10

Meta-Modeling in Multiobjective Optimization

Joshua Knowles[1] and Hirotaka Nakayama[2]

[1] School of Computer Science, University of Manchester,
Oxford Road, Manchester M13 9PL, UK
j.knowles@manchester.ac.uk

[2] Konan University, Dept. of Information Science and Systems Engineering,
8-9-1 Okamoto, Higashinada, Kobe 658-8501, Japan
nakayama@konan-u.ac.jp

Abstract. In many practical engineering design and other scientific optimization problems, the objective function is not given in closed form in terms of the design variables. Given the value of the design variables, the value of the objective function is obtained by some numerical analysis, such as structural analysis, fluidmechanic analysis, thermodynamic analysis, and so on. It may even be obtained by conducting a real (physical) experiment and taking direct measurements. Usually, these evaluations are considerably more time-consuming than evaluations of closed-form functions. In order to make the number of evaluations as few as possible, we may combine iterative search with *meta-modeling*. The objective function is modeled during optimization by fitting a function through the evaluated points. This model is then used to help predict the value of future search points, so that high performance regions of design space can be identified more rapidly. In this chapter, a survey of meta-modeling approaches and their suitability to specific problem contexts is given. The aspects of dimensionality, noise, expensiveness of evaluations and others, are related to choice of methods. For the multiobjective version of the meta-modeling problem, further aspects must be considered, such as how to define improvement in a Pareto approximation set, and how to model each objective function. The possibility of interactive methods combining meta-modeling with decision-making is also covered. Two example applications are included. One is a multiobjective biochemistry problem, involving instrument optimization; the other relates to seismic design in the reinforcement of cable-stayed bridges.

10.1 An Introduction to Meta-modeling

In all areas of science and engineering, models of one type or another are used in order to help understand, simulate and predict. Today, numerical methods

Reviewed by: Jerzy Błaszczyński, Poznan University, Poland
Yaochu Jin, Honda Research Institute Europe, Germany
Koji Shimoyama, Tohoku University, Japan
Roman Słowiński, Poznan University of Technology, Poland

make it possible to obtain models or simulations of quite complex and large-scale systems, even when closed-form equations cannot be derived or solved. Thus, it is now a commonplace to model, usually on computer, everything from aeroplane wings to continental weather systems to the activity of novel drugs.

An expanding use of models is to optimize some aspect of the modeled system or process. This is done to find the best wing profile, the best method of reducing the effects of climate change, or the best drug intervention, for example. But there are difficulties with such a pursuit when the system is being modeled numerically. It is usually impossible to find an optimum of the system directly and, furthermore, iterative optimization by trial and error can be very expensive, in terms of computation time.

What is required, to reduce the burden on the computer, is a method of further *modeling the model*, that is, generating a simple model that captures only the relationships between the relevant input and output variables — not modeling any underlying process. *Meta-modeling*, as the name suggests, is such a technique: it is used to build rather simple and computationally inexpensive models, which hopefully replicate the relationships that are *observed* when samples of a more complicated, high-fidelity model or simulation are drawn.[1] Meta-modeling has a relatively long history in statistics, where it is called the response surface method, and is also related to the Design of Experiments (DoE) (Anderson and McLean, 1974; Myers and Montgomery, 1995).

Meta-modeling in Optimization

Iterative optimization procedures employing meta-models (also called surrogate models in this context) alternate between making evaluations on the given high-fidelity model, and on the meta-model. The full-cost evaluations are used to train the initial meta-model, and to update or re-train it, periodically. In this way, the number of full-cost evaluations can often be reduced substantially, whilst a high accuracy is still achieved. This is the main advantage of using a meta-model. A secondary advantage is that the trained meta-model may represent important information about the cost surface and the variables in a relatively simple, and easily interpretable manner. A schematic of the meta-modeling approach is shown in Figure 10.1.

The following pseudocode makes more explicit this process of optimization using models and meta-models:

1. Take an initial sample I of $(\boldsymbol{x},y)$ pairs from the high-fidelity model.
2. From some or all the samples collected, build/update a model M of $p(y \in Y|\boldsymbol{x} \in X)$ or $f : X \to Y$.

[1] Meta-modeling need not refer to modeling of a computational *model*; real processes can also be meta-modeled.

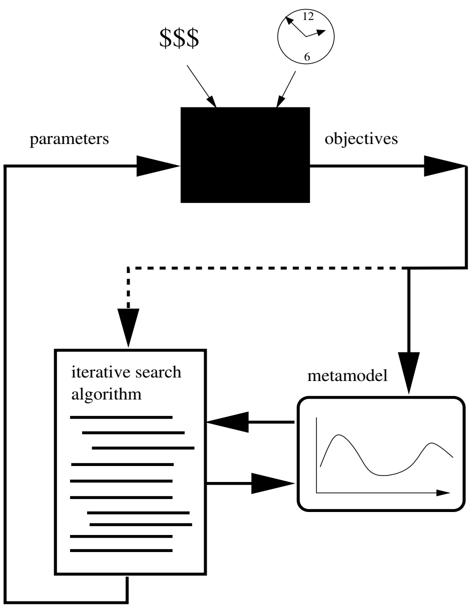


Fig. 10.1. A schematic diagram showing how meta-modeling is used for optimization. The high-fidelity model or function is represented as a black box, which is expensive to use. The iterative searcher makes use of evaluations on both the meta-model and the black box function.

3. Using M, choose a new sample P of points and evaluate them on the high-fidelity model.
4. Until stopping criteria satisfied, return to 2.

The pseudocode is intentionally very general, and covers many different specific strategies. For instance, in some methods the choice of new sample(s) P is made solely based on the current model M, e.g. by finding the optimum on the approximate model (cf. EGO, Jones *et al.* (1998) described in Section 3). Whereas, in other methods, P may be updated based on a memory of previously searched or considered points: e.g. an evolutionary algorithm (EA) using a meta-model may construct the new sample from its current population via the usual application of variation operators, but M is then used to screen out points that it predicts will not have a good evaluation (see (Jin, 2005) and Section 4).

The criterion for selecting the next point(s) to evaluate, or for screening out points, is not always based exclusively on their predicted value. Rather, the estimated *informativeness* of a point may also be accounted for. This can be estimated in different ways, depending on the form of the meta-model. There is a natural tension between choosing of samples because they are predicted to be high-performance points and because they would yield much information, and this tension can be resolved in different ways.

The process of actually constructing a meta-model from data is related to classical regression methods and also to machine learning. Where a model is built up from an initial sample of solutions *only*, the importance of placing those points in the design space in a theoretically well-founded manner is emphasized, a subject dealt with in the classical design of experiments (DoE) literature. When the model is updated using new samples, classical DoE principles do not usually apply, and one needs to look to machine learning theory to understand what strategies might lead to optimal performance. Here care must be taken, however. Although the *supervised learning* paradigm is usually taken as the default method used for training meta-models, it is worth noting that in supervised learning, it is usually a base assumption that the available data for training a regression model are independent and identically distributed samples drawn from some underlying distribution. But in meta-modeling, the samples are not drawn randomly in this way: they are *chosen*, and this means that training sets will often contain highly correlated data, which can affect the estimation of goodness of fit and/or generalization performance. Also, meta-modeling in optimization can be related to *active learning* (Cohn *et al.*, 1996), since the latter is concerned with the iterative choice of training samples; however, active learning is concerned only with maximising what is *learned* whereas meta-modeling in optimization is concerned mainly or partially with seeking optima, so neither supervised learning or active learning are identical with meta-modeling. Finally, in meta-modeling, samples are often added to a 'training set' incrementally (see, e.g. (Cauwenberghs and Poggio, 2001)), and the model is re-trained periodically; how this re-training is achieved also leads to a variety of methods.

Interactive and Evolutionary Meta-modeling

Meta-modeling brings together a number of different fields to tackle the problem of how to optimize expensive functions on a limited budget. Its basis in the DoE literature gives the subject a classical feel, but evolutionary algorithms employing meta-models have been emerging for some years now, too (for a comprehensive survey, see(Jin, 2005)).

In the case of multiobjective problems, it does not seem possible or desirable to make a clear distinction between interactive and evolutionary approaches to meta-modeling. Some methods of managing and using meta-models seek to add one new search point, derived from the model, at every iteration; others are closer to a standard evolutionary algorithm, with a population of solutions used to generate a set of new candidate points, which are then filtered using the model, so that only a few are really evaluated. We deal with both types of methods here (and those that lie in between), and also consider how interaction with a decision maker can be used.

Organization of the Chapter

We begin by considering the different settings in which (single-objective) meta-modeling may be used, and the consequences for algorithm design. In the succeeding section, we survey, in more detail, methods for constructing meta-models, i.e., different regression techniques and how models can be updated iteratively when new samples are collected. Section 10.4 explicitly considers how to handle meta-models in the case of multiobjective optimization. This section elaborates on the way in which a Pareto front approximation is gradually built up in different methods, and also considers how interaction with a DM may be incorporated. Practical advice for evaluating meta-models in multiobjective optimization is given in section 10.5, including ideas for performance measures as well as baseline methods to compare against. Two application sections follow, one from analytical chemistry and one from civil engineering.

10.2 Aspects of Managing Meta-models in Optimization

The combination of meta-modeling and optimization can be implemented in many different ways. In this section, we relate some of the ways of using meta-models to properties of the particular optimization scenario encountered: such properties as the cost of an evaluation, the number that can be performed concurrently, and the features of the cost landscape.[2]

10.2.1 Choice of Model Type

The whole concept of using a model of the cost landscape to improve the performance of an optimizer rests on the assumption that the model will aid in choosing worthy new points to sample, i.e., it will be predictive of the real evaluations of those points, to some degree. This assumption will hold only if the function being optimized is amenable to approximation by the selected *type* of meta-model. It is not realistic to imagine that functions of arbitrary form and complexity can be optimized more efficiently using meta-modeling. This important consideration is related to the 'no free lunch' theorems for search (Wolpert and Macready, 1997) and especially to the pithy observation

[2] The concept of a cost landscape relies on the fact that proximity of points in design space is defined, which in turn, assumes that a choice of problem representation has already been made. We shall not venture any further into the subject of choices of representation here.

of Thomas English that 'learning is hard and optimization is easy in the typical function' (English, 2000).

In theory, given a cost landscape, there should exist a type of model that approximates it fastest as data is collected and fitting progresses. However, we think it is fair to say that little is yet known about which types of model accord best with particular features of a landscape and, in any case, very little may be known to guide this choice. Nonetheless, some basic considerations of the problem do help guide in the choice of suitable models and learning algorithms. In particular, the dimension of the design space is important, as certain models cope better than others with higher dimensions. For example, naive Bayes' regression (Eyheramendy *et al.*, 2003) is used routinely in very high-dimensional feature spaces (e.g. in spam recognition where individual word frequencies form the input space). On the other hand, for low-dimensional spaces, where local correlations are important, naive Bayes' might be very poor, whereas a Gaussian process model (Schwaighofer and Tresp, 2003) might be expected to perform more effectively.

Much of the meta-modeling literature considers only problems over continuous design variables, as these are common in certain engineering domains. However, there is no reason why this restriction need prevail. Meta-modeling will surely become more commonly used for problems featuring discrete design spaces or mixed discrete and continuous variables. Machine learning methods such as classification and regression trees (C&RT) (Breiman, 1984), genetic programming (Langdon and Poli, 2001), and Bayes' regression may be more appropriate for modeling these high-dimensional landscapes than the splines and polynomials that are used commonly in continuous design spaces. Meta-modeling of cost functions in discrete and/or high-dimensional spaces is by no means the only approach to combining machine learning and optimization, however. An alternative is to model the distribution over the variables that leads to high-quality points in the objective space — an approach known as model-based search or estimation of distribution algorithms (Larranaga and Lozano, 2001; Laumanns and Ocenasek, 2002). The learnable evolution model is a related approach (Michalski, 2000; Jourdan *et al.*, 2005). These approaches, though interesting rivals to meta-models, do not predict costs or model the cost function, and are thus beyond the scope of this chapter.

When choosing the type of model to use, other factors that are less linked to properties of the cost landscape/design space should also be considered. Models differ in how they scale in terms of accuracy and speed of training as the number of training samples varies. Some models can be trained incrementally, using only the latest samples (some particular SVMs), whereas others (most multi-layer perceptrons) can suffer from 'catastrophic forgetting' if trained on new samples only, and need to use complete re-training over all samples when some new ones become available, or some other strategy of rehearsal(Robins, 1997). Some types of model need cross-validation to control overfitting, whereas others use regularization. Finally, some types of model, such as Gaussian random fields, model their own error, which can be a distinct advantage when deciding where to sample next (Jones *et al.*, 1998; Emmerich *et al.*, 2006). A more detailed survey of types of model, and methods for training them, is given in section 3.

10.2.2 The Cost of Evaluations

One of the most important aspects affecting meta-modeling for optimization is the actual cost of an evaluation. At one extreme, a cost function may only take the order of $1s$ to evaluate, but is still considered expensive in the context of evolutionary algorithm searches where tens of thousands of evaluations are typical. At the other extreme, when the design points are very expensive, such as vehicle crash tests (Hamza and Saitou, 2005), then each evaluation may be associated with financial costs and/or may take days to organize and carry out. In the former case, there is so little time between evaluations that model-fitting is best carried out only periodically (every few generations of the evolutionary algorithm) and new sample points are generated mainly as a result of the normal EA mechanisms, with the meta-model playing only a subsidiary role of filtering out estimated poor points. In the latter case, the overheads of fitting and cross-validating models is small compared with the time between evaluations, so a number of alternative models can be fitted and validated after every evaluation, and might be used very carefully to decide on the best succeeding design point, e.g. by searching over the whole design space using the meta-model(s) to evaluate points.

The cost of evaluations might affect the meta-modeling strategy in more complicated and interesting ways than the simple examples above, too. In some applications, the cost of evaluating a point is not uniform over the design space, so points may be chosen based partly on their expected cost to evaluate (though this has not been considered in the literature, to our knowledge). In other applications, the cost of determining whether a solution is feasible or not is expensive, while evaluating it on the objective function is cheap. This was the case in (Joslin *et al.*, 2006), and led to a particular strategy of only checking the constraints for solutions passing a threshold on the objective function.

In some applications, the cost of an evaluation can be high in time, yet many can be performed in parallel. This situation occurs, for example, in using optimization to design new drugs via combinatorial chemistry methods. Here, the time to prepare a drug sample and to test it can be of the order of 24 hours, but using high-throughput equipment, several hundred or thousands of different drug compounds can be made and tested in each batch (Corne *et al.*, 2002). Clearly, this places very particular constraints on the meta-modeling/optimization process: there is not always freedom to choose how frequently updates of the model are done, or how many new design points should be evaluated in each generation. Only future studies will show how to best deal with these scenarios.

10.2.3 Advanced Topics

Progress in meta-modeling seems to be heading in several exciting new directions, worth mentioning here.

Typically, when fitting a meta-model to the data, a single global model is learned or updated, using all available design points. However, some research is departing from this by using local models (Atkeson *et al.*, 1997), which are trained only on local subsets of the data (Emmerich *et al.*, 2006). Another departure from the single global model is the possibility of using an ensemble of meta-models (Hamza and Saitou, 2005). Ensemble learning has general advantages in supervised learning scenarios (Brown *et al.*, 2005) and may increase the accuracy of meta-models too. Moreover, Jin and Sendhoff (2004) showed that ensemble methods can be used to predict the quality of the estimation, which can be very useful in the meta-modeling approach.

Noise or stochasticity is an element in many systems that require optimization. Many current meta-modeling methods, especially those based on radial basis functions or Kriging (see next section) assume noiseless evaluation, so that the uncertainty of the meta-model at the evaluated points is assumed to be zero. However, with noisy functions, to obtain more accurate estimates of the expected quality of a design point, several evaluations may be needed. Huang *et al.* (2006) consider how to extend the well-known EGO algorithm (see next section) to account for the case of noise or stochasticity on the objective function. Other noisy optimization methods, such as those based on EAs (Fieldsend and Everson, 2005), could be combined with meta-models in future work.

A further exciting avenue of research is the use of *transductive learning.* It has been shown by Chapelle *et al.* (1999) that in supervised learning of a regression model, knowledge of the future test points (just in design space), at the time of training, can be used to improve the training and lead to better ultimate prediction performance on those points. This results has been imported into meta-modeling by Schwaighofer and Tresp (2003), which compares transductive Gaussian regression methods with standard, inductive ones, and finds them much more accurate.

10.3 Brief Survey of Methods for Meta-modeling

The Response Surface Method (RSM) is probably the most widely applied to meta-modeling (Myers and Montgomery, 1995). The role of RSM is to predict the response y for the vector of design variables $\boldsymbol{x} \in R^n$ on the basis of the given sampled obsevation $(\boldsymbol{x}_i, y_i) \quad (i = 1, \dots, p)$.

Usually, the Response Surface Method is a generic name, and it covers a wide range of methods. Above all, methods using experimental design are famous. However, many of them select sample points only on the basis of statisitical analysis of design variable space. They may provide a good approximation of black-box functions with a mild nonlineality. It is clear, however, that in cases in which the black-box function is highly nonlinear, we can obtain better performance by methods taking into account not only the statistical property of design variable space but also that of range space of the black-box function (in other words, the shape of function).

Moreover, machine learning techniques such as RBF (Radial Basis Function) networks and Support Vector Machines (SVM) have been recently applied for approximating the black-box function (Nakayama *et al.*, 2002, 2003).

10.3.1 Using Design of Experiments

Suppose, for example for simplicitly, that we consider a response function given by a quadratic polynomial:

$$y = \beta_0 + \sum_{i=1}^{n} \beta_i x_i + \sum_{i=1}^{n} \beta_{ii} x_i^2 + \sum_{i<j} \beta_{ij} x_i x_j \tag{10.1}$$

Since the above equation is linear with respect to β_i, we can rewrite the equation (10.1) into the following:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{10.2}$$

where $E(\boldsymbol{\varepsilon}) = 0$, $V(\boldsymbol{\varepsilon}) = \sigma^2 I$.

The above (10.2) is well known as linear regression, and the solution $\boldsymbol{\beta}$ minimizing the squarred error is given by

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{y} \tag{10.3}$$

The variance covariance matrix $V(\hat{\boldsymbol{\beta}}) = cov(\hat{\beta}_i, \hat{\beta}_j)$ of the least squarred error prediction $\hat{\boldsymbol{\beta}}$ given by (10.3) becomes

$$\begin{aligned} V(\hat{\boldsymbol{\beta}}) = cov(\hat{\beta}_i, \hat{\beta}_j) &= E((\hat{\boldsymbol{\beta}} - E(\hat{\boldsymbol{\beta}}))(\hat{\boldsymbol{\beta}} - E(\hat{\boldsymbol{\beta}})) && (10.4) \\ &= (\boldsymbol{X}^T \boldsymbol{X})^{-1} \sigma^2, && (10.5) \end{aligned}$$

where σ^2 is the variance of error in the response y such that $E(\boldsymbol{\varepsilon}\boldsymbol{\varepsilon}^T) = \sigma^2 I$.

i) Orthogonal Design

Orthogonal design is usually applied for experimental design with linear polynomials. Selecting sample points in such a way that the set $\boldsymbol{X}$ is orthogonal, the matrix $\boldsymbol{X}^T \boldsymbol{X}$ becomes diagonal. It is well known that the orthogonal design with the first order model is effective for cases with only main effects or first order interaction effects. For the polynomial regressrion with higher order (≥ 2), orthogonal polynomials are usually used in order to make the design to be orthogonal (namely, $\boldsymbol{X}^T\boldsymbol{X}$ is diagonal). Then the coefficients of polynomials are easily evaluated by using orthogonal arrays.

Another kind of experimental design, e.g., CCD (Cetral Composite Design) is applied mostly for experiments with quadratic polynomials.

ii) D-optimality

Considering the equation (10.5), the matrix $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ should be minimized so that the variance of the predicted $\boldsymbol{\beta}$ may decrease. Since each element of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ has $\det(\boldsymbol{X}^T\boldsymbol{X})$ in the denomnator, we can expect to decrease not only variance but also covariance of β_i by maximizing $\det(\boldsymbol{X}^T\boldsymbol{X})$. This is the idea of D-optimality in design of experiments. In fact, it is usual to use the moment matrix

$$\boldsymbol{M} = \frac{\boldsymbol{X}^T\boldsymbol{X}}{p}, \tag{10.6}$$

where p is the number of sample points.

Other criteria are possible: to minimize the trace of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ (A-optimiality), to minimize the maximal value of the diagonal components of $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$ (minimax criterion), to maximize the minimal eigen value of X^TX (E-optimality). In general, however, the D-optimality criterion is widely used for many practical problems.

10.3.2 Kriging Method

Consider the response $y(\boldsymbol{x})$ as a realization of a random function, $Y(\boldsymbol{x})$ such that

$$Y(\boldsymbol{x}) = \mu(\boldsymbol{x}) + Z(\boldsymbol{x}). \tag{10.7}$$

Here, $\mu(\boldsymbol{x})$ is a global model and $Z(\boldsymbol{x})$ reflecting a deviation from the global model is a random function with zero mean and nonzero covariance given by

$$cov[Z(\boldsymbol{x}), Z(\boldsymbol{x}')] = \sigma^2 R(\boldsymbol{x}, \boldsymbol{x}') \tag{10.8}$$

where R is the correlation between $Z(\boldsymbol{x})$ and $Z(\boldsymbol{x}')$. Usually, the stochastic process is supposed to be stationary, which implies that the correlation $R(\boldsymbol{x}, \boldsymbol{x}')$ depends only on $\boldsymbol{x} - \boldsymbol{x}'$, namely

$$R(\boldsymbol{x}, \boldsymbol{x}') = R(\boldsymbol{x} - \boldsymbol{x}'). \tag{10.9}$$

A commonly used example of such correlation functions is

$$R(\boldsymbol{x}, \boldsymbol{x}') = \exp[-\sum_{i=1}^{n} \theta_i |x_i - x_i'|^2], \tag{10.10}$$

where x_i and x_i' are i-th component of $\boldsymbol{x}$ and $\boldsymbol{x}'$, respectively.

Although a linear regressrion model $\sum_{j=1}^{k} \mu_j f_j(\boldsymbol{x})$ can be applied as a global model in (10.7) (*universal Kriging*), $\mu(\boldsymbol{x}) = \mu$ in which μ is unknown but constant is commonly used in many cases (*ordinary Kriging*). In the ordinary Kriging, the best linear unbiased predictor of y at an untried x can be given by

$$\hat{y}(\boldsymbol{x}) = \hat{\mu} + \boldsymbol{r}^T(\boldsymbol{x})\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu}), \tag{10.11}$$

where $\hat{\mu} = (\mathbf{1}^T\boldsymbol{R}^{-1}\mathbf{1})^{-1}\mathbf{1}^T\boldsymbol{R}^{-1}\boldsymbol{y}$ is the generalized least squares estimator of μ, $\boldsymbol{r}(\boldsymbol{x})$ is the $n \times 1$ vector of correlations $R(\boldsymbol{x}, \boldsymbol{x}_i)$ between Z at $\boldsymbol{x}$ and sampled points $\boldsymbol{x}_i$ $(i = 1, \ldots, p)$, $\boldsymbol{R}$ is an $n \times n$ correlation matrix with (i, j)-element defined by $R(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\mathbf{1}$ is a unity vector whose components are all 1.

Using Expected Improvement

Jones *et al.* (1998) suggested a method called EGO (Efficient Global Optimization) for black-box objective functions. They applied a stochastic process model (10.7) for predictor and the expected improvement as a figure of merit for additional sample points.

The estimated value of the mean of the stochastic process, $\hat{\mu}$, is given by

$$\hat{\mu} = \frac{\mathbf{1}^T\boldsymbol{R}^{-1}\boldsymbol{y}}{\mathbf{1}^T\boldsymbol{R}^{-1}\mathbf{1}}. \tag{10.12}$$

In this event, the variation σ^2 is estimated by

$$\hat{\sigma}^2 = \frac{(\boldsymbol{y} - \mathbf{1}\hat{\mu})^T\boldsymbol{R}^{-1}(\boldsymbol{y} - \mathbf{1}\hat{\mu})}{n}. \tag{10.13}$$

The mean squared error of the predictor is estimated by

$$s^2(\boldsymbol{x}) = \sigma^2[1 - \boldsymbol{r}^T\boldsymbol{R}^{-1}\boldsymbol{r} + \frac{(1 - \mathbf{1}^T\boldsymbol{R}^{-1}\boldsymbol{r})^2}{\mathbf{1}^T\boldsymbol{R}^{-1}\mathbf{1}}]. \tag{10.14}$$

In the following $s = \sqrt{s^2(\boldsymbol{x})}$ is called a standard error.

Using the above predictor on the basis of stochastic process model, Jones *et al.* applied the expected improvemnet for adding a new sample point. Let $f^p_{\min} = \min\{y_1, \ldots, y_p\}$ be the current best function value. They model the uncertainty at $y(\boldsymbol{x})$ by treating it as the realization of a normally distributed random variable Y with mean and standard deviation given by the above predictor and its standard error.

For minimization cases, the improvement at $\boldsymbol{x}$ is $I = [\max(f^p_{\min} - Y,\ 0)$. Therefore, the expected improvement is given by

$$\mathrm{E}[I(\boldsymbol{x})] = \mathrm{E}[\max(f^p_{\min} - Y,\ 0)].$$

It has been shown that the above formula can be expanded as follows:

$$E(I) = \begin{cases} (f^p_{\min} - \hat{y})\Phi(\frac{f^p_{\min} - \hat{y}}{s}) + s\phi(\frac{f^p_{\min} - \hat{y}}{s}) & \text{if } s < 0 \\ 0 & \text{if } s = 0, \end{cases} \tag{10.15}$$

where ϕ is the standard normal density and Φ is the distribution function.

We can add a new sample point which maximizes the expected improvement. Although Jones *et al.* proposed a method for maximizing the expected improvement by using the branch and bound method, it is possible to select the best one among several candidates which are generated randomly in the design variable space.

Furthermore, Schonlau (1997) extended the expected improvement as follows: Letting $I^g = \max((f^p_{\min} - Y)^g,\ 0)$, then

$$E(I^g) = s^g \sum_{i=0}^{g} (-1)^i (\frac{g!}{i!(g-i)!})(f^{p'}_{min})^{g-i} T_i \tag{10.16}$$

where

$$f^{p'}_{min} = \frac{f^p_{min} - \hat{y}}{s}$$

and

$$T_k = -\phi(f^{p'}_{min})(f^{p'}_{min})^{(k-1)} + (k-1)T_{k-2}.$$

Here

$$T_0 = \Phi(f^{p'}_{min})$$
$$T_1 = -\phi(f^{p'}_{min}).$$

It has been observed that larger value of g makes the global search, while smaller value of g the local search. Therefore, we can control the value of g depending upon the situation.

10.3.3 Computational Intelligence

Multi-layer Perceptron Neural Networks

The multi-layer perceptron (MLP) is used in several meta-modeling applications in the literature (Jin *et al.*, 2001; Gaspar-Cunha and Vieira, 2004). It is well-known that MLPs are universal approximators, which makes them attractive for modeling black box functions for which little information about their form is known. But, in practice, it can be difficult and time-consuming to train MLPs effectively as they still have biases and it is easy to get caught in local minima which give far from desirable performance. A large MLP with many weights has a large *capacity*, i.e. it can model complex functions, but it is also easy to over-fit it, so that generalization performance may be poor. The use of a regularization term to help control the complexity is necessary to ensure better generalization performance. Cross-validation can also be used during the training to mitigate overfitting.

Disadvantages of using MLPs may include the difficulty to train it quickly, especially if cross-validation with several folds is used (a problem in some applications). It is not easy to train incrementally (compare with RBFs). Moreover, an MLP does not estimate its own error (compare with Kriging), which means that it can be difficult to estimate the best points to sample next.

Radial Basis Function Networks

Since the number of sample points for predicting objective functions should be as few as possible, incremental learning techniques which predict black-box functions by adding learning samples step by step, are attractive. RBF Networks (RBFN) and Support Vector Machines (SVM) are effective to this end. For RBFN, the necessary information for incremental learning can be easily updated, while the information of support vector can be utilized in selecting additional samples as the sensitivity in SVM. The details of these approaches can be seen in (Nakayama *et al.*, 2002) and (Nakayama *et al.*, 2003). Here, we introduce the incremental learning by RBFN briefly in the following.

The output of an RBFN is given by

$$f(\boldsymbol{x}) = \sum_{j=1}^{m} w_j h_j(\boldsymbol{x}),$$

where $h_j,\ j = 1, \dots, m$ are radial basis functions, e.g.,

$$h_j(\boldsymbol{x}) = e^{-\|\boldsymbol{x}-\boldsymbol{c}_j\|^2/r_j}.$$

Given the training data $(\boldsymbol{x}_i, \hat{y}_i),\ i = 1, \cdots, p$, the learning of RBFN is usually made by solving

$$\min \quad E = \sum_{i=1}^{p} (\hat{y}_i - f(\boldsymbol{x}_i))^2 \ + \ \sum_{j=1}^{m} \lambda_j w_j^2$$

where the second term is introduced for the purpose of regularization.

In general cases with a large number of training data p, the number of basis functions m is set to be less than p in order to avoid overlearning. However, the number of training data is not so large in this paper, because it is desired to be as small as possible in applications under consideration. The value m is set, therefore, to be equal to p in later sections in this paper. Also, the center of radial basis function $\boldsymbol{c}_i$ is set to be $\boldsymbol{x}_i$. The values of λ_j and r_j are usually determined by cross-validation test. It is observed through our experience that in many problems we have a good performance with $\lambda_j = 0.01$ and a simple estimate for r_j given by

$$r = \frac{d_{max}}{\sqrt[n]{np}}, \tag{10.17}$$

where d_{max} is the maximal distance among the data; n is the dimension of data; p is the number of data.

Letting $A = (H_p^T H_p + \Lambda)$, we have

$$A\boldsymbol{w} = H_p^T \hat{\boldsymbol{y}},$$

as a necessary condition for the above minimization. Here

$$H_p^T = [\boldsymbol{h}_1 \ \cdots \ \boldsymbol{h}_p],$$

where $\boldsymbol{h}_j^T = [h_1(\boldsymbol{x}_j), \ldots, h_m(\boldsymbol{x}_j)]$, and Λ is a diagonal matrix whose diagonal components are $\lambda_1 \ \cdots \ \lambda_m$.
Therefore, the learning in RBFN is reduced to finding

$$A^{-1} = (H_p^T H_p + \Lambda)^{-1}.$$

The incremental learning in RBFN can be made by adding new samples and/or a basis function, if necesary. Since the learning in RBFN is equivalent to the matrix inversion A^{-1}, the additional learning here is reduced to the incremental calculation of the matrix inversion. The following algorithm can be seen in (Orr, 1996):

(i) Adding a New Training Sample
Adding a new sample $\boldsymbol{x}_{p+1}$, the incremental learning in RBFN can be made by the following simple update formula: Let

$$H_{p+1} = \begin{bmatrix} H_p \\ \boldsymbol{h}_{p+1}^T \end{bmatrix},$$

where $\boldsymbol{h}_{p+1}^T = [h_1(\boldsymbol{x}_{p+1}), \ldots, h_m(\boldsymbol{x}_{p+1})]$.
Then

$$A_{p+1}^{-1} = A_p^{-1} - \frac{A_p^{-1}\boldsymbol{h}_{p+1}\boldsymbol{h}_{p+1}^T A_p^{-1}}{1 + \boldsymbol{h}_{p+1}^T A_p^{-1}\boldsymbol{h}_{p+1}}.$$

(ii) Adding a New Basis Function
In those cases where a new basis function is needed to improve the learning for a new data, we have the following update formula for the matrix inversion: Let

$$H_{m+1} = \begin{bmatrix} H_m \ \boldsymbol{h}_{m+1} \end{bmatrix},$$

where $\boldsymbol{h}_{m+1}^T = [h_{m+1}(\boldsymbol{x}_1), \ldots, h_{m+1}(\boldsymbol{x}_p)]$.
Then

$$A_{m+1}^{-1} = \begin{bmatrix} A_m^{-1} & \mathbf{0} \\ \mathbf{0}^T & 0 \end{bmatrix}$$

$$+ \frac{1}{\lambda_{m+1} + \boldsymbol{h}_{m+1}^T (I_p - H_m A_m^{-1} H_m^T)\boldsymbol{h}_{m+1}} \times \begin{bmatrix} A_m^{-1} H_m^T \boldsymbol{h}_{m+1} \\ -1 \end{bmatrix} \begin{bmatrix} A_m^{-1} H_m^T \boldsymbol{h}_{m+1} \\ -1 \end{bmatrix}^T.$$

10.3.4 Support Vector Machines

Support vector machines (SVMs) were originally developed for pattern classification and later extended to regression (Cortes and Vapnik, 1995; Vapnik, 1998; Cristianini and Shawe-Tylor, 2000; B.Schölkopf and A.J.Smola, 2002). Regression using SVMs, called often support vector regression, plays an important role in meta-modeling. However, the essential idea of support vector regression lies in SVMs for classification. Therefore, we start with a brief review of SVM for classification problems.

Let X be a space of conditional attributes. For binary classification problems, the value of $+1$ or -1 is assigned to each pattern $\boldsymbol{x}_i \in X$ according to its class $\mathcal{A}$ or $\mathcal{B}$. The aim of machine learning is to predict which class newly observed patterns belong to on the basis of the given training data set $(\boldsymbol{x}_i, y_i)$ $(i = 1, \dots, p)$, where $y_i = +1$ or -1. This is performed by finding a discriminant function $f(\boldsymbol{x})$ such that $f(\boldsymbol{x}) \geqq 0$ for $\boldsymbol{x} \in \mathcal{A}$ and $f(\boldsymbol{x}) < 0$ for $\boldsymbol{x} \in \mathcal{B}$. Linear discriminant functions, in particular, can be expressed by the following linear form

$$f(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{x} + b$$

with the property

$$\begin{aligned} \boldsymbol{w}^T \boldsymbol{x} + b &\geqq 0 \quad \text{for} \quad \boldsymbol{x} \in \mathcal{A} \\ \boldsymbol{w}^T \boldsymbol{x} + b &< 0 \quad \text{for} \quad \boldsymbol{x} \in \mathcal{B}. \end{aligned}$$

In cases where training data set X is not linearly separable, we map the original data set X to a feature space Z by some nonlinear map ϕ. Increasing the dimension of the feature space, it is expected that the mapped data set becomes linearly separable. We try to find linear classifiers with maximal margin in the feature space. Letting $\boldsymbol{z}_i = \phi(\boldsymbol{x}_i)$, the separating hyperplane with maximal margin can be given by solving the following problem with the normalization $\boldsymbol{w}^T \boldsymbol{z} + b = \pm 1$ at points with the minimum interior deviation:

$$\begin{aligned} \min_{\boldsymbol{w}, b} \quad & ||\boldsymbol{w}|| && (\text{SVM}_{hard})_P \\ \text{s.t.} \quad & y_i \left(\boldsymbol{w}^T \boldsymbol{z}_i + b \right) \geqq 1, \ i = 1, \dots, p. \end{aligned}$$

Dual problem of $(\text{SVM}_{hard})_P$ with $\frac{1}{2}||\boldsymbol{w}||_2^2$ is

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{p} \alpha_i \alpha_j y_i y_j \phi(\boldsymbol{x}_i)^T \phi(\boldsymbol{x}_j) && (\text{SVM}_{hard})_D \\ \text{s.t.} \quad & \sum_{i=1}^{p} \alpha_i y_i = 0, \\ & \alpha_i \geqq 0, \ i = 1, \dots, p. \end{aligned}$$

Using the kernel function $K(\boldsymbol{x}, \boldsymbol{x}') = \phi(\boldsymbol{x})^T \phi(\boldsymbol{x}')$, the problem $(\text{SVM}_{hard})_D$ can be reformulated as follows:

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{p} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{p} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) && (\text{SVM}_{hard}) \\ \text{s.t.} \quad & \sum_{i=1}^{p} \alpha_i y_i = 0, \\ & \alpha_i \geqq 0, \ i = 1, \dots, p. \end{aligned}$$

Although several kinds of kernel functions have been suggested, the Gaussian kernel

$$K(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}'||^2}{2r^2}\right)$$

is popularly used in many cases.

MOP/GP Approaches to Support Vector Classification

In 1981, Freed and Glover suggested to get just a hyperplane separating two classes with as few misclassified data as possible by using goal programming (Freed and Glover, 1981) (see also (Erenguc and Koehler, 1990)). Let ξ_i denote the exterior deviation which is a deviation from the hyperplane of a point $\boldsymbol{x}_i$ improperly classified. Similarly, let η_i denote the interior deviation which is a deviation from the hyperplane of a point $\boldsymbol{x}_i$ properly classified. Some of main objectives in this approach are as follows:

i) Minimize the maximum exterior deviation (decrease errors as much as possible)

ii) Maximize the minimum interior deviation (i.e., maximize the margin)

iii) Maximize the weighted sum of interior deviation

iv) Minimize the weighted sum of exterior deviation.

Introducing the idea iv) above, the well known soft margin SVM with slack variables (or, exterior deviations) ξ_i $(i = 1, \dots, p)$ which allow classification errors to some extent can be formulated as follows:

$$\begin{aligned} \min_{\boldsymbol{w}, b, \xi_i} \quad & \frac{1}{2}||\boldsymbol{w}||_2^2 + C\sum_{i=1}^{p} \xi_i \\ \text{s.t.} \quad & y_i\left(\boldsymbol{w}^T \boldsymbol{z}_i + b\right) \geqq 1 - \xi_i, \\ & \xi_i \geqq 0, \ \ i = 1, \dots, p, \end{aligned} \qquad (\text{SVM}_{soft})_P$$

where C is a trade-off parameter between minimizing $||\boldsymbol{w}||_2^2$ and minimizing $\sum_{i=1}^{p} \xi_i$.

Using a kernel function in the dual problem yields

$$\begin{aligned} \max_{\alpha_i} \quad & \sum_{i=1}^{p} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{p} \alpha_i \alpha_j y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \text{s.t.} \quad & \sum_{i=1}^{p} \alpha_i y_i = 0, \\ & 0 \leqq \alpha_i \leqq C, \ \ i = 1, \dots, p. \end{aligned} \qquad (\text{SVM}_{soft})$$

Lately, taking into account the objectives (ii) and (iv) of goal programming, we have the same formulation of ν-support vector algorithm developed by Schölkopf and Smola (1998):

$$\begin{array}{lll} \underset{\boldsymbol{w},b,\xi_i,\rho}{\min} & \frac{1}{2}||\boldsymbol{w}||_2^2 - \nu\rho + \frac{1}{p}\sum_{i=1}^{p}\xi_i & (\nu\text{−SVM})_P \\ \text{s.t.} & y_i\left(\boldsymbol{w}^T\boldsymbol{z}_i + b\right) \geqq \rho - \xi_i, & \\ & \rho \geqq 0,\ \ \xi_i \geqq 0,\ \ i = 1,\dots,p, & \end{array}$$

where $0 \leqq \nu \leqq 1$ is a parameter.

Compared with the existing soft margin algorithm, one of the differences is that the parameter C for slack variables does not appear, and another difference is that the new variable ρ appears in the above formulation. The problem $(\nu\text{−SVM})_P$ maximizes the variable ρ which corresponds to the minimum interior deviation (i.e., the minimum distance between the separating hyperplane and correctly classified points).

The Lagrangian dual problem to the problem $(\nu\text{−SVM})_P$ is as follows:

$$\begin{array}{lll} \underset{\alpha_i}{\max} & -\frac{1}{2}\sum_{i,j=1}^{p} y_i y_j \alpha_i \alpha_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) & (\nu\text{−SVM}) \\ \text{s.t.} & \sum_{i=1}^{p} y_i\alpha_i = 0, & \\ & \sum_{i=1}^{\ell} \alpha_i \geqq \nu, & \\ & 0 \leqq \alpha_i \leqq \frac{1}{p},\ \ i = 1,\dots,p. & \end{array}$$

Other variants of SVM considering both slack variables for misclassified data points (i.e., exterior deviations) and surplus variables for correctly classified data points (i.e., interior deviations) are possible (Nakayama and Yun, 2006a): Considering iii) and iv) above, we have the fomula of total margin SVM, while ν−SVM can be derived from i) and iii).

Finally, μ−ν−SVM is derived by considering the objectives i) and ii) in MOP/GP:

$$\begin{array}{lll} \underset{\boldsymbol{w},b,\rho,\sigma}{\min} & \frac{1}{2}\|\boldsymbol{w}\|_2^2 - \nu\rho + \mu\sigma & (\mu - \nu\text{−SVM})_P \\ \text{s.t.} & y_i\left(\boldsymbol{w}^T\boldsymbol{z}_i + b\right) \geqq \rho - \sigma,\ \ i = 1,\dots,p, & \\ & \rho \geqq 0,\ \ \sigma \geqq 0, & \end{array}$$

where ν and μ are parameters.

The dual formulation is given by

$$\begin{aligned}
\max_{\alpha_i} \quad & -\frac{1}{2}\sum_{i,j=1}^{p} \alpha_i\alpha_j y_i y_j K\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) && (\mu-\nu\text{–SVM}) \\
\text{s.t.} \quad & \sum_{i=1}^{p} \alpha_i y_i = 0, \\
& \nu \leqq \sum_{i=1}^{p} \alpha_i \leqq \mu, \\
& \alpha_i \geqq 0, \ \ i = 1, \dots, p.
\end{aligned}$$

Letting $\boldsymbol{\alpha}^*$ be the optimal solution to the problem (μ–ν–SVM), the offset b^* can be chosen easily for any i satisfying $\alpha_i^* > 0$. Otherwise, b^* can be obtained by the similar way with the decision of the b^* in the other algorithms.

Support Vector Regression

Support Vector Machines were extended to regression by introducing the ε insensitive loss function by Vapnik (1998). Denote the given sample data by $(\boldsymbol{x}_i, y_i)$ for $i = 1, ..., p$. Suppose that the regression function on the Z space is expressed by $f(\boldsymbol{z}) = \sum_{i=1}^{p} w_i \boldsymbol{z}_i + b$. The linear ε insensitive loss function is defined by

$$L^{\varepsilon}(\boldsymbol{z}, y, f) = |y - f(\boldsymbol{z})|_{\varepsilon} = \max(0, |y - f(\boldsymbol{z})| - \varepsilon).$$

For a given insensitivity parameter ε,

$$\begin{aligned}
\min_{\boldsymbol{w}, b, \varepsilon, \xi_i, \acute{\xi}_i} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\Big(\frac{1}{p}\sum_{i=1}^{p}(\xi_i + \acute{\xi}_i)\Big) && (soft\text{–SVR})_P \\
\text{s.t.} \quad & \big(\boldsymbol{w}^T \boldsymbol{z}_i + b\big) - y_i \leqq \varepsilon + \xi_i, \ \ i = 1, \dots, p, \\
& y_i - \big(\boldsymbol{w}^T \boldsymbol{z}_i + b\big) \leqq \varepsilon + \acute{\xi}_i, \ \ i = 1, \dots, p, \\
& \varepsilon, \ \ \xi_i, \ \ \acute{\xi}_i \geqq 0
\end{aligned}$$

where C is a trade-off parameter between the norm of $\boldsymbol{w}$ and ξ ($\acute{\xi}$).

The dual formulation to $(soft\text{–SVR})_P$ is given by

$$\begin{aligned}
\max_{\alpha_i,\acute{\alpha}_i} \quad & -\frac{1}{2}\sum_{i,j=1}^{p}\left(\acute{\alpha}_i-\alpha_i\right)\left(\acute{\alpha}_j-\alpha_j\right)K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right) \qquad (soft\text{−SVR})\\
& +\sum_{i=1}^{p}\left(\acute{\alpha}_i-\alpha_i\right)y_i-\varepsilon\sum_{i,j=1}^{p}\left(\acute{\alpha}_i+\alpha_i\right)\\
\text{s.t.} \quad & \sum_{i=1}^{p}\left(\acute{\alpha}_i-\alpha_i\right)=0,\\
& 0\leqq\acute{\alpha}_i\leqq\frac{C}{p},\ \ 0\leqq\alpha_i\leqq\frac{C}{p},\ \ i=1,\ldots,p.
\end{aligned}$$

In order to decide ε automatically, Schölkopf and Smola proposed ν-SVR as follows (Schölkopf and Smola, 1998):

$$\begin{aligned}
\min_{\boldsymbol{w},b,\varepsilon,\xi_i,\acute{\xi}_i} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2+C\Big(\nu\varepsilon+\frac{1}{p}\sum_{i=1}^{p}(\xi_i+\acute{\xi}_i)\Big) \qquad (\nu\text{−SVR})_P\\
\text{s.t.} \quad & \big(\boldsymbol{w}^T\boldsymbol{z}_i+b\big)-y_i\leqq\varepsilon+\xi_i,\ \ i=1,\ldots,p,\\
& y_i-\big(\boldsymbol{w}^T\boldsymbol{z}_i+b\big)\leqq\varepsilon+\acute{\xi}_i,\ \ i=1,\ldots,p,\\
& \varepsilon,\ \ \xi_i,\ \ \acute{\xi}_i\geqq 0,
\end{aligned}$$

where C and ν are trade-off parameters between the norm of $\boldsymbol{w}$ and ε and ξ_i $(\acute{\xi}_i)$.

The dual formulation to $(\nu\text{−SVR})_P$ is given by

$$\begin{aligned}
\max_{\alpha_i,\acute{\alpha}_i} \quad & -\frac{1}{2}\sum_{i,j=1}^{p}\left(\acute{\alpha}_i-\alpha_i\right)\left(\acute{\alpha}_j-\alpha_j\right)K\left(\boldsymbol{x}_i,\boldsymbol{x}_j\right) \qquad (\nu\text{−SVR})\\
& +\sum_{i=1}^{p}\left(\acute{\alpha}_i-\alpha_i\right)y_i\\
\text{s.t.} \quad & \sum_{i=1}^{p}\left(\acute{\alpha}_i-\alpha_i\right)=0,\\
& \sum_{i=1}^{p}\left(\acute{\alpha}_i+\alpha_i\right)\leqq C\cdot\nu,\\
& 0\leqq\acute{\alpha}_i\leqq\frac{C}{p},\ \ 0\leqq\alpha_i\leqq\frac{C}{p},\ \ i=1,\ldots,p.
\end{aligned}$$

In a similar fashion to classification, we can obtain $(\mu-\nu\text{−SVR})$ as follows:

$$\begin{aligned}
\min_{\boldsymbol{w},b,\varepsilon,\xi,\acute{\xi}} \quad & \frac{1}{2}\|\boldsymbol{w}\|_2^2 + \nu\varepsilon + \mu(\xi + \acute{\xi}) && (\mu - \nu\text{−SVR})_P \\
\text{s.t.} \quad & \left(\boldsymbol{w}^T\boldsymbol{z}_i + b\right) - y_i \leqq \varepsilon + \xi, \ \ i = 1, \ldots, p, \\
& y_i - \left(\boldsymbol{w}^T\boldsymbol{z}_i + b\right) \leqq \varepsilon + \acute{\xi}, \ \ i = 1, \ldots, p, \\
& \varepsilon, \ \xi, \ \acute{\xi} \geqq 0,
\end{aligned}$$

where ν and μ are trade-off parameters between the norm of $\boldsymbol{w}$ and ε and $\acute{\xi}$.

The dual formulation of $\mu - \nu$−SVR is as follows:

$$\begin{aligned}
\max_{\alpha_i,\acute{\alpha}_i} \quad & -\frac{1}{2}\sum_{i,j=1}^{p}(\acute{\alpha}_i - \alpha_i)(\acute{\alpha}_j - \alpha_j)K(\boldsymbol{x}_i, \boldsymbol{x}_j) && (\mu - \nu\text{−SVR}) \\
& + \sum_{i=1}^{p}(\acute{\alpha}_i - \alpha_i)\, y_i \\
\text{s.t.} \quad & \sum_{i=1}^{p}(\acute{\alpha}_i - \alpha_i) = 0, \\
& \sum_{i=1}^{p}\acute{\alpha}_i \leqq \mu, \ \ \sum_{i=1}^{p}\alpha_i \leqq \mu, \\
& \sum_{i=1}^{p}(\acute{\alpha}_i + \alpha_i) \leqq \nu, \\
& \acute{\alpha}_i \geqq 0, \ \ \alpha_i \geqq 0, \ \ i = 1, \ldots, p.
\end{aligned}$$

10.4 Managing Meta-models of Multiple Objectives

Meta-modeling in the context of multiobjective optimization has been considered in several works in recent years (Chafekar *et al.*, 2005; Emmerich *et al.*, 2006; Gaspar-Cunha and Vieira, 2004; Keane, 2006; Knowles, 2006; Nain and Deb, 2002; Ray and Smith, 2006; Voutchkov and Keane, 2006). The generalization to multiple objective functions has led to a variety of approaches, with differences in what is modeled, and also how models are updated. These differences follow partly from the different possible methods that there are of building up a Pareto front approximation (see Figure 10.2).

In a modern multiobjective evolutionary algorithm approach like NSGA-II, selection favours solutions of low dominance rank and uncrowded solutions, which helps build up a diverse and converged Pareto set approximation. A straightforward way to obtain a meta-modeling-based multiobjective optimization algorithm is thus to take NSGA-II and simply plug in meta-models of each independent objective function. This can be achieved by running NSGA-II for several generations on the meta-model (initially constructed from a DoE sample), and then cycling through phases of selection, evaluation on the

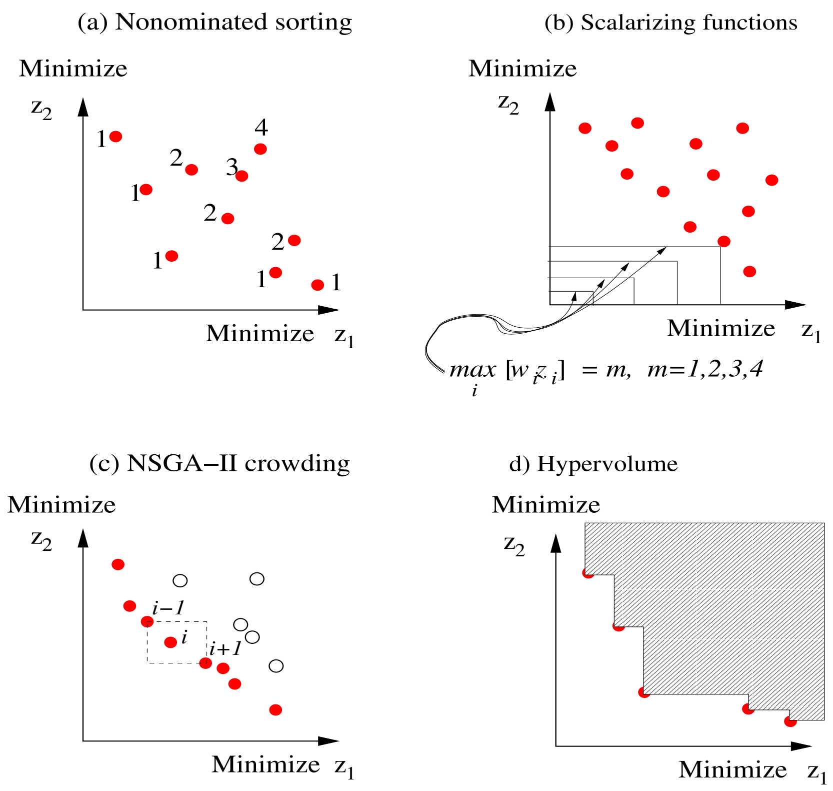


Fig. 10.2. Building up a Pareto front approximation can be achieved by different routes. Four are shown here: nondominated sorting, scalarizing, crowding, and maximising hypervolume.

real model, and update of the meta-model. This approach is the one taken by Voutchkov and Keane (2006) wherein a variety of response surface methods are compared, including splines, radial basis functions and polynomial regression. As the authors explain, an advantage of this approach is that each objective could, in theory, be modeled by a different type of response surface method (appropriate to it), or some objectives may be cheap to compute and may not need modeling at all.

An alternative to simply plugging meta-models into the existing selection step of an MOEA, is to use the meta-models, instead, to *pre-screen* points. This approach, much like EGO, may base the screening on both the predicted value of points and the confidence in these predictions. In a method still based on NSGA-II, Emmerich *et al.* (2006) proposed a number of different pre-screening criteria for multiobjective optimization, including the expected improvement and the probability of improvement. Note that in the case of multiobjective optimization, improvement is relative to the whole Pareto set approximation achieved so far, not a single value. Thus, to measure improvement, the estimated increase in hypervolume (Zitzler *et al.*, 2003) of the current approximation set (were a candidate point added to it) is used, based on a meta-model for each objective function. In experiments, Emmerich et al compared four different screening criteria on two and three-objective problems, and found improvement over the standard NSGA-II in all cases.

The approach of Emmerich et al is a sophisticated method of generalizing the use of meta-models to multiobjective optimization, via MOEAs, though it is as yet open whether this sophistication leads to better performance than the simpler method of Voutchkov and Keane (2006). Moreover, it does seem slightly unnatural to marry NSGA-II, which uses dominance rank and crowdedness to select its 'parents', with a meta-modeling approach that uses the hypervolume to estimate probable improvement. It would seem more logical to use evolutionary algorithms that themselves maximize hypervolume as the fitness assignment method, such as (Emmerich *et al.*, 2005). It remains to be seen whether such approaches would perform even better.

One worry with the methods described so far is that fitness assignments based on dominance rank (like NSGA-II) can perform poorly when the number of objectives is greater than three or four (Hughes, 2005). Hypervolume may be a better measure but it is very expensive to compute for large dimension, as the complexity of known methods for computing it is polynomial in the set size but exponential in d. Thus, scaling up objective dimension in methods based on either of the approaches described above might prove difficult.

A method that does not use either hypervolume or dominance rank is the ParEGO approach proposed by Knowles (2006). This method is a generalization to multiobjective optimization of the well-founded EGO algorithm (Jones *et al.*, 1998). To build up a Pareto front, ParEGO uses a series of weighting vectors to scalarize the objective functions. At each iteration of the algorithm, a new candidate point is determined by (i) computing the expected improvement (Jones *et al.*, 1998) in the 'direction' specified by the weighting vector drawn for that iteration, and (ii) searching for a point that maximizes this expected improvement (a single-objective evolutionary algorithm is used for this search). The use of such scalarizing weight vectors has been shown to scale well to many objectives, compared with Pareto ranking (Hughes, 2005). The ParEGO method has the additional advantage that it would be relatively straightforward to make it interactive, allowing the user to narrow down the set of scalarizing weight vectors to allow focus on a particular region of the Pareto front. This can further reduce the number of function evaluations it is necessary to perform.

Yet a further way of building up a Pareto front is exemplified in the final method we review here. (Chafekar *et al.*, 2005) proposes a genetic algorithm with meta-models OEGADO, based closely on their own method for single objective optimization. To make it work for the multiobjective case, a distinct genetic algorithm is run for each objective, with information exchange occurring between the algorithms at intervals, which helps the GAs to find the compromise solutions. The fact that each objective is optimized by its own genetic algorithm means that objective functions with different computational overhead can be appropriately handled — slow objectives do not slow down the evaluation of faster ones. The code may also be trivially implemented on parallel architectures.

10.4.1 Combining Interactive Methods and EMO for Generating a Pareto Frontier

Aspiration Level Methods for Interactive Multiobjective Programming

Since there may be many Pareto solutions in practice, the final decision should be made among them taking the total balance over all criteria into account. This is a problem of value judgment of DM. The totally balancing over criteria is usually called *trade-off.* Interactive multiobjective programming searches a solution in an interactive way with DM while making trade-off analysis on the basis of DM's value judgment. Among them, the aspiration level approach is now recognized to be effective in practice, because

(i) it does not require any consistency of DM's judgment,
(ii) aspiration levels reflect the wish of DM very well,
(iii) aspiration levels play the role of probe better than the weight for objective functions.

As one of aspiration level approaches, one of authors proposed the satisficing trade-off method (Nakayama and Sawaragi, 1984). Suppose that we have objective functions $\boldsymbol{f}(\boldsymbol{x}) := (f_1(\boldsymbol{x}), \ldots, f_r(\boldsymbol{x}))$ to be minimized over $\boldsymbol{x} \in X \subset R^n$. In the satisficing trade-off method, the aspiration level at the k-th iteration $\overline{\boldsymbol{f}}^k$ is modified as follows:

$$\overline{\boldsymbol{f}}^{k+1} = T \circ P(\overline{\boldsymbol{f}}^k).$$

Here, the operator P selects the Pareto solution nearest in some sense to the given aspiration level $\overline{\boldsymbol{f}}^k$. The operator T is the trade-off operator which changes the k-th aspiration level $\overline{\boldsymbol{f}}^k$ if DM does not compromise with the shown solution $P(\overline{\boldsymbol{f}}^k)$. Of course, since $P(\overline{\boldsymbol{f}}^k)$ is a Pareto solution, there exists no feasible solution which makes all criteria better than $P(\overline{\boldsymbol{f}}^k)$, and thus DM has to trade-off among criteria if he wants to improve some of criteria. Based on this trade-off, a new aspiration level is decided as $T \circ P(\overline{\boldsymbol{f}}^k)$. Similar process is continued until DM obtains an agreeable solution.

On the Operation P

The operation which gives a Pareto solution $P(\overline{\boldsymbol{f}}^k)$ nearest to $\overline{\boldsymbol{f}}^k$ is performed by some auxiliary scalar optimization. It has been shown in Sawaragi-Nakayama-Tanino (1985) that the only one scalarization technique, which provides any Pareto solution regardless of the structure of problem, is of the Tchebyshev norm type. However, the scalarization function of Tchebyshev norm type yields not only a Pareto solution but also a weak Pareto solution. Since weak Pareto solutions have a possibility that there may be another solution which improves a criteria while others being fixed, they are not necessarily "*efficient*" as a solution in decision making. In order to exclude weak Pareto solutions, the following scalarization function of the augmented Tchebyshev type can be used:

$$\max_{1 \leqq i \leqq r} \omega_i \left(f_i(\boldsymbol{x}) - \overline{f}_i\right) + \alpha \sum_{i=1}^{r} \omega_i f_i(\boldsymbol{x}), \tag{10.18}$$

where α is usually set a sufficiently small positive number, say 10^{-6}.

The weight ω_i is usually given as follows: Let f_i^* be an ideal value which is usually given in such a way that $f_i^* < \min\{f_i(\boldsymbol{x}) \mid \boldsymbol{x} \in X\}$. For this circumstance, we set

$$\omega_i^k = \frac{1}{\overline{f}_i^k - f_i^*}. \tag{10.19}$$

The minimization of (10.18) with (10.19) is usually performed by solving the following equivalent optimization problem, because the original one is not smooth:

$$\begin{array}{lll} \text{(AP)} & \underset{z,\ \boldsymbol{x}}{\text{minimize}} & z + \alpha \displaystyle\sum_{i=1}^{r} \omega_i f_i(\boldsymbol{x}) \\ & \text{subject to} & \omega_i^k \left(f_i(\boldsymbol{x}) - \overline{f}_i^k\right) \leqq z \\ & & \boldsymbol{x} \in X. \end{array} \tag{10.20}$$

On the Operation T

In cases that DM is not satisfied with the solution for $P(\overline{\boldsymbol{f}}^k)$, he/she is requested to answer his/her new aspiration level $\overline{\boldsymbol{f}}^{k+1}$. Let $\boldsymbol{x}^k$ denote the Pareto solution obtained by projection $P(\overline{\boldsymbol{f}}^k)$, and classify the objective functions into the following three groups:

(i) the class of criteria which are to be improved more,
(ii) the class of criteria which may be relaxed,
(iii)the class of criteria which are acceptable as they are.

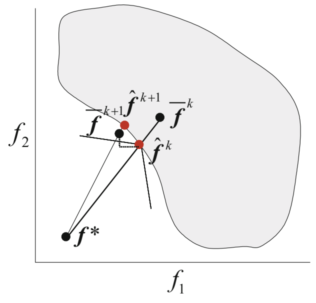


Fig. 10.3. Satisficing Trade-off Method

Let the index set of each class be denoted by I_I^k, I_R^k, I_A^k, respectively. Clearly, $\overline{f}_i^{k+1} < f_i(x^k)$ for all $i \in I_I^k$. Usually, for $i \in I_A^k$, we set $\overline{f}_i^{k+1} = f_i(\boldsymbol{x}^k)$. For $i \in I_R^k$, DM has to agree to increase the value of $\overline{f}_i^{k+1}$. It should be noted that an appropriate sacrifice of f_j for $j \in I_R^k$ is needed for attaining the improvement of f_i for $i \in I_I^k$.

Combining Satisficing Trade-off Method and Sequential Approximate Optimization

Nakayama and Yun proposed a method combining the satisficing trade-off method for interactive multiobjective programming and the sequential approximate optimization using $\mu - \nu-$SVR (Nakayama and Yun, 2006b). The procedure is summarized as follows:

Step 1. (Real Evaluation)
Evaluate actually the values of objective functions $\boldsymbol{f}(\boldsymbol{x}_1), \boldsymbol{f}(\boldsymbol{x}_2), \dots, \boldsymbol{f}(\boldsymbol{x}_\ell)$ for sampled data $\boldsymbol{x}_1, \dots, \boldsymbol{x}_\ell$ through computational simulation analysis or experiments.

Step 2. (Approximation)
Approximate each objective function $\hat{f}_1(\boldsymbol{x}), \dots, \hat{f}_m(\boldsymbol{x})$ by the learning of $\mu - \nu-$SVR on the basis of real sample data set.

Step 3. (Find a Pareto Solution Nearest to the Aspiration Level and Generate Pareto Frontier)
Find a Pareto optimal solution nearest to the given aspiration level for the approximated objective functions $\hat{\boldsymbol{f}}(\boldsymbol{x}) := (\hat{f}_1(\boldsymbol{x}), \dots, \hat{f}_m(\boldsymbol{x}))$. This is performed by using GA for minimizing the augmented Tchebyshev scalarization function (10.18). In addition, generate Pareto frontier by MOGA for accumulated individuals during the procedure for optimizing the augmented Tchebyshev scalarization function.

Step 4. (Choice of Additional Learning Data)
Choose the additional ℓ_0-data from the set of obtained Pareto optimal solutions. Go to Step 1. (Set $\ell \leftarrow \ell + \ell_0$.)

<u>*how to choose the additional data*</u>

Stage 0. First, add the point with highest achievement degree among Pareto optimal solutions obtained in Step 3. ($\leftarrow$ local information)

Stage 1. Evaluate the ranks for the real sampled data of Step 1 by the ranking method (Fonseca and Fleming, 1993).

Stage 2. Approximate the rank function associated with the ranks calculated in the Stage 1 by $\mu-\nu-$SVR.

Stage 3. Calculate the expected fitness for Pareto optimal solutions obtained in Step 3.

Stage 4. Among them, add the point with highest rank. ($\leftarrow$global information)

Next, we consider the following problem (Ex-1):

$$\begin{aligned} \text{minimize} \quad & f_1 := x_1 + x_2 \\ & f_2 := 20\cos(15x_1) + (x_1-4)^4 + 100\sin(x_1x_2) \\ \text{subject to} \quad & 0 \leqq x_1,\ x_2 \leqq 3. \end{aligned}$$

The true function of each objective function f_1 and f_2 in the problem (Ex-1) are shown in Fig. 10.4.

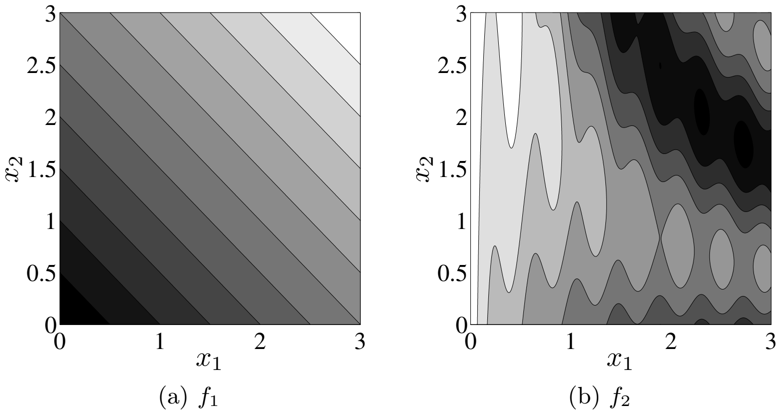
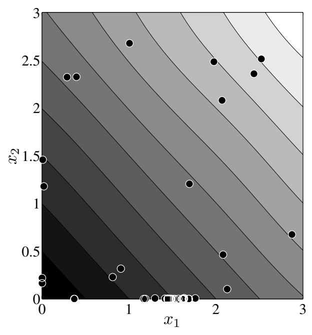
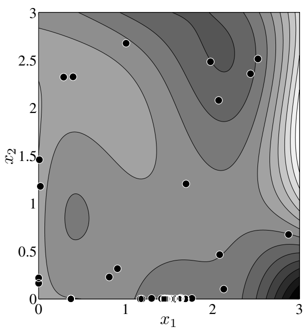


Fig. 10.4. The true contours to the problem

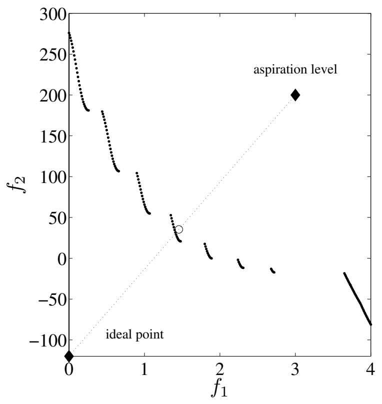
In our simulation, the ideal point and the aspiration level is respectively given by



(a) contour of f_1



(b) contour of f_2



(c) population at the final generation

Fig. 10.5. # sample data : 50 points (Ex-1)

$$\left(f_1^*,\ f_2^*\right) = (0,\ -120),$$
$$\left(\overline{f}_1,\ \overline{f}_2\right) = (3,\ 200),$$

and the closest Pareto solution to the above aspiration level is as follows:

$$\text{exact optimal solution } \left(\hat{x}_1,\ \hat{x}_2\right) = (1.41321,\ 0)$$
$$\text{exact optimal value } \left(\hat{f}_1,\ \hat{f}_2\right) = (1.41321,\ 30.74221)$$

Starting with initial data 10 points randomly, we obtained the following approximate solution by proposed method after 50 real evaluations:

$$\text{approximate solution } \left(x_1,\ x_2\right) = (1.45748,\ 0)$$
$$\text{approximate value } \left(f_1,\ f_2\right) = (1.45748,\ 35.34059)$$

The final result is shown in Fig. 10.5

Additionally, a rough configuration of Pareto frontier is also obtained in Fig. 10.6 through the step 3. Although this figure shows a rough approximation of the whole Pareto frontier, it may be expected to provide a reasonable approximation to it in a neighborhood of the Pareto optimal point nearest to the aspiration level. This information of the approximation of Pareto frontier in a neighborhood of the obtained Pareto solution helps the decision maker to make trade-off analysis.

It has been observed that a method combining the satisficing trade-off method and meta-modeling is effective for supporting DM to get a final solution in a reasonable number of computational experiments. It is promising in practical problems since it has been observed that the method reduces the number of function evaluations up to less than 1/100 to 1/10 of usual methods such as MOGAs and usual aspiration level methods through several numerical experiments.

10.5 Evaluation of Meta-modeling Methods

Improvement in the design of heuristic search algorithms may sometimes be based on good theoretical ideas, but it is generally driven by the empirical testing and comparison of methods. Meta-model-based multiobjective optimization algorithms are relatively new, so, as yet, there has been little serious direct comparison of methods. In the coming years, this will become more of a focus.

10.5.1 What to Evaluate

In meta-modeling scenarios, full-cost evaluations are expensive and should be reduced. Therefore, it follows that whatever assessments of Pareto set approximation are used, these should generally be plotted against the number of full-cost function evaluations, so that it is possible to see how performance evolves over ‘time’. In the case of multiobjective optimization, this is often overlooked, because of the desire to show Pareto front approximations. For two-objective problems, the use of snapshots of Pareto fronts can be informative, but for general dimension, plotting the value of an indicator, such as the hypervolume, against full-cost evaluation number (averaged over a number of runs) is a more advisable approach.

An alternative statistical method that has been used little to date, is the attainment function method of comparing two optimizers (da Fonseca *et al.*, 2001). With this, the number of full cost evaluations can be considered an additional objective. For the first-order attainment function, the method returns an overall significance result indicating whether the attainment function (which describes the probability of attaining a certain point by a certain time, for all points and times) differs significantly between the two algorithms.

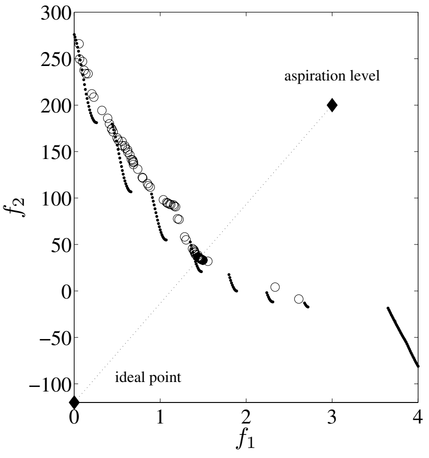


Fig. 10.6. The whole Pareto frontier with 50 sample points

In addition to the performance at optimizing a function, meta-modeling also involves its approximation. Therefore, it is sometimes desirable to show the time evolution of the accuracy of the model over all or some region of the design space.

In (Emmerich *et al.*, 2006), the accuracy of the model, as seen by the evolutionary algorithm optimizer was measured. This was done by computing the precision and recall of the pre-screening methods used in terms of being able to correctly rank solutions (rather than get their absolute evaluation correct). In a similar ilk, a number of measures of model quality that are based on evaluating the model's utility at making the 'correct' selection of individuals within an EA, were proposed in (Hüsken *et al.*, 2005).

Evaluation of interactive optimization and decision-making methods is even more of a challenge, and is dealt with in Chapter 7 of this book.

10.5.2 Adversaries

Assessment of progress in meta-modeling would be facilitated by using common, simple adversary algorithms or methods to compare against. Perhaps the simplest adversary is the random search. It is very interesting to compare with random search (as done in (Knowles, 2006) and see also (Hughes, 2006)) because it is not necessarily trivial to outperform the algorithm when the number of evaluations is small, and depending on the function. Moreover, when approximating a higher dimensional Pareto front, random search may be better than some multiobjective EAs, such as NSGA-II. The obvious additional adversary is the algorithm being proposed, with the meta-model removed (if this is possible).

10.6 Real Applications

10.6.1 Closed-Loop Mass-Spectrometer Optimization

Mass spectrometers are analytical instruments for determining the chemical compounds present in a sample. Typically, they are used for testing a hypothesis as to whether a particular compound is present or not. When used in this way, the instrument can be configured according to standard principles and settings provided by the instrument manufacturer. However, modern biological applications aim at using mass-spectrometry to mine data without a hypothesis, i.e. to measure/detect simultaneously the hundreds of compounds contained in complex biological samples. For such applications, the mass spectrometer will not perform well in a standard configuration, so it must be optimized.

(O'Hagan *et al.*, 2006) describes experiments to optimize the configuration of a GCxGC-TOF mass spectrometer with the aim of improving its effectiveness at detecting all the metabolites (products of the metabolic system) in a human serum (blood) sample. To undertake this optimization, real experiments were conducted with the instrument in different configuration set-ups, as dictated by the optimization algorithm. Instrument configurations are specified by 15 continuous parameters. Typically, a single evaluation of an instrument configuration lasts of the order of one hour, although the throughput of the instrument (i.e. how long it takes to process the serum sample) was also one of the objectives and thus subject to some variation. The overall arrangement showing the optimization algorithm and how it is connected up to the instrument in a closed-loop is shown in Figure 10.7.

The experiments reported in (O'Hagan *et al.*, 2006) used just 320 experiments (evaluations) to increase the number of metabolites observed (primary objective) substantially, as measured by the number of peaks in the mass-spectrometer's output. Simultaneously, the signal to noise ratio and the throughput of the instrument were optimized. A plot of the evolution of the three objectives is given in Figure 10.8.

The meta-modeling algorithm used for the optimization was ParEGO (Knowles, 2006), a multiobjective version of the efficient global optimization (EGO) algorithm (Jones *et al.*, 1998). ParEGO uses a design and analysis of computer experiments (DACE) approach to model the objective function(s) (in this case, the instruments' behaviour under different operating configurations), based on an initial Latin hypercube sampling of the parameter space. Subsequently, the model is used to suggest the next experiment (set of instrumentation parameter values), such that the 'expected improvement' in the objective function(s) is maximized. The notion of expected improvement implicitly ensures that ParEGO balances exploration of new parameter combinations with exploitation and fine-tuning of parameter values that have led to good design points in previous experiments. The DACE model is updated after each fitness evaluation.

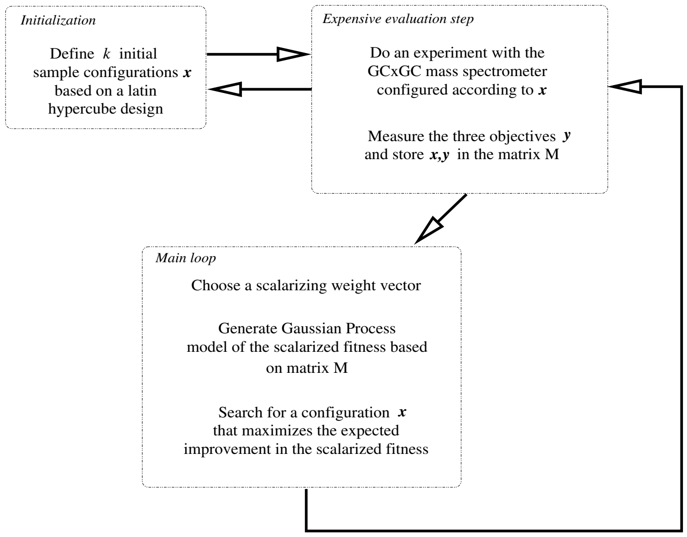

Initialization
Define k initial sample configurations x based on a latin hypercube design
Expensive evaluation step
Do an experiment with the GCxGC mass spectrometer configured according to x
Measure the three objectives y and store x,y in the matrix M
Main loop
Choose a scalarizing weight vector
Generate Gaussian Process model of the scalarized fitness based on matrix M
Search for a configuration x that maximizes the expected improvement in the scalarized fitness

Fig. 10.7. Closed-loop optimization of mass spectrometer parameters using the ParEGO algorithm

10.6.2 Application to Reinforcement of Cable-Stayed Bridges

After the big earthquake in Kobe in 1995, many in-service structures were required to improve their anti-seismic property by law regulation in Japan. However, it is very difficult for large and/or complicated bridges, such as suspension bridges, cable-stayed bridges, arch bridges and so on, to be reinforced because of impractical executing methods and complicated dynamic responses. Recently, many kinds of anti-seismic device have been developed (Honda *et al.*, 2004). It is practical in the bridge to be installed a number of small devices taking into account of strength and/or space, and to obtain the most reasonable arrangement and capacity of the devices by using optimization technique. In this problem, the form of objective function is not given explicitly in terms of design variables, but the value of the function is obtained by seismic response analysis. Since this analysis needs much cost and long time, it is strongly desirable to make the number of analyses as few as possible. To this end, radial basis function networks (RBFN) are employed in predicting the form of objective function, and genetic algorithms (GA) in searching the optimal value of the predicted objective function (Nakayama *et al.*, 2006).

The proposed method was appplied to a problem of anti-seismic improvement of a cable-stayed bridge which typifies the difficulty of reinforcement of in-service structure. In this investigation, we determine an efficient arrangement and amount of additional mass for cables to reduce the seismic response of the tower of a cable-stayed bridge (Fig. 10.9).

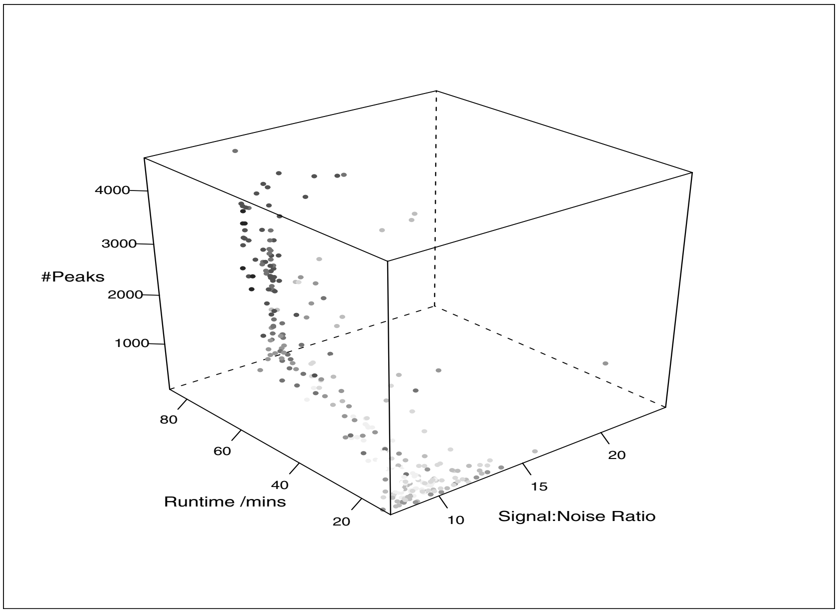


Fig. 10.8. Three-objective closed-loop optimization of GCxGC-TOF-mass spectrometry configurations for the analysis of human serum. The number of peaks and signal:noise ratio are maximized; the runtime of the mass spectrometer is minimized. The shading of the points represents the experiment number. Darker circles represent later experiments, and the six back dots represent replications of the same chosen final configuration. The number of peaks has risen to over 3000, whilst sufficient signal:noise has been maintained and runtime kept down to around 60 minutes.

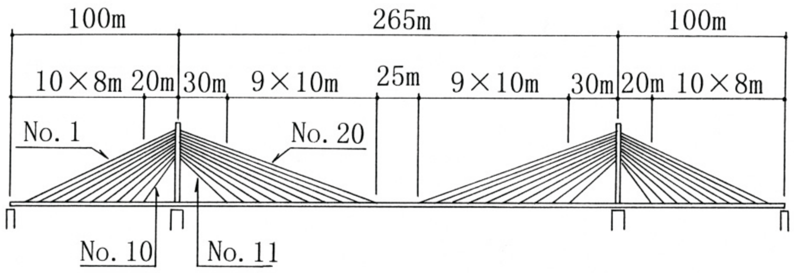


Fig. 10.9. Cable-stayed bridge

The influence of additional mass on cables was investigated by numerical sensitivity analysis. The analytical model shown in Fig. 10.9 is a 3-span continuous and symmetrical cable-stayed bridge whose 2 20 cables are in one plane and the towers stand freely in their transverse direction. The mass must be distributed over cables uniformly to prevent them from concentrating deformation.

The seismic response of interest was the stress at the fixed end of the tower when an earthquake occurs in the transverse direction. Seismic response analysis was carried out by a spectrum method. As there are a lot of modes whose natural frequencies were close to each other, the response was evaluated by the complete quadratic combination method. The input spectrum is given in the new Specifications for Highway Bridges in Japan.

The natural frequencies of modes accompanied with the bending of the tower (natural frequency of the tower alone is 1.4Hz) range from 0.79Hz to 2.31Hz, due to coupling with the cables.

As mentioned above, the seismic response of the tower can be controlled by additional mass to cables, but each cable influences other ones in a complex way. Thus, the most effective distribution of additional mass must be decided by optimization.

10.6.3 Case 1

The objective is to minimize the bending moment M at the base of tower. The variables are ratios of additional mass and mass of cables. The number of variables is 20. The lower bound and upper bound of each variable are 0.0, and 1.0, respectively. For comparison, we applied a quasi-Newton method based on approximated differentials as an existing method. We made five trials with different initial points in order to obtain a global optimum.

In applying our proposed method, we used BLX-α as a genetic algorithm which is observed to be effective for continuous variables. The population is 10, and the number of generation is 200. We set $\lambda = 0.01$, and decided the value of width r of Gaussian by the simple estimate given by (10.17).

We started the iteration with 60 sample points. The first 20 sample points are generated randomly with one of variables fixed at the upper bound 1 by turns; the next 20s are generated similarly with one of variables fixed at the lower bound 0 by turns; the last 20s similary with one of variables fixed at the mid-value 0.5 by turns. The parameters for convergence are $C_x^0 = 20$, $C_f^0 = 20$ and $l_0 = 0.1$.

The result is shown in Table 10.1. It is seen that the proposed method can find out fairly good solutions within 1/10 or less times of analysis than the conventional optimization.

10.6.4 Case 2

Now, we take the number of cables to be added with masses, N, as another objective function in addition to the bending moment M. Namely, our objective function is

$$F = (M/M_0) + \alpha(N/N_0) \tag{10.21}$$

where α is a parameter for trade-off between the first term and the second one. M_0 andN_0 are used for normalization of the bending moment and the

Table 10.1. Result for Case 1

		exisiting method	RBF Network best	RBF Network average
cable No.	1	0.32	0.04	0.40
	2	1.00	0.69	0.84
	3	0.49	0.18	0.51
	4	0.62	0.82	0.80
	5	0.81	0.57	0.64
	6	0.52	0.43	0.56
	7	0.49	1.00	0.39
	8	0.52	0.44	0.66
	9	0.48	0.94	0.50
	10	0.48	0.50	0.56
	11	0.50	0.45	0.47
	12	0.55	1.00	0.74
	13	0.70	0.85	0.71
	14	0.61	0.50	0.30
	15	0.61	1.00	0.58
	16	0.46	0.24	0.37
	17	0.22	0.10	0.13
	18	1.00	0.95	0.91
	19	0.98	1.00	0.94
	20	1.00	1.00	0.91
bending moment (MN·m)		50.3	54.90	63.70
#analysis		1365	150.00	124.80

number of cables, respectively. In this experiment, we set $M_0 = 147.0$MN·m and $N_0 = 20$.

In this experiment, we used a simple GA, because some of variables are discrete. The parameters for calculation are the same as in Case 1. The result is given in Table 10.2. It should be noted that the number of analysis in our proposed method is reduced to about 1/20 of the conventional method. Although the precision of solution in our method is behind the conventional method, it is sufficiently acceptable in practice.

Table 10.2. Result for Case 2

		existing method		RBF network	
		best	average	best	average
cable No.	1	0.00	0.00	0.00	0.83
	2	0.00	0.00	0.00	0.09
	3	0.00	0.00	0.00	0.00
	4	0.00	0.00	0.00	0.04
	5	0.00	0.00	0.00	0.00
	6	0.00	0.00	0.00	0.00
	7	0.00	0.00	0.00	0.00
	8	0.00	0.00	1.00	0.99
	9	0.00	0.00	0.00	0.10
	10	0.00	0.00	0.86	0.53
	11	0.00	0.00	0.00	0.00
	12	0.00	0.00	1.00	0.63
	13	0.00	0.00	0.00	0.13
	14	0.00	0.00	0.86	0.53
	15	0.00	0.00	0.00	0.00
	16	0.00	0.00	0.00	0.00
	17	0.00	0.00	0.00	0.00
	18	0.00	0.00	0.00	0.00
	19	0.71	0.74	1.00	1.00
	20	0.86	0.83	0.86	0.79
bending moment (MN·m)		62.6	62.8	67.1	69.7
#cable with additional mass		2	2	6	6.2
objective fn.		0.526	0.527	0.756	0.784
#analysis		4100	3780	199	193.3

The well known Response Surface Method (RSM, in short) is competitive with the proposed method. However, the proposed method has been observed to have advantage over RSM especially for highly nonlinear cases. On the other hand, EGO (Efficient Global Optimization) for black-box objective functions (Jones *et al.*, 1998; M.J. Sasena, 2000; Schonlau, 1997) takes time to calculate the expected improvement, while it is rather simple and easy to add two kinds of additional samples for global information and local information for approximation (Nakayama *et al.*, 2002, 2003).

10.7 Concluding Remarks

The increasing desire to apply optimization methods in expensive domains is driving forward research in meta-modeling. Up to now, meta-modeling has been applied mainly in continuous, low dimensional design variable spaces, and methods from design of experiments and response surfaces have been used. High-dimensional discrete spaces may also arise in applications involving expensive evaluations and this will motivate research into meta-modeling of these domains too. Research in meta-modeling for multiobjective optimization is relatively young and there is still much to do. So far, there are few standards for comparisons of methods, and little is yet known about the relative performance of different approaches. The state of the art research surveyed in this chapter is beginning to grapple with the issues of incremental learning and the trade-off between exploitation and exploration within meta-modeling. In the future, scalability of methods in variable dimension and objective space dimension will become important, as will methods capable of dealing with noise or uncertainty. Interactive meta-modeling is also likely to be investigated more thoroughly, as the number of evaluations can be further reduced by these approaches.

References

Anderson, V.L., McLean, R.A.: Design of Experiments: A Realistic Approach. Marcel Dekker, New York (1974)

Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning for control. Artificial Intelligence Review 11(1), 75–113 (1997)

Breiman, L.: Classification and Regression Trees. Chapman and Hall, Boca Raton (1984)

Brown, G., Wyatt, J.L., Tiňo, P.: Managing diversity in regression ensembles. The Journal of Machine Learning Research 6, 1621–1650 (2005)

Schölkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2002)

Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. Advances in Neural Information Processing Systems 13, 409–415 (2001)

Chafekar, D., Shi, L., Rasheed, K., Xuan, J.: Multiobjective ga optimization using reduced models. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 35(2), 261–265 (2005)

Chapelle, O., Vapnik, V., Weston, J.: Transductive inference for estimating values of functions. Advances in Neural Information Processing Systems 12, 421–427 (1999)

Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. Journal of Artificial Intelligence Research 4, 129–145 (1996)

Corne, D.W., Oates, M.J., Kell, D.B.: On fitness distributions and expected fitness gain of mutation rates in parallel evolutionary algorithms. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 132–141. Springer, Heidelberg (2002)

Cortes, C., Vapnik, V.: Support vector networks. Machine Learning 20, 273–297 (1995)

Cristianini, N., Shawe-Tylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods. Cambridge University Press, Cambridge (2000)

Grunert da Fonseca, V., Fonseca, C.M., Hall, A.O.: Inferential Performance Assessment of Stochastic Optimisers and the Attainment Function. In: Zitzler, E., Deb, K., Thiele, L., Coello Coello, C.A., Corne, D.W. (eds.) EMO 2001. LNCS, vol. 1993, pp. 213–225. Springer, Heidelberg (2001)

Emmerich, M.T.M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 62–76. Springer, Heidelberg (2005)

Emmerich, M., Giannakoglou, K., Naujoks, B.: Single-and Multi-objective Evolutionary Optimization Assisted by Gaussian Random Field Metamodels. IEEE Transactions on Evolutionary Computation 10(4), 421–439 (2006)

English, T.M.: Optimization is easy and learning is hard in the typical function. In: Proceedings of the 2000 Congress on Evolutionary Computation (CEC00), pp. 924–931. IEEE Computer Society Press, Piscataway (2000)

Erenguc, S.S., Koehler, G.J.: Survey of mathematical programming models and experimental results for linear discriminant analysis. Managerial and Decision Economics 11, 215–225 (1990)

Eyheramendy, S., Lewis, D., Madigan, D.: On the naive Bayes model for text categorization. In: Proceedings Artificial Intelligence & Statistics 2003 (2003)

Fieldsend, J.E., Everson, R.M.: Multi-objective Optimisation in the Presence of Uncertainty. In: 2005 IEEE Congress on Evolutionary Computation (CEC'2005), Edinburgh, Scotland, September 2005, vol. 1, pp. 243–250. IEEE Computer Society Press, Los Alamitos (2005)

Fonseca, C.M., Fleming, P.J.: Genetic algorithms for multi-objective optimization: Formulation, discussion and generalization. In: Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 416–426 (1993)

Freed, N., Glover, F.: Simple but powerful goal programming models for discriminant problems. European J. of Operational Research 7, 44–60 (1981)

Gaspar-Cunha, A., Vieira, A.: A multi-objective evolutionary algorithm using neural networks to approximate fitness evaluations. International Journal of Computers, Systems, and Signals (2004)

Hamza, K., Saitou, K.: Vehicle crashworthiness design via a surrogate model ensemble and a co-evolutionary genetic algorithm. In: Proc. of IDETC/CIE 2005 ASME 2005 International Design Engineering Technical Conference, California, USA (2005)

Honda, M., Morishita, K., Inoue, K., Hirai, J.: Improvement of anti-seismic capacity with damper braces for bridges. In: Proceedings of the Seventh International Conference on Motion and Vibration Control (2004)

Huang, D., Allen, T.T., Notz, W.I., Zeng, N.: Global Optimization of Stochastic Black-Box Systems via Sequential Kriging Meta-Models. Journal of Global Optimization 34(3), 441–466 (2006)

Hughes, E.J.: Evolutionary Many-Objective Optimisation: Many Once or One Many? In: 2005 IEEE Congress on Evolutionary Computation (CEC'2005), vol. 1, pp. 222–227. IEEE Computer Society Press, Los Alamitos (2005)

Hughes, E.J.: Multi-Objective Equivalent Random Search. In: Runarsson, T.P., Beyer, H.-G., Burke, E.K., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 463–472. Springer, Heidelberg (2006)

Hüsken, M., Jin, Y., Sendhoff, B.: Structure optimization of neural networks for evolutionary design optimization. Soft Computing 9(1), 21–28 (2005)

Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft Computing - A Fusion of Foundations, Methodologies and Applications 9(1), 3–12 (2005)

Jin, Y., Sendhoff, B.: Reducing fitness evaluations using clustering techniques and neural network ensembles. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3102, pp. 688–699. Springer, Heidelberg (2004)

Jin, Y., Olhofer, M., Sendhoff, B.: Managing approximate models in evolutionary algorithms design optimization. In: Proceedings of the 2001 Congress on Evolutionary Computation, CEC2001, pp. 592–599 (2001)

Jones, D., Schonlau, M., Welch, W.: Efficient global optimization of expensive black-box functions. Journal of Global Optimization 13, 455–492 (1998)

Joslin, D., Dragovich, J., Vo, H., Terada, J.: Opportunistic fitness evaluation in a genetic algorithm for civil engineering design optimization. In: Proceedings of the Congress on Evolutionary Computation (CEC 2006), pp. 2904–2911. IEEE Computer Society Press, Los Alamitos (2006)

Jourdan, L., Corne, D.W., Savic, D.A., Walters, G.A.: Preliminary Investigation of the 'Learnable Evolution Model' for Faster/Better Multiobjective Water Systems Design. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 841–855. Springer, Heidelberg (2005)

Keane, A.J.: Statistical improvement criteria for use in multiobjective design optimization. AIAA Journal 44(4), 879–891 (2006)

Knowles, J.: ParEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. IEEE Transactions on Evolutionary Computation 10(1), 50–66 (2006)

Langdon, W.B., Poli, R.: Foundations of Genetic Programming. Springer, Heidelberg (2001)

Larranaga, P., Lozano, J.A.: Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation. Kluwer Academic Publishers, Dordrecht (2001)

Laumanns, M., Očenášek, J.: Bayesian optimization algorithms for multi-objective optimization. In: Guervós, J.J.M., Adamidis, P.A., Beyer, H.-G., Fernández-Villacañas, J.-L., Schwefel, H.-P. (eds.) PPSN 2002. LNCS, vol. 2439, pp. 298–307. Springer, Heidelberg (2002)

Michalski, R.: Learnable Evolution Model: Evolutionary Processes Guided by Machine Learning. Machine Learning 38(1), 9–40 (2000)

Sasena, M.J., Papalambros, P.Y., Goovaerts, P.: Metamodeling sample criteria in a global optimization framework. In: 8th AIAA/NASA/USAF/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, AIAA-2000-4921 (2000)

Myers, R.H., Montgomery, D.C.: Response Surface Methodology: Process and Product Optimization using Designed Experiments. Wiley, Chichester (1995)

Nain, P., Deb, K.: A computationally effective multi-objective search and optimization technique using coarse-to-fine grain modeling. Kangal Report 2002005 (2002)

Nakayama, H., Sawaragi, Y.: Satisficing trade-off method for multi-objective programming. In: Grauer, M., Wierzbicki, A. (eds.) Interactive Decision Analysis, pp. 113–122. Springer, Heidelberg (1984)

Nakayama, H., Yun, Y.: Generating support vector machines using multiobjective optimization and goal programming. In: Jin, Y. (ed.) Multi-objective Machine Learning, pp. 173–198. Springer, Heidelberg (2006a)

Nakayama, H., Yun, Y.: Support vector regression based on goal programming and multi-objective programming. IEEE World Congress on Computational Intelligence, CD-ROM, Paper ID: 1536 (2006b)

Nakayama, H., Arakawa, M., Sasaki, R.: Simulation based optimization for unknown objective functions. Optimization and Engineering 3, 201–214 (2002)

Nakayama, H., Arakawa, M., Washino, K.: Optimization for black-box objective functions. In: Tseveendorj, I., Pardalos, P.M., Enkhbat, R. (eds.) Optimization and Optimal Control, pp. 185–210. World Scientific, Singapore (2003)

Nakayama, H., Inoue, K., Yoshimori, Y.: Approximate optimization using computational intelligence and its application to reinforcement of cable-stayed bridges. In: Zha, X.F., Howlett, R.J. (eds.) Integrated Intelligent Systems for Engineering Design, pp. 289–304. IOS Press, Amsterdam (2006)

O'Hagan, S., Dunn, W.B., Knowles, J.D., Broadhurst, D., Williams, R., Ashworth, J.J., Cameron, M., Kell, D.B.: Closed-Loop, Multiobjective Optimization of Two-Dimensional Gas Chromatography/Mass Spectrometry for Serum Metabolomics. Analytical Chemistry 79(2), 464–476 (2006)

Orr, M.: Introduction to radial basis function networks. Centre for Cognitive Science, University of Edinburgh (1996), `http://www.cns.ed.ac.uk/people/mark.html`

Ray, T., Smith, W.: Surrogate assisted evolutionary algorithm for multiobjective optimization. In: 47th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, pp. 1–8 (2006)

Robins, A.: Maintaining stability during new learning in neural networks. In: IEEE International Conference on Systems, Man, and Cybernetics, 1997, 'Computational Cybernetics and Simulation', vol. 4, pp. 3013–3018 (1997)

Schölkopf, B., Smola, A.: New support vector algorithms. Technical Report NC2-TR-1998-031, NeuroCOLT2 Technical report Series (1998)

Schonlau, M.: Computer Experiments and Global Optimization. Ph.D. thesis, Univ.of Waterloo, Ontario, Canada (1997)

Schwaighofer, A., Tresp, V.: Transductive and Inductive Methods for Approximate Gaussian Process Regression. Advances in Neural Information Processing Systems 15, 953–960 (2003)

Vapnik, V.N.: Statistical Learning Theory. John Wiley & Sons, Chichester (1998)

Voutchkov, I., Keane, A.J.: Multiobjective optimization using surrogates. Presented at Adaptive Computing in Design and Manufacture (ACDM 06), Bristol, UK (2006)

Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1, 67–82 (1997)

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C.M., Grunert da Fonseca, V.: Performance Assessment of Multiobjective Optimizers: An Analysis and Review. IEEE Transactions on Evolutionary Computation 7(2), 117–132 (2003)