Chapter 1
On-line Metamodel-Assisted Optimization with Mixed Variables

Rajan Filomeno Coelho, Manuel Herrera, Manyu Xiao and Weihong Zhang

Abstract The optimization of complex civil engineering structures remains a major scientific challenge, mostly because of the high number of calls to the finite element analysis required by the complete design process. To achieve a significant reduction of this computational effort, a popular approach consists in substituting the high-fidelity simulation by a lower-fidelity regression model, also called a metamodel. However, most metamodels (like kriging, radial basis functions, etc.) focus on continuous variables, thereby neglecting the large amount of problems characterized by discrete, integer, or categorical data. Therefore, in this chapter, a complete metamodel-assisted optimization procedure is proposed to deal with mixed variables. The methodology includes a multi-objective evolutionary algorithm and a multiple kernel regression model, both adapted to mixed data, as well as an efficient on-line enrichment of the metamodel during the optimization. A structural benchmark test case illustrates the proposed approach, followed by a critical discussion about the generalization of the concepts introduced in this chapter for metamodel-assisted optimization.

R. Filomeno Coelho (✉)
Building, Architecture & Town Planning (BATir) Department,
Brussels School of Engineering, Université libre de Bruxelles (ULB),
Avenue F.D. Roosevelt, 50 (CP 194/2), 1050 Brussels, Belgium
e-mail: rfilomen@ulb.ac.be

M. Herrera
Department of Civil and Environmental Engineering, Skempton Building,
South Kensington Campus Imperial College London, SW7 2AZ London, UK
e-mail: a.herrera-fernandez@imperial.ac.uk

M. Xiao
Department of Applied Mathematics, Northwestern Polytechnical University,
127 Youyi West Road, Xi'an, Shaanxi 710072, People's Republic of China
e-mail: manyuxiao@nwpu.edu.cn

W. Zhang
Engineering Simulation and Aerospace Computing, School of Mechanical Engineering,
Northwestern Polytechnical University, 127 Youyi West Road, Xi'an, Shaanxi 710072,
People's Republic of China
e-mail: zhangwh@nwpu.edu.cn

J. Magalhães-Mendes and D. Greiner (eds.), *Evolutionary Algorithms and Metaheuristics in Civil Engineering and Construction Management*,
Computational Methods in Applied Sciences 39, DOI 10.1007/978-3-319-20406-2_1

Fig. 1.1 Nijmegen city bridge "De Oversteek" during its construction (September 2013)

Keywords Genetic algorithms · Mixed variables · Categorical variables · Multiple kernel regression · Support vector regression

1.1 Mixed Variables in Civil Engineering

The need for stronger, safer, and greener structures built at shortened delays and at a competitive cost pushes the art of construction to favor elegant lightweight structures, as the city bridge of Nijmegen (The Netherlands) designed by L. Ney and C. Poulissen (see Fig. 1.1),[1] for which an optimization study was conducted by the first author of this chapter.

Nowadays, the design of such remarkable structures is greatly improved by numerical methods, in particular through an efficient combination of evolutionary algorithms to explore the design space, and general regression models (also called metamodels) to avoid a systematic call to the structural finite element analysis [11, 15]. Seen through a simulation-based perspective, in a classical structural optimization process the simulations are integrated within an optimization iterative loop, as depicted in Fig. 1.2. The scientific and technical challenge resides in carrying out a good balance between the use of the high-fidelity (i.e. finite elements) and the low-fidelity (i.e. regression) models, in order to find the best compromise between accuracy and CPU time.

Before such numerical considerations, the first step requires the definition of relevant design variables. As a general rule, the parameterization of civil engineering structures (bridges, dams, buildings, etc.) involves several types of variables representing respectively the geometry (sizing of the elements, overall shape and topology), the materials used, and the boundary conditions (supports). Mathematically, these variables can be classified as follows:

[1]Credits: http://commons.wikimedia.org/.

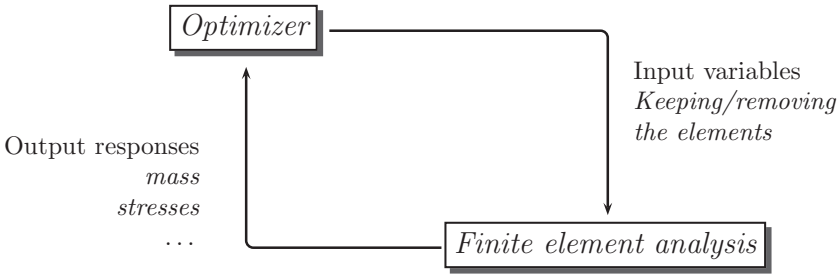

Fig. 1.2 Classical structural optimization process

- *continuous* variables are real numbers defined within an interval $[x^{\min}, x^{\max}] \subseteq \mathbb{R}$;
- *discrete* variables are continuous variables only available among a discrete set. For instance, the cross-section area of a beam profile is an intrinsically continuous parameter defined in square meters, but whose availability might be limited to a discrete sampling taken from a catalog of beam profiles $\{A^{(1)}, A^{(2)}, \ldots, A^{(n)}\}$. This means however that values not available in catalogs can still be computed if necessary, and then rounded off;
- *integer* variables are strictly defined in $\mathbb{N}$. Contrary to discrete variables, intermediate values have no physical meaning. For example, the number of holes drilled in a plate to reduce its weight can be equal to 3 or 4, but any value between these two integer numbers does not correspond to a physical design;
- *categorical* variables represent the remaining non-numerical parameters. Although largely neglected in the optimization literature, they have a huge practical interest in civil engineering, since they can represent for instance the choice of a material ($\{steel, aluminum, \ldots\}$), the type of connection in the assembly of frames ($\{rigid,\ semi\text{-}rigid,\ articulated\}$), the shape of a cross-section ($\{$ O , ■ , I ,... $\}$), etc. If they are endowed with a predefined ranking (e.g. a size $\{S, M, L, XL\}$, a qualitative appreciation $\{weak, normal, strong\}$), they are referred to as *ordinal* variables; otherwise, without intrinsic ordering, they are purely *nominal*.

As mentioned above for the case of discrete variables, a common practice consists in treating non-continuous variables as real numbers, performing an approximation and/or optimization task, and eventually rounding off the solution. The very simple example below will show the danger of using such techniques to deal with intrinsically non-continuous problems.

Let us consider the following three-variable minimization problem:

$$\min_{x_1,x_2,x_3} \quad f(x_1,x_2,x_3) \equiv x_1 + x_2 + x_3$$
$$\text{subject to:} \begin{cases} g(x_1,x_2,x_3) \equiv x_1 + x_2 + x_3 - 10 \geq 0 \\ h(x_1,x_2,x_3) \equiv x_1 - x_2 = 0 \\ x_1, x_2, x_3 \in \{1,2,3,4,5\} \end{cases} \tag{1.1}$$

Figure 1.3 shows the design space and a few solutions of problem (1). On one hand, $\mathbf{x}^* = (3, 3, 4)$ is a discrete optimum characterized by an objective function value equal to $f(\mathbf{x}^*) = 10$. On the other hand, $\mathbf{x}^{**} = (3.6, 3.6, 2.8)$ is a continuous solution of problem of (1.1) without considering the last constraint on the discrete nature of the variables, and characterized by the same value at the optimum $f(\mathbf{x}^{**}) = 10$; however, by rounding off a posteriori the corresponding design variables to the nearest integer values $\mathbf{x}^{**}_{\text{rounded off}}(4, 4, 3)$, all constraints are now satisfied but the optimality has been lost ($f(\mathbf{x}^{**}_{\text{rounded off}}) = 11$).

This simple application demonstrates the need for considering the discrete, integer, or categorical nature of the variables directly in the optimization phase. Of course, *mixed-integer programming*, implying the relaxation of discrete/integer variables [17], is a powerful approach to handle such problems, and is widely used in combinatorial optimization. Nevertheless, its efficiency depends largely on the mathematical properties of the functions involved (linearity, convexity, etc.), which are usually not guaranteed in civil engineering problems. Besides, recent works on the symmetry of optimal designs in sizing and topology optimization proved that the solution of symmetric problems is always symmetric with continuous variables, but might be asymmetric when discrete variables are involved [19]. All these observations should convince the reader of the importance of treating mixed variables appropriately.

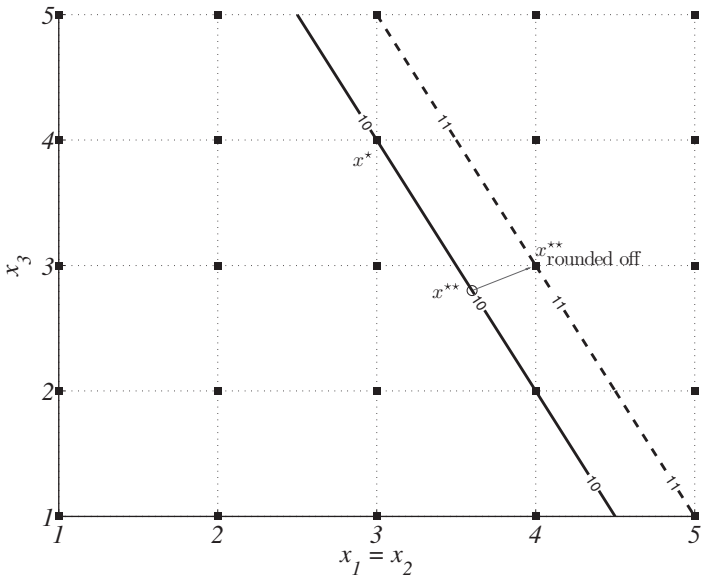

Fig. 1.3 Three-variable example showing the danger of using continuous optimization followed by rounding off the solution for discrete or integer optimization. The design space is depicted in 2D, thereby satisfying automatically the equality constraint $x_1 = x_2$. The black squares represent the discrete design space. While x^* constitutes a valid feasible solution, $x^{**}_{\text{rounded off}}$ obtained by rounding off a valid continuous solution loses the optimality criterion

In this chapter, the emphasis will be put on the simultaneous presence of continuous and categorical variables. After a discussion on the representation of mixed variables (Sect. 1.2), a multiple kernel regression metamodel is described (Sect. 1.3), followed by an on-line metamodel-assisted optimization procedure applied to the design of a rigid frame (Sect. 1.4), and by the conclusions (Sect. 1.5).

1.2 Representation of Mixed Variables

Before diving into the regression and optimization procedures, it is worth considering how mixed data can be mingled together.

Being familiar with genetic algorithms, or simply with computer science in general, the first idea to deal with different types of data is to adopt a *binary coding*. According to the range of the available data (or precision for real numbers), a binary conversion can be operated as shown in Fig. 1.4.

However, it is often preferable to propose a representation closer to the variable types [13]. In this case, a real-number array can be obtained, as illustrated in Fig. 1.5.

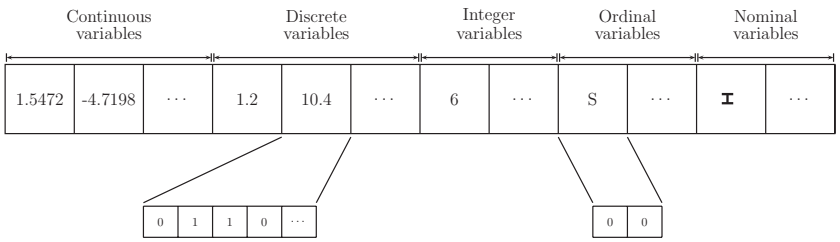

Fig. 1.4 Mixed data converted into an array of binary digits

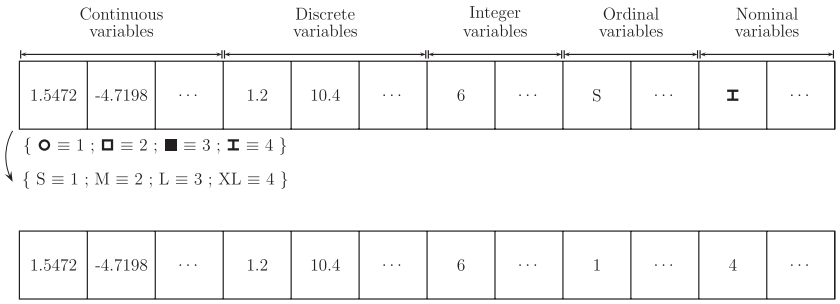

Fig. 1.5 Mixed data converted into an array of real numbers

Table 1.1 Mapping of categorical attributes onto a real number (a binary vector, respectively)

Nominal variable	Mapping 1		Mapping 2	
	Real	Binary	Real	Binary
○	1	(0,0)	2	(0,1)
□	2	(0,1)	4	(1,1)
■	3	(1,0)	1	(0,0)
工	4	(1,1)	3	(1,0)

Both mapping operators are equally valid, but might lead to a different behavior of the approximation/optimization tools without specific care

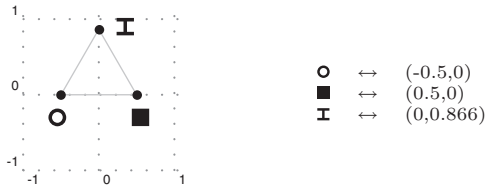

Fig. 1.6 Representation of a nominal variable with three attributes in the 2D space by a standard regular simplex

However, both approaches require an arbitrary mapping of the categorical variables. As shown in Table 1.1, different mappings lead to different conversions, which—without specific care—might lead to a different behavior of the approximation/optimization tools, viz. to a different output prediction (approximation) or optimal solution (optimization).

Therefore, to alleviate this shortcoming, the basic idea of the *regular simplex* method is to assume that any pair of levels of a categorical variable are separated by the same distance (see Fig. 1.6). To achieve this, each level of an n-level variable is associated with a distinct vertex of a regular simplex in $(n-1)$ dimensional space [12, 14]. For simplicity, the Euclidean distance between levels is assumed to be 1. For example, if x^{categ} can take values in a set of $n_{attr} = 3$ possible attributes $\{○ ; ■ ; 工\}$, these attributes can be drawn in a $(n_{attr} - 1)$ space in such a way that each attribute is converted to the vertex coordinates of a standard regular simplex. By construction, all potential values are thus equally distant [4].

Numerical validation performed on several analytical benchmark test cases [8] showed that a real-simplex mapping (i.e. real conversion for continuous, discrete, integer, and ordinal variables, and regular simplex for the nominal variables) is a sound and competitive conversion technique.

The next sections will describe how such mapping techniques can be seamlessly integrated within approximation and optimization procedures.

1.3 Approximation by On-line Multiple Kernel Regression

1.3.1 Multiple Kernel Regression and Support Vector Regression (SVR)

As discussed in the introduction, a reliable and efficient structural optimization process should harmoniously combine an optimization algorithm with an approximate model in order to reach a feasible solution at an affordable computational time. This is the purpose of *metamodel-assisted optimization*, also referred to as *surrogate-based optimization*.

The main steps to devise a surrogate-based optimization algorithm are the following [6]:

1. the variables to be optimized are selected, often due to their importance, as determined by preliminary experiments;
2. a series of initial designs are analyzed by means of the high-fidelity simulation (in structural design, the simulation usually consists in a finite element analysis). The set of designs are selected according to a pre-defined sampling scheme, for example through Latin hypercube sampling [22];
3. a metamodel (e.g. kriging, radial basis function networks, artificial neural networks, polynomial response surfaces, etc.) is used to build a low-fidelity model;
4. the optimization is performed using the low-fidelity model;
5. the results of the optimization are post-processed in order to keep the best point(s). According to the infill criterion selected, new designs are assessed by the high-fidelity simulation, and the corresponding results are added to the existing database to improve the reliability of the metamodel. The process can go back to step 3 and the optimization-approximation cycling is repeated until the stopping criterion is reached (convergence or maximum number of cycles attained).

An off-line optimization process would stop just after step 4, the best solution(s) found being re-assessed by the high-fidelity simulation for verification purposes. In the approach advocated by the authors, an *on-line learning* process is proposed (see step 5). It consists in updating the metamodel both by adding new information and by updating the regression parameters. Thus, this on-line model is updated by the results of an optimization process adapting itself to work better in areas close to the optimum (or optima in multi-objective optimization) found at each cycle (see Fig. 1.7).

First, the metamodel for mixed variables must be presented. In this study, a *kernel*-based approach is followed. Common kernel-based learning methods [20] use an implicit mapping of the input data into a high dimensional feature space defined by a kernel function, i.e. a function K returning the inner product $\langle \phi(\mathbf{x}), \phi(\mathbf{x}') \rangle$ between the images of two data points $\mathbf{x}$, $\mathbf{x}'$ in the feature space (see Table 1.2). The choice of the map ϕ aims at converting the nonlinear relations into

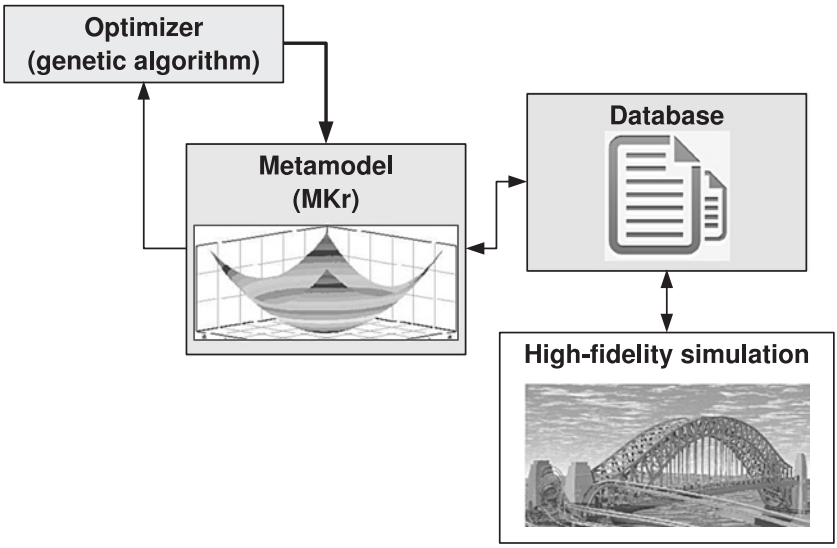

Fig. 1.7 On-line optimization with high-fidelity (= structural finite element analysis) and low-fidelity (= metamodel by multiple kernel regression) simulations

Table 1.2 Short list of some common kernel functions

Name	Expression
Gaussian	$K(\mathbf{x},\mathbf{x}') = \exp\left(-\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{2\sigma^2}\right)$
ANOVA	$K(\mathbf{x},\mathbf{x}') = \sum \exp\left(-\sigma\left(\mathbf{x}^k - \mathbf{x}'^k\right)^2\right)^d$
Linear	$K(\mathbf{x},\mathbf{x}') = \mathbf{x}^T\mathbf{x}' + c$
Polynomial	$K(\mathbf{x},\mathbf{x}') = (\alpha\mathbf{x}^T\mathbf{x}' + c)^d$
Rational quadratic	$K(\mathbf{x},\mathbf{x}') = 1 - \frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\|\mathbf{x}-\mathbf{x}'\|^2+c}$

linear ones. The learning then takes place in the feature space and the learning algorithm can be expressed so that the data points only appear inside dot products with other points. This is often referred to as the "kernel trick" [21].

The use of kernel methods is well adapted to the problem of data integration as it enables multiple types of data to be converted into a common usable format, using one of the representations mentioned in Sect. 1.2. These can be combined eventually with a weighted summation and used as training data for a classical support vector regression (SVR) scheme.

A detailed explanation of SVR is outside the scope of this chapter, but its main principles are summarized hereafter. The key characteristic of SVR is that it allows to specify a margin ε within which we are ready to accept errors in the sample data without affecting the prediction quality. The SVR predictor is defined by the points lying outside the region formed by the band of width $\pm\varepsilon$ around the regression (see Eq. 1.2). Those vectors are the so-called *support vectors*.

$$\hat{f}(\mathbf{x}) = \langle \mathbf{w}, \phi(\mathbf{x}) \rangle + b \tag{1.2}$$

The goal is to find a function $\hat{f}(x)$ that deviates at most by ε from the observed output y_i for the regression based on the training data, and minimizing at the same time the model complexity (see Eq. 1.3):

$$\begin{aligned} &\min_{\mathbf{w},b} \ \frac{1}{2}||\mathbf{w}||^2 \\ &\text{subject to: } y_i - \langle \mathbf{w}, \phi(x_i) \rangle - b \leq \varepsilon \\ &\qquad\qquad \langle \mathbf{w}, \phi(x_i) \rangle + b - y_i \leq \varepsilon \end{aligned} \tag{1.3}$$

The constraints in Eq. 1.3 assume that $\hat{f}(\mathbf{x})$ exists for all y_i with precision $\pm\varepsilon$. Nevertheless, the solution may actually not exist or it would be possible to achieve better predictions if outliers were allowed. Consequently, *slack variables* ξ^+ and ξ^- are introduced:

$$\xi^+ = \hat{f}(x_i) - y(x_i) > \varepsilon \tag{1.4}$$

$$\xi^- = y(x_i) - \hat{f}(x_i) > \varepsilon \tag{1.5}$$

and the objective function and constraints for SVR are

$$\begin{aligned} &\min_{\mathbf{w},b} \ \frac{1}{2}||\mathbf{w}||^2 + C\frac{1}{n}\sum_{i=1}^{n}(\xi_i^+ + \xi_i^-) \\ &\text{subject to: } \ y_i - \langle \mathbf{w}, \phi(x_i) \rangle - b \leq \varepsilon + \xi_i^+, \\ &\qquad\qquad \langle \mathbf{w}, \phi(x_i) \rangle + b - y_i \leq \varepsilon + \xi_i^-, \\ &\qquad\qquad \xi_i^+, \xi_i^- \geq 0 \ i = 1, \ldots, n \end{aligned} \tag{1.6}$$

where n is the number of training patterns and C is a trade-off parameter between model complexity and training error. Additionally, ξ^+ and ξ^- are slack variables for exceeding the target value by more than ε and for being below the target value by more than ε, respectively. This method of tolerating errors is known as ε *-insensitive*.

The usual SVR implementations use a single mapping function ϕ, and hence a single kernel function K. If a data set has a locally varying distribution, using a single kernel may not catch up correctly the varying distribution. Kernel fusion can help to deal with this problem [2]. Recent applications [10] and developments based on support vector machines have shown that using multiple kernels instead of a single one can enhance interpretation of the decision function and improve classifier performance. By the use of different kernels we can address problems from different data nature too. It will reveal beneficial in the perspective of mixed variable programming [1, 7].

We will adopt the weighted sum fusion with the following mapping functions:

$$\Phi(\mathbf{x}) = [\sqrt{\mu_1}\phi_1(\mathbf{x}), \sqrt{\mu_2}\phi_2(\mathbf{x}), \ldots, \sqrt{\mu_M}\phi_M(\mathbf{x})] \tag{1.7}$$

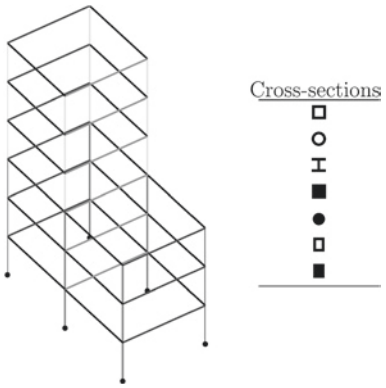

Fig. 1.8 Example of the design of a rigid frame

where $\mu_1, \mu_2, \ldots, \mu_M$ are weights of component functions. Now, the regression problem includes the optimization of two parts. One part is the regression hyperplane $f(\mathbf{x})$ and the other part is the weight vector $\mathbf{m} = [\mu_1, \mu_2, \ldots, \mu_M]$. The idea is to address these two parts of the optimization process in one step, based on the parametric dependence idea.

The resulting multi-kernel, expressed by Eq. 1.8:

$$\begin{aligned}\tilde{K}(x_i, x_j) &= \langle \Phi(x_i), \Phi(x_j) \rangle \\ &= \mu_1 \langle \phi_1(x_i), \phi_1(x_j) \rangle + \mu_2 \langle \phi_2(x_i), \phi_2(x_j) \rangle \\ &\quad + \cdots + \mu_M \langle \phi_M(x_i), \phi_M(x_j) \rangle \\ &= \mu_1 K_1(x_i, x_j) + \mu_2 K_2(x_i, x_j) + \cdots + \mu_M K_M(x_i, x_j) \\ &= \sum_{s=1}^{M} \mu_s K_s(x_i, x_j)\end{aligned} \tag{1.8}$$

is the weighted sum of M kernel functions, constituting a new kernel function. We can solve the regression hyperplane by plugging this multi-kernel into the expression of the SVR regression surface, as shown in Eq. 1.9:

$$\hat{f}(\mathbf{x}) = b + \sum_{i=1}^{n} (\alpha_i^+ - \alpha_i^-) \tilde{K}(x_i, \mathbf{x}). \tag{1.9}$$

1.3.2 On-line Multiple Kernel Regression

In metamodel-assisted optimization, the metamodel is not defined once for all, but is likely to be updated whenever new information from the high-fidelity simulation is made available. Therefore, a metamodel-updating on-line technique is mandatory.

Most of the kernel-based algorithms cannot be used to operate on-line due to a number of difficulties such as time and memory complexities (because of the

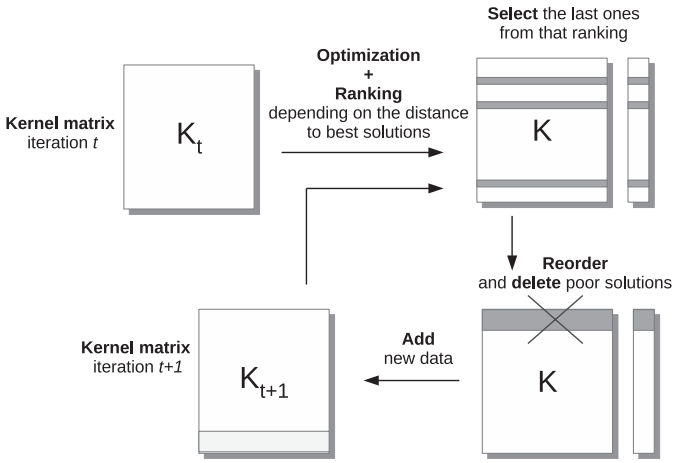

Fig. 1.9 Windowing strategy

growing kernel matrix), and due to the need to avoid over-fitting. However, a few recent experiments were successfully conducted in this sense [9]. For example, a kernel-based recursive least-squares algorithm implementing a fixed size "sliding-window" technique was proposed in [24].

In this chapter, an extension of this methodology to the use of multiple kernels for mixed variables is proposed. Moreover, the windowing process is embedded here into an optimization process. Some additional improvements have thus been included, the main one being to discard the data far from the optimum at each iteration of the process (see Fig. 1.9).

Practically, in the *sliding window* approach proposed here, only the last N pairs of the stream are selected to perform the multi-kernel regression. When a new observed pair $\{x_{n+1}, y_{n+1}\}$ is obtained, the kernel matrix $K_j^{(n)}$ is first down-sized by extracting the contribution from x_{n-N} (see Eq. 1.10):

$$\breve{K}_j^{(n)} = \begin{pmatrix} K_j^{(n)}(2,2) & \cdots & K_j^{(n)}(2,N) \\ \vdots & \ddots & \vdots \\ K_j^{(n)}(N,2) & \cdots & K_j^{(n)}(N,N) \end{pmatrix} \tag{1.10}$$

and then the $K_j^{(n)}$ dimension is augmented again by importing the data input x_{n+1} to obtain the kernel expressed in Eq. 1.11.

$$K_j^{(n+1)} = \begin{pmatrix} \breve{K}_j^{(n)} & K_j(\mathbf{X}_n, x_{n+1}) \\ K_j(x_{n+1}, \mathbf{X}_n) & K_j(x_{n+1}, x_{n+1}) + \lambda \end{pmatrix} \tag{1.11}$$

where $\mathbf{X}_n = (x_{n-N+1}, \ldots, x_n)^T$ and λ is a correction factor.

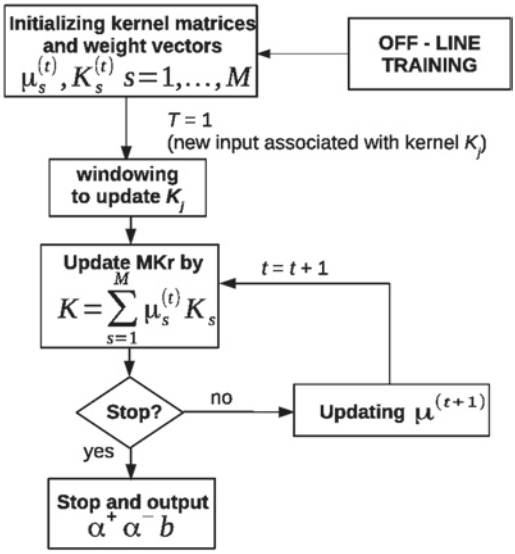

Fig. 1.10 On-line multiple kernel regression: tuning of the regression parameters

Next, the kernel matrices are summed again (see Fig. 1.10) and their weights μ are updated too. Afterwards, the weights and parameters are tuned by a multi-start trust region algorithm for the off-line part [18] (a fast version of this algorithm is included to update the values of the weights at every step of the on-line process).

In the next section, this on-line metamodeling procedure will be coupled to a multi-objective evolutionary algorithm, and applied to a structural optimization test case.

1.4 On-line Metamodel-Assisted Optimization

In order to illustrate the efficiency of on-line regression within an optimization scheme, a structural design optimization problem will be considered (see Fig. 1.8). The numerical application consists in the multi-objective design optimization of a three-dimensional rigid frame with respect to categorical and continuous variables (see Fig. 1.8 [16]). The problem is formulated as follows:

$$\begin{cases} \min\limits_{\mathbf{x}} \mathbf{f}(\mathbf{x}) = \left\{ \begin{array}{c} f_1 \equiv \text{mass} \\ \\ f_2 \equiv \log(\text{compliance}) \end{array} \right\} \\ \text{subject to: } \mathbf{x} = \{c_1, c_2, c_3, c_4, c_5, l_1, l_2, l_3, l_4, l_5\}, \\ \qquad c_i \in \{ \square\ ;\ \circ\ ;\ \text{I}\ ;\ \blacksquare\ ;\ \bullet\ ;\ \square\ ;\ \blacksquare\ \}, i = 1\ldots,5, \\ \qquad l_i \in [0.09, 0.11], i = 1\ldots,5. \end{cases} \tag{1.12}$$

Evolutionary algorithms, thanks to the flexibility of their data structure, are well adapted to deal with mixed variables [23]. The multi-objective optimizer used in this study is the second version of the Non-dominated Sorting Genetic Algorithm (NSGA-II) [3], where the probabilities of simulated binary crossover and mutation are respectively set to 0.9 and 0.5, and the distribution index for simulated binary crossover (η_c) and mutation (η_m) are respectively set to 10 and 20. Its implementation has been modified to tackle nominal variables by adapting the evolutionary operators as follows [5]:

- *crossover*: for each nominal variable and at the user-defined probability of crossover, the operation consists in swapping the values of the parents provided a randomly generated number is above 0.5;
- *mutation*: for each nominal variable and for the user-defined probability of mutation, the operation consists in changing the value of the variable randomly among the set of attributes.

The population of the evolutionary algorithm is set to 200 individuals. In the metamodel-assisted optimization, an initial database of 2,000 samples is needed. Then, at each cycle (i.e. after each optimization with the low-fidelity model), the best 60 designs (according to the NSGA-II ranking criterion) are calculated by means of the high-fidelity finite element analysis, and added to the training database to eventually update the metamodel. The numerical results show (surprisingly) a better coverage of the Pareto front than with the high-fidelity model alone, apparently due to a smoothing of the objective functions predicted by the metamodels (see Fig. 1.11), leading in this case to an improved behavior of the optimizer. Besides, these excellent results are obtained for a reduced number of calls to the finite element (FE) program: at the final (25th) generation, 3,500 FE runs are needed for the on-line process, to be compared to the 5,000 FE runs when the FE program is called systematically.

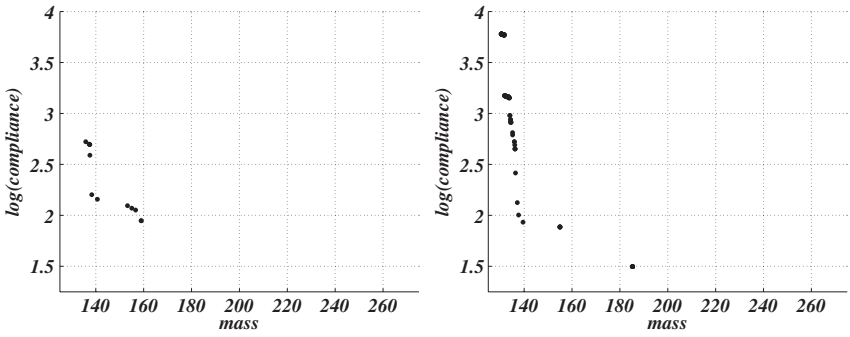

Fig. 1.11 Comparison between high-fidelity optimization process ($\Leftrightarrow$ without using a metamodel, (*left*) and metamodel-assisted on-line optimization (*right*), at the 25th iteration/cycle

1.5 Conclusions

In this chapter, mixed-variate metamodel-assisted optimization has been introduced. Based on the experience of the authors and peers in the field of evolutionary optimization with mixed variables, the reader should keep in mind the following points:

- for both approximation and optimization tasks, the inner nature of the variables should be taken into account. In particular, rounding off continuous optimal solutions to discrete values should be avoided or used with utmost care;
- categorical data should be handled by means of appropriate coding techniques like the regular simplex mapping used in classification and clustering;
- a multi-kernel function is well adapted to deal with variables of different types. Although presented here in the context of support vector regression, it is believed by the authors that the general idea could be fruitfully used in other families of metamodels (for instance with radial basis function networks, which also make use of kernels);
- an efficient on-line procedure to update the metamodels can be coupled seamlessly to an evolutionary algorithm. However, although not treated in detail in this chapter, the main issue consists in defining a proper interaction between the metamodel and the optimizer, by means of suitable *infill criteria* [6]. These criteria define which points should be assessed by the high-fidelity model and added to the database. New points can typically be the best points obtained so far, the points endowed with the highest prediction error of the metamodels (hence requiring a re-sampling to improve the accuracy of the low-fidelity model), or a combination of both paradigms.

Future prospects in this field include the handling of all kinds of variables (continuous, discrete, integer, and categorical altogether), and its validation on several benchmark test cases.

Acknowledgments This work has been supported by Innoviris (Brussels-Capital Region, Belgium) through a BB2B project entitled “Multicriteria optimization with uncertainty quantification applied to the building industry”.

The authors also acknowledge support by the Basic Project Foundation of Northwestern Polytechnical University (JC20120241), and by the National Natural Science Foundation of China (Grants No. 11302173, 51275424, and 11432011).

References

1. Abramson M, Audet C, Dennis DE Jr (2007) Filter pattern search algorithms for mixed variable constrained optimization problems. Pac J Optim 3(3):477–500
2. Christmann A, Hable R (2012) Consistency of support vector machines using additive kernels for additive models. Comput Stat Data Anal 56(4):854–873
3. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6(2):182–197

4. Filomeno Coelho R (2014) Metamodels for mixed variables based on moving least squares—Application to the structural analysis of a rigid frame. Optim Eng 15(2):311–329
5. Filomeno Coelho R, Xiao M, Guglielmetti A, Herrera M, Zhang W (2015) Investigation of three genotypes for mixed variable evolutionary optimization. In: Greiner D et al (eds) Advances in evolutionary and deterministic methods for design, optimization and control in engineering and sciences. Computational methods in applied sciences, vol 36. Springer, New York, pp 309−319
6. Forrester AIJ, Keane AJ (2009) Recent advances in surrogate-based optimization. Prog Aerosp Sci 45(1–3):50–79
7. Hemker T, Fowler KR, Farthing MW, von Stryk O (2008) A mixed-integer simulation-based optimization approach with surrogate functions in water resources management. Optim Eng 9:341–360
8. Herrera M, Guglielmetti A, Xiao M, Filomeno Coelho R (2014) Metamodel-assisted optimization based on multiple kernel regression for mixed variables. Struct Multidiscip Optim 49(6):979–991
9. Hoi SCH, Jin R, Zhao P, Yang T (2013) Online multiple kernel classification. Mach Learn 90:289–316
10. Luts J, Molenberghs G, Verbeke G, Van Huffel S, Suykens JAK (2012) A mixed effects least squares support vector machine model for classification of longitudinal data. Comput Stat Data Anal 56(3):611–628
11. Machairas V, Tsangrassoulis A, Axarli K (2014) Algorithms for optimization of building design: A review. Renew Sustain Energy Rev 31:101–112
12. McCane B, Albert MH (2008) Distance functions for categorical and mixed variables. Pattern Recogn Lett 29(7):986–993
13. Michalewicz Z (1996) Genetic Algorithms + Data Structures = Evolution Programs. Springer, New York
14. Mortier F, Robin S, Lassalvy S, Baril CP, Bar-Hen A (2006) Prediction of Euclidean distances with discrete and continuous outcomes. Multivar Anal 97(8):1799–1814
15. Nguyen A-T, Reiter S, Rigo P (2014) A review on simulation-based optimization methods applied to building performance analysis. Appl Energy 113:1043–1058
16. Papadrakakis M, Lagaros ND (2002) Reliability-based structural optimization using neural networks and Monte Carlo simulation. Comput Methods Appl Mech Eng 191:3491–3507
17. Pardalos PM, Prokopyev OA, Busygin S (2006) Continuous approaches for solving discrete optimization problems. In: Appa G, Pitsoulis L, Williams HP (eds) Handbook on modelling for discrete optimization. International Series in Operations Research & Management Science. Springer Science+Business Media B.V., pp 39–60
18. Peri D, Tinti F (2012) A multistart gradient-based algorithm with surrogate model for global optimization. Commun Appl Ind Math 3(1)
19. Richardson JN, Adriaenssens S, Bouillard Ph, Filomeno Coelho R (2013) Symmetry and asymmetry of solutions in discrete variable structural optimization. Struct Multidiscip Optim 47(5):631–643
20. Schölkopf B, Smola AJ (2001) Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge
21. Smola AJ, Schölkopf B (2004) A tutorial on support vector regression. Stat Comput 14(3):199–222
22. Swiler LP, Wyss GD (2004) A user's guide to Sandia's Latin hypercube sampling software: LHS UNIX library/standalone version. Technical report, Sandia National Laboratories, Albuquerque, New Mexico
23. Tang X, Bassir DH, Zhang W (2011) Shape, sizing optimization and material selection based on mixed variables and genetic algorithm. Optim Eng 12:111–128
24. van Vaerenbergh S, Vía J, Santamaría I (2006) A sliding-window kernel RLS algorithm and its application to nonlinear channel identification. In: IEEE international conference on acoustics, speech and signal processing—ICASSP 2006, vol 5, pp 789–792