Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model

Qingfu Zhang, *Senior Member, IEEE,* Wudong Liu, Edward Tsang, and Botond Virginas

***Abstract*—In some expensive multiobjective optimization problems (MOPs), several function evaluations can be carried out in a batch way. Therefore, it is very desirable to develop methods which can generate multipler test points simultaneously. This paper proposes such a method, called MOEA/D-EGO, for dealing with expensive multiobjective optimization. MOEA/D-EGO decomposes an MOP in question into a number of single-objective optimization subproblems. A predictive model is built for each subproblem based on the points evaluated so far. Effort has been made to reduce 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. Extensive experimental studies have been carried out to investigate the ability of the proposed algorithm.**

***Index Terms*—Evolutionary algorithm, expensive optimization, Gaussian stochastic processes, multiobjective optimization, Pareto optimality.**

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, a method is of great practical interest if it is able to produce reasonably good solutions within a given (often very tight) budget on computational cost/time. This paper aims at developing such a method for approximating the Pareto front (PF) of an expensive multiobjective optimization problem (MOP).

The Gaussian stochastic process model (also called Kriging [1] or the design and analysis of computer experiments (DACE) stochastic process model [2]) has been widely accepted as one of the most efficient and effective tools for dealing with expensive single-objective optimization problems [3]–[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 such as the expected improvement [6] and the probability of improvement [7] for evaluation of the objective function at a new point could be defined and calculated for determining which point should be evaluated next. The efficient global optimization (EGO) algorithm, proposed by Jones *et al.* [8], is one of the most popular methods for expensive optimization using the Gaussian stochastic process model. It optimizes the expected improvement by a branch-and-bound algorithm. Much effort has been made to incorporate the Gaussian stochastic process model and other response surface methods [3], [9], [10] into evolutionary algorithm (EA) frameworks for single-objective optimization [11]–[18].

In the case of multiobjective optimization, the goal is to find a number of well-representative Pareto optimal solutions. A few attempts have been made to introduce the Gaussian stochastic process model into evolutionary multiobjective optimization [19]–[25]. Keane [22] and Emmerich *et al.* [24] have generalized the probability of improvement and the expected improvement to multiobjective optimization. Single-objective EAs are used for maximizing one of these metrics for determining which point should be evaluated next. Although one can select multiple test points with high-metric values at each iteration (e.g., [24]), these scalar metrics on their own could not predict the most likely shape and location of the whole PF. Therefore, they might not be stochastically sound for locating multiple test points. It should be pointed out that it could be possible in theory, although very difficult in practice, to generalize these metrics for measuring the merit of evaluating several points at the same time.

ParEGO, proposed by Knowles [25], 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 100–250 function evaluations is very promising. The major drawback of the ParEGO algorithm 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. S-Metric Selection based EGO (SMS-EGO) [26] extends the idea of Emmerich *et al.* [24]. It optimizes the *S*-metric, a hypervolume-based metric, by the Covariance matrix adaptation evolution strategy (CMS-ES) algorithm to decide which point will be evaluated next. Like ParEGO, SMS-EGO

Manuscript received February 18, 2009; revised June 1, 2009. First version published December 15, 2009; current version published May 28, 2010.

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

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

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TEVC.2009.2033671

evaluates only one single test point at each iteration. In [21], Jeong and Obayashi used NSGA-II [27] to optimize the expected improvements of all the individual objectives in an 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 full use of statistical information provided by the Gaussian stochastic 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 produce several test points at each iteration. This paper represents an attempt along this line.

Multiobjective evolutionary algorithm based on decomposition (MOEA/D) [28] is a recent multiobjective evolutionary algorithm framework. Like multiobjective genetic local search [29]–[31] and multiple single objective Pareto sampling [32], it is based on conventional aggregation approaches. MOEA/D decomposes an 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 [28], [33]–[36].

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 stochastic 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 are as follows.

1) Due to the parallel nature of MOEA/D, MOEA/D-EGO is able to generate several test points to evaluate in a batch way, which makes it suitable in parallel computing environments.
2) 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.
3) A fuzzy clustering-based modeling method is used. It aims to improve the prediction quality without significantly increasing the computational cost.
4) 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 follows. Section II introduces MOPs. Section III presents the algorithm framework. Sections IV–VI give 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:

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

where $-\infty < a_i < b_i < +\infty$ for all $i = 1, \dots, n$. $F : \prod_{i=1}^{n}[a_i, b_i] \rightarrow R^m$ consists of m individual objective functions f_i, $i = 1, \dots, m$. 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 $u = (u_1, \dots, u_m)^T$, $v = (v_1, \dots, v_m)^T \in R^m$ be two vectors, u is said to dominate v if $u_i \leq v_i$ for all $i = 1, \dots, m$, 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 [37].

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 an MOP into a number of single-objective optimization subproblems [37]–[40]. In the following, we introduce two most commonly used approaches.

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

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

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 Pareto set (PS) is not convex, the weighted sum approach could be not able to find some Pareto optimal solutions.

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

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

where $z^* = (z_1^*, \ldots, z_m^*)$ is the ideal point in the objective space, i.e.,

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

for each $i = 1, \ldots, m$. Under some mild condition [37], 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 stochastic 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 ith 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;
- 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 $P_{\text{eval}} = \{x^1, \ldots, x^{K_I}\}$.

Step 2 Stopping Condition: If a predetermined 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 Stochastic Process Modeling

1) *Assumptions:* To build a cheap surrogate model for an expensive function $y = g(x)$, $x \in R^n$, Gaussian stochastic process modeling makes the following assumptions [2], [8].

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

$$\mu + \epsilon(x) \quad (5)$$

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 with constant mean μ and constant variance σ^2.

b) 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

$$c(x, x') = \exp[-d(x, x')] \quad (6)$$

where

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

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

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 the Gaussian stochastic process modeling can be found in [41].

2) *Hyperparameter Estimation:* Given K points $x^1, \ldots, x^K \in R^n$ and their g-function values $y^1, \ldots, y^K$, the hyperparameters 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 $x = x^i$ $(i = 1, \ldots, K)$ [8]

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

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).

Equation (8) is multimodal [8], 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 [42]. The control parameters in DE are set as:

a) the population size=20;
b) the number of generations=50;
c) the crossover rate=0.1;
d) the value of F=0.5.

3) *The Best Linear Unbiased Prediction and Predictive Distribution:* Given hyperparameter 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 [2], [8]

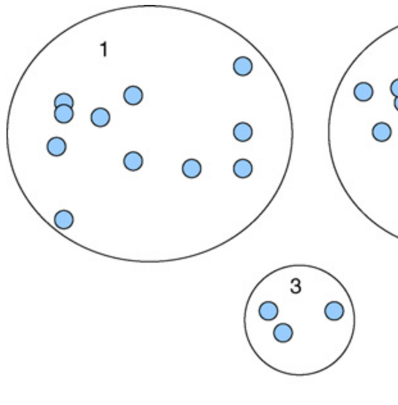
$$\hat{y}(x) = \hat{\mu} + r^T C^{-1} (y - \mathbf{1}\hat{\mu}) \tag{11}$$

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 $i = 1, \ldots, K$.

4) *Fuzzy Clustering-based Method for Modeling (FuzzyCM):* The computational overhead in estimating the hyperparameters and in computing $\hat{y}(x)$ and $s^2(x)$ depends mainly on K. When K is large, it could be computationally unbearable to optimize the hyperparameters if all the points are directly used [41]. 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 [43], [44]. Obviously, this strategy does not 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 [45]–[47]. 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.



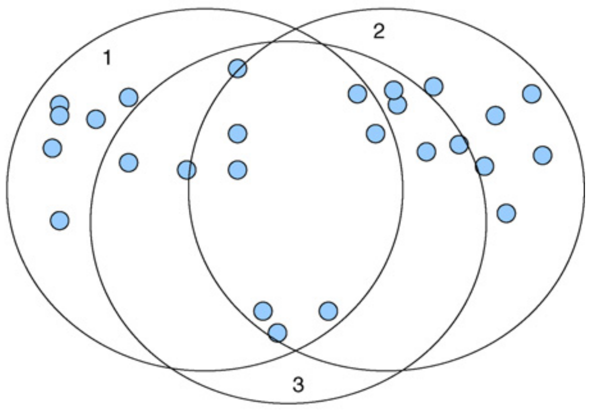


Fig. 1. In crisp clustering, all the points are clustered into several clusters of different sizes, and each point belongs to just one cluster. In FuzzyCM used in this paper, all the clusters are of the same size and one point might belong to several different clusters. (a) Crisp clustering. (b) FuzzyCM.

We propose a fuzzy clustering [48] based modeling method (FuzzyCM) to relieve this drawback. FuzzyCM needs two control parameters L_1 and L_2 where $L_1 > L_2$.

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

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.

a) Step 1 Set the number of clusters

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

b) Step 2 By applying the Fuzzy C-means (FCM) 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.

c) 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 hyperparameters.

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 hyperparameter 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 FCM clustering on all the points. Each P_j contains exactly L_1 points which are most likely to belong to cluster 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, or 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 $g^{ws}(x|\lambda)$.

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

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 directly from [49] (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

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

where

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

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

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

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

$$\varphi(t) = (2\pi)^{-1/2}\exp(-t^2/2)$$

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 [49], 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

$$\begin{aligned} & g^{te}(x|\lambda) = \\ & \max\{\max\{\lambda_i(f_1(x) - z_1^*), \lambda_i(f_2(x) - z_2^*)\}, \lambda_3(f_3(x) - z_3^*)\} \end{aligned} \tag{20}$$

we can, following [49], 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 three objectives before solving it since the number of solutions required for approximating the PF of an 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 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 [8]

$$E[I(x)] = [(g_{\min} - \hat{y}(x)]\Phi(\tfrac{g_{\min}-\hat{y}}{\hat{s}(x)}) + \hat{s}(x)\phi(\tfrac{g_{\min}-\hat{y}}{\hat{s}(x)}). \tag{21}$$

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

$E[I(x)]$ is monotonically decreasing with respect to $\hat{y}(x)$ and increasing with respect to $\hat{s}(x)$ [8]. Note that $\hat{y}(x)$ is the predicted value of $g(x)$ and $\hat{s}(x)$ measures the prediction uncertainty, maximizing $E[I(x)]$ balances exploitation and exploration in a sense. The expected improvement has been used in the famous EGO with great success [8].

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 [33], 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.

1) 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.
2) Each solution $\tilde{x}^i$ obtained in Step 4 of MOEA/D-EGO is for maximizing ξ^i and therefore its evaluation is good for optimizing its 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^j are close.
3) 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.

```
1) Step 1  Set $Q = \emptyset$.
           For $i = 1$ to $N$,
           If $\tilde{x}^i$ is different from any points in $P_{\text{eval}} \cup Q$,
           then add $\tilde{x}^i$ to $Q$.
           End For
2) 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.
3) Step 3  Select the point with the highest $\xi$-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 MOPs 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.

1) 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.
2) As shown in [26], SMS-EGO performs very well on several test instances. It generates only one single test point at each generation. The major difference between SMS-EGO and ParEGO is that the S-metric used in SMS-EGO is defined as the hypervolume contribution of a new point to an estimated PF while the metric in ParEGO is the EI value of a scalar function.

A. Test Instances

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

1) ZDT1 [50]
$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \sqrt{f_1(x)/g(x)}\right]$$
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.

2) ZDT2 [50]
$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - (f_1(x)/g(x))^2\right]$$
where $g(x)$ and the range and dimensionality of x are the same as in ZDT1. The PF of ZDT2 is nonconvex.

3) ZDT3 [50]
$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \sqrt{f_1(x)/g(x)} - \frac{f_1(x)}{g(x)}\sin(10\pi x_1)\right]$$
where $g(x)$ and the range and dimensionality of x are the same as in ZDT1. Its PF is disconnected.

4) ZDT4 [50]
$$f_1(x) = x_1$$
$$f_2(x) = g(x)\left[1 - \sqrt{f_1(x)/g(x)}\right]$$
where
$$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}\left[x_i^2 - 10\cos(4\pi x_i)\right]$$
and $x = (x_1, \ldots, x_n)^T \in [0,1] \times [-5,5]^{n-1}$. It has many local PFs.

5) ZDT6 [50]
$$f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$$
$$f_2(x) = g(x)\left[1 - (f_1(x)/g(x))^2\right]$$
where
$$g(x) = 1 + 9\left[(\sum_{i=2}^{n} x_i)/(n-1)\right]^{0.25}$$
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 PF in the objective space.

6) LZ08-F1 [33]
$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}[x_j - x_1^{0.5+\frac{3(j-2)}{2(n-2)}}]^2$$
$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}[x_j - x_1^{0.5+\frac{3(j-2)}{2(n-2)}}]^2$$
where
$$J_1 = \{j | j \text{ is odd and } 2 \le j \le n\}$$
$$J_2 = \{j | j \text{ is even and } 2 \le j \le n\}$$
and $x = (x_1, \ldots, x_n)^T \in [0,1]^n$. Unlike ZDT test instances, the PS of LZ08-F1 is a nonlinear curve
$$x_j = x_1^{0.5+\frac{3(j-2)}{2(n-2)}}, \quad j = 2, \ldots, n$$
where $x_1 \in [0,1]$.

7) LZ08-F2 [33]
$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - \sin(6\pi x_1 + \frac{j\pi}{n}))^2 \tag{22}$$
$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - \sin(6\pi x_1 + \frac{j\pi}{n}))^2 \tag{23}$$
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 $x_1 \in [0,1]$.

8) LZ08-F3 [33]
$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - 0.8x_1\cos(6\pi x_1 + \frac{j\pi}{n}))^2$$
$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - 0.8x_1\sin(6\pi x_1 + \frac{j\pi}{n}))^2$$
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 $x_1 \in [0,1]$.

9) LZ08-F4 [33]
$$f_1 = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}[x_j - 0.8x_1\cos(2\pi x_1 + \frac{j\pi}{3n})]^2$$
$$f_2 = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}[(x_j - 0.8x_1\sin(6\pi x_1 + \frac{j\pi}{n})]^2$$
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 $x_1 \in [0,1]$.

10) DTLZ2 [51]
$$f_1(x) = (1 + g(x))\cos(x_1\pi/2)\cos(x_2\pi/2)$$
$$f_2(x) = (1 + g(x))\cos(x_1\pi/2)\sin(x_2\pi/2)$$
$$f_3(x) = (1 + g(x))\sin(x_1\pi/2)$$
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 $f_i \ge 0, i = 1, 2, 3$.

11) KNO1 [25]

$$f_1(x) = 20 - r\cos(\phi)$$
$$f_2(x) = 20 - r\sin(\phi)$$

where

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

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.

12) VLMOP2 [52]

$$f_1(x) = 1 - \exp(-\sum_{i=1,\dots,n}(x_i - 1/\sqrt{n})^2)$$
$$f_2(x) = 1 + \exp(-\sum_{i=1,\dots,n}(x_i - 1/\sqrt{n})^2)$$

where $x = (x_1, \dots, x_n)^T \in [-2,2]^n$. 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, $i = 1, \dots, n$.

B. Experimental Settings

1) General Settings

a) *The number of decision variables (n)*: It is set to be 2 for KNO1 and VLMOP2, 8 for all the other two-objective test instances, and 6 for DTLZ2.

b) *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 [53].

c) *The maximal number of function evaluation*: 300 for DTLZ2 and 200 for all the two-objective instances.

d) *The number of independent runs*: We run each algorithm for each test instance 10 times independently.

2) MOEA/D-EGO

a) *The number of function evaluation at each generation*: K_E is set to be 5 in all the test instances.

b) *The number of subproblems N and weight vectors* $\lambda^1, \dots, \lambda^N$: They are controlled by an integer H. More precisely, $\lambda^1, \dots, \lambda^N$ are all the weight vectors in which each individual weight takes a value from

$$\left\{\frac{0}{H}, \frac{1}{H}, \dots, \frac{H}{H}\right\}.$$

Therefore, the number of subproblems (weight vectors) is

$$N = C^{m-1}_{H+m-1}$$

where m is the number of objectives. H is set to be 299 for all the two-objective test instances, and 33 for the three-objective instances. Consequently, N is 300 for the two-objective instances and 595 for the three-objective instances.

c) *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

a) All the parameter settings of these two methods are the same as in [25] and [26], respectively.

b) The source codes of these two methods used in our experimental studies are obtained from their authors.

C. Performance Metrics

The inverted generational distance (IGD) metric [28] and hypervolume difference (I_H^-) metric [54] are used in assessing the performance of the algorithms in our experimental studies.

1) *IGD Metric:* 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 on the PF and let these points be P^* for each test instance with two objectives, and 990 points for each test instance with three objectives.

2) I_H^- *Metric:* The I_H^- metric is defined as

$$I_H^-(P^*, P) = I_H(P^*) - I_H(P)$$

where $I_H(P)$ is the hypervolume between the set P and a reference point.

Both the IGD metric and the I_H^- metric measure convergence and diversity. To have low IGD and I_H^- values, P must be close to the PF and cannot miss any part of the whole PF.

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 eight variables with 200 function evaluations. Actually, SMS-EGO takes about 3 h for a single run with 200 function evaluation even on problems with two variables on a computer with 2.8 GHz. For this reason, we test SMS-EGO only on two test instances with two variables in our experiments. The approximation set generated by each algorithm in a single run is all the nondominated solutions found during the run.

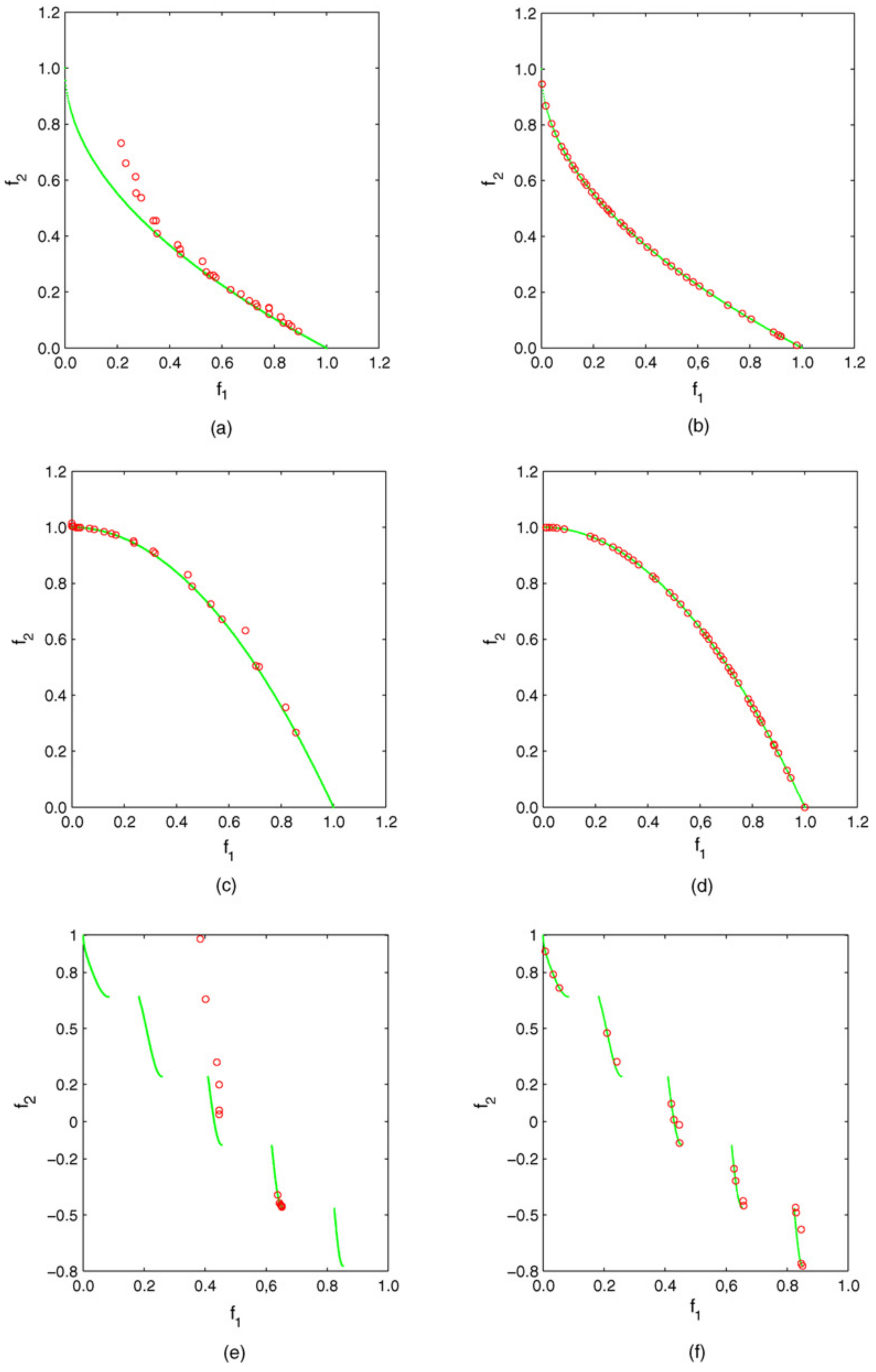


Fig. 2. Plots of the final approximations with the lowest IGD values in the objective space on ZDT1, ZDT2, and ZDT3. (a) ParEGO: ZDT1. (b) MOEA/D-EGO: ZDT1. (c) ParEGO: ZDT2. (d) MOEA/D-EGO: ZDT2. (e) ParEGO: ZDT3. (f) MOEA/D-EGO: ZDT3.

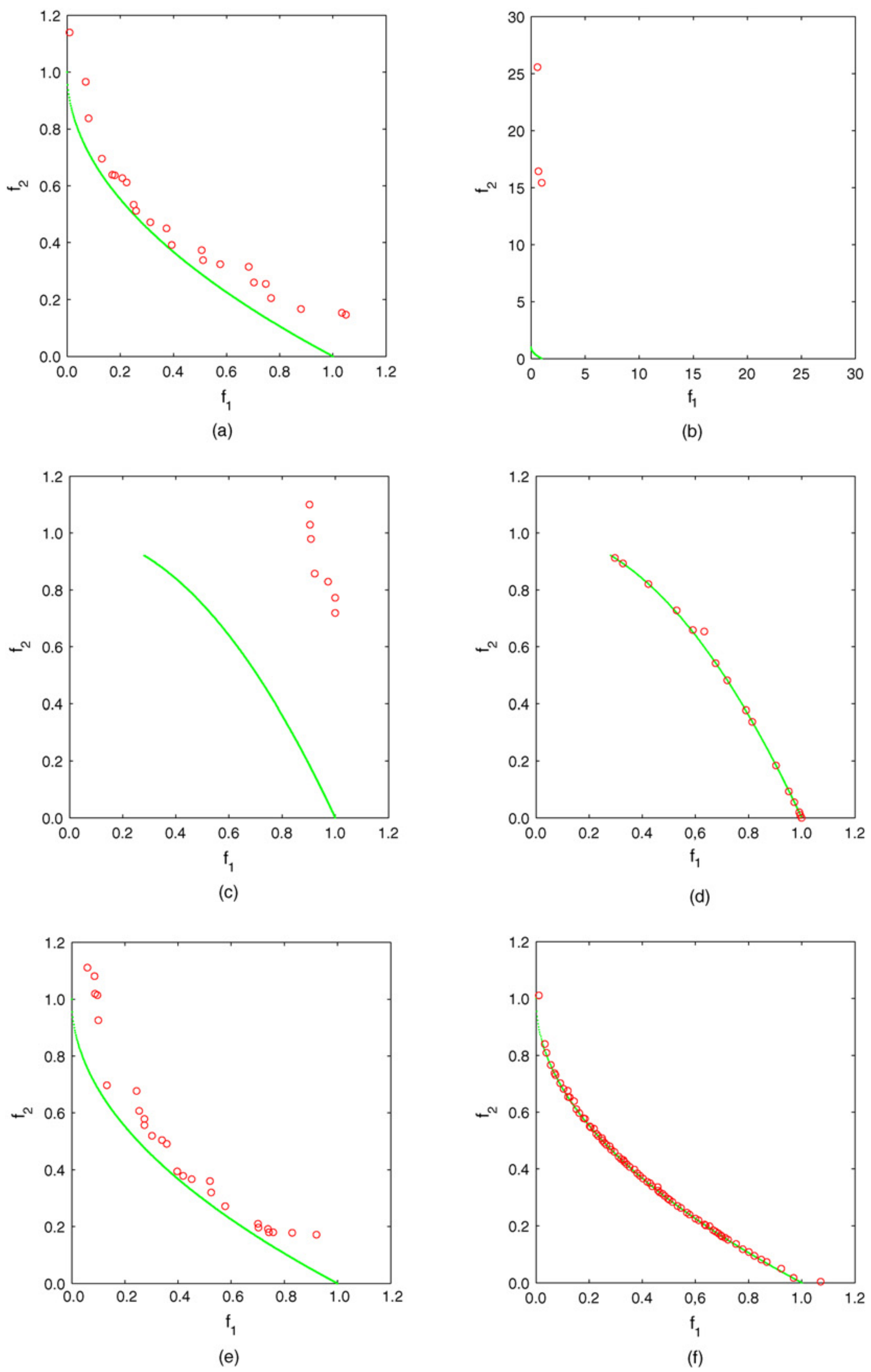


Fig. 3. Plots of the final approximations with the lowest IGD values in the objective space on ZDT4, ZDT6, and LZ08-F1. (a) ParEGO: ZDT4. (b) MOEA/D-EGO: ZDT4. (c) ParEGO: ZDT6. (d) MOEA/D-EGO: ZDT6. (e) ParEGO: LZ08-F1. (f) MOEA/D-EGO: LZ08-F1.

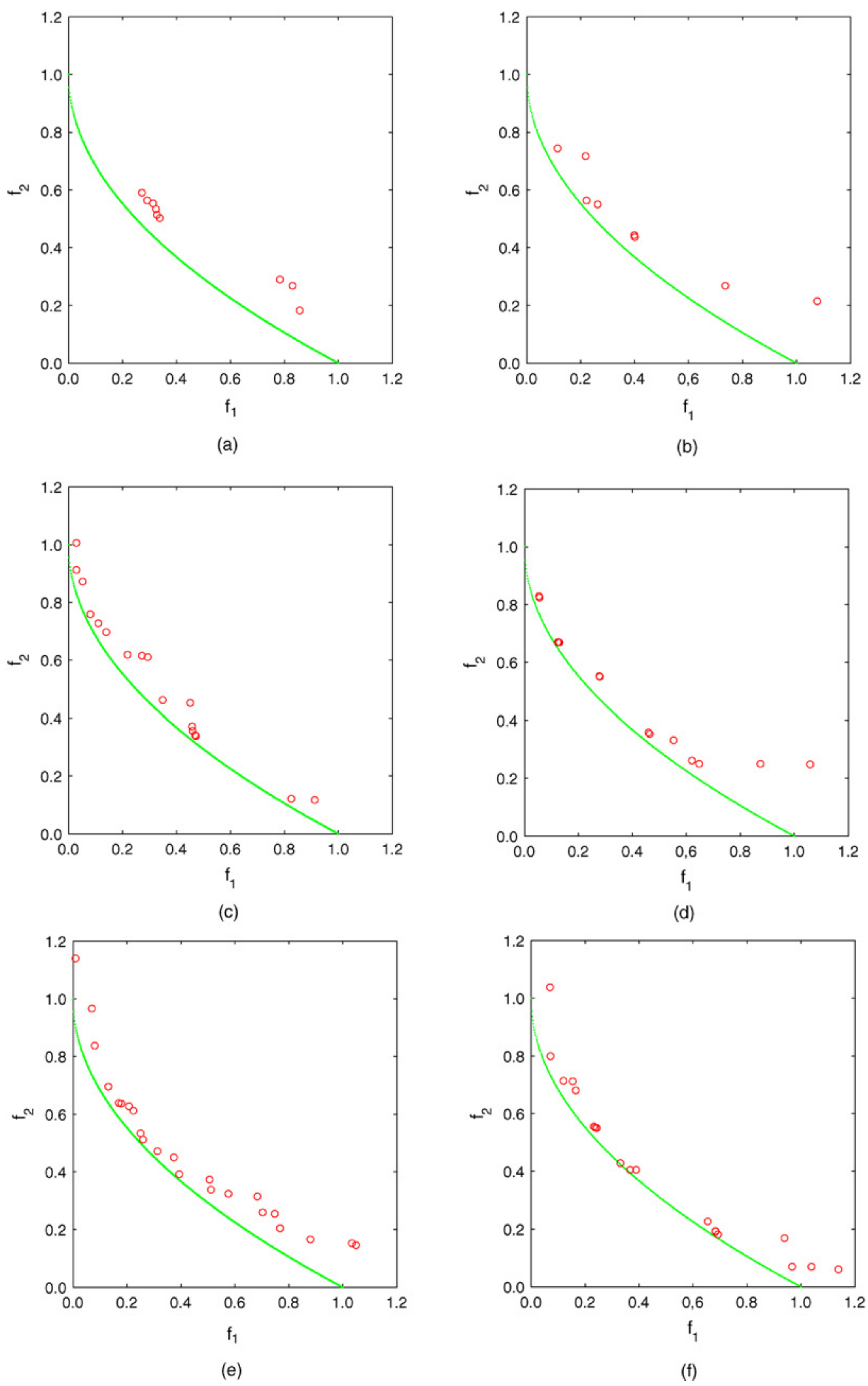
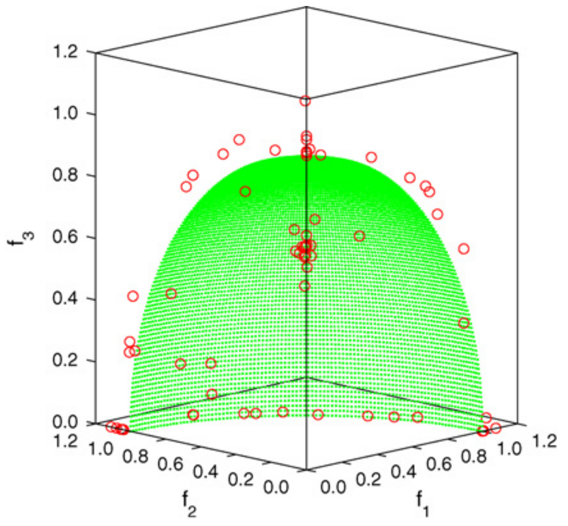
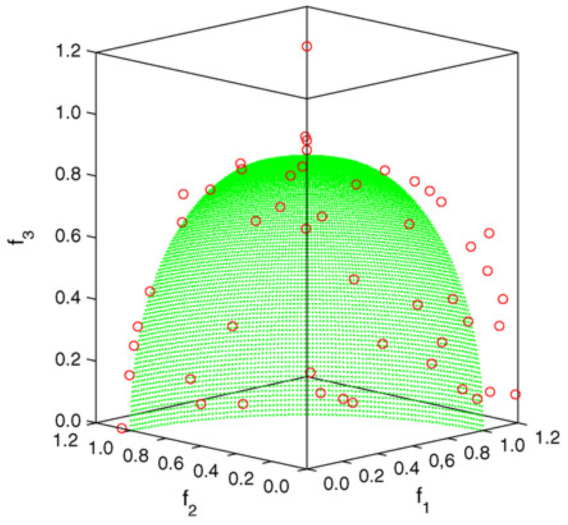


Fig. 4. Plots of the final approximations with the lowest IGD values in the objective space on LZ08-F2, LZ08-F3, and LZ08-F4. (a) ParEGO: LZ08-F2. (b) MOEA/D-EGO: LZ08-F2. (c) ParEGO: LZ08-F3. (d) MOEA/D-EGO: LZ08-F3. (e) ParEGO: LZ08-F4. (f) MOEA/D-EGO: LZ08-F4.



(a)



(b)

Fig. 5. Plots of the final approximations with the lowest IGD values in the objective space on DTLZ2. (a) ParEGO: DTLZ2. (b) MOEA/D-EGO: DTLZ2.

1) *Comparison:* Tables I and II present the statistics of the IGD and I_H^- values of the approximation sets obtained by each algorithm in ten independent runs, respectively. Figs. 2–6 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 Tables I and II that MOEA/D-EGO significantly outperforms ParEGO in terms of both the IGD and I_H^- metrics 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 DTLZ2, there is no much difference between ParEGO and MOEA/D-EGO in terms of both metrics. Actually, it is very difficult to judge which method is significantly better from the plots of the final solutions on these four instances.

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

On KNO1 and VLMOP2, Both the IGD and I_H^- metrics 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 no 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 than ParEGO and SMS-EGO.

E. More Discussions

1) *The Effect of the Gaussian Model:* MOEA/D-EGO uses the Gaussian stochastic process model in MOEA/D for dealing with expensive MOPs within a very tight computational budget. A question which naturally arises is the effect of the Gaussian model in MOEA/D-EGO. To address this question, we have run the version of MOEA/D proposed in [28] with population size = 20 and neighborhood size = 5 on ZDT1 and ZDT2. The number of function of evaluations is set to be 200, the same as in MOEA/D-EGO on these two instances in our experimental studies in Section VII-B. Fig. 8 plots the final approximations with the lowest IGD values among ten independent runs in the objective space obtained by MOEA/D-EGO and MOEA/D. It is evident that MOEA/D-EGO significantly outperforms MOEA/D on these two test instances.

2) *FuzzyCM Versus One Single Model:* MOEA/D-EGO uses FuzzyCM to reduce the overhead in Gaussian modeling. Training points in FuzzyCM are clustered into several small data sets, each of which is then used for building a model. One would like to know how much more CPU time will be incurred and how much the algorithm performance can be improved if all the training points are used to built one single model. To address these issues, we modify MOEA/D-EGO by using one single Gaussian model learnt from all the available points and test the resultant algorithm, MOEA/D-EGO with single model, on ZDT1 and ZDT2. All its other settings are the same as in MOEA/D-EGO. The average IGD values of the final approximations obtained by MOEA-D-EGO with single model for these two instances are 0.0048 and 0.0058, respectively. Fig. 9 plots the final approximations with the lowest IGD values among ten independent runs in the objective space obtained by these two versions of MOEA/D-EGO for these two instances. We can conclude that MOEA/D-EGO with single model does performance better in terms of solution quality. The average CPU times among ten independent runs used by two algorithms on these two instances are presented in Table III. Clearly, MOEA/D-EGO with single model is much

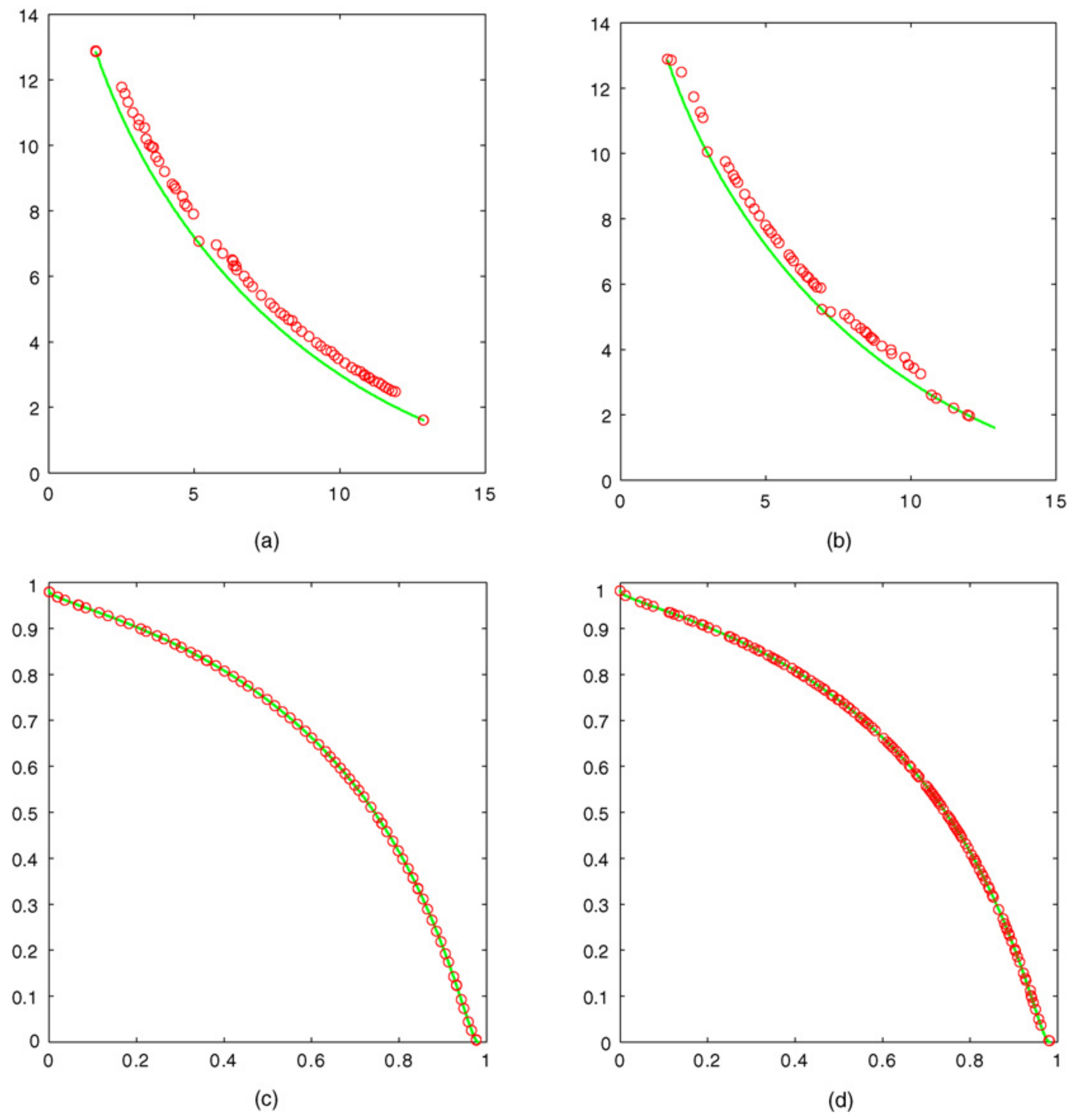


Fig. 6. Plots of the final approximations with the lowest IGD values in the objective space on KNO1 and VLMOP2. (a) SMS-EGO: KNO1. (b) MOEA/D-EGO: KNO1. (c) SMS-EGO: VLMOP2. (d) MOEA/D-EGO: VLMOP2.

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

Instance	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

Std, stands for standard deviation.

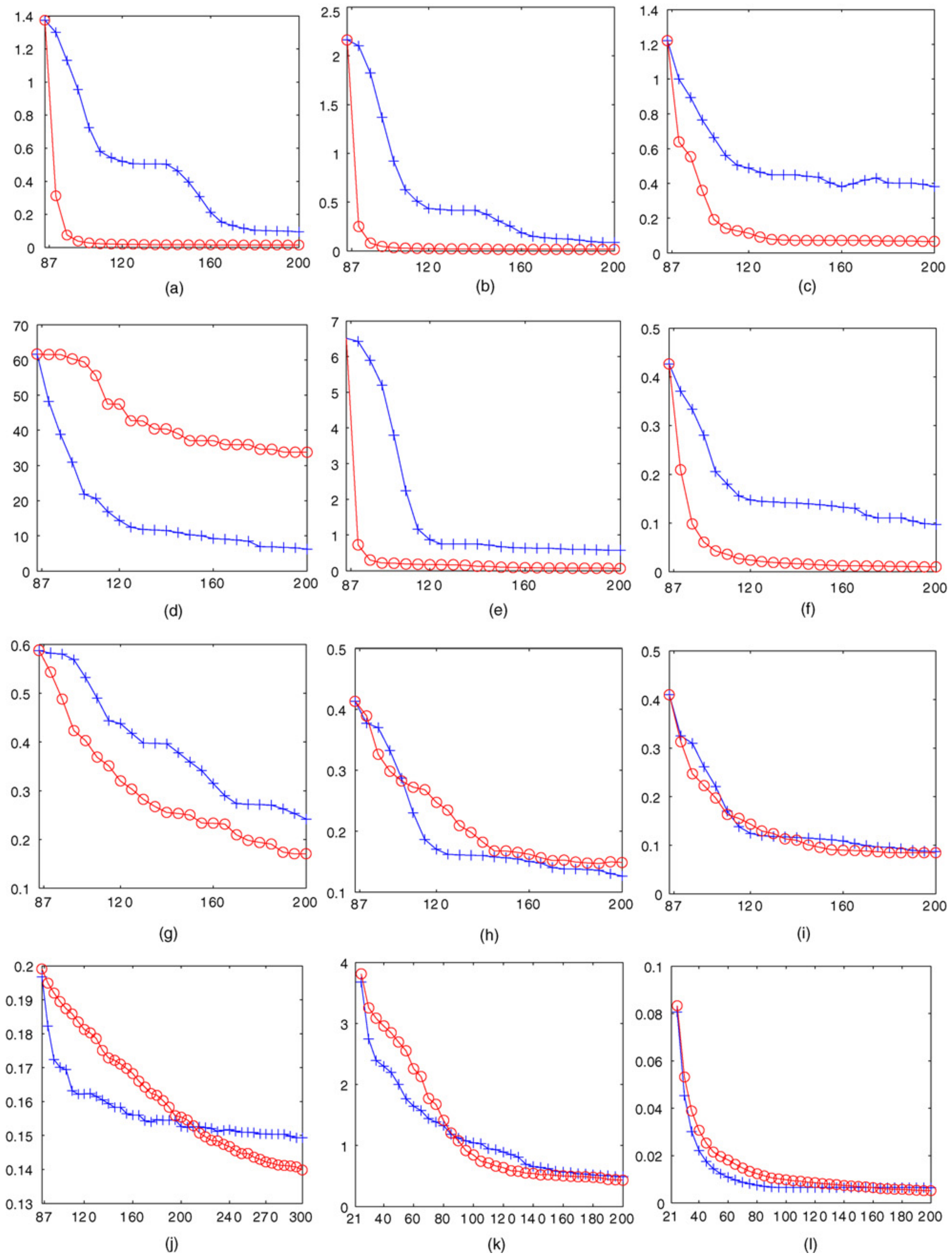


Fig. 7. Evolution of the mean of IGD values versus the number of function evaluations. (a)–(j) are for comparison between MOEA/D-EGO and ParEGO, and (k) and (l) for MOEA/D-EGO and SMS-EGO. The lines marked with "o" are for MOEA/D-EGO. The lines marked with "+" are for ParEGO in (a)–(j), and for SMS-EGO in (k) and (l). (a) ZDT1. (b) ZDT2. (c) ZDT3. (d) ZDT4. (e) ZDT6. (f) LZ08-F1. (g) LZ08-F2. (h) LZ08-F3. (i) LZ08-F4. (j) DTLZ2. (k) KNO1. (l) VLMOP2.

TABLE II

STATISTICS OF THE I_H^- VALUES OF THE FINAL APPROXIMATION SETS OBTAINED BY MOEA/D-EGO, PAREGO, AND SMS-EGO WITH THE SAME NUMBER OF FUNCTION EVALUATIONS

Instance	Reference Point	ParEGO			MOEA/D-EGO		
		Lowest	Mean	Std	Lowest	Mean	Std
ZDT1	(10, 10)	0.4653	0.9998	0.2543	0.0182	0.0672	0.0801
ZDT2	(10, 10)	0.0767	0.6377	0.7147	0.0160	0.0198	0.0035
ZDT3	(20, 20)	1.1874	4.9230	2.9147	0.0308	0.1178	0.0707
ZDT4	(50, 50)	166.87	336.34	197.35	785.08	1709.52	593.51
ZDT6	(10, 10)	8.1609	9.0669	1.0610	0.0386	0.1000	0.1009
LZF1	(10, 10)	1.5043	1.8733	0.2548	0.0993	0.1547	0.0405
LZF2	(10, 10)	1.5286	3.0632	1.1807	1.8166	2.8784	0.6833
LZF3	(10, 10)	0.9184	1.7880	0.5060	1.5683	2.5061	0.4284
LZF4	(10, 10)	0.6197	1.3613	0.4451	0.7402	1.8971	0.6182
DTLZ2	(10, 10, 10)	0.1892	0.2199	0.0163	0.2801	0.4845	0.1244
		SMS-EGO			MOEA/D-EGO		
KNO1	(20, 20)	7.3591	15.2214	6.1403	6.8166	14.0601	7.7610
VLMOP2	(10, 10)	0.0066	0.0205	0.0146	0.0053	0.0228	0.0117

Std, stands for standard deviation. Reference Point: The reference point used in computing I_H^-.

TABLE III

AVERAGE CPU TIMES (IN SECONDS) USED BY MOEA/D-EGO WITH SINGLE MODEL AND MOEA/D-EGO WITH FUZZCM

Instance	Single Model	FuzzyCM
ZDT1	4248	936
ZDT2	4428	1260

TABLE IV

RE VALUE OF f_2 IN ALL THE TEST INSTANCES

Instance	RE
ZDT1	0.000154
ZDT2	0.0000786
ZDT3	0.000330
ZDT4	0.00304
ZDT6	0.0000687
LZF1	0.000370
LZF2	0.00329
LZF3	0.00668
LZF4	0.00101
KNO1	0.00201
VLMOP2	0.00000471
DTLZ2	0.00162

slower than MOEA/D-EGO with FuzzyCM. Since the computational complexity of model building increases cubically with the number of training points [24], MOEA/D-EGO with single model could soon become impractical as the number of function evaluations increases.

3) *Why Some Instances Are Harder Than Others:* 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 stochastic process modeling is better on ZDT2 than on ZDT4. To verify it, we build Gaussian predictive models for f_2 in all the test instances. In our modeling:

a) the number of decision variables n is set to be 2 for KNO1 and VLMOP2, 8 for all the other two objective test instances, and 6 for DTLZ2;
b) the points for model building are generated by the Latin Square method;

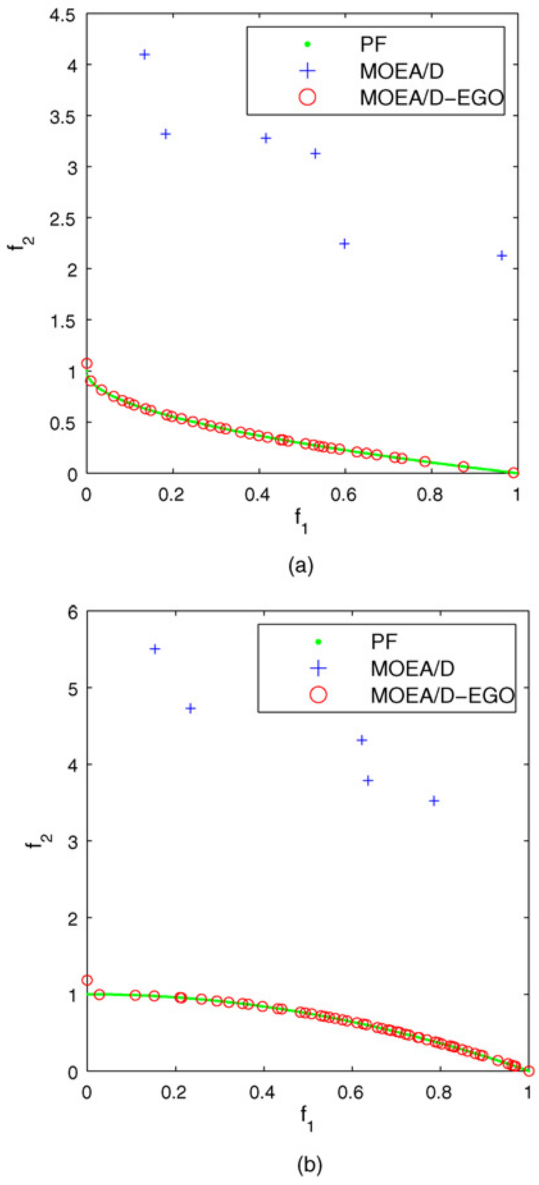


Fig. 8. Plots of the final approximations with the lowest IGD values in the objective space on ZDT1 and ZDT2 obtained by MOEA/D-EGO and MOEA/D. (a) ZDT1. (b) ZDT2.

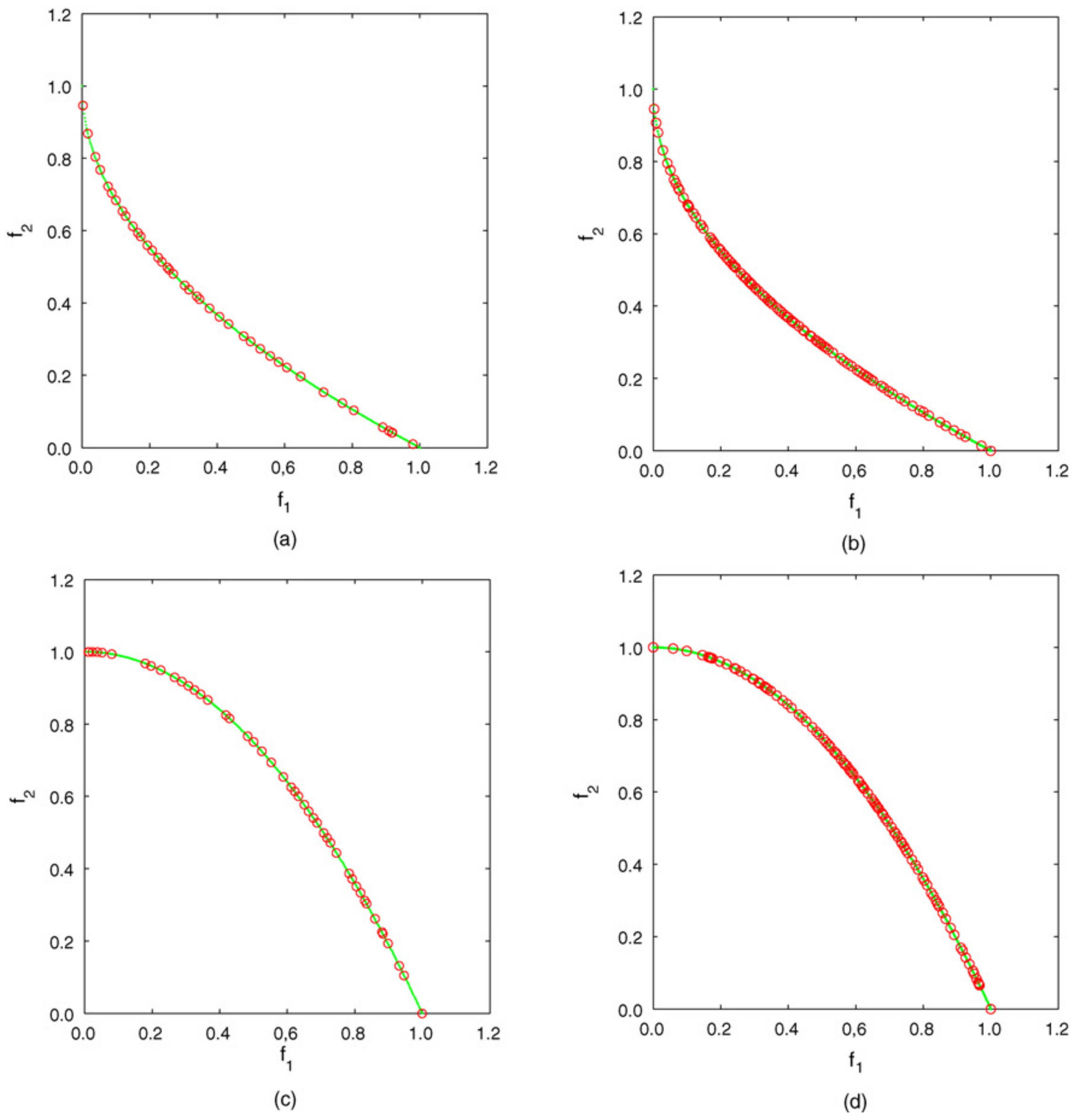


Fig. 9. Plots of the final approximations with the lowest IGD values in the objective space on MOEA/D-EGO with FuzzyCM,MOEA/D-EGO with single model (ZDT1), and MOEA/D-EGO with FuzzyCM,MOEA/D-EGO with single model (ZDT2) obtained by MOEA/D-EGO with single model and FuzzyCM. (a) MOEA/D-EGO with FuzzyCM: ZDT1. (b) MOEA/D-EGO with single model: ZDT1. (c) MOEA/D-EGO with FuzzyCM: ZDT2. (d) MOEA/D-EGO with single model: ZDT2.

c) the number of the points for model building is $11n - 1$.

Then, for each test instance, we randomly select 5000 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 values are presented in Table IV.

It is evident that an MOP with a high-RE value tends to be hard for the algorithms. Actually, the RE value is 0.0000786 for ZDT2 while it is 0.00304 for ZDT4. It implies that Gaussian stochastic process modeling might not be suitable for ZDT4. One should introduce other techniques to deal with it [3]. Tables II–IV also imply that the prediction quality of the model is not an only factor determining the algorithm performances. For example, the RE value in ZDT6 is very low while its IGD values in both algorithms are relatively high.

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 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 in a batch way, 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 MOPs. Like other MOEA/D algorithms, it decomposes an 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 stochastic 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 no 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 stochastic 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.
         • $\alpha$ and $\| * \|$ used in computing the objective $J$,
           and $\varepsilon$ 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 [48].

Appendix B
The Maximum of Several Normally Distributed Variables [49]

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 [28]

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;
  • $\delta$ : 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 $\lambda^i$.
    Step 1.2 Generate $N$ points by uniformly randomly sampling from $\prod_{i=1}^{n} [a_i, b_j]$ and compute their
    $\xi^i$-function values, $i = 1, \ldots, N$. Set $\tilde{x}^i$ to be the
    point with the largest $\xi^i$-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 [33]. More discussions about MOEA/D-DE can be found in [33].

References

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

[2] 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.

[3] 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.

[4] 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.

[5] K. Grave, J. Ramon, and L. Raedt, "Active learning for high throughput screening," in *Proc. 11th Int. Conf. Discov. Sci.*, Berlin, Heidelberg, Germany, 2008, pp. 185–196.

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

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

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

[11] 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.

[12] H. Ulmer, F. Streichert, and A. Zell, "Evolution strategies assisted by Gaussian processes with improved preselection criterion," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Canberra, Australia, Dec. 2003, pp. 692–699.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput. Fusion Found. Method. Appl.*, vol. 9, no. 1, pp. 3–12, Jan. 2005.

[19] B. S. Yang, Y. S. Yeun, and W. S. Ruy, "Managing approximation models in multiobjective optimization," *Struct. Multidis. Optim.*, vol. 24, no. 2, pp. 141–156, Sep. 2002.

[20] H.-S. Chung and J. J. Alonso, "Multiobjective optimization using approximation model-based genetic algorithms," in *Proc. 10th Am. Instit. Aeronaut. Astronaut./Int. Soc. Struct. Multidis. Optim. Symp. Multidisciplinary Anal. Optim.*, New York, Sep. 2004.

[21] S. Jeong and S. Obayashi, "Efficient global optimization (EGO) for multiobjective problem and data mining," in *Proc. IEEE Congr. Evol. Comput.*, vol. 3. Edinburgh, U.K., Sep. 2005, pp. 2138–2145.

[22] A. J. Keane, "Statistical improvement criteria for use in multiobjective design optimization," *Am. Instit. Aeronaut. Astronaut. J.*, vol. 44, no. 4, pp. 879–891, Apr. 2006.

[23] M. K. Karakasis and K. C. Giannakoglou, "On the use of metamodel-assisted, multiobjective evolutionary algorithms," *Eng. Optim.*, vol. 38, no. 8, pp. 941–957, Dec. 2006.

[24] 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.

[25] 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.

[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. 10th Parallel Problem Solving Nat.*, Dortmund, Germany, Sep. 2008, pp. 784–794.

[27] 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.

[28] 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.

[29] H. Ishibuchi and T. Murata, "A multiobjective 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.

[30] 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.

[31] 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.

[32] E. Hughes, "Multiple single objective Pareto sampling," in *Proc. IEEE Congr. Evol. Comput.*, vol. 4. Canberra, Australia, Dec. 2003, pp. 2678–2684.

[33] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 12, no. 2, pp. 284–302, Apr. 2009.

[34] P. C. Chang, S. H. Chen, Q. Zhang, and J. L. Lin, "MOEA/D for flowshop scheduling problems," in *Proc. IEEE Congr. Evol. Comput.*, Hong Kong, Jun. 2008, pp. 1433–1438.

[35] W. Peng, Q. Zhang, and H. Li, "Comparison between MOEA/D and NSGA-II on the multiobjective travelling salesman problem," in *Multiobjective Memetic Algorithms* (Studies in Computational Intelligence Series), vol. 171, C.-K. Goh, Y.-S. Ong, and K. C. Tan, Eds. Berlin, Germany: Springer-Verlag, 2009, pp. 309–324.

[36] 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 Int. Conf. Devoted Evol. Multicriterion Optim.*, Nantes, France, Apr. 2009, pp. 438–452.

[37] K. Miettinen, "A posteriori methods: Part 2," in *Nonlinear Multiobjective Optimization*, Norwell, MA: Kluwer, 1999, pp. 77–95.

[38] A. Pascoletti and P. Serafini, "Scalarizing vector optimization problems," *J. Opt. Theory Appl.*, vol. 42, no. 4, pp. 499–524, Apr. 1984.

[39] M. Ehrgott, "Scalarization techniques," in *Multicriteria Optimization*. New York: Springer-Verlag, 2005, pp. 97–121.

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

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

[42] K. V. Price, R. M. Storn, and J. A. Lampinen, "Differential evolution algorithms," in *Differential Evolution: A Practical Approach to Global Optimization*, New York: Springer-Verlag, 2005, pp. 37–47.

[43] S. Seo, M. Wallat, T. Graepel, and K. Obermayer, "Gaussian process regression: Active data selection and test point rejection," in *Proc. IEEE–Int. Neural Netw. Soc.–Eur. Neural Netw. Soc. Int. Joint Conf. Neural Netw.*, vol. 3. Jul. 2000, pp. 241–246.

[44] N. Lawrence, M. Seeger, and R. Herbrich, "Fast sparse Gaussian process methods: The informative vector machine," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 15. British Columbia, Canada, Dec. 2002, pp. 609–616.

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

[46] A. Choudhury, P. B. Nair, and A. J. Keane, "A data parallel approach for large-scale gaussian process modeling," in *Proc. 2nd Soc. Ind. Appl. Math. Int. Conf. Data Mining*, Arlington, VA, Apr. 2002, pp. 95–111.

[47] A. Schwaighofer and V. Tresp, "Transductive and inductive methods for approximate Gaussian process regression," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 15. British Columbia, Canada, Dec. 2002, pp. 977–984.

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

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

[50] K. Deb, "Multiobjective genetic algorithms: Problem difficulties and construction of test problems," *Evol. Comput.*, vol. 7, no. 3, pp. 205–230, Feb. 1999.

[51] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multiobjective optimization test problems," in *Proc. IEEE Congr. Evol. Comput.*, vol. 1. Honolulu, HI, May 2002, pp. 825–830.

[52] D. A. van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in *Proc. Assoc. Comput. Mach. Symp. Appl. Comput.*, San Antonio, TX, Feb. 1999, pp. 351–357.

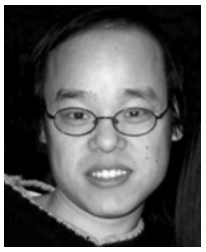
[53] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 42, no. 1, pp. 55–61, Feb. 2000.

[54] 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.

Qingfu Zhang (M'01–SM'06) received the B.S. degree in mathematics from Shanxi University, Shanxi, China, in 1984, and the M.S. degree in applied mathematics and the Ph.D. degree in information engineering from Xidian University, Xi'an, China, in 1991 and 1994, respectively.

He is currently a Professor with the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K. From 1994 to 2000, he was with the National Laboratory of Parallel Processing and Computing, National University of Defense Science and Technology, China, Hong Kong Polytechnic University, Kowloon, Hong Kong, the German National Research Center for Information Technology (now Fraunhofer-Gesellschaft, Germany), and the University of Manchester Institute of Science and Technology, Manchester, U.K. He holds two patents and is the author of more than 70 research publications. His main research interests include evolutionary computation, optimization, neural networks, data analysis, and their applications.

Dr. Zhang is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION and the IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS–PART B. He is also an Editorial Board Member of three other international journals. He won the Unconstrained Multiobjective Optimization Algorithm Competition at the Congress of Evolutionary Computation in 2009, and has received the 2010 IEEE Transactions on Evolutionary Computation Outstanding Paper Award.



Wudong Liu received the B.S. and M.S. degrees in computer science from Wuhan University, Wuhan, China, in 2001 and 2004, respectively. He is currently working toward the Ph.D. degree at the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K.

From 2004 to 2005, he worked as a Research Engineer with the State Key Laboratory of Software Engineering, Wuhan University. His main research interests include multiobjective optimization, evolutionary algorithms, machine learning, intelligent scheduling, and also software engineering.

Edward Tsang (M'05) received the Bachelor degree in business administration from the Chinese University of Hong Kong, Shatin, Hong Kong, in 1977, and the M.Sc. and Ph.D. degrees in computer science from the University of Essex, Colchester, U.K., in 1983 and 1987, respectively.

He is currently a Professor in Computer Science at the School of Computer Science and Electronic Engineering, University of Essex, Colchester, U.K., where he is also the Co-Founder and the Director of the Center for Computational Finance and Economic Agents, an interdisciplinary research center. He was with companies including British Telecom, London, U.K., Honda Europe, Ghent, Belgium, and OANDA, NY. Main techniques used in his research include heuristic search, optimization, and evolutionary computation. With background and experience in industry, he has broad interest in business applications of artificial intelligence, including computational finance, computational economics, and constraint satisfaction.

Prof. Tsang was the Founding Chair of the IEEE Computational Finance and Economics Technical Committee in which he is an Active Member.

Botond Virginas received the M.Eng. degree in automation and computer science from the Technical University of Cluj-Napoca, Napoca, Romania, in 1990, and the M.S. degree in environmental engineering from the University of Manchester, Manchester, U.K., in 1995. He received the Ph.D. degree in computer science from Portsmouth University, Portsmouth, U.K., in 1999.

From 1997 to 2002, he was a Senior Lecturer in Computer Science at the School of Computing, Staffordshire University, Staffordshire, U.K. Since 2003, he has been working as a Principal Research Professional with Intelligent Systems Research Center, BT Exact, Adastral Park, Ipswich, U.K. He has been working on various projects investigating the applications of AI and optimization technologies to field resource management. He currently works on a project where he addresses the personalization of next-generation communication and information services. His research interests include intelligent agents, data mining, planning, scheduling, machine learning, and knowledge-based systems.

Dr. Virginas is a Member of the British Computer Society, a Chartered Engineer, and an Overseas Public Body Member of the Hungarian Academy of Sciences.