Catalysis Today 159 (2011) 84–95

Contents lists available at ScienceDirect

Catalysis Today

journal homepage: www.elsevier.com/locate/cattod

Generator approach to evolutionary optimization of catalysts and its integration with surrogate modeling

Martin Holeňa[a,b,*], David Linke[a], Uwe Rodemerck[a]

[a] Leibniz Institute for Catalysis, Albert-Einstein-Str. 29a, D-18059 Rostock, Germany
[b] Institute of Computer Science, Academy of Sciences, Pod vodárenskou věží 2, CZ-18207 Prague, Czech Republic

ARTICLE INFO

Article history:
Available online 1 July 2010

Keywords:
Optimization of catalytic materials
Evolutionary optimization
Surrogate modeling
Artificial neural networks
Multilayer perceptron
Regression boosting

ABSTRACT

This paper presents some unpublished aspects and ongoing developments of the recently elaborated generator approach to the evolutionary optimization of catalytic materials, the purpose of which is to obtain evolutionary algorithms precisely tailored to the problem being solved. It briefly recalls the principles of the approach, and then it describes how the employed evolutionary operations reflect the specificity of the involved mixed constrained optimization tasks, and how the approach tackles checking the feasibility of large polytope systems, frequently resulting from the optimization constraints. Finally, the paper discusses the integration of the approach with surrogate modeling, paying particular attention to surrogate models enhanced with boosting. The usefulness of surrogate modeling in general and of boosted surrogate models in particular is documented on a case study with data from a high-temperature synthesis of hydrocyanic acid.

1. Introduction

The *challenge of finding an optimal catalyst for a specific reaction* leads quickly to large search spaces when high-throughput approaches are used. During the optimization of catalytic materials, catalysts with active elements, supports and dopants chosen from increasingly large pools and prepared using an increasingly broad spectrum of different preparation methods are tested. Consequently, the optimum has to be searched in a high-dimensional descriptor space. To this end, the traditional design of experiments methods are not applicable since they tend to treat uniformly the whole set of possible materials. However, high-performance catalysts are typically found only in small areas of contiguous compositions. Therefore, function optimization methods have been employed for searching optimal catalysts in the descriptor space since the late 1990s. From their point of view, the search for high-performance catalysts is a *complex optimization* task with the following features:

(i) *high dimensionality* (30–50 variables are not an exception);
(ii) *mixture* of continuous and discrete variables;
(iii) *constraints*;
(iv) *objective function*, i.e., the performance indicator of the catalyst (such as yield, conversion, or selectivity) cannot be explicitly described, its values must be *obtained empirically*.

Most common optimization methods, such as steepest descent, conjugate gradient methods or second order methods (e.g., Gauss–Newton or Levenberg–Marquardt) cannot be employed to this end. Indeed, to obtain sufficiently precise numerical estimates of gradients or second order derivatives of the empirical objective function, those methods need to evaluate the function in points some of which would have a smaller distance than is the empirical error of catalytic measurements. That is why *methods not requiring any derivatives* have been used to solve the above optimization task—both deterministic ones, in particular the simplex method and holographic strategy [1,2], and stochastic ones, such as simulated annealing [3,4], or genetic and other *evolutionary algorithms (EA)* [5–9]. Especially evolutionary algorithms have become quite popular in the search for optimal catalytic materials, mainly due to the possibility to establish a straightforward correspondence between multiple optimization paths followed by the algorithm and the way how the catalysts proposed by that algorithm are tested—namely, simultaneous testing of a whole generation of catalysts in the channels of a high-throughput reactor.

However, their application to the optimization of catalytic materials is still challenging from the implementation point of view. *Generic* implementations do not sufficiently deal with mixing continuous and discrete variables, and the low-level coding they use is hardly comprehensible for chemists. On the other hand, 10 years

* Corresponding author at: Leibniz Institute for Catalysis, Albert-Einstein-Str. 29a, D-18059 Rostock, Germany.
E-mail addresses: martin.holena@catalysis.de, martin@cs.cas.cz (M. Holeňa), david.linke@catalysis.de (D. Linke), uwe.rodemerck@catalysis.de (U. Rodemerck).

0920-5861/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.cattod.2010.05.031

of experience with *specialized EA implementations* designed specifically for catalyst optimization show that they entail another kind of problems. They are quite inflexible with respect to constraints interconnecting different descriptors, in particular constraints connecting discrete descriptors (such as qualitative composition, or the choice of a particular preparation procedure) with continuous descriptors (e.g., proportions of individual components, or reaction conditions). Whenever such constraints substantially change, the algorithm needs to be reimplemented. A possible solution to those problems is offered by the recently proposed *generator approach* [10]: to generate problem-tailored EA implementations automatically by a *program generator*, the input to which is a *formal specification* of the searched materials.

In this paper, some aspects of the generator approach relevant to the Eurocombicat conference that were not described in [10] are first dealt with. Then the paper turns to the integration of the approach with surrogate modeling, which restricts the costly and time-consuming evaluation of the empirical objective function to only some points, and evaluates some surrogate model of that function otherwise. In particular, it suggests to increase the accuracy of a surrogate model through boosting.

The paper is organized as follows. The next section explains how the generated EA tackle mixed constrained optimization, which was not explained in [10]. Then, surrogate modeling and its enhancement with boosting are discussed. Finally, our experience with surrogate modeling is documented on a case study with data from a high-temperature synthesis of HCN. The paper does not repeat explanations of the generator approach presented in [10]. Therefore, it is easier to understand for a reader who has looked at [10] first.

2. Generator approach to optimization of catalytic materials

Due to the problem-dependency of EA inputs, it is quite difficult to use general EA software, such as Matlab's Genetic Algorithm and Direct Search Toolbox [11] for solving optimization problems in the development of catalytic materials. Indeed, such general EA software only optimizes functions with input spaces of low-level data types, such as vectors of real numbers and bit-strings. And encoding the qualitative and quantitative compositions of catalytic materials, and their preparation and reaction conditions with low-level data types is a tedious and error-prone activity. Furthermore, it requires a great deal of mathematical erudition.

For all these reasons, it is not surprising that general EA software was in catalysis used only at the very beginning, to optimize the distribution of active sites of two catalytic components [12,13], and later only in rarely encountered multiobjective optimization [14,15]. Nevertheless, also in a more recently reported catalytic application of multiobjective evolutionary optimization, the employed genetic programming approach was specifically adapted to catalytic design [16], although the same authors have reported also the application of a general genetic programming method [17] to search an optimal approximation to the time dependency of conversion [18]. Apart from the above-mentioned exceptions, the application of EA in this area did not take the route of using general EA software, but instead that of developing specific algorithms for the optimization of catalytic materials. The implementation of such an algorithm for solid catalyst development was first undertaken by Wolf and coworkers at the Institute of Applied Chemistry in Germany, in the late 1990s [5,16,20]. Similar algorithms were later also developed at the Institute of Chemical Technology in Spain [16–22], at the Institute for Catalytic and Environmental Research in France [9,23,24], as well as at other institutions [8,25,26]. Experience gained with these algorithms shows that any new method of catalyst design, in particular if it introduces new constraints interconnecting discrete and continuous descriptors, can substantially decrease the usefulness of an earlier-implemented specific EA after several years. To retain a high usefulness of the implementation for a long period requires, during the development of that implementation, the anticipation of all constraints between the composition of catalytic materials and their preparation methods for which the implemented algorithm might need to be employed in the future. In addition, the more possibilities the implementation of an evolutionary algorithm attempts to cover, the more heuristic parameters it needs to achieve the desired flexibility.

These problems with EA developed specifically for the optimization of catalytic materials inspired the basic idea behind the approach proposed in [10]: to postpone the implementation of the algorithm as much as possible, i.e., to only implement the algorithm immediately before it is used to solve a particular optimization task. Since at that time all requirements concerning the task are already known, this approach enables the EA to be precisely problem-tailored.

However, traditional implementation undertaken by human programmers is not feasible in such a situation since it is error-prone, expensive and too slow. In [10], it was therefore proposed that problem-tailored EA implementations for the search of catalytic materials should be *generated automatically*. This is in accordance with a general trend in evolutionary computation to translate high level descriptions of evolutionary algorithms into low level library functions [11], which are in efficient implementations compiled to lower-level languages like C++ and Java [27]. To this end, a *program generator* is required, i.e., a software system that converts given requirements into executable programs. In contrast to a human programmer, for a program generator the requirements have to be expressed in a rigorously formal way. In the prototype of such a generator developed at the Leibniz Institute for Catalysis (LIKAT) in Germany, the *Catalyst Description Language (CDL)* [28] has been used to this end (see [10] for an example of a CDL description). An overall schema of the functionality of the LIKAT prototype is depicted in Fig. 1. The program generator accepts text files with CDL descriptions as input, and produces EA implementations as output. In addition, it provides a graphical interface, the purpose of which is to remove from the users the necessity to write CDL descriptions manually, and the necessity to understand CDL syntax and semantics.

As far as the implementation of the prototype is concerned, the generated EA consists of a persistent part, called *skeleton*, and of a variable part, generated as a binary code, which is input to the EA skeleton. The skeleton has been implemented in Matlab, and makes calls to the procedure `ga` from its Global Optimization Toolbox (successor of Genetic Algorithm and Direct Search Toolbox) [29], as well as to the constructor and several methods of the class `polytope` from the Multi-Parametric Toolbox from ETH Zurich [30]. The generated EA take into account the fact that the objective function is empirical. Therefore, if its values have to be obtained through experimental testing, the EA implementation runs only once and then exits. However, the approach also previews the possibility of obtaining those values from a surrogate model instead, which is discussed in the next section. In such a case, the EA implementation alternates with that program for as many generations as desired.

The employed way of solving mixed constrained optimization tasks is based on restricting the set of considered constraints to only *linear constraints*. Even if the set of possible solutions of the task is not constrained linearly in reality, the finite measurement precision of the involved continuous variables always allows to constrain it

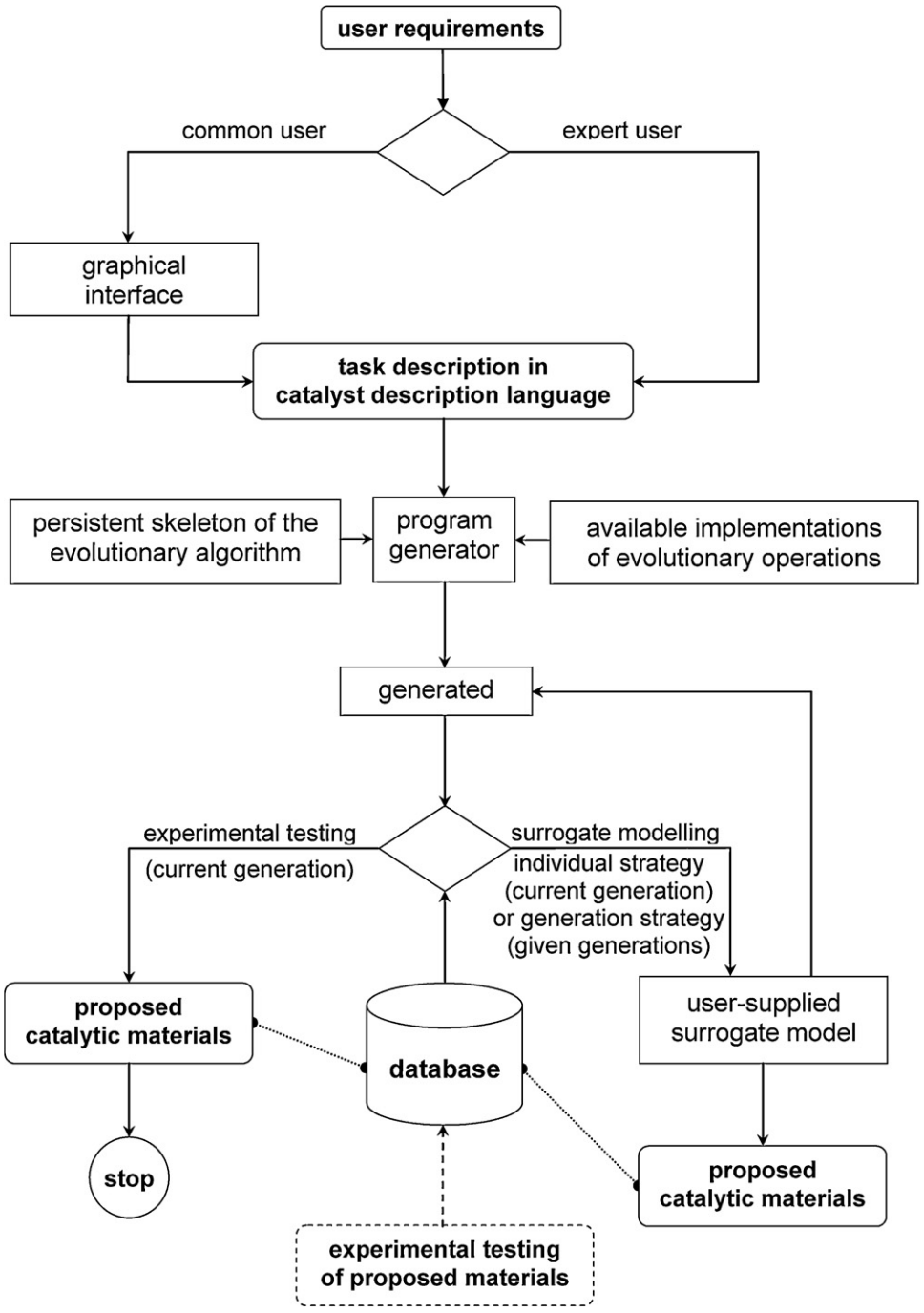

Fig. 1. Schema of the functionality of the LIKAT prototype of a program generator that generates problem-tailored evolutionary algorithms according to CDL descriptions.

piecewise linearly and to indicate the relevant linear piece with an additional discrete variable. Consequently, combinations of values of the continuous variables that form possible solutions together with a particular combination of values of the discrete variables are actually the solutions $\boldsymbol{x}$ of a particular system of linear inequalities $\boldsymbol{Ax} \leq \boldsymbol{b}$, determined through a matrix $\boldsymbol{A}$ and a vector $\boldsymbol{b}$. Naturally, the matrix $\boldsymbol{A}$ and the vector $\boldsymbol{b}$ depend on the considered combination of values of the discrete variables. In mathematical notation, the solutions $\boldsymbol{x}$ of $\boldsymbol{Ax} \leq \boldsymbol{b}$ form the set

$$\boldsymbol{P} = \{\boldsymbol{x} : \boldsymbol{Ax} \leq \boldsymbol{b}\}, \tag{1}$$

which is called *polytope*. It can be empty, and has its specific dimension, between 1 (closed interval) and the number of continuous variables. The polytope is called *feasible* if it is nonempty because only then a combination of values of the continuous variables exists that forms together with the considered combination of values of the discrete variables a possible solution of the optimization task.

To explain how feasibility of polytopes described with (1) is checked in the LIKAT prototype, let us recall several facts from algebra:

(i) The emptiness/nonemptiness of the polytope (1) is invariant with respect to any permutation of columns of $\boldsymbol{A}$, as well as with respect to any permutation of rows of $(\boldsymbol{Ab})$, where $(\boldsymbol{Ab})$ stands for the concatenation of $\boldsymbol{A}$ and $\boldsymbol{b}$.
(ii) Consider a relation $\equiv$ between polytopes, defined for any polytopes $\boldsymbol{P}$, $\boldsymbol{P}'$ as follows:

$\boldsymbol{P}$ is related to $\boldsymbol{P}'$ through the relation $\equiv$ (notation $\boldsymbol{P} \equiv \boldsymbol{P}'$) provided that $(\boldsymbol{A}'\boldsymbol{b}')$ can be obtained from $(\boldsymbol{A}\boldsymbol{b})$ through some permutation of columns of $\boldsymbol{A}$, followed by some permutation of rows of the result and of $\boldsymbol{b}$. (2)

This relation has the following properties:

- $\boldsymbol{P} \equiv \boldsymbol{P}$ for any polytope $\boldsymbol{P}$;
- if $\boldsymbol{P} \equiv \boldsymbol{P}'$, then also $\boldsymbol{P}' \equiv \boldsymbol{P}$;
- if $\boldsymbol{P} \equiv \boldsymbol{P}'$ and $\boldsymbol{P}' \equiv \boldsymbol{P}''$, then also $\boldsymbol{P} \equiv \boldsymbol{P}''$.

In algebra, a relation with these three properties is called *equivalence*. Hence, the relation (2) is an equivalence on the set of all polytopes.

(iii) Every equivalence partitions the set on which it is defined into disjoint subsets of equivalent elements, called *equivalence classes*. Thus also the relation (2) partitions the set of all polytopes into equivalence classes of polytopes. Polytopes $\boldsymbol{P}$ and $\boldsymbol{P}'$ such that $\boldsymbol{P}' \equiv \boldsymbol{P}$ always belong to the same class.

A consequence of the facts (i) and (iii) is that whenever a polytope $\boldsymbol{P}$ is nonempty, then all plytopes $\boldsymbol{P}'$ that belong to the same equivalence class are also nonempty, and whenever a polytope $\boldsymbol{P}$ is empty, then all plytopes $\boldsymbol{P}'$ that belong to the same equivalence class are also empty. This plays a crucial role in checking the feasibility, i.e., nonemptiness, of polytopes if the number of solution polytopes is large. Whereas a separate check of the nonemptiness of each polytope would make such optimization tasks insoluble by the EA, forming the equivalence classes is fast, and then only one polytope from each class needs to be checked. In Fig. 2, the graphical interface of the generated EA implementations is shown for a simple artificial example used during the development of the LIKAT program generator (left), and for one of the real tasks tackled during the first year of its routine use (right). In the artificial example, the 6494 polytopes actually formed only 9 equivalence classes, whereas the immense number of 583 232 160 polytopes in the real task formed 480 equivalence classes.

This way of tackling constraints together with the fact that the constraints determine which combinations of values of discrete variables to consider, suggest the following steps for dealing with mixed constrained optimization:

1. A separate continuous optimization task is formulated for each combination of values of discrete variables that can be for some combination of values of continuous variables feasible with respect to the specified constraints.
2. The set of all solution polytopes resulting from step 1, is partitioned according to the relation (2).
3. One polytope from each partition class is checked for nonemptiness, taking into account the discernibility of considered variables.
4. On the set of nonempty polytopes, discrete optimization is performed, using operations selection, mutation and crossover developed specifically to this end. In particular:
   - *Selection* of nonempty polytopes is performed in the first generation uniformly, in subsequent generations proportionally to the importance of the polytope due to points from earlier generations. A measure of that importance is based on the value of the objective function, which is usually called fitness in the context of evolutionary optimization. More precisely, the difference between the fitness of a point and the minimal fitness of points from the earlier generations is used, summed over all points with values of discrete variables corresponding to the polytope.
   - *Mutation* is performed simply through replacing an existing polytope with another one, uniformly selected. The values of continuous variables forming a point in that polytope are subsequently obtained in step 5. If the mutation rate is μ, then the proportion μ of the population is selected in this way, and the proportion $1 - \mu$ is selected using the proportional selection described above.
   - *Crossover* relies on the fact that a polytope (1) is determined on the one hand by the assignment of continuous variables to the columns of $\boldsymbol{A}$, on the other hand by a particular combination of values of some or all discrete variables. We assume that for the optimization of catalytic materials, most important is the assignment of variables to the columns of $\boldsymbol{A}$, and we have implemented a crossover operation based on this assumption: it always exchanges exactly one of the continuous variables assigned to the parent polytopes, and attempts to include as many corresponding discrete variables as possible.
5. In each of the polytopes obtained through the discrete optimization, standard linearly-constrained continuous optimization is performed. In the LIKAT prototype, Matlab's Global Optimization Toolbox [29] is employed to this end. The combinations of values of continuous variables found in this way, combined with the combinations of values of discrete variables corresponding to the respective polytope, form together the population of solutions of the mixed constrained optimization task.

3. Application of surrogate modeling

Like other methods relying solely on function values, evolutionary algorithms need the fitness to be evaluated in a large number of points. In the context of optimization of empirical objective functions, this is quite disadvantageous because the evaluation of such a function is often costly and time-consuming. This is also the case for the objective functions pertaining to the optimization of catalytic materials, such as yield, conversion, selectivity.

The usual approach to decreasing the cost and time of optimization of empirical objective functions is to evaluate the function only sometimes and to evaluate its suitable regression model otherwise. That model is termed *surrogate model* of the function, and the approach is referred to as *surrogate modeling* [31–38], or sometimes also *metamodeling* [39]. In the context of the optimization of catalytic materials, surrogate modeling can be viewed as replacing real experiments with simulated, virtual experiments in a computer. In [40,41], such virtual experiments are called *virtual screening*; in [42], they are called *pre-screening*.

Although surrogate modeling has been applied also to conventional optimization [43], it is most frequently encountered in connection with evolutionary algorithms because for them, the approach leads to the approximation of the fitness function, whose usefulness in evolutionary computation has been already recognized [44,45]. For the progress of evolutionary optimization, most important indicators are on the one hand points that indicate closeness to the global optimum (through highest values of the fitness), on the other hand points that most contribute to the diversity of the population.

In the literature, various possibilities of combining evolutionary optimization with surrogate modeling have been discussed [37,38,46]. Nevertheless, all of them are controlled by one of these two strategies:

A. The *individual-based strategy* consists in choosing between the evaluation of the empirical fitness and the evaluation of its surrogate model individual-wise, for example, in the following steps:

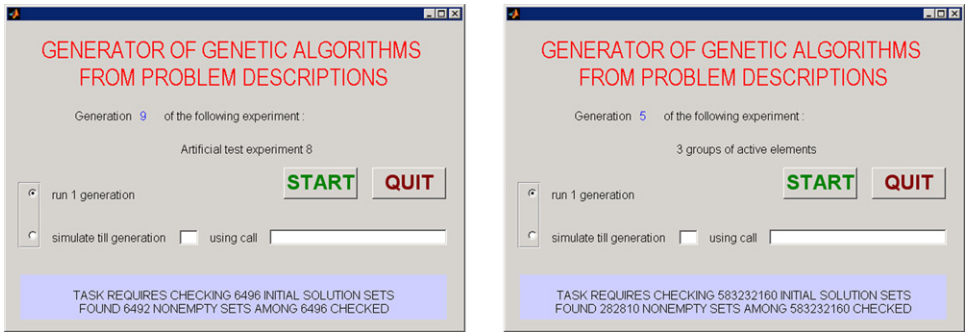

Fig. 2. Screenshots of the graphical interface of EA implementations generated by the prototype program generator developed at LIKAT: for a simple artificial example used during its development (left), and for a real task tackled during the first year of its routine use (right). To check the nonemptiness of the reported number of solution polytopes, the algorithm actually needed to check only 9 equivalence classes in the task on the left screenshot, and 480 classes in the task on the right screenshot.

(i) An initial set of individuals in which the considered empirical fitness was evaluated is collected (e.g., individuals forming several first generations of the evolutionary algorithm).
(ii) The surrogate model is constructed.
(iii) The evolutionary algorithm is run with the fitness replaced by the model for one generation with a population q-times larger than is the desired population size for the original fitness, where q is a prescribed ratio (e.g., $q = 10$ or $q = 100$).
(iv) A final population of the desired size is selected from the q-times larger one so as to contain those individuals that are most important according to the considered indicators for the progress of optimization.
(v) For this final population, the empirical fitness is evaluated.

B. The *generation-based strategy* consists in choosing between both kinds of evaluation generation-wise, for example, in the following steps:
(i) An initial set of individuals in which the empirical fitness was evaluated is collected like with the individual-based strategy.
(ii) The surrogate model is constructed.
(iii) Relying on the error of the surrogate model, an appropriate number g_m of generations is chosen, during which the fitness should be replaced by the model. More precisely, a superposition of errors in all those generations is assumed, and those errors are estimated with the cross-validation error of the model. Therefore, g_m is taken to be the smallest multiple that lets the cross-validation error exceed a given threshold.
(iv) The evolutionary algorithm is run with the fitness replaced by the model for g_m generations with populations of the originally desired size.
(v) Finally, the algorithm is run once more with the empirical fitness without changing the population size.

Our research into the applicability of surrogate modeling to the optimization of catalytic materials pursues the objective of enhancing the EA generated by the program generator, and focuses on flexible surrogate models, capable to convey the highly nonlinear nature of dependences encountered in this area (Fig. 3):

- artificial neural networks, which are the most frequent nonlinear regression model in catalysis (cf. the survey paper [47], applications from recent years are reported, e.g., in [40,42,48–50]), kernel regression based on the support vector approach [51], which has also already been introduced to catalysis [52,53], and
- regression trees [54], which have been used in this area even repeatedly [55–58], as well as
- piecewise-linear neural networks, useful for the extraction of logical rules in data mining [59].

3.1. Improving the accuracy of surrogate models with boosting

A crucial aspect of surrogate modeling is that the agreement between the results obtained with a surrogate model and those obtained with the original function depends on the accuracy of the model.

A general possibility to increase the accuracy of surrogate models is provided by *boosting*. Boosting is a popular approach to increase the accuracy of classification; its success in classification incited also several methods of regression boosting. Both for classification and for regression, the basic approach to increasing the relative influence of training data that most contributed to the overall error is re-sampling the training data according to a distribution that gives to data most contributing to the error a higher probability of occurrence. This is equivalent to re-weighting the contributions of the individual training pairs with higher weights corresponding to higher values of the error measure.

We are aware of the fact that in computer science, the usefulness of boosting has been known since the late 1990s. Therefore, we have not tested the approach on any artificial computer science benchmark problems such as those employed for testing new developments of evolutionary algorithms [60]. Instead, we have tested it on several case studies in heterogeneous catalysis, one

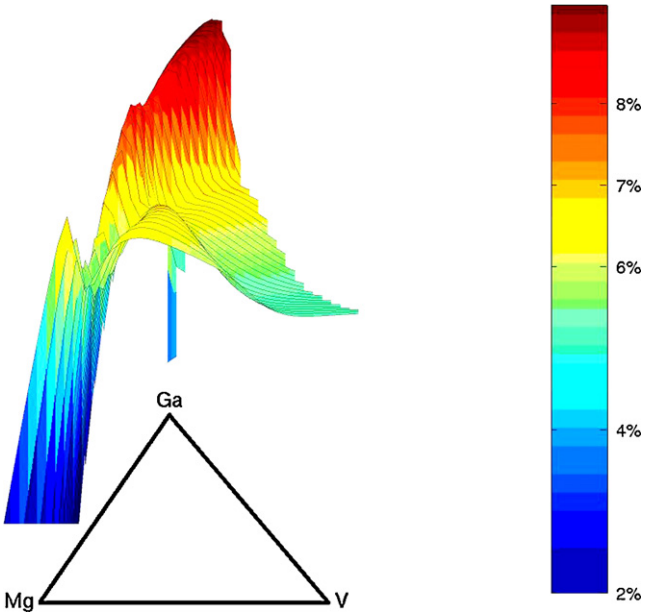

Fig. 3. A regression model, computed by an artificial neural network of the kind multilayer peceptron, modeling the dependence of yield on the fractions of Mg, V and Ga in mixed metal oxide catalysts for the oxidative dehydrogenation of propane.

of which will be presented in Section 4. Since surrogate models are regression models, any method for regression boosting (such as [61–63]) is suitable for them. In the following, the method *AdaBoost.R2* [64] will be explained in detail, which is a regression modification of the most popular classification boosting method, AdaBoost [61,65].

Similarly to other adaptive boosting methods, each of the training pairs $(\boldsymbol{x}_1,\boldsymbol{y}_1),\ldots,(\boldsymbol{x}_p,\boldsymbol{y}_p)$ is in the first iteration of AdaBoost.R2 used exactly once. This corresponds to resampling them according to the uniform probability distribution $\boldsymbol{P}_1$ with $\boldsymbol{P}_1(\boldsymbol{x}_k,\boldsymbol{y}_k)=1/\boldsymbol{p}$ for $\boldsymbol{k}=1,\ldots,\boldsymbol{p}$. In addition, the weighted average error of the first iteration is set to zero, $\bar{\boldsymbol{E}}_1=0$.

In the subsequent iterations $\boldsymbol{i}\geq 2$, the following sequence of steps is performed:

1. A sample $(\boldsymbol{\xi}_1,\boldsymbol{\eta}_1),\ldots,(\boldsymbol{\xi}_p,\boldsymbol{\eta}_p)$ is obtained through resampling $(\boldsymbol{x}_1,\boldsymbol{y}_1),\ldots,(\boldsymbol{x}_p,\boldsymbol{y}_p)$ according to the distribution $\boldsymbol{P}_{i-1}$.
2. Using $(\boldsymbol{\xi}_1,\boldsymbol{\eta}_1),\ldots,(\boldsymbol{\xi}_p,\boldsymbol{\eta}_p)$ as training data, a regression model $\boldsymbol{F}_i$ is constructed.
3. A [0,1]-valued error vector $\boldsymbol{E}_i$ of $\boldsymbol{F}_i$ with respect to $(\boldsymbol{x}_1,\boldsymbol{y}_1),\ldots,(\boldsymbol{x}_p,\boldsymbol{y}_p)$ is calculated as

$$\begin{aligned}\boldsymbol{E}_i &= (\boldsymbol{E}_i(1),\ldots,\boldsymbol{E}_i(\boldsymbol{p}))\\ &= \frac{1}{\max\limits_{\boldsymbol{k}=1,\ldots,\boldsymbol{p}}(\boldsymbol{F}_i(\boldsymbol{x}_k)-(\boldsymbol{y}_k))^2}\left((\boldsymbol{F}_i(\boldsymbol{x}_1)-\boldsymbol{y}_1)^2,\ldots,(\boldsymbol{F}_i(\boldsymbol{x}_p)-\boldsymbol{y}_p)^2\right).\end{aligned}$$

4. The weighted average error of the $\boldsymbol{i}$-th iteration is calculated as

$$\bar{\boldsymbol{E}}_i=\frac{1}{\boldsymbol{p}}\sum_{\boldsymbol{k}=1}^{\boldsymbol{p}}\boldsymbol{P}_i(\boldsymbol{x}_k,\boldsymbol{y}_k)\boldsymbol{E}_i(\boldsymbol{k}).$$

5. Provided $\bar{\boldsymbol{E}}_i<0.5$, the probability distribution for resampling $(\boldsymbol{x}_1,\boldsymbol{y}_1),\ldots,(\boldsymbol{x}_p,\boldsymbol{y}_p)$ is updated according to

$$\boldsymbol{P}_i(\boldsymbol{x}_k,\boldsymbol{y}_k)=\frac{\boldsymbol{P}_{i-1}(\boldsymbol{x}_k,\boldsymbol{y}_k)(\bar{\boldsymbol{E}}_i/(1-\bar{\boldsymbol{E}}_i))^{(1-\boldsymbol{E}_i(\boldsymbol{k}))}}{\sum\limits_{l=1}^{\boldsymbol{p}}\boldsymbol{P}_{i-1}(\boldsymbol{x}_k,\boldsymbol{y}_k)(\bar{\boldsymbol{E}}_i/(1-\bar{\boldsymbol{E}}_i))^{(1-\boldsymbol{E}_i(\boldsymbol{k}))}},\quad \boldsymbol{k}=1,\ldots,\boldsymbol{p}.$$

(The case $\bar{\boldsymbol{E}}_i\geq 0.5$ will be discussed below.)

6. As the *boosting approximation* in the $\boldsymbol{i}$-th iteration serves the median of the approximations $\boldsymbol{F}_1,\ldots,\boldsymbol{F}_i$ with respect to the probability distribution proportional to the vector

$$\left(\frac{\bar{\boldsymbol{E}}_1}{1-\bar{\boldsymbol{E}}_i},\ldots,\frac{\bar{\boldsymbol{E}}_i}{1-\bar{\boldsymbol{E}}_i}\right) \tag{3}$$

The errors used to asses the quality of the boosting approximation are then called *boosting errors*, e.g., *boosting MSE*, or *boosting MAE*, where MSE refers to the *mean squared error* between the computed and measured values, whereas MAE refers to the *mean absolute error*, i.e., to the mean absolute distance between them. For simplicity, also the approximation in the first iteration, $\boldsymbol{F}_1$, is called boosting approximation if boosting is performed, and the respective errors are then called boosting errors, although boosting actually does not introduce any modifications in the first iteration.

The above formulation of the method deals only with the case $\bar{\boldsymbol{E}}_i<0.5$. For $\bar{\boldsymbol{E}}_i\geq 0.5$, the original formulation of the method in [64] proposes to stop the boosting. However, that is not allowed if the stopping criterion should be based on an independent set of validation data. Indeed, the calculation of $\bar{\boldsymbol{E}}_i$ does not rely on any such independent data set, but it relies solely on the data employed to construct the regression model. A possible alternative for the case $\bar{\boldsymbol{E}}_i\geq 0.5$ is reinitialization, i.e., proceeding as in the first iteration [66].

In connection with the above recalled fact that the nonlinear regression models most frequently encountered in heterogeneous catalysis are artificial neural networks [47,67], it is important to be aware of the difference between the iterations of boosting and the iterations of neural network training. Indeed, boosting iterates on a higher level; one iteration of boosting includes a complete training of a neural network, which can proceed for many hundreds of iterations. Nevertheless, both kinds of iterations are similar in the sense that starting with some iteration, the network overtrains. Therefore, also over-training due to boosting can be reduced through *stopping* in the iteration after which the *error for an independent set of data first time increases*. Moreover, *cross-validation* can be used to find the iteration most appropriate for stopping [47].

4. Case study with high-temperature synthesis of HCN

The integration of surrogate modeling with the EA generated by the prototype program generator developed at LIKAT is a matter of ongoing development. However, we already have more than five years experience with surrogate modeling in the context of the first specific algorithm for the optimization of catalytic materials [5] and of its later modifications [7]. Relying on that experience, we would like to illustrate the application of surrogate modeling and of regression boosting on a case study with data from the investigation of catalytic materials for the *high-temperature synthesis of hydrocyanic acid*. This investigation and its results were recently described in [58]. The investigation was performed through high-throughput experiments in a circular 48-channel reactor. The data consist of 7 EA generations with a population of 92 catalytic materials, and 52 additional catalysts, with composition designed directly by the experimenter, without using an evolutionary algorithm. Consequently, data from a total of 696 catalytic materials were collected.

The composition and preparation of the materials and the conditions to which they had been exposed have been described in detail in [58]. Here, only those facts are recalled that are important for understanding which variables are considered in the surrogate model.

(i) All the materials tested contained a *support*, which was sequentially impregnated with one to six active metal additives. As support, in every case one of the following *15 materials* was used: pure α-Al_2O_3 (alsint), as well as the compounds AlN, Mo_2C, TiB_2, TiN, NB_2O_3, BN, ZrO_2, Sm_2O_3, SrO, CaO, MgO, TiO_2, SiC, and Si_3N_4, bound in an alumina matrix.

(ii) The active components by which the supports were covered had been selected from the pool of 11 compounds: $Y(NO_3)_3$, $La(NO_3)_3$, $ZrO(NO_3)_2$, H_2MoO_4, $ReCl_3$, $IrCl_4$, $NiCl_2$, H_2PtCl_6, $Zn(NO_3)_2$, $AgNO_3$, and $HAuCl_4$, providing in turn the 11 metal additives Y, La, Zr, Mo, Re, Ir, Ni, Pt, Zn, Ag and Au. It is important to realize that the proportions of these components in the active part of the catalyst are not completely independent since they sum up to 100%.

(iii) The catalytic performance, in particular the *degrees of conversion of CH_4 and NH_3*, and the *HCN yield*, was measured only after the final composition of the inlet feed gas was reached through stepwise addition of CH_4 to the initial ammonia/argon mixture. That final composition amounted to 10.7 vol.% NH_3, 9.3 vol.% CH_4 and 80 vol.% Ar. The catalytic performance was then measured for *temperatures in the range 1173–1373 K*. Consequently, it is not possible to simply use all the performance measurement results as values of output variables—either the measurement temperature has to be added to the input vari-

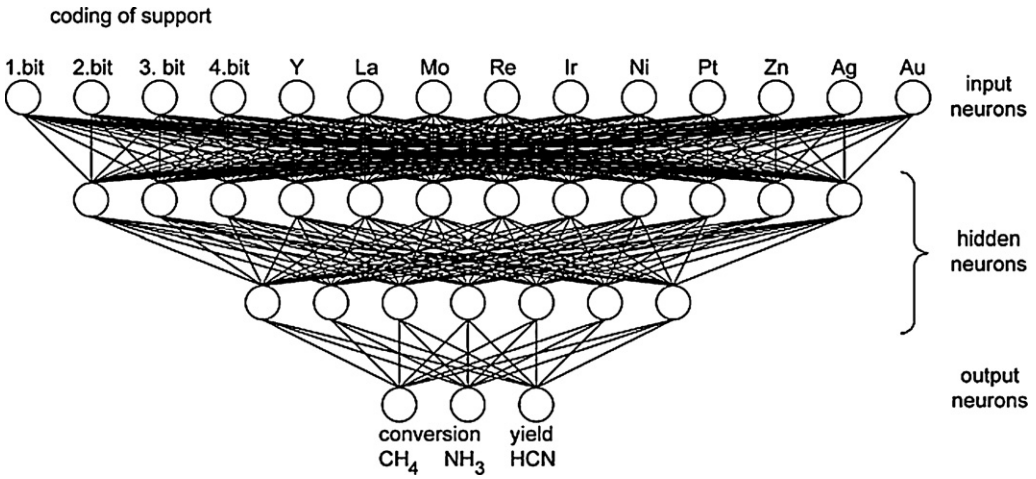

Fig. 4. Example multilayer perceptron with two hidden layers, used in the case study. Reprinted from [70].

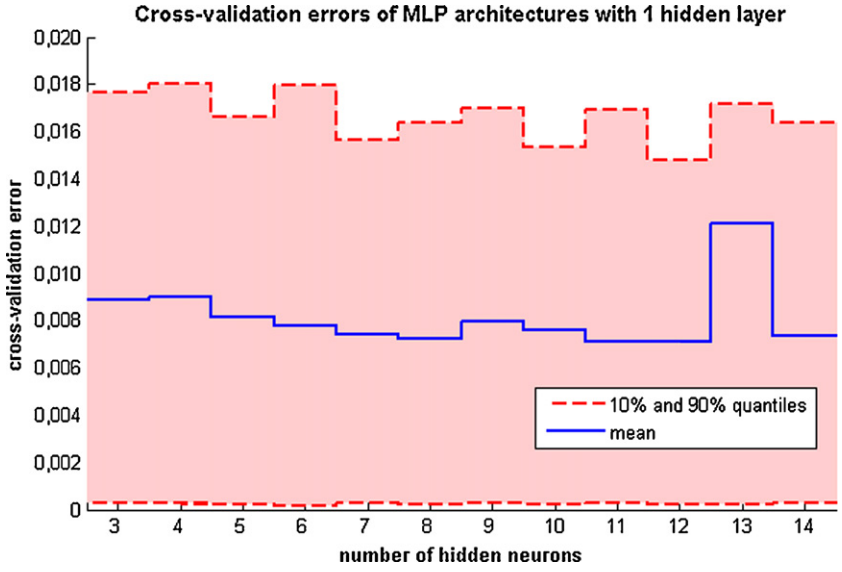

Fig. 5. Mean values, 10% percentiles and 90% percentiles of the cross-validation MSE on test data for each of the 12 MLP architectures with one hidden layer fulfilling the condition $3 \leq n_H \leq 14$.

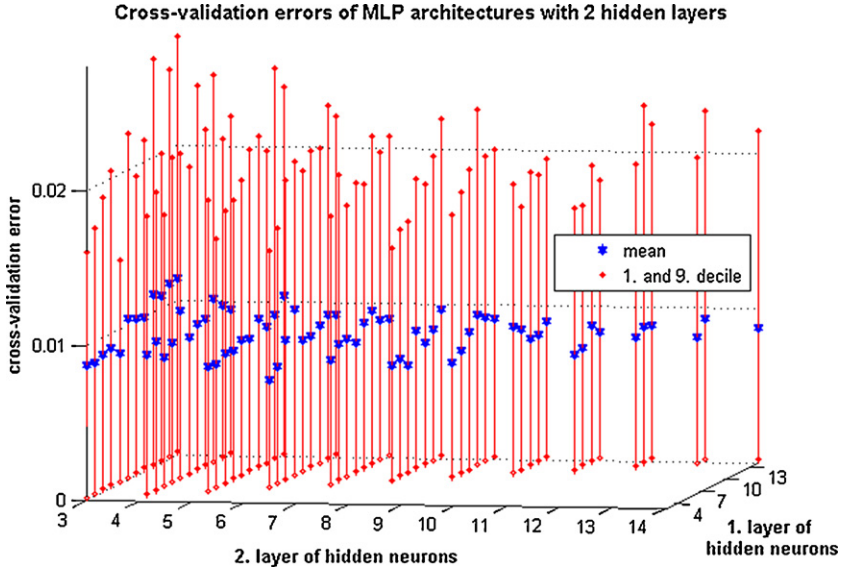

Fig. 6. Mean values, 10% percentiles and 90% percentiles of the cross-validation MSE on test data for each of the 78 MLP architectures with two hidden layers fulfilling the condition $3 \leq n_{H2} \leq n_{H1} \leq 14$.

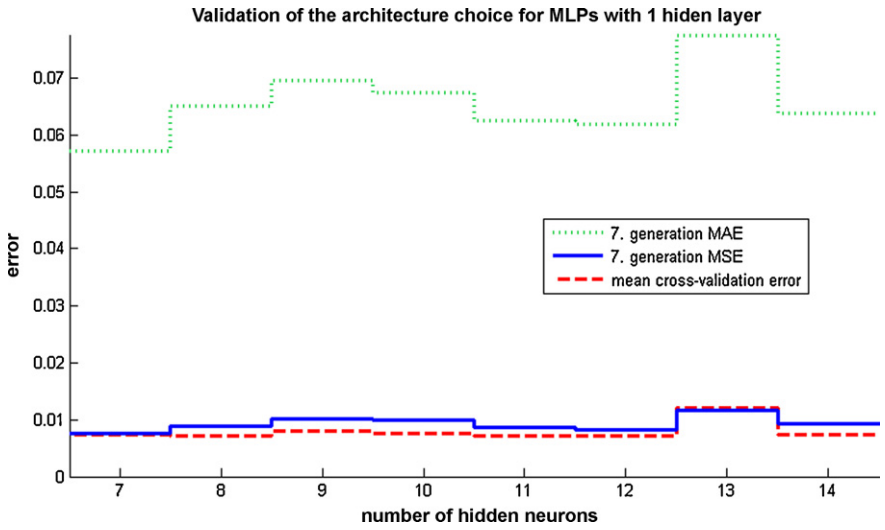

Fig. 7. MAE and MSE of predictions computed for materials from the 7th generation of the evolutionary algorithm by MLPs with one hidden layer fulfilling $7 \leq n_H \leq 14$, trained with all the data considered during the search of the most appropriate architecture. For comparison, mean cross-validation error values of the involved architectures are recalled from Fig. 5.

ables, or only results corresponding to a single temperature can be used. The latter possibility was preferred and only data collected at the highest temperature 1373 K were taken into account.

(iv) As the surrogate model, artificial neural networks of the kind *multilayer perceptron (MLP)* were employed, in accordance with their leading role among nonlinear regression models in the area of catalytic materials [68]. Each considered neural network had *14 input neurons*: 4 of them coding the material used as support, the other 10 corresponding to the proportions of the 10 metal additives belonging to independent variables; output neurons were 3, corresponding to the considered measures of catalytic performance (Fig. 4).

The most appropriate MLP architectures were searched by means of cross-validation, the importance of which for checking generalization properties of regression model is meanwhile well known also in catalysis [15,42,47,48,52,53,57,69,70]. For the architecture search, only data about catalysts from the 1st to 6th generation of the evolutionary algorithm and about the 52 catalysts with manually designed composition were used, thus altogether data about 604 catalytic materials. Data about catalysts from the 7th generation were completely excluded and left out for validating the search results. To use as much information as possible from the available data, cross-validation was applied as the extreme 604-fold variant, i.e., leave-1-out validation. The set of architectures within which the search was performed was delimited by means of the heuristic *pyramidal condition*: the number of neurons in a subsequent layer must not exceed the number of neurons in a previous layer. Denote $\boldsymbol{n}_I$, $\boldsymbol{n}_H$ and $\boldsymbol{n}_O$ the numbers of input, hidden and output neurons, respectively, and $\boldsymbol{n}_{H1}$ and $\boldsymbol{n}_{H2}$ the numbers of neurons in the first and second hidden layer, respectively. Then the pyramidal condition reads:

(i) For MLPs with one hidden layer: $\boldsymbol{n}_O \leq \boldsymbol{n}_H \leq \boldsymbol{n}_I$, in our case $3 \leq \boldsymbol{n}_H \leq 14$ (12 architectures).

(ii) For MLPs with two hidden layers: $\boldsymbol{n}_O \leq \boldsymbol{n}_{H2} \leq \boldsymbol{n}_{H1} \leq \boldsymbol{n}_I$, in our case $3 \leq \boldsymbol{n}_{H2} \leq \boldsymbol{n}_{H1} \leq 14$ (78 architectures).

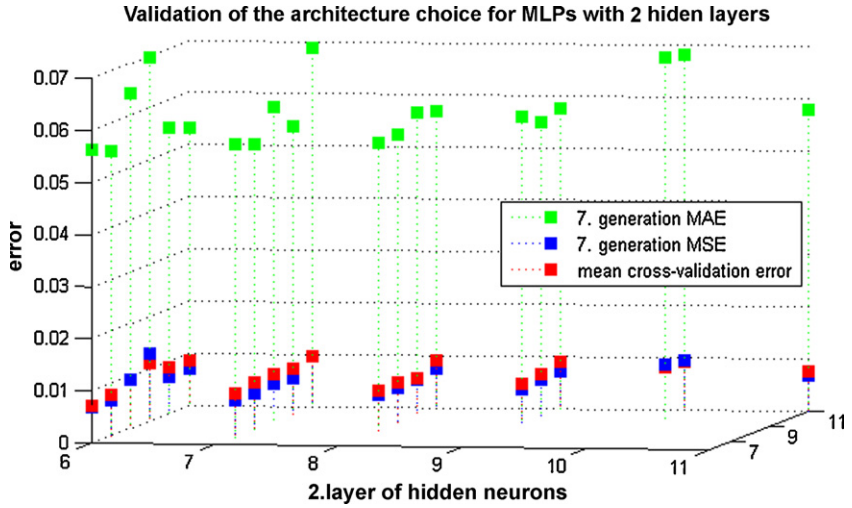

Fig. 8. MAE and MSE of approximations computed for materials from the 7th generation of the evolutionary algorithm by MLPs with two hidden layers fulfilling $6 \leq n_{H2} \leq n_{H1} \leq 11$, trained with all the data considered during the architecture search. For comparison, mean cross-validation error values of the involved architectures are recalled from Fig. 6.

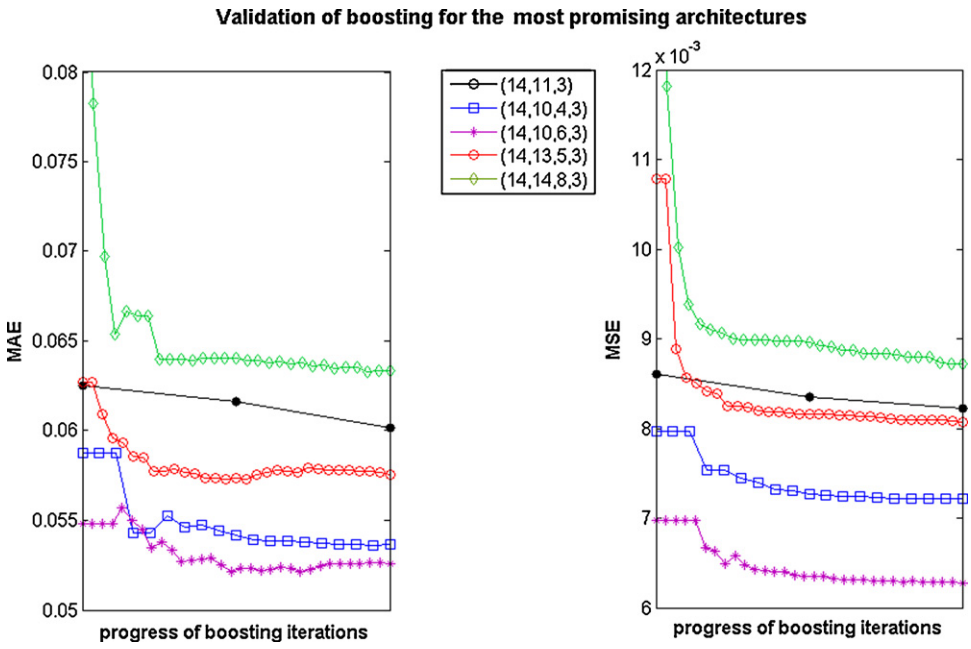

Fig. 9. History of the boosting MSE and MAE on the data from the 7th generation of the evolutionary algorithm for MLPs with the five architectures included in the validation of boosting. Reprinted from [70].

The results about cross-validation MSE on test data are summarized for the considered MLP architectures with one hidden layer in Fig. 5, and for the architectures with two hidden layers in Fig. 6. Besides the mean value of this cross-validation error, they contain its first and last decile (10%- and 90%-percentile). Consequently, the interval between these two additional values is the 80% confidence interval, based on the leave-1-out validation, of the average squared bias of HCN yield, CH_4 conversion and NH_3 conversion predicted by MLPs with the respective numbers of hidden neurons.

The results of searching the most appropriate architecture for the MLP serving as surrogate model were validated on the data about the 92 catalytic materials from the 7th generation of the evolutionary algorithm. For the validation, only one-hidden-layer architectures with the number of hidden neurons $\boldsymbol{n}_H$ from the range $7 \leq \boldsymbol{n}_H \leq 14$, and 2-hidden layers architectures with the numbers of hidden neurons $\boldsymbol{n}_{H1}$ and $\boldsymbol{n}_{H2}$ from the ranges $6 \leq \boldsymbol{n}_{H2} \leq \boldsymbol{n}_{H1} \leq 11$ were considered, due to the fact that the five best architectures of each kind had the numbers of hidden neurons from those ranges. The validation proceeded as follows:

1. Each MLP was employed to approximate the conversions of CH_4 and NH_3 and the yield of HCN for the 92 materials from the 7th generation.
2. From the two conversions and the yield predicted by those approximations, and from the corresponding measured values, the MSE and MAE were calculated for each MLP.
3. For each of the 8 architectures with one hidden layer and each of the 21 architectures with two hidden layers with numbers of hidden neurons from the above ranges, a single MLP was trained, using the data about all the 604 catalytic materials considered during the architecture search.

The obtained MSE and MAE values are depicted and compared with the average values of the cross-validation MSE on test data

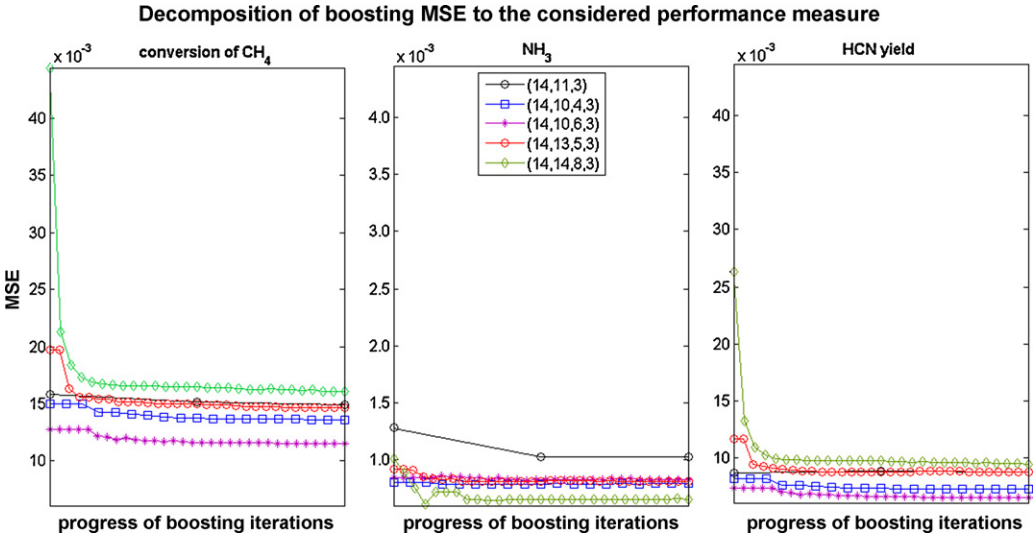

Fig. 10. Decomposition of the boosting MSE for the MLPs from Fig. 9 to the conversions of CH_4 and NH_3 to of the yield of HCN.

Table 1
MLPs for which boosting improved the average MSE on the test data in leave-one-out validation, together with the final iteration until which it was improved, and its values in the first and final iteration. Reprinted from [70].

Architecture	Final iteration	Boosting MSE on the test data (10^{-3})		Architecture	Final iteration	Boosting MSE on the test data (10^{-3})	
		First iteration	Final iteration			First iteration	Final iteration
(14,3,3)	2	8.89	8.88	(14,10,10,3)	3	9.70	9.41
(14,4,3)	7	8.99	8.86	(14,11,3,3)	7	11.30	9.03
(14,5,3)	3	8.17	7.11	(14,11,5,3)	8	9.81	7.98
(14,6,3)	3	7.80	7.75	(14,11,6,3)	6	8.69	7.82
(14,8,3)	2	7.22	7.20	(14,11,7,3)	7	9.59	8.92
(14,9,3)	2	7.95	7.92	(14,11,8,3)	24	9.18	7.53
(14,11,3)	3	7.13	7.11	(14,11,10,3)	2	9.28	9.13
(14,13,3)	2	12.11	12.03	(14,12,3,3)	11	10.94	7.31
(14,14,3)	3	7.38	7.35	(14,12,4,3)	13	10.80	8.46
(14,3,3,3)	2	8.78	8.67	(14,12,5,3)	8	8.97	8.32
(14,4,3,3)	19	8.74	7.50	(14,12,6,3)	6	9.18	8.71
(14,4,4,3)	11	9.25	8.67	(14,12,7,3)	7	10.11	9.17
(14,5,3,3)	5	8.94	8.16	(14,10,10,3)	3	9.70	9.41
(14,5,5,3)	3	8.28	7.92	(14,11,3,3)	7	11.30	9.03
(14,6,3,3)	8	9.16	8.46	(14,11,5,3)	8	9.81	7.98
(14,6,4,3)	3	8.55	8.22	(14,11,6,3)	6	8.69	7.82
(14,6,6,3)	3	7.22	7.16	(14,11,7,3)	7	9.59	8.92
(14,7,3,3)	5	8.54	7.70	(14,11,8,3)	24	9.18	7.53
(14,7,4,3)	5	9.24	8.22	(14,11,10,3)	2	9.28	9.13
(14,7,5,3)	2	8.58	8.29	(14,12,8,3)	6	8.13	7.85
(14,7,6,3)	5	7.75	7.56	(14,12,9,3)	7	9.91	9.10
(14,7,7,3)	4	8.27	7.27	(14,12,10,3)	2	8.47	8.40
(14,8,3,3)	17	10.47	8.18	(14,12,12,3)	13	8.62	7.31
(14,8,4,3)	6	11.06	9.96	(14,13,3,3)	9	11.43	9.30
(14,8,5,3)	5	8.54	7.91	(14,13,4,3)	2	10.08	8.70
(14,8,6,3)	14	9.23	8.34	(14,13,5,3)	31	9.55	6.74
(14,8,8,3)	2	7.66	7.53	(14,13,6,3)	20	9.55	7.27
(14,9,3,3)	8	10.24	7.60	(14,13,7,3)	7	9.29	8.59
(14,9,4,3)	2	9.06	8.58	(14,13,8,3)	3	8.68	8.14
(14,9,5,3)	10	8.96	8.13	(14,13,9,3)	8	9.52	8.19
(14,9,6,3)	8	10.94	10.08	(14,13,10,3)	3	8.44	8.22
(14,9,7,3)	5	9.09	8.74	(14,13,12,3)	13	9.08	7.60
(14,10,3,3)	8	10.08	9.00	(14,13,13,3)	13	8.39	7.41
(14,10,4,3)	19	9.63	7.07	(14,14,3,3)	6	11.52	8.89
(14,10,5,3)	3	8.78	8.35	(14,14,4,3)	15	9.56	7.13
(14,10,6,3)	32	8.69	6.55	(14,14,5,3)	7	10.43	8.82
(14,10,7,3)	4	8.63	8.04	(14,14,7,3)	7	9.04	7.87

in Fig. 7 (for MLPs with one hidden layer) and Fig. 8 (for MLPs with two hidden layers). Figs. 7 and 8 show that errors of the predictions computed by the trained multilayer perceptrons for catalytic materials from the 7th generation were usually lower if the mean cross-validation errors of their architectures were also lower when searching the most appropriate architecture for the surrogate model. Similarly, higher mean cross-validation errors of the architectures typically correspond to higher errors of obtained predictions.

To investigate the usefulness of boosting for the employed surrogate models, the same data were used and the same set of architectures was considered as for their architecture search. In each iteration, a leave-one-out validation was performed [47]. Boosting was considered useful to those architectures for which the average MSE was in the second iteration lower than in the first iteration. The final iteration of boosting was taken when the average MSE no longer decreased. According to the above criterion, boosting was useful to nine out of the 12 considered MLPs with one hidden layer and to 65 out of the 78 considered MLPs with two hidden layers. For those architectures, basic information relevant to boosting is summarized in Table 1.

To validate the most promising results of the investigation of the usefulness of boosting in our case study, again the data from the 7th generation of the evolutionary algorithm were used. The validation included the surrogate models that were most promising for boosting from the point of view of the *lowest boosting MSE on the test data in the final iteration*. According to Table 1, these were the MLPs with architectures (14,10,6,3), (14,14,8,3), (14,13,5,3), (14,10,4,3) and (14,11,3). For each of them, the validation proceeded as follows:

1. In each iteration up to the final boosting iteration corresponding to that surrogate model, a single MLP was trained with data about all the 604 catalytic materials considered during their architecture search.
2. Each of those MLPs was employed to approximate the conversions of CH_4 and NH_3 and the yield of HCN for the 92 materials from the 7th generation of the evolutionary algorithm.
3. In each iteration, the medians with respect to the probability distribution (3) of the approximations of the two conversions and of the HCN yield obtained up to that iteration were used as their boosting approximations.
4. From the conversions and the yield predicted by the boosting approximations, and from the measured values, the boosting MSE and boosting MAE were calculated for each MLP.

The resulting boosting errors (MSE and MAE) are summarized in Fig. 9. They clearly confirm the usefulness of boosting for the five considered architectures. For each of them, boosting leads to an overall decrease of both considered error measures, MSE and MAE, on new data from the 7th generation of the evolutionary algorithm. Moreover, the decrease of the MSE (which is the measure employed in the preceding investigation of the usefulness of boosting) is uninterrupted or nearly uninterrupted until the final boosting iteration.

Finally, Fig. 10 decomposes the boosting MSE for the five considered architectures to the three properties employed as catalyst performance measures in our case study—conversion of CH_4, conversion of NH_3 and yield of HCN. Apparent is that boosting did not lead to any decrease of the error of the conversion of NH_3, which on the other hand is already from the beginning much lower than the two other performance measures (notice that the scale of the $\boldsymbol{y}$-axis is 10-times finer for the conversion of NH_3 than for the conversion of CH_4 and HCN yield). The explanation for the different behaviors of the conversion of NH_3 is the substantially lower variability of its values in the seventh generation of the evolutionary algorithm, used for validating the usefulness of boosting (standard deviation, SD: 2.8, interquartile range, IQR: 1.6), compared to the conversion of CH_4 (SD: 26.1, IQR: 45.0) and HCN yield (SD: 20.1, IQR: 35.9). Due to so low variability, the conversion of NH_3 appears effectively as nearly constant during the validation of boosting, which in turn accounts for a nearly constant MSE.

5. Conclusions

The paper presented ongoing developments of the recent generator approach to the evolutionary optimization of catalytic materials [10]. It explained how the employed evolutionary operations reflect the specificity of mixed constrained optimization tasks entailed by the search for catalysts with optimal performance. Even more importantly, it explained and documented on numerical examples how the generated EA tackle checking the feasibility of large polytope systems, frequently resulting from the optimization constraints.

As to the ongoing development of the approach, the paper emphasized that it is oriented to integration with surrogate modeling, and pointed out the role of regression boosting for improving the accuracy of surrogate models. Our orientation to the application of surrogate modeling is heading in the same directions as the research at the Institute for Catalytic and Environmental Research in France [40,41,71] and at the Institute of Chemical Technology in Spain [42]. However, this paper is to our knowledge the first in catalytic literature that introduces boosted surrogate models. Recently, our group reported using boosted artificial neural networks for the analysis of data from catalytic experiments [70], but without an explicit connection to surrogate modeling. The usefulness of surrogate modeling in general and of boosted surrogate models in particular has been documented on a case study with data from the high-temperature synthesis of hydrocyanic acid.

Acknowledgements

The research reported in this paper was supported by the German Federal Ministry of Education and Research (BMBF) and the state Mecklenburg—Western Pommerania, as well as by the Grant No. 201/08/0802 of the Grant Agency of the Czech Republic.

References

[1] A. Holzwarth, P. Denton, H. Zanthoff, C. Mirodatos, Catal. Today 67 (2001) 309–318.
[2] L. Végvári, A. Tompos, S. Göbölös, J.F. Margitfalvi, Catal. Today 81 (2003) 517–527.
[3] B. Li, P. Sun, Q. Jin, J. Wang, D. Ding, J. Mol. Catal. A: Chem. 148 (1999) 189–195.
[4] A. Eftaxias, J. Font, A. Fortuny, J. Giralt, A. Fabregat, F. Stüber, Appl. Catal. B: Environ. 33 (2001) 175–190.
[5] D. Wolf, O.V. Buyevskaya, M. Baerns, Appl. Catal. A: Gen. 200 (2000) 63–77.
[6] K. Huang, X.L. Zhan, F.Q. Chen, D.W. Lü, Chem. Eng. Sci. 58 (2003) 81–87.
[7] U. Rodemerck, M. Baerns, M. Holeňa, D. Wolf, Appl. Surf. Sci. 223 (2004) 168–217.
[8] Y. Watanabe, T. Umegaki, M. Hashimoto, K. Omata, M. Yamada, Catal. Today 89 (2004) 455–464.
[9] R.M. Pereira, F. Clerc, D. Farrusseng, J.C. Waal, T. Maschmeyer, QSAR Comb. Sci. 24 (2005) 45–57.
[10] M. Holeňa, T. Čukić, U. Rodemerck, D. Linke, J. Chem. Inf. Model. 48 (2008) 274–282.
[11] Genetic Algorithm and Direct Search Toolbox, The MathWorks, Inc., Natick, 2008.
[12] A.S. McLeod, M.E. Johnston, L.F. Gladden, J. Catal. 167 (1997) 279–285.
[13] A.S. McLeod, L.F. Gladden, J. Chem. Inf. Comput. Sci. 40 (2000) 981–987.
[14] O.C. Gobin, J.A. Martinez, F. Schüth, J. Catal. 252 (2007) 205–214.
[15] O.C. Gobin, F. Schüth, J. Comb. Chem. 10 (2008) 835–846.
[16] L.A. Baumes, P. Collet, Examination of genetic programming paradigm for high-throughput experimentation and heterogeneous catalysis, Comput. Mater. Sci. 45 (2009) 27–40.
[17] H. Majeed, C. Ryan, in: P. Collet, M. Tomassini, M. Ebner, S. Gustafson, A. Ekárt (Eds.), Genetic Programming, Springer, Berlin, 2006, pp. 36–48.
[18] L.A. Baumes, A. Blansché, P. Serna, A. Tchougang, N. Lachiche, P. Collet, A. Corma, Mater. Manuf. Processes. 24 (2009) 282–292.
[19] O.V. Buyevskaya, D. Wolf, M. Baerns, Catal. Today 62 (2000) 91–99.
[20] U. Rodemerck, D. Wolf, O.V. Buyevskaya, P. Claus, S. Senkan, M. Baerns, Chem. Eng. J. 82 (2001) 3–11.
[21] A. Corma, J.M. Serra, A. Chica, Catal. Today 81 (2003) 495–506.
[22] J.M. Serra, A. Chica, A. Corma, Appl. Catal. A: Gen. 239 (2003) 35–42.
[23] L. Baumes, P. Jouve, D. Farrusseng, M. Lengliz, N. Nicoloyannis, C. Mirodatos, in: V. Palade, R.J. Howlett, L.C. Jain (Eds.), Knowledge-based Intelligent Information and Engineering Systems, 2003, pp. 265–270.
[24] F. Clerc, M. Lengliz, D. Farrusseng, C. Mirodatos, S.R.M. Pereira, R. Rakotomata, Rev. Sci. Instrum. 76 (2005) 062208.
[25] G. Kirsten, W.F. Maier, Appl. Surf. Sci. 223 (2004) 87–101.
[26] A. Ohrenberg, C. Törne, A. Schuppert, B. Knab, QSAR Comb. Sci. 24 (2005) 29–37.
[27] P. Collet, E. Lutton, M. Schonauer, J. Louchet, in: M. Schonauer, K. Deb, G. Rudolph, X. Yao, J.J. Merelo, H.P. Schwefel (Eds.), Parallel Problem Solving from Nature, Springer, Berlin, 2007, pp. 891–901.
[28] M. Holeňa, A. Hagemeyer, P. Strasser, in: A.F. Volpe (Ed.), High-throughput Screening in Chemical Catalysis, Wiley-WCH, Weinheim, 2004, pp. 53–172.
[29] Global Optimization Toolbox 3, The MathWorks, Inc., Natick, 2010.
[30] M. Kvasnica, P. Grieder, M. Baotic, F.J. Christophersen, Multi-Parametric Toolbox (MPT), ETH, Zurich, 2005.
[31] A. Ratle, Artif. Intell. Eng. Des. Anal. Manuf. 15 (2001) 37–49.
[32] Y. Jin, M. Olhofer, B. Sendhoff, IEEE Trans. Evol. Comput. 6 (2002) 481–491.
[33] Y.S. Hong, H. Lee, M.J. Tahk, Eng. Optim. 35 (2003) 91–102.
[34] H. Ulmer, F. Streichert, A. Zell, in: E. Cantú-Paz, J.A. Foster, K. Deb, et al. (Eds.), Genetic and Evolutionary Computation—GECCO 2003, Springer, Berlin, 2003, pp. 610–621.
[35] K.S. Won, R. Tapabrata, K. Tai, in: B. McKay (Ed.), Congress on Evolutionary Computation–CEC 2003, IEEE, Piscataway, 2003, pp. 1520–1527.
[36] H. Ulmer, F. Streichert, A. Zell, in: W. Greenwood (Ed.), Congress on Evolutionary Computation–CEC 2004, IEEE, Piscataway, 2004, pp. 1569–1576.
[37] Y.S. Ong, P.B. Nair, A.J. Keane, K.W. Wong, in: Y. Jin (Ed.), Knowledge Incorporation in Evolutionary Computation, Springer, Berlin, 2005, pp. 307–331.
[38] Z.Z. Zhou, Y.S. Ong, P.B. Nair, A.J. Keane, K.Y. Lum, IEEE Trans. Syst. Man Cybern. C: Appl. Rev. 37 (2007) 66–76.
[39] M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, K. Giannakoglou, J.J.M. Guervos, P. Adamidis, in: H.G. Beyer, J.L. Fernandez-Villacañas (Eds.), Parallel Problem Solving from Nature, Springer, Berlin, 2002, pp. 371–380.
[40] L.A. Baumes, D. Farrusseng, M. Lengliz, C. Mirodatos, QSAR Comb. Sci. 23 (2004) 767–778.
[41] D. Farrusseng, F. Clerc, C. Mirodatos, N. Azam, F. Gilardoni, J.W. Thybaut, P. Balasubramaniam, G.B. Marin, Comb. Chem. High Throughput Screen. 10 (2007) 85–97.
[42] S. Valero, E. Argente, V. Botti, J.M. Serra, P. Serna, M. Moliner, A. Corma, Comput. Chem. Eng. 33 (2009) 225–238.
[43] A.J. Brooker, J. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, M. Trosset, Struct. Multi. Optim. 17 (1998) 1–13.
[44] A. Ratle, in: A.E. Guervos, T. Bäck, M. Schoenauer, H.P. Schwefel (Eds.), Parallel Problem Solving from Nature, Springer, Berlin, 2002, pp. 87–96.
[45] Y. Jin, M. Hüsken, M. Olhofer, B. Sendhoff, in: Y. Jin (Ed.), Knowledge Incorporation in Evolutionary Computation, Springer, Berlin, 2005, pp. 281–306.
[46] H. Ulmer, F. Streichert, A. Zell, in: Y. Jin (Ed.), Knowledge Incorporation in Evolutionary Computation, Springer, Berlin, 2005, pp. 333–355.
[47] M. Holeňa, M. Baerns, in: J.N. Cawse (Ed.), Combinatorial Experimental Design, John Wiley, New York, 2003, pp. 163–202.
[48] D. Farrusseng, C. Klanner, L. Baumes, M. Lengliz, C. Mirodatos, F. Schüth, QSAR Comb. Sci. 24 (2005) 78–93.
[49] A. Tompos, J.L. Margitfalvi, E. Tfirst, L. Végvári, Appl. Catal. A: Gen. 303 (2006) 72–80.
[50] P. Serna, L.A. Baumes, M. Moliner, A. Corma, J. Catal. 258 (2008) 25–34.
[51] B. Schölkopf, A.J. Smola, Learning with Kernels, MIT Press, Cambridge, 2002.
[52] L.A. Baumes, J.M. Serra, P. Serna, A. Corma, J. Comb. Chem. 8 (2006) 583–596.
[53] J.M. Serra, L.A. Baumes, M. Moliner, P. Serna, A. Corma, Comb. Chem. High Throughput Screen. 10 (2007) 13–24.
[54] L. Breiman, J.H. Friedman, R.A. Olshen, C.J. Stone, Classification and Regression Trees, Wadsworth, Belmont, 1984.
[55] C. Klanner, Evaluation of descriptors for solids, Thesis, Ruhr-University, Bochum, 2004.
[56] C. Klanner, D. Farrusseng, L. Baumes, M. Lenliz, C. Mirodatos, F. Schüth, Angew. Chem. Int. Ed. 43 (2004) 5347–5349.

[57] T. Cukic, R. Kraehnert, M. Holeňa, D. Herein, D. Linke, U. Dingerdissen, Appl. Catal. A: Gen. 323 (2007) 25–37.
[58] S. Moehmel, N. Steinfeldt, S. Engelschalt, M. Holeňa, S. Kolf, M. Baerns, U. Dingerdissen, D. Wolf, R. Weber, M. Bewersdorf, Appl. Catal. A: Gen. 334 (2008) 73–83.
[59] M. Holeňa, Neural Comput. 18 (2006) 2813–2853.
[60] T. Bartz-Beielstein, Experimental Research in Evolutionary Computation, Springer, Berlin, 2006.
[61] Y. Freund, R. Schapire, J. Comput. Syst. Sci. 55 (1997) 119–139.
[62] J. Friedman, Ann. Stat. 29 (2001) 1189–1232.
[63] D.L. Shrestha, Neural Comput. 18 (2006) 1678–1710.
[64] H. Drucker, in: D.H. Fisher (Ed.), Proceedings of the 14th International Conference on Machine Learning, Morgan Kaufmann, San Francisco, 1997, pp. 107–115.
[65] Y. Freund, R. Schapire, in: P. Vitányi (Ed.), Computational Learning Theory, Springer, Berlin, 1995, pp. 23–37.
[66] H. Altinçay, Pattern Anal. Appl. 7 (2004) 285–295.
[67] M. Holeňa, M. Baerns, Catal. Today 81 (2003) 485–494.
[68] M. Holeňa, M. Baerns, in: G. Ertl, H. Knözinger, F. Schüth, J. Weitkamp (Eds.), Handbook of Heterogeneous Catalysis, Wiley-WCH, Weinheim, 2008, pp. 66–81.
[69] G. Rothenberg, Catal. Today 137 (2008) 2–10.
[70] M. Baerns, M. Holeňa, Combinatorial Development of Solid Catalytic Materials, Imperial College Press, London, 2009.
[71] F. Clerc, Optimization and data mining for catalysts library design, Thesis, Université Lyon 1, 2006.