Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model

Qingfu Zhang, Wudong Liu, Edward Tsang and Botond Virginas

Technical Report CES-489

Feb., 2009

The School of Computer Science & Electronic Engineering, University of Essex, Colchester, CO4, 3SQ, UK.

Abstract

In some expensive multiobjective optimization problems, several function evaluations can be carried out at one time. Therefore, it is very desirable to develop methods which can generate several test points simultaneously. This paper proposes such a method, called MOEA/D-EGO, for dealing with expensive multiobjective optimization. MOEA/D-EGO decomposes a MOP in question into a number of single objective optimization subproblems. A predictive model is built for each subproblem based on the points already evaluated. Effort has been made to save the overhead for modeling and to improve the prediction quality. At each generation, MOEA/D is used for maximizing the expected improvement metric values of all the subproblems and then several test points are selected for evaluation. Experimental results on a number of test instances have shown that MOEA/D-EGO is very promising.

Index Terms

multiobjective optimization, expensive optimization, evolutionary algorithm, Pareto optimality.

Q. Zhang W. Liu, and E. Tsang are with the School of Computer Science and Electronic Engineering, University of Essex, Wivenhoe Park, Colchester, CO4 3SQ, U.K. (e-mail: {qzhang,wliui,edward}@essex.ac.uk).

B. Virginas is with BT Exact, Adastral Park, Ipsiwich, IP5 3RE, U.K. (e-mail: botond.viginas@bt.com)

I. Introduction

In some engineering optimization problems, the evaluation of candidate solutions could be extremely computationally and/or financially expensive since it requires time-consuming computer simulation or expensive physical experiments. Therefore, any methods, which are able to produce reasonably good solutions within a given (often very tight) budget on computational cost/time, are of great practical interest. This paper aims at developing such a method for approximating the Pareto front of an expensive multiobjective optimization problem (MOP).

Since the publication of the seminal work of Jones *et al* on the EGO in 1998 [1], the Gaussian stochastic process model (also called Kriging [2] or DACE stochastic process model [3]) has been widely accepted as one of the most efficient methods for dealing with an expensive single-objective optimization problem [4], [5]. It assumes that the objective function is a sample of a Gaussian stochastic process. The distribution of the objective function value at any untested point can be estimated based on data collected during the previous search, then a figure of merit for evaluation of the objective function at a new point, such as the expected improvement [6] and the probability of improvement [7], could be defined for determining which point should be evaluated next. Much effort has been made to incorporate the Gaussian stochastic process and other response surface methods [4], [8], [9] into evolutionary algorithm (EA) frameworks for single objective optimization [10]–[17].

In the case of multiple objectives, the goal is to find a number of well-representative Pareto optimal solutions. A few attempts have been made to introduce the Gaussian process model into evolutionary multiobjective optimization [18]–[24]. Keane [21] and Emmerich *et al* [23] have generalized the probability of improvement and the expected improvement to multiobjective optimization. All these generalized metrics of a untested point are based on the likelihood that it could not be dominated by the non-dominated solutions found so far. Single objective EAs are used for maximizing one of these metrics for determining which point should be evaluated next. By nature, these scalar metrics on their own are unable to predict the most possible location of the whole Pareto front and thus could not generate multiple candidate solutions at one single iteration, which makes them not very suitable on parallel computing environments.

The ParEGO, proposed by Knowles [24], applies the EGO algorithm to a randomly-selected aggregation function for finding a point to evaluate next in the search space. Its performance on a set of test instances with 150-200 function evaluations is very promising. The major drawback of the ParEGO is that it just considers one aggregation function at each iteration and therefore its criterion for locating a new point to evaluate may be appropriate for optimization of the selected aggregation function but not necessarily optimal for the multiobjective problem itself. ParEGO is not able to generate multiple candidate points at one iteration either. In [20], Jeong and Obayashi used NSGA-II [25] to optimize the expected improvements of all the individual objectives in a MOP and locate candidate solutions to evaluate next. They tested their method on two real-life problems. In principle this method can generate several candidate points to evaluate at one iteration; it, however, does not make the full use of statistical information provided by the Gaussian process model and thus it is not very efficient in dealing with expensive MOPs as shown in [26].

Naturally, parallel computing is a basic tool for solving an expensive MOP, particularly when function evaluation only involves computer simulation as in many engineering design problems. Parallel computing could be able to carry out several function evaluations at the same time. Therefore, it is very desirable to develop and study methods which could generate several test points simultaneously at each iteration. This paper represents an attempt along this line.

MOEA/D (multiobjective evolutionary algorithm based on decomposition) [27] is a recent multiobjective evolutionary algorithm framework. Like MOGLS [28]–[30] and MSOPS [31], it is based on conventional aggregation approaches. MOEA/D decomposes a MOP into a number of single objective optimization subproblems. The objective of each subproblem is a (linear or nonlinear) weighted aggregation of all the individual objectives in the MOP. Neighborhood relations among these subproblems are defined based on the distances among their aggregation weight vectors. Each subproblem is optimized in MOEA/D by using information mainly from its neighboring subproblems. The experimental studies have shown that MOEA/D performs very well on a number of multiobjective problems [27], [32]–[35].

In this paper, we propose MOEA/D-EGO, MOEA/D with the Gaussian stochastic process model for expensive multiobjective optimization. At each iteration in MOEA/D-EGO, a Gaussian process model for each subproblem is built based on data obtained from the previous search, and the expected improvements (or any other metrics) of these subproblems are optimized simultaneously by using MOEA/D for generating a set of candidate test points. Then a few of them are selected for evaluation. The major features of MOEA/D-EGO include:

- Due to the parallel nature of MOEA/D, MOEA/D-EGO is able to generate several test points to evaluate, which makes it suitable in parallel computing environment.
- Since MOEA/D works with a number of single optimization subproblems, any metrics for measuring the merit for function evaluation at a new point, developed for single objective optimization such as the expected improvement and the probability of improvement, could be readily used in MOEA/D-EGO.
- A fuzzy clustering based modeling method is used. It could improve the prediction quality without significantly increasing the computational cost.
- At each iteration, a predictive distribution model is first built for each individual objective in the MOP, the predictive model for the objective in each subproblem is then induced. In such a way, the overhead for modeling is reduced significantly.

The remainder of this paper is organized as following. Section II introduces multiobjective optimization problems. Section III presents the algorithm framework. Section IV-VI gives the details of the algorithm. Section VII presents experimental results. Finally, Section VIII concludes the paper.

II. Multiobjective Optimization Problem

This paper considers the following continuous multiobjective optimization problem (MOP):

$$\begin{array}{ll} \text{minimize} & F(x) = (f_1(x), \ldots, f_m(x))^T \\ \text{subject to} & x = (x_1, \ldots, x_n)^T \in \prod_{i=1}^{n}[a_i, b_i] \end{array} \tag{1}$$

where $-\infty < a_i < b_i < +\infty$ for all $i = 1, \ldots, n$. $F : \prod_{i=1}^{n}[a_i, b_i] \rightarrow R^m$ consists of m individual objective functions f_i, $i = 1, \ldots, m$. All the individual objectives are continuous. $\prod_{i=1}^{n}[a_i, b_i]$ is called the decision space (or search space) and R^m is the objective space.

Let $u = (u_1, \ldots, u_m)^T$, $v = (v_1, \ldots, v_m)^T \in R^m$ be two vectors, u is said to dominate v if $u_i \leq v_i$ for all $i = 1, \ldots, m$, and $u \neq v$. A point $x^\star \in \prod_{i=1}^{n}[a_i, b_i]$ is called (globally) Pareto optimal if there is no $x \in \prod_{i=1}^{n}[a_i, b_i]$ such that $F(x)$ dominates $F(x^\star)$. 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*$= \{F(x) \in R^m | x \in$ *PS*$\}$, is called the Pareto front [36].

In this paper, we assume that the evaluation of $F(x)$ is very expensive and only a very few F-function evaluations can be carried out in parallel. If an iterative algorithm performs a few function evaluations in parallel at each generation (i.e. iteration) and its other computational overhead can be negligible compared with its F-function evaluations, then its overall cost can be measured by the number of its generations. The purpose of our proposed method in this paper is to reduce it as much as possible.

III. Algorithm Framework

A. MOP Decomposition

Several approaches have been proposed for decomposing a MOP into a number of single objective optimization subproblems [36]–[39]. In the following, we introduce two most commonly-used approaches.

1) Weighted Sum Approach: Let $\lambda = (\lambda_1, \ldots, \lambda_m)^T$ be a weight vector, i. e., $\sum_{i=1}^{m} \lambda_i = 1$ and $\lambda_i \geq 0$ for all $i = 1, \ldots, m$. Then the optimal solutions to the following single optimization problems:

$$\begin{array}{ll} \text{minimize} & g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x) \\ \text{subject to} & x \in \prod_{i=1}^{n}[a_i, b_i] \end{array} \tag{2}$$

are Pareto optimal to (1) if the PF of (1) is convex, where we use $g^{ws}(x|\lambda)$ to emphasize that λ is a weight vector in this objective function while x is the variable vector to optimize. However, when the PS is not convex, the weighted sum approach could be not able to find some Pareto optimal solutions.

2) The Tchebycheff Approach [36]: In this approach, the single objective functions to be minimized are in the form

$$g^{te}(x|\lambda) = \max_{1 \leq i \leq m} \{\lambda_i (f_i(x) - z_i^*)\} \tag{3}$$

where $z^* = (z_1^*, \ldots, z_m^*)$ is the reference point, i. e.

$$z_i^* = \min_{x \in \prod_{i=1}^{n}[a_i, b_i]} f_i(x) \tag{4}$$

for each $i = 1, \ldots, m$. Under some mild condition, for each Pareto optimal point x^* there exists a weight vector λ such that x^* is the optimal solution of (3) and each optimal solution of (3) is a Pareto optimal to (1). This approach can deal with nonconvex PFs but its objective function is not smooth at some points. However, it can still be used in the evolutionary algorithm framework proposed in this paper since our algorithm does not require the derivatives of the objective functions.

To obtain a set of different Pareto optimal solutions of (1), one can solve a set of single objective optimization problems defined by (2), (3) or any other decompostion methods with different weight vectors.

B. Framework

To solve the expensive MOP (1), MOEA/D with the Gaussian process model (MOEA/D-EGO), proposed in this paper, chooses a set of N weight vectors $\lambda^1, \ldots, \lambda^N$ and converts approximation of the PF of (1) into N single objective optimization problems by a decomposition method. It attempts to solve these N subproblems simultaneously. Suppose that the i-th subproblem is associated with weight vector λ^i and its objective function is $g(x|\lambda^i)$, MOEA/D-EGO works as follows:

Algorithmic Parameters:

- $\lambda^1, \ldots, \lambda^N$: N weight vectors;
- K_I: the number of initial points in Initialization; and

- K_E: the number of function evaluations at each generation.

Step 1 Initialization: Generate K_I points $x^1, \ldots, x^{K_I}$ from the search space $\prod_{i=1}^{n}[a_i, b_i]$ by using an experimental design method and evaluate the F-function values of these K_I points. Set $P_{eval} = \{x^1, \ldots, x^{K_I}\}$.

Step 2 Stopping Condition: If the preset stopping condition is met, output the F-function values of all the nondominated solutions in P_{eval} as an approximation to the PF and stop.

Step 3 Models Building: By using the F-function values of the points in P_{eval}, build a predictive distribution model for each objective $g(x|\lambda^i)$ and then use it to define $\xi^i(x)$, a metric measuring the merit of evaluating point x for optimizing $g(x|\lambda^i)$.

Step 4 Locating Candidate Points: Using MOEA/D, obtain $\tilde{x}^1, \ldots, \tilde{x}^N$, where $\tilde{x}^i$ is an approximate solution for maximizing $\xi^i(x)$.

Step 5 Selecting Points for Function Evaluation: Select K_E points from $\tilde{x}^1, \ldots, \tilde{x}^N$ using a selection scheme.

Step 6 Function Evaluations: Evaluate the F-function values of all the K_E selected points in **Step 5**, then add all these points to P_{eval} and go to **Step 2**.

P_{eval} is the set of the solutions which have been evaluated during the previous search. **Step 1** initializes P_{eval}. **Step 3** defines $\xi^i(x)$, which are maximized in **Step 4** by using MOEA/D. **Step 5** determines which points will be evaluated in **Step 6**. At each generation of MOEA/D-EGO, K_E points can be evaluated in parallel.

In Sections IV-VI, we will give and discuss the details of steps 3-5.

IV. Model Building

This section discusses the implementation of **Step 3** in MOEA/D-EGO.

A. Gaussian Process Modeling

1) Assumptions: To build a cheap surrogate model for an expensive function $y = g(x)$, $x \in R^n$, Gaussian process modeling makes the following assumptions [1]:

- For any x, $g(x)$ is a sample of the following Gaussian random variable:

$$\mu + \epsilon(x) \tag{5}$$

  where $\epsilon(x) \sim N(0, \sigma^2)$, μ and σ are two constants independent of x. In other words, the prior distribution of $g(x)$ is Gaussian distribution with constant mean μ and constant variance σ^2.
- For any $x, x' \in R^n$, $c(x, x')$, the correlation between $\epsilon(x)$ and $\epsilon(x')$, depends only on $x - x'$, more precisely,

$$c(x, x') = \exp[-d(x, x')], \tag{6}$$

  where

$$d(x, x') = \sum_{i=1}^{n} \theta_i |x_i - x'_i|^{p_i}, \tag{7}$$

  $\theta_i > 0$ and $1 \leq p_i \leq 2$.

Clearly, $c(x, x') \to 1$ as $d(x, x') \to 0$ and $c(x, x') \to 0$ as $d(x, x') \to +\infty$, which is desirable in modeling a continuous function $g(x)$. p_i is related to the smoothness of $g(x)$ with respect to x_i, and θ_i indicates the importance of x_i on $g(x)$. More details of Gaussian process modeling can be found in [40].

2) Hyper Parameter Estimation: Given K points $x^1, \ldots, x^K \in R^n$ and their g-function values $y^1, \ldots, y^K$, the hyper parameters μ, σ, $\theta_1, \ldots, \theta_n$, and $p_1, \ldots, p_n$ can be estimated by maximizing the likelihood that $g(x) = y^i$ at $x = x^i$ $(i = 1, \ldots, K)$ [1]:

$$\frac{1}{(2\pi\sigma^2)^{K/2}\sqrt{det(C)}} \exp\left[-\frac{(y - \mu\mathbf{1})^T C^{-1}(y - \mu\mathbf{1})}{2\sigma^2}\right] \tag{8}$$

where C is a $K \times K$ matrix whose (i, j)-element is $c(x^i, x^j)$, $y = (y^1, \ldots, y^K)^T$ and $\mathbf{1}$ is a K-dimensional column vector of ones.

To maximize (8), the values of μ and σ^2 must be:

$$\hat{\mu} = \frac{\mathbf{1}^T C^{-1} y}{\mathbf{1}^T C^{-1} \mathbf{1}} \tag{9}$$

and

$$\hat{\sigma}^2 = \frac{(y - \mathbf{1}\hat{\mu})^T C^{-1}(y - \mathbf{1}\hat{\mu})}{K}. \tag{10}$$

Substituting (9) and (10) into (8) eliminates the unknown parameters μ and σ from (8). As a result, the likelihood function depends only on θ_i and p_i for $i = 1, \ldots, n$. An optimization method can then be used for maximizing (8) to obtain estimates $\hat{\theta}_i$ and $\hat{p}_i$. Then estimates $\hat{\mu}$ and $\hat{\sigma}^2$ can be readily obtained from (9) and (10).

(8) is multimodal [1], gradient based methods could be trapped on its local optima. In the experiments in this paper, Differential Evolution (DE) is employed for maximizing (8). The version used is *rand/1/bin* whose details can be found in [41]. The control parameters in DE are set as:

- the population size=20;
- the number of generations=50;
- the crossover rate=0.1; and
- the value of F=0.5.

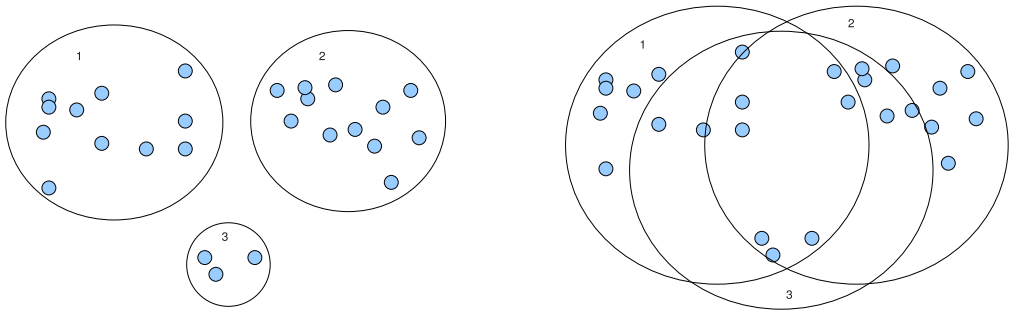
3) The Best Linear Unbiased Prediction and Predictive Distribution: Given hyper parameter estimates $\hat{\theta}_i$, $\hat{p}_i$, $\hat{\mu}$ and $\hat{\sigma}^2$, one can predict $g(x)$ at any untested point x based on the g-function values y^i at x^i for $i = 1, \ldots, K$. The best linear unbiased predictor of $g(x)$ is [1]:

$$\hat{y}(x) = \hat{\mu} + r^T C^{-1}(y - \mathbf{1}\hat{\mu}) \tag{11}$$

and its mean squared error is:

$$s^2(x) = \hat{\sigma}^2[1 - r^T C^{-1} r + \frac{(1 - \mathbf{1}^T C^{-1} r)^2}{\mathbf{1}^T C^{-1} r}] \tag{12}$$

where $r = (c(x, x^1), \ldots, c(x, x^K))^T$. $N(\hat{y}(x), s^2(x))$ can be regarded as a predictive distribution for $g(x)$ given the g-function values y^i at x^i for $i = 1, \ldots, K$.



(a) Crisp Clustering (b) FuzzyCM

Figure 1. In crisp clustering, all the points are clustered into several clusters of different sizes, each point belongs to just one cluster. In FuzzyCM used in this paper, all the clusters are of the same size and one point might belongs to several different clusters.

4) Fuzzy Clustering based Method for Modeling (FuzzyCM): The computational overhead in estimating the hyper parameters and in computing $\hat{y}(x)$ and $s^2(x)$ depends mainly on K. When K is larger than 200, it is computationally unbearable to optimize the hyper parameters if all the points are directly used [40]. There are several strategies to deal with this issue. Among them one is to select a small number of evaluated points for estimation and prediction [42], [43]. Obviously, this strategy doesn't make the full use of all the evaluated points. Another commonly used strategy is to cluster all the evaluated points into several small clusters and then build several local predictive models based on these clusters, the prediction of an untested point is based on a local predictive model with the closest cluster center to it [44]–[46]. To the best of our knowledge, all these clustering-based methods use crisp clustering, each evaluated point is assigned to exactly one cluster. One of their major drawbacks is that their prediction quality is poor if the point to be predicted is in boundary areas among different clusters, since evaluated points close to it have not been used efficiently.

We propose a fuzzy clustering [47] based modeling method (FuzzyCM) to relieve this drawback. FuzzyCM needs two control parameters:

- L_1: the maximal number of points used for building a local model.
- L_2: the number of points for adding one more local model.

where $L_1 > L_2$. When $K \leq L_1$, all the K evaluated points are directly used for building a predictive distribution of $g(x)$ (i.e. computing the values of $\hat{y}(x)$ and $\hat{s}^2(x)$ in (11) and (12)) at any untested point x. When $K > L_1$, FuzzyCM works as follows:

Step 1 Set the number of clusters:

$$c_{size} = 1 + \lceil \frac{K - L_1}{L_2} \rceil. \tag{13}$$

Step 2 By applying the Fuzzy C-Means Clustering (its details can be found in Appendix A), on all the points in P_{eval}, generate c_{size} fuzzy clusters; the center of cluster i is v^i. Set P_i to be the set containing the L_1 points from P_{eval} with the highest membership degrees to fuzzy cluster i.

Step 3 For each P_i, $i = 1, \ldots, c_{size}$, maximize the likelihood function (8) of all the L_1 points in it and obtain a set of estimate values of the hyper parameters.

To build a predictive distribution $N(\hat{y}(x), \hat{s}^2(x))$ of $g(x)$ at a untested point x, we first find its closest fuzzy cluster center v^k (In the case when x has more than one closest cluster centers, the tie is broken at random.), then use the hyper parameter estimates associated with P_k and points only in P_k to compute $\hat{y}(x)$ in (11) and $\hat{s}^2(x)$ in (12). In other words, $N(\hat{y}(x), \hat{s}^2(x))$ is a predictive distribution given the g-function values on the points in P_k.

In the above method, c_{size} clusters are generated by the Fuzzy C-means Clustering on all the points. Each P_j contains exactly L_1 points which are most likely to belong to cluster center v^j when $K > L_1$. Since $L_1 > L_2$, there are overlaps among different P_j's. Therefore, the prediction quality in boundary areas is improved. Fig. 1 illustrates the difference between crisp clustering and FuzzyCM.

In the experiments in this paper, L_1 is set to be 80 and L_2 is 20.

B. Model Composition

If the predictive distribution models for all the aggregated functions in **Step 3** of MOEA/D-EGO are constructed one by one by using our proposed FuzzyCM, it will incur very high, even unbearable, computational cost. Note that each $g(x|\lambda^i)$ is an aggregation of the MOP individual objective functions $f_1, \ldots, f_m$, we propose to use the FuzzyCM to build predictive distribution models for $f_1, \ldots, f_m$ and then from them derive a predictive distribution model for each $g(x|\lambda^i)$.

In the following, we explain how to do it in the cases when $g(x|\lambda^i)$ are given by (2) or (3). We assume that we have obtained a predictive distribution model $N(\hat{y}_i(x), \hat{s}_i^2(x))$ for each MOP individual objective function $f_i(x)$ in **Step 3** of MOEA/D-EGO and for simplicity, we further assume that all these individual functions are independent of one another statistically.

1) Weight Sum Aggregation: Since

$$g^{ws}(x|\lambda) = \sum_{i=1}^{m} \lambda_i f_i(x),$$

$g^{ws}(x|\lambda)$ can be regarded as a sample of Gaussian random variable $N(\hat{y}^{ws}, (\hat{s}^{ws})^2)$. [1], where

$$\hat{y}^{ws} = \sum_{i=1}^{m} \lambda_i \hat{y}_i(x), \tag{14}$$

and

$$(\hat{s}^{ws})^2 = \sum_{i=1}^{m} [\lambda_i \hat{s}_i^2(x)]^2. \tag{15}$$

Therefore, in **Step 3** of MOEA/D-EGO, we can use $N(\hat{y}^{ws}, (\hat{s}^{ws})^2)$ as a predictive model of $g^{ws}(x|\lambda)$.

2) Tchebycheff Aggregation: Since

$$f_i(x) \sim N(\hat{y}_i(x), \hat{s}_i^2(x))$$

for all $i = 1, \ldots, m$, we have

$$\lambda_i(f_i(x) - z_i^*) \sim N(\lambda_i(\hat{y}_i(x) - z_i^*), [\lambda_i \hat{s}_i(x)]^2).$$

$f_i(x)$ $(i = 1, \ldots, m)$ are independent, so are $\lambda_i(f_i(x) - z_i^*)$ $(i = 1, \ldots, m)$.

In the case of $m = 2$, it is from [48] (the details can be found in Appendix B) that

$$E[g^{te}(x|\lambda)] = \mu_1 \Phi(\alpha) + \mu_2 \Phi(-\alpha) + \tau\varphi(\alpha) \tag{16}$$

and

$$\begin{aligned} E([g^{te}(x|\lambda)]^2) = \quad & (\mu_1^2 + \sigma_1^2)\Phi(\alpha) + (\mu_2^2 + \sigma_2^2)\Phi(-\alpha) \\ & + (\mu_1 + \mu_2)\varphi(\alpha) \end{aligned} \tag{17}$$

where

$$\sigma_i^2 = [\lambda_i \hat{s}_i(x)]^2 \text{ for } i = 1, 2,$$

$$\mu_i = \lambda_i(\hat{y}_i(x) - z_i^*) \text{ for } i = 1, 2,$$

$$\tau = \sqrt{[\lambda_1 \hat{s}_1(x)]^2 + [\lambda_1 \hat{s}_2(x)]^2},$$

[1]Both $\hat{y}^{ws}$ and $(\hat{s}^{ws})^2$ are functions of λ and x. For simplicity, we drop them in the notations.

$$\alpha = (\mu_1 - \mu_2)/\tau,$$
$$\varphi(t) = (2\pi)^{-1/2} exp(-t^2/2),$$

and

$$\Phi(t) = \int_{-\infty}^{t} \varphi(\theta) d\theta.$$

Let

$$\hat{y}^{te} = E(g^{te}(x|\lambda)) \tag{18}$$

and

$$(\hat{s}^{te})^2 = E([g^{te}(x|\lambda)]^2) - (\hat{y}^{te})^2. \tag{19}$$

As suggested in [48], the distribution of $g^{te}(x|\lambda)$ can be approximated by $N(\hat{y}^{te}, (\hat{s}^{te})^2)$, which is used in our experiments as a predictive distribution of $g^{te}(x|\lambda)$.

In the case of $m = 3$, note that

$$g^{te}(x|\lambda) = \max\{\max\{\lambda_i(f_1(x) - z_1^*), \lambda_i(f_2(x) - z_2^*)\}, \lambda_3(f_3(x) - z_3^*)\},$$

we can, following [48], first build an approximate Gaussian distribution for $\max\{\lambda_i(f_1(x) - z_1^*), \lambda_i(f_2(x) - z_2^*)\}$ as in the case of $m = 2$, and then estimate an approximate Gaussian distribution for $g^{te}(x|\lambda)$ by treating it as the maximum of two independent Gaussian random variables $\max\{\lambda_i(f_1(x) - z_1^*), \lambda_i(f_2(x) - z_2^*)\}$ and $\lambda_3(f_3(x) - z_3^*)$.

The above method can be easily generalized to the case when $m > 3$. However, it might be worthwhile to reduce the number of objectives for an expensive MOP with more than 3 objectives before solving it since the number of solutions required for approximating the PF of a MOP increases exponentially with the number of objectives.

C. Expected Improvement

After building a predictive distribution model for each aggregated objective in **Step 3** of MOEA/D-EGO, we then define a metric for measuring the merit of evaluating it at a new untested point for minimizing $g(x|\lambda)$. In principle, any metrics developed for single objective optimization can be serve for this purpose. In the experiments in this paper, we use the expected improvement (EI) proposed in [6].

Suppose $N(\hat{y}(x), [\hat{s}(x)]^2)$ is a predictive distribution model for an objective function $g(x)$, and the minimal value of $g(x)$ over all the evaluated points in P_{eval} is g_{min}, then the improvement of $g(x)$ at a untested point x is

$$I(x) = \max\{g_{min} - g(x), 0\}$$

Thus the expected improvement is:

$$E[I(x)] = E[\max\{g_{min} - g(x), 0\}]$$

It can be written as [1]:

$$E[I(x)] = [(g_{min} - \hat{y}(x)]\Phi(\frac{g_{min} - \hat{y}}{\hat{s}(x)}) + \hat{s}(x)\phi(\frac{g_{min} - \hat{y}}{\hat{s}(x)}) \tag{20}$$

The above formula is used in our experiments for computing the expected improvement in our experiments.

Clearly, $E[I(x)]$ is larger at a point x when a predicted g-function value at x (i.e., $\hat{y}(x)$) is much smaller than the minimal function value found so far (i.e., g_{min}) and/or the value of $g(x)$ is very uncertain (i.e., $\hat{s}(x)$ is very large). In a sense, maximizing $E[I(x)]$ balances exploitation and exploration. The expected improvement has been used in the famous EGO with great success [1].

In summary, **Step 3** of MOEA/D-EGO first uses FuzzyCM to build a predictive probability model for each individual f_i, and then derives predictive probability models for all the aggregation objective functions, finally sets $\xi^i(x)$ to be the expected improvement of $g(x|\lambda^i)$.

V. Locating Candidate Points

Step 4 of MOEA/D-EGO is for solving N single optimization problems with the different objectives $\xi^i(x)$ $i = 1, \ldots, N$. If λ^i is close to λ^j, we call $\xi^i(x)$ and $\xi^j(x)$ neighbors. Recall that $\xi^i(x)$ is set to be the expected improvement of $g(x|\lambda^i)$, neighboring objectives should be similar. Therefore, optimization of $\xi^i(x)$ can utilize information from its neighboring objectives. This is the basic idea behind MOEA/D. In **Step 4** of MOEA/D-EGO, we use MOEA/D-DE, proposed in [32], for finding the optimal solutions of these N single optimization problems. Its details can be found in Appendix C.

VI. Selecting Points for Function Evaluation

Step 5 in MOEA/D-EGO is to select K_E points from N candidate solutions generated in **Step 4**. These selected K_E points will be evaluated at **Step 6**. We have the following considerations in implementing **Step 5**:

- Since the goal of MOEA/D-EGO is to generate a number of well representative Pareto optimal solutions, the points to be evaluated should be different from the points already evaluated;
- Each solution $\tilde{x}^i$ obtained in **Step 4** of MOEA/D-EGO is for maximizing ξ^i and therefore the evaluation of these points is good for optimizing their associated aggregative functions $g(x|\lambda^i)$. To encourage the diversity of final solutions in the objective space, $\tilde{x}^j$ should not be selected if $\tilde{x}^i$ has been selected and λ^i and λ^i are close.
- The selected points should have high EI values so that the evaluation of these points will, hopefully, lead to substantial improvement in solution quality.

Based on the above considerations, in **Step 5** of MOEA/D-EGO, we select K_E points as follows:

Step 1 Set $Q = \emptyset$.

For $i = 1$ to N,
If $\tilde{x}^i$ is different from any points in $P_{eval} \cup Q$, then add $\tilde{x}^i$ to Q.
End For

Step 2 Using K-Means clustering, cluster all the weight vectors associated with the solutions in Q into K_E clusters. Correspondingly, Q is clustered into K_E clusters.

Step 3 Select the point with the highest ξ-value from each cluster.

In the above method, Step 1 guarantees that all the selected points in Step 3 are different from one another and from any points evaluated in the previous search. Point $\tilde{x}^i$ is the solution for maximizing ξ^i and associated with weight vector λ^i. Steps 2 and 3 ensure that the K_E selected points are diverse in the objective space since it does not allow any two selected points to be associated with very similar aggregative functions. In our implementation of Step 1, two points are different if and only if their Euclidean distance is larger than 10^{-5}.

VII. Comparison With ParEGO and SMS-EGO

This paper is for dealing with expensive multiobjective optimization problems when a small number of function evaluations can be carried out in parallel. In this section, we compare our proposed MOEA/D-EGO with ParEGO and SMS-EGO [26]. We would like to make the following comments on ParEGO and SMS-EGO.

- Both ParEGO and MOEA/D-EGO are aggregation based. ParEGO optimizes the EI value of one aggregation function and can generate only one point to evaluate at each generation, while MOEA/D-EGO considers a number of aggregation functions and is able to produce several points for evaluation. In a sense, MOEA/D-EGO can be regarded as a generalization of ParEGO.
- As shown in [26], SMS-EGO outperforms ParEGO and the method of Jeong and Obayashi in terms of the number of function evaluations. SMS-EGO uses the method of Emmerich *et al* [23] for defining a figure of merit for evaluating a new point. Like ParEGO, it can only generate one point to evaluate at each generation.

A. Test Instances

The following MOP test instances with various characteristics are used in our experimental studies. All these test instances are minimization problems.

- ZDT1 [49]

$$\begin{aligned} f_1(x) &= x_1, \\ f_2(x) &= g(x)\left[1 - \sqrt{f_1(x)/g(x)}\right] \end{aligned}$$

where

$$g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$$

and $x = (x_1, \ldots, x_n)^T \in [0,1]^n$. Its PF is convex.

- ZDT2 [49]

$$\begin{aligned} f_1(x) &= x_1, \\ f_2(x) &= g(x)\left[1 - (f_1(x)/g(x))^2\right] \end{aligned}$$

where $g(x)$ and the range and dimensionality of x are the same as in ZDT1. The PF of ZDT2 is nonconvex.

- ZDT3 [49]

$$\begin{aligned} f_1(x) &= x_1, \\ f_2(x) &= g(x)\left[1-\sqrt{f_1(x)/g(x)}-\frac{f_1(x)}{g(x)}\sin\left(10\pi x_1\right)\right] \end{aligned}$$

where $g(x)$ and the range and dimensionality of x are the same as in ZDT1. Its PF is disconnected.

- ZDT4 [49]

$$\begin{aligned} f_1(x) &= x_1, \\ f_2(x) &= g(x)\left[1-\sqrt{f_1(x)/g(x)}\right] \end{aligned}$$

where

$$g(x) = 1+10(n-1)+\sum_{i=2}^{n}\left[x_i^2-10\cos\left(4\pi x_i\right)\right],$$

and $x=(x_1,\ldots,x_n)^T \in [0,1]\times[-5,5]^{n-1}$. It has many local PFs.

- ZDT6 [49]

$$\begin{aligned} f_1(x) &= 1-\exp\left(-4x_1\right)\sin^6\left(6\pi x_1\right), \\ f_2(x) &= g(x)\left[1-\left(f_1(x)/g(x)\right)^2\right] \end{aligned}$$

where

$$g(x) = 1+9\left[\left(\sum_{i=2}^{n}x_i\right)/(n-1)\right]^{0.25},$$

and $x=(x_1,\ldots,x_n)^T \in [0,1]^n$. Its PF is nonconvex. The distribution of the Pareto solutions very nonuniform, i.e. For a set of uniformly distributed points in the Pareto set in the decision space, their images is not uniformly distributed in the Pareto front in the objective space.

- LZ08-F1 [32]

$$\begin{aligned} f_1 &= x_1+\frac{2}{|J_1|}\sum_{j\in J_1}[x_j-x_1^{0.5+\frac{3(j-2)}{2(n-2)}}]^2, \\ f_2 &= 1-\sqrt{x_1}+\frac{2}{|J_2|}\sum_{j\in J_2}[x_j-x_1^{0.5+\frac{3(j-2)}{2(n-2)}}]^2 \end{aligned}$$

where

$$\begin{aligned} J_1 &= \{j|j \text{ is odd and } 2\le j\le n\}, \\ J_2 &= \{j|j \text{ is even and } 2\le j\le n\}, \end{aligned}$$

and $x=(x_1,...,x_n)^T \in [0,1]^n$. Unlike ZDT test instances, the PS of LZ08-F1 is a nonlinear curve:

$$x_j = x_1^{0.5+\frac{3(j-2)}{2(n-2)}},\ \ j=2,\ldots,n.$$

where $x_1 \in [0,1]$.

- LZ08-F2 [32]

$$\begin{aligned} f_1 &= x_1+\frac{2}{|J_1|}\sum_{j\in J_1}(x_j-\sin(6\pi x_1+\frac{j\pi}{n}))^2 \\ f_2 &= 1-\sqrt{x_1}+\frac{2}{|J_2|}\sum_{j\in J_2}(x_j-\sin(6\pi x_1+\frac{j\pi}{n}))^2 \end{aligned}$$

where J_1 and J_2 are the same as those of LZ08-F1. $x=(x_1,...,x_n)^T \in [0,1]\times[-1,1]^{n-1}$. Its PS is the following nonlinear curve:

$$x_j = \sin(6\pi x_1+\frac{j\pi}{n}), j=2,\ldots,n.$$

where $x_1 \in [0,1]$.

- LZ08-F3 [32]

$$\begin{aligned} f_1 &= x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - 0.8x_1\cos(6\pi x_1 + \frac{j\pi}{n}))^2 \\ f_2 &= 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - 0.8x_1\sin(6\pi x_1 + \frac{j\pi}{n}))^2 \end{aligned}$$

where J_1 and J_2 are the same as those of LZ08-F1. $x = (x_1, ..., x_n)^T \in [0,1]\times[-1,1]^{n-1}$. Its PS is the following nonlinear curve:

$$x_j = \begin{cases} 0.8x_1\cos(6\pi x_1 + \frac{j\pi}{n}) & j \in J_1 \\ 0.8x_1\sin(6\pi x_1 + \frac{j\pi}{n}) & j \in J_2 \end{cases}$$

where $x_1 \in [0,1]$.

- LZ08-F4 [32]

$$\begin{aligned} f_1 &= x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}[x_j - 0.8x_1\cos(2\pi x_1 + \frac{j\pi}{3n})]^2 \\ f_2 &= 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}[(x_j - 0.8x_1\sin(6\pi x_1 + \frac{j\pi}{n})]^2 \end{aligned}$$

where J_1 and J_2 are the same as those in LZ08-F1, and $x = (x_1, \ldots, x_n)^T \in [0,1]\times[-1,1]^{n-1}$. Its PS is the following nonlinear curve:

$$x_j = \begin{cases} 0.8x_1\cos(2\pi x_1 + \frac{j\pi}{3n}) & j \in J_1, \\ 0.8x_1\sin(6\pi x_1 + \frac{j\pi}{n}) & j \in J_2. \end{cases}$$

where $x_1 \in [0,1]$.

- DTLZ2 [50]

$$\begin{aligned} f_1(x) &= (1+g(x))\cos(x_1\pi/2)\cos(x_2\pi/2), \\ f_2(x) &= (1+g(x))\cos(x_1\pi/2)\sin(x_2\pi/2), \\ f_3(x) &= (1+g(x))\sin(x_1\pi/2), \end{aligned}$$

where

$$g(x) = \sum_{i=3}^{n} x_i^2$$

and $x = (x_1, \ldots, x_n)^T \in [0,1]^2 \times [-1,1]^{n-2}$. Its PF is nonconvex. The function value of a Pareto optimal solution satisfies $\sum_{i=1}^{3} {f_i}^2 = 1$ with $f_i \geq 0, i = 1, 2, 3$.

- KNO1 [24]

$$\begin{aligned} f_1(x) &= 20 - r\cos(\phi) \\ f_2(x) &= 20 - r\sin(\phi) \end{aligned}$$

where

$$\begin{aligned} r &= 9 - \{3\sin[\frac{5}{2(x_1+x_2)^2}] + 3\sin[4(x_1+x_2)] \\ &\quad +5\sin[2(x_1+x_2)+2]\} \\ \phi &= \frac{\pi}{12(x_1 - x_2 + 3)} \end{aligned}$$

where $x = (x_1, x_2)^T \in [0,3]^2$. Its PS lies in the line defined by $x_1 + x_2 = 4.4116$. there are 15 locally optimal fronts in this problem.

- VLMOP2 [51]

$$\begin{aligned} f_1(x) &= 1 - \exp(-\sum_{i=1,\ldots,n}(x_i - 1/\sqrt{n})^2) \\ f_2(x) &= 1 + \exp(-\sum_{i=1,\ldots,n}(x_i - 1/\sqrt{n})^2) \end{aligned}$$

where $x = (x_1, \ldots, x_n)^T \in [-2,2]^n$. Its PF is concave. The Pareto optima lie on the diagonal from $x_i = -1/\sqrt{n}$ to $x_i = 1/\sqrt{n}$ in decision space, $i = 1, \ldots, n$.

B. Experimental Settings

1) General Settings:

- *The number of decision variables (n)*: it is set to be 2 in KNO1 and VLMOP2, 8 for all the other two objective test instances and 6 for DTLZ2, and 6 for DTLZ2.
- *Initial test points*: the number of initial test points (i.e. K_I in MOEA/D-EGO) is set to be $11n-1$ in all the test instances. The $11n-1$ initial test points are generated using the Latin hypercube sampling method [52].
- *The maximal number of function evaluation*: 300 for DTLZ2 and 200 for all the 2-objective instances.
- *The number of independent runs*: we run each algorithm for each test instance 10 times independently.

2) MOEA/D-EGO:

- *The number of function evaluation at each generation*: K_E is set to be 5 in all the test instances.
- *The number of subproblems N and weight vectors $\lambda^1, \ldots, \lambda^N$*: They are controlled by an integer H. More precisely, $\lambda^1, \ldots, \lambda^N$ are all the weight vectors in which each individual weight takes a value from

$$\{\frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H}\}.$$

  Therefore, the number of subproblems (weight vectors) is:

$$N = C_{H+m-1}^{m-1},$$

  where m is the number of objectives. H is set to be 299 for all the 2-objective test instances, and 33 for the 3-objective instances. Consequently, N is 300 for the 2-objective instances and 595 for the 3-objective instances.
- *Aggregation Method*: The Techebycheff approach is used. The smallest value of f_i found so far is use to substitute z_i^* in z^*.

3) ParEGO and SMS-EGO:

- All the parameter settings of these two methods are the same as in [24] and [26], respectively.
- The source codes of these two methods used in our experimental studies are obtained from their authors.

C. Performance Metric

The inverted generational distance (IGD) [53] 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 IGD from P^* to P is defined as:

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

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, IGD(P^*, P) could measure both the diversity and convergence of P in a sense. To have a low value of IGD(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 990 points for each test instance with 3 objectives.

D. Experimental Results

Since SMS-EGO uses all the tested points for building models and it was implemented in MATLAB, it is very difficult to test it on problems with 8 variables with 200 function evaluations. Actually, SMS-EGO takes about three hours for a single run with 200 function evaluation even on problems with 2 variable on a computer with 2.8 GHZ. For this reason, we test SMS-EGO only on two test instances with 2 variables in our experiments. The approximation set generated by each algorithm in a single run is all the nondominated solutions found during the run.

1) Comparison: Table I presents the statistics of the IGD values of the approximation sets obtained by each algorithm in ten independent runs. Figs. 2 to 6 plot, in the objective space, the distribution of the approximation set obtained in the run with the lowest IGD-value of each algorithm for each test instance. Fig. 7 shows the evolution of the average IGD value of the nondominated solutions found so far with the number of function evaluations in each algorithm for each test instance.

It is evident from Table I that MOEA/D-EGO significantly outperforms ParEGO in terms of IGD metric on ZDT1, ZDT2, ZDT3, ZDT6 and LZ08-F1. The plots of the final solutions in the objective space in the figures in this section also clearly show that the approximations generated by MOEA/D-EGO are better than those by ParEGO on these five instances.

On LZ08-F2 LZ08-F3, LZ08-F4 and DTLZ3, there is no much difference between ParEGO and MOEA/D-EGO in terms of IGD values. It is also very difficult to judge which method is significantly better from the plots of the final solutions on these four instances.

Table I
STATISTICS OF THE IGD VALUES OF THE FINAL APPROXIMATION SETS OBTAINED BY MOEA/D-EGO, PAREGO AND SMS-EGO WITH THE SAME NUMBER OF FUNCTION EVALUATIONS (STD STANDS FOR STANDARD DEVIATION)

Problem	The number of evaluations	ParEGO			MOEA/D-EGO		
		lowest	mean	std	lowest	mean	std
ZDT1	200	0.0513	0.0945	0.0316	0.0120	0.0148	0.0016
ZDT2	200	0.0440	0.0864	0.0483	0.0121	0.0156	0.0022
ZDT3	200	0.2071	0.3818	0.1385	0.0435	0.0665	0.0189
ZDT4	200	2.9322	6.2364	3.9068	14.8688	33.3456	12.9428
ZDT6	200	0.4751	0.5736	0.074212	0.0273	0.0585	0.0421
LZF1	200	0.0692	0.0966	0.012500	0.0083	0.0100	0.0011
LZF2	200	0.1624	0.2419	0.045743	0.1297	0.1704	0.0241
LZF3	200	0.0784	0.1264	0.0275	0.1010	0.1488	0.0297
LZF4	200	0.0702	0.087072	0.0116	0.0659	0.0844	0.0144
DTLZ2	300	0.1356	0.1493	0.0101	0.1306	0.1398	0.0085
		SMS-EGO			MOEA/D-EGO		
KNO1	200	0.3933	0.4473	0.0375	0.3552	0.4300	0.1471
VLMOP2	200	0.0058	0.0064	0.0003	0.0043	0.0051	0.0004

On ZDT4, It is clear that ParEGO performs much better than MOEA/D-EGO in terms of IGD metric. However, the plots of the final solutions show that neither of these two algorithms can approximate the PF satisfactorily.

On KNO1 and VLMOP2, the IGD metric values in Table I indicate that MOEA/D-EGO performs very similarly to SMS-EGO, which is confirmed by the plots of the final solutions.

We can conclude that overall, the performance of MOEA/D-EGO is not worse than ParEGO and SMS-EGO if the computational cost is measured by the number of function evaluations. In our experiments, however, MOEA/D-EGO can evaluate five points simultaneously at every generation, while ParEGO and SMS-EGO can evaluate only one point. Therefore, in a parallel computing environment, MOEA/D-EGO is much more efficient and than ParEGO and SMS-EGO.

2) More Discussions: It is very clear that the performances of the algorithms vary greatly from instance to instance. For example, Both ParEGO and MOEA/D-EGO perform much better on ZDT2 than on ZDT4. One of the major reasons might be that the prediction quality of Gaussian process modeling is better on ZDT2 than on ZDT4. To verify it, we build Gaussian predictive models for f_2 in ZDT2 and ZDT4. In our modeling:

- The number of variables n is 8,
- The points for model building are generated by the Latin Square method,
- The number of the points for model building is 87.

Then, for each test instance, we randomly select $5,000$ points $x^1, \ldots, x^{5000}$ from its search space,compute the relative error:

$$RE = \frac{\{\sum_{i=1}^{5000}[f_2(x^i) - \hat{y}(x^i)]^2\}^{1/2}}{\sum_{i=1}^{5000} f_2(x^i)}$$

where $\hat{y}(x^i)$ is the predicted mean of f_2 at point x^i. The RE value is 0.0051 for ZDT2 while it is 0.2192 for ZDT4. It implies that Gaussian process modeling might not be suitable for ZDT4. One should introduce other techniques to deal with it [4].

In addition, we can observe from Fig. 3(d) that MOEA/D-EGO has found several optimal solutions for ZDT6. However, fig. 7(e) shows that the IGD value has not effectively reduced during the last 60 function evaluations. It suggests that the the last 60 test points are mainly for exploration, which wastes the computational resource in this instance. Therefore, the expected improvement does not always work properly for balancing exploitation and exploration. It should be worthwhile studying how to overcome this shortcoming for improving the algorithm performance.

VIII. CONCLUSION

In some real world applications, several function evaluations can be carried out at the same time, therefore, it is desirable to develop methods in which several different test points can be generated simultaneously at each iteration. This paper proposed such a method, MOEA/D-EGO, for dealing with expensive multiobjective optimization problems. Like other MOEA/D algorithms, it decomposes a MOP into a number of single objective optimization subproblems. At each iteration, a predictive distribution model is built for each individual objective in the MOP by using Fuzzy clustering and Gaussian process modeling. Then, a predictive model for the objective of each subproblem can be induced. MOEA/D are used for maximizing the expected improvement metrics of all the subproblems and several test points are then selected for evaluation.

Experimental results on a set of test instances have shown that MOEA/D-EGO is not worse than ParEGO and SMS-EGO if the computational cost is measured by the number of function evaluations. However, MOEA/D-EGO can evaluate several solutions at the same time, which makes it more suitable for solving expensive MOPs in some real-world applications. We also found that the prediction quality of Gaussian process modeling is poor in ZDT4 and it makes the algorithms fail in

approximating the PF, and the expected improvement metric could not balance exploitation and exploration very well in some instances such as ZDT6. Therefore, more effort is needed in the future to address these issues.

The source code of MOEA/D-EGO can be downloaded from: *http://dces.essex.ac.uk/staff/qzhang/*.

APPENDIX A: FUZZY C-MEANS (FCM) CLUSTERING

Given K points $x^1, \ldots, x^K$ in R^n, FCM clustering partitions them into c_{size} clusters such that the following objective function

$$J = \sum_{i=1}^{K} \sum_{j=1}^{c_{size}} u_{ij}^{\alpha} ||x^i - v^j||^2$$

is minimized, where α is a constant larger than 1, $v^j \in R^n$ is the center of cluster j, u_{ij} is the degree of membership of x_i in cluster j, and $\| * \|$ is a metric in R^n.

The algorithm works as follows:

Input:
- $x^1, \ldots, x^K \in R^n$: the points to be clustered.
- c_{size}: the number of clusters.
- α and $\| * \|$ used in computing the objective J, and ε used in the stopping criterion.

Output
- $v^1, \ldots, v^{c_{size}} \in R^n$: the cluster centers.
- u_{ij} for $i = 1, \ldots, K$ and $j = 1, \ldots, c_{size}$: the membership of x^i in cluster j.

Step 1 Initialize u_{ij}^0 for for $i = 1, \ldots, K$ and $j = 1, \ldots, c_{size}$ and set $t = 0$.

Step 2 Compute

$$v^j = \frac{\sum_{i=1}^{K} (u_{ij}^t)^{\alpha} x^i}{\sum_{i=1}^{K} (u_{ij}^t)^{\alpha}}$$

for $j = 1, \ldots, c_{size}$.

Step 3 Compute

$$u_{ij}^{t+1} = \frac{1}{\sum_{k=1}^{c_{size}} \left(\frac{||x^i - v^j||}{||x^i - v^k||} \right)^{2/(m-1)}}$$

Step 4 If $\max_{1 \le i \le k}$, and $_{1 \le j \le c_{size}} |u_{ij}^{t+1} - u_{ij}^t| < \varepsilon$, stop and output v^j and $u_{ij} = u_{ij}^{t+1}$. Otherwise, set $t = t + 1$ and go to Step 1.

In our experiments, $\alpha = 2$, $\varepsilon = 0.05$ and $|| * ||$ is Euclidean norm. For more details of FCM clustering, please refer to [47].

APPENDIX B: THE MAXIMUM OF SEVERAL NORMALLY DISTRIBUTED VARIABLES [48]

Suppose η_1 and η_2 are two normally distributed random variables:

$$\eta_i \sim N(\sigma_i, \sigma_i^2) \text{ for } i = 1, 2.$$

and r be the correlation between η_1 and η_2. Let

$$\xi = \max\{\eta_1, \eta_2\}$$

Then

$$E(\xi) = \mu_1 \Phi(\alpha) + \mu_2 \Phi(-\alpha) + \tau \varphi(\alpha),$$

$$E(\xi^2) = (\mu_1^2 + \sigma_1^2)\Phi(\alpha) + (\mu_2^2 + \sigma_2^2)\Phi(-\alpha) + (\mu_1 + \mu_2)\tau\varphi(\alpha),$$

where

$$\tau = \sqrt{\sigma_1^2 + \sigma_2^2 - 2r\sigma_1\sigma_2}$$

and

$$\alpha = (\mu_1 - \mu_2)/\tau.$$

APPENDIX C : MOEA/D FOR LOCATING CANDIDATE POINTS [27]

In **Step 5** of MOEA/D-EGO, MOEA/D is used for locating candidate points. It works as follows:

Algorithmic Parameters:

- T : the number of the weight vectors in the neighborhood of each weight vector;
- δ : the probability that parent solutions are selected from the neighborhood;
- n_r : the maximal number of solutions replaced by each child solution.

Step 1 Initialization

Step 1.1 Compute the Euclidean distances between any two weight vectors and then work out the T closest weight vectors to each weight vector. For each $i = 1, \ldots, N$, set $B(i) = \{i_1, \ldots, i_T\}$ where $\lambda^{i_1}, \ldots, \lambda^{i_T}$ are the T closest weight vectors to λ^i.

Step 1.2 Generate N points by uniformly randomly sampling from $\prod_{i=1}^{n}[a_i, b_j]$ and compute their ξ^i-function values, $i = 1, \ldots, N$. Set $\tilde{x}^i$ to be the point with the largest ξ^i-function value.

Step 2 Update

For $i = 1, \ldots, N$, do

Step 2.1 Selection of Mating/Update Range: Uniformly randomly generate a number $rand$ from $(0, 1)$. Then set

$$P = \begin{cases} B(i) & \text{if } rand < \delta, \\ \{1, \ldots, N\} & \text{otherwise.} \end{cases}$$

Step 2.2 Reproduction: Set $r_1 = i$ and randomly select two different indexes r_2 and r_3 from P, and then generate a solution $\bar{y}$ from $\tilde{x}^{r_1}, \tilde{x}^{r_2}$ and $\tilde{x}^{r_3}$ by a DE operator, and then perform a mutation operator on $\bar{y}$ to produce a new solution y.

Step 2.3 Repair: If an element of y is out of the boundary of $\prod_{i=1}^{n}[a_i, b_i]$, its value is reset to be a randomly selected value inside the boundary.

Step 2.4 Update of Solutions: Set $c = 0$ and then do the following:

(1) If $c = n_r$ or P is empty, go to **Step 3**. Otherwise, randomly pick an index j from P.
(2) If $\xi^j(y) > \xi^j(\tilde{x}^j)$, then set $\tilde{x}^j = y$, and $c = c + 1$.
(3) Remove j from P and go to (1).

Step 3 Stopping Criteria If the stopping criteria is satisfied, then stop and output $\{\tilde{x}^1, \ldots, \tilde{x}^N\}$. Otherwise go to **Step 2**.

In our experiments, the DE and mutation operators and their control parameters used in Step 2.2 are the same as that used in [32]. More discussions about MOEA/D-DE can be found in [32].

REFERENCES

[1] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient global optimization of expensive black-box functions," *J. Global Optim.*, vol. 13, no. 4, pp. 455–492, Dec. 1998.

[2] M. L. Stein, *Interpolation of Spatial Data: Some Theory for Kriging*. New York: Springer-Verlag, 1999.

[3] J. Sacks, W. J. Welch, T. J. Mitchell, and H. P. Wynn, "Design and analysis of computer experiments (with discussion)," *Statist. Sci.*, vol. 4, no. 4, pp. 409–423, Nov. 1989.

[4] D. R. Jones, "A taxonomy of global optimization methods based on response surfaces," *J. Global Optim.*, vol. 21, no. 4, pp. 345–383, Dec. 2001.

[5] D. Huang, T. T. Allen, W. I. Notz, and N. Zeng, "Global optimization of stochastic black-box systems via sequential kriging meta-models," *J. Global Optim.*, vol. 34, no. 3, pp. 441–466, Mar. 2006.

[6] J. Mockus, V. Tiesis, and A. Zilinskas, "The application of Bayesian methods for seeking the extremum," in *Towards Global Optimization*, L. C. W. Dixon and G. P. Szego, Eds. Amsterdam: Elsevier, 1978, vol. 2, pp. 117–129.

[7] B. E. Stuckman, "A global search method for optimizing nonlinear systems," *IEEE Trans. Syst., Man, Cybern.*, vol. 18, no. 6, pp. 965–977, Nov.-Dec. 1988.

[8] G. E. P. Box and N. R. Draper, *Emperical Model Building and Response Surfaces*. New York: Wiley, 1987.

[9] T. A. Donnelly, "Response-surface experimental design," *IEEE Potentials*, vol. 11, no. 1, pp. 19–21, Feb. 1992.

[10] Y. Jin, M. Olhofer, and B. Sendhoff, "A framework for evolutionary optimization with approximate fitness functions," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 481–494, Oct. 2002.

[11] H. Ulmer, F. Streichert, and A. Zell, "Evolution strategies assisted by Gaussian processes with improved preselection criterion," in *Proc. IEEE Congress on Evolutionary Computation CEC '03*, vol. 1, Canberra, Australia, Dec. 2003, pp. 692–699.

[12] R. Regis and C. Shoemaker, "Local function approximation in evolutionary algorithms for the optimization of costly functions," *IEEE Trans. Evol. Comput.*, vol. 8, no. 5, pp. 490–505, Oct. 2004.

[13] D. Buche, N. N. Schraudolph, and P. Koumoutsakos, "Accelerating evolutionary algorithms with Gaussian process fitness function models," *IEEE Trans. Syst., Man, Cybern. C*, vol. 35, no. 2, pp. 184–194, May 2005.

[14] Y.-S. Ong, P. B. Nair, and K. Y. Lum, "Max-min surrogate-assisted evolutionary algorithm for robust design," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 392–404, Aug. 2006.

[15] Z. Zhou, Y.-S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum, "Combining global and local surrogate models to accelerate evolutionary optimization," *IEEE Trans. Syst., Man, Cybern. C*, vol. 37, no. 1, pp. 66–76, Jan. 2007.

[16] I. Paenke, J. Branke, and Y. Jin, "Efficient search for robust solutions by means of evolutionary algorithms and fitness approximation," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 405–420, Aug. 2006.

[17] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 9, no. 1, pp. 3–12, Jan. 2005.

[18] B. S. Yang, Y. S. Yeun, and W. S. Ruy, "Managing approximation models in multiobjective optimization," *Structural and Multidisciplinary Optimization*, vol. 24, no. 2, pp. 141–156, Sept. 2002.

[19] H.-S. Chung and J. J. Alonso, "Multiobjective Optimization Using Approximation Model-Based Genetic Algorithms," in *Proc. 10th AIAA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*, New York, USA, Sept. 2004.

[20] S. Jeong and S. Obayashi, "Efficient global optimization (EGO) for multi-objective problem and data mining," in *Proc. IEEE Congress on Evolutionary Computation (CEC '03)*, vol. 3, Edinburgh, U.K., Sept. 2005, pp. 2138–2145.

[21] A. J. Keane, "Statistical improvement criteria for use in multiobjective design optimization," *AIAA Journal*, vol. 44, no. 4, pp. 879–891, Apr. 2006.

[22] M. K. Karakasis and K. C. Giannakoglou, "On the use of metamodel-assisted, multi-objective evolutionary algorithms," *Engineering Optimization*, vol. 38, no. 8, pp. 941–957, Dec. 2006.

[23] M. Emmerich, K. Giannakoglou, and B. Naujoks, "Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels," *IEEE Trans. Evol. Comput.*, vol. 10, no. 4, pp. 421–439, Aug. 2006.

[24] J. Knowles, "ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Trans. Evol. Comput.*, vol. 10, no. 1, pp. 50–66, Feb. 2006.

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

[26] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze, "Multiobjective optimization on a limited budget of evaluations using model-assisted s-metric selection," in *Proc. Parallel Problem Solving from Nature (PPSN X)*, Dortmund, Germany, Sept. 2008, pp. 784–794.

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

[28] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Trans. Syst., Man, Cybern. C*, vol. 28, no. 3, pp. 392–403, Aug 1998.

[29] H. Ishibuchi, T. Yoshida, and T. Murata, "Balance between genetic search and local search in memetic algorithms for multiobjective permutation flowshop scheduling," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 204–223, Apr. 2003.

[30] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - a comparative experiment," *IEEE Trans. Evol. Comput.*, vol. 6, no. 4, pp. 402–412, Aug. 2002.

[31] E. Hughes, "Multiple single objective pareto sampling," in *Proc. IEEE Congress on Evolutionary Computation (CEC '03)*, vol. 4, Canberra, Australia, Dec. 2003, pp. 2678–2684.

[32] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, accepted, 2008.

[33] P. C. Chang, S. H. Chen, Q. Zhang, and J. L. Lin, "MOEA/D for flowshop scheduling problems," in *Proc. IEEE Congress on Evolutionary Computation (CEC '08)*, Hong Kong, June 2008, pp. 1433–1438.

[34] W. Peng, Q. Zhang, and H. Li, "Comparision between MOEA/D and NSGA-II on the multi-objective travelling salesman problem," in *Multi-Objective Memetic Algorithms*, ser. Studies in Computational Intelligence, C.-K. Goh, Y.-S. Ong, and K. C. Tan, Eds. Heidelberg, Berlin: Springer, 2009, vol. 171.

[35] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Adaptation of scalarizing functions in MOEA/D: An adaptive scalarizing function-based multiobjective evolutionary algorithm," in *Proc. 5th International Conference devoted to Evolutionary Multi-Criterion Optimization (EMO '09)*, Nantes, France, Apr. 2009.

[36] K. Miettinen, *Nonlinear Multiobjective Optimization*. Norwell, MA: Kluwer, 1999.

[37] A. Pascoletti and P. Serafini, "Scalarizing vector optimization problems," *Journal of Optimization Theory and Applications*, vol. 42, no. 4, pp. 499–524, Apr. 1984.

[38] M. Ehrgott, *Multicriteria Optimization*. New York: Springer-Verlag, 2005.

[39] G. Eichfelder, *Adaptive Scalarization Methods in Multiobjective Optimization*. Heidelberg, Berlin: Springer, 2008.

[40] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA: The MIT Press, 2005.

[41] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. New York: Springer-Verlag, 2005.

[42] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian process regression: active data selection and test point rejection," in *Proc. IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN 2000)*, vol. 3, July 2000, pp. 241–246.

[43] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: the informative vector machine," in *Advances in Neural Information Processing Systems 15*, British Columbia, Canada, Dec. 2002, pp. 609–616.

[44] L. Bottou and V. Vapnik, "Local learning algorithms," *Neural Comput.*, vol. 4, no. 6, pp. 888–900, Nov. 1992.

[45] A. Choudhury, P. B. Nair, and A. J. Keane, "A data parallel approach for large-scale gaussian process modeling," in *Proc. Second SIAM International Conference on Data Mining (SDM '02)*, Arlington, VA, Apr. 2002.

[46] A. Schwaighofer and V. Tresp, "Transductive and inductive methods for approximate Gaussian process regression," in *Advances in Neural Information Processing Systems 15*, British Columbia, Canada, Dec. 2002, pp. 977–984.

[47] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. Norwell, MA: Kluwer, 1981.

[48] C. E. Clark, "The greatest of a finite set of random variables," *Operations Research*, vol. 9, no. 2, pp. 45–162, Mar.-Apr. 1961.

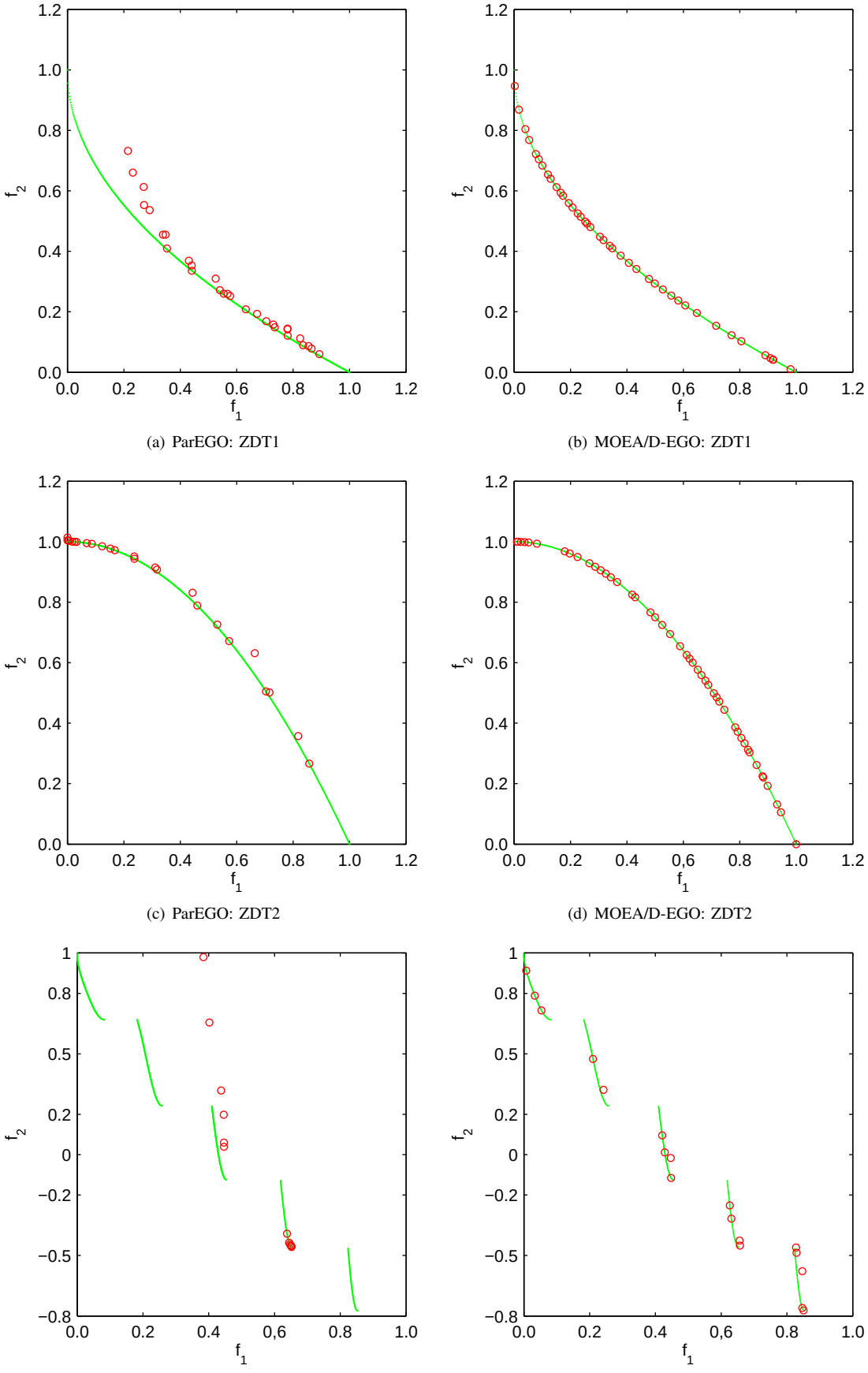
[49] K. Deb, "Multi-objective genetic algorithms: Problem difficulities and construction of test problems," *Evolutionary Computation*, vol. 7, no. 3, pp. 205–230, Fall 1999.

[50] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *Proc. IEEE Congress on Evolutionary Computation (CEC '02)*, vol. 1, Honolulu, HI, May 2002, pp. 825–830.

[51] D. A. van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in *Proc. ACM Symposium on Applied Computing (SAC '99)*, San Antonio, TX, Feb. 1999, pp. 351–357.

[52] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, Feb. 2000.

[53] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 117–132, Apr. 2003.



(a) ParEGO: ZDT1

(b) MOEA/D-EGO: ZDT1

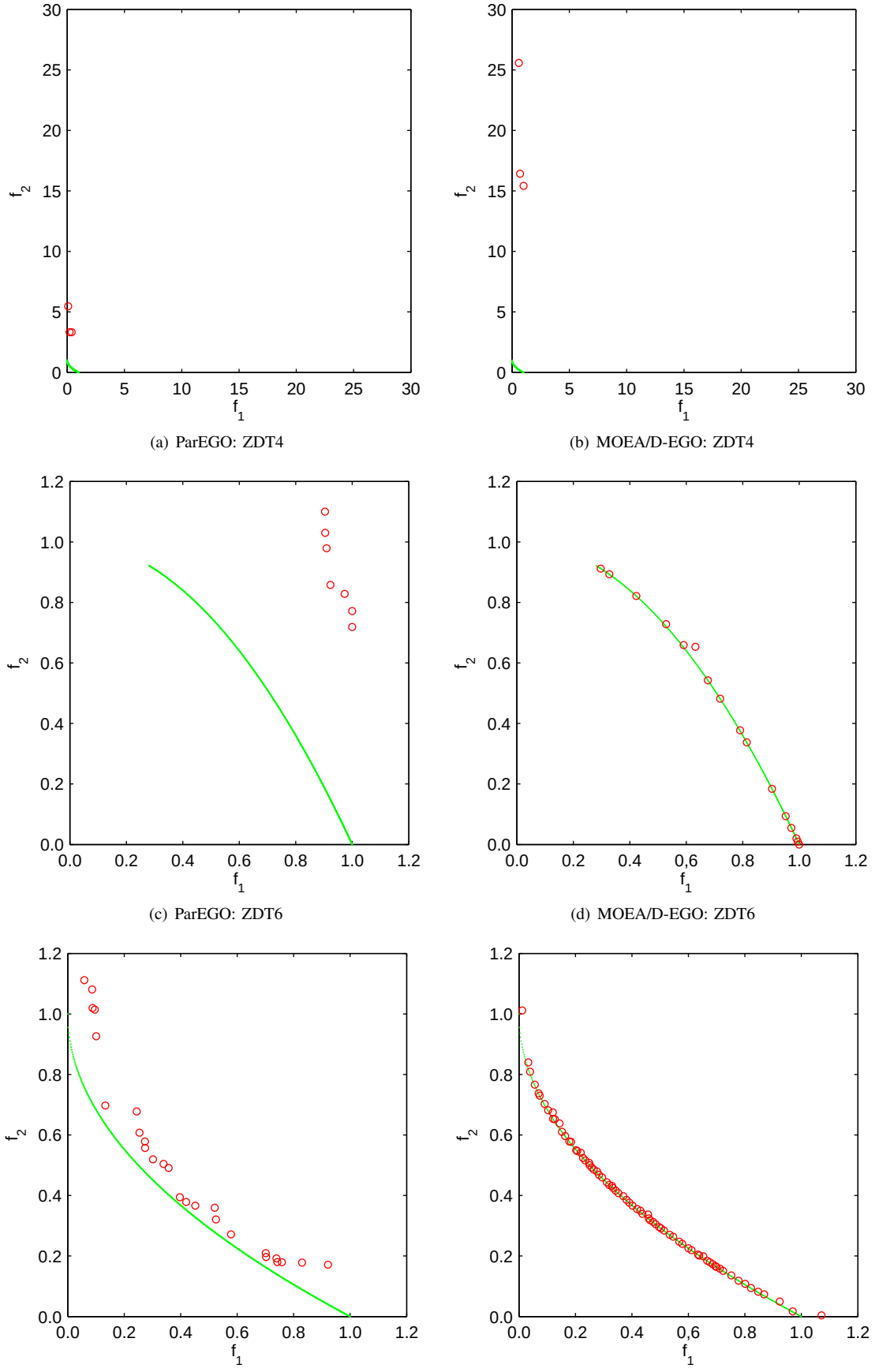
(c) ParEGO: ZDT2

(d) MOEA/D-EGO: ZDT2

(e) ParEGO: ZDT3

(f) MOEA/D-EGO: ZDT3

Figure 2. Plots of the final approximations with the lowest IGD values in the objective space on ZDT1, ZDT2 and ZDT3.



(a) ParEGO: ZDT4

(b) MOEA/D-EGO: ZDT4

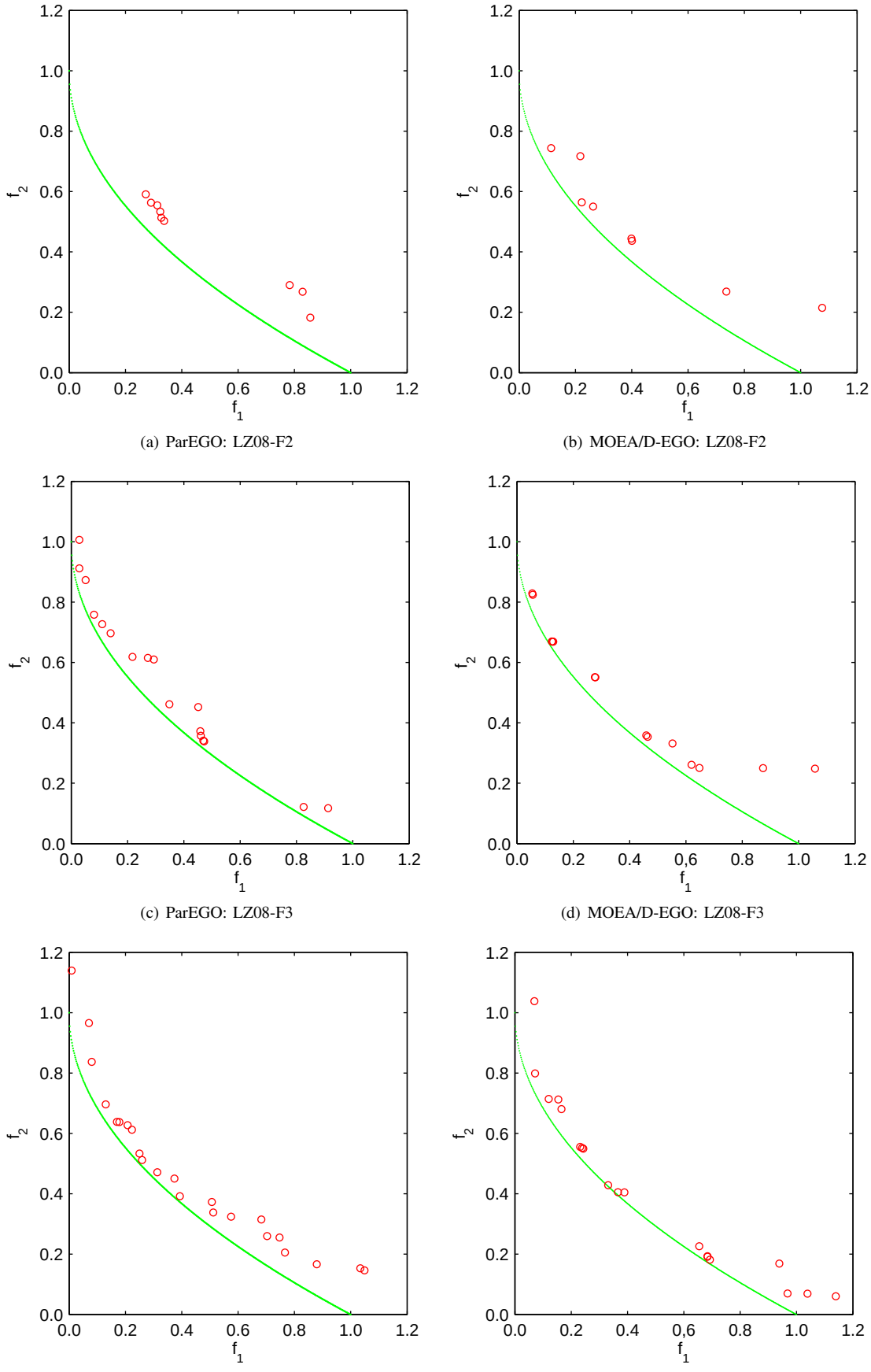
(c) ParEGO: ZDT6

(d) MOEA/D-EGO: ZDT6

(e) ParEGO: LZ08-F1

(f) MOEA/D-EGO: LZ08-F1

Figure 3. Plots of the final approximations with the lowest IGD values in the objective space on ZDT4, ZDT6 and LZ08-F1.



(a) ParEGO: LZ08-F2

(b) MOEA/D-EGO: LZ08-F2

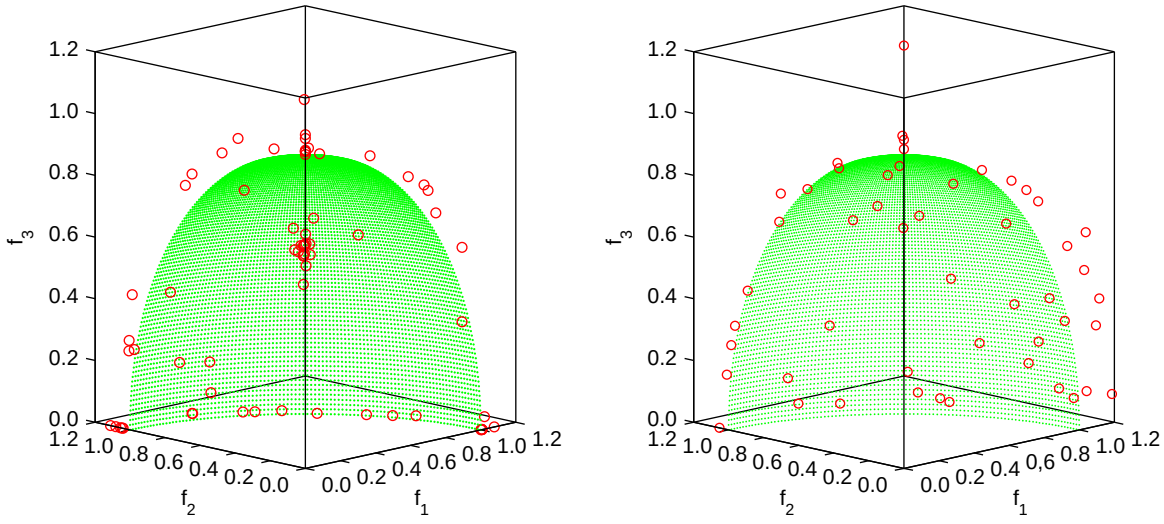
(c) ParEGO: LZ08-F3

(d) MOEA/D-EGO: LZ08-F3

(e) ParEGO: LZ08-F4

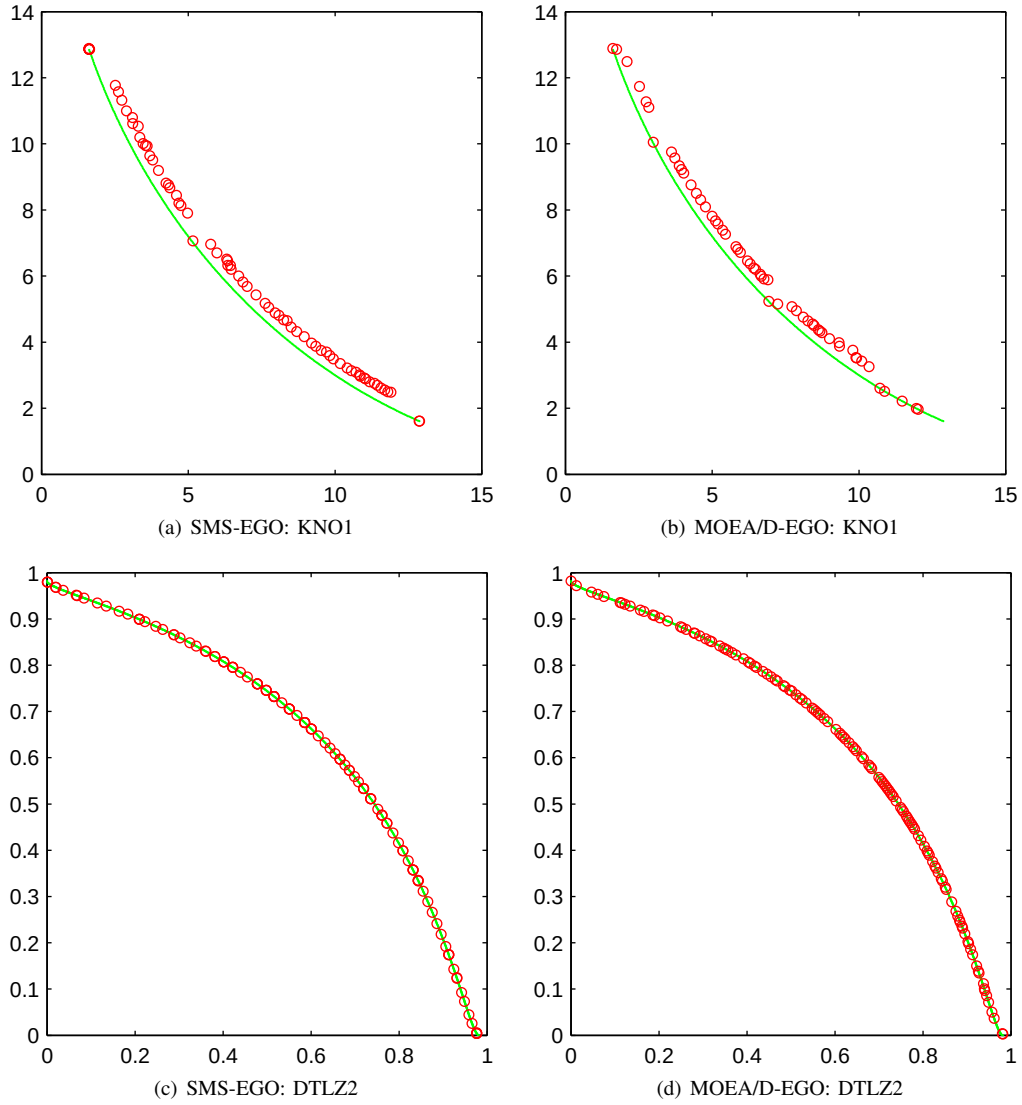
(f) MOEA/D-EGO: LZ08-F4

Figure 4. Plots of the final approximations with the lowest IGD values in the objective space on LZ08-F2 LZ08-F3 and LZ08-F4.



(a) ParEGO: DTLZ2 (b) MOEA/D-EGO: DTLZ2

Figure 5. Plots of the final approximations with the lowest IGD values in the objective space on DTLZ2.



(a) SMS-EGO: KNO1 (b) MOEA/D-EGO: KNO1

(c) SMS-EGO: DTLZ2 (d) MOEA/D-EGO: DTLZ2

Figure 6. Plots of the final approximations with the lowest IGD values in the objective space on KNO1 and VLMOP2.

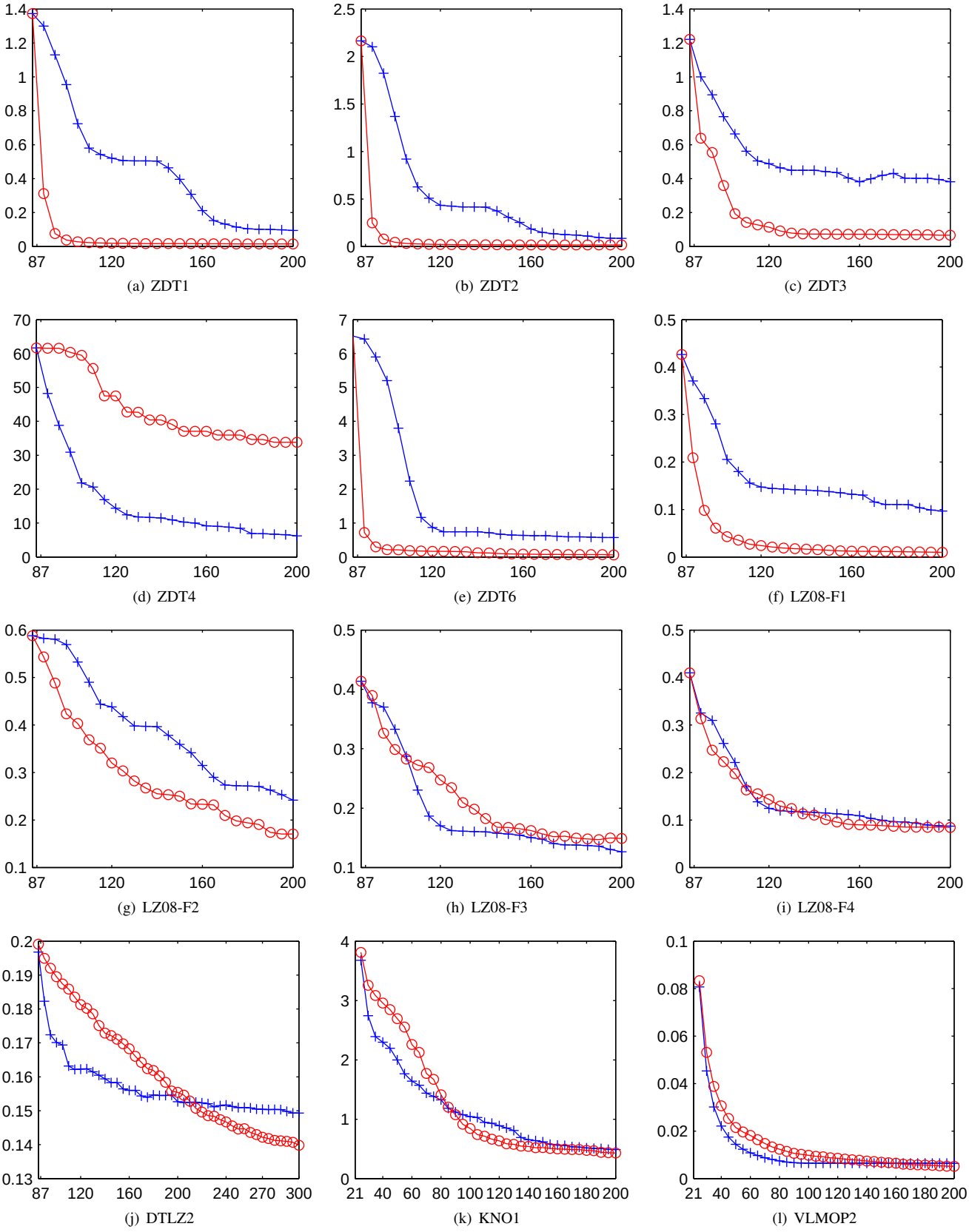


Figure 7. Evolution of the mean of IGD values versus the number of function evaluations. Figs 7(a) to 7(j) are for comparison between MOEA/D-EGO and ParEGO, and Figs 7(k) and 7(l) for MOEA/D-EGO and SMS-EGO. the lines marked with "o" are for MOEA/D-EGO. The lines marked with "+" are for ParEGO in Figs 7(a) to 7(j), and for SMS-EGO in Figs 7(k) and 7(l).