European Congress on Computational Methods in Applied Sciences and Engineering
ECCOMAS 2004
P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, J. Périaux, and D. Knörzer (eds.)
Jyväskylä, 24–28 July 2004

ON THE USE OF SURROGATE EVALUATION MODELS IN MULTI-OBJECTIVE EVOLUTIONARY ALGORITHMS

Marios K. Karakasis* **and Kyriakos C. Giannakoglou***

* Laboratory of Thermal Turbomachines
National Technical University of Athens
Iroon Polytexniou 9, P.O. Box 64069, Athens 157 10, Greece
e-mails: mkarakasis@mail.ntua.gr, kgianna@central.ntua.gr

Key words: Multi-Objective Optimization, Metamodel-Assisted Optimization, Evolutionary Algorithms, Radial-Basis Function networks, Self-Organizing Maps

Abstract. *The use of surrogate evaluation models has found widespread use in evolutionary optimization. Regardless of the model itself and the implementation scheme, approximation models are used to replace exact but costly evaluations, leading thus to lower design computational cost. However, the gain in computational cost reduces considerably in Multi-Objective optimization problems, where the prediction capability of surrogate models worsens with respect to Single-Objective problems. The reasons for this behaviour along with remedies to alleviate the problem are exposed in this paper. Despite the fact that the surrogate model used herein is a Radial-Basis Function network, the proposed ideas can readily be extended to other models as well.*

1 INTRODUCTION – PROBLEM DEFINITION

Thorough comparisons of evolutionary and gradient-based optimization methods along with their advantages and disadvantages can be found in many textbooks or papers, so there is no need to focus on them any more. Well known are also the efforts of scientists working with Evolutionary Algorithms (EAs) to reduce their computational cost, in order to render their use affordable for industrial purposes. Whenever the evaluation software is costly, which is always the case in aeronautics and turbomachinery, the remedy to the aforementioned problem is one: to replace as many as possible of the exact and thus costly evaluations required by the EA with less exact but also less costly ones. This remedy calls for the development and use of *surrogate evaluation models* which are also refered to as *metamodels*. The latter exploit a number of previously evaluated solution patterns so as to approximate the performance of new candidate solutions which come up during the evolution [1]. Response surface methods, artificial neural networks (multilayer perceptron or radial-basis function, RBF, networks [2]) or Gaussian processes (including kriging [3]) can be used as surrogate evaluation models.

The authors of this paper represent a research group which, for many years, focused on the development of low-cost, metamodel-assisted EAs [4]; this work was recently included in a review paper [5]. For either Single- (SOO) or Multi-Objective optimization (MOO) problems, the method proposed and used by this group was conceptually based on the so-called Inexact Pre-Evaluation (IPE) technique, supported by on-line trained, local RBF networks (RBFNs). In the IPE algorithm, upon completion of the first few generations, which are needed in order to form a database (DB) of paired input-output sets, the population members of the subsequent generation are first evaluated using purposely devised local RBFNs and, then, only the most promising among them are exactly re-evaluated. Additional improvements to the standard EA-IPE algorithm can be achieved through its distributed variant D(EA-IPE) [6]; a distributed EA handles population subsets that evolve (using both the exact evaluation model and the metamodel) in a semi-isolated manner by regularly exchanging their best individuals.

However, it should be emphatically reported that the gain in computational cost through IPE technique worsens significantly as the number of objectives is increased. Even between single- and two-objective problems, the difference in the performance is noticeable. The most important reasons for this behaviour are listed below:

- In SOO problems, after the first generations, the population becomes clustered around the global optimal solution, in the vicinity of which there is adequate information for training the metamodel. In contrast, in MOO problems treated with the Pareto front approach, the available information is spread over a continuous or discontinuous front. Consequently, the metamodel predictions become less dependable and this badly affects the computational cost.

- Regardless of the method used to derive the single cost of a candidate solution in MOO (NSGA, SPEA, etc. [7, 8, 9]), checking the dominality or the strength of one

solution with respect to any other is a delicate procedure which might be seriously affected by errors in the metamodel predictions.

- In MOO, exactly re-evaluating only a small percentage of the population in each generation has an important impact on the number of solutions entering the Pareto front and affects both the quality and the extent of the front.

In view of the above, many questions about the most appropriate use of the IPE technique in MOO arise, such as

- Are RBFNs suitable for MOO problems? Should their parameters be more carefully determined in MOO?
- How is it possible to extract the maximum of information from the DB of already evaluated (though quite scattered, as previously metioned) solution patterns?
- How safely can a metamodel extrapolate and how outlying individuals should be detected and further treated?

Answers to the previous points will be given as the paper develops. Given the inherent differences between SOO and MOO, the proposed metamodel-related procedures aim at improving the performance of surrogate models; the ultimate goal is to make the IPE phase as efficient as in SOO.

2 THE EA-IPE ALGORITHM IN MULTI-OBJECTIVE OPTIMIZATION

The standard EA(μ, λ)-IPE[1] algorithm, as used in MOO problems ([5], Figure 1) is described below:

Phase 1: An initial population, with λ individuals, is generated at random.

Phase 2: The population evolves for a limited number (usually, two or three) of generations, with exact evaluations for all of its members. The parametric data for these individuals along with their cost function values are stored in a DB.

Phase 3: During the subsequent generations, for each individual:

(a) If the individual exists in the DB, its exact payoff value is restored and the algorithm proceeds with the next individual. Otherwise, a local RBFN is trained once the T closest DB entries to each new individual have been found.

(b) The trained RBFN is used to pre-evaluate the candidate solution; as it is evident, this inexact payoff value is not stored in the DB.

[1] μ and λ stand for the parent and offspring populations, respectively.

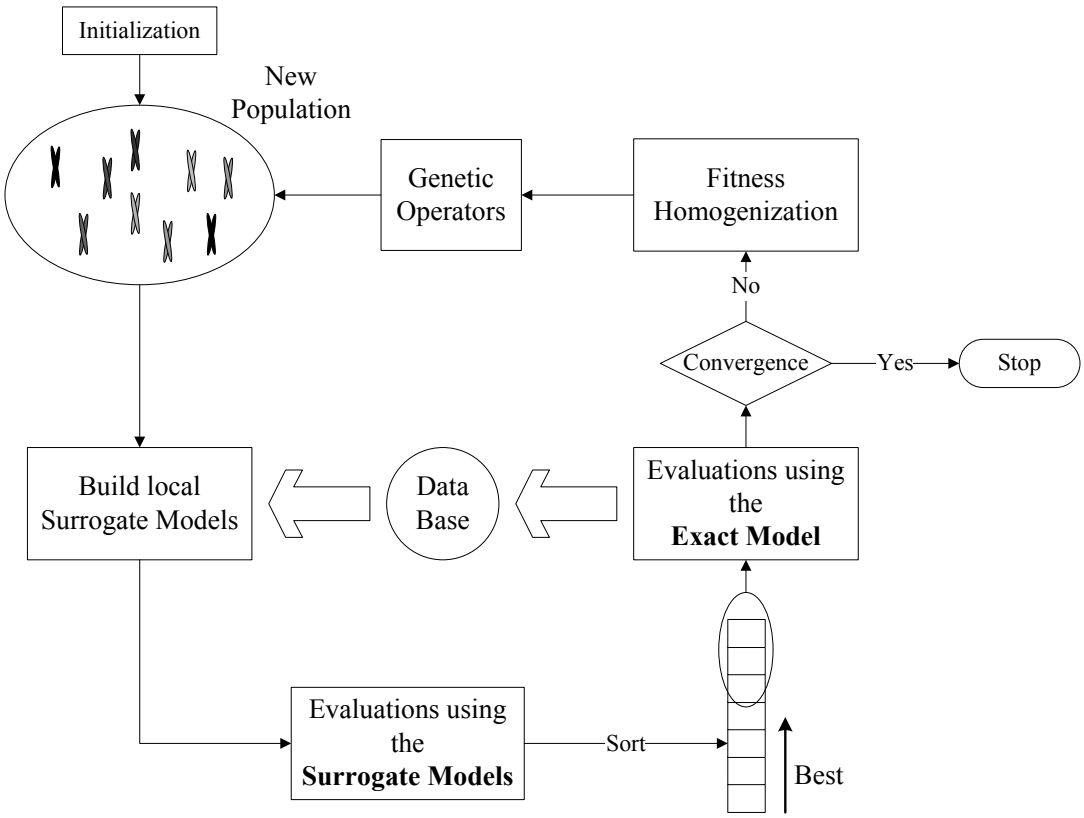


Figure 1: The Inexact Pre-Evaluation concept in Evolutionary Algorithms.

Phase 4: For each one of the λ individuals, a single cost value is computed using a user-selected method (such as NSGA, SPEA, etc.). In NSGA, for instance, by ranking the λ individuals in fronts by means of the standard nondomination criteria, a preliminary Pareto front is defined. Its members undergo exact evaluations through calls to the costly tool and then mix up with the previous Pareto front, which is updated accordingly. It should become clear that this procedure is based on the previously computed inexact payoff values. The $\sigma\lambda$, $0 < \sigma < 1$, most promising individuals are identified and those among them which have taken on inexact payoff values will be exactly re-evaluated. The so-computed exact payoff values overwrite the inexact ones and the DB is further enriched.

Phase 5: μ new parents are defined, from which, by merging exact and inexact scores, λ new offspring individuals are created. Phases 2 to 4 are repeated up to the final convergence.

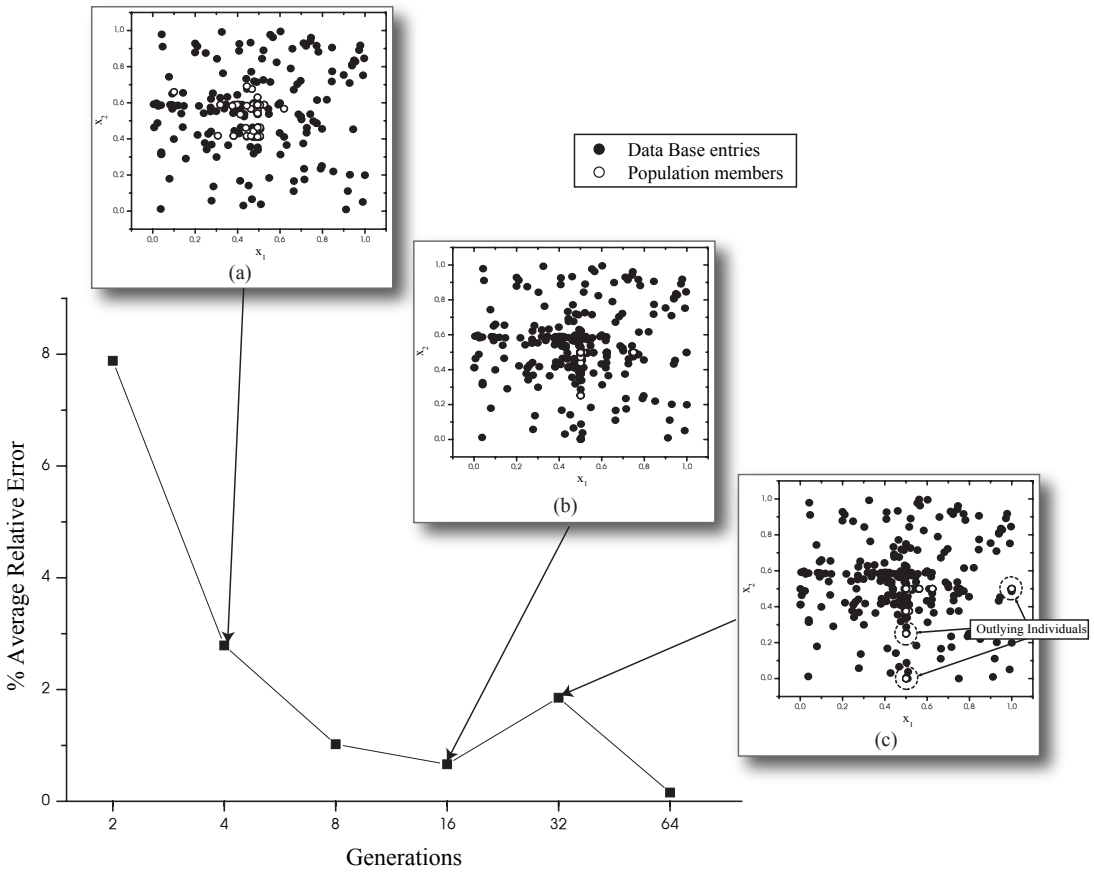


Figure 2: Single-Objective: Predictive capability of surrogate models in course of evolution.

3 THE USE OF METAMODELS IN SOO AND MOO

A principal trait of IPE, as described earlier, is the construction of surrogate models in the course of EA. The design space is not a priori probed at prescribed points; rather, the points used for training are those previously visited by the EA. This process is inevitably affected by the kind of solution sought: a "single-point" solution in SOO and a "set-of-points" solution in MOO treated with the Pareto front approach. The effect of these two different aims on building "on-line" surrogate models is investigated in this section.

As already stated, in SOO the solution consists of a single point. A well performing EA will progressively generate solutions hopefully clustered around the global optimum, with the exception of some outlying individuals. The latter stand for the results of mutation, which prevents the solution from being captured in local minima. This behaviour acts in favour of the metamodel and, therefore, its predictive capability is progressively increased as generations advance. This can readily be shown in the following example of minimizing

the Rastrigin function

$$f(x) = 10N + \sum_{i=1}^{N}(x_i^2 - 10\cos\ (2\pi x_i)) \tag{1}$$

where, for visualization purposes, the number of variables was $N = 2$, varying in $[-5.12, 5.12]$. Concerning the synthesis of the population, $\mu = \lambda = 80$.

At each generation, all the individuals were first evaluated with locally trained RBFNs, with the centers of the hidden nodes coinciding with the training patterns. The matter of pattern selection is addressed in Section 4, while the architecture of RBFNs is, in brief, revisited in Section 5. The entire population was subsequently exactly evaluated and the average relative error between the real and the approximated fitness values was computed. This error was logged and plotted as a function of the number of generations in Figure 2.

This figure confirms that the quality of approximations given by the surrogate models increases with the number of generations. The fact that an EA progressively focuses on a certain area of the search domain, is clearly seen from the first two inserts in Figure 2. These inserts are a kind of "snapshots" taken at the respective generations, depicting the points in the design space already visited by the EA and stored in a DB along with the current population. For some outlying individuals, however, the approximate fitness value may not be quite accurate (see, for instance, generation 32, Figure 2(c)). The importance of outlying individuals in escaping from local minima, combined to the risk of being poorly approximated by the surrogate models, due to their sparse neighbourhood, makes their detection an important issue, addressed in oncoming sections.

The case is quite different in MOO problems. Since a set of points is sought as Pareto front solutions, throughout evolution the population in a well performing EA is spread over a relatively wide region of the search domain. This may cause trouble to any metamodel in yielding good approximations, especially when a new individual does not lie in the close vicinity of its neighbours. As an illustrative example, the simple problem of minimizing two offset quadratic functions has been worked out. The two objectives were formulated as

$$\begin{cases} f_1 = \sum_{i=1}^{N} x_i^2 \\ f_2 = \sum_{i=1}^{N} (x_i - x_{c,i})^2 \end{cases} \tag{2}$$

where $N = 2$ and the second paraboloid was centered at $\mathbf{x_c} = [2, 2]^T$. The search domain was the $[-5, 5] \times [-5, 5]$, while for the population $\mu = \lambda = 80$.

The average over the entire population relative errors between surrogate models' approximation and real values for both objectives are logged in Figure 3. From the inserts 3(a) and 3(c), it can be seen that the population remains spread over a relatively extended region of the search domain. This calls for a better generalization capability of the surrogate model, compared to what is needed in SOO, which occasionally may not be feasible (generation 8 in Figure 3). The poorly approximated individuals in insert 3(b), though not outlying, were at the margin of the available information in the DB. As

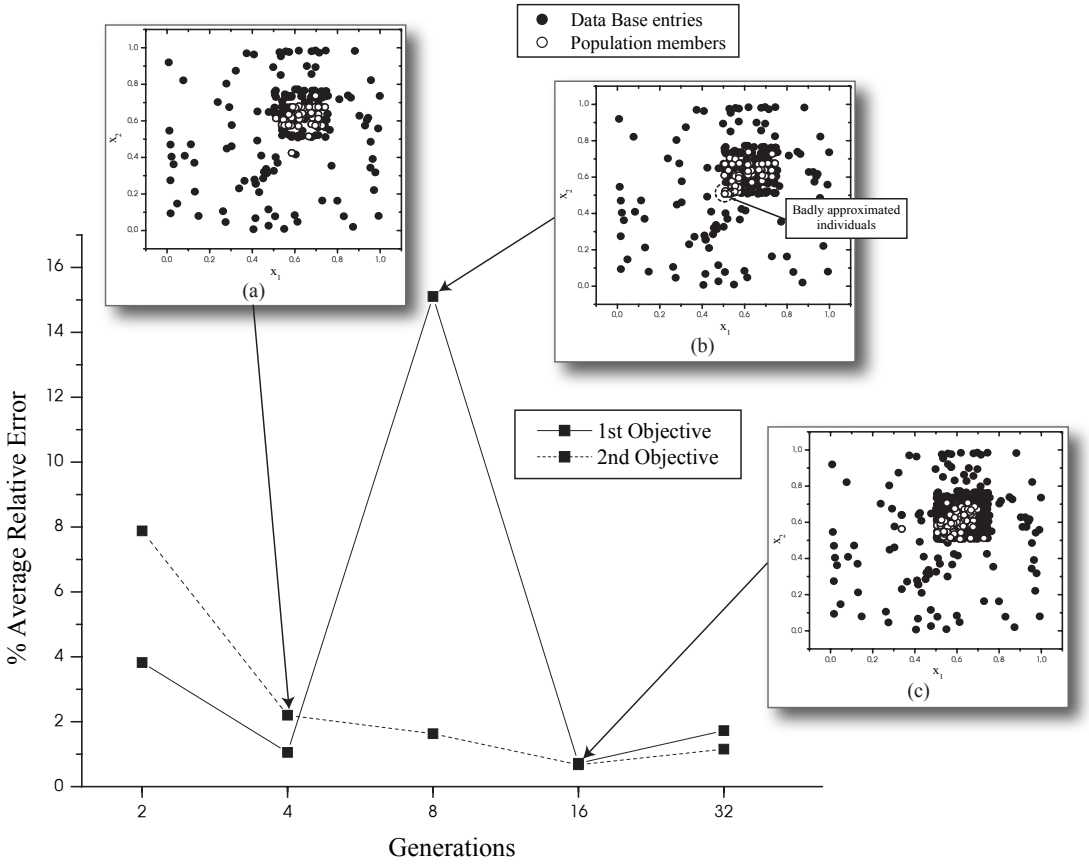


Figure 3: Multi-Objective: Predictive capability of surrogate models in course of evolution.

a consequence, the average approximation error is higher in this simple MOO example than in the case of minimizing the much more complex Rastrigin function. The effect of badly pre-evaluated individuals on convergence is further pronounced through the use of dominance criteria to designate the fitness of each individual. Since comparisons with respect to all the objectives are required, even a single poorly estimated objective may mislead the selection process.

4 SELECTION OF TRAINING PATTERNS AND DETECTION OF OUTLYING INDIVIDUALS

The selection of patterns for training each surrogate model has an important impact on their performance. Retrieving too few individuals from the DB does not entirely profit of the information gathered by the EA. On the other hand, taking into consideration individuals not lying in the vicinity of the new one can spoil the training of the surrogate model. In addition, as it has been demonstrated through the previous examples, the fitnesses of outlying individuals are most often poorly approximated, due to their sparse

neighbourhood. The detection of individuals, which are not located close to the available DB entries, could be a useful notification that one should not rely on the prediction of the surrogate model but should perform an exact evaluation instead. Below, these two issues have been addressed in a somehow unified manner.

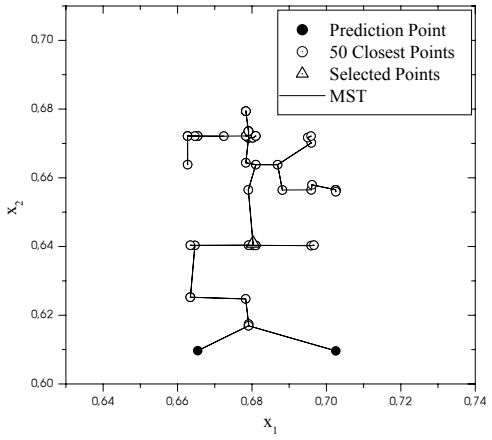
The selection of training patterns is based on the calculation of distances in the design variables' space. For each new individual, for which a surrogate model is to be built, its $100-150$ closest neighbours in the design space are retrieved from the DB. This number of neighbours is rather indicative and can be smaller at early generations when the DB contains few data. From this neighbouring set we intend to keep only the patterns that form a compact neighbourhood, discarding those lying too far from the rest. For this purpose it would be desirable to "connect" appropriately the patterns and check the length of the connections so as to decide whether to keep or not each one of them. A connection of a set of points with the important property of minimum total length of the connecting line is the Minimum Spanning Tree (MST) [10, 11, 12]. The MST is not unique, but its length *is* minimum.

The main idea is to create the MST of all the gathered points, the current individual included, and start "walking" along its branches from the new individual. Each tree's node — i.e. an exactly evaluated individual, retrieved from the DB— that is reached is added to the selected ones and the average length of the branches connecting the selected nodes is updated. The standard deviation of these lengths is also calculated. To traverse a branch and reach its end node, its length should be less than the current average length plus a tolerance based on the standard deviation. The algorithm terminates either when all the individuals retrieved from the DB have been selected, or when there is no adjacent branch to traverse with length satisfying the aforementioned criterion.

An application of this algorithm is illustrated in Figure 4. The new individuals ("prediction points") have been selected from the population at the 10^{th} generation of the EA solving the problem of the two paraboloids (Equation 2).

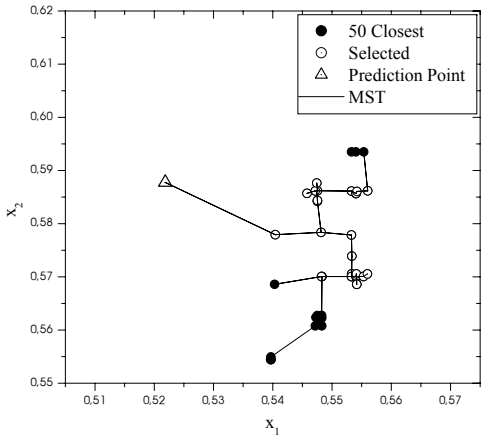
Having selected the patterns that will be used for training, the decision whether the new individual is outlying or not is quite straightforward. It depends on whether its distance from the k closest training pattern (usually $k = 2$) is less than the average length of branches connecting the selected patterns plus m ($m \simeq 3$) times the standard deviation of these lengths. Both the average length and the standard deviation have been computed during the selection process.

In the same test case as before, at the 10^{th} generation, the individuals detected as outlying, along with the respective relative errors of the guessed values of the objective functions, are tabulated below:



Error in f_1: 0.267%
Error in f_2: 0.662%

(a) The new individual is adequately close to DB entries.



Error in f_1: 55.6%
Error in f_2: 5.43%

(b) The new individual is too far from the DB entries.

Figure 4: Selection of training patterns based on their Minimum Spanning Tree.

Individual ID	15	23	33	59
Error in f_1	55.6%	23.5%	96.6%	23.6%
Error in f_2	5.43%	92.4%	8.39%	9.42%

Average errors over the entire population (outlying individuals excluded):
Error in f_1: 0.76%
Error in f_2: 1.5%

It is obvious that for outlying individuals the surrogate models fail to give reliable predictions for the objective functions. The distinction between an individual that is close to previously visited points in the design space and another one that lies in an unexplored region can be clearly seen in Figures 4(a) and 4(b).

5 THE RBFNs AS SURROGATE MODELS

We recall that the role of the surrogate model is to reconstruct the real function that maps the design variables to the space of objectives. The approach of locally reconstructing this mapping, even though it has to be repeated for each new member of the population, seems to give better approximations than trying to build a global metamodel. In addition, RBFNs incorporating ingredients from multivariate interpolation theory and regularization theory enjoy many desirable properties, the most important being that of *universal approximation*.

In MOO, the points in design space that constitute the Pareto front are spread over a quite extended region. This makes inadequate the use of an RBFN that only interpolates

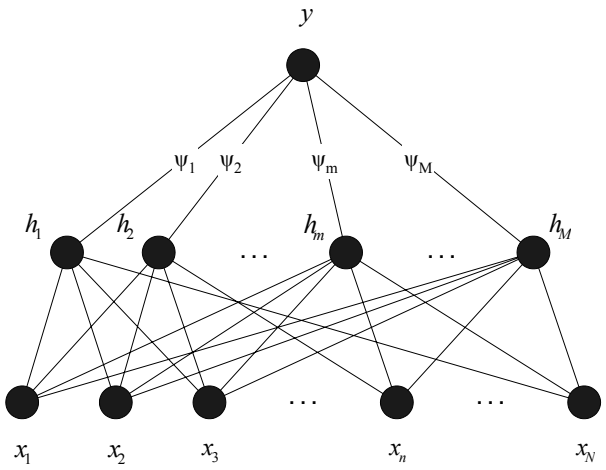


Figure 5: Radial-Basis Function network.

the training patterns. Such a network "learns" perfectly the details of the input-output mapping in the close vicinity of the training patterns, but usually fails to generalize, i.e. it generally estimates poorly the objective functions for a new individual that is located in the intermediate space between the training patterns. To overcome this difficulty the centers of the RBFN must be less than the training patterns, so as the latter to be approximated instead of interpolated. The details of such a formulation are discussed in Section 5.2, after presenting, in brief, a typical RBFN in the next section.

5.1 A typical RBFN

An RBFN for single-output approximations is a three-layer, fully-connected, feedforward network [13, 14] (Figure 5), which performs a nonlinear mapping $\mathrm{H} : \mathbb{R}^N \to \mathbb{R}^M$ from the N input units to a layer of M hidden nodes, followed by a linear one $\Psi : \mathbb{R}^M \to \mathbb{R}$ to the output unit $y \in \mathbb{R}$

$$y = \Psi \mathrm{H}\, \mathbf{x} \tag{3}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the input, i.e. the vector of design variables.

Each hidden node is associated with a point $\mathbf{c}_m$, $m = 1, M$ in the design variables' space $\mathbb{R}^N$, which is referred to as the RBF center. The non-linear mapping H is performed by the application of an activation function G_m, $m = 1, M$ — the radial-basis function — to the deviation of the input $\mathbf{x} \in \mathbb{R}^N$ from the corresponding center on every hidden node. Hence, the responce of each hidden node is

$$h_m(\mathbf{x}) = G_m(\|\mathbf{x} - \mathbf{c}_m\|) = G(\|\mathbf{x} - \mathbf{c}_m\|, r_m) \tag{4}$$

A typical activation function with translational and rotational invariance is the Gaussian

$G(u,r) = \exp(-u^2/r^2)$. The network's response is finally given as

$$y = y(\mathbf{x}) = \sum_{m=1}^{M} \psi_m h_m(\mathbf{x}) = \sum_{m=1}^{M} \psi_m G_m(\|\mathbf{x} - \mathbf{c}_m\|) \tag{5}$$

Suppose that the center and the radius of each activation function have been appropriately set (an issue discussed in the following section), then the only adjustable parameters are the M synaptic weights $[\psi_1, \psi_2, \ldots, \psi_M]^T = \boldsymbol{\psi}$. To do so the criterion is to best fit the network's responses to the available T training patterns $\{\hat{\mathbf{x}}_t, \hat{y}_t\}$, $t = 1, T$. This is achieved by minimizing the error

$$\min_{\boldsymbol{\psi}} E(\psi) = \min_{\psi_m,\, m=1,M} \frac{1}{2} \sum_{t=1}^{T} \Big(\hat{y}_t - \sum_{m=1}^{M} \psi_m G_m(\|\hat{\mathbf{x}}_t - \mathbf{c}_m\|)\Big)^2 \tag{6}$$

The solution $\boldsymbol{\psi}^*$ to this linear least-squares problem is given by the normal equations

$$\mathbf{G}^T\mathbf{G}\,\boldsymbol{\psi}^* = \mathbf{G}^T\hat{\mathbf{y}} \tag{7}$$

where

$$\mathbf{G} = \begin{bmatrix} G_1(\|\mathbf{x}_1 - \mathbf{c}_1\|) & G_2(\|\mathbf{x}_1 - \mathbf{c}_2\|) & \ldots & G_M(\|\mathbf{x}_1 - \mathbf{c}_M\|) \\ G_1(\|\mathbf{x}_2 - \mathbf{c}_1\|) & G_2(\|\mathbf{x}_2 - \mathbf{c}_2\|) & \ldots & G_M(\|\mathbf{x}_2 - \mathbf{c}_M\|) \\ \vdots & \vdots & & \vdots \\ G_1(\|\mathbf{x}_T - \mathbf{c}_1\|) & G_2(\|\mathbf{x}_T - \mathbf{c}_2\|) & \ldots & G_M(\|\mathbf{x}_T - \mathbf{c}_M\|) \end{bmatrix} \tag{8}$$

and $\hat{\mathbf{y}} = [\hat{y}_1, \ldots, \hat{y}_t]^T$.

In the simpler case where centers and training patterns coincide, the response surface of the network interpolates the training patterns and the synaptic weights $\boldsymbol{\psi}^*$ are given by

$$\mathbf{G}\,\boldsymbol{\psi}^* = \hat{\mathbf{y}} \tag{9}$$

5.2 Growing RBFNs with Self-Organizing Selection of Centers

When the number of available training patterns is high enough and good generalization is required from the network, the number of centers should be less than the training patterns. If the training patterns were chosen as centers, the resulting network might adapt itself to all the peculiarities of the vicinity of training patterns, it would fail, though, to capture the general flavour of the modeled input-output mapping.

To tackle the need for good generalization, especially in the context of IPE, where no interaction with the user is allowed, the approach of growing RBFNs, [15], has been adopted. An RBFN is created initiallly with very few centers. The error in the approximating training patterns is monitored and the center responsible for the higher errors is split. This process is repeated, hence the term "growing", until no improvement in the network's performance is achieved. The selection of centers is done in an unsupervised

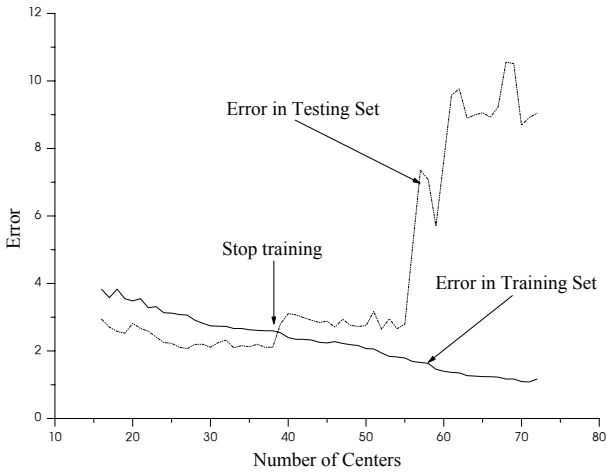


Figure 6: Variation of error in training and testing set as the number of RBFN centers increases.

manner with Learning Vector Quantization (LVQ), which aims at quantizing the training set with the centers of the RBFN. This can be done either by Self-Organizing Maps (SOM) [16, 13], herein adopted, or by Fuzzy Algorithms for LVQ [17]. After the network's centers have been selected, the widths (radii) of the radial-basis functions are computed using heuristics.

It is important to note that, in order to achieve good generalization of the trained network, it is necessary to randomly divide the training patterns into two sets, one used for training (Equation 6), and the other for testing. A characteristic variation of the training and testing error with the number of hidden nodes is given in Figure 6.The algorithmic framework just described is more explicitly given below:

Algorithm A (Growing SOM-RBFN) Form an RBFN with Self-Organizing selection of centers. Start with a few centers and keep growing until no improvement in predictions is achieved or a maximum number of hidden nodes is exceeded.

A1. [Initialization] Prescribe the minimum, M_{min}, and maximum, M_{max}, number of centers and set $M \leftarrow M_{min}$.

A2. [Unsupervised Learning Level] Determine the centers $\mathbf{c}_m$, $m = 1, M$ with an SOM process (see Algorithm B).

A3. [Heuristics] Compute the radii r_m, $m = 1, M$ (see Algorithm C).

A4. [Supervised Learning Level] Compute the synaptic weights ψ_m, $m = 1, M$ by minimizing the error on the training set (Equation 6).

A5. [Split centers or stop] If there is still improvement on the network's predictions for the testing set and $M < M_{max}$, split one of the centers ($M \leftarrow M + 1$) and go to

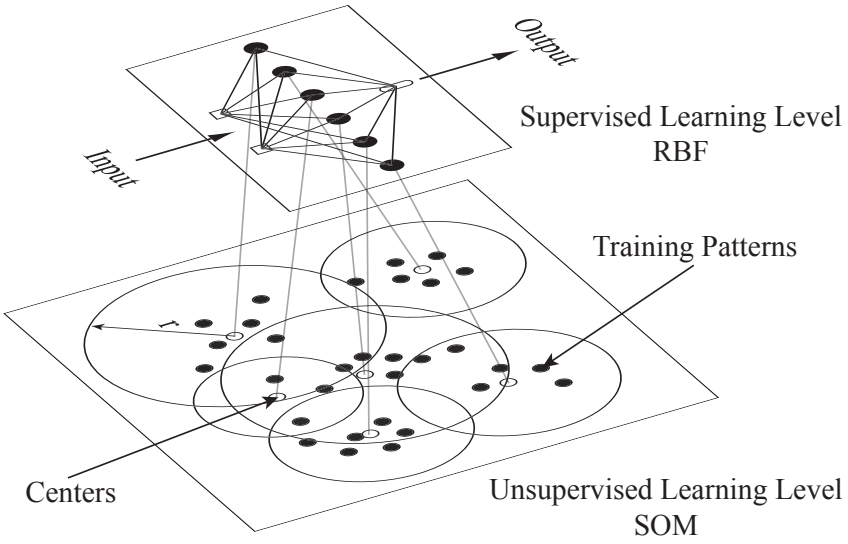


Figure 7: A two-level SOM-RBF network.

A2. Otherwise, terminate.

The algorithm just described consists mainly of two levels: that of unsupervised learning, whose outcome are the centers and the corresponding radii, and that of supervised learning, which computes the synaptic weights. This is schematically depicted in Figure 7. In the next paragraphs some important aspects of this algorithm are described in more detail.

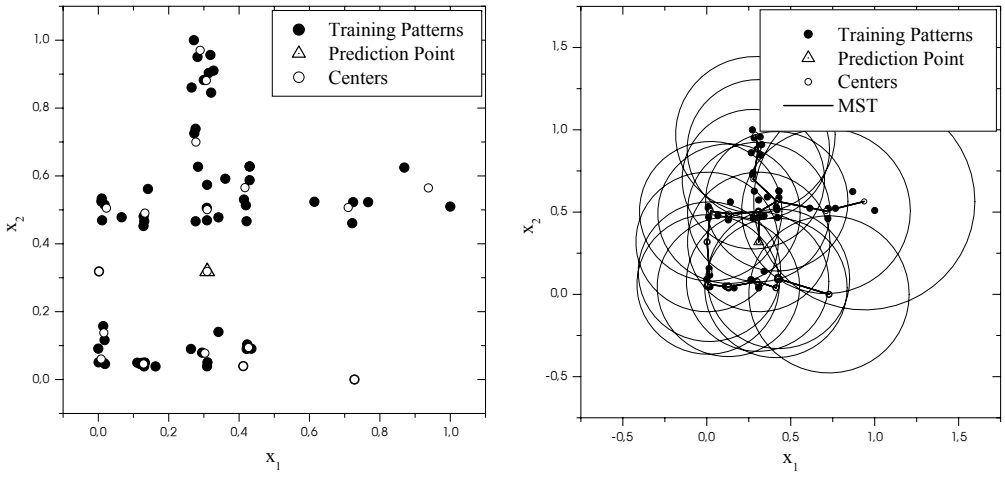
Determining the centers The centers of the overlying RBFN are determined as the feature map of all the training patterns (from both the training and testing set) and of the point at which a prediction is required — i.e. in the context of IPE the new individual under consideration (Figure 8(a)). For this purpose the standard algorithm for SOMs proposed by Kohonen [16] is used.

Algorithm B (Self-Organizing Map) The standard SOM is adapted to the framework of Algorithm A. It is assumed that the initial centers $\mathbf{c}_m^{(0)}$ are either randomly set or come from a previous iteration of Algorithm A.

B1. [Similarity matching] Draw a training pattern $\hat{\mathbf{x}}_t$ and locate the closest center $\mathbf{c}_i^{(n)}$ at the current iteration n:

$$i(\hat{\mathbf{x}}_t) = \arg\min_m \|\hat{\mathbf{x}}_t - \mathbf{c}_m^{(n)}\|, \quad m = 1, M$$

B2. [Update centers] Update the centers, by taking into account the learning rate parameter $\eta(n)$ and the neighbourhood parameter $h_{m,i}(n)$ centered at the "winning"



(a) Centers (b) Widths

Figure 8: Self-organizing selection of the centers and the widths of the radial-basis functions.

center i:

$$\mathbf{c}_m^{(n+1)} = \mathbf{c}_m^{(n)} + \eta(n) h_{m,i}(n)(\hat{\mathbf{x}}_t - \mathbf{c}_m^{(n)})$$

Both $\eta(n)$ and $h_{m,i}(n)$ should be monotonically decreasing with n to achieve convergence.

B3. [Continuation] If no noticeable change in the locations of centers occurs, terminate. Otherwise, increase n ($n \leftarrow n+1$) and go to **B1**.

Computing the radii The widths of the radial-basis functions are computed after the centers, through the following heuristic algorithm:

Algorithm C Determine the widths of radial-basis functions.

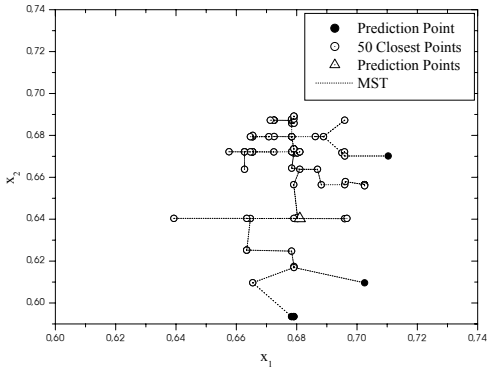
C1. [MST] Construct the MST of centers.

C2. [Neighbours of each center] For the center $\mathbf{c}_m$ collect all the centers that can be reached, if a maximum of n_b branches of the tree are traversed starting from $\mathbf{c}_m$. Denote by $\mathcal{C}_m$ their set.

C3. [Set width] For the radial-basis function G_m set

$$r_m = \frac{1}{|\mathcal{C}_m|} \sum_{i:\mathbf{c}_i \in \mathcal{C}_m} \|\mathbf{c}_m - \mathbf{c}_i\|$$

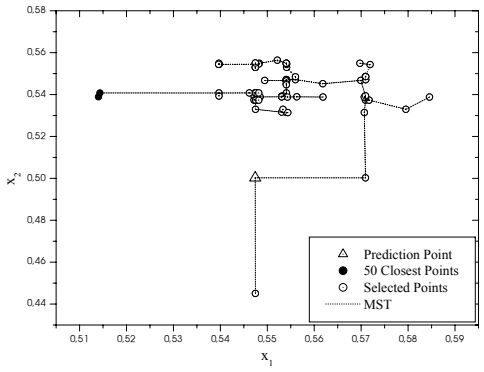
C4. [Continuation] Repeat **C2** – **C3** for $m = 1, M$.



Interpolation	
Error in f_1:	0.0109%
Error in f_2:	0.0882%

Approximation	
Error in f_1:	0.887%
Error in f_2:	3.94%

(a) The new individual lies close to DB entries.



Interpolation	
Error in f_1:	74.7%
Error in f_2:	0.371%

Approximation	
Error in f_1:	1.44%
Error in f_2:	0.0606%

(b) The new individual lies far from DB entries.

Figure 9: Interpolate or approximate the training patterns? It depends on the topology of the neighbourhood of each individual.

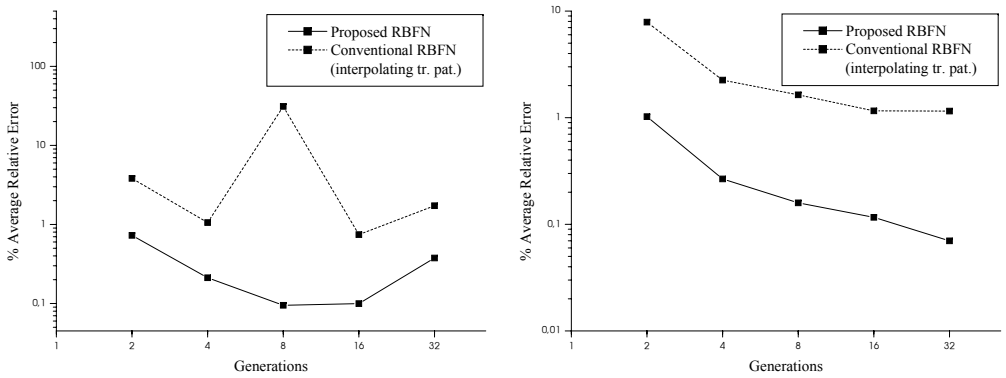
A value for $n_b \simeq 4$ gives usually satisfactory coverage of the training patterns, however, care must be taken so as a radius not to become too small. The radii derived from this algorithm are depicted in Figure 8(b).

Splitting a center The center to be split at each iteration of Algorithm A, similar to what is proposed in [15], is the one that is most responsible for the greatest errors in training patterns. To implement this, the error between the network's predictions and the real values is computed for every training pattern from both the training and testing sets. The s% of individuals with the greatest errors are picked out. A reasonable value of s could be 30%. The center that is closest to most of these patterns is the one to be split.

5.3 Which RBFN to use?

The RBFN that interpolates the training patterns "learns" very well their close neighbourhood. Such a network is preferable, and, of course, faster to train, if a new individual appears very close to the DB entries. This is clearly seen from Figure 9(a), where the conventional RBFN gives better estimations of the objective functions than the hybrid SOM-RBFN.

On the contrary, when a new individual lies far from the DB entries, the property of good generalization is indispensable. As illustrated in Figure 9(b), the difference between the two types of RBFN is enormous. It is interesting to note that the hybrid SOM-RBFN



(a) Errors in 1st Objective (b) Errors in 2nd Objective

Figure 10: A comparison between the performance of conventional RBFNs and the one proposed herein; the latter automatically switches between interpolation and approximation of the training patterns, depending on the topology of the neighbourhood of each new individual.

in the previous case does not perform as badly as the conventional RBF in this case.

Therefore, in the algorithm implemented for use in the context of IPE, the decision, which type of RBFN to use, depends on the position of the new individual with respect to its closest neighbours, as deduced from the construction of an appropriate MST.

6 APPLICATIONS

The two test cases presented below, one numerical and one from the turbomachinery field, aim at demonstrating the predictive capability of the proposed RBFN. The context of IPE is simulated by taking "snapshots" of the EA at prescribed generations. Each snapshot contains the DB entries at this particular generation and the corresponding population. Using this information only, it is attempted to predict the objective functions of the individuals, through the use of the proposed metamodel.

6.1 Numerical case

The numerical case consists in the minimization of the two quadratic functions already defined in Section 3, Equation 2.

In Figure 3, a conventional RBFN, which interpolates the training patters [4], has been used. These errors are compared to the ones obtained with the herein proposed metamodel, Figure 10. The improvement in prediction capability is apparent; at all the sampled generations the average relative error is less than 1%.

The effect of the dimension of the design space on the prediction capability of the network is investigated in Table 1. All the errors have been measured at the 10^{th} generation of an EA with $\mu = \lambda = 80$. Since, at a given generation, the number of DB entries is more or less the same, as the dimension of the design space increases, the error in predictions also increases. However, the proposed RBFN constantly outperforms the conventional

Conventional RBFN							
Dimension		2	4	8	16	32	64
Average Error	1^{st} Objective	0.74%	3.4%	5.9%	3.6%	3.0%	4.3%
	2^{nd} Objective	2.1%	8.1%	3.3%	5.2%	4.7%	3.5%

Proposed RBFN							
Dimension		2	4	8	16	32	64
Average Error	1^{st} Objective	0.22%	0.79%	2.4%	2.3%	2.5%	2.8%
	2^{nd} Objective	0.67%	1.5%	1.3%	3.4%	3.9%	2.4%

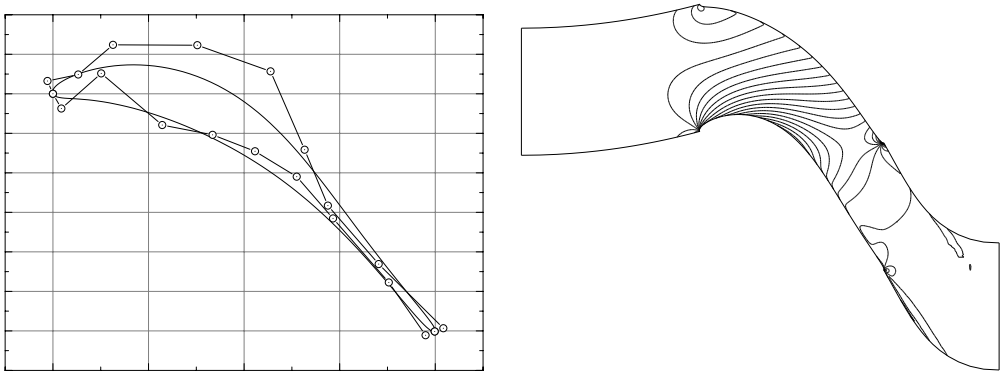
Table 1: The effect of the dimension of the design space on the quality of surrogate models' predictions.

one, keeping the errors reasonably small.

6.2 Design of a Turbine Cascade

This problem is dealing with the design optimization of a 2D turbine airfoil with maximum flow turning and minimum losses, operating at $\alpha_1 = 19.4^o$ and $M_{is} = 0.7$.

The airfoil was parameterized with Bézier curves (Figure 11(a)). As design parameters the ordinates of 3 control points in the middle of pressure side and of 3 control points in the middle of suction side have been used. Concerning the population size both μ and λ were set equal to 50.



(a) Blade airfoil and Bézier control points (b) Iso-Mach lines

Figure 11: Blade airfoil optimization; maximize flow turning with minimum losses.

The prediction errors of the herein proposed RBFN are monitored in Table 2 for different generations and for both objectives. It is clear that these errors are extremely low and this is in favour of the RBFN-based IPE technique, incorporated in the evolutionary optimization.

Generation		5	10	20
Average Error	1^{st} Objective	0.010%	0.12%	0.13%
	2^{nd} Objective	0.0014%	0.018%	0.017%

Table 2: Blade airfoil design: Average errors of the proposed RBFN.

7 CONCLUSIONS

In this paper, the problem of efficiently employing metamodel-assisted inexact pre-evaluation techniques in Evolutionary Algorithms was scrutinized. Weaknesses of handling Multi-Objective optimization as a straightforward extention of the metamodel-assisted Single-Objective optimization are reported and remedies are proposed. The latter concern a particular surrogate evaluation model, namely the Radial-Basis Function networks, but the messages can readily be generalized. The major conclusion of the paper in hands is that extra attention should be paid in Multi-Objective optimization concerning the construction and training of the metamodel used.

Acknowledgements

This research was sponsored by Dassault Aviation with Prof. J. Périaux, Pôle Scientifique Dassault Aviation/UPMC, as Project Monitor.

The first author was supported by an A. S. Onassis Public Benefit Foundation Scholarship.

REFERENCES

[1] Hajela Prabhat. Soft computing in multidisciplinary aerospace design—new directions for research. *Progress in Aerospace Sciences*, 38:1–21, 2002.

[2] A.P. Giotis, K.C. Giannakoglou, and J. Périaux. A reduced cost multi-objective optimization method based on the pareto front technique, neural networks and PVM. In *ECCOMAS 2000*, Barcelona, Spain, 2000.

[3] M. Emmerich, A. Giotis, M. Ozdemir, T. Bäck, and K.C. Giannakoglou. Metamodel-assisted evolution strategies. In *PPSN VII*, Granada, Spain, 2002.

[4] K.C. Giannakoglou, A.P. Giotis, and M.K. Karakasis. Low–cost genetic optimization based on inexact pre–evaluations and the sensitivity analysis of design parameters. *Journal of Inverse Problems in Engineering*, 9:389–412, 2001.

[5] K.C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38:43–76, 2002.

[6] M.K. Karakasis, A.P. Giotis, and K.C. Giannakoglou. Inexact information aided, low-cost, distributed genetic algorithms for aerodynamic shape optimization. *International Journal for Numerical Methods in Fluids*, 43:1149–1166, 2003.

[7] K. Deb, S. Agarwal, A. Pratap, and T. Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. Technical Report 200001, Indian Institute of Technology Kanpur, 2000.

[8] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2 improving the Stength Pareto evolutionary algorithm. TIK-Report No. 103, Computer Engineering and Communication Networks Lab, ETH Zurich, 2001.

[9] C.A. Coello Coello. An updated survey of evolutionary multiobjective optimization techniques: State of the art and future trends. Progress on Evolutionary Computation, 1999.

[10] T. Cormen, C. Leiserson, R. Rivest, and Stein C. *Introduction to Algorithms*. The MIT Press, Cambridge MA, 2nd edition, 2002.

[11] Robert Sedgewick. *Algorithms*. Addison-Wesley Pub. Co., 2nd edition, 1988.

[12] Donald Knuth. *Fundamental Algorithms*, volume 1 of *The Art of Computer Programming*. Addison-Wesley, 3rd edition, 1997.

[13] Simon Haykin. *Neural Networks, A Comprehensive Foundation*. Prentice Hall, 2nd edition, 1999.

[14] T. Poggio and F. Girosi. Networks for approximation and learning. *Proceedings of the IEEE*, 78(9):1481–1497, 1990.

[15] N. Karayiannis and W. M. Glenn. Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques. *IEEE Transactions on Neural Networks*, 8(6):1492–1506, November 1997.

[16] Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, 2nd edition, 1997.

[17] N. Karayiannis. A methodology for constructing fuzzy algorithms for learning vector quantization. *IEEE Transactions on Neural Networks*, 8(3):505–518, May 1997.