq b_i^j, i = 1, \ldots, m-1\}. \end{aligned}$$

  Ψ^j in Step 2.2 extends Φ^j by 50% along each of the directions $U_1^j, \ldots, U_{m-1}^j$. The motivation behind this extension is that $Pop(t)$ often could not cover the whole *PS* and thus the union of all these smallest hyper-rectangles Φ^j can only approximate part of the *PS*, while the union of all the Ψ^j's may provide a good approximation to the *PS*. Fig. 2 illustrates this motivation.
- $\lambda_m^j, \lambda_{m+1}^j, \ldots, \lambda_n^j$ characterize the derivation of the points in S^j from its centroid. It is ideal to model ε as

$$\varepsilon = \sum_{i=m}^{n} \lambda_i^j U_i^j \varepsilon^i \tag{6}$$

  when $\varepsilon^m, \ldots, \varepsilon^n$ are $n-m$ independent $N(0,1)$ noises. Since it is often the case that $m << n$ and λ_i^j, $i = m, \ldots, n$ is very small compared with the first $(m-1)$ largest eigenvalues, the difference between the noise sampled from $N(0, \sigma_j)$ and that in (6) is tiny when σ_j is defined as in (5). Moreover, sampling from (6) is more complicated than from $N(0, \sigma_j)$. This is why we use $N(0, \sigma_j)$ for facilitating the sampling procedure. σ_j also controls the degree of the exploration of the search in the reproduction step.

D. Reproduction

In our implementation of Reproduction in this paper, N new solutions are generated. The first issue we need to consider is how many new trial solutions are generated around each Ψ^i. Since it is desirable that final solutions are uniformly distributed on the *PS*. We set

$$Prob(A^j) = \frac{vol(\Psi^j)}{\sum_{i=1}^{K} vol(\Psi^i)}$$

where $vol(\Psi^j)$ is the $(m-1)$-D volume of Ψ^j. Therefore, the probability that a new solution is generated around Ψ^j is proportional to $vol(\Psi^j)$.

A simple reproduction scheme for generating a new solution x, used in our experimental studies, works as follows:

Reproduction by Sampling (Reproduction/S)

Step 1 Randomly generate an integer $\tau \in \{1, 2, \ldots, K\}$ with

$$Prob(\tau = k) = \frac{vol(\Psi^k)}{\sum_{j=1}^{K} vol(\Psi^j)}$$

Step 2 Uniformly randomly generate a point x' from Ψ^τ. Generate a noise vector ε' from $N(0, \sigma_\tau I)$.

Step 3 Return $x = x' + \varepsilon'$.

In Step 3 in RM-MEDA, N new solutions can be produced by repeating Reproduction/S N times.

In Reproduction/S, x', the main part of x is from Ψ^τ, which is the extension of Φ^τ. Therefore, the search will, hopefully, perform exploitation around the *PF*. Such exploitation, guided by the extension, is mainly along the first $m-1$ principal components. ε', the noise part of x is necessary since Ψ^τ is just an approximation to part of the *PF*. Moreover, it provides diversity for the further search.

E. Selection

In principle, any selection operators for MOEAs can be used in RM-MEDA. The selection procedure used in the experimental studies of this paper is based on the non-dominated sorting of NSGA-II [6]. It is called NDS-Selection in the following.

NDS-Selection partitions $Q \cup Pop(t)$ into different fronts $F_1, \ldots, F_l$ such that the j-th front F_j contains all the non-dominated solutions in $\{Q \cup Pop(t)\} \backslash (\cup_{i=1}^{j-1} F_i)$. Therefore, there is no solution in $\{Q \cup Pop(t)\} \backslash (\cup_{i=1}^{j-1} F_i)$ that could dominate a solution in F_j. Roughly speaking, F_1 is the best non-dominated front in $Q \cup Pop(t)$, F_2 is the second best non-dominated front and so on.

The crowding distance, $d_c(x, S)$, of point x in set S is defined as the average side length of the largest m-D rectangle in the objective space subject to the two constraints: a) each of its sides is parallel to a coordinate axis, and b) $\vec{F}(x)$ is the only point in $\vec{F}(S) = \{\vec{F}(y) | y \in S\}$ that is an interior point in the rectangle. A solution with larger crowding distance should be given priority to enter $Pop(t+1)$ since it could increase the diversity of $Pop(t+1)$.

The details of the selection procedure is given as follows:

NDS-Selection

Step 1 Partition $Q \cup Pop(t)$ into different fronts $F_1, \ldots, F_l$ by using the fast non-dominated sorting approach. Set $Pop(t+1) = \emptyset$ and $k = 0$.

Do

$k = k + 1$,

$Pop(t+1) = Pop(t+1) \cup F_k$,

Until $|Pop(t+1)| \geq N$.

Step 2 While $|Pop(t+1)| > N$, Do

For all the members in $F_k \cap Pop(t+1)$, compute their crowding distances in $F_k \cap Pop(t+1)$. Remove the element in $F_k \cap Pop(t+1)$ with the smallest crowding distance from $Pop(t+1)$. In the case when there are more than one members with the smallest crowding distance, randomly choose one and remove it.

This selection operators selects N members from $Q \cup Pop(t)$ for forming $Pop(t+1)$. Step 1 implements an elitism mechanism. The best fronts in $Q \cup Pop(t)$ are added to $Pop(t+1)$. After Step 1,

$$Pop(t+1) = \bigcup_{j=1}^{k} F_j,$$

the last front in $Pop(t+1)$ is F_k, the k-th front, which contains the "worst" solutions in $Pop(t+1)$. When $|Pop(t+1)|$ is larger than N, $|F_k|$ will be larger than $|Pop(t+1)| - N$. Step 2 removes $|Pop(t+1)| - N$ worst solutions from $Pop(t+1)$ to reduce its size to N. The crowding distance is used to compare the quality of solutions in Step 2.

This selection procedure is the same as that used in NSGA-II except that we remove solutions from $Pop(t+1)$ one by one and we re-calculate the crowding distances before deciding which solution should be deleted from $Pop(t+1)$, which can increase the diversity of $Pop(t+1)$ at extra computational cost. We noticed that GDE3 [31] use a very similar selection.

IV. Three other algorithms in comparison

The major purpose of this work is to tackle variable linkages in continuous MOPs. The studies conducted in [16] show that PCX-NSGA-II and GDE3 [31] perform better than others algorithms for continuous MOPs with variable linkages. In this paper, we compare RM-MEDA with these two algorithms. Since RM-MEDA is an EDA based on regularity, we also compare it with MIDEA [24], which is an EDA without using regularity, to investigate whether or not using regularity can improve the performance of EDAs.

A. Generalized Differential Evolution 3 (GDE3) [31]

Let $x = (x_1, \ldots, x_n)^T$ be a solution in the current population, its offspring $y = (y_1, \ldots, y_n)^T$ in GDE3 is generated as follows:

1. Randomly select from the current population three distinct solutions x^{r1}, x^{r2}, x^{r3}.
2. Set $z = x^{r1} + F \times (x^{r2} - x^{r3})$.
3. Set the i-th element of y:

$$y_i = \begin{cases} z_i & \text{if } rand < CR, \\ x_i^{r1} & \text{otherwise.} \end{cases}$$

where $rand$ is a random number uniformly generated from $[0, 1]$, F and CR are two control parameters.

In GDE3, all the solutions in the current population first generate their offspring, these offspring and their parents undergo a selection operator similar to one used in RM-MEDA and then the selected solutions form a new population for the next generation. The details of GDE3 can be found in [31]. The code of GDE3 used in comparison is written in C by ourselves.

B. PCX-NSGA-II [16]

Parent-Centric Recombination (PCX) [37] generates a trial solution y from μ parent solutions $x^1, \ldots, x^\mu$ as follows:

1. Select a solution x^p from $x^1, \ldots, x^\mu$ and let it be the "index parent".
2. Compute the direction vector d^p from x^p to the center of these μ parent solutions, i.e., $d^p = x^p - \frac{1}{\mu}\sum_{i=1}^{\mu} x^i$, and $n-1$ orthogonal directions $e^1, \ldots, e^n$ to d^p. Compute average prependicular distance, $\overline{D}$, of the other $\mu - 1$ parents to the line from passing x^p and the center.
3. Set

$$y = x^p + \omega_\zeta d^p + \sum_{i=1, i \neq p}^{\mu} \omega_\eta \overline{D} e^i$$

where ω_ζ and ω_η are two zero-mean normally distributed random variables with variance σ_ζ^2 and σ_η^2, respectively.

In PCX-NSGA-II [16], PCX with $\mu = 3$ is used to generate a set of new trial solutions. All of these new solutions are mutated by a polynomial mutation operator. The PCX-NSGA-II used in comparison is implemented in C by modifying the code of SBX-NSGA-II from KanGAL web (http://www.iitk.ac.in/kangal/). In our implementation of PCX-NSGA-II, the NDS-selection operator described in Section III.E is used for selecting from the old solutions and new solutions for creating a new population for the next generation, this is the only difference from the implementation of Deb *et al.* in [16]. Our first experiments show that our implementation is slightly better in terms of solution quality. The motivation that we use the NDS-selection in the implementation of PCA-NSGA-II is to have a fair comparison with RM-MEDA.

C. MIDEA [24]

MIDEA is an EDA for MOPs. It has been applied to continuous MOPs. Its major difference from RM-MEDA is that it uses a mixture of Gaussians. It does not take the regularity of the *PS* into consideration and thus does not impose any constraints on distribution of the centers of the Gaussians. A specialized diversity preserving selection is used in MIDEA. The number of Gaussians is determined by an adaptive clustering method. The C code of MIDEA written by its authors is used in our experimental studies.

V. Comparison Studies

A. Performance Metric

The inverted generational distance (IGD) is used in assessing the performance of the algorithms in our experimental studies.

Let P^* be a set of uniformly distributed points in the objective space along the *PF*. Let P be an approximation to the *PF*, the inverted generational distance from P^* to P is defined as:

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

where $d(v, P)$ is the minimum Euclidean distance between v and the points in P. If $|P^*|$ is large enough to represent the *PF* very well, $D(P^*, P)$ could measure both the diversity and convergence of P in a sense. To have a low value of $D(P^*, P)$, P must be very close to the *PF* and cannot miss any part of the whole *PF*.

In our experiments, we select 500 evenly distributed points in PF and let these points be P^* for each test instance with 2 objectives, and $1,000$ points for each test instance with 3 objectives.

IGD has been recently used by some other researchers (e.g, [38]). As its name suggests, it is an inverted variation of the widely-used generational distance (GD) performance metric. [39]. The GD represents the average distance of the points in an approximation to the true *PF*, which cannot effectively measure the diversity of the approximation. There are several different ways in computing and averaging the distances in the GD performance metrics (e.g., [39] and [6]), the version of IGD used in this paper inverts the Υ version of GD in [6].

B. General Experimental Setting

RM-MEDA is implemented in C. The machine used in our experiments is Pentium(R) 4 (3.40GHz, 1.00GB RAM). In this section, the experimental setting is as follows:

- *The number of new trial solutions generated at each generation*: The number of new solutions generated at each generation in all the four algorithms is set to be 100 for all the test instances with two objectives, and 200 for the test instances with three objectives. Since in RM-MEDA, GDE3 and PCX-NSGA-II, the population size is the same as the number of new trial solutions generated at each generation, these three algorithms have the same population size for each test instance in our experiments.
- *The Number of Decision Variables:* It is set to be 30 for all the test instances.
- *Parameter Setting in RM-MEDA:* In Local PCA algorithm, K, the number of clusters, is set to be 5.
- *Parameter Setting in GDE3:* Both CR and F in the DE operator is set to be 1, which was the best setting for most test instances in the simulation studies in [16].
- *Parameter Setting in PCX-NSGA-II:* σ in PCX is set to be 0.4, which worked very well for most test instances in the simulation studies in [16].
- *Parameter Setting in MIDEA:* τ is set to be 0.3, therefore, the population size is $\lceil \frac{100}{1-0.3} \rceil = 143$ in the case of two objectives and $\lceil \frac{200}{1-0.3} \rceil = 286$ in the case of three objectives. In [24], large populations were used in simulation studies. Our first experiments show that a large population does not improve the performance of MIDEA on the test instances with variable linkages significantly. Following [24], we set $\delta = 1.5$.
- *Number of Runs and Stopping Condition:* We run each algorithm independently 20 times for each test instance. The algorithms stop after a given number of $\vec{F}$-function evaluations. The maximal number of function evaluations in each algorithm is 10,000 for F1, F2, F5 and F6, 40,000 for F4 and F8, and 100,000 for F3, F7, F9 and F10.
- *Dealing with Boundary in RM-MEDA:* In all the test instances, the feasible decision space is a hyperrectangle. If an element of a solution x, sampled from a model in RM-MEDA, is out of boundary, we simple reset its value to be a randomly selected value inside the boundary. This method is very similar to that used in GDE3.
- *Initialization:* Initial populations in all the algorithms are randomly generated.

C. Test Instances with Linear Variable Linkages

We first compare RM-MEDA with the other three algorithms on continuous MOPs with linear variable linkages. The test instances F1-4 in Table I are used for this purpose.

F1-4 are variants of ZDT1, ZDT2, ZDT6 [6] and DTLZ2 [40], respectively. Due to $g(x)$ used in these instances, F1-3

have the same *PS*. Their *PS* is a line segment:

$$x_1 = \ldots = x_n,$$
$$0 \leq x_i \leq 1, 1 \leq i \leq n.$$

The *PS* of F4 is a 2-D rectangle:

$$x_1 = x_3 = \ldots = x_n,$$
$$0 \leq x_i \leq 1, 1 \leq i \leq n.$$

There are linear variable linkages in these test instances. The variable linkages in these instances [41] are obtained by performing the following linear mapping on the variables in the original ZDT and DTLZ instances:

$$\begin{aligned} x_1 &\rightarrow x_1 \\ x_i &\rightarrow x_i - x_1, \ i = 2, \ldots, n. \end{aligned}$$

Therefore, our scheme for introducing variable linkages can be regarded as a special implementation of the general strategy proposed in [16], which is based on variable linear or nonlinear mappings.

Fig. 3-6 show that the evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of function evaluations in four algorithms. As many researchers pointed out [11], no single metric is always able to rank different algorithms appropriately. For this reason, Fig. 7-10 plot the nondominated front with the lowest D-metric obtained in 20 runs of each algorithm on each test instance. All the 20 fronts found are also plotted together for showing their distribution ranges in the objective space in Fig. 7-10.

It is clear from the above experimental results that both RM-MEDA and GDE3 perform significantly better than PCX-NSGA-II and MIDEA on these four test instances. Note than $CR = 1$ is set in GDE3, an offspring y is a linear combination of three old solutions x^{r1}, x^{r2}, x^{r3} and the *PS*s of these four test instances are line segments or a 2-D rectangle. Therefore, if x^{r1}, x^{r2}, x^{r3} are close to the *PS*, so is y. GDE3 exploits the regularity of these *PS*s as RM-MEDA does in a sense. In contrast, PCX-NSGA-II and MIDEA have no efficient mechanism for using the regularity. This could be a major reason why RM-MEDA and GDE3 are winners.

GDE3 slightly outperforms RM-MEDA on F1, F2 and F4 in terms of D-metric. The reason might be that RM-MEDA, like many other EDAs, does not directly use the location information of previous solutions in generating new solutions, which makes it poorer than GDE3 at refining a solution, particularly, where it is close to the *PS*. A possible way to overcome this shortcoming is to hybridize location information and globally statistical information, which has been proved very successful in the guided mutation for scalar optimization in [42] and [43].

F3 is the hardest among these four test instances. The distribution of Pareto optimal solutions in the objective space in this instance is very different from that in the other three ones. If we uniformly sample a number of points in the *PS* of F3 in the decision space, most of the corresponding Pareto optimal vectors in the objective space will be more likely to be in the left part of the *PF*. This makes it very difficult for an algorithm to approximate the whole *PF*. RM-MEDA performs much better than GDE3 on F3. This could be attributed to the fact that RM-MEDA does extension along the principal directions so that it has a good chance to approximate the whole *PF*.

D. Test Problem with Nonlinear Variable Linkages

F5-8 in Table I are test instances with nonlinear variable linkages. The *PS* of F5-7 is a bounded continuous curve defined by

$$x_1 = x_i^2 \quad i = 2, \ldots, n.$$

$$0 \leq x_1 \leq 1.$$

The *PS* of F8 is a 2-D bounded continuous surface defined by:

$$x_1 = x_i^2 \quad i = 3, \ldots, n.$$

$$0 \leq x_1, x_2 \leq 1.$$

The variable linkages in these instances [41] are obtained by performing the following nonlinear mapping on the variables in the original ZDT and DTLZ instances:

$$\begin{aligned} x_1 &\rightarrow x_1 \\ x_i &\rightarrow x_i^2 - x_1, \ i = 2, \ldots, n. \end{aligned}$$

Fig. 11-14 present that the evolution of the average D-metric of the nondominated solutions in the current population and Fig. 15-18 plot the final nondominated fronts founded by each algorithm for each instance.

The experimental results show that RM-MEDA outperforms all other three algorithms on these four instances. In fact, only RM-MEDA is able to produce nondominated solutions which approximate the whole *PF* well, in all or some runs for all the test instances. These results are not surprising since only RM-MEDA considers modelling nonlinear variable linkages. In other three algorithms, even if all the parent solutions are Pareto optimal, it is possible that their offspring is far from the *PS*.

It is evident from Fig. 11-14 that PCX-NSGA-II and MIDEA are stuck in terms of D-metric, which implies that these two algorithms may not be able to make any more improvement on their solution quality even if more computational time is used. This observation, in conjunction with the results of Deb *et al.* [16], suggests that reproduction operators are crucial in MOEAs. One should carefully design these operators for dealing with nonlinear variable linkages.

Fig. 17 suggests that F7 is the hardest instance for RM-MEDA. This might be due to the fact that the Pareto optimal solutions of F7 are not uniformly distributed as in its linear linkage counterpart F3 and RM-MEDA samples points uniformly around the *PS* in the decision variable space. To improve the performance of RM-MEDA on problems like F3 and F7, one may need to consider the distribution of solutions in the objective space when sampling solutions from the models.

E. Test Instances with Many Local Pareto Fronts

There are nonlinear variable linkages in F9 and F10. Furthermore, these two instances have many local Pareto Fronts since their $g(x)$ has many locally minimal points. The experimental results presented in Figure 19 show that RM-MEDA can approximate the *PF* very well in each run and significantly outperforms other three algorithms on F9. It is also evident that all the four algorithms fail in converging to the global Pareto front of F10. The failure could be because that, as Deb *et al.* have noticed on the ability of their NSGA-II [6], it is not an easy task for these evolutionary algorithms to find the globally minimal point of $g(x)$. To tackle MOPs with many local Pareto-optimal fronts, efficient mechanisms for global optimization of a scalar objective may be worthwhile tailored and used in MOEAs.

VI. Sensitivity, Scalability and CPU-Time Costs

The experimental results in Section V have shown that, overall, RM-MEDA outperforms the other three algorithms, and GDE3 came second on these continuous MOP test instances with variable linkages.

We have compared the sensitivity of population size in RM-MEDA and GDE3 and investigated the effect of the number of clusters on the performance of RM-MEDA on several test instances. We also study how the computational cost, in terms of the number of $\vec{F}$-function evaluations, increases as the number of decision variables increases in both RM-MEDA and GDE3. We have also recorded the CPU time used by each algorithm. Due to the limit of the paper length, however, only the experimental results on F5 with nonlinear variable linkages are presented in this section. The parameter settings in the experiments are presented in Tables II and III. 20 independent runs have made for each parameter setting.

A. Sensitivity to Population Size in RM-MEDA and GDE3

To study how the performances of RM-MEDA and GDE3 are sensitive to the population size, we have tried different values of population size: 20, 30, 40, 50, 60, 80, 100, 150, 200, 250, 300, 350, 400 in both algorithms for F5.

Fig. 21 shows the average D-metrics v.s. the numbers of $\vec{F}$-function evaluations under different population size settings in RM-MEDA and GDE3, respectively. It is clear that RM-MEDA is much less sensitive to the setting of population size than GDE3 on F5. In fact, RM-MEDA can, with all the tested population sizes, lower the D-metric value below 0.05 within 20,000 function evaluations. The convergence speed does not change much over different population sizes from 20 to 200 in RM-MEDA. In contrast, the performance of GDE3 is best with the population size being 60 or 100, and worsens considerably for small or large population sizes.

B. Sensitivity to the Number of Clusters in RM-MEDA

We have tested all the different cluster numbers from 1 to 15 in RM-MEDA. Fig. 22 presents the average D-metric values v.s. the numbers of $\vec{F}$-function evaluations with different cluster numbers on F5. As clearly shown in this figure, RM-MEDA is able to reduce the D-metric below 0.002 with $15,000$ function evaluations when the cluster number is from 2 to 13. It is also evident from this figure that the convergence speed does not change dramatically over this range. Thus, we could claim that RM-MEDA is not very sensitive to the setting of the cluster number for MOP instances which are somehow similar to F5. We should point out that the range of appropriate cluster numbers in RM-MEDA is still problem-dependent.

The experimental results in Fig. 22 reveal that RM-MEDA performs poorly when the cluster number is 1 or larger than 13. Part of the reason could be that the Local PCA reduces to classic PCA and consequently underfits the population when the cluster number is 1, and it could overfit in the case when the cluster number is larger than 13.

We have tried RM-MEDA and GDE3 on different numbers of decision variables. Fig. 23 presents the number of the successful runs in which each algorithm has reached each of four given levels of the D-metric within 20,000 function evaluations for F5 with different numbers of decisions variables. The average numbers of function evaluations among the successful runs in each algorithm are also plotted in this figure. It can be seen that RM-MEDA succeeds in every run for all the numbers of decision variables. It is also clear that the number of function evaluations, for lowering the D-metric below each of four levels, linearly scales up with the number of decision variables in RM-MEDA. Although the average numbers of function evaluations among the successful runs in GDE3 also linearly scale up, its successful rate deceases substantially as the number of decision variables increases. In fact, GDE3 cannot reduce the D-metric value below 0.05 in any single run when the number of the decision variables is 100.

The linear scalability of RM-MEDA in this test instance should be due to two facts: a) the *PS* of $F5$ is always a 1-D continuous curve no matter how large its number of the decision variables is, which may explain that the hardness of $F5$ does not exponentially increase as its number of the decision variables increases; and b) in RM-MEDA, only 1-D Local PCA is needed for $F5$ and sampling is always performed along a 1-D curve, then the curse of dimensionality may not exist in this test instance. Note that many real-world continuous MOPs should meet the regularity condition, these results imply that RM-MEDA could be more suitable for solving large-scale MOPs.

C. CPU-Time Cost

The good performance of RM-MEDA does not come without price. Modelling and reproduction in RM-MEDA is more complicated than genetic operators such as crossover and mutation used in other MOEAs. RM-MEDA needs more CPU time for running Local PCA at each generation. The CPU time used by RM-MEDA and GDE3 for $F5$ with different numbers of decision variables are given in Table IV.

Since we use the setting in Section VI.C, the total number of the function evaluations is $20,000$ and the number of generations is 200 in both algorithms. Therefore, the Local PCA was

run 200 times in RM-MEDA. From the above experimental results, we can conclude that although RM-MEDA is more time consuming than GDM3, it is still affordable in many applications, where the CPU time of each function evaluation is under half an hour and a cluster of (say, 100) computers is available. To significantly reduce the number of function evaluations in RM-MEDA and make it more applicable for MOPs with very expensive function evaluations, one may need to incorporate meta modelling techniques into RM-MEDA.

VII. CONCLUSION

It has not been well studied how to generate new trial solutions in multiobjective evolutionary optimization. Reproduction operators such as crossover and mutation, which were originally developed for scalar optimization, are directly used in most of current multiobjective evolutionary algorithms. This could be one of the major reasons why these algorithms did not perform well on MOPs with variable linkages. In this paper, the regularity property of continuous MOPs is used as a basis for an estimation of distribution algorithm for dealing with variable linkages. RM-MEDA, the proposed algorithm, models a promising area in the search space by a probability model whose centroid is a piecewise continuous manifold. The Local PCA algorithm was employed for building such a model. New trial solutions are sampled from the model thus built.

Experimental studies have shown that, overall, RM-MEDA performs better than GDE3, PCX-NSGA-II and MIDEA on a set of test instances with variable linkages. The sensitivity and scalability of RM-MEDA and GDE3 have also been experimentally studied.

We have found that RM-MEDA could perform slightly poorer than GDE3 on some test instances with linear variable linkages. We argued that it could be because RM-MEDA did not directly use the location information of individual solutions for generating new trial solutions. The experimental results also reveal that RM-MEDA may fail in test instances with many local Pareto fronts.

The future research topics along this line should include

- Combination of location information of individual solutions and globally statistical information for improving the ability of RM-MEDA to refine a solution. Guided mutation [42], [43] could be worthwhile studying for this purpose.
- Incorporating other techniques into RM-MEDA for improving its ability for global search. Effective global search techniques for scalar optimization should be considered.
- Combination of RM-MEDA with metamodelling techniques [44] for reducing the number of function evaluations.
- Use of other machine learning techniques, particularly, generative model learning for building distribution models such as mixtures of probabilistic principal component analyzers [45], for building models in RM-MEDA.
- Extension of RM-MEDA to constrained and/or dynamic MOPs. We should exploit properties of these MOPs in modifying or designing algorithms.

ACKNOWLEDGMENT

The authors would like to acknowledge the help of Mr H. Li, Dr. B. Sendhoff, and Professor E. Tsang on this work. They also thank X. Yao, the anonymous reviewers, and the anonymous associate editor for their insightful comments.

REFERENCES

[1] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*. Pittsburgh, PA, USA: Lawrence Erlbaum Associates, July 1985, pp. 93–100.

[2] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Baffins Lane, Chichester: John Wiley & Sons, LTD, 2001.

[3] C. A. Coello Coello, D. A. van Veldhuizen, and G. B. Lamont, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Kluwer Academic Publishers, 2002.

[4] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective Evolutionary Algorithms and Applications*. Springer-Verlag, 2005.

[5] J. Knowles and D. Corne, *Recent Advances in Memetic Algorithms*. Springer, 2004, ch. Memetic Algorithms for Multiobjective Optimization:Issues, Methods and Prospects, pp. 313–352.

[6] K. Deb, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation and Control*. Barcelona, Spain: CIMNE, 2002, pp. 95–100.

[8] C. A. Coello Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, June 2004.

[9] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[10] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched Pareto genetic algorithm for multiobjective optimization," in *Proceedings of the First IEEE Conference on Evolutionary Computation*, vol. 1. Piscataway, New Jersey: IEEE Service Center, 1994, pp. 82–87.

[11] J. D. Knowles and D. W. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 100–116, 2003.

[12] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[13] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 204–223, 2003.

[14] A. Jaszkiewicz, "Genetic local search for multiple objective combinatorial optimization," *European Journal of Operational Research*, vol. 137, no. 1, pp. 50–71, 2002.

[15] J. Knowles and D. Corne, "M-PAES: A memetic algorithm for multiobjective optimization," in *Proceedings of Congress on Evolutionary Computation (CEC 2000)*, vol. 1. Piscataway, New Jersey: IEEE Press, 2000, pp. 325–332.

[16] K. Deb, A. Sinha, and S. Kukkonen, "Multi-Objective Test Problems, Linkages, and Evolutionary Methodologies," in *2006 Genetic and Evolutionary Computation Conference (GECCO 2006)*, M. K. et al., Ed., vol. 2. Seattle, Washington, USA: ACM Press, July 2006, pp. 1141–1148.

[17] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. Kluwer's International Series in Operations Research & Management Science. Kluwer Academic Publishers, 1999, vol. 12.

[18] M. Ehrgott, *Multicriteria Optimization*, ser. Lecture Notes in Economics and Mathematical Systems. Springer, 2005, vol. 491.

[19] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich, "Covering Pareto sets by multilevel evolutionary subdivision techniques," in *Second International Conference on Evolutionary Multi-Criterion Optimization (EMO 2003)*, ser. Lecture Notes in Computer Science, vol. 2632. Faro, Portugal: Springer, April 2003, pp. 118–132.

[20] Y. Jin and B. Sendhoff, "Connectedness, regularity and the success of local search in evolutionary multi-objective optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC 2003)*. Canberra, Australia: IEEE Press, December 2003, pp. 1910–1917.

[21] Y. Jin, T. Okabe, and B. Sendhoff, "Adapting weighted aggregation for multiobjective evolution strategies," in *Proceedings of the First International Conference on Evolutionary Multi-criterion Optimization(EMO-2001)*, ser. Lecture Notes in Computer Science, vol. 1993. Zurich, Switzerland: Springer, March 2001, pp. 96–110.

[22] P. Larrañaga and J. A. Lozano, Eds., *Estimation of Distribution Algorithms : A New Tool for Evolutionary Computation*. Norwell, MA, USA: Kluwer Academic Publishers, 2001.

[23] T. Okabe, Y. Jin, B. Sendhoff, and M. Olhofer, "Voronoi-based estimation of distribution algorithm for multi-objective optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC 2004)*. Portland, Oregon, USA: IEEE Press, June 2004, pp. 1594–1601.

[24] P. A. N. Bosman and D. Thierens, "The naive MIDEA: A baseline multi-objective EA," in *Third International Conference on Evolutionary Multi-Criterion Optimization(EMO 2005)*, ser. Lecture Notes in Computer Science, vol. 3410. Guanajuato, Mexico: Springer, 2005, pp. 428–442.

[25] M. Pelikan, K. Sastry, and D. Goldberg, "Multiobjective hBOA, clustering, and scalability," Illinois Genetic Algorithms Laboratory (IlliGAL), Tech. Rep. 2005005, 2005.

[26] V. Cherkassky and F. Mulier, *Learning From Data: Concepts, Theory, and Methods*. John Wiler & Sons, 1998.

[27] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer-Verlag, 2001.

[28] A. Zhou, Q. Zhang, Y. Jin, E. Tsang, and T. Okabe, "A model-based evolutionary algorithm for bi-objective optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC 2005)*. Edinburgh, U.K: IEEE Press, September 2005, pp. 2568–2575.

[29] A. Zhou, Y. Jin, Q. Zhang, B. Sendhoff, and E. Tsang, "Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion," in *Proceedings of the Congress on Evolutionary Computation (CEC 2006)*. Vancouver, BC, Canada: IEEE Press, July 2006, pp. 3234–3241.

[30] K. Deb, A. Pratap, and T. Meyarivan, "Constrained test problems for multi-objective evolutionary optimization," KanGAL Report 2000002, 2000.

[31] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *Proceedings of the Congress on Evolutionary Computation (CEC 2005)*. Edinburgh, U.K: IEEE Press, September 2005, pp. 443–450.

[32] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff, "On test functions for evolutionary multi-objective optimization," in *Parallel Problem Solving from Nature (PPSN VIII)*, ser. Lecture Notes in Computer Science, vol. 3242. Birmingham, UK: Springer, September 2004, pp. 792–802.

[33] Q. Zhang, "On stability of fixed points of limit models of univariate marginal distribution algorithm and factorized distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 80–93, 2004.

[34] Q. Zhang and H. Mühlenbein, "On the convergence of a class of estimation of distribution algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 2, pp. 127–136, 2004.

[35] T. Hastie and W. Stuetzle, "Principal curves," *Journal of the American Statistical Association*, vol. 84, no. 406, pp. 502–516, June 1989.

[36] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural Computation*, vol. 9, no. 7, pp. 1493–1516, October 1997.

[37] K. Deb, A. Anand, and D. Joshi, "A computationally efficient evolutionary algorithm for real-parameter optimization," *Evolutionary Computation*, vol. 10, no. 4, pp. 371–395, 2002.

[38] M. Reyes Sierra and C. A. Coello Coello, "A study of fitness inheritance and approximation techniques for multi-objective particle swarm optimization," in *Proceedings of the Congress on Evolutionary Computation (CEC 2005)*. Edinburgh, U.K: IEEE Press, September 2005, pp. 65–72.

[39] D. A. van Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a Pareto front," in *Late Breaking Papers at the Genetic Programming Conference*. Madison, Wisconsin, USA: Stanford University Bookstore, July 1998, pp. 221–228.

[40] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable Test Problems for Evolutionary Multiobjective Optimization," in *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, A. Abraham, L. Jain, and R. Goldberg, Eds. USA: Springer, 2005, pp. 105–145.

[41] H. Li and Q. Zhang, "A multiobjective differential evolution based on decomposition for multiobjective optimization with variable linkages," in *International Conference on Parallel Problem Solving from Nature (PPSN IX)*, 2006, pp. 583–592.

[42] Q. Zhang, J. Sun, and E. Tsang, "Evolutionary algorithm with the guided mutation for the maximum clique problem," *IEEE Transactions on Evolutionary Computation*, vol. 9, no. 2, pp. 1–9, April 2005.

[43] Q. Zhang, J. Sun, G. Xiao, and E. Tsang, "Evolutionary algorithms refining a heuristic: Hyper-heuristic for shared-path protections in WDM networks under SRLG constraints," *IEEE Transactions on Systems, Man and Cybernetics, Part B*, 2006, accepted for publication.

[44] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2005.

[45] M. E. Tipping and C. M. Bishop, "Mixture of probabilistic principal component analysers," *Neural Computation*, vol. 11, no. 2, pp. 443–482, 1999.

TABLE I

TEST INSTANCES

Instance	Variables	Objectives	Characteristics
$F1$	$[0,1]^n$	$f_1(x) = x_1$	convex *PF*
		$f_2(x) = g(x)[1 - \sqrt{f_1(x)/g(x)}]$	linear variable linkage
		$g(x) = 1 + 9(\sum_{i=2}^{n}(x_i - x_1)^2)/(n-1)$	
$F2$	$[0,1]^n$	$f_1(x) = x_1$	concave *PF*
		$f_2(x) = g(x)[1 - (f_1(x)/g(x))^2]$	linear variable linkage
		$g(x) = 1 + 9(\sum_{i=2}^{n}(x_i - x_1)^2)/(n-1)$	
$F3$	$[0,1]^n$	$f_1(x) = 1 - exp(-4x_1)sin^6(6\pi x_1)$	concave *PF*
		$f_2(x) = g(x)[1 - (f_1(x)/g(x))^2]$	nonuniformly distributed
		$g(x) = 1 + 9[\sum_{i=2}^{n}(x_i - x_0)^2/9]^{0.25}$	linear variable linkage
$F4$	$[0,1]^n$	$f_1(x) = cos(\frac{\pi}{2}x_1)cos(\frac{\pi}{2}x_2)(1+g(x))$	concave *PF*
		$f_2(x) = cos(\frac{\pi}{2}x_1)sin(\frac{\pi}{2}x_2)(1+g(x))$	linear variable linkage
		$f_3(x) = sin(\frac{\pi}{2}x_1)(1+g(x))$	3 objectives
		$g(x) = \sum_{i=3}^{n}(x_i - x_1)^2$	
$F5$	$[0,1]^n$	$f_1(x) = x_1$	convex *PF*
		$f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$	nonlinear variable linkage
		$g(x) = 1 + 9(\sum_{i=2}^{n}(x_i^2 - x_1)^2)/(n-1)$	
$F6$	$[0,1]^n$	$f_1(x) = \sqrt{x_1}$	concave *PF*
		$f_2(x) = g(x)[1 - (f_1(x)/g(x))^2]$	nonlinear variable linkage
		$g(x) = 1 + 9(\sum_{i=2}^{n}(x_i^2 - x_1)^2)/(n-1)$	
$F7$	$[0,1]^n$	$f_1(x) = 1 - exp(-4x_1)sin^6(6\pi x_1)$	concave *PF*
		$f_2(x) = g(x)[1 - (f_1(x)/g(x))^2]$	nonuniformly distributed
		$g(x) = 1 + 9[\sum_{i=2}^{n}(x_i^2 - x_0)^2/9]^{0.25}$	nonlinear variable linkage
$F8$	$[0,1]^n$	$f_1(x) = cos(\frac{\pi}{2}x_1)cos(\frac{\pi}{2}x_2)(1+g(x))$	concave *PF*
		$f_2(x) = cos(\frac{\pi}{2}x_1)sin(\frac{\pi}{2}x_2)(1+g(x))$	nonlinear variable linkage
		$f_3(x) = sin(\frac{\pi}{2}x_1)(1+g(x))$	3 objectives
		$g(x) = \sum_{i=3}^{n}(x_i^2 - x_1)^2$	
$F9$	$[0,1]\times$	$f_1(x) = x_1$	concave *PF*
	$[0,10]^{n-1}$	$f_2(x) = g(x)[1 - \sqrt{f_1(x)/g(x)}]$	nonlinear variable linkage
		$g(x) = \frac{1}{4000}\sum_{i=2}^{n}(x_i^2 - x_1)^2 - \prod_{i=2}^{n}cos(\frac{(x_i^2 - x_1)}{\sqrt{i-1}}) + 2$	multimodal with Griewank function
$F10$	$[0,1]\times$	$f_1(x) = x_1$	concave *PF*
	$[0,10]^{n-1}$	$f_2(x) = g(x)[1 - \sqrt{f_1(x)/g(x)}]$	nonlinear variable linkage
		$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}[(x_i^2 - x_1)^2 - 10cos(2\pi(x_i^2 - x_1))]$	multimodal with Rastrigin function

TABLE II

THE PARAMETER SETTINGS OF RM-MEDA FOR F5 IN SECTION VI

	Population Size	K in Local PCA	# of Decision Variables	Max. # of $\vec{F}$ Evaluations
Section VI.A	$20 \sim 400$	5	50	20,000
Section VI.B	100	$1 \sim 15$	50	20,000
Section VI.C	100	5	$20 \sim 100$	20,000
Section VI.D	100	5	$20 \sim 100$	20,000

TABLE III

THE PARAMETER SETTINGS OF RM-MEDA FOR F5 IN SECTION VI

	Population Size	CR	F	# of Decision Variables	Max. # of $\vec{F}$ Evaluations
Section VI.A	$20 \sim 400$	1	1	50	20,000
Section VI.C	100	1	1	$20 \sim 100$	20,000
Section VI.D	100	1	1	$20 \sim 100$	20,000

TABLE IV

THE CPU TIME (IN SECONDS) USED BY RM-MEDA AND GDE3 FOR $F5$ WITH DIFFERENT NUMBERS OF DECISION VARIABLES

	The number of decision variables						
Method	20	30	40	50	60	80	100
GDE3	0.75	0.79	0.75	0.78	0.82	0.79	0.78
RM-MEDA	8.09	16.65	28.92	45.73	69.00	136.04	238.28

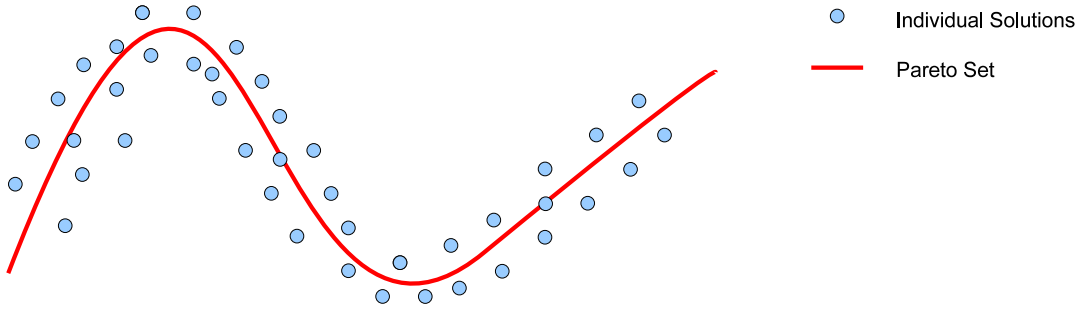

Fig. 1. Illustration of the basic idea. Individual solutions should be scattered around the *PS* in the decision space in a successful MOEA.

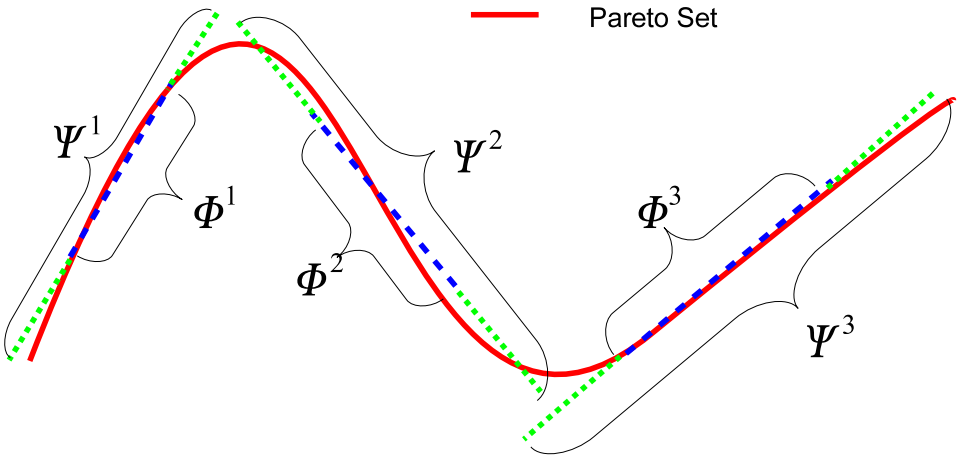

Fig. 2. Illustration of Extension: The number of clusters (K) is 3 in this illustrative example. Φ^1, Φ^2 and Φ^3 could not approximate the *PS* very well, their extensions Ψ^1, Ψ^2 and Ψ^3, however, can provide a better approximation.

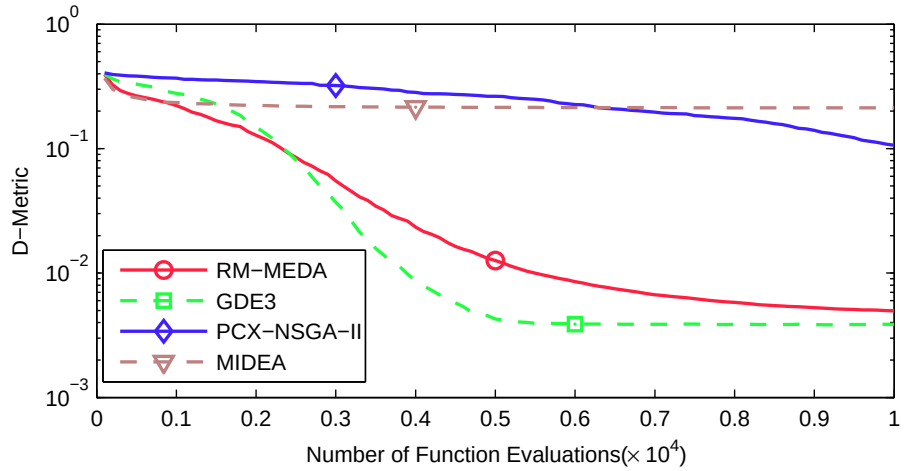

Fig. 3. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of 10,000-function evaluations in four algorithms for F1.

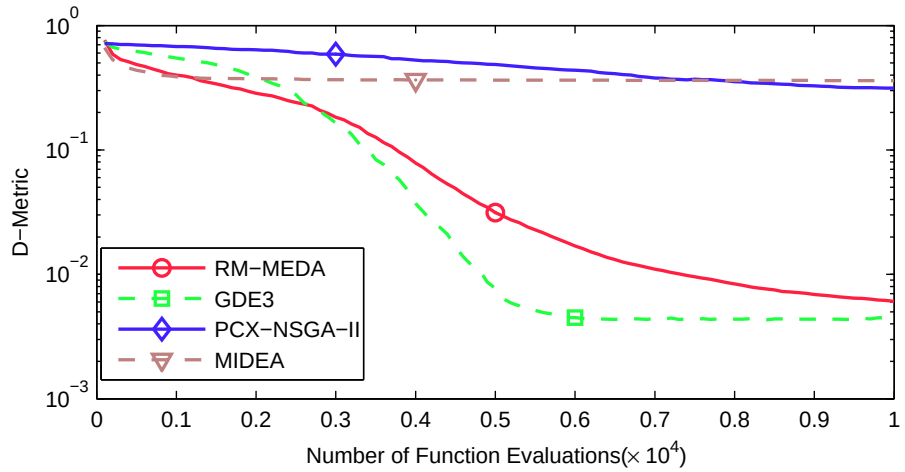

Fig. 4. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of 10,000-function evaluations in four algorithms for F2.

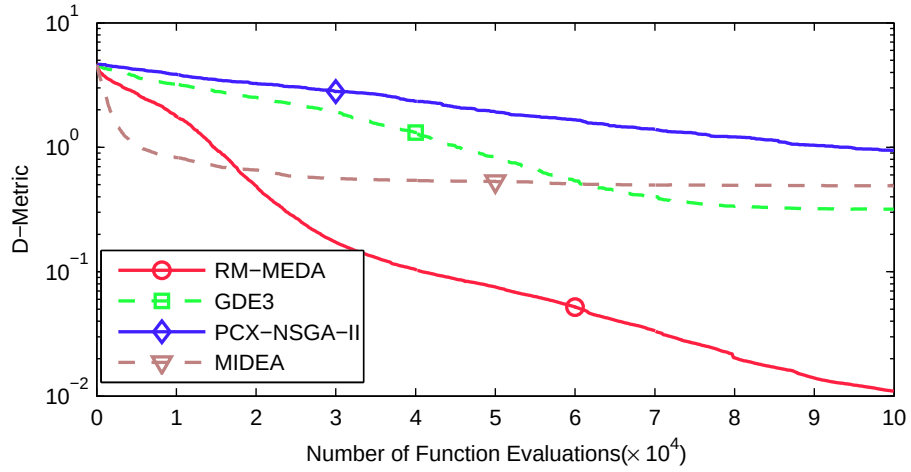

Fig. 5. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of $100,000$-function evaluations in four algorithms for F3.

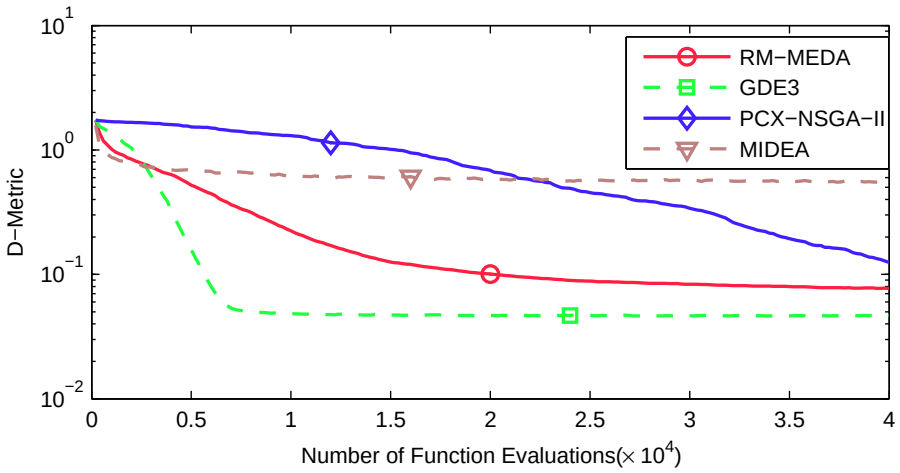

Fig. 6. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of 40,000-function evaluations in four algorithms for F4.

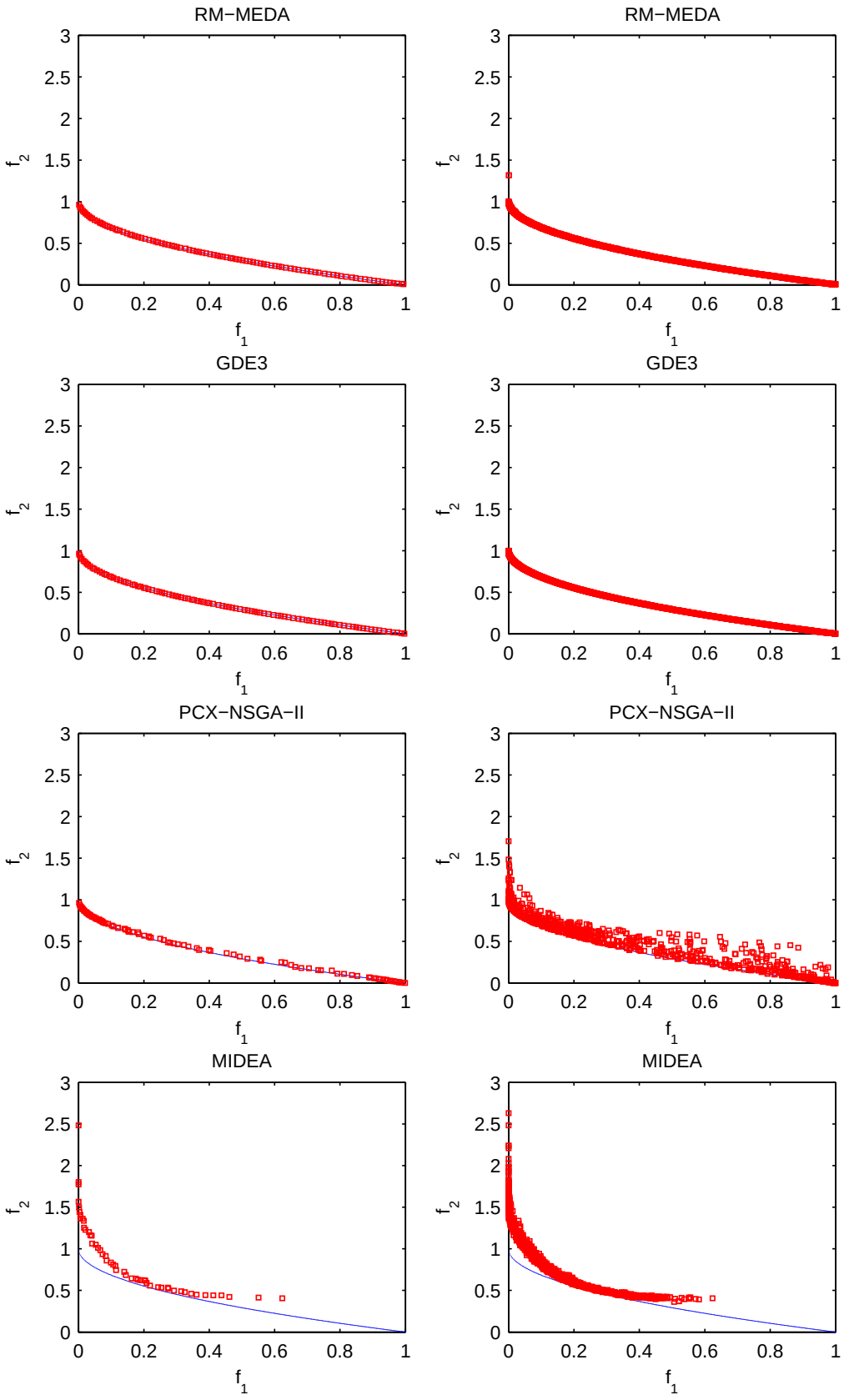

Fig. 7. The final nondominated fronts found by each algorithm on F1. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

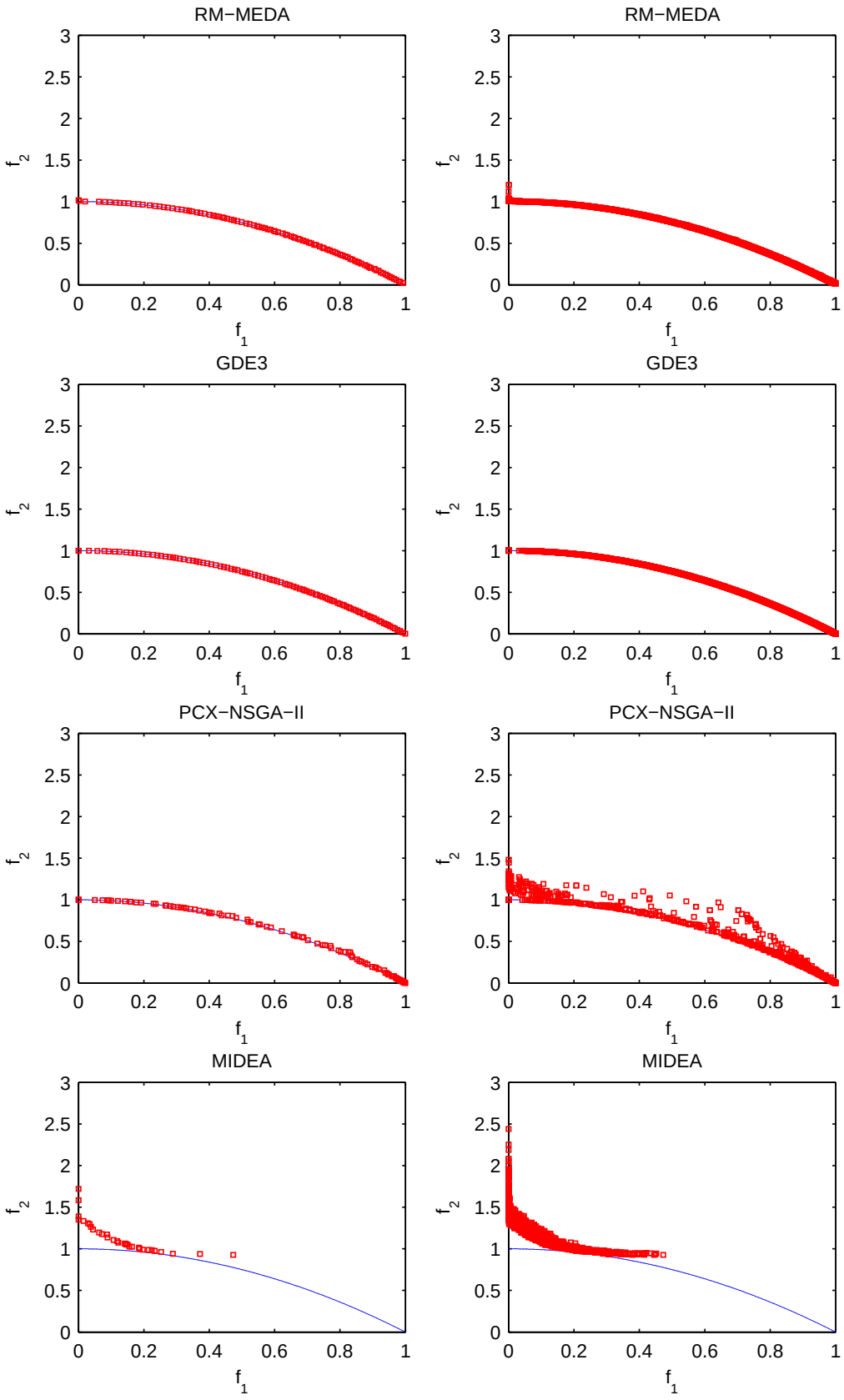

Fig. 8. The final nondominated fronts found by each algorithm on F2. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

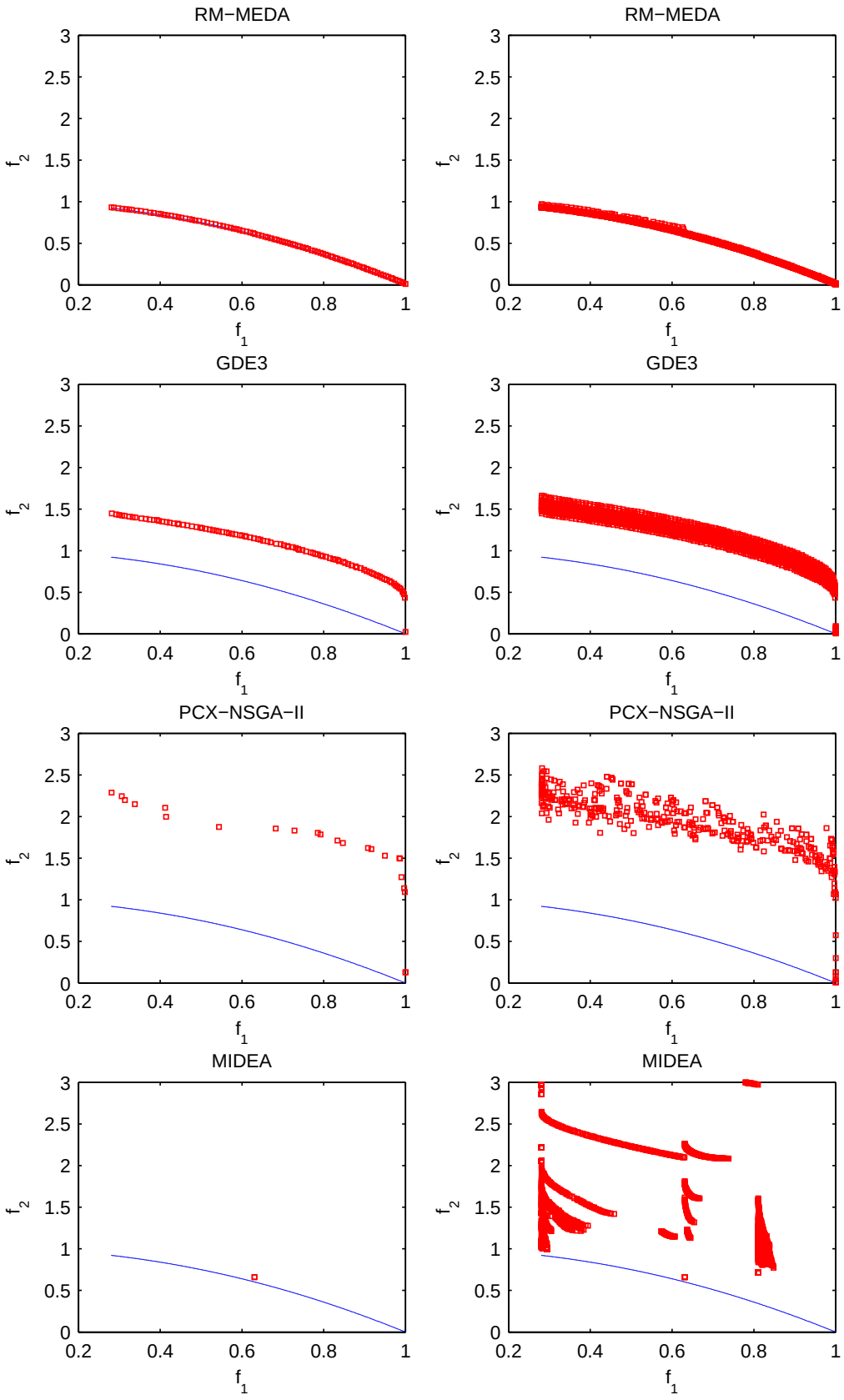

Fig. 9. The final nondominated fronts found by each algorithm on F3. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

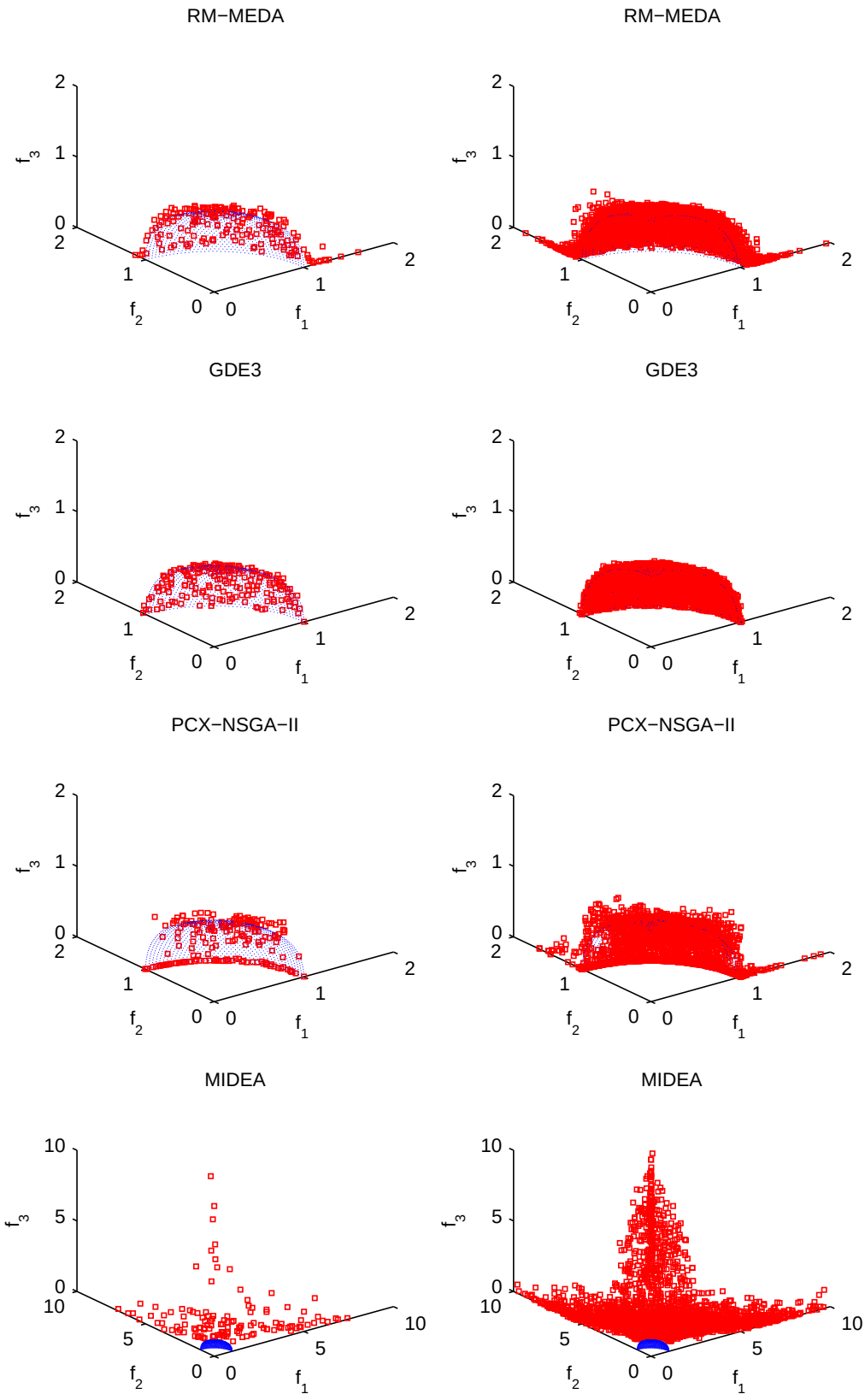

Fig. 10. The final nondominated fronts found by each algorithm on F4. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

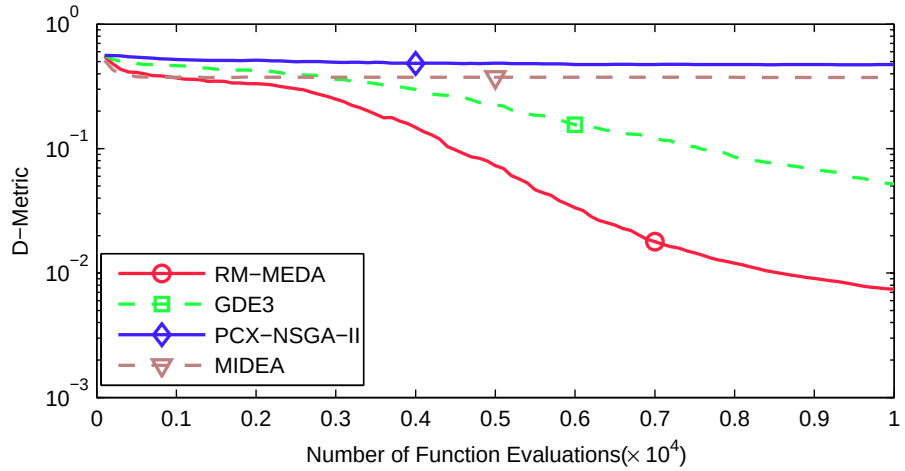

Fig. 11. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of 10,000-function evaluations in four algorithms on F5.

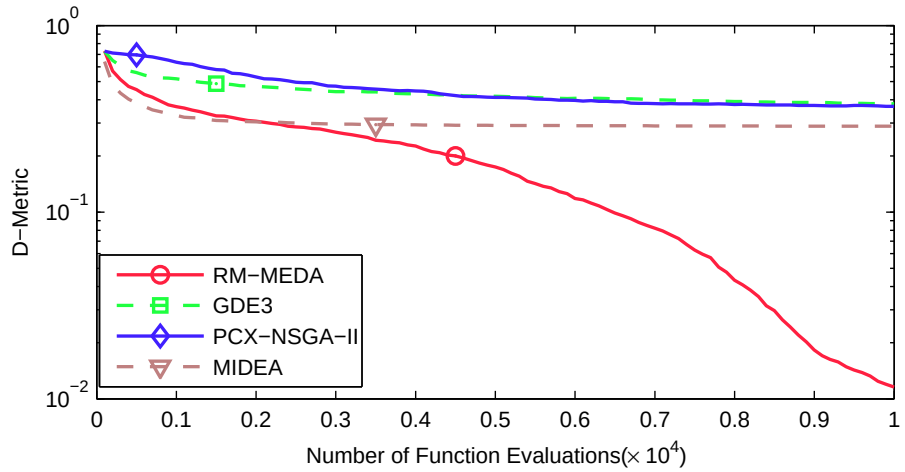

Fig. 12. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of 10,000-function evaluations in four algorithms on F6.

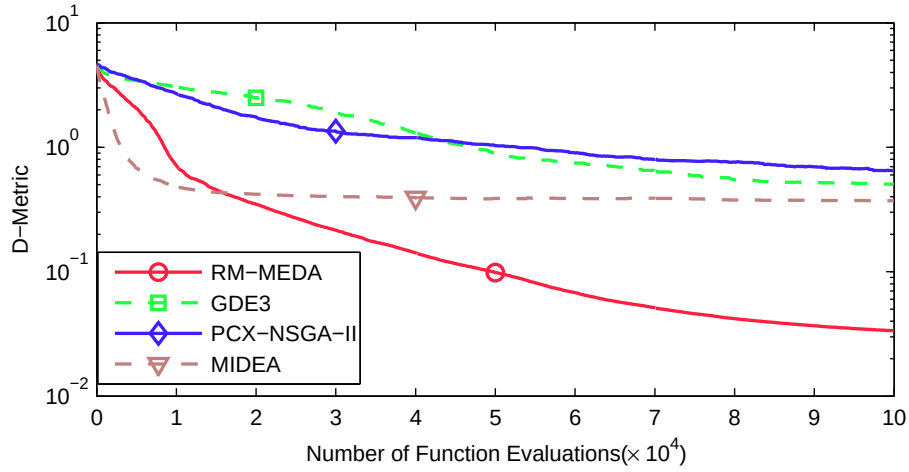

Fig. 13. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of 100,000-function evaluations in four algorithms on F7.

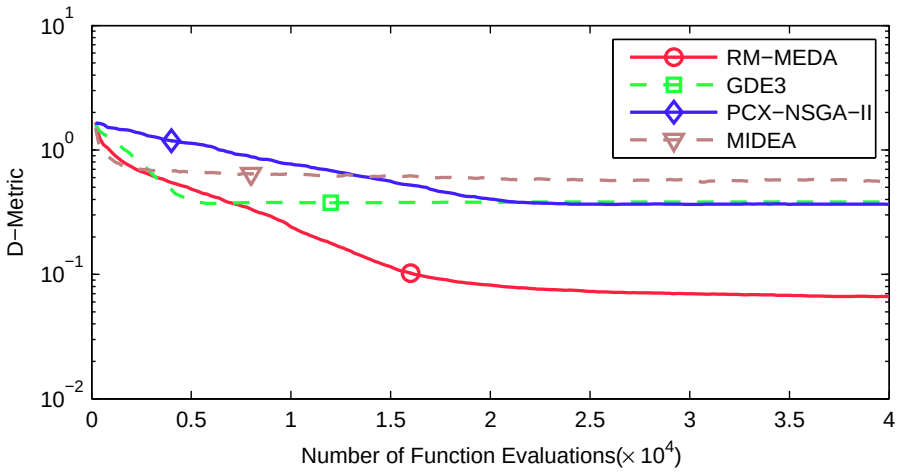

Fig. 14. The evolution of the average D-metric of the nondominated solutions in the current populations among 20 independent runs with the number of 40,000-function evaluations in four algorithms on F8.

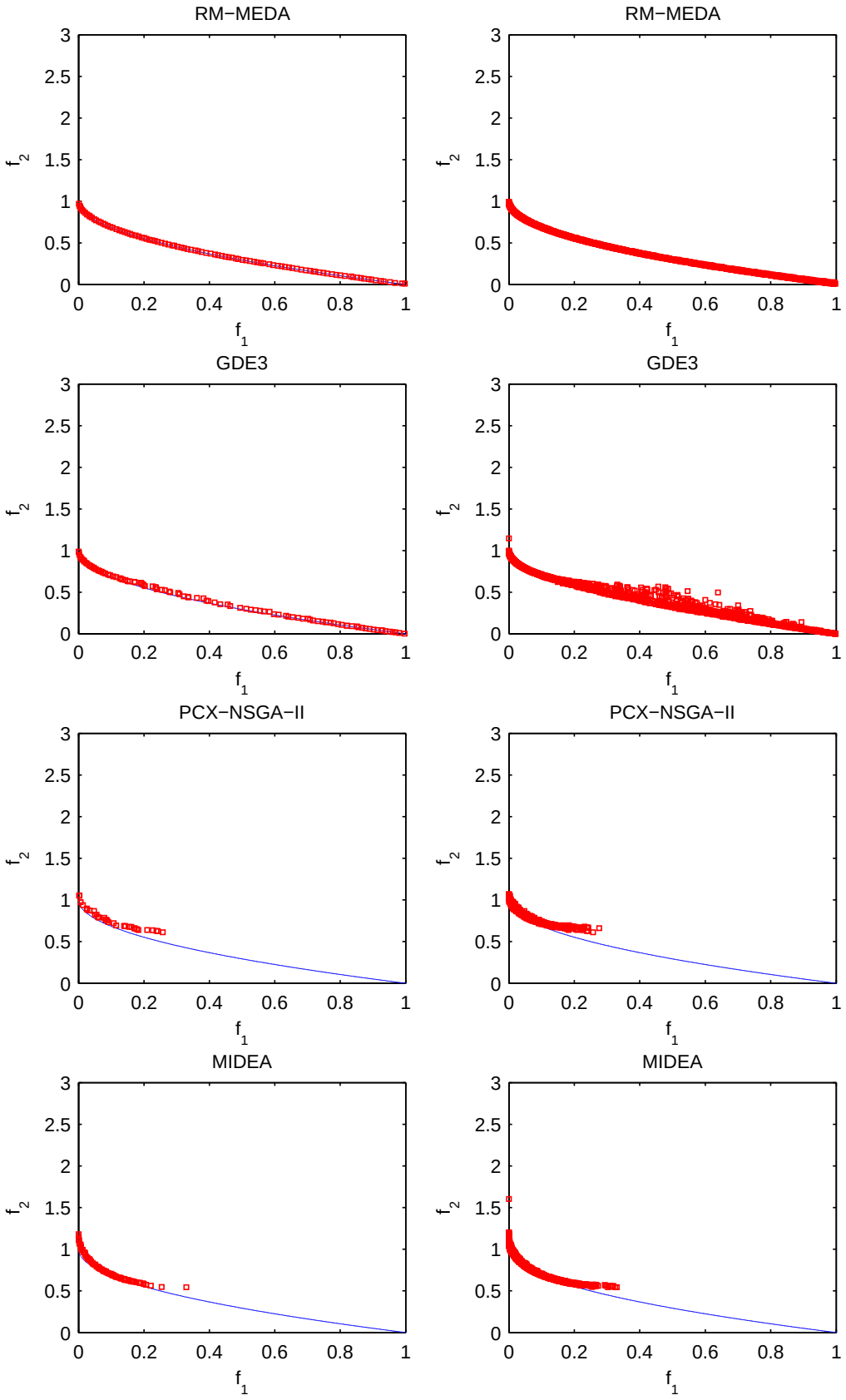

Fig. 15. The final nondominated fronts found by each algorithm on F5. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

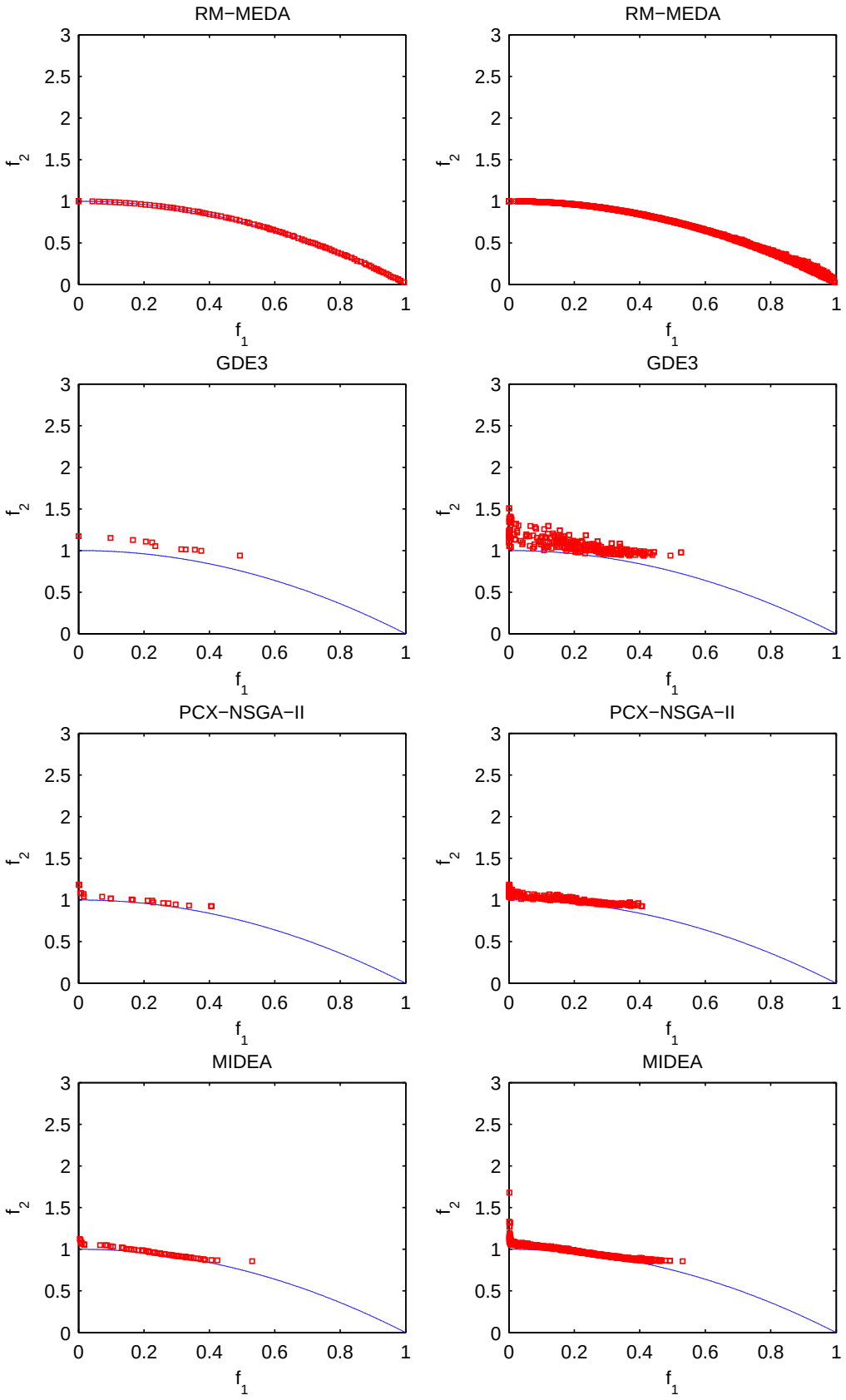

Fig. 16. The final nondominated fronts found by each algorithm on F6. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

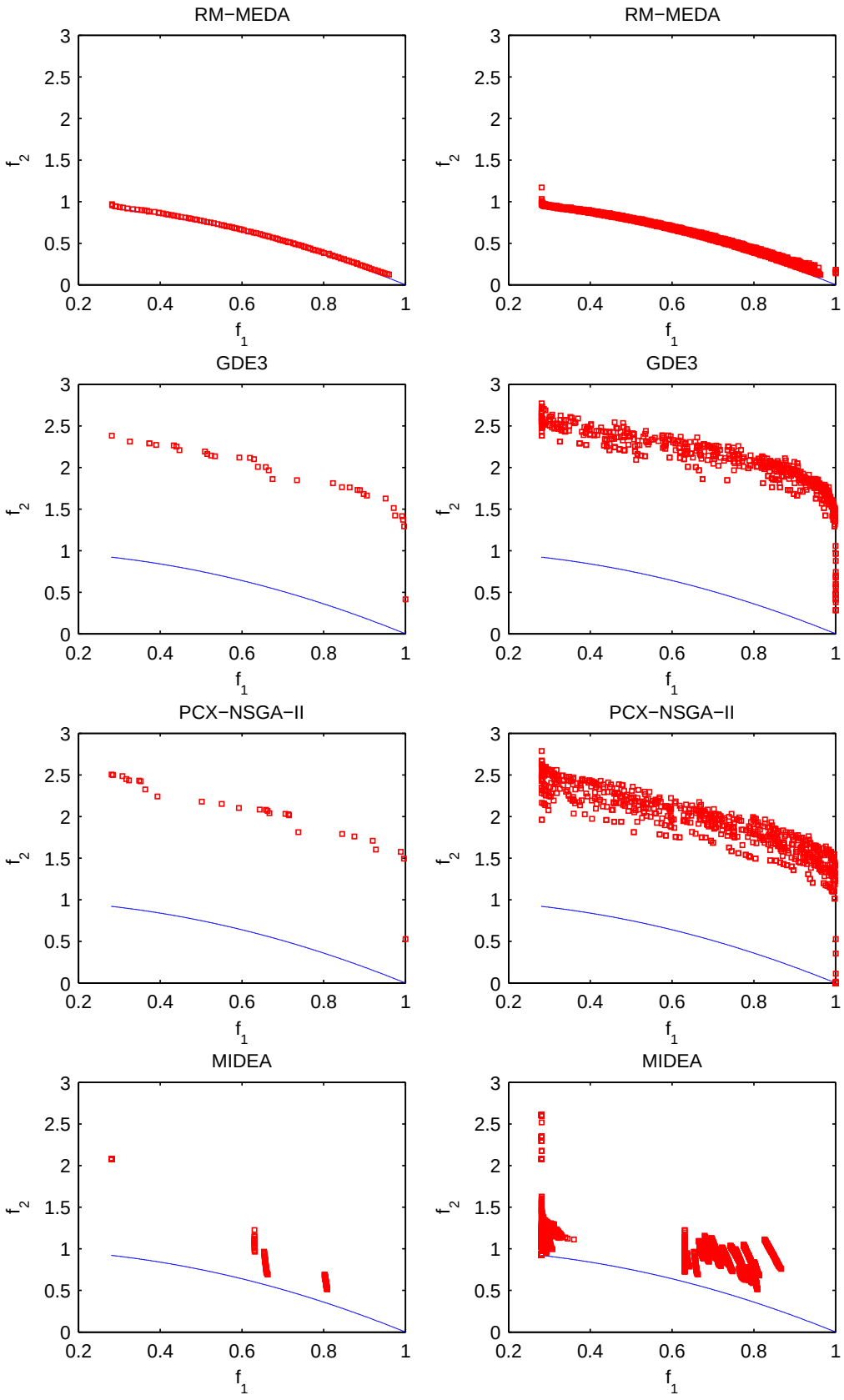

Fig. 17. The final nondominated fronts found by each algorithm on F7. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

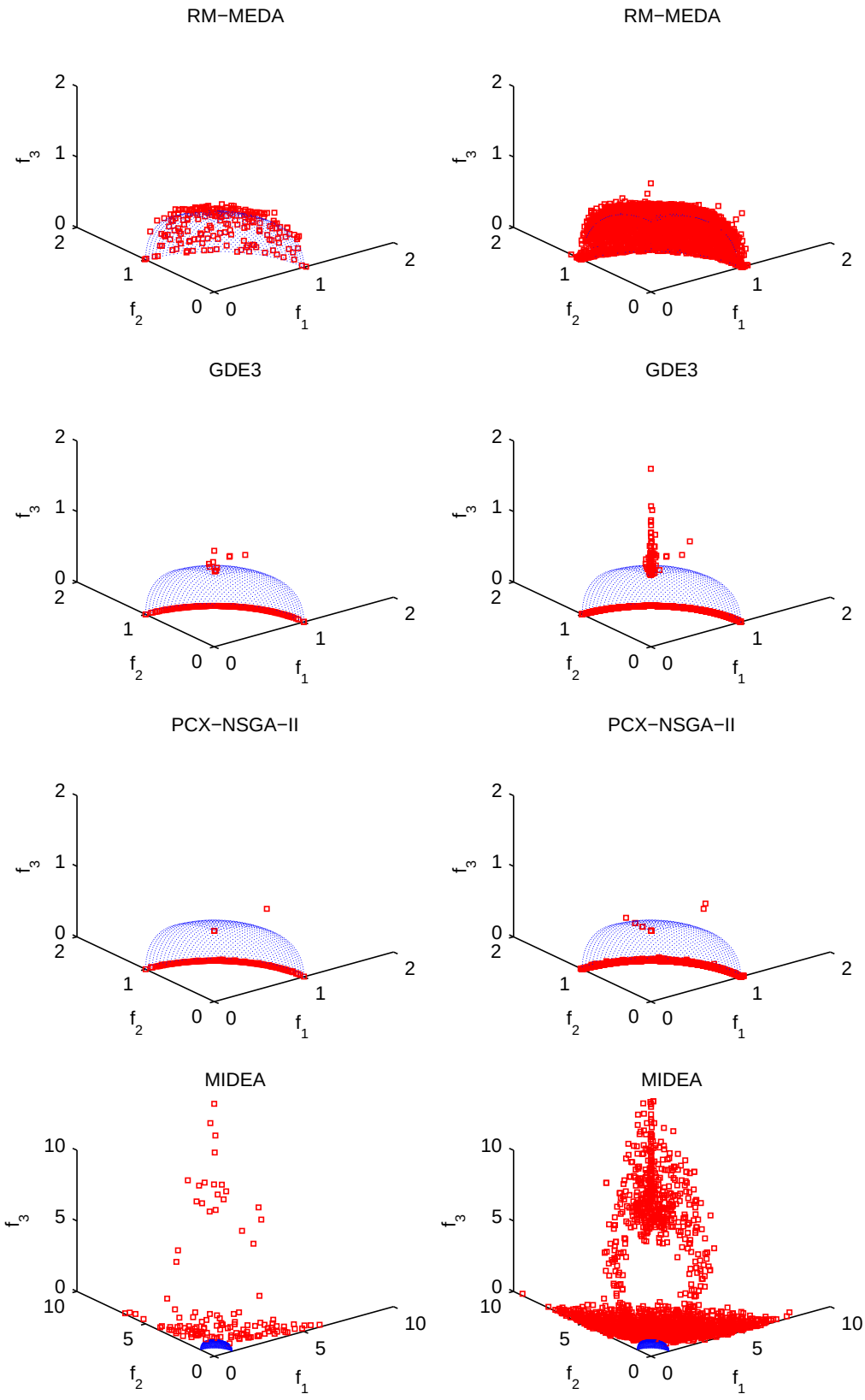

Fig. 18. The final nondominated fronts found by each algorithm on F8. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

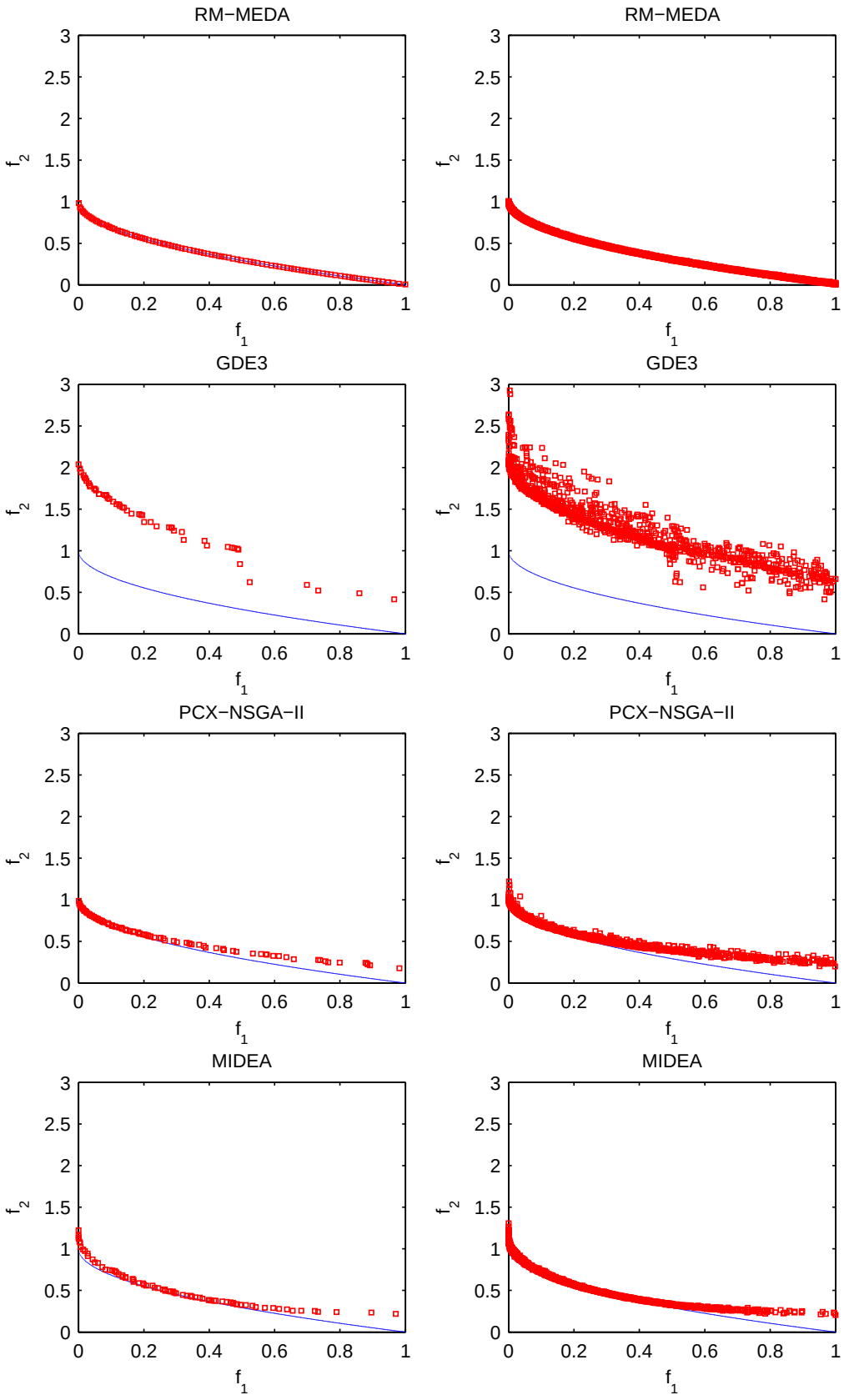

Fig. 19. The final nondominated fronts found by each algorithm on F9. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

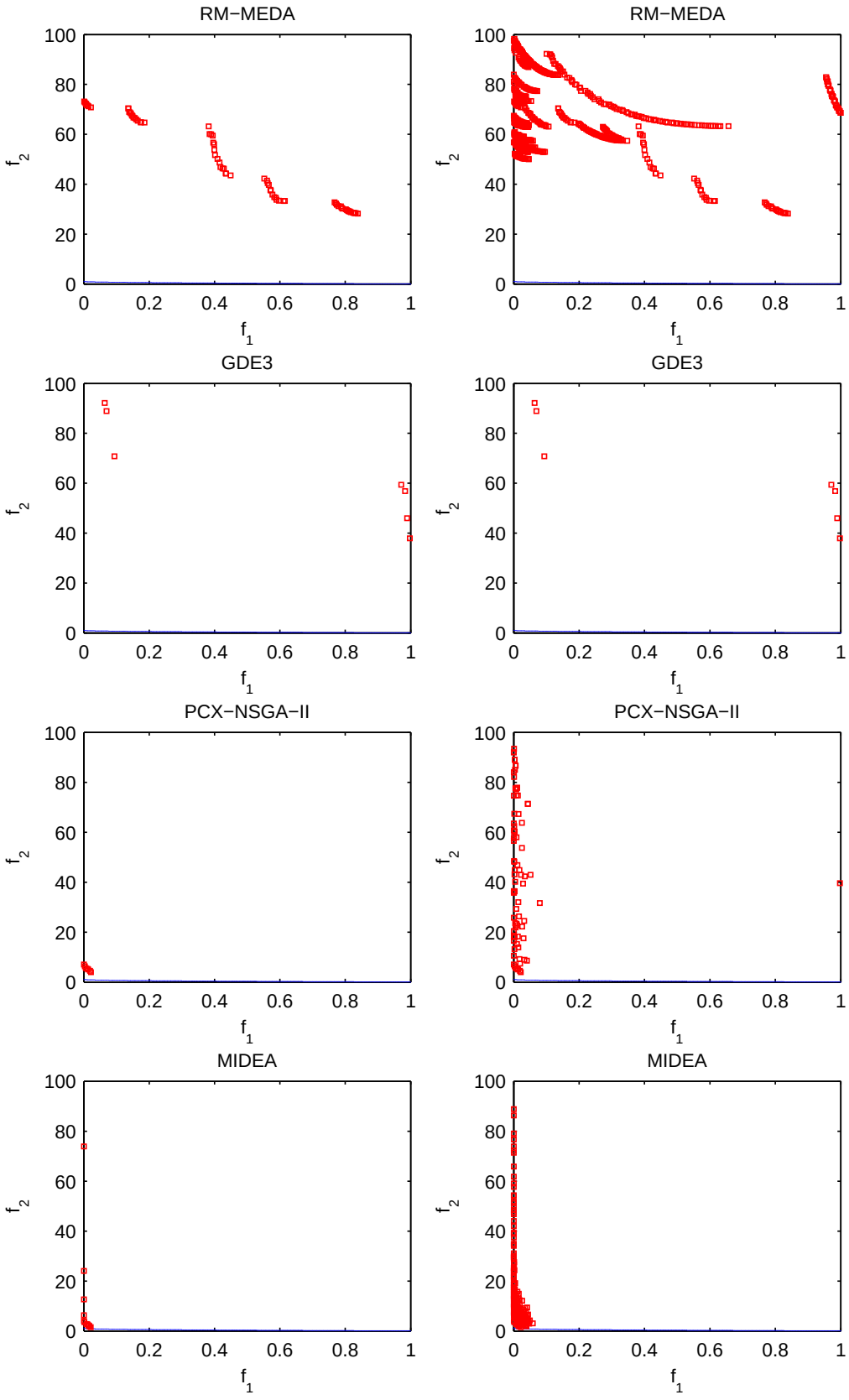

Fig. 20. The final nondominated fronts found by each algorithm on F10. The left panels show the nondominated fronts with the lowest D-metric obtained by each algorithm, while the right panels plot all the 20 fronts together found by each algorithm.

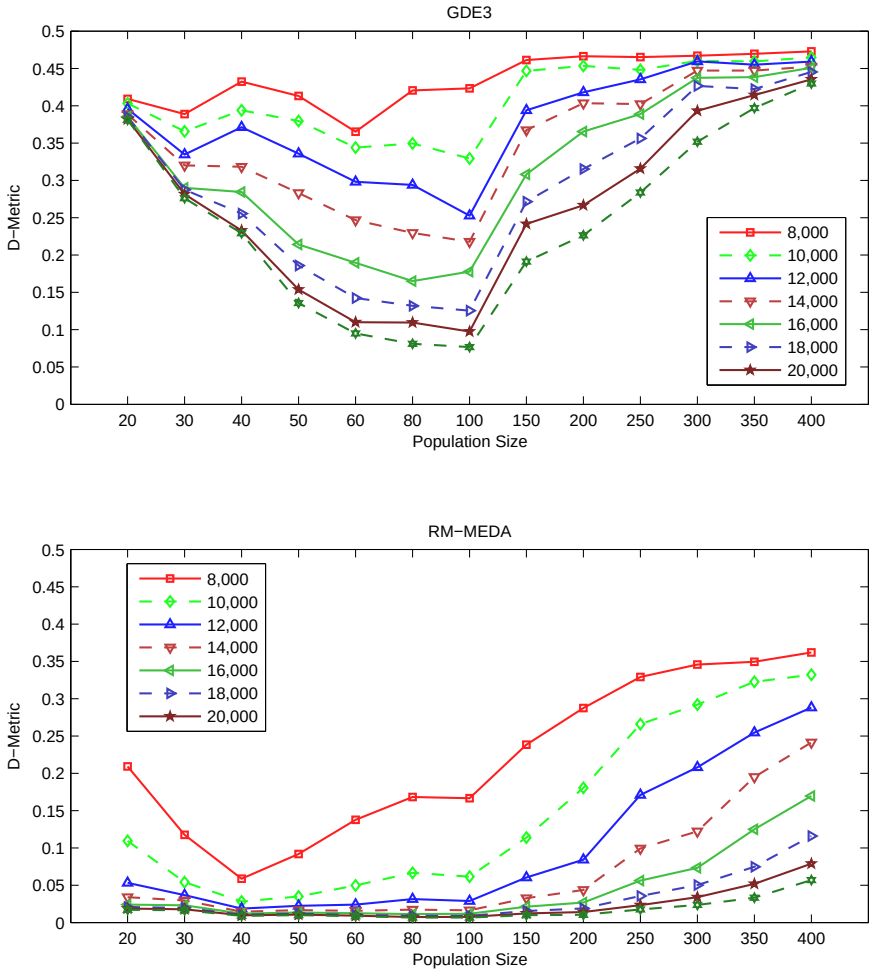

Fig. 21. The average D-metric value v.s. the number of $\vec{F}$-function evaluations under the different settings of population sizes in RM-MEDA and GDE3 for F5.

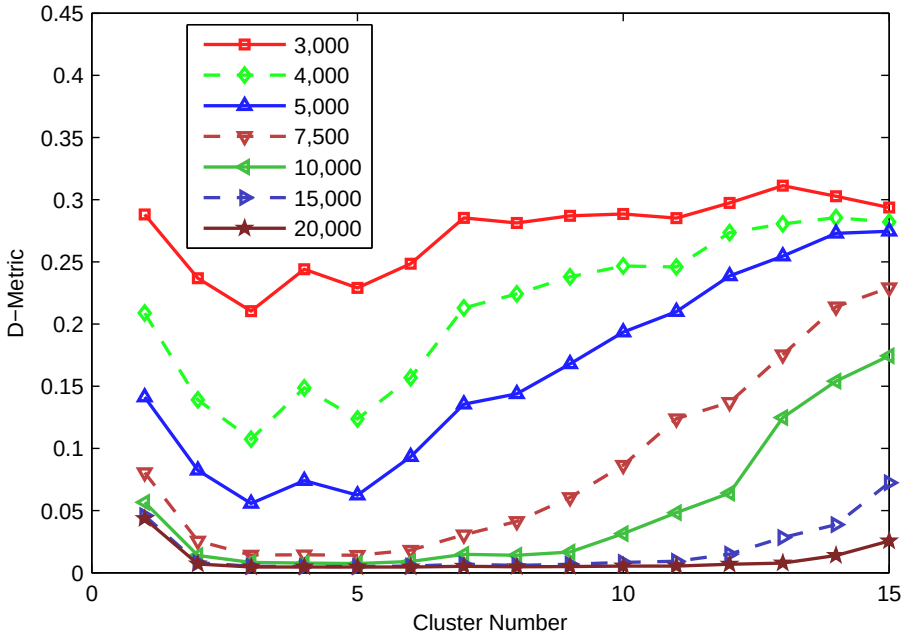

Fig. 22. The average D-metric value v.s. the numbers of $\vec{F}$-function evaluations with different cluster numbers in RM-MEDA for F5.

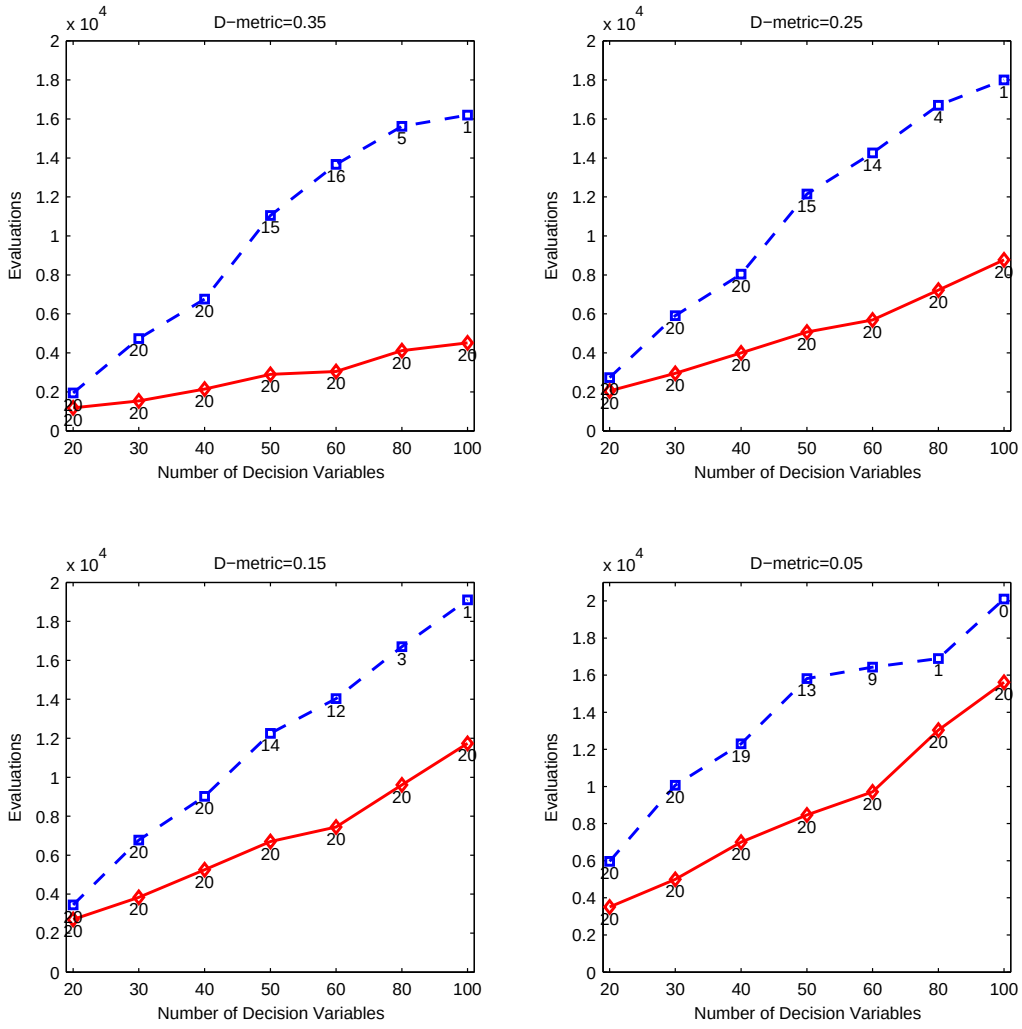

Fig. 23. The average number of function evaluations among the successful runs for reaching each of four given levels of D-metric v.s. the number of decision variables in two algorithms. The marked number is the number of successful runs. The solid line is for RM-MEDA while dash line for GDE3.