Noname manuscript No.
(will be inserted by the editor)

A Comprehensive Survey of Fitness Approximation in Evolutionary Computation

Yaochu Jin

Honda Research Institute Europe
63073 Offenbach/Main, Germany
Email: yaochu.jin@honda-ri.de

The date of receipt and acceptance will be inserted by the editor

Abstract Evolutionary algorithms (EAs) have received increasing interests both in the academy and industry. One main difficulty in applying EAs to real-world applications is that EAs usually need a large number of fitness evaluations before a satisfying result can be obtained. However, fitness evaluations are not always straightforward in many real-world applications. Either an explicit fitness function does not exist, or the evaluation of the fitness is computationally very expensive. In both cases, it is necessary to estimate the fitness function by constructing an approximate model. In this paper, a comprehensive survey of the research on fitness approximation in evolutionary computation is presented. Main issues like approximation levels, approximate model management schemes, model construction techniques are reviewed. To conclude, open questions and interesting issues in the field are discussed.
Keywords: evolutionary computation, fitness approximation, meta-model, optimization

1 Introduction

Evolutionary computation has found a wide range of applications in various fields of science and engineering. Among others, evolutionary algorithms have been proved to be powerful global optimizers. Generally, evolutionary algorithms outperform conventional optimization algorithms for problems which are discontinuous, non-differential, multi-modal, noisy and not well-defined problems, such as art design, music composition and experimental designs [75]. Besides, evolutionary algorithms are also well suitable for multi-criteria problems.

Despite the great successes achieved in real-world applications, evolutionary algorithms have also encountered many challenges. For most evolutionary algorithms, a large number of fitness evaluations (performance calculations) are needed before a well acceptable solution can be found. In many real-world applications, fitness evaluation is not trivial. There are several situations in which fitness evaluation becomes difficult and computationally efficient approximations of the fitness function have to be adopted.

Several issues need to be addressed in employing fitness approximations in evolutionary computation. First, which levels of the fitness approximation should be used. While an experimental verification can be seen as the true fitness value of a given solution, fully computational simulations, simplified computational simulations as well as functional approximations (meta-models) can be used for fitness calculation. So far, several models have been used for fitness approximation. The most popular ones are polynomials (often known as response surface methodology), the kriging model, most popular in design and analysis of computer experiments (DACE), the feedforward neural networks, including multi-layer perceptrons and radial-basis-function networks and the support vector machines. Due to the lack of data and the high dimensionality of input space, it is very difficult to obtain a perfect global functional approximation of the original fitness function. To tackle this problem, two main measures can be taken. Firstly, the approximate model should be used together with the original fitness function. In most cases, the original fitness function is available, although it is computationally very expensive. Therefore, it is very important to use the original fitness function efficiently. This is known as model management in conventional optimization [17] or evolution control in evolutionary computation [36]. Secondly, the quality of the approximate model should be improved as much as possible given a limited number of data. Several aspects are important to improve the model quality, such as selection of the model, use of active data sampling and weighting (both on-line and off-line), selection of training method and selection of error measures.

The work on fitness approximation in evolutionary computation has been distributed in several different areas. Therefore, this survey aims to provide readers a comprehensive picture of fitness approximation in evolutionary computation. In Section 2, different motivations for using approximation in evolutionary computation are presented. In the following section, three major levels of approximation, namely, problem approximation, functional approximation and *evolutionary* approximation are presented. Different approaches to the incorporation of fitness approximations are described in Section 4. The main functional approximation models that have been used in fitness approximation are introduced in Section 5. Data sampling techniques, which are very important for the quality of models, are given in Section 6. Finally, open questions and promising research topics are discussed in Section 7.

2 Motivations

So far, approximation of the fitness function in evolutionary computation has been applied mainly in the following cases.

- The computation of the fitness is extremely time-consuming. One good example is structural design optimization [30, 43, 44, 51, 77, 59, 37, 55]. In aerodynamic design optimization, it is often necessary to carry out computational fluid dynamics (CFD) simulations to evaluate the performance of a given structure. A CFD simulation is usually computationally expensive, especially if the simulation is 3-dimensional, which takes over ten hours on a high-performance computer for one calculation. Therefore, approximate models have widely been used in structure optimization [4].
  Fitness approximation has also been reported in protein structure prediction using evolutionary algorithms [52, 58]. A neural network has been used for feature extraction from amino acid sequence in evolutionary protein design and drug design [73, 72, 71].
- There is no explicit model for fitness computation. In many situations, such as in art design and music composition as well as in some areas of industrial design, the evaluation of the fitness depends on the human user. Generally, these problems can be addressed using interactive evolutionary computation [80]. However, a human user can easily get tired and an approximate model that embodies the opinions of the human evaluator is also very helpful [5, 40].
- The environment of the evolutionary algorithm is noisy. Usually, there are two methods to deal with noisy fitness functions. The first one is to sample the fitness several times and to average [25]. However, this method requires a large number of additional fitness evaluations. The second method is to calculate the fitness of an individual by averaging the value of this individual as well as that of other individuals in its neighborhood. To avoid additional computational cost, the individuals that participate in the averaging can be chosen from the current and previous generations [7]. A more flexible alternative is to estimate the fitness of the individuals in the neighborhood using a statistical model constructed with history data [69, 8].
- The fitness landscape is multi-modal. The basic assumption is that a global model can be constructed to approximate and smoothen out the local optima of the original multi-modal fitness function without changing the global optimum and its location. A Gaussian kernel has been used to realize coarse-to-fine smoothing of the original multi-modal function [84]. Approximation for smoothing multi-modal functions has also been reported in [45, 46], where global polynomial models are used instead of Gaussian kernel functions. Note that it is generally difficult to build an approximate model that has the same global optimum on the same location. Therefore, the coarse-to-fine modeling approach seems to be more realistic.

3 Levels of Approximation

The concept of approximation in optimization is not new [4]. Traditionally, there are two basic approaches, i.e., functional approximation and problem approximation. Additionally, special approximation techniques for evolutionary fitness evaluation have also been suggested.

- Problem approximation. Problem approximation tries to replace the original statement of the problem by one which is approximately the same to the original problem but which is easier to solve. For example, to evaluate the performance of a turbine blade, wind tunnel experiments need to be carried out. However, computational fluid dynamics (CFD) simulations instead of wind tunnel experiments are often used to evaluate the performance of a blade during design. In CFD simulations, the fluid dynamics are described with three-dimensional (3D) Navier-Stokes equations with a turbulence model [2]. If the viscosity, mass diffusion and thermal conductivity of the flow are neglected, then flow dynamics can be described with three-dimensional Euler equations. The Euler equations are computationally more efficient to solve than the Navier-Stokes equations. Under certain conditions, the 3D flow field can be solved by 2D computations, which is known as quasi-3D solvers and computationally more efficient [6].
  Of course, many *ad hoc* methods can also be developed. For example, in [29], random sampling instead of complete sampling is used for solving image registration problems using genetic algorithms. Another example is the work reported in [1], where fitness approximation is studied in terms of discretization.
- Functional approximation. In functional approximation, an alternate and explicit expression is constructed for the objective function (in evolutionary computation, it is usually called fitness function). Take the blade design again as an example, instead of evaluating its performance using CFD simulations, an explicit mathematical model can be constructed whose inputs and outs are the blade geometry and the blade performance, respectively.
- *Evolutionary approximation.* This type of approximation is specific for evolutionary algorithms. A popular class of the evolutionary approximation methods is known as fitness inheritance [79, 85, 70, 15]. In these methods, fitness evaluations can be spared by estimating the fitness value of the offspring individuals from the fitness value of their parents.
  In the second class of fitness approximation, the individuals are clustered into several groups [42]. Then, only the individual that represents its cluster will be evaluated using the fitness function. The fitness value of other individuals in the same cluster will be estimated from the representative individual based a distance measure. We term it *fitness imitation* in contrast to fitness inheritance.

4 Incorporation of Approximate Fitness Models

4.1 Incorporation Mechanisms

The incorporation of approximate models constructed using history data in evolutionary computation can be seen one of the methods for incorporating knowledge into evolutionary systems [35]. Very interestingly, approximate models have been embedded in almost every element of evolutionary algorithms, including migration, initialization, recombination, mutation, fitness evaluations.

- Use of approximate fitness evaluations through migration [82, 18, 76]. The island model based architecture has been proposed to incorporate information from approximate models to speed up the evolutionary algorithm [18], as illustrated in Fig. 1. It can be seen that one sub-population has been introduced for each level of approximation. Usually, each population evolves separately using its own level of approximation. At a certain frequency, the individuals in the sub-populations that uses higher accuracy approximations will be migrated into those using approximations of lower accuracy. An extension of this architecture has been suggested in [76], where individuals can migrate from sub-populations using approximations of higher accuracy to sub-populations using approximate models of lower accuracy and vise versa Fig. 2.
- Use of approximate fitness models for initializing the population and for guiding crossover and mutation [3, 62, 63, 74]. Using approximate models in population initialization and guidance of genetic operators rather than directly in fitness evaluations is believed

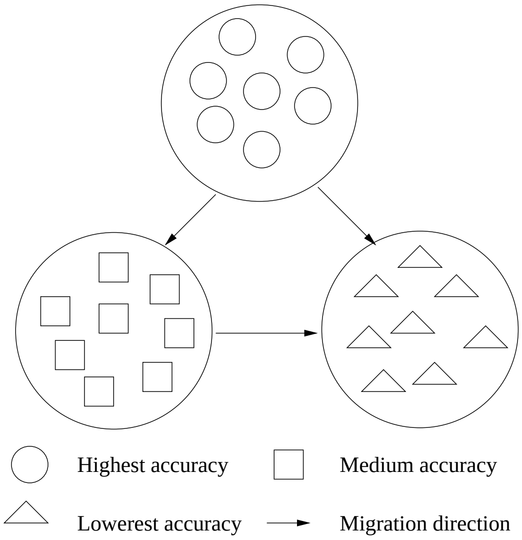

Fig. 1 Island model of parallel GA for incorporating approximate fitness models. In the figure, a large circle means a sub-population, small circles denote individuals using the fitness function with the highest accuracy, rectangles denote the individuals using models with the medium accuracy and the triangles are the individuals using models with the lowest accuracy.

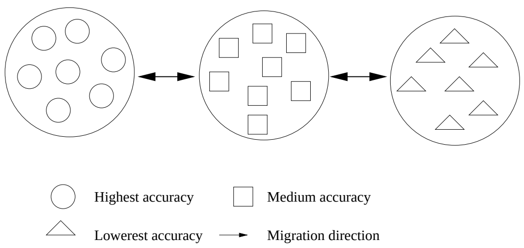

Fig. 2 Hierarchical GA for incorporating approximate fitness models.

to reduce the risk of misleading the search direction by the approximate models [62]. The reason for is that initialization, crossover and mutation are usually carried out randomly. Thus, initializations and genetic operations guided by an approximate model even with lower accuracy should usually better than do them randomly. However, the reduction of fitness evaluations may not be significant.

– Use of approximate fitness models through fitness evaluations. In most research, the approximate model has been directly used in fitness evaluations in order to reduce the number of fitness calculation [44, 56, 57, 48, 64, 65, 20, 21, 51, 12, 59, 23, 36, 37, 38, 27, 22]. Different approximate models, including polynomials, kriging models and neural networks have been applied. An interesting idea in [22] is that a confidence interval for the fitness estimation is calculated to modify the model prediction so that the search in unexplored regions is encouraged. It has been found that strategy leads to better performance, especially when the original fitness function is multimodal. Most recently, approximate fitness evaluations have also been employed in evolutionary multiobjective optimization [23, 24, 50, 53].

4.2 Model Management or Evolution Control

Among the three approaches to incorporating approximate models, the use of approximate models for fitness evaluations may reduce the number of fitness evaluations most significantly. However, the application of approximation models to evolutionary computation is not as straightforward as one may expect. There are two major concerns in using approximate models for the fitness evaluation. First, it should be ensured that the evolutionary algorithm converges to the global optimum or a near-optimum of the original fitness function. Second, the computational cost should be reduced as much as possible. One essential point is that it is very difficult to construct an approximate model that is globally correct due to the high dimensionality, ill distribution and limited number of training samples. It is found that if an approximate model is used for fitness evaluation, it is very likely that the evolutionary algorithm will converge to a false optimum. A false optimum is an optimum of the approximate model, which is not one of the original fitness function, refer to Fig.3 for an example.

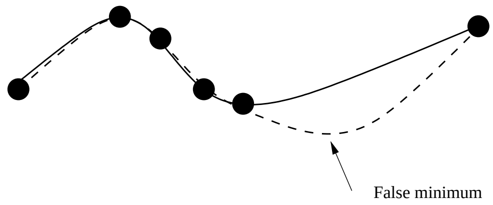

Fig. 3 An example of a false minimum in the approximate model. Solid line denotes the original fitness function, dashes line the approximate model and the dots the available samples.

Therefore, it is very essential in most cases that the approximate model should be used together with the original fitness function. This can be regarded as the issue of model management or evolution control. By evolution control, it is meant that in evolutionary computation using approximate models, the original fitness function is used to evaluate some of the individuals or all individuals in some generations [36]. An individual that is evaluated using the original fitness function is called a controlled individual. Similarly, a generation in which all its individual are evaluated using the original fitness function is called a controlled generation.

Generally, model management in evolutionary computation can be divided into three main approaches from the viewpoint of evolution control.

- No Evolution Control. Very often, the approximate model is assumed to be of high-fidelity and therefore, the original fitness function is not at all used in evolutionary computation, such as in [5, 66, 40].
- Fixed Evolution Control. The importance to use both the approximate model and the original function for fitness evaluation has been recognized [64]. There are generally two approaches to evolution control, one is individual-based [30, 12, 36], and the other is generation-based [64, 65, 36]. By individual-based control, it is meant that in each generation, some of the individuals use the approximate model for fitness evaluation and others the original function for fitness evaluation. In individual-based evolution control, either a random strategy or a best strategy can be used to select the individuals to be controlled [36]. In the best strategy, the best individual (based on the ranking evaluated by the approximate model) in the current generation is reevaluated using the original function [30], see Fig.4, while in the random strategy, the individuals to be controlled are selected randomly. It has been shown that the best strategy shows To reduce the computational cost further, individual-based evolution control can be carried out only in a selected number of generations [12]. In contrast, the random strategy selects certain number of individuals randomly for reevaluation using the original fitness function [36]. An alternative to the best strategy and the random strategy is to evaluate the mean of the individuals in the current population [51].
  Generation-based evolution control can also be implemented [64, 65]. In [64], generation-based evolution control is carried out when the evolutionary algorithm converges on the approximate model. More heuristically, evolution control is carried out once in a fixed number of generations, see Fig. 5.

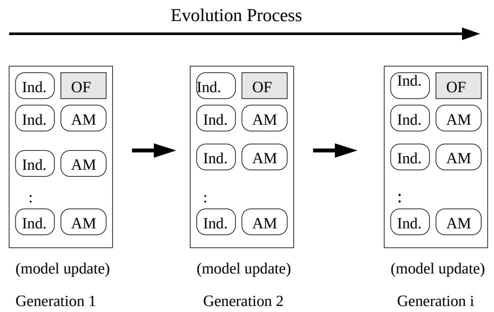

Fig. 4 The best individual is controlled in each generation. AM: approximate model; OF: original function.

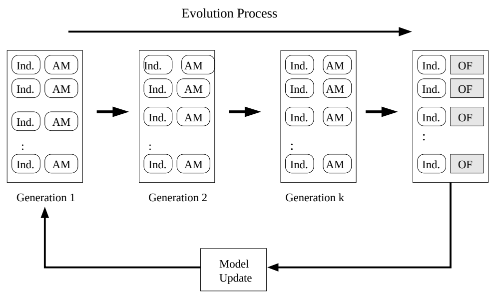

Fig. 5 Generation-based evolution control. AM: approximate model; OF: original function.

  One drawback in the aforementioned methods is that the frequency of evolution control is fixed. This is not very practical because the fidelity of the approximate model may vary significantly during optimization. In fact, a predefined evolution control frequency may cause strong oscillation during optimization due to large model errors, as observed in [64].
- Adaptive Evolution Control. It is straightforward to imagine that the frequency of evolution control should depend on the fidelity of the approximate model. A method to adjust the frequency of evolution control based on the trust region framework [17] has been suggested in [51], in which the generation-based approach is used. A framework for approximate model management has also been suggested in [37], which has successfully been applied to 2-dimensional aerodynamic design optimization, see Fig. 6.

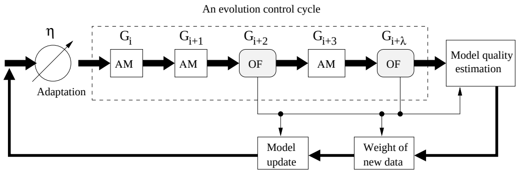

Fig. 6 Adaptive generation-based evolution control. In the evolution control cycle, there are λ generations, η ($\eta \leq \lambda$) generations will be controlled. AM: approximate model; OF: original function.

5 Approximation Models

5.1 Polynomial Models

The most widely used polynomial approximation model is the second-order model which has the following form:

$$\hat{y} = \beta_0 + \sum_{1 \leq i \leq n} \beta_i x_i + \sum_{1 \leq i \leq j \leq n} \beta_{n-1+i+j} x_i x_j, \qquad (1)$$

where β_0 and β_i are the coefficients to be estimated, and the number of terms in the quadratic model is $n_t = (n+1)(n+2)/2$ in total, where n is the number of input variables.

To estimate the unknown coefficients of the polynomial model, both least square method (LSM) and gradient method can be used:

- **Least Square Method** To get a unique estimation of the coefficients using LSM, it is required that the number of samples (N) drawn from the original function should be equal to or larger than the number of coefficients n_t. Let

$$\mathbf{y} = [y^{(1)}, y^{(2)}, \cdots, y^{(N)}]^T, \qquad (2)$$

and

$$\mathbf{X} = \begin{bmatrix} 1 & x_1^{(1)} & x_2^{(1)} & \cdots & (x_n^{(1)})^2 \\ 1 & x_1^{(2)} & x_2^{(2)} & \cdots & (x_n^{(2)})^2 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_1^{(N)} & x_2^{(N)} & \cdots & (x_n^{(N)})^2 \end{bmatrix}, \qquad (3)$$

then the following equation holds:

$$\mathbf{y} = \mathbf{X}\boldsymbol{\Theta}, \qquad (4)$$

The LSM algorithm works as follows,

$$\hat{\boldsymbol{\Theta}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}, \qquad (5)$$

where $\hat{\boldsymbol{\Theta}}$ denotes the estimate of $\boldsymbol{\Theta}$. One assumption here is that the rows of $\mathbf{X}$ are linearly independent.

- **Gradient Method** The main drawback of the least square method is that the computational expense becomes unacceptable as the dimensionality increases. To address this problem, the gradient method can be used. Define the following square error function for the k-th sample:

$$E^{(k)} = \frac{1}{2}(y - y^{(k)})^2, \qquad (6)$$

where y is defined in equation (1), it is then straightforward to get the update rule for the unknown coefficients:

$$\Delta\beta_0 = -\xi \cdot (y - y^{(k)}) \qquad (7)$$
$$\Delta\beta_i = -\xi \cdot (y - y^{(k)}) x_i^{(k)} \qquad (8)$$
$$\Delta\beta_{n-1+i+j} = -\xi \cdot (y - y^{(k)}) x_i^{(k)} x_j^{(k)}, \qquad (9)$$
$$1 \leq i \leq j \leq n.$$

5.2 Kriging Models

The kriging model can be seen as a combination of a global model plus a localized "deviation":

$$y(\mathbf{x}) = g(\mathbf{x}) + Z(\mathbf{x}), \qquad (10)$$

where $g(\mathbf{x})$ is a known function of $\mathbf{x}$ as a global model of the original function, and $Z(\mathbf{x})$ is a Gaussian random function with zero mean and non-zero covariance that represents a localized deviation from the global model. Usually, $g(\mathbf{x})$ is a polynomial and in many cases, it is reduced to a constant β.

The covariance of $Z(\mathbf{x})$ is expressed as

$$\mathrm{Cov}[Z(\mathbf{x}^{(j)}), Z(\mathbf{x}^{(k)})] = \sigma^2 \mathbf{R}[R(\mathbf{x}^{(j)}, \mathbf{x}^{(k)})], \qquad (11)$$
$$j, k = 1, \cdots, N,$$

where R is the correlation function between any two of the N samples, and $\mathbf{R}$ is the symmetric correlation matrix of dimension $N \times N$ with values of unity along the diagonal. The form of the correlation matrix can be selected by the user, and the following form has often been used [11, 28, 78]:

$$R(\mathbf{x}^{(j)}, \mathbf{x}^{(k)}) = \exp[-\sum_{i=1}^{n} \theta_i |x_i^{(j)} - x_i^{(k)}|^2], \qquad (12)$$

where θ_i are the unknown correlation parameters, $x_i^{(j)}$ and $x_i^{(k)}$ are the i-th component of sample points $\mathbf{x}^{(j)}$

and $\mathbf{x}^{(k)}$. Thus, the prediction of $y(\mathbf{x})$ is a function of unknown parameters β and $\theta_i, i = 1, 2, \cdots, n$:

$$\hat{y} = \hat{\beta} + \mathbf{r}^T(\mathbf{x})\mathbf{R}^{-1}(\mathbf{y} - \beta\mathbf{I}), \tag{13}$$

where, $\hat{y}$ is the estimated value of y given the N samples and the current input $\mathbf{x}$, $\hat{\beta}$ is the estimated value of β, $\mathbf{y}$ is a vector of length N as defined in Eqn. (2), $\mathbf{I}$ is a unit vector of length N, and $\mathbf{r}$ is the correlation vector of length N between the given input $\mathbf{x}$ and the samples $\{\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(N)}\}$:

$$\mathbf{r}^T(\mathbf{x}) = [R(\mathbf{x}, \mathbf{x}^{(1)}), R(\mathbf{x}, \mathbf{x}^{(2)}), \cdots, R(\mathbf{x}, \mathbf{x}^{(N)})]^T. \tag{14}$$

The estimation of the parameters can be carried out using the maximum likelihood method.

One advantage of using kriging models is that a confidence interval of the estimation can be obtained without much additional computational cost. Note, however, that it is necessary to perform matrix inversions for estimating the output in the kriging model, which increases the computational expense significantly when the dimensionality becomes high.

5.3 Neural Networks

Neural networks have shown to be effective tools for function approximation. Both feedforward multilayer perceptrons and radial-basis-function networks have widely been used.

- **Multilayer perceptrons** An MLP with one input layer, two hidden layers and one output neuron can be described by the following equation:

$$y = \sum_{l=1}^{L} v_l f(\sum_{k=1}^{K} w_{kl}^{(2)} f(\sum_{i=1}^{n} w_{ik}^{(1)} x_i)), \tag{15}$$

  where, n is the input number, K and L are the number of hidden nodes, and $f(\cdot)$ is called activation function, which usually is the logistic function

$$f(z) = \frac{1}{1 + \mathrm{e}^{-az}}, \tag{16}$$

  where a is a constant.
- **Radial-Basis-Function Networks** The theory of radial-basis-function (RBF) networks can also be tracked back to interpolation problems [61]. An RBF network with one single output can be expressed as follows:

$$y(\mathbf{x}) = \sum_{j=1}^{N} w_j \phi(\| \mathbf{x} - \mathbf{x}^{(j)} \|), \tag{17}$$

  where $\phi(\cdot)$ is a set of radial-basis functions, $\| \cdot \|$ is usually a Euclidean norm, the given samples $\mathbf{x}^{(j)}, j = 1, \cdots, N$ are the centers of the radial-basis function, and w_j are unknown coefficients. However, this model is expensive to implement if the number of samples is large. Therefore, a generalized RBF network is more practical

$$y(\mathbf{x}) = \sum_{j=1}^{L} w_j \phi(\| \mathbf{x} - \mu^{(j)} \|). \tag{18}$$

  The main difference is that the number of hidden nodes (L) is ordinarily smaller than the number of samples (N), and the centers of the basis functions ($\mu^{(j)}$) are also unknown parameters that have to be learned. Usually, the output of a generalized RBF network can also be normalized:

$$y(\mathbf{x}) = \frac{\sum_{j=1}^{L} w_j \phi(\| \mathbf{x} - \mu^{(j)} \|)}{\sum_{j=1}^{L} \phi(\| \mathbf{x} - \mu^{(j)} \|)}. \tag{19}$$

5.4 Support Vector Machines

The theory of support vector machines is mainly inspired from statistical learning theory [81]. Major advantages of the support vector machines over other machine learning models such as neural networks, are that there is no local minima during learning and the generalization error does not depend on the dimension of the space. Given l samples $(\mathbf{x}_i, y_i), i = 1, ..., l$, the construction of a model is reduced to the minimization of the following regularized $\epsilon-$insensitive loss function:

$$L = ||\mathbf{w}||^2 + C \cdot \frac{1}{l} \sum_{i=1}^{l} \max \{|y_i - f(\mathbf{x}_i)| - \epsilon\}, \tag{20}$$

where ϵ is the tolerable error, C is a regularization constant and f is the function to be estimated:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b, \qquad \mathbf{w}, \mathbf{x} \in R^n, b \in R. \tag{21}$$

The minimization of (20) is equivalent to the following constrained optimization problem:

$$\text{minimize} \quad \frac{1}{2}||\mathbf{w}||^2 + C \cdot \frac{1}{l} \sum_{i=1}^{l} (\xi_i + \xi_i^*), \tag{22}$$

$$\text{subject to} \quad ((\mathbf{w} \cdot \mathbf{x}_i) + b) - y_i \leq \epsilon + \xi_i \tag{23}$$

$$y_i - ((\mathbf{w} \cdot \mathbf{x}_i) + b) \leq \epsilon + \xi_i^* \tag{24}$$

$$\xi_i, \xi_i^* \geq 0, i = 1, ..., l. \tag{25}$$

Thus, quadratic programming techniques can be applied to solve the minimization problem.

5.5 Comparative Remarks

There are several papers that compare the performance of different approximation models [13, 14, 28, 78, 77, 34]. However, no clear conclusions on the advantages and disadvantages of the different approximation models have been drawn. This is reasonable not only because the performance may depend on the problem to be addressed, but also because more than one criterion needs to be considered. The most important factors are accuracy, both on training data and test data, computational complexity and transparency. It has been found in [36] that an approximate model may introduce false optima, although it has very good performance on the training data, refer to Fig. 3. This is more harmful than a lower approximation accuracy if the model is used in global optimization such as evolutionary optimization. Methods to prevent a neural network model from generating false minima have been suggested in [36], which are very effective for lower dimensional problems.

Some general remarks can still be made on different approximation models, although it is difficult to provide explicit rules on model selection. Firstly, it is recommended to implement first a simple approximate model for a given problem, for example, a lower order polynomial model to see if the given samples can be fit reasonably. If a simple model is found to underfit the samples, a model with higher complexity should be considered, such as higher order polynomials or neural network models. However, if the input space (design space) is high-dimensional and the number of samples is limited, a neural network model is preferred. It is recalled that to estimate the unknown parameters of a second-order polynomial model, at least $(n + 1) \times (n + 2)/2$ data samples are required. Otherwise, the model will be undetermined.

Secondly, if a neural network model, in particular a multilayer perceptrons network is used, it is necessary to consider regulating the model complexity to avoid overfitting. It may also be necessary to try other more efficient training methods [67] if the gradient descent based method is found to be of slow convergence. Besides, RBF networks have found to be of good accuracy as well as of fast training in some studies [77, 34].

6 Data Sampling Techniques

If an approximate model is used for evolutionary computation, both off-line and on-line training will be involved if the evolution is controlled. Off-line learning denotes the training process before the model is used in evolutionary computation. In contract, on-line learning denotes the update of the model during optimization. Usually, the samples for off-line learning can be generated using Monte-Carlo method, however, it has been shown in different research areas that active selection of the samples will improve the model quality significantly. During on-line learning, data selection is strongly related to the search process.

6.1 Off-line Data Sampling

Several data sampling methods have been suggested in the fields of design of experiments [49, 31], statistics and machine learning. Some popular methods are:

- **Design of experiments (DOE)** Orthogonal arrays (OA), central composite designs (CCD), and D-optimality are most widely used in design of experiments. A first-order orthogonal design is one for which $\mathbf{X}^T\mathbf{X}$ is a diagonal matrix, where $\mathbf{X}$ is the extended sample array as defined in Eqn. (3). In other words, the columns of $\mathbf{X}$ are mutually orthogonal.
  Central composite design enables the efficient construction of second-order polynomial models. CCDs are basically first-order (2^n) designs augmented by $2n$ "star" points obtained by perturbing each variable in both positive and positive directions from the central point and the central point to allow estimation of the coefficients of a second-order model. An example of CCD designs is given in Fig. 7 for a two-dimensional problem.
  D-optimality takes advantage of the properties of polynomial models in data sampling. The accuracy of the least square estimate in Eqn. (5) is defined as:
  $$\mathrm{Var}(\hat{\Theta}) = (\mathbf{X}^T\mathbf{X})^{-1}\sigma^2, \tag{26}$$
  where σ^2 is the variance of the estimate error. From Eqn. (26), it can be seen that to improve the quality of fit, one should maximize the determinant of $\mathbf{X}^T\mathbf{X}$. Therefore, the D-optimality is to select the samples in such a way that the determinant of $\mathbf{X}^T\mathbf{X}$

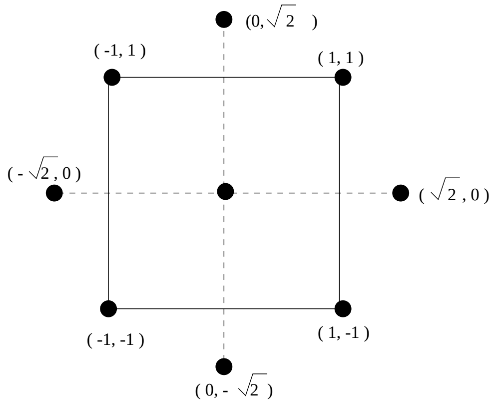

Fig. 7 Central composite designs for $n = 2$. The dots represent the sample points. The samples on the solid lines are the first-order design and those on the dashed lines are central and star points ($\sqrt{n}$).

is maximized. Although developed from the polynomial models, the D-optimality has also shown to be beneficial in data selection for constructing neural networks [16].

- **Active Learning** Active learning has widely been studied in the field of neural network learning [47, 83, 41]. The basic idea is to select the location of the next sampling data in such a way that an objective function is optimized. The objective function can be information gain, entropy reduction, or generalization error. It has been shown that active data selection can improve the generalization ability of neural networks without increasing the number of training samples.

6.2 On-line Data Sampling

Active data selection is also important in the case that the training data has been collected and therefore, the target is how to select a subset of the data for efficient training.

- **Bagging and boosting** Bagging [9] and boosting [26] are two statistical learning methods that have been developed to improve the quality of approximate model using bootstrap techniques [19]. In bagging (bootstrap aggregating), a number of bootstrap models are constructed using different bootstrap samples and the final output is the average of the models. It is shown that bagging is able to reduce the variance of estimate error efficiently. An adaptive bagging technique can reduce both variance and bias [10].
  Boosting algorithms are able to boost a weak learning algorithm into a strong one. A weak algorithm can be inaccurate rules of thumb that are slightly better than a random guess. The main difference between boosting and bagging is that in boosting, the bootstrap samples are affected by the performance of the current model. In addition, the final output is a weighted average of the different models.
- **Active data selection** Some of the statistical active learning methods can also be applied to this type of data selection [60]. A special case of integrated mean square error, called integrated squared bias is used as the criterion to select a subset from available data to improve learning performance. However, it is assumed that the data is noiseless.
- **Data weighting guided by evolution** In [37], a method to weight the available data using the information from the evolutionary algorithm has been suggested. The basic idea is that if information on search direction of the evolutionary algorithm is available, then a larger weight should be given to the data samples located in the region where the evolutionary algorithm will most probably visit in the next generation.
  In [65], several strategies for data sampling have been studied. For example, some strategies use the best individuals to replace the worse ones in the training samples, or the ones that are randomly selected. Some strategies create new points randomly and replace the worst ones in the training samples. It has been found that the strategy that simply re-evaluates the best individuals (best in the sense of the approximate model) with the original fitness function exhibits the best performance. This is actually the best strategy in individual-based evolution control.

7 Discussions

Fitness approximation in evolutionary computation is a research area that has not yet attracted sufficient attention in the evolutionary computation community. Among others, the following points still need to be clarified:

- It is theoretically still unclear in which way the evolutionary algorithm can benefit from the approximate model, although several studies have shown very promising results using approximate models in evolutionary computation. In the least sense, as pointed out in [65], the approximate model can prevent the information in the history of optimization from being lost, although approximate models themselves do not create new information.

– Which type of models helps most, a local one or a global one? It is straightforward to imagine that a global model is able to simplify the search process if the approximate model does not change the properties of the original fitness function. However, from the viewpoint of modeling, to build a local model is much more feasible than to build a global model. In [55], local approximate models are constructed only for local search within a Lamarkian framework of evolution.
– Which is the most effective way of using the approximate models? Generally speaking, the incorporation of approximate models by migrations is not efficient mainly due to the fact that if the models of lower complexity have false optima whose fitness values are higher than those in the correct but computationally complex models, then the migrations between different subpopulations will not help much in any sense. In contrast, the use of approximate models for genetic operations has lower requirement on the model quality because theoretically, the approximate models can help so long as the prediction of the model is better than random guess. However, it is unclear how many evaluations can be spared. The use of the approximate models in fitness evaluation can reduce the number of fitness calculation most effectively. Nevertheless, a poor model quality can degrade the efficiency and even lead to false optima.

In addition, there are also several topics that deserve further research. Some of them are:

– Development of learning algorithms that are efficient and less sensitive to the number of training data. Learning of problem class [33] and incorporation of *a priori* knowledge [39] are two possible approaches. A preliminary study has shown that the structurally optimized neural networks exhibit much better performance than fully connected neural networks for fitness approximation in aerodynamic design optimization [32].
– Approximate model with a variable input dimension. During optimization, the input dimension may change in many cases. For example, if an adaptive representation is used in design optimization, the number of parameters increases or decreases during optimization [54].
– How to handle problems with general nonlinear constraints. In optimization, a penalty term is usually added to the object function if the constraints are voilated. Unfortunately, samples containing a large penalty value cause big difficulties for the model training. Thus, knowledge about the constraints should be extracted and reused in a proper way rather than directly be incorporated in the approximate model.
– Management of different levels of approximation. So far, only functional approximation has been discussed. However, there are different levels of problem approximation in many applications. For example, in computational fluid dynamics simulation, 2D Euler-Lagrange equations, Navier-Stokes equations, quasi 3D simulations and 3D simulations are different approximations of the original problem. Thus, combining different levels of approximation with the approximate model is very interesting [18, 68].

References

1. L. Albert and D.E. Goldberg. Efficient disretization scheduling in multiple dimensions. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 271–278. Morgan Kaufmann, 2002.
2. J.D. Anderson. *Computational Fluid Dynamics: The Basics with Applications.* McGraw Hill, 1995.
3. K. Anderson and Y. Hsu. Genetic crossover strategy using an approximation concept. In *IEEE Congress on Evolutionary Computation*, pages 527–533, Washington D.C., 1999. IEEE.
4. J.-F.M. Barthelemy. Approximation concepts for optimimum structural design - a review. *Structural Optimization*, 5:129–144, 1993.
5. J. A. Biles. Genjam: A genetic algorithm for generating jazz solos. In *Proceedings of International Computer Music Conference*, pages 131–137, 1994.
6. P. Bradshaw, G.A. Mizner, and K. Unsworth. Calculation of compressible turbulent boundary layers

on straight-tapered swept wings,. *AIAA Journal,*, 14:399–400, 1976.

7. J. Branke. Creating robust solutions by means of evolutionary algorithms. In *Proceedings of Parallel Problem Solving from Nature*, Lecture Notes in Computer Science, pages 119–128. Springer, 1998.
8. J. Branke, C. Schmidt, and H. Schmeck. Efficient fitness estimation in noisy environment. In L. Spector et al, editor, *Proceedings of Genetic and Evolutionary Computation*, pages 243–250, San Francisco, CA, July 2001. Morgan Kaufmann.
9. L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.
10. L. Breiman. Using adaptive bagging to debias regressions. Technical Report 547, Dept. of Statistics, University of California, Berkeley, 1999.
11. A. J. Brooker, J. Dennis, P. D. Frank, D. B. Serafini, V. Torczon, and M. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural Optimization*, 17:1–13, 1998.
12. L. Bull. On model-based evolutionary computation. *Soft Computing*, 3:76–82, 1999.
13. W. Carpenter and J.-F. Barthelemy. A comparison of polynomial approximation and artificial neural nets as response surface. Technical Report 92-2247, AIAA, 1992.
14. W. Carpenter and J.-F. Barthelemy. Common misconceptions about neural networks as approximators. *ASCE Journal of Computing in Civil Engineering*, 8(3):345–358, 1994.
15. J.-H. Chen, D.E. Goldberg, S.-Y. Ho, and K. Sastry. Fitness inheritance in multi-objective optimization. In *Proceedings of genetic and Evolutionary Computation Conference*. Morgan Kaufmann, 2002.
16. M.H. Choueiki and C.A. Mount-Campbell. Training data development with the D-optimality criterion. *IEEE Transactions on Neural Networks*, 10(1):56–63, 1999.
17. J. Dennis and V. Torczon. Managing approximate models in optimization. In N. Alexandrov and M. Hussani, editors, *Multidisciplinary design optimization: State-of-the-art*, pages 330–347. SIAM, 1997.
18. D. Eby, R. Averill, W. Punch, and E. Goodman. Evaluation of injection island model GA performance on flywheel design optimization. In *Third Conference on Adaptive Computing in Design and manufacturing*, pages 121–136. Springer, 1998.
19. B. Efron and R. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.
20. M.A. El-Beltagy and A.J. Keane. Optimization for multi-level problems: A comparison of various algorithms. In I. Parmee, editor, *Proceedings of Third International Conference on Adaptive Computing in Design and Manufacture*, pages 111–120. Springer, 1998.
21. M.A. El-Beltagy, P.B. Nair, and A.J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In *Proceedings of Genetic and Evolutionary Conference*, pages 196–203, 1999.
22. M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 371–380. Springer, 2002.
23. M. Farina. A minimal cost hybrid strategy for Pareto optimal front approximation. *Evolutionary Optimization*, 3(1):41–52, 2001.
24. M. Farina. A neural network based generalized response surface multiobjective evolutionary algorithms. In *Congress on Evolutionary Computation*, pages 956–961. IEEE Press, 2002.
25. J.M. Fitzpatrick and J.J. Grefenstette. Genetic algorithms in noisy environments. *Machine Learning*, 3:101–120, 1988.
26. Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2):256–285, 1995.
27. A. Giotis, M. Emmerich, B. Naujoks, and K. Giannakoglou. Low kost stochastic optimization for engineering applications. In *Proceedings of International Conference on Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems*, 2001.
28. A.A. Giunta and L. Watson. A comparison of approximation modeling techniques: Polynomial versus interpolating models. Technical Report 98-4758, AIAA, 1998.

29. J.J. Grefenstette and J.M. Fitzpatrick. Genetic search with approximate fitness evaluations. In *Proceedings of the International Conference on Genetic Algorithms and Their Applications*, pages 112–120, 1985.
30. D.E. Grierson and W.H. Pak. Optimal sizing, geometrical and topological design using a genetic algorithm. *Structural Optimization*, 6(3):151–159, 1993.
31. R.T. Haftka, E.P. Scott, and J.R. Cruz. Optimization and experiments: A survey. *Applied Mechanics Review*, 51(7):435–448, 1998.
32. M. Hüsken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for evolutionary design optimization. In *2002 GECCO Workshop on Approximation and Learning in Evolutionary Computation*, pages 13–16, 2002.
33. M. Hüsken and B. Sendhoff. Evolutionary optimization for problem classes with lamarckian inheritance. In *IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, pages 98–109, 2000.
34. R. Jin, W. Chen, and T.W. Simpson. Comparative studies of metamodeling techniques under miltiple modeling criteria. Technical Report 2000-4801, AIAA, 2000.
35. Y. Jin. *Knowledge in Evolutionary and Learning Systems*. Shaker, Aachen, 2002.
36. Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–792. Morgan Kaufmann, 2000.
37. Y. Jin, M. Olhofer, and B. Sendhoff. Managing approximate models in evolutionary aerodynamic design optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 1, pages 592–599, May 2001.
38. Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.
39. Y. Jin and B. Sendhoff. Knowledge incorporation into neural networks from fuzzy rules. *Neural Processing Letters*, 10(3):231–242, 1999.
40. B. Johanson and R. Poli. GP-music: An interactice genetic programming system for music generation with automated fitness raters. In John R. Koza, Wolfgang Banzhaf, Kumar Chellapilla, Kalyanmoy Deb, Marco Dorigo, David B. Fogel, Max H. Garzon, David E. Goldberg, Hitoshi Iba, and Rick Riolo, editors, *Proceedings of the Third Annual Conference on Genetic Programming*, pages 181–186, 1998.
41. F. Kenji. Statistical active learning in multilayer perceptrons. *IEEE Transactions Neural Networks*, 11(1):16–26, 2000.
42. H.-S. Kim and S.-B. Cho. An efficient genetic algorithms with less fitness evaluation by clustering. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 887–894. IEEE, 2001.
43. S. Kodiyalam, S. Nagendra, and J. DeStefano. Composite sandwich structural optimization with application to satellite components. *AIAA Journal*, 34(3):614–621, 1996.
44. J. Lee and P. Hajela. Parallel genetic algorithms implementation for multidisciplinary rotor blade design. *Journal of Aircraft*, 33(5):962–969, 1996.
45. K.-H. Liang, X. Yao, and C. Newton. Combining landscape approximation and local search in global optimization. In *1999 Congress on Evolutionary Computation*, pages 1514–1520, 1999.
46. K.-H. Liang, X. Yao, and C. Newton. Evolutionary search of approximated n-dimensional landscape. *International Journal of Knowledge-based Intelligent Engineering Systems*, 4(3):172–183, 2000.
47. D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4):305–318, 1992.
48. T. Morimoto, J. De Baerdemaeker, and Y. Hashimoto. An intelligent approach for optimal control of fruit-storage process using neural networks and genetic algorithms. *Computers and Electronics in Agriculture*, 18:205–224, 1997.
49. R. Myers and D. Montgomery. *Response Surface Methodology*. John Wiley & Sons, Inc., New York, 1995.
50. P. Nain and K. Deb. A computationally effective multi-objective search and optimization techniques using coarse-to-fine grain modeling. In *2002 PPSN Workshop on Evolutionary Multiobjective Optimiza-*

tion, 2002.

51. P.B. Nair and A.J. Keane. Combining approximation concepts with algorithm-based structural optimization procedures. In *Proceedings of 39th AIAA/ASMEASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference*, pages 1741–1751, 1998.
52. A. Neumaier. Molecular modeling of proteins and mathematical prediction of protein structures. *SIAM Review*, 39(3):407–460, 1997.
53. V. Oduguwa and R. Roy. Multiobjective optimization of rolling rod product design using metamodeling approach. In *Genetic and Evolutionary Computation Conference*, pages 1164–1171, New York, 2002. Morgan Kaufmann.
54. M. Olhofer, Y. Jin, and B. Sendhoff. Adaptive encoding for aerodynamic shape optimization using evolutiona stratgies. In *Proceedings of IEEE Congress on Evolutionary Computation*, volume 1, pages 576–583, May 2001.
55. Y.S. Ong, P.B. Nair, and A.J. Keane. Evolutionary optimization of computationally expensive problems via surrogate modeling. *AIAA Journal*, 2003.
56. M. Papadrakakis, N. Lagaros, and Y. Tsompanakis. Structural optimization using evolution strategies and neural networks. In *Fourth U.S. National Congress on Computational Mechanics*, San Francisco, 1997.
57. M. Papadrakakis, N. Lagaros, and Y. Tsompanakis. Optimization of large-scale 3D trusses using evolution strategies and neural networks. *Int. J. Space Structures*, 14(3):211–223, 1999.
58. A. Piccolboni and G. Mauri. Application of evolutionary algorithms to protein folding prediction. In J.-K. Hao et al, editor, *Proceedings of the Artificial Evolution 97*, volume 1363 of *Lecture Notes in Computer Science*, pages 123–136. Springer, 1997.
59. S. Pierret. Turbomachinery blade design using a Navier-Stokes solver and artificial neural network. *ASME Journal of Turbomachinery*, 121(3):326–332, 1999.
60. M. Plutowski and H. White. Selecting concise training sets from clean data. *IEEE Transactions on Neural Networks*, 4(2):305–318, 1993.
61. M. Powell. Radial basis functions for multi-variable interpolation: A review. In C. Mason and M.G. Cox, editors, *Algorithms for Approximation*, pages 143–167. Oxford University Press, Oxford, U.K., 1987.
62. K. Rasheed. Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models. In *Proceedings of genetic and Evolutionary Computation Conference*, pages 628–635, Las Vegas, 2000. Morgan Kaufmann.
63. K. Rasheed, S. Vattam, and X. Ni. Comparison of methods for using reduced models to speed up design optimization. In *Proceedings of Genetic and Evolutionary Computation Conference*, pages 1180–1187, New York, 2002. Morgan Kaufmann.
64. A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume V, pages 87–96, 1998.
65. A. Ratle. Optimal sampling strategies for learning a fitness model. In *Proceedings of 1999 Congress on Evolutionary Computation*, volume 3, pages 2078–2085, Washington D.C., July 1999.
66. J. Redmond and G. Parker. Actuator placement based on reachable set optimization for expected disturbance. *Journal Optimization Theory and Applications*, 90(2):279–300, August 1996.
67. R.D. Reed and R.J. Marks II. *Neural Smithing*. MIT, Cambridge, MA, 1999.
68. G. M. Robinson and A. J. Keane. A case for multi-level optimisation in aeronautical design. In *Proceedings of the RAeS Conf. on Multidisciplinary Design and Optimisation*, pages pp. 9.1–9.6. The Royal Aeronautical Society, 1998.
69. Y. Sano and H. Kita. Optimization of noisy fitness functions by means of genetic algorithms using history. In M. Schoenauer et al, editor, *Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*. Springer, 2000.
70. K. Sastry, D.E. Goldberg, and M. Pelikan. Don't evaluate, inherit. In *Proceedings of genetic and Evolutionary Computation Conference*, pages 551–558. Morgan Kaufmann, 2001.
71. G. Schneider. Neural networks are useful tools for drug design design. *Neural Networks*, 13:15–16,

2000.

72. G. Schneider, W. Schrödl, and G. Wallukat. Peptide design by artificial neural networks and computaer-based evolutionary search. *Proceedings of National Academy of Science*, 95:12197–12184, 1998.
73. G. Schneider, J. Schuchhardt, and P. Wrede. Artificial neural networks and simulated molecular evolution are potential tools for sequence-oriented protein design. *CABIOS*, 10(6):635–645, 1994.
74. K. Abboud M. Schoenauer. Surrogate deterministic mutation. In *Artificial Evolution'01*, pages 103–115. Springer, 2002.
75. H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, 1995.
76. M. Sefrioui and J. Periaux. A hierarchical genetic algorithm using multiple models for optimization. In *Parallel Problem Solving from Nature*, volume 1917 of *Lecture Notes in Computer Science*, pages 879–888. Springer, 2000.
77. W. Shyy, P. K. Tucker, and R. Vaidyanathan. Response surface and neural network techniques for rocket engine injector optimization. Technical Report 99-2455, AIAA, 1999.
78. T. Simpson, T. Mauery, J. Korte, and F. Mistree. Comparison of response surface and Kriging models for multidiscilinary design optimization. Technical Report 98-4755, AIAA, 1998.
79. R. Smith, B. Dike, and S. Stegmann. Fitness inheritance in genetic algorithms. In *Proceedings of ACM Symposiums on Applied Computing*, pages 345–350. ACM, 1995.
80. H. Takagi. Interactive evolutionary computation. In *Proceedings of the 5th International Conference on Soft Computing and Information / Intelligent Systems*, pages 41–50, Iizuka, Japan, October 1998. World Scientific.
81. V. Vapnik. *Statistical Learning Theory*. Wiley, 1998.
82. H. Vekeria and I. Parmee. The use of a cooperative multi-level CHC GA for structuralshape optimization. In *Proceedings of fourth European Congress on Intelligent Techniques and Soft Computing*, volume I, pages 471–475, Aachen, 1996.
83. S. Vijayakumar and H. Ogawa. Improving generalization ability through active learning. *Neural Computing*, 1998.
84. D. Yang and S.J. Flockton. Evolutionary algorithms with a coarse-to-fine function smoothing. In *IEEE International Conference on Evolutionary Computation*, pages 657–662, Perth, Australia, 1995. IEEE Press.
85. X. Zhang, B. Julstrom, and W. Cheng. Design of vector quantization codebooks using a genetic algorithm. In *Proceedings of the IEEE Conference on Evolutionary Computation*, pages 525–529. IEEE, 1997.