A Surrogate-assisted Reference Vector Guided Evolutionary Algorithm for Computationally Expensive Many-objective Optimization

Tinkle Chugh, Yaochu Jin, *Fellow, IEEE*, Kaisa Miettinen, Jussi Hakanen, Karthik Sindhya

***Abstract*—We propose a surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive optimization problems with more than three objectives. The proposed algorithm is based on a recently developed evolutionary algorithm for many-objective optimization that relies on a set of adaptive reference vectors for selection. The proposed surrogate-assisted evolutionary algorithm uses Kriging to approximate each objective function to reduce the computational cost. In managing the Kriging models, the algorithm focuses on the balance of diversity and convergence by making use of the uncertainty information in the approximated objective values given by the Kriging models, the distribution of the reference vectors as well as the location of the individuals. In addition, we design a strategy for choosing data for training the Kriging model to limit the computation time without impairing the approximation accuracy. Empirical results on comparing the new algorithm with the state-of-the-art surrogate-assisted evolutionary algorithms on a number of benchmark problems demonstrate the competitiveness of the proposed algorithm.**

***Index Terms*—multiobjective optimization, reference vectors, surrogate-assisted evolutionary algorithms, model management, Kriging, computational cost**

I. Introduction

Many industrial optimization problems have multiple objectives to be optimized and these objectives are typically conflicting in nature, i.e. improvement in one objective will lead to deterioration of at least one of the other objectives. Such problems are known as multiobjective optimization problems (MOPs). In this paper, we consider MOPs in the following form :

$$\begin{aligned} &\text{minimize } \{f_1(x), \ldots, f_k(x)\} \\ &\text{subject to } x \in S \end{aligned} \tag{1}$$

with $k(\geq 2)$ objective functions $f_i(x)$: $S \rightarrow \Re$. The vector of objective function values is denoted by $f(x) = (f_1(x), \ldots, f_k(x))^T$. The (nonempty) feasible space S is a subset of the decision space $\Re^n$ and consists of decision vectors $x = (x_1, \ldots, x_n)^T$ that satisfy all the constraints. The image of the feasible region S in the objective space $\Re^k$ is called the feasible objective set denoted by Z. The elements of Z are called feasible objective vectors denoted by $f(x)$ or $z = (z_1, \ldots, z_k)^T$, where $z_i = f_i(x), i = 1, \ldots, k$, are the objective function values. As the objectives are conflicting, there typically does not exist a single optimal solution, but multiple so-called Pareto optimal solutions. The set of all optimal solutions in the objective space is called the Pareto front and in the decision space the Pareto set.

A large number of optimization methods have been reported in the literature. These methods can be classified into two main different fields, namely, multiple criteria decision making (MCDM) [33] and evolutionary multiobjective optimization (EMO) [10], [13]. Methods either find a representative set of Pareto optimal solutions or the most preferred solution for a decision maker and they differ in the way the solutions are obtained. For instance, in the MCDM community, an MOP is often scalarized into a single objective optimization problem. Moreover, a decision maker is usually involved in the solution process to identify preferred solutions. On the other hand, EMO algorithms work with a population of candidate solutions and often aim to find a set of solutions representing the whole Pareto front. The decision maker is involved usually after a set of Pareto optimal solution is found [42].

Evolutionary algorithms (EAs) have become popular in past decades due to several advantages. For example, they have the ability to handle different kinds of decision variables e.g. binary, integer, real or mixed and they do not assume any convexity or differentiability of objective functions and/or constraints involved. Despite of these advantages, EAs are often criticized because of slow convergence and a large number of function evaluations needed before an acceptable solution can be found. For instance, in aerodynamic optimization, one function evaluation involving computational fluid dynamics simulations may take substantial amount of time and it will become computationally prohibitive to use an EA to solve aerodynamic optimization problems.

One of the popular approaches to reduce computation time in evolutionary optimization is to introduce approximations, especially function approximation. Computational models for functional approximation are often known as surrogates and EAs using objective values estimated by surrogates are often referred to as surrogate-assisted evolutionary algorithm (SAEAs). A surrogate is also known as a metamodel in the literature, which can in part replace the computationally expensive objective functions. For more details about SAEAs, see [9], [24], [43].

Although numerous algorithms have been proposed in using surrogates in an EA, many challenges remain. One is the choice of the surrogate, as different types of surrogate techniques exist in the literature, e.g. Kriging, neural networks, support vector regression and polynomial approximation. Nevertheless, there is no simple rule for choosing the right type

This work was supported by the FiDiPro project DeCoMo funded by the Finnish Funding Agency for Innovation (TEKES). *(Corresponding author: Yaochu Jin)*

[1]Tinkle Chugh, Yaochu Jin, Kaisa Miettinen, Jussi Hakanen and Karthik Sindhya are with the Faculty of Information Technology, University of Jyvaskyla, Finland. Yaochu Jin is also with the Department of Computer Science, University of Surrey, Guildford, United Kingdom. Email: [first-name.last-name]@jyu.fi, yaochu.jin@surrey.ac.uk.

of surrogates for approximating the given computationally expensive objective or constraint functions. A second challenge is how to use a surrogate, i.e. what to approximate using the surrogate. The most conventional way is to approximate the objective or constraint functions. In addition, a surrogate can be used to estimate the rank of individuals [31], or some other quality measure, e.g. distance to the nondominated solutions [38], [39], [40], or hypervolume [2]. The third challenge is the computational cost for constructing the surrogate, which is often neglected in SAEAs. In practical, training a surrogate may take a substantial amount of time, and the main aim of reducing computation time will be jeopardized. The fourth challenge is how to update the surrogate i.e. how to choose individuals in the current population to be re-evaluated using the original functions. Different ways exist in the literature for selecting the individuals, e.g. selecting a set of best solutions [25] or nondominated solutions [17] according to the surrogate and selecting a set of representative solutions [27]. If the Kriging model is used, one can select solutions that maximize the expected improvement [47], the probability of improvement [12] and hypervolume improvement [18]. Selection of individuals to be re-evaluated is also called updating criterion or infilling criterion. The fifth challenge is to determine when the surrogate needs to be updated. For instance, it may be possible that a surrogate is accurate enough and does not need to be updated even if new training samples are available.

In SAEAs, which individuals are to be re-evaluated using the original objective functions, how to update the surrogate and when to update the surrogate are referred to as model management, which is also known as evolution control [26]. In [26], two methods were mentioned for managing the surrogate, i.e., fixed evolution control and adaptive evolution control. In fixed evolution control, updating the surrogate is based on a prefixed criterion, while in adaptive evolution control, a surrogate is updated based on its performance.

As pointed out in [9], little work has been reported on using SAEAs for solving computationally expensive problems having more than three objectives. During the years 2008-2015, only three algorithms [6], [38], [41] have been tested on multi-objective benchmark problems with more than three objectives. While many industrial problems, e.g., optimization of the controller of a hybrid car [36], involve more than three computationally expensive objectives, surrogate-assisted evolutionary optimization of many-objective problems has not attracted much attention in the evolutionary computation community and SAEAs developed so far cannot be directly extended to many-objective optimization. Therefore, our work is an effort to fill this gap.

Apart from the challenges resulting from the large number of objectives, it is notoriously difficult to achieve high-quality surrogates for large scale optimization problems due to the curse of dimensionality. For this reason, the number of decision variables SAEAs have handled is by far up to 50. According to a recent survey [9], only seven SAEAs have been tested on optimization problems with more than 20 decision variables and six of them were benchmarks. Note that SAEAs using neural networks as the surrogate were tested on more than 20 variables, while SAEAs using Kriging models as the surrogate have been used to solve problems with up to eight decision variables. This can be attributed to the fact that the computational time for training the Kriging model will become too long when the number of training samples increases [35].

This paper focuses on developing an efficient SAEA for solving computationally expensive many-objective optimization problems. One of the major reasons limiting the applicability of existing algorithms to many-objective optimization is the lack of an efficient surrogate management method suited for the evolutionary algorithm used. In SAEAs when managing the surrogates, individuals should be selected by taking into account of both convergence and diversity. To select such individuals, surrogates need to be seamlessly embedded into the evolutionary algorithm. Most existing SAEAs are dominance based and thus are not well suited for handling many objectives. Therefore, the major contribution of the paper is to propose an efficient algorithm to manage the surrogates for handling a large number of objectives. To this end, we adopt the reference vector guided evolutionary algorithm (RVEA) [8] for many-objective optimization to be used as an evolutionary algorithm. Two sets of reference vectors adaptive and fixed, together with uncertainty information from the Kriging models as well as the location of the individuals are exploited for surrogate management. To limit the computation time for training the Kriging models, a strategy for choosing training samples is proposed so that the maximum number of training data is fixed.

The rest of the paper is organized as follows. In Section II, we provide a relatively detailed description of RVEA as well as Kriging models so that the paper is self-contained. The Kriging assisted RVEA, called K-RVEA is introduced in Section III. Section IV presents the numerical results of K-RVEA on benchmark problems and compared them with a few state-of-the-art SAEAs. Finally, conclusions are drawn and future work briefly discussed in Section V.

II. Background

In this section, we first summarize main components of RVEA, which we use as the underlying evolutionary algorithm. Next, we present a brief description of Kriging, including a discussion about its advantages and disadvantages over other surrogate models.

A. Reference vector guided evolutionary algorithm

Two major difficulties in solving problems with high number of objectives are convergence to the Pareto front and maintaining a good diversity between solutions. Several evolutionary algorithms have been proposed for solving many-objective optimization problems, by, for instance, using a revised dominance relationship, decomposing the multi-objective optimization into several single objective optimization problems, an indicator-based objective function, or using reference points. For more details about these algorithms and challenges in solving problems with more than three objectives, see [23], [29], [45].

RVEA is an EMO algorithm most recently developed for many-objective optimization [8]. While MOEA/D [50] and

NSGA-III [14] use a set of weights and reference points, respectively, RVEA adopts a set of reference vectors. The main difference between RVEA and the MOEA/D and NSGA-III lies in its selection strategy. In RVEA, selection is based on a criterion known as angle penalized distance (APD), which is used to manage both convergence and diversity. It has been shown [8] that APD is better scalable to the increase in the number of objectives in maintaining a balance between convergence and diversity. APD relies on a set of reference vectors, which partitions the objective space into a number of subspaces, where selection of individuals is performed independently. The main components of RVEA are presented in Algorithm 1.

Algorithm 1 RVEA

Input: t_{max} = maximum number of generations; N = number of reference vectors; $V_0 = \{v_{01}, v_{02}, \dots v_{0N}\}$ a set of unit reference vectors;

Output: nondominated solutions from population $P_{t_{max}}$

1: Create an initial population P_0 of size N randomly and set generation counter t = 0
2: **while** $t < t_{max}$ **do**
3: Generate offspring Q_t
4: Combine parent and offspring populations, $L_t = P_t \cup Q_t$
5: Select parents (P_{t+1}) for the next generation
6: Update reference vectors (V_{t+1})
7: **end while**

RVEA uses elitism and offspring generation strategies similar to other state-of-the-art EMO algorithms such as NSGA-II [15] and NSGA-III [14]. RVEA distinguishes itself with NSGA-III in its selection strategy and the adaptation of reference vectors, which are Steps 5 and 6 in Algorithm 1. In the following, we present the four main components of RVEA, i.e., generation of reference vectors, assignment of individuals to reference vectors, selection and adaptation of reference vectors.

1) Generation of reference vectors: RVEA uses a set of reference vectors in the objective space to guide the search process. To generate a uniformly distributed set of reference vectors, first a set of uniformly distributed reference points (p) is generated on a unit hyperplane using the canonical simplex-lattice design method [11], [7].

$$\begin{cases} p_i = (p_i^1, p_i^2, \dots, p_i^k), \\ p_i^j = \left\{ \frac{0}{H}, \frac{1}{H}, \dots \frac{H}{H} \right\}, \sum_{j=1}^{k} p_i^j = 1 \end{cases} \quad (2)$$

where $i = 1, 2, \dots, N$ with N being the number of uniformly distributed points, k is the number of objectives and H is a positive integer used in simplex-lattice design. An example is shown in Figure 1 for a biobjective optimization problem, where the dots represent the reference points generated on a unit hyperplane. The corresponding reference vector is then obtained by projecting the reference points from the hyperplane to a hypersphere

$$v_i = \frac{p_i}{||p_i||}. \quad (3)$$

where $||p_i||$ represents the L_2-norm of p_i. As a result, these

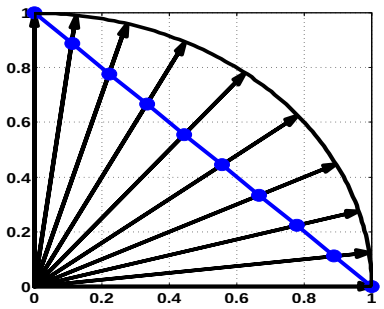

Figure 1. An illustrative example of reference vectors for a biobjective optimization problem.

reference vectors partition the objective space into a number of subspaces. In the next subsection, we describe how the individuals in a population are assigned to these reference vectors and how the population is partitioned into different subpopulations.

2) Assignment of individuals to reference vectors: After the generation of the reference vectors, individuals are assigned to them as follows. First, objective values of all individuals at the current generation are are translated, i.e. $\bar{f}_i^j = f_i^j - f_i^*$, where f_i^j represents the objective value of f_i for the j^{th} individual and f_i^* the minimum objective values of f_i at the current generation. We denote the translated objective vector by $\bar{f} = (\bar{f}_1, \bar{f}_2, \dots, \bar{f}_k)$. Translation of objective functions ensures that the initial point of reference vectors is always the origin and all the translated objective values are inside the positive orthant. After the translation, the acute angle is measured between an individual and all the reference vectors. For instance, let us consider the situation shown in Figure 2, with two reference vectors v_i and v_{i+1}, and three individual $\bar{f}^1$, $\bar{f}^2$ and $\bar{f}^3$. As the angle θ_i^1 between the individual $\bar{f}^1$ and the reference vector v_i is less than the angle θ_{i+1}^1 between the individual and the other reference vector v_{i+1}, this individual is assigned to the first reference vector v_i. Similarly, $\bar{f}^2$ and $\bar{f}^3$ will be assigned to reference vector v_i and v_{i+1}, respectively. Therefore, an individual is assigned to a reference vectors if and only if the angle between it and the reference vector is minimum among all reference vectors. In this way, assignment of individuals to the reference vectors partitions the population into subpopulations. Other notations in Figure 2 are used in the next subsection.

3) Selection mechanism in each subpopulation: After the generation of reference vectors and the assignment of individuals them, one individual is selected from each subpopulation (Step 5 in Algorithm 1). The selection criterion consists of two subcriteria that are meant for managing convergence and diversity, respectively. Convergence is taken care by the distance between the translated objective vector and the origin. As selection is performed in each subpopulation independently, let us take the subpopulation corresponding to reference vector v_i. In the illustrative example shown in Figure 2, two individuals $\bar{f}^1$ and $\bar{f}^2$ are assigned to the reference vector v_i and the distance between them and the origin is denoted by $||\bar{f}^1||$ and $||\bar{f}^2||$, respectively. As the aim is to find solutions closer to the origin, individual $\bar{f}^1$ is preferred over individual $\bar{f}^2$ and

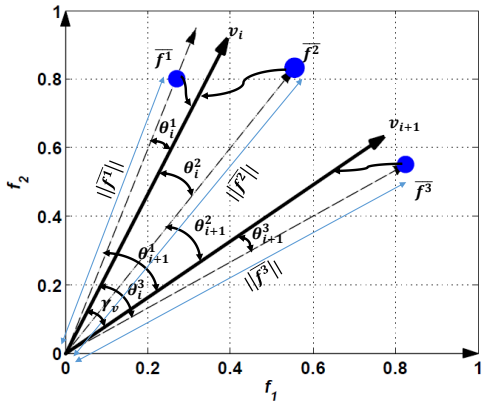

Figure 2. An illustration for assignment of individuals to a reference vector assignment and calculation of the selection criteria.

will therefore be selected for this subpopulation.

In RVEA, diversity is accounted by the angle between the translated objective vector and the reference vector the individual is assigned to. The individual with the smallest angle is preferred over other individuals. For instance, for reference vector v_i in Figure 2, individual $\bar{f}^1$ with the angle θ_i^1 is preferred over individual $\bar{f}^2$ as $\theta_i^1 < \theta_i^2$.

To combine the two subcriteria for convergence and diversity, the following angle penalized distance (APD) is defined:

$$d^j = (1 + P(\theta^j)) \cdot ||\bar{f}^j||, \tag{4}$$

where $||\bar{f}^j||$ is the distance from the translated objective vector corresponding to the j^{th} individual to the origin, and θ^j is the angle between the j^{th} individual and the reference vector it is assigned to. In APD, $P(\theta^j)$ is the penalty function defined as follows:

$$P(\theta_i) = k \cdot (\frac{t}{t_{max}})^{\alpha} \cdot \frac{\theta^j}{\gamma_v}, \tag{5}$$

where γ_v is defined as the smallest angle between the reference vectors v_i and its closest neighboring reference vector v_j i.e. $\gamma_v = min_{i \in \{1,\dots,N\}, i \neq j} \langle v_i, v_j \rangle$. The angle γ_v is used to normalize the angles and is important when the distribution of the reference vectors is either too dense or too sparse. As highly dense or sparse reference vectors may generate a very small or a very large angle between the individual and the reference vector, normalization of the angle can be helpful to alleviate this problem. Parameter α is used to change the rate of the penalty function $P(\theta)$. The rate of the penalty function is used to emphasize convergence at the early stage and diversity at the later stage of the evolutionary search process. For instance, at the early stage of solution process, convergence is preferred to push the individuals closer to the Pareto front. Once individuals have converged to the Pareto front, diversity is then preferred by distributing individuals along the Pareto front. Therefore, the rate of the penalty function depends on the current generation number t, the maximum number of generations t_{max} and parameter α used in (5). As careful empirical studies for setting the parameter α have been performed in [8], we use the same parameter setting in this work. After calculating APD for all individuals in each subpopulation, one individual with the minimum APD value is selected from each subpopulation for the next generation.

4) Adaptation of reference vectors: In order to find a set of uniformly distributed nondominated individuals as close to the Pareto front as possible. For some optimization problems, e.g. those in the WFG test suite [22] where objective functions are scaled to different ranges, a uniformly distributed set of reference vectors is not best suited for getting uniformly distributed individuals. To tackle this issue, one possible solution is to adapt the reference vectors. In RVEA, reference vectors are adaptive. In other words, they change their position according to the location of individuals in the objective space. The adaptation of reference vectors for the next generation (v_{t+1}) is applied in the following way:

$$v_{t+1,i} = \frac{v_{0,i} \circ (z_t^{max} - z_t^{min})}{||v_{0,i} \circ (z_t^{max} - z_t^{min})||}, \tag{6}$$

where $\circ$ represents the Hadamard product [1] that multiplies two matrices of the same size element-wise, $v_{0,i}$ is the uniformly generated reference vector in the initialization phase in RVEA and z_t^{max} and z_t^{min} are the maximum and minimum values of each objective function in the t^{th} generation, respectively. The adaptation of reference vectors ensures that a set of uniformly distributed nondominated individuals will be obtained even for optimization problems whose objective functions have different ranges.

B. Kriging

We use Kriging, also known as Gaussian process to approximate each objective function. As per the survey on computationally expensive multiobjective optimization problems [9], Kriging has been frequently used for surrogate techniques, mainly because it is able to deliver uncertainty information of the approximated values, which is very useful in managing surrogates [24]. In this work, we use uncertainty information from Kriging models to update the surrogates, which will be further discussed in the next section.

Kriging approximates the objective function value of an individual x as

$$y(x) = \mu(x) + \epsilon(x), \tag{7}$$

where μ is the prediction of a regression model $F(\beta, x)$ i.e. $\mu = F\beta$ and $\epsilon(x)$ is a Gaussian distribution of zero mean and the standard deviation σ

$$\epsilon(x) = N(0, \sigma^2). \tag{8}$$

The regression model $F(\beta, x) = \beta_1 g_1(x) + \ldots + \beta_l g_l(x)$ is a linear combination of l chosen functions with coefficients β.

To get an approximated value from (7) for any new input, Kriging model needs to be trained using training samples, which are the pre-evaluated individuals in SAEAs. Let matrix $X = \left[x^1, \ldots, x^{N_I}\right]^T$ represent the training data in the decision space with their corresponding objective vector $y = \left[y^1, \ldots, y^{N_I}\right]^T$, where $i = 1, 2, \ldots, N_I$ represents the number of samples or the size of the training data. One should note that the size of X is $N_I \times n$, where n is the number of decision variables, i.e., $x^i = [x_1, \ldots, x_n]$ for $i = 1, \ldots, N_I$.

For any two arbitrary inputs x^i and x^j, the covariance between two random processes $\epsilon(x^i)$ and $\epsilon(x^j)$ is defined by

$$cov[\epsilon(x^i), \epsilon(x^j)] = \sigma^2 \mathbf{R}([R(x^i, x^j)]), \tag{9}$$

where $\mathbf{R}$ is the correlation matrix of size $N_I \times N_I$

$$\mathbf{R} = \begin{bmatrix} R(x^1, x^2) & \cdots & R(x^1, x^{N_I}) \\ \vdots & \ddots & \vdots \\ R(x^{N_I}, x^1) & \cdots & R(x^{N_I}, x^{N_I}) \end{bmatrix} \tag{10}$$

and $R(x^i, x^j)$ is the correlation function between $\epsilon(x^i)$ and $\epsilon(x^j)$. The commonly used correlation function is

$$R(x^i, x^j) = exp(-\sum_{k=1}^{n} \theta_k |x_k^i - x_k^j|^2), \tag{11}$$

where $\theta_i, i = 1, \ldots, N_I$ denote the hyperparameters.

For a new input $\bar{x}$, an approximated value from (7) can be written as

$$\bar{y} = \beta + r^T(\bar{x})\mathbf{R}^{-1}(y - F\beta), \tag{12}$$

where y contains the values of given N_I individuals, $r(\bar{x})$ is the correlation vector of size N_I between the new input $\bar{x}$ and the training data $\left[x^1, \ldots, x^{N_I}\right]$ i.e.

$$r^T(\bar{x}) = \left[R(\bar{x}, x^1), \ldots, R(\bar{x}, x^{N_I})^T\right]. \tag{13}$$

To obtain a new approximated value $\bar{y}$, we need to specify coefficients β and hyperparameters θ. Equation (12) has the generalized least square solution,

$$\beta = (F^T\mathbf{R}^{-1}F)^{-1}F^T\mathbf{R}^{-1}y \tag{14}$$

and the estimated variance σ^2 is given by

$$\sigma^2 = \frac{1}{N_I}(y - F\beta)^T\mathbf{R}^{-1}(y - F\beta). \tag{15}$$

Values of θ are obtained by maximizing the likelihood function

$$\psi(\theta) = -\frac{1}{2}(N_I \ln \sigma^2 + \ln det(\mathbf{R})) \tag{16}$$

where *det*($\mathbf{R}$) is the determinant of the correlation matrix $\mathbf{R}$. After getting hyperparameters θ, coefficients β and the variance σ^2 are calculated from (14) and (15), respectively, which are further used to approximate the objective function value from (12).

While Kriging is a very attractive surrogate model due to its ability to deliver uncertainty information, it also suffers from potentially serious weaknesses resulting from the computational complexity for training surrogates. As indicated in [20], the computational complexity of training Kriging is $O(n^3)$, where n is the number of training samples. The issue of high computational complexity will become worse if the hyperparameters θ are determined by maximize the likelihood function using an optimization algorithm, which has often been done in Kriging assisted SAEAs. For instance, MOEA/D-EGO uses differential evolution [44] and SMS-EGO employs covariance matrix adaptation (CMA-ES) [19] to optimize the hyperparameters, while in ParEGO, the Nelder and Mead algorithm [37] is used.

In this work, we use a modification of Hookes and Jeeves algorithm [21], which is implemented in DACE toolbox [30]. The main reason is that it is not practical to use population based techniques to optimize the hyperparameters due to the prohibitively high computation time thus incurred, since in SAEAs, Kriging models need to be frequently re-trained. We will provide a brief empirical comparison in Section IV.

III. Surrogate-assisted reference vector guided evolutionary algorithm

Model management is crucial to the success of surrogate-assisted evolutionary algorithms [24]. It is mainly concerned with how to use and update surrogates, including choosing individuals to be re-evaluated using the original objective functions. These re-evaluated individuals can then be used as training data for updating (retraining) the surrogate. Both convergence and diversity have to be taken into account in selecting individuals to be re-evaluated, which becomes more difficult for problems with a large number of objectives. In this paper, we focus on selection of training data in such a way that both convergence and diversity are managed given a large number of objectives, which is one major contribution of this paper.

The computation time for training the surrogate depends on the size of the training data set and the type of the surrogate used. The Kriging model is widely used due to its unique property of being able to predict with an error bound. Unfortunately, the computation time for training Kriging models will become prohibitive when the number of training data is large. Therefore, the second contribution of this paper is the proposal of a strategy to choose training samples so that the size of the training data can be kept sufficiently small. To this end, we use an archive to store the training data for updating the Kriging model.

The proposed Kriging-assisted reference vector guided evolutionary algorithm, called K-RVEA, is presented in Algorithm 2. This algorithm consists of three main phases, the initialization phase followed by the phase where a surrogate is used and finally updated in the last phase. Algorithm 2 is composed of two other algorithms, Algorithm 3 and Algorithm 4. Algorithm 3 selects the individuals for re-evaluation, and therefore also for updating the surrogate, while Algorithm 4 manages the training data in archive A_1. In addition to the training archive A_1, a second archive A_2 is used to store non-nondominated solutions as the final solution set.

A. Initialization

In the initialization phase, an initial population is generated e.g. using the Latin hypercube sampling [32]. These individuals are evaluated with the original expensive functions and added to two archives A_1 and A_2. Individuals stored in A_1 are used to build a Kriging model for each objective function.

B. Using the surrogate

In the phase of using the surrogate, we use Kriging models instead of the original functions to calculate objective function values. Kriging models are used for fitness evaluations for a prefixed number of generations without updating them.

Algorithm 2 K-RVEA

Input: FE_{max}, maximum number of expensive function evaluations; u = number of individuals to be re-evaluated and to be used for updating Kriging models; w_{max} = prefixed number of generations before updating Kriging models;

Output: nondominated solutions of all evaluated ones from A_2

/*Initialization*/

1: Create an initial population of size N_I using e.g. the Latin hypercube sampling, initialize the number of function evaluations- *FE* = 0, the generation counter for using Kriging models w = 1 and a counter for the number of updates, $t_u = 0$. Initialize two empty archives A_1 and A_2, i.e. $|A_1| = |A_2| = \phi$

2: Evaluate the initial population with the original functions and add them to A_1 and A_2, update *FE* = *FE*+N_I, update $|A_1| = |A_1| + N_I$ and $|A_2| = |A_2| + N_I$

3: Train a Kriging model for each objective function by using individuals in A_1

4: **while** *FE* $\leq$ FE_{max}

/*Using the surrogate*/

5: **while** $w \leq w_{max}$

6: Run Steps 3-6 of Algorithm 1 with Kriging models instead of the original functions and update *w*=*w*+1

7: **end while**

/*Updating the surrogate*/

8: Select *u* individuals using Algorithm 3 and re-evaluate them with the original functions and update *FE* = *FE* + u

9: Add individuals from step 8 to A_1 and A_2 and update $|A_1| = |A_1| + u$ and $|A_2| = |A_2| + u$

10: Remove $|A_1| - N_I$ individuals from A_1 using Algorithm 4, update w = 1 and $t_u = t_u + 1$ and go to step 3

end while

Empirically, the prefixed number of generations should be set in such a way that it allows the evolutionary algorithm to perform adequate search on the fitness landscape defined by the surrogate, while the search should be terminated if no further improvement in either convergence or diversity can be made. Ideally, the frequency of updating the surrogates can be made adaptive based on their performance, e.g., as done in [26]. However, a rigorous guideline for adapting the frequency still lacks. For simplicity, in this work we adopt a prefixed frequency based on a sensitivity analysis of the performance on the prefixed number of generations.

Once the function evaluations are done, simulated binary crossover and polynomial mutations are applied to generate offspring, similar to [15]. The parent and offspring populations are combined and then the selection criteria in RVEA detailed in Section II are used to select parents for the next generation.

C. Updating the surrogate

After an evolutionary search using the Kriging models for a fixed number of generations, the Kriging models will be updated. As previously mentioned, selection of individuals to be re-evaluated using the original functions, which will also be used for updating the surrogates, is essential for the performance of SAEAs. In this paper, we use information from the underlying evolutionary algorithm, RVEA, and uncertainty information from the Kriging models for selecting individuals to be re-evaluated and then for re-training the surrogate. As mentioned in Section I, selection of uncertain individuals not only helps in finding the unexplored regions but can also improve the quality of the surrogate. Therefore, we select individuals with the maximum uncertainty whenever diversity is needed. If a satisfactory degree of diversity is already achieved, we select individuals with the minimum angle penalized distance, which is one of the selection criteria in RVEA that contributes to convergence. Full details of the strategy for selecting individuals for re-evaluation is given in the next subsection, also summarized in Algorithm 3. The selected individuals are then re-evaluated with the original functions and these data samples are added to both archives A_1 and A_2. To keep a fixed number of individuals in the archive A_1, we will eliminate extra individuals from from A_1 using a strategy to be introduced in the following.

A maximum number of function evaluations is used as the termination criterion of the evolutionary optimization process. After the evolutionary search is complete, nondominated individuals in A_2 are taken as the final solutions.

1) Strategy for selecting individuals for re-evaluation: In RVEA, the reference vectors are adaptive. In this work, we introduce an additional set of fixed reference vectors that are evenly distributed in the objective space. The number of fixed reference vectors is the same as the number of adaptive vectors. For convenience, we denote the fixed reference vectors by V_f and the adaptive ones by V_a.

The fixed reference set is mainly for determining whether diversity or convergence should be prioritized in selecting individuals for re-evaluation. This is done by comparing the number of empty vectors in the fixed reference set during the previous surrogate update, denoted by $|V_f^{ia}|_{t_u-1}$ and that in the current update denoted by $|V_f^{ia}|_{t_u}$. A vector is called empty or non-active if no individual is assigned to this vector. In case $|V_f^{ia}|_{t_u}$ - $|V_f^{ia}|_{t_u-1} > \delta$, where $\delta > 0$ is a small integer, which means the increase in the number of empty fixed reference vectors has exceeded a given threshold, diversity should be prioritized. In this case, individuals for re-evaluation should be selected based on the uncertainty information offered by the Kriging models. By contrast, if $|V_f^{ia}|_{t_u}$ - $|V_f^{ia}|_{t_u-1} < \delta$, which means that the diversity of the population is not the major concern, priority will be given to the convergence criterion. In other words, individuals should be selected using the APD according to the adaptive reference vectors.

The next step is to determine which individuals should be selected according to either the amount of uncertainty or the value of APD. To this end, we divide the active adaptive reference vectors into a given number of clusters (Step 1 in Algorithm 3), from each of which one individual that has either the maximum amount of uncertainty (in case diversity is prioritized) or the minimum APD (in case convergence is to be taken care of) in the corresponding cluster. Thus, the number of clusters is always equal to the number of individuals,

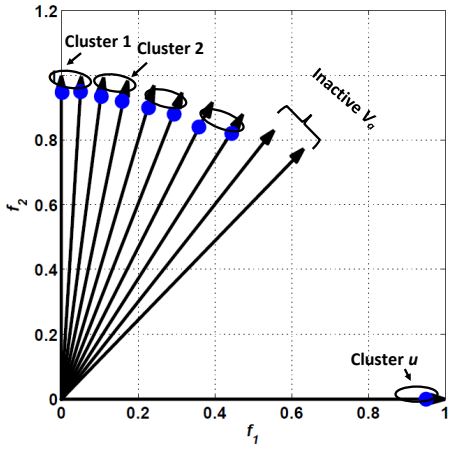

Figure 3. Clustering of active adaptive reference vectors V_a into a predefined number of clusters u.

denoted by u, to be selected for re-evaluation and for updating the surrogate. The process for selecting solutions to be re-evaluation described above is summarized in Algorithm 3.

Algorithm 3 Selection of individuals for updating the surrogate

Input: Sets of adaptive V_a and fixed V_f reference vectors, I = individuals obtained from the latest generation, $|V_f^{ia}|_{t_u-1}$ = number of inactive fixed reference vectors from the previous update, δ = parameter to decide whether to use APD or uncertainty from Kriging models for updating Kriging models, u = number of individuals to update Kriging models
Output: u = Individuals for updating Kriging models
1: Cluster active adaptive reference vectors into $min\{u, |V_a^a|\}$ clusters
2: Identify individuals closest to active adaptive reference vectors within each cluster
3: Assign I to fixed reference vectors and identify the number of inactive fixed reference vectors, i.e. $|V_f^{ia}|$
4: Calculate the change in the number of inactive fixed reference vectors from the previous update i.e. $\Delta V_f = |V_f^{ia}|_{t_u} - |V_f^{ia}|_{t_u-1}$
5: **If** $\Delta V_f \leq \delta$
6: Select one individual from each cluster with minimum APD
7: **else**
8: Select one individual from each cluster with maximum uncertainty
9: **end if**

To elaborate the above selection process, let us consider a few different situations shown in Figure 4 for a biobjective optimization problem, where the fixed reference vectors as well as the individuals (denoted by dots) associated to each vector are shown in updating Kriging models. Figure 4(a) illustrate the fixed reference vectors and the individuals assigned to them during the previous update, i.e., at the counter for updating the surrogate $t_u - 1$. Two different cases at the current update, i.e., at t_u, are shown in Figure 4(b) and Figure 4(c), respectively. In the situation shown in 4(b), the number of inactive fixed reference vectors, denoted by $|V_f^{ia}|_{t_u}$, has decreased, which indicates that diversity is not a concern and convergence should be emphasized. By contrast, Figure 4(c) shows a situation in which the number of inactive fixed reference vectors has increased, which indicates that diversity needs to be prioritized in updating the Kriging models. In the former case shown in Figure 4(b), individuals for re-evaluation are selected based on APD calculated using the reference vector set, while in the latter case, the amount o uncertainty, where uncertainty is calculated using the average of the standard deviations obtained from Kriging models. Selected individuals are re-evaluated with the original functions and the obtained data added to both archives A_1 and A_2. Note that if the number of active adaptive reference vectors is smaller than u, i.e., $|V_a^a| < u$, we group them into $|V_a^a|$ clusters. In the first update of the Kriging models, APD is always used for selecting individuals for re-evaluation.

In the following, we describe the strategy for managing the training data in A_1, when the number of available training data is large than the predefined maximum number defined by the size of A_1.

2) Managing the training data: In order to limit the computation time for re-training the Kriging models, the number of training data in archive A_1 is fixed. To this end, some data need to be discarded if the number of available training data is larger than the archive size. Which data samples should be kept in A_1 becomes critical for the quality of the Kriging model and thus the overall performance of K-RVEA. In this work, the maximum size of the training data, i.e., the size of archive A_1, is set to N_I. The main steps for managing training data archive A_1 are presented in Algorithm 4.

Algorithm 4 Managing individuals in the archive

Input: Archive A_1, adaptive reference vectors V_a, u= individuals selected to update Kriging models, N_I=maximum number of individuals in the archive A_1
Output: Updated individuals in A_1
1: Remove duplicate individuals from the training archive A_1 and update $|A_1|$
2: **If** $|A_1| > N_I$
3: Assign u individuals to V_a and identify inactive adaptive reference vectors, denote these reference vectors by V_a^{ia}
4: Assign $A_1 \setminus u$ individuals i.e. individuals other than recently evaluated ones in the archive to V_a^{ia}
5: Identify active reference vectors from inactive adaptive reference vectors V_a^{ia}
6: Cluster active set of inactive adaptive reference vectors V_a^{ia} into $N_I - u$ clusters
7: Select one individual from each cluster randomly and remove other individuals
11: **end if**

We first add recently evaluated individuals (u) obtained with Algorithm 3 to archive A_1 and remove duplicate data points from it (step 1 in Algorithm 4). If the number of training data is still larger than the archive size, we eliminate some training samples other than the recently evaluated one

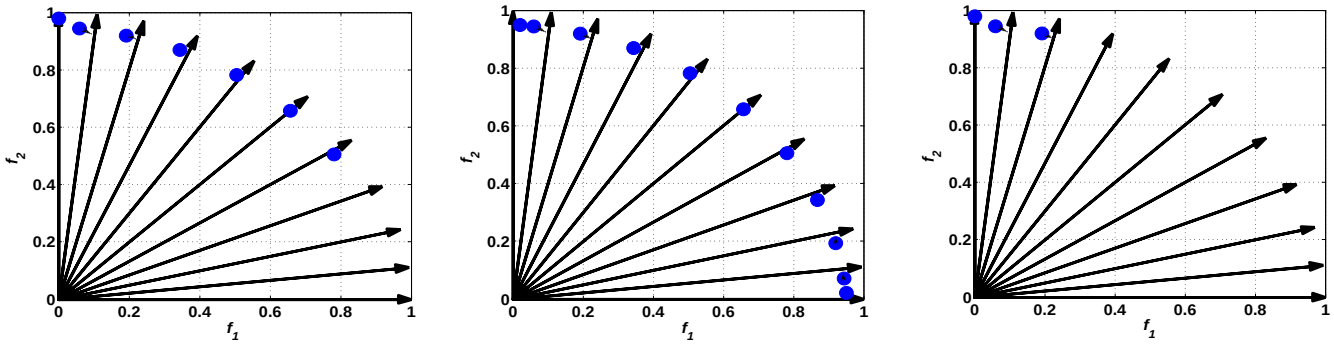

Figure 4. Different cases for fixed reference vectors while updating Kriging models. (a): An example of the fixed reference vectors, (b): An example showing the fixed reference vectors at the current update, where the number of inactive reference vectors has decreased compared to the previous update in (a), (c): An example showing the fixed reference vectors at the current update, where the number of inactive reference vectors has increased compared to the previous update in (a).

(step 2 in Algorithm 4). For this purpose, data points other than the recently evaluated individuals are assigned to the adaptive reference vectors. For instance, consider the situation in Figure 5 for a biobjective optimization problem. In Figure 5(a), individuals (training data) obtained with Algorithm 3 are assigned to the adaptive reference vectors V_a and inactive reference vectors are identified, denoted by V_a^{ia}. Then, we assign $A_1 \setminus u$ data points to these vectors V_a^{ia} and identify active reference vectors from them, as shown in Figure 5(b) (steps 4 and 5 in Algorithms 4). The active adaptive reference vectors are then grouped into $N_I - u$ clusters and data points assigned to these reference vectors in each cluster are identified (step 6 Algorithm 4). We randomly select one data point from each cluster and eliminate the rest of the data. In this way, a fixed number of diverse set of training data is maintained in A_1, thereby to improve the quality of Kriging as much as possible while keeping computation time limited.

IV. Numerical experiments

In this section, numerical experiments are conducted to examine the performance of K-RVEA on the DTLZ benchmark problems [16] for 3, 4, 6, 8 and 10 objectives are presented. The number of decision variables was set to 10. We also compared K-RVEA with representative Kriging based SAEAs, e.g. MOEA/D-EGO [51], SMS-EGO [41], [46], ParEGO [28], and with the original RVEA without using the surrogates. We also included RVEA for comparison to give the reader a sense of how differently an algorithm with and without surrogates performs on computationally expensive problems.

A. Parameter settings

1) Number of individuals to be evaluated using the original objective functions in the initialization phase (from the literature [28], [51]) = maximum number of individuals in the training archive, $N_I = 11n - 1$
2) Number of independent runs = 10
3) Maximum number of function evaluations, FE_{max} = 300
4) Number of individuals to update Kriging models, $|u|$ = 5
5) Number of reference vectors (N): number of reference vectors is determined by the design factors for simplex-lattice design [11] and the number of objectives and listed in the supplementary material
6) Parameter while updating the Kriging models $\delta = 0.05N$
7) Number of generations before updating the Kriging models w_{max} = 20.

For RVEA, a population size of 50 was used and for other algorithms, the same parameters were used as recommended by the authors in the respective articles. Inverted generational distance (IGD) [3] used as the performance measure to compare different algorithms. A Wilcoxon rank sum test was used to compare the results obtained by K-RVEA and the other four algorithms at a significance level of 0.05. Results are collected in Tables I, II and III, where symbol $\uparrow$ indicates that K-RVEA performed statistically better than a compared algorithm, and $\downarrow$ means that other algorithms performed better than K-RVEA, while $\approx$ means that there is no significant difference between the results obtained by K-RVEA and the other algorithm. One should note that for a given number of decision variables, the landscape of the problems is getting easier as the number of objectives increases. This is due to the fact that the number of decision variables for driving convergence decreases with the increase in the number of objectives [22]. Therefore, we have added the WFG suite [22] in our experiments and present the results in the supplementary material.

B. Performance on DTLZ problems

The results obtained with the four compared algorithms over 25 independent runs are collected in Table I. No results are given for ParEGO for more than four objectives as the current implementation given by the authors of ParEGO was limited to four objectives, which is denoted by 'NA' in the tables. The presented results include the minimum, mean and maximum values of IGD. The best values are highlighted.

Before we discuss the results, we want to mention an important issue in measuring the performance of many-objective evolutionary algorithms. IGD and hypervolume are two widely used performance indicators. However, the evaluation result may heavily depends on their parameters, particularly when the number of objectives is large. For instance, the IGD value is very sensitive to the size of the reference set, which has not been explicitly discussed. In [4], [48], 100000 reference points were used irrespective of the number of objectives, while in [52], only 500 reference points were used. In [5],

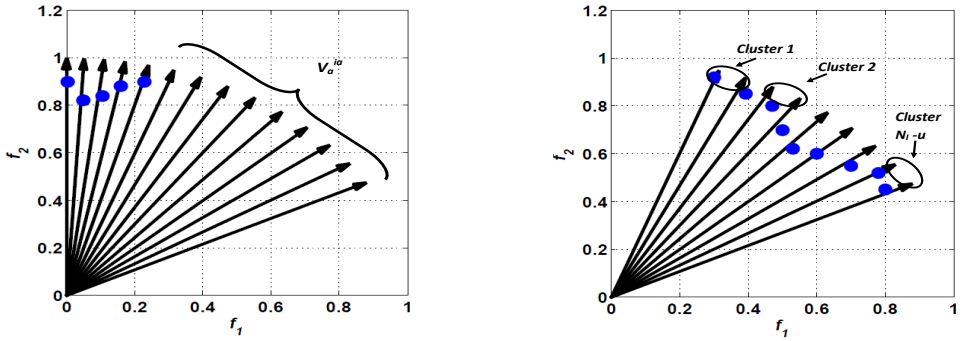

Figure 5. Managing training data in the archive A_1, (a):Assignment of the recently evaluated individuals u to the active adaptive reference vectors V_a, (b):Assignment of $A_1 \setminus u$ individuals to inactive adaptive reference vectors V_a^{ia}.

Table I. Statistical results for IGD values obtained by K-RVEA, RVEA and ParEGO. The best results are highlighted

Problem	k	K-RVEA				RVEA				ParEGO		
		min	mean	max		min	mean	max		min	mean	max
DTLZ1	3	82.03	106.9	125.2	≈	42.65	82.87	115.1	↓	**13.42**	**52.47**	**112.7**
	4	48.23	73.21	101.4	≈	39.65	59.18	97.71	↓	**18.63**	**45.45**	**87.76**
	6	**8.031**	28.83	**35.22**	≈	12.24	**22.94**	36.85		NA	NA	NA
	8	**0.699**	**6.991**	**13.29**	≈	1.250	7.406	15.66		NA	NA	NA
	10	0.198	0.347	**0.655**	≈	**0.193**	**0.339**	1.105		NA	NA	NA
DTLZ2	3	**0.092**	**0.155**	0.262	↑	0.227	0.288	0.335	↑	0.151	0.191	**0.243**
	4	**0.191**	**0.276**	**0.376**	↑	0.280	0.332	0.383	↑	0.289	0.337	0.408
	6	**0.316**	**0.342**	**0.362**	↑	0.375	0.404	0.440		NA	NA	NA
	8	**0.360**	**0.395**	**0.522**	↑	0.466	0.541	0.704		NA	NA	NA
	10	**0.419**	**0.446**	**0.470**	↑	0.539	0.608	0.733		NA	NA	NA
DTLZ3	3	181.5	280.1	353.1	≈	133.7	256.1	347.9	↓	**81.15**	**145.5**	**261.6**
	4	85.56	210.9	314.5	≈	89.95	198.6	306.3	↓	**66.93**	**138.1**	**209.4**
	6	61.61	105.0	**156.4**	≈	**43.54**	**95.97**	157.7		NA	NA	NA
	8	12.36	26.49	43.51	≈	**8.569**	**25.27**	**42.17**		NA	NA	NA
	10	0.781	1.299	2.303	≈	**0.761**	**1.228**	**1.836**		NA	NA	NA
DTLZ4	3	**0.190**	0.448	**0.737**	≈	0.205	**0.399**	0.959	↑	0.387	0.646	0.947
	4	**0.268**	**0.458**	**0.648**	≈	0.320	0.514	0.737	↑	0.505	0.725	0.960
	6	**0.422**	**0.585**	**0.754**	≈	0.503	0.615	0.800		NA	NA	NA
	8	**0.547**	0.635	**0.728**	≈	0.554	**0.628**	0.731		NA	NA	NA
	10	**0.553**	**0.608**	**0.672**	↑	0.599	0.667	0.761		NA	NA	NA
DTLZ5	3	0.050	0.112	0.211	↑	0.201	0.247	0.316	↓	**0.039**	**0.055**	**0.072**
	4	**0.046**	**0.123**	**0.242**	↑	0.149	0.294	0.393	↑	0.090	0.288	0.428
	6	**0.032**	**0.102**	**0.153**	↑	0.159	0.280	0.431		NA	NA	NA
	8	**0.023**	**0.048**	**0.107**	↑	0.104	0.260	0.748		NA	NA	NA
	10	**0.009**	**0.017**	**0.022**	↑	0.224	0.488	0.746		NA	NA	NA
DTLZ6	3	**2.121**	**2.727**	**3.343**	↑	3.651	4.960	5.613	↑	5.030	6.378	6.867
	4	**1.306**	**2.446**	**3.060**	↑	3.027	4.044	5.208	↑	5.652	5.916	6.034
	6	1.133	**1.597**	**2.174**	↑	**1.025**	2.524	3.600		NA	NA	NA
	8	**0.377**	**0.660**	**1.049**	↑	0.247	1.004	1.870		NA	NA	NA
	10	**0.054**	**0.153**	**0.373**	↑	0.140	0.297	0.751		NA	NA	NA
DTLZ7	3	**0.088**	**0.111**	**0.150**	↑	0.400	0.515	0.637	↑	0.621	0.829	1.201
	4	**0.188**	**0.243**	**0.298**	↑	0.532	0.691	0.926	↑	0.719	0.892	1.149
	6	**0.391**	**0.500**	**0.627**	↑	0.889	1.088	1.808		NA	NA	NA
	8	**0.745**	**0.886**	**1.030**	↑	1.162	1.359	1.634		NA	NA	NA
	10	**0.917**	**1.030**	**1.134**	↑	1.343	1.900	3.327		NA	NA	NA

different sizes for the reference set were used for different numbers of objectives. Here, we also use different sizes of the reference set for different numbers of objectives, as we believe more reference points are needed as the number of objectives increases, referring to the Supplementary material for details. Similarly, different criteria for setting the reference point in calculating the hypervolume have been adopted in different papers. It has been found in [49] that choosing a reference point slightly better than the nadir point is able to strike a balance between convergence and diversity of the solution set. Therefore, in this work, we use the worst objective function values of the non-dominated solutions found by all compared algorithms plus a small threshold. The detailed comparative results in terms of hypervolume are provided in the Supplementary material due to the page limit. We must emphasize that how to fairly compare the performance of EAs for many-objective optimization has received little attention in the literature and deserves more research.

As can be seen in Table I, overall, K-RVEA performed the best among all algorithms compared in this study, except on DTLZ1 and DTLZ3. We surmise that DTLZ1 and DTLZ3 have many local Pareto optimal solutions. Both RVEA and K-RVEA try to keep a well-distributed set of solutions because of the distribution in the reference vectors and, therefore, the convergence rate is relatively slow on these problems. Nondominated solutions of the run producing the best IGD values for K-RVEA, RVEA and ParEGO for DTLZ1 are shown in Figure 6. As can be seen, solutions of K-RVEA and RVEA are better distributed than those of ParEGO. However, the IGD values of all three algorithms are high, in other words, the

solutions obtained by these algorithms are all far from the true Pareto front. These results indicate that solving such problems requires more function evaluations to reach the Pareto front.

To compare the results with SMS-EGO, we selected a different stopping criterion. As SMS-EGO tries to maximize the expected hypervolume improvement and was implemented in MATLAB, it took about seven hours to complete one run on a computer with the i5 processor and 4GB RAM. The needed large amount of computation time of SMS-EGO was also mentioned in [51], for which reason the authors compared their algorithm with SMS-EGO only on two test problems. In this paper, we allowed SMS-EGO to run for 24 hours with 10 parallel runs and stored all the solutions. The number of function evaluations reached during this time was used as the stopping criterion for comparison with the other algorithms. The results for three and four objectives are obtained with 120 and 115 function evaluations, respectively, which are presented in Table II. These results obtained with a small number of function evaluations show that K-RVEA performed better than the compared algorithms. We did not compare K-RVEA with SMS-EGO for problems within more than four objectives, as SMS-EGO requires even more time for such problems.

The comparison of K-RVEA and MOEA/D-EGO for three objective functions after 300 original function evaluations is shown in Table III. An implementation of MOEA/D-EGO from the authors was available for only two and three objectives. As can be seen in Table III, K-RVEA performed either better or comparably to MOEA/D-EGO for all problems except on DTLZ5. As the Pareto front of DTLZ5 is a curve that covers a small subspace in the objective space, most of the reference vectors in K-RVEA and RVEA are empty, i.e., no solutions are assigned to them. We observed that for both K-RVEA and RVEA, almost 70% of the reference vectors are empty, which makes the solution process to converge slowly to the Pareto front. For such problems, a large number of reference vectors could be helpful while using K-RVEA.

Nondominated solutions from the run with the best IGD values obtained by the compared algorithms on the three-objective DTLZ7 are shown in Figure 7. As can be seen from the figure, nondominated solutions obtained of K-RVEA and RVEA are much closer to the Pareto front than those of ParEGO and MOEA/D-EGO. For DTLZ7 which has a disconnected Pareto front, RVEA and K-RVEA have a good potential to get solutions close to the Pareto front because of the adaptation in the reference vectors. Remind that we did not run SMS-EGO for optimization problems with more than four objectives as the runtime is prohibitive. In addition, a parallel coordinates plot for DTLZ2 with 10 objectives is presented in Figure 8, which we can see that the ranges of solutions obtained by K-RVEA are bigger than those obtained by RVEA. In other words, the solutions obtained by K-RVEA have a better distribution. The reason for a lower density of the solutions obtained by RVEA is due to the small population size. As the maximum number of function evaluations for termination is quite low and the performance of RVEA depends on the maximum number of generations, we reduced the population of RVEA to 50. To more convincingly demonstrate the performance of K-RVEA, we tested and compared the algorithm on the WFG problems [22]. Results in terms of both IGD and hypervolume are provided in the Supplementary material, which show that K-RVEA was able to perform better than the compared algorithms.

As mentioned in Section II.B, the computation time for training Kriging models varies a lot depending on the specific implementation and the number of training, which may become prohibitive large. One contribution of K-RVEA is the development of a strategy to select training samples reference vectors, the number of samples for training Kriging models is kept constant, the computation time for training K-RVEA is remains constant as the number of expensive fitness evaluations increases. To empirically verify this, the run time of the different implementations in the compared SAEAs for training the Kriging model has been investigated. The results over the number of training samples obtained on the 3-objective DTLZ2 are shown in Figure 9. We can observe that the training time of K-RVEA remains constant, as the maximum number of training samples is kept constant, the training time for ParEGO increases slightly. In contrast, the training time of MOEA/D-EGO increases quickly over the number of training samples as a piecewise continues function. As already mentioned, the computation time of SMS-EGO increases dramatically over the number of training samples. Note however that SMS-EGO and K-RVEA are implemented in MATLAB, ParEGO is implemented in C and MOEA/D in Java. Therefore, the absolute times used by the different algorithms may not be directly comparable, although the different behaviours of the change in training time over the number allowed true function evaluations are of more interest.

In what follows, we consider the effect of different parameters in K-RVEA. The parameter δ in Algorithm 2 is used to select individuals for updating surrogates to balance between convergence and diversity. We did a sensitivity analysis to see the effect of δ on the diversity and the performance of the algorithm. As diversity in both RVEA and K-RVEA can easily be measured using reference vectors, we studied the effect of δ by measuring the change in the number of empty reference vectors. We also measured the hypervolume to see the effect on the performance of the algorithm and provide the results in the supplementary material. As expected, increase in the value of δ deteriorates the diversity and thus the hypervolume. This is due to fact that frequency of using uncertainty information from Kriging models decreases with the increase in value of δ. In other words, if the change in the number of the empty reference vectors is smaller than δ, individuals are selected based on convergence, otherwise based on uncertainty of Kriging models. Increase in the value of δ will force the algorithm to select the individuals based on convergence and thus deteriorates the diversity. In this article we fixed the value of δ to be $0.05 \times N$, where N is the number of reference vectors used and adaptation of δ will be our future work.

Apart from δ, two other parameters can also influence the performance of K-RVEA, which are the number of individuals to be selected to update the surrogates and the number of generations (w_{max}) used before updating the surrogates. The first parameter depends on the characteristics of the problem, e.g., multi-modality, and the evolutionary algorithm used. We study

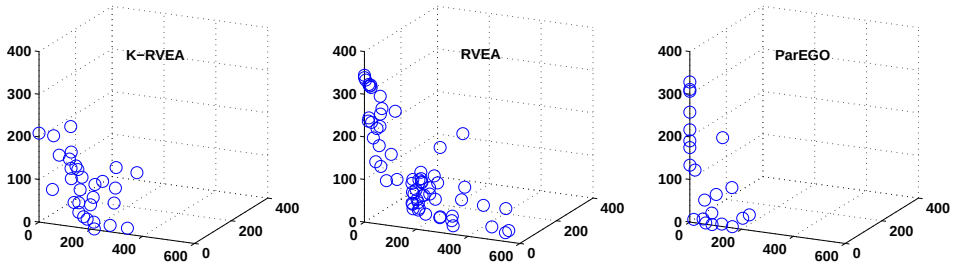

Figure 6. Nondominated solutions obtained with K-RVEA, RVEA and ParEGO of the run with the best IGD value for three-objective DTLZ1, which are all very far away from the true Pareto front.

Table II. Statistical results for IGD values obtained by K-RVEA, RVEA, ParEGO, SMS-EGO and MOEA/D-EGO for three and four objectives with 120 and 115 function evaluations, respectively. The best results are highlighted

Problem	k	K-RVEA				RVEA				ParEGO				SMS-EGO				MOEA/D-EGO		
		min	mean	max		min	mean	max		min	mean	max		min	mean	max		min	mean	max
DTLZ1	3	82.03	130.2	170.8	≈	**66.19**	**129.6**	171.4	≈	100.7	124.2	148.8	↑	85.47	114.2	**148.8**	↑	173.1	267.5	388.2
	4	61.92	90.29	115.7	≈	81.52	102.2	133.3	≈	75.23	99.81	128.5	↑	93.33	130.2	173.7		NA	NA	NA
DTLZ2	3	0.305	0.360	0.408	↑	0.401	0.436	0.494	≈	**0.271**	**0.356**	**0.396**	↑	0.365	0.444	0.522	↑	0.325	0.384	0.456
	4	**0.376**	0.427	**0.464**	↑	0.421	0.463	0.499	≈	0.384	**0.422**	0.474	↑	0.447	0.487	0.532		NA	NA	NA
DTLZ3	3	217.3	**324.3**	**383.7**	≈	236.3	366.6	495.6	≈	232.4	368.3	460.7	≈	220.8	325.3	459.2	≈	**189.7**	339.9	523.41
	4	173.8	302.1	370.5	≈	177.6	**283.1**	359.1	≈	**172.5**	291.9	**357.2**	≈	209.1	314.1	403.8		NA	NA	NA
DTLZ4	3	0.452	0.711	0.979	≈	0.537	0.678	0.967	≈	0.586	0.769	0.911	≈	0.649	0.690	0.722	↓	**0.393**	**0.434**	**0.479**
	4	**0.587**	**0.750**	0.914	≈	0.669	0.803	0.928	≈	0.716	0.819	0.993	≈	0.680	0.746	**0.815**		NA	NA	NA
DTLZ5	3	0.173	0.282	0.360	≈	0.272	0.326	0.397	↑	0.597	0.615	0.638	↑	0.447	0.498	0.546	↓	**0.170**	**0.215**	**0.308**
	4	**0.226**	**0.254**	**0.301**	↑	0.248	0.283	0.316	↑	0.432	0.454	0.484	↑	0.360	0.413	0.465		NA	NA	NA
DTLZ6	3	3.104	3.974	4.629	↑	5.737	6.110	6.462	↑	6.504	6.707	6.825	↓	**1.603**	**1.756**	**2.007**	↑	4.072	5.078	6.624
	4	**2.962**	**3.910**	**4.585**	↑	4.406	5.036	5.629	↑	5.720	5.868	6.024	↑	5.842	5.883	6.010		NA	NA	NA
DTLZ7	3	**0.119**	**0.167**	**0.216**	↑	0.632	0.760	1.264	↑	3.138	5.573	7.417	↓	0.241	0.260	0.277	↑	0.618	1.743	3.830
	4	**0.297**	**0.401**	**0.647**	↑	0.778	1.018	1.219	↑	5.304	7.376	8.921	↓	0.578	0.644	0.706	↑	NA	NA	NA

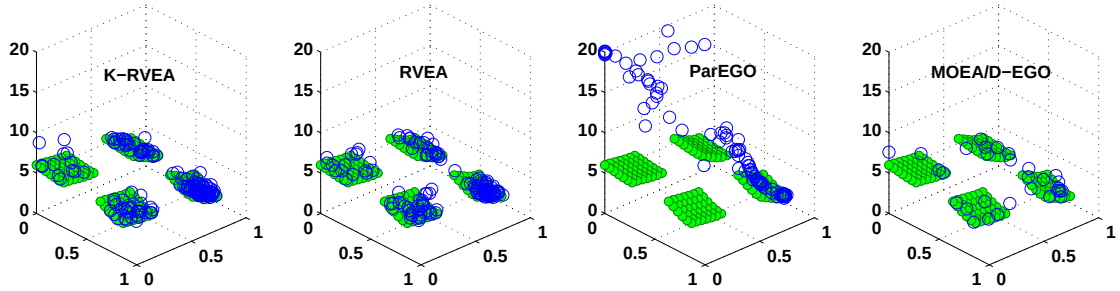

Figure 7. Nondominated solutions obtained by K-RVEA, RVEA, ParEGO and MOEA/D-EGO, denoted by circles, in the run with the best IGD value for three-objective DTLZ7, where the dots represent the Pareto front.

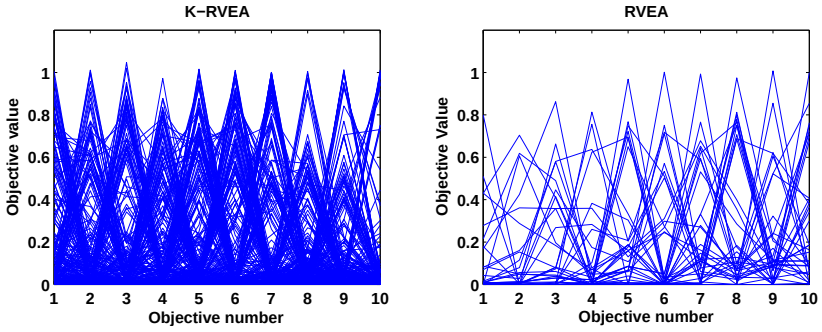

Figure 8. Parallel coordinate plot of the nondominated solutions obtained by K-RVEA and RVEA in the run with the best IGD value on the 10-objective DTLZ2.

Table III. Statistical results for IGD values obtained by K-RVEA and MOEA/D-EGO for three objectives after 300 function evaluations. The best results are highlighted

Problem	K-RVEA				MOEA/D-EGO		
	min	mean	max		min	mean	max
DTLZ1	**82.03**	**106.9**	**125.2**	↑	145.9	177.9	224.5
DTLZ2	0.092	0.155	0.262	≈	**0.081**	**0.103**	**0.212**
DTLZ3	181.5	280.1	353.1	≈	**161.5**	**205.9**	**281.8**
DTLZ4	**0.190**	0.448	0.737	≈	0.357	**0.436**	0.574
DTLZ5	0.050	0.112	0.211	↓	**0.035**	**0.046**	**0.071**
DTLZ6	2.121	2.727	**3.343**	≈	**0.491**	**2.551**	4.126
DTLZ7	**0.088**	**0.111**	**0.150**	↑	0.154	0.646	1.254

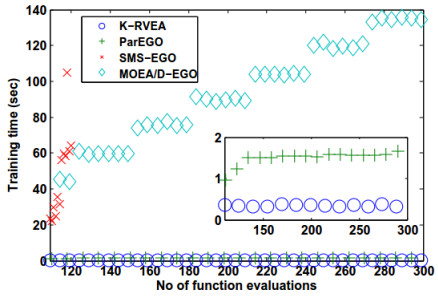

Figure 9. Training time over the number of function evaluations in K-RVEA, ParEGO, SMS-EGO and MOEA/D-EGO on the 3-objective DTLZ2.

the effect of the parameter in the Supplementary material. Our results show that the value of the parameter is problem-specific and an adaptive way of using the parameter is needed. For the second parameter i.e. the frequency of updating the surrogates or when to update the surrogate is very important in surrogate management, although, unfortunately, there is no solid theory for guiding when to update the surrogates. We have performed a sensitivity analysis on the performance of K-RVEA given different prefixed numbers of generations before the Kriging models are re-trained. The results are also included in the Supplementary material.

We also tested K-RVEA, RVEA, ParEGO and MOEA/D-EGO on a three objective real-world polymerization problem [34]. Even though K-RVEA and RVEA have been proposed for more than three objectives, they still performed better in solution quality and computation time than the compared algorithms. Details and results for this problem are given in the supplementary material.

V. Conclusions and Future Research

In this paper, a Kriging-assisted reference vector guided evolutionary algorithm, called K-RVEA, has been proposed for solving computationally expensive optimization problems with more than three objectives, where a Kriging model is used to approximate each objective function. We take care of both convergence and diversity in choosing individuals for re-evaluation with the original expensive objective functions. For this purpose, we introduced a set of fixed, uniformly distributed reference vectors in addition to the adaptive reference vectors in RVEA. In updating the Kriging models, attention is paid to limiting the computation time for training the surrogate by means of selecting training samples according to their relationships to the reference vectors, thereby limiting the number of training data.

We have examined the performance of K-RVEA on benchmark problems with 3, 4, 6, 8 and 10 objectives. We also compared K-RVEA with three state-of-the art SAEAs. Empirical results show that overall the K-RVEA obtained much better performance than the compared algorithms given the same number of function evaluations using the original expensive objective function.

In this paper, the number of decision variables was set to 10, as done in other papers in the literature that use Kriging as the surrogate model. This can be attributed to the factor that for solving optimization problems with a higher number of decision variables, much more training data will be needed, which will not only require more computational resource but also poses more serious challenges to Kriging based surrogate techniques. Nevertheless, it is highly desired that SAEAs can be applicable to optimization problems having a larger number of decision variables, which will be our future work. Another topic for future study is to investigate the performance of the proposed K-RVEA for constrained computationally expensive optimization problems. Finally, in the present work, we fixed the number of generations for updating the surrogates. Although our empirical results indicate that the performance of K-RVEA is relatively insensitive to the frequency of updating the surrogates for the benchmark problems studied in this work, our previous work [26] indicated that it is likely to adapt the frequency of updating the surrogates to further enhance the performance of SAEAs. Consequently, developing an adaptive method for updating the surrogates will also be our future research work.

VI. Acknowledgment

The authors would like to thank R. Cheng, J. Knowles, T. Wagner, Q. Zhang and A. Zhou for providing the codes of the algorithms compared in this work.

References

[1] T. Ando. Majorization relations for hadamard products. *Linear Algebra and its Applications*, 223-224:57–64, 1995.

[2] N. Azzouz, S. Bechikh, and L. Said. Steady state IBEA assisted by MLP neural networks for expensive multi-objective optimization problems. In C. Igel, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 581–588. ACM, 2014.

[3] P. Bosman and D. Thierens. The balance between proximity and diversity in multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 7:174–188, 2003.

[4] P. Chandan, M. Islam, K. Murase, and X. Yao. Evolutionary path control strategy for solving many-objective optimization problems. *IEEE Transactions on Cybernetics*, 45:702–715, 2015.

[5] J. Cheng, G. Yen, and G. Zhang. A many-objective evolutionary algorithm with enhanced mating and environmental selections. *IEEE Transactions on Evolutionary Computation*, 19:592–605, 2015.

[6] L. Cheng, K. Shimoyama, and S. Obayashi. Kriging model based many-objective optimization with efficient calculation of hypervolume improvement. In *In proceedings of IEEE Congress on Evolutionary Computation*, pages 1187–1194. IEEE, 2014.

[7] R. Cheng, Y. Jin, K. Narukawa, and B. Sendhoff. A multiobjective evolutionary algorithm using gaussian process-based inverse modeling. *IEEE Transactions on Evolutionary Computation*, 19:838–856, 2015.

[8] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff. A reference vector guided evolutionary algorithm for many objective optimization. *IEEE Transactions on Evolutionary Computation*, 2016 (accepted).

[9] T. Chugh, K. Sindhya, J. Hakanen, and K. Miettinen. Handling computationally expensive multiobjective optimization problems with evolutionary algorithms-A survey. Technical Report Series B, Scientific Computing No. B 4/2015, Department of Mathematical Information Technology, University of Jyvaskyla, 2015.

[10] C. Coello, G. Lamont, and D. Veldhuizen. *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer, New York, 2nd edition, 2007.

[11] J. Cornell. *Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data*. John Wiley & Sons, 2011.

[12] I. Couckuyt, D. Deschrijver, and T. Dhaene. Fast calculation of multiobjective probability of improvement and expected improvement criteria for Pareto optimization. *Journal of Global Optimization*, 60:575–594, 2014.

[13] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, Chichester, 2001.

[14] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18:577–601, 2014.

[15] K. Deb, A. Prarap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6:182–197, 2002.

[16] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 825–830. IEEE, 2002.

[17] A. G. Di Nuovo, G. Ascia, and V. Catania. A study on evolutionary multi-objective optimization with fuzzy approximation for computational expensive problems. In *Parallel Problem Solving from Nature*, volume LNCS 7492, pages 102–111. Springer, 2012.

[18] M. Emmerich, K. Giannakoglou, and B. Naujoks. Single- and multi-objective evolutionary optimization assisted by Gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10:421–439, 2006.

[19] N. Hansen. The CMA evolution strategy: A comparing review , pp. 17691776. In J. Lozano, P. Larranaga, I. Inza, and E. Bengoetxea, editors, *Towards a New Evolutionary Computation*, pages 75–102. Springer Berlin Heidelberg, 2006.

[20] J. Hensman, N. Fusi, and N. Lawrence. Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intelligence*, 2013.

[21] R. Hooke and T. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the Association for Computing Machinery (ACM)*, 8:212–229, 1961.

[22] S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In C. Coello, A. H. Aguirre, and E. Zitzler, editors, *Evolutionary Multi-Criterion Optimization*, pages 280–295. Springer, 2005.

[23] H. Ishibuchi, N. Tsukamoto, and Y. Nojima. Evolutionary many-objective optimization: A short review. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 2419–2426. IEEE, 2008.

[24] Y. Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1(2):61–70, 2011.

[25] Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–793. Morgan Kaufmann, 2000.

[26] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6:481–494, 2002.

[27] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural network ensembles. In K. Deb, editor, *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 688–699. Springer, Berlin, Heidelberg, 2004.

[28] J. Knowles. ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Transactions on Evolutionary Computation*, 10:50–66, 2006.

[29] B. Li, J. Li, K. Tang, and X. Yao. Many-objective evolutionary algorithms: A survey. *ACM Computing Surveys*, 48:13–35, 2015.

[30] S. Lophaven, H. Nielsen, and J. Sondergaard. DACE: a MATLAB Kriging toolbox. Technical report, Technical University of Denmark, 2002.

[31] I. Loshchilov, M. Schoenauer, and M. Sebag. Dominance-based Pareto-surrogate for multi-objective optimization. In K. Deb, A. Bhattacharya, N. Chakroborty, S. Das, J. Dutta, S. Gupta, A. Jain, V. Aggarwal, J. Branke, S. Louis, and K. Tan, editors, *Proceedings of the Simulated Evolution and Learning*, pages 230–239. Springer, Berlin, Heidelberg, 2010.

[32] M. Mckay, R. Beckman, and W. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42:55–61, 2000.

[33] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer, Boston, 1999.

[34] A. Mogilicharla, T. Chugh, S. Majumder, and K. Mitra. Multi-objective optimization of bulk vinyl acetate polymerization with branching. *Materials and Manufacturing Processes*, 29:210–217, 2014.

[35] H. Nakayama, K. Inoue, and Y. Yoshimori. Approximate optimization using computational intelligence and its application to reinforcement of cable-stayed bridges. In *Proceedings of the Integrated Intelligent Systems for Engineering Design*, pages 289–304. IOS Press, 2006.

[36] K. Narukawa and T. Rodemann. Examining the performance of evolutionary many-objective optimization algorithms on a real-world application. In *Proceedings of Genetic and Evolutionary Computing*, pages 316–319. IEEE, 2012.

[37] J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[38] M. Pilát and R. Neruda. ASM-MOMA: Multiobjective memetic algorithm with aggregate surrogate model. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 1202–1208. IEEE, 2011.

[39] M. Pilát and R. Neruda. Improving many-objective optimizers with aggregate meta-models. In *Proceedings of the 11th International Conference on Hybrid Intelligent Systems*, pages 555–560. IEEE, 2011.

[40] M. Pilat and R. Neruda. Aggregate meta-models for evolutionary multi-objective and many-objective optimization. *Neurocomputing*, 116:392–402, 2013.

[41] W. Ponweiser, T. Wagner, D. Biermann, and M. Vincze. Multiobjective optimization on a limited budget of evaluations using model-assisted S-metric selection. In G. Rudolph, T. Jansen, S. Lucas, C. Poloni, and N. Beume, editors, *Proceedings of the Parallel Problem Solving from Nature-PPSN X*, pages 784–794. Springer, Berlin, Heidelberg, 2008.

[42] R. Purshouse, K. Deb, M. Mansor, S. Mostaghim, and R. Wang. A review of hybrid evolutionary multiple criteria decision making methods. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1147–1154. IEEE, 2014.

[43] L. Santana-Quintero, A. Montano, and C. Coello. A review of techniques for handling expensive functions in evolutionary multi-objective optimization. In Y. Tenne and C.-K. Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, pages 29–59. Springer, Berlin, Heidelberg, 2010.

[44] R. Storn and K. Price. Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[45] C. von Lücken, B. Barán, and C. Brizuela. A survey on multi-objective evolutionary algorithms for many-objective problems. *Computational Optimization and Applications*, 58:707–756, 2014.

[46] T. Wagner. *Planning and Multi-Objective Optimization of Manufacturing Processes by Means of Empirical Surrogate Models*. PhD thesis, Schriftenreihe des ISF, Vulkan Verlag, Essen, 2013.

[47] T. Wagner, M. Emmerich, A. Deutz, and W. Ponweiser. On expected-improvement criteria for model-based multi-objective optimization. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Proceedings of the Parallel Problem Solving from Nature-PPSN XI*, pages 718–727. Springer, Berlin, Heidelberg, 2010.

[48] Y. Yazawa, H. Aguirre, A. Oyama, and K. Tanaka. Evolutionary many-objective optimization optimization using ϵ-Hoods and chebyshev function. In *In the proceedings of IEEE Congress on Evolutionary Computation*, pages 1861–1868. IEEE, 2015.

[49] Y. Yuan, H. Xu, B. Wang, and B. Zhang. Balancing convergence and diversity in decomposition-based many-objective optimizers. *IEEE Transactions on Evolutionary Computation*, 2016 (accepted).

[50] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11:712–731, 2007.

[51] Q. Zhang, W. Liu, E. Tsang, and B. Virginas. Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Transactions on Evolutionary Computation*, 14:456–474, 2010.

[52] X. Zhang, Y. Tian, and Y. Jin. A knee point-driven evolutionary algorithm for many-objective optimization. *IEEE Transactions on Evolutionary Computation*, 19:761–776, 2015.

Tinkle Chugh is a PhD student in the Industrial Optimization Group within Faculty of Information Technology, University of Jyvaskyla, Finland. He received his Master of Technology degree in Chemical Engineering from Indian Institute of Technology Hyderabad, India, in 2012. His research interests include evolutionary computation, surrogate-assisted multi-/many- objective optimization, and machine learning.

Yaochu Jin (M'98-SM'02-F'16) received the B.Sc., M.Sc., and Ph.D. degrees from Zhejiang University, Hangzhou, China, in 1988, 1991, and 1996 respectively, and the Dr.-Ing. degree from Ruhr University Bochum, Germany, in 2001.

He is a Professor in Computational Intelligence, Department of Computer Science, University of Surrey, Guildford, U.K., where he heads the Nature Inspired Computing and Engineering Group. He is also a Finland Distinguished Professor, University of Jyvaskyla, Finland and a Changjiang Distinguished Professor, Northeastern University, China. His main research interests include evolutionary computation, machine learning, computational neuroscience, and evolutionary developmental systems, with their application to data-driven optimization and decision-making, self-organizing swarm robotic systems, and bioinformatics. He has (co)authored over 200 peer-reviewed journal and conference papers and has been granted eight patents on evolutionary optimization.

Dr Jin is the Editor-in-Chief of the IEEE TRANSACTIONS ON COGNITIVE AND DEVELOPMENTAL SYSTEMS and Complex & Intelligent Systems. He was an IEEE Distinguished Lecturer (2013-2015) and Vice President for Technical Activities of the IEEE Computational Intelligence Society (2014-2015). He was the recipient of the Best Paper Award of the 2010 IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology and the 2014 IEEE Computational Intelligence Magazine Outstanding Paper Award. He is a Fellow of IEEE.

Kaisa Miettinen is Professor of Industrial Optimization and vice-rector of the University of Jyvaskyla. She received her Ph.D. and M.Sc. degrees at the Department of Mathematics of the University of Jyvaskyla in 1994 and 1988, respectively. Her research interests include theory, methods, applications and software of nonlinear multiobjective optimization including interactive and evolutionary approaches and she heads the Research Group on Industrial Optimization. She has authored about 150 refereed journal, proceedings and collection papers, edited 13 proceedings, collections and special issues and written a monograph Nonlinear Multiobjective Optimization. She is a member of the Finnish Academy of Science and Letters, Section of Science and the Immediate-Past President of the International Society on Multiple Criteria Decision Making. She belongs to the editorial boards of six international journals, the Steering Committee of Evolutionary Multi-Criterion Optimization and is a member of the IEEE Computational Intelligence Society Task Force Group on Multiobjective Evolutionary Algorithms. She has worked at IIASA, International Institute for Applied Systems Analysis in Austria, KTH Royal Institute of Technology in Stockholm, Sweden and at Helsinki School of Economics in Finland.

Jussi Hakanen is a senior researcher with the Faculty of Information Technology at the University of Jyvaskyla, Finland. He received MSc degree in mathematics and PhD degree in mathematical information technology, both from the University of Jyvaskyla. His research is focused on multiobjective optimization with an emphasis on interactive multiobjective optimization methods and computationally expensive problems. He has participated in several industrial projects involving different applications of multiobjective optimization, e.g. in chemical engineering. He has been a visiting researcher in Cornell University, Carnegie Mellon, University of Surrey, University of Wuppertal, University of Malaga and the VTT Technical Research Center of Finland. He also has a title of docent in industrial optimization at the University of Jyvaskyla.

Karthik Sindhya is a post-doctoral researcher with the Faculty of Information Technology, University of Jyvaskyla, Finland. He has a bachelor's and master's degree in chemical engineering and a PhD in the area of multi-criteria optimisation. His PhD thesis on hybrid algorithms was nominated as one of the top three dissertations in multiple criteria decision making during the period of 2010-2013. He was also the recipient of the "Best algorithm award" at IEEE CEC 2007 and best poster award at MCDM 2011 conferences. His research interests include evolutionary algorithms especially evolutionary multi-objective optimisation (EMO) algorithms, handling computationally expensive objective functions that arise in industry and multiple criteria decision making approaches.