MOEA/D assisted by RBF Networks for Expensive Multi-Objective Optimization Problems

Saúl Zapotecas Martínez and Carlos A. Coello Coello

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
México D.F. 07300, MÉXICO
saul.zapotecas@gmail.com, ccoello@cs.cinvestav.mx

Abstract. The development of multi-objective evolutionary algorithms (MOEAs) assisted by surrogate models has increased in the last few years. However, in real-world applications, the high modality and dimensionality that functions have, often confuse such models. Therefore, if the Pareto optimal set of a MOP is located in a search space in which the surrogate model is not able to shape this region, the search could be misinformed and converge to wrong regions. Because of this, most of the researchers have tried to improve the prediction of the surrogate models by adding the new solutions to the training set and retraining the surrogate model. When the size of the training set increases, the training complexity can increase significantly. In this paper, we present a surrogate model which maintains the size of the training set, and the function prediction is improved by using a set RBF networks in a cooperative way. Preliminary results have shown that our proposed approach can produce good quality results when it is restricted to 200, 1,000 and 5,000 fitness function evaluations, which have been used for solving a set of standard test problems and an airfoil design problem.

Keywords: Multi-objective optimization, RBF neural networks, expensive optimization problems.

1 Introduction

Multi-Objective Evolutionary algorithms (MOEAs) have been successfully adopted to solve multi-objective optimization problems (MOPs) in a wide variety of engineering and scientific problems [1]. However, in real-world applications is common to find objective functions which are very expensive to evaluate (in terms of computational time). This has considerably limited the use of these evolutionary techniques to these types of problems. In recent years, several researchers have developed different strategies for reducing the computational time

(measured in terms of the number of fitness function evaluations) that a MOEA requires to solve a determined problem. From such strategies, the use of surrogate models has been one of the most common techniques adopted to solve complex problems. In the specialized literature, several authors have reported the use of surrogate models for dealing with MOPs—see for example [2–7] among others. However, the features of some problems, such as high modality and dimensionality, often present major obstacles to surrogate models. Therefore, if the Pareto optimal set of a MOP is located in a search space in which the surrogate model is not able to shape the corresponding region, the search could be misinformed and converge to wrong regions. Because of this, an important number of researchers have tried to improve the prediction of surrogate models by adding new solutions to the training set and then retraining the surrogate model at each iteration of the MOEA, see for example [7, 6]. However, as the training set gets larger, the complexity of the training also increases.

In this paper, we present an algorithm based on the well-known MOEA/D [8] which is assisted by radial basis function (RBF) networks. The proposed approach uses a static size for the training set and the function prediction of the surrogate model is improved by using a set of RBF networks in a cooperative way. With this, the computational complexity (measured in terms of computational time) is not dependent on the size of the training set, being limited only to the number of decision variables that the MOP has. The main goal of this paper is to contribute to the state-of-the-art regarding MOEAs assisted by surrogate models, which we believe is of interest in a wide variety of real-world problems (given the wide applicability of MOEAs [1]). We show in this paper that cooperative RBF networks can significantly reduce the number of fitness function evaluations that are required to produce reasonably good approximations of MOPs of different complexity.

The remainder of this paper is organized as follows. In Section 2, we present the basic concepts to understand the rest of the paper. In Section 3, we describe in detail our proposed approach. In Section 4, the test problems adopted to validate our approach are described. In Section 5, we show and discuss the results obtained by our proposed approach. Finally, in Section 6, we provide our conclusions and some possible paths for future research.

2 Basic Concepts

2.1 Multi-Objective Optimization

Without loss of generality we will assume only minimization problems. Thus, a nonlinear multi-objective optimization problem can be formulated as:

$$\min_{\boldsymbol{x}\in\Omega} \quad \{\boldsymbol{F}(\boldsymbol{x})\} \tag{1}$$

where Ω defines the decision variable space and $\boldsymbol{F} : \Omega \to \mathbb{R}^k$ defines the vector of objective functions $\boldsymbol{F}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))^T$, such that $f_i : \mathbb{R}^n \to \mathbb{R}$ is a

nonlinear function. In order to describe the concept of optimality in which we are interested, the following definitions are introduced [9]:

Definition 1. *Let $\boldsymbol{x}, \boldsymbol{y} \in \Omega$, we say that $\boldsymbol{x}$* dominates *$\boldsymbol{y}$ (denoted by $\boldsymbol{x} \prec \boldsymbol{y}$) if and only if, $f_i(\boldsymbol{x}) \leq f_i(\boldsymbol{y})$ and $\boldsymbol{F}(\boldsymbol{x}) \neq \boldsymbol{F}(\boldsymbol{y})$.*

Definition 2. *Let $\boldsymbol{x}^\star \in \Omega$, we say that $\boldsymbol{x}^\star$ is a* Pareto optimal *solution, if there is no other solution $\boldsymbol{y} \in \Omega$ such that $\boldsymbol{y} \prec \boldsymbol{x}^\star$.*

Definition 3. *The* Pareto optimal set *PS is defined by:*

$$PS = \{\boldsymbol{x} \in \Omega | \boldsymbol{x} \text{ is Pareto optimal solution}\}$$

Definition 4. *The* Pareto optimal front *PF is defined as:*

$$PF = \{\boldsymbol{F}(\boldsymbol{x}) | \boldsymbol{x} \in PS\}$$

We are interested in maximizing the number of elements of the Pareto optimal set and maintaining a well-distributed set of solutions along the Pareto optimal front.

2.2 Decomposing Multi-Objective Optimization Problems

It is well-known that a Pareto optimal solution to a MOP, under some assumptions, is an optimal solution of a scalar optimization problem in which, the objective is an aggregation of all the objective functions f_i's. Therefore, an approximation of the Pareto optimal front can be decomposed into a number of scalar objective optimization subproblems. In the specialized literature, there are several approaches for transforming a MOP into multiple single-objective optimization subproblems [10, 9]. In the following, we briefly describe an approach based on the normal boundary intersection (NBI) [11] method, which is referred to in this work.

Penalty Boundary Intersection Approach The Penalty Boundary Intersection (PBI) approach proposed by Zhang and Li [8], uses a weighting vector $\boldsymbol{w}$ and a penalty value θ for minimizing both the distance to the utopian vector d_1 and the direction error to the weighting vector d_2 from the solution $\boldsymbol{F}(\boldsymbol{x})$. Therefore, the optimization problem can be stated as:

$$\text{minimize:} \quad g(\boldsymbol{x}|\boldsymbol{w}, \boldsymbol{z}^\star) = d_1 + \theta d_2 \tag{2}$$

where

$$d_1 = \frac{||(\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{z}^\star)^T \boldsymbol{w}||}{||\boldsymbol{w}||}$$
$$\text{and} \quad d_2 = \left|\left|(\boldsymbol{F}(\boldsymbol{x}) - \boldsymbol{z}^\star) - d_1 \tfrac{\boldsymbol{w}}{||\boldsymbol{w}||}\right|\right|$$

such that $\boldsymbol{x} \in \Omega$ and $\boldsymbol{z}^\star = (z_1, \ldots, z_k)^T$, such that: $z_i = \min\{f_i(\boldsymbol{x}) | \boldsymbol{x} \in \Omega\}$.

Therefore, a good representation of the Pareto front can be obtained by solving a set of problems defined by a well-distributed set of weighting vectors. This is the main idea behind current decomposition-based MOEAs—see for example [8, 12, 13].

2.3 MOEA/D Framework

The Multi-Objective Evolutionary Algorithm Based on Decomposition (MOEA/D) [8], transforms a MOP into several scalarization problems. Therefore, an approximation of the Pareto front is obtained by solving the N scalarization subproblems in which a MOP is decomposed.

Considering $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ as the set of evenly spread weighting vectors, MOEA/D finds the best solution of each subproblem defined by each weighting vector using the PBI approach. The objective function of the j^{th} subproblem is then defined by $g(\boldsymbol{x}|\boldsymbol{w}_j, \boldsymbol{z})$, where $\boldsymbol{w}_j \in W$ and $\boldsymbol{z} = (z_1, \ldots, z_k)^T$ is the artificial utopian vector whose component z_i is the minimum value found so far for the objective f_i. In MOEA/D, a neighborhood of the weighting vector $\boldsymbol{w}_i$ is defined as a set of its closest weighting vectors in W. Therefore, the neighborhood of the i^{th} subproblem consists of all the subproblems with the weighting vectors from the neighborhood of $\boldsymbol{w}_i$ and it is denoted by $B(\boldsymbol{w}_i)$.

At each generation, MOEA/D finds the best solution to each subproblem throughout the evolutionary process and maintains: 1) a population of N points $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$, where $\boldsymbol{x}_i \in \Omega$ is the current solution to the i^{th} subproblem; 2) $FV^1, \ldots, FV^N$, where FV^i is the F-value of $\boldsymbol{x}_i$, i.e., $FV^i = \boldsymbol{F}(\boldsymbol{x}_i)$ for each $i = 1, \ldots, N$; 3) an external population EP, which is used to store the nondominated solutions found during the search. Algorithm 1 presents the general framework of MOEA/D, although the interested reader can be referred to [8] for a more detailed description.

2.4 Radial Basis Function Networks

Radial Basis Function (RBF) networks are a feed-forward kind of neural network, which are commonly represented with three layers: an input layer with n nodes, a hidden layer with h nonlinear RBFs (or neurons), and an output layer with k nodes. The function value in a RBF depends on the distance from each point $\boldsymbol{x}$ to the origin, i.e. $g(\boldsymbol{x}) = g(||\boldsymbol{x}||)$. This function value can be generalized to distances from some other point $\boldsymbol{c}_j$, commonly called *center* of the basis function, that is:

$$g(\boldsymbol{x}, \boldsymbol{c}_j) = g(||\boldsymbol{x} - \boldsymbol{c}_j||)$$

The z^{th} output $\varphi_z : \mathbb{R}^n \mapsto \mathbb{R}$ of the network is defined as:

$$\varphi_z(\boldsymbol{x}) = \sum_{j=1}^{h} w_j g(||\boldsymbol{x} - \boldsymbol{c}_j||), \quad z = 1, \ldots, k \tag{3}$$

where h is the number of neurons in the hidden layer, $\boldsymbol{c}_j$ is the center vector for the j^{th} neuron, and w_j's are the weights of the linear output neuron. In its basic form, all inputs are connected to each hidden neuron. The norm is typically taken to be the Euclidean distance and the basis function g or kernel is taken to be Gaussian, although other basis functions are also possible (see for example those shown in Table 1).

Algorithm 1: General Framework of MOEA/D

Input:
a stopping criterion;
N: the number of the subproblems considered in MOEA/D;
W: a well-distributed set of weighting vectors $\{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$;
T: the number of weight vectors in the neighborhood of each weighting vector.
Output:
EP: the nondominated solutions found during the search;
P: the final population found by MOEA/D.

begin
  $EP = \emptyset$;
  Generate an initial population $P = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ randomly;
  $FV^i = \boldsymbol{F}(\boldsymbol{x}_i)$;
  $B(\boldsymbol{w}_i) = \{\boldsymbol{w}_{i_1}, \ldots, \boldsymbol{w}_{i_T}\}$ where $\boldsymbol{w}_{i_1}, \ldots, \boldsymbol{w}_{i_T}$ are the T closest weighting vectors to $\boldsymbol{w}_i$, for each $i = 1, \ldots, N$;
  $\boldsymbol{z} = (+\infty, \ldots, +\infty)^T$;
  while *stopping criterion is not satisfied* **do**
    for $\boldsymbol{x}_i \in P$ **do**
      Reproduction: Randomly select two indexes k, l from $B(\boldsymbol{w}_i)$, and then generate a new solution $\boldsymbol{y}$ from $\boldsymbol{x}_k$ and $\boldsymbol{x}_l$ by using genetic operators.
      Mutation: Apply a mutation operator on $\boldsymbol{y}$ to produce $\boldsymbol{y}'$.
      Update of $\boldsymbol{z}$: For each $j = 1, \ldots, k$, if $z_j < f_j(\boldsymbol{x})$, then set $z_j = f_j(\boldsymbol{y}')$.
      Update of Neighboring Solutions: For each index $j \in B(\boldsymbol{w}_i)$, if $g(\boldsymbol{y}'|\boldsymbol{w}_j, \boldsymbol{z}) \leq g(\boldsymbol{x}_j|\boldsymbol{w}_j, \boldsymbol{z})$, then set $\boldsymbol{x}_j = \boldsymbol{y}'$ and $FV^j = \boldsymbol{F}(\boldsymbol{y}')$.
      Update of EP: Remove from EP all the vectors dominated by $\boldsymbol{F}(\boldsymbol{y}')$. Add $\boldsymbol{F}(\boldsymbol{y}')$ to EP if no vectors in EP dominate $\boldsymbol{F}(\boldsymbol{y}')$.
    end
  end
end

RBF networks can be used to interpolate a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ when the values of that function are known on a finite number of points: $f(\boldsymbol{x}_i) = y_i, i = 1 \ldots, N$. Taking into account the h centers $\boldsymbol{c}_j$'s ($j = 1, \ldots, h$) and evaluating the values of the basis functions at the points $\boldsymbol{x}_i$, i.e., $\phi_{ij} = g(||\boldsymbol{c}_j - \boldsymbol{x}_i||, \sigma_j)$ the weights can be solved from the equation:

$$\begin{pmatrix} \phi_{11} & \phi_{12} & \ldots & \phi_{1m} \\ \phi_{21} & \phi_{22} & \ldots & \phi_{2m} \\ \vdots & & \ddots & \vdots \\ \phi_{N1} & \phi_{N2} & \ldots & \phi_{Nm} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_m \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix} \tag{4}$$

Therefore, the weights w_i's can be solved by simple linear algebra, using the least squares method, that is:

$$\boldsymbol{w} = (\boldsymbol{\Phi}^T \boldsymbol{\Phi})^{-1} \boldsymbol{\Phi}^T \boldsymbol{y} \tag{5}$$

Table 1. Kernels for a RBF neural network, where $r = ||\boldsymbol{x} - \boldsymbol{c}_i||$

Kernel	Description
Cubic	$g(r) = r^3$
Thin Plate Spline	$g(r) = r^2 \ln(r)$
Gaussian	$g(r, \sigma) = \exp(-r^2/2\sigma^2)$
Multi-quadratic	$g(r, \sigma) = \sqrt{r^2 + \sigma^2}$
Inverse multi-quadratic	$g(r, \sigma) = \sqrt{r^2 + \sigma^2}$

The parameter σ_j of the kernels (Gaussian, multi-quadratic and inverse multi-quadratic) determines the amplitude of each basis function and it can be adjusted to improve the model accuracy.

3 Our Proposed Approach

3.1 General Framework

Our proposed MOEA/D assisted by Radial Basis Functions (MOEA/D-RBF) decomposes the MOP (1) into N single-objective optimization problems. MOEA/D-RBF uses a well-distributed set of N weight vectors $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ to define a set of single-objective optimization subproblems by using the PBI approach. Each subproblem is solved by MOEA/D, which is assisted by a surrogate model based on RBF networks. For a better understanding of the proposed approach, Algorithm 2 shows the general framework of the proposed MOEA/D-RBF. In the following sections, we describe in detail the components of our MOEA/D-RBF which are outlined in Algorithm 2.

3.2 Initialization

Initially, a training set $T_{set} = \{\boldsymbol{x}_i, \ldots, \boldsymbol{x}_{N_t}\}$ of N_t well-spread solutions is generated. For this task, we employed the Latin hypercube sampling method [14]. The set of solutions T_{set} is evaluated by using the real fitness function. The number of current fitness function evaluations n_{eval} is initially set as $n_{eval} = N_t$. MOEA/D-RBF uses an external archive A to store the nondominated solutions found so far in the evolutionary process. This archive is initialized with the nondominated solutions found in T_{set}. At the beginning, a population $\hat{P} = \{\boldsymbol{x}_i, \ldots, \boldsymbol{x}_N\}$ of N solutions is generated by employing the Latin hypercube sampling method. The stopping criterion considered in MOEA/D-RBF is the number of fitness function evaluations and, therefore, the stopping criterion is initially set as false, i.e. $stopping\_criterion = FALSE$.

3.3 Building the Model

As previously indicated, we use a surrogate model based on RBF networks. In order to improve the prediction of the surrogate model, the Gaussian, the multi-quadratic and the inverse multi-quadratic kernels are used in a cooperative way

Algorithm 2: General framework of MOEA/D-RBF

Input:
$W = \{\boldsymbol{w}_i, \ldots, \boldsymbol{w}_N\}$: A well-distributed set of weight vectors.
N_t: The number of points in the initial training set.
E_{max}: The maximum number of evaluations allowed in MOEA/D-RBF.
Output:
A: An approximation to the PF.

begin
  Initialization: Generate a set $T_{set} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_t}\}$ of N_t points such that $\boldsymbol{x}_i \in \Omega$ $(i = 1, \ldots N_t)$, by using an experimental design method. Evaluate the $\boldsymbol{F}$-functions values of these points. Set A as the set of nondominated solutions found in T_{set}. Set $n_{eval} = N_t$. Generate a population $\hat{P} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$ of N individuals such that $\boldsymbol{x}_i \in \Omega$ $(i = 1, \ldots N)$, by using an experimental design method. $stopping\_criterion = FALSE$. For details of this step see **Section 3.2**.
  while $(stopping\_criterion == FALSE)$ **do**
    Model Building: Using the $\boldsymbol{F}$-function values of the points in T_{set}, build the predictive surrogate model by using different RBF networks. Calculate the weights for each RBF network according to its training error in T_{set}. For details of this step see **Section 3.3**.
    Evaluate $\hat{P}$: Evaluate the population $\hat{P}$ using the surrogate model.
    Find an approximation to PF: By using MOEA/D, the surrogate model and the population $\hat{P}$, obtain $\hat{P}^\star = \{\hat{\boldsymbol{x}}_i, \ldots, \hat{\boldsymbol{x}}_{N_t}\}$, where $\hat{P}^\star$ is an approximation to PF, see **Section 3.4**.
    Select points for updating T_{set}: By using the selection scheme, select a set of solutions from $\hat{P}^\star$ to be evaluated and included in the training set T_{set}. Update A using the selected solutions. For each evaluated solution, set $n_{eval} = n_{eval} + 1$. **If** $n_{eval} < E_{max}$ **then** $stopping\_criteria = TRUE$. For a detailed description of this step see **Section 3.5**.
    Update population $\hat{P}$: Update the population $\hat{P}$ according to the updating scheme, see **Section 3.6**.
  end
  return A;
end

for obtaining the approximated value of a solution. In the following sections, we describe the necessary components for building the surrogate model.

Hidden Nodes The hidden nodes in an RBF network play an important role in the performance of the RBF network. In general, there is no method available for estimating the number of hidden nodes in an RBF network. However, it has been suggested in [15–18] that Kolmogorov's theorem [19] concerning the realization of arbitrary multivariate functions, provides theoretical support for neural networks that implement such functions.

Theorem 1 (Kolmogorov [19]). *A continuous real-valued function defined as* $f : [0,1]^n \mapsto \mathbb{R}$, $n \geq 2$, *can be represented in the form:*

$$f(x_1, \ldots, x_n) = \sum_{j=1}^{2n+1} g_j \left(\sum_{i=1}^{n} \phi_{ij}(x_j) \right) \tag{6}$$

where the g_j*'s are properly chosen continuous functions of one variable, and the* ϕ_{ij}*'s are continuous monotonically increasing functions independent of* f*.*

The basic idea in Kolmogorov's theorem is captured in the network architecture of Figure 1, where a universal transformation M maps $\mathbb{R}^n$ into several unidimensional transformations. The theorem states that one can express a continuous multivariate function on a compact set in terms of sums and compositions of a finite number of single variable functions.

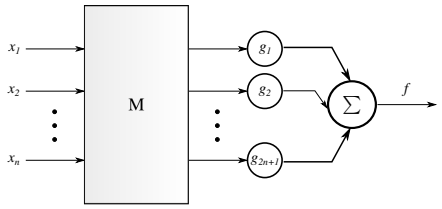

Fig. 1. Network representation of Kolmogorov's theorem

Motivated by this idea, the surrogate model built here, uses $2n + 1$ hidden nodes (where n is the number of decision variables of the MOP). Considering T_{set} as the training set of N_t solutions used by the surrogate model, the centers of the $2n+1$ basis functions are defined by using the well-known k-means algorithm [20] on the training set T_{set} (with $k = 2n + 1$). This criterion establishes that the cardinality of T_{set} should be greater than $2n + 1$, i.e., $2n + 1 < N_t$.

Building the surrogate model The high modality and dimensionality of some real-world functions, often produce problems to surrogate models. When the surrogate model is not able to properly shape the region of the search space in which the Pareto set is located, then the search may be biased towards inappropriate regions. In order to improve the function prediction, MOEA/D-RBF uses different kernels for building different RBF networks. Each RBF network provides different shape of the search space and all of them provide information to predict the value of an arbitrary solution. Here, three different kernels are adopted: Gaussian, multi-quadratic and inverse multi-quadratic; these kernels are chosen because they possess the parameter σ which can be adjusted to improve the model accuracy, see Table 1. Note however that other types of kernels can also be adopted, although the use of more kernels could significantly increase

the training time. In the following description, we consider the case with one single output node, i.e. with a single function. Note however, that this model can be generalized for more than one function.

Let $T_{set} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{N_t}\}$ be the set of N_t solutions evaluated with the real fitness function. Let h be the number of hidden nodes (or basis functions) considered in the RBF network. Let $\boldsymbol{c}_j$ and σ_j $(j = 1, \ldots, m)$ be the center and the amplitude of each basis function, respectively. The training of the RBF network for a determined kernel K consists in finding the weight vector $\boldsymbol{w} = (w_1, \ldots, w_m)^T$ such that it solves equation (5). Each parameter σ_j of each basis function is initially defined by the standard deviation of the solutions contained in each cluster obtained by the k-means algorithm (with mean $\boldsymbol{c}_j$).

Once the weight vector $\boldsymbol{w}$ is obtained, the model accuracy is improved by adjusting the vector of parameters $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_m)^T$. Since the value of the adopted kernel depends of σ_j, from equation (4), the training error on the training set T_{set}, can be written as:

$$\psi(\boldsymbol{\sigma}) = ||\boldsymbol{\Phi}\boldsymbol{w} - \boldsymbol{y}|| \tag{7}$$

where $\boldsymbol{y} = (y_1, \ldots, y_{Nt})^T$ is the vector of the real function values for each solution $\boldsymbol{x}_i \in T_{set}$, i.e., $y_i = f(\boldsymbol{x}_i)$. $\boldsymbol{\Phi}$ is the matrix which contains the evaluations of each point $\boldsymbol{x}_i \in T_{set}$ for each basis function, i.e., $\phi_{ij} = g(||\boldsymbol{c}_j - \boldsymbol{x}_i||, \sigma_j)$, for $i = 1, \ldots, N_t$ and $j = 1, \ldots, h$.

The parameters σ_j are then adjusted by using the Differential Evolution (DE) algorithm [21], whose objective is to minimize the training error defined in equation (7). Once the σ_j parameters are adjusted, the prediction function for a determined kernel K of a solution $\boldsymbol{x} \in \Omega$ can be calculated by:

$$\hat{\varphi}_K(\boldsymbol{x}) = \sum_{j=1}^{h} w_j \cdot g(||\boldsymbol{x} - \boldsymbol{c}_j||, \sigma_j) \tag{8}$$

Cooperative surrogate models and Function Prediction Once the three RBF networks are built, each of them using the three above mentioned kernels, the prediction of the function is carried out. Let $\varphi_{GK}(\boldsymbol{x}), \varphi_{MK}(\boldsymbol{x})$ and $\varphi_{IMK}(\boldsymbol{x})$ be the predicted value given by RBF networks using the Gaussian, multi-quadratic and inverse multi-quadratic kernel, respectively. These three RBF networks cooperate by providing information of the search space that they model. Therefore, the function prediction $\hat{f}$ for an arbitrary $\boldsymbol{x} \in \Omega$ is defined by:

$$\hat{f}(\boldsymbol{x}) = \lambda_1 \cdot \varphi_{GK}(\boldsymbol{x}) + \lambda_2 \cdot \varphi_{MK}(\boldsymbol{x}) + \lambda_3 \cdot \varphi_{IMK}(\boldsymbol{x}) \tag{9}$$

where $\boldsymbol{\Lambda} = (\lambda_1, \lambda_2, \lambda_3)^T$ is a weight vector, i.e. $\lambda_i \geq 0$ and $\sum_{i=1}^{3} \lambda_i = 1$. Therefore, the weight for each predicted value needs to be calculated.

Let T_{set} be the knowledge set for training the different RBF networks. The weight vector $\boldsymbol{\Lambda}$ is then calculated by:

$$\lambda_i = \frac{\alpha_i}{|T_{set}|}, i = 1, 2, 3 \tag{10}$$

where α_i is the number of solutions in T_{set} with the lowest prediction error for the i^{th} RBF network (Gaussian, multi-quadratic and inverse multi-quadratic, respectively).

3.4 Finding an Approximation to PF

MOEA/D-RBF approximates solutions to PF by using the well-know MOEA/D [8]. The search is conducted by the set of weight vectors $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$. MOEA/D searches the solutions to each scalar problem defined by each weight vector $\boldsymbol{w}_i \in W$. The evolutionary process of MOEA/D is performed during a determined number of generations by employing the prediction function defined in equation (9). The final population denoted as $\hat{P}^{\star}$ is then reported as an approximation to PF.

3.5 Selecting Points to Evaluate

Let $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ be the well-distributed set of weight vectors used by MOEA/D. Let $\hat{P}^{\star}$ be the approximation to PF obtained by MOEA/D. Let $W_s = \{\boldsymbol{w}^s_1, \ldots, \boldsymbol{w}^s_{N_s}\}$ be a well-distributed set of weight vectors, such that $|W_s| < |W|$. For each $\boldsymbol{w}^s_i \in W_s$, we define $B_s(\boldsymbol{w}^s_i) = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{N_a}\}$, such that $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_{N_a} \in W$ are the $N_a = \lfloor \frac{N}{N_s} \rfloor$ closest weight vectors from W to $\boldsymbol{w}^s_i$. With that, an association of weight vectors from W to W_s is defined. This association defines a set of neighborhoods $B_s(\boldsymbol{w}^s_i)$ which are distributed along the whole set of weight vectors W, see Figure 2. Once the neighborhoods $B_s(\boldsymbol{w}^s_i)$ have been defined, a set of solutions is selected to be included in the training set T_{set}, according to the next description.

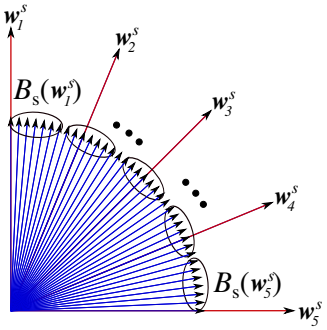

Fig. 2. Association of weight vectors from W to W_s. The vectors in blue represent the set W, while the vectors in red define the set W_s. This association defines the neighborhoods $B_s(\boldsymbol{w}^s_1)$ to $B_s(\boldsymbol{w}^s_5)$

Selecting Points to be Evaluated using the Real Fitness Function A set $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{w}_{N_s}\}$ of N_s solutions taken from $\hat{P}$ is chosen to be evaluated using the real fitness function. Each solution in S is selected such that it minimizes the problem defined by a weight vector $\boldsymbol{w}_j \in B_s(\boldsymbol{w}_i^s)$, where $i = 1, \ldots, N_s$ and $j = 1, \ldots N_a$.

At each call of the selection procedure, the weight vector $\boldsymbol{w}_j$ is selected by sweeping the set of weight vectors in $B_s(\boldsymbol{w}_i^s)$ in a cyclic way, i.e., once the last weight vector is selected, the next one is picked up from the beginning. Since the neighborhoods $B_s(\boldsymbol{w}_i^s)$ are distributed along the whole weight set W, the selection of solutions in each neighborhood should obtain spread solutions along the PF. No solution in S should be duplicated. If this is the case, the repeated solution should be removed from S. For each new evaluated solution, we set $n_{eval} = n_{eval} + 1$, if $n_{eval} \geq E_{max}$ then we set $stopping\_criterion = TRUE$, where n_{eval} and E_{max} are the current and the maximum number of fitness function evaluations, respectively.

Updating the Training Set and the External Archive The maximum number of solutions in the training set T_{set} is defined by the parameter N_t. The updating of T_{set} is carried out by defining a well-distributed set of N_t weight vectors $W_t = \{\boldsymbol{w}_1^t, \ldots, \boldsymbol{w}_{N_t}^t\}$. Therefore, the best N_t different solutions from $T = \{T_{set} \cup S\}$, such that they minimize the subproblems defined by each weight vector $\boldsymbol{w}_i^t \in W_t$, are used to update T_{set}. If after updating the training set, any solution $\boldsymbol{s}_j \in S$ was not selected to be included in T_{set}, then, it is added by replacing the closest solution (in the objective space) in T_s. With this, all solutions in S are included in T_{set} and the model can be improved even if it has been misinformed formerly.

The external archive A contains the nodominated solutions found along the search. For each $\boldsymbol{s}_j \in S$, the external archive is updated by removing from A all the solutions dominated by $\boldsymbol{s}_j$, and then, $\boldsymbol{s}_j$ is stored in A if no solutions in A dominate $\boldsymbol{s}_i$.

3.6 Updating the Population

Once the external archive is updated, the population $\hat{P}$ is also updated for the next iteration of MOEA/D. Considering the external archive A as the set of nondominated solutions found by MOEA/D-RBF, the population $\hat{P}$ of N solutions is updated according to the following description.

Let $\boldsymbol{m}$ and $\boldsymbol{\sigma}$ be the average and standard deviation of the solutions contained in A. Then, new bounds in the search space are defined according to:

$$\begin{aligned} \boldsymbol{L}_{bound} &= \boldsymbol{m} - \boldsymbol{\sigma} \\ \boldsymbol{U}_{bound} &= \boldsymbol{m} + \boldsymbol{\sigma} \end{aligned}$$

where $\boldsymbol{L}_{bound}$ and $\boldsymbol{U}_{bound}$ are the vectors which define the lower and upper bounds of the new search space, respectively.

Once the new bounds have been defined, a well-distributed set Q of $N - |A|$ solutions is generated by means of the Latin hypercube sampling method [14] in the new search space. The population $\hat{P}$ is then redefined by the union of Q and A, that is $\hat{P} = \{Q \cup A\}$.

4 Test Problems

In order to assess the performance of our proposed approach (MOEA/D-RBF), we compare its results with respect to those obtained by the original MOEA/D [8] and by MOEA/D-EGO [7], which uses a surrogates model (based on the Gaussian stochastic process model). We adopted the Zitzler-Deb-Thiele (ZDT) test problems [22] except for ZDT5 (which is a discrete problem). The detailed description of such problems can be found in [22]. In order to evaluate the capabilities of MOEA/D-RBF when dealing with computationally expensive problems, we also tested its performance using a real-world problem, as a case study. Next, we describe the airfoil problem which was adopted to validate our proposed approach.

4.1 Airfoil Shape Optimization: A case study

Our case study consists of the multi-objective optimization of an airfoil shape problem adapted from [23] (called here MOPRW). This problem corresponds to the airfoil shape optimization of a standard-class glider, aiming to obtain an optimum performance for a sailplane.

Problem Statement Two conflicting objective functions are defined in terms of a sailplane average weight and operating conditions [23]. They are defined as:

i) minimize: C_D/C_L
s.t. $C_L = 0.63$, $Re = 2.04 \cdot 10^6$, $M = 0.12$
ii) minimize: $C_D/C_L^{3/2}$
s.t. $C_L = 1.05$, $Re = 1.29 \cdot 10^6$, $M = 0.08$

where C_D/C_L and $C_D/C_L^{3/2}$ correspond to the inverse of the glider's gliding ratio and sink rate, respectively. Both are important performance measures for this aerodynamic optimization problem. C_D and C_L are the drag and lift coefficients.

The aim is to maximize the gliding ratio (C_L/C_D) for objective (i), while minimizing the sink rate in objective (ii). Each of these objectives is evaluated at different prescribed flight conditions, given in terms of Mach and Reynolds numbers. The aim of solving this MOP is to find a better airfoil shape, which improves a reference design.

Table 2. Parameter ranges for our modified PARSEC airfoil representation

Design Variable	Lower Bound	Upper Bound
r_{leup}	0.0085	0.0126
r_{lelo}	0.0020	0.0040
α_{te}	7.0000	10.0000
β_{te}	10.0000	14.0000
Z_{te}	-0.0060	-0.0030
ΔZ_{te}	0.0025	0.0050
X_{up}	0.4100	0.4600
Z_{up}	0.1100	0.1300
Z_{xxup}	-0.9000	-0.7000
X_{lo}	0.2000	0.2600
Z_{lo}	-0.0230	-0.0150
Z_{xxlo}	0.0500	0.2000

Geometry Parametrization In the present case study, the PARSEC airfoil representation [24] was adopted. Fig. 3 illustrates the 11 basic parameters used for this representation: r_{le} leading edge radius, X_{up}/X_{lo} location of maximum thickness for upper/lower surfaces, Z_{up}/Z_{lo} maximum thickness for upper/lower surfaces, Z_{xxup}/Z_{xxlo} curvature for upper/lower surfaces, at maximum thickness locations, Z_{te} trailing edge coordinate, ΔZ_{te} trailing edge thickness, α_{te} trailing edge direction, and β_{te} trailing edge wedge angle.

For the present case study, the modified PARSEC geometry representation adopted here allows us to define independently the leading edge radius, both for upper and lower surfaces. Thus, a total of 12 decision variables are used. Their allowable ranges are defined in Table 2.

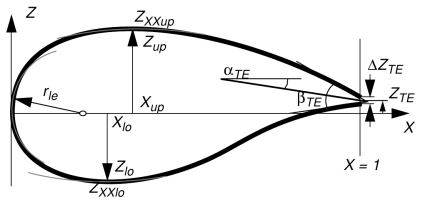

Fig. 3. PARSEC airfoil parametrization.

The PARSEC airfoil geometry representation uses a linear combination of shape functions for defining the upper and lower surfaces. These linear combinations are given by:

$$Z_{upper} = \sum_{n=1}^{6} a_n x^{\frac{n-1}{2}}, \quad Z_{lower} = \sum_{n=1}^{6} b_n x^{\frac{n-1}{2}} \tag{11}$$

In the above equations, the coefficients a_n, and b_n are determined as functions of the 12 described geometric parameters, by solving two systems of linear equations, one for each surface. It is important to note that the geometric parameters r_{leup}/r_{lelo}, X_{up}/X_{lo}, Z_{up}/Z_{lo}, Z_{xxup}/Z_{xxlo}, Z_{te}, ΔZ_{te}, α_{te}, and β_{te} are

the actual design variables in the optimization process, and that the coefficients a_n, b_n serve as intermediate variables for interpolating the airfoil's coordinates, which are used by the CFD solver (we used the Xfoil CFD code [25]) for its discretization process.

5 Comparison of Results

5.1 Performance Measure

In order to evaluate the performance of our proposed approach, we compared its results with respect to those obtained by MOEA/D and MOEA/D-EGO. For comparing these algorithms, we adopted the I_H^- performance measure which is described below.

I_H^- metric The Hypervolume (I_H) performance measure was proposed by Zitzler [26]. This performance measure is Pareto compliant [27] and quantifies the approximation of nondominated solutions to the Pareto optimal front. The hypervolume corresponds to the non-overlapped volume of all the hypercubes formed by a reference point $\boldsymbol{r}$ (given by the user) and each solution p in the Pareto set approximation (P). It is mathematically defined as:

$$I_H(P) = \Lambda\left(\bigcup_{\boldsymbol{p}\in P}\{\boldsymbol{x}|\boldsymbol{p}\prec\boldsymbol{x}\prec\boldsymbol{r}\}\right)$$

where Λ denotes the Lebesgue measure and $\boldsymbol{r}\in\mathbb{R}^k$ denotes a reference vector being dominated by all valid candidate solutions in P. The I_H^- performance measure is then defined as:

$$I_H^-(P^\star, P) = I_H(P^\star) - I_H(P)$$

where $I_H(P^\star)$ is the hypervolume between the Pareto optimal front $P^\star$ and a reference point $\boldsymbol{r}$. I_H^- assesses both convergence and spread of the Pareto front. A low I_H^- value, indicates that our approximation P is close to PF and has a good spread towards the extreme portions of the Pareto front.

In our experiments, as in [7], we select 500 evenly distributed points on the Pareto optimal front PF and let these points be $P^\star$ for each standard test problem. Since the Pareto optimal front of the the airfoil shape problem is unknown, only the I_H performance measure is used. In this case, obtaining a high I_H value, indicates that our approximation P is close to PF.

5.2 Experimental Setup

As indicated before, the proposed approach is compared with respect to MOEA/D and MOEA/D-EGO. For each MOP, 30 independent runs were performed with each algorithm. As in [7], the number of decision variables is set to be eight

for the ZDT benchmark. Each algorithm was restricted to 200 fitness function evaluations and the results of MOEA/D-EGO were directly taken from [7].

For the airfoil design problem, the search was restricted to 5,000 fitness function evaluations. Since the computational complexity of the model building in MOEA/D-EGO increases with the number of training points, MOEA/D-EGO becomes impractical as the number of fitness function evaluations increases, see [7]. Because of this, the comparison of results in the airfoil design problem is carried out only between MOEA/D and MOEA/D-RBF. In addition, we also show the results obtained by MOEA/D-RBF and MOEA/D using 30 and 10 decision variables for the ZDT test problems, as it was suggested by Zitzler et al. [22]. Since the difficulty to solve the ZDT test problems increases with respect to the number of decision variables, the search for these algorithms was restricted to 1,000 fitness function evaluations.

The parameters used for both MOEA/D and MOEA/D-RBF, were set as in [8], since there is empirical evidence that indicates that these are the most appropriate parameters for solving the ZDT test suite—see [8]. The weight vectors for the algorithms were generated as in [8], i.e., the setting of N and $W = \{\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N\}$ is controlled by a parameter H. More precisely, $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_N$ are all the weight vectors in which each individual weight takes a value from

$$\left\{ \frac{0}{H}, \frac{1}{H}, \ldots, \frac{H}{H} \right\}$$

Therefore, the number of such vectors in W is given by:

$$N = C_{H+k-1}^{k-1}.$$

where k is the number of objective functions.

For all the MOPs, MOEA/D was tested with $H = 99$, i.e. $N = 100$ weight vectors. For MOEA/D-RBF (and for MOEA/D-EGO) $H = 299$, i.e. 300 weight vectors. The set W_t was generated with $H = 10n - 1$, therefore $N_t = 10n$ weight vectors (which define the cardinality of the training set), where n is the number of decision variables of the MOP. The set W_s uses $H = 9$, i.e. $N_s = 10$ weight vectors. For adjusting the parameter σ in the model building, the DE/rand/1/bin strategy was adopted. The parameters for DE were set to: $CR = 0.5, F = 1$, a population size 20 and the number of generations was restricted to 50 generations. The execution of the algorithms was carried out on a computer with a 2.66GHz processor and 4GB in RAM.

As indicated before, the algorithms were evaluated using the I_H^- and the I_H metrics. The results obtained are summarized in Tables 3–5. These tables display both the *average* and the standard deviation (σ) of the I_H^- and I_H indicators for each MOP, respectively. The reference vector $\boldsymbol{r}$ used for computing the performance measures, for each MOP, is shown in each table of results. For an easier interpretation, the best results are presented in **boldface** for each test problem adopted.

Table 3. Results of the I_H^- performance measure for MOEA/D-RBF, MOEA/D-EGO and MOEA/D, using eight decision variables

MOP	MOEA/D-RBF	MOEA/D-EGO	MOEA/D	reference vector ($\boldsymbol{r}$)
	average (σ)	*average* (σ)	*average* (σ)	
ZDT1	**0.009170** (0.000890)	0.067200 (0.080100)	17.209529 (3.310987)	$(10, 10)^T$
ZDT2	**0.009703** (0.001520)	0.019800 (0.003500)	26.223664 (3.945007)	$(10, 10)^T$
ZDT3	**0.115358** (0.898476)	0.117800 (0.070700)	41.218250 (7.307372)	$(20, 20)^T$
ZDT4	**1681.548786** (469.374655)	1709.520000 (593.510000)	1749.365260 (436.261557)	$(50, 50)^T$
ZDT6	1.690381 (2.006687)	**0.100000** (0.100900)	68.134088 (2.672056)	$(10, 10)^T$

5.3 Results and Discussion

ZDT Test Problems Table 3 shows the results obtained for the I_H performance measure when the algorithms are tested on the ZDT test problems using eight decision variables. From this table, it can be clearly seen that the best results were obtained by MOEA/D-RBF and MOEA/D-EGO. With respect to MOEA/D-RBF and MOEA/D-EGO, it is possible to see that MOEA/D-RBF obtained better results than those obtained by MOEA/D-EGO in most of the adopted MOPs. The exception was ZDT6 where MOEA/D-EGO was significantly better than MOEA/D-RBF. However, this does not mean that the performance of MOEA/D-RBF is bad. The average CPU times of 30 independent runs performed by MOEA/D-RBF oscillate between 15 and 16 seconds for each ZDT test problem. According to studies by Zhang et al. [7] MOEA/D-EGO (using the Fuzzy C means method) employed between 936 and 1,260 seconds for solving ZDT1 and ZDT2, respectively. From such studies and the one reported here, we conclude that MOEA/D-RBF is much faster than MOEA/D-EGO, when solving the ZDT test problems using eight decision variables.

In Table 4, we show the results obtained by MOEA/D-RBF and the original MOEA/D in the ZDT test problems using 30 and 10 decision variables, respectively (for a detailed description see [22]). From this table it is possible to see that MOEA/D-RBF obtained a better approximation to PF than the one achieved by MOEA/D in most of the test problems adopted. This comparison was performed in order to show the effectiveness of our proposed MOEA/D-RBF in terms of the quality of the solutions that it reached with respect to the solutions obtained by MOEA/D. As we can see, MOEA/D-RBF significantly outperformed MOEA/D in most of the ZDT test problems. The exception was ZDT4 which is a multi-frontal MOP which evidently causes difficulties to the surrogate model proposed here.

Airfoil Design Problem According to Table 5, we can see that MOEA/D-RBF obtained better hypervolume values than MOEA/D. Figure 4 shows the convergence graph for the I_H performance measure. From this graph, we can see

Table 4. Results of the I_H^- performance measure for MOEA/D-RBF and MOEA/D, using 30 and 10 decision variables, respectively

MOP	MOEA/D-RBF *average* (σ)	MOEA/D *average* (σ)	reference vector ($\boldsymbol{r}$)
ZDT1	**0.004831** (0.000382)	9.758961 (1.235485)	$(5,5)^T$
ZDT2	**0.006654** (0.000744)	14.040807 (1.806522)	$(5,5)^T$
ZDT3	**2.298589** (0.952347)	11.029210 (1.526359)	$(5,5)^T$
ZDT4	1828.493909 (382.604894)	**575.670683** (185.030230)	$(50,50)^T$
ZDT6	**1.244183** (0.996797)	22.057469 (2.385873)	$(5,5)^T$

that the results obtained by MOEA/D with 5,000 fitness function evaluations were achieved by our MOEA/D-RBF using a lower number of fitness function evaluation (only 1,250 evaluations, on average). MOEA/D employed, on average, 5,050 seconds to achieve the value reported in Table 5 in the I_H performance measure, while our MOEA/D-RBF required 2,000 seconds to achieve a similar value. Thus, we argue that our proposed MOEA/D-RBF is a good choice for dealing with computationally expensive MOPs. The approximations to the Pareto front for this problem is presented in Figure 5, which corresponds to the set of nondominated solutions found by each algorithm in the run with the value nearest to the mean value of the I_H performance measure.

Table 5. Results of the I_H performance measure for MOEA/D-RBF and MOEA/D for the airfoil design problem

MOP	MOEA/D-RBF *average* (σ)	MOEA/D *average* (σ)	reference vector ($\boldsymbol{r}$)
MOPRW1	**2.493786e-07** (6.483342e-09)	2.149916e-07 (2.446593e-08)	$(0.007610, 0.005236)^T$

6 Conclusions and Future Work

We have proposed here a version of MOEA/D which is assisted by cooperative RBF networks, with the aim of improving the prediction of the function value. The RBF networks employed here, use different kernels in order to have different shapes of the fitness landscape. With that, each RBF network provides information which is used to improve the value of the objective function. According to the results reported here, our proposed MOEA/D-RBF is able to outperform both to the original MOEA/D and to MOEA/D-EGO (which is a version of MOEA/D that also adopts surrogates) when performance only 200 and 1,000 fitness function evaluations. We also validated our proposed approach with a

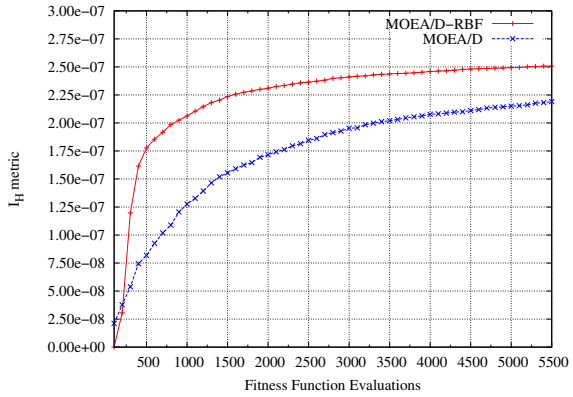

Fig. 4. I_H convergence graph for the airfoil design problem

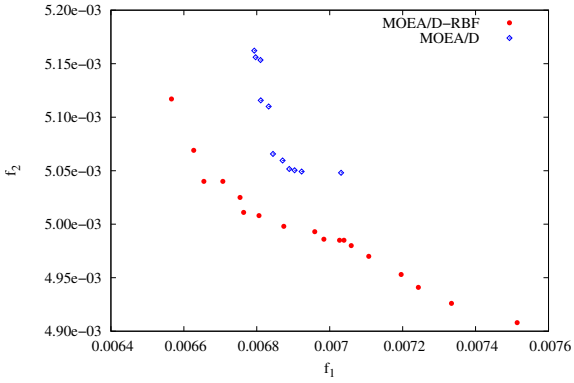

Fig. 5. Approximation to PF given by MOEA/D-RBF and MOEA/D for the airfoil design problem

real-world computationally expensive multi-objective optimization problem: airfoil design. In this case, our proposed MOEA/D-RBF was able to reduce by more than half the CPU time required by MOEA/D to achieve a certain I_H value. This illustrates the potential of our proposed approach for solving computationally expensive multi-objective problems.

As part of our future work, we plan to use our approach in problems having three or more objectives. Also, we pretend to couple to our approach a local search mechanism, so that, while MOEA/D-RBF obtains candidate solutions to be evaluated with the real fitness function, the local search refines these solutions, thus accelerating the convergence towards the true Pareto front. Finally, we are also interested in testing our approach with more real-world problems having more decision variables, and that is indeed part of our ongoing research.

References

1. Coello Coello, C.A., Lamont, G.B., Van Veldhuizen, D.A.: Evolutionary Algorithms for Solving Multi-Objective Problems. Second edn. Springer, New York (2007) ISBN 978-0-387-33254-3.
2. Ong, Y.S., Nair, P.B., Keane, A.J.: Evolutionary optimization of computationally expensive problems via surrogate modeling. AIAA Journal **41**(4) (2003) 687–696
3. Emmerich, M.T.M., Giannakoglou, K., Naujoks, B.: Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. IEEE Transactions on Evolutionary Computation **10**(4) (2006) 421–439
4. Knowles, J.: Parego: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation **10**(1) (2006) 50–66
5. Isaacs, A., Ray, T., Smith, W.: An evolutionary algorithm with spatially distributed surrogates for multiobjective optimization. In: ACAL. (2007) 257–268
6. Zapotecas Martínez, S., Coello Coello, C.A.: A Memetic Algorithm with Non Gradient-Based Local Search Assisted by a Meta-Model. In: PPSN XI. Volume 6238., Springer. Lecture Notes in Computer Science (2010) 576–585
7. Zhang, Q., Liu, W., Tsang, E., Virginas, B.: Expensive Multiobjective Optimization by MOEA/D with Gaussian Process Model. Evolutionary Computation, IEEE Transactions on **14**(3) (2010) 456 –474
8. Zhang, Q., Li, H.: MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. IEEE Transactions on Evolutionary Computation **11**(6) (2007) 712–731
9. Miettinen, K.: Nonlinear Multiobjective Optimization. Kluwer Academic Publishers, Boston, Massachuisetts (1999)
10. Ehrgott, M.: Multicriteria Optimization. 2nd edition edn. Springer, Berlin (2005)
11. Das, I., Dennis, J.E.: Normal-boundary intersection: a new method for generating Pareto optimal points in multicriteria optimization problems. SIAM Journal on Optimization **8**(3) (1998) 631–657
12. Peng, W., Zhang, Q.: A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems. In: IEEE International Conference on Granular Computing, 2008. GrC 2008. (2008) 534 –537
13. Zapotecas Martínez, S., Coello Coello, C.A.: A Multi-objective Particle Swarm Optimizer Based on Decomposition. In: GECCO'2011, Dublin, Ireland, ACM Press (2011) 69–76
14. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics **21**(2) (1979) 239–245
15. Hecht-Nielsen, R.: Kolmogorov's mapping neural network existence theorem. In: Proceedings of IEEE First Annual International Conference on Neural Networks. Volume 3. (1987) 11–14
16. Hecht-Nielsen, R.: Neurocomputing. Addison-Wesley, Redwood City, CA (1990)
17. Lippmann, R.P.: An introduction to computing with neural nets. IEEE Magazine on Accoustics, Signal, and Speech Processing **4** (1987) 4–22
18. Sprecher, D.A.: A universal mapping for kolmogorov's superposition theorem. Neural Netw. **6**(8) (1993) 1089–1094
19. Kolmogorov, A.K.: On the representation of continuous functions of several variables by superposition of continuous functions of one variable and addition. Doklady Akademii Nauk SSSR **114** (1957) 369–373

20. MacQueen, J.B.: Some Methods for Classification and Analysis of Multivariate Observations. In: Proceedings of the fifth Berkeley Symposium on Mathematical Statistics and Probability. Volume 1., University of California Press (1967) 281–297
21. Storn, R.M., Price, K.V.: Differential Evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical Report TR-95-012, ICSI, Berkeley, CA (1995)
22. Zitzler, E., Deb, K., Thiele, L.: Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. Evolutionary Computation **8**(2) (2000) 173–195
23. Szöllös, A., Smíd, M., Hájek, J.: Aerodynamic optimization via multi-objective micro-genetic algorithm with range adaptation, knowledge-based reinitialization, crowding and epsilon-dominance. Advances in Engineering Software **40**(6) (2009) 419–430
24. Sobieczky, H.: Parametric Airfoils and Wings. In Fuji, K., Dulikravich, G.S., eds.: Notes on Numerical Fluid Mechanics, Vol.. 68, Wiesbaden, Vieweg Verlag (1998) 71–88
25. Drela, M.: XFOIL: An Analysis and Design System for Low Reynolds Number Aerodynamics. In: Conference on Low Reynolds Number Aerodynamics, University Of Notre Dame, IN (1989)
26. Zitzler, E., Thiele, L.: Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In Eiben, A.E., ed.: PPSN V, Amsterdam, Springer-Verlag (1998) 292–301
27. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., da Fonseca, V.G.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation **7**(2) (2003) 117–132