Automatically Generated Problem-Tailored Genetic Algorithms for the Optimization of Chemical Processes

M. HOLEŇA[*], U. RODEMERCK, T. CUKIC, D. LINKE, U. DINGERDISSEN
Leibniz Institute for Catalysis, Branch Berlin,
Richard-Willstätter-Str.12, 12489 Berlin, GERMANY
[*] corresponding author, second address: Institute of Computer Science, Academy of Sciences,
Pod vodárenskou věží 2, 18207 Praha , CZECH REPUBLIC

Abstract: The performance of chemical processes is often determined by the selectivity and activity of catalysts used in them. Optimizing a catalyst accordingly leads to a high-dimensional constraint optimization task for both continuous and discrete variables. To solve that task, genetic algorithms are most frequently used in catalysis, though their routine use is still hindered by a lack of appropriate implementations. Generic implementations of this method do not address all features of the optimization task, and use low-level coding of input variables, which is unacceptable for chemists. On the other hand, specific algorithms developed directly for the optimization of catalytic materials are usable only for a narrow spectrum of problems. This paper presents an approach the main idea of which is to automatically generate problem-tailored genetic algorithms from requirements concerning the optimized materials. For a specification of those requirements, a formal description language has been developed. To automatically generate corresponding algorithms from the formal descriptions, a program generator is needed. In this paper, the requirements expressible with the description language are reviewed and an overall scheme of the approach is outlined. Finally, a first prototype of a program generator for algorithms generated from that language is sketched.

Key-Words: Computer applications in chemistry, Optimization methods, Empirical objective function, Genetic algorithms, Problem-tailoring, Formal description language, Program generator

1. Introduction

In chemistry, much effort is devoted to increasing the performance of industrially important chemical processes, i.e., to achieving a higher yield of the desired reaction products without higher material or energy costs. Over 90% of the processes use a catalyst to speed upp the reaction or to improve its selectivity to the desired products. Catalyst are materials that decrease the energy needed to activate a chemical reaction without being themselves consumed in it. Catalytic materials often consist of several components with different purpose in the catalytic process to increase their functionality. The components typically can be selected from among many substances. Chemical properties of those substances usually constrain the possible ratios of their proportions, but since the proportions are continuously-valued, they still allow for an infinite number of catalyst compositions. Moreover, the catalyst can usually be prepared from the individual components in a number of ways, and the preparation method also influences the performance of the catalytic chemical process. Consequently, the search for catalysts leading to possibly high performance entails 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* 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 [5,14], and stochastic ones, such as simulated annealing [1,11], or genetic and other evolutionary algorithms [6,12,13,15,16]. Especially *genetic algorithms (GA)* 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 way how the catalysts proposed by that algorithm are subsequently tested – namely, channels of a high-throughput reactor. Nevertheless, a lack of appropriate implementations still hinders them to be routinely used. Generic GA implementations do not address all the above features (i.)–(iv.), and the low-level coding of input variables makes them unacceptable for chemists. On the other hand, first experience with GA implemented specifically for the optimization of catalytic materials shows that such algorithms are usable only for a narrow spectrum of problems and have to be reimplemented for other problems.

In this paper, we present an approach that can solve those difficulties: *automatically generated problem-tailored GA* for the optimization of catalytic materials from requirements concerning materials under consideration. For a formal specification of those requirements, we use the Catalyst Description Language developed at the Leibniz Institute for Catalysis (LIKat) in Berlin. Automatic generation of problem-tailored GA requires to first implement a sophisticated program generator, which translates formal descriptions into the corresponding GA. At LIKat Berlin, a prototype program generator of that kind has just been developed and is currently entering the early testing phase.

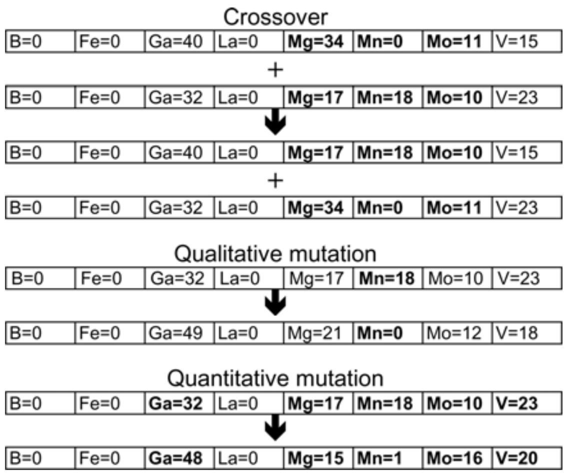
In the next section, theoretical principles of GA are recalled and an overview of using them in the optimization of catalytic materials is given. Main requirements expressible with the Catalyst Description Language are reviewed and an overall scheme of our approach is outlined in Section 3. Finally, the first implementation of a program generator for GA generated from that description language is sketched in Section 4.

2. Genetic Algorithms and Their Use in Catalyst Optimization

The term "genetic algorithms" refers to the fact that their particular way of incorporating random influences into the optimization process has been inspired by the *biological evolution of a genotype* [3,7-10]. Basically, that way consists in:

- randomly exchanging coordinates of points between two particular points in the input space of the optimized function (*recombination, crossover*),
- randomly modifying coordinates of a particular point in the input space of the optimized function (*mutation*),
- *selecting* the points for crossover and mutation according to a probability distribution, either uniform or skewed towards points at which the optimized function takes high values (the latter being a probabilistic expression of the survival-of-the-fittest principle).

In the context of catalytic materials, it is useful to differentiate between *quantitative mutation*, which modifies merely the proportions of substances already present in the material and *qualitative mutation*, which enters new substances or removes present ones (Fig. 1).



Crossover

B=0	Fe=0	Ga=40	La=0	**Mg=34**	**Mn=0**	**Mo=11**	V=15
B=0	Fe=0	Ga=32	La=0	**Mg=17**	**Mn=18**	**Mo=10**	V=23
B=0	Fe=0	Ga=40	La=0	**Mg=17**	**Mn=18**	**Mo=10**	V=15
B=0	Fe=0	Ga=32	La=0	**Mg=34**	**Mn=0**	**Mo=11**	V=23

Qualitative mutation

B=0	Fe=0	Ga=32	La=0	Mg=17	**Mn=18**	Mo=10	V=23
B=0	Fe=0	Ga=49	La=0	Mg=21	**Mn=0**	Mo=12	V=18

Quantitative mutation

B=0	Fe=0	**Ga=32**	La=0	**Mg=17**	**Mn=18**	**Mo=10**	**V=23**
B=0	Fe=0	**Ga=48**	La=0	**Mg=15**	**Mn=1**	**Mo=16**	**V=20**

Fig. 1 Illustration of operations used in genetic algorithms; the values in the examples are molar proportions (in %) of oxides of the indicated elements in the active component of the catalyst

The difference between qualitative and quantitative mutation is due to the specific meaning of the coordinates of points in the optimization of catalytic materials. However, it is not possible to express that meaning using *generic GA software*, such as the Genetic Algorithm and Direct Search Toolbox of Matlab [2]. Indeed, generic GA software encodes coordinates of points with low-level data types, typically with real numbers or bit-strings. But to encode the composition of catalysts and their preparation solely with low-level data types is tedious, error-prone, and requires a great deal of mathematical erudition. Therefore, it is unacceptable for chemists to encode their optimization tasks in that way. This difficulty, together with the fact that generic GA software does not address all the above features (i.)–(iv.) of the considered optimization tasks gave rise to the development of *specific genetic algorithms for the optimization of catalytic materials* [6,12,13,15,16]. However, experience gained with those algorithms shows that they bring another kind of difficulties. Namely, any design decision made when a specific algorithm is implemented restricts the spectrum of problems for which the implementation can be subsequently employed. This can be a serious disadvantage since the change of focus to other kinds of catalytic materials and the emergence of new catalyst preparation methods may substantially decrease the usefulness of a specific GA implementation after several years.

Main objective of this paper is to show that it is possible to tackle problems connected with a specific GA without having to resort to general GA software. The approach we propose is to replace an in advance developed specific GA implementation with an *implementation that is automatically generated just immediately before it is used to solve a particular optimization task*. Since at that time, all requirements concerning the problem are already known, this approach enables the GA implementations to be *precisely problem-tailored*. Our approach is presented in the following section.

3. Generating Problem-Tailored GA from Catalyst Descriptions

To automatically generate problem-tailored GA implementations, we need a *program generator*, i.e., a software system that transforms given requirements to an executable program. Differently to a human programmer, a program generator needs the requirements to be expressed in a rigorously formal way. To this end, we use a *Catalyst Description Language (CDL)*, developed at LIKat Berlin [4]. The language allows to express a broad variety of user requirements on the catalytic materials to be sought by the genetic algorithm, as well as on the algorithm itself (Fig. 2).

```
⋮
FeedbackTable GlobalParameter example
FeedbackField GlobalParameter fitness
example_catalyst ComposedOf support InProportion support_fraction,
& support_dopants InProportion support_dopants_fraction, Pd InProportion
& Pd_fraction, Pt InProportion Pt_fraction, dopants InProportion
& dopants_fraction PreparedUsing  dopants_preparation
support_fraction FromInterval 80,100 WithPrecision 1
support_dopants_fraction FromInterval 0,20 WithPrecision 0.1
Pd_fraction OneOf 0.5,1
Pt_fraction OneOf 0,0.2,0.4,0.6,0.8,1
dopants_fraction FromInterval 0,1.2 WithPrecision 0.01
dopants_preparation OneOf sequential, intermediate_drying
support_fraction + support_dopants_fraction + Pd_fraction + Pt_fraction
& + dopants_fraction = 1
support OneOf silica, alumina, titania, zirconia, magnesia, ceria
support_dopants ComposedOf number_of_support_dopants FromAmong
& B InProportion B_fraction, Ga InProportion Ga_fraction, Ce
& InProportion Ce_fraction, Fe InProportion Fe_fraction, Mn InProportion
& Mn_fraction, La InProportion La_fraction, Mo InProportion Mo_fraction,
& Cr InProportion Cr_fraction, V InProportion V_fraction
number_of_support_dopants OneOf 0,1,2
B_fraction FromInterval 1,10 WithPrecision 0.1
Ga_fraction FromInterval 1,10 WithPrecision 0.1
Ce_fraction FromInterval 1,10 WithPrecision 0.1
Fe_fraction FromInterval 1,10 WithPrecision 0.1
Mn_fraction FromInterval 1,10 WithPrecision 0.1
La_fraction FromInterval 1,10 WithPrecision 0.1
Mo_fraction FromInterval 1,10 WithPrecision 0.1
Cr_fraction FromInterval 1,10 WithPrecision 0.1
V_fraction FromInterval 1,10 WithPrecision 0.1
B_fraction + Ga_fraction + Ce_fraction + Fe_fraction + Mn_fraction +
& La_fraction + Mo_fraction + Cr_fraction + V_fraction =
& support_dopants_fraction
silica IsPrimitive
alumina IsPrimitive
titania IsPrimitive
zirconia IsPrimitive
magnesia IsPrimitive
ceria IsPrimitive
          ⋮
```

Fig. 2 Example fragment of a CDL-description (description keywords are in boldface)

Most important among them are the following requirements:

a) *Which substances should form the pool* from which the various components of the catalyst are selected.

b) In which hierarchy should the components from the pool be organized (Fig. 3). CDL allows the specified hierarchy (called "*ComposedOf hierarchy*") to be arbitrarily complex. At the highest level, it contains general types of components, such as active components, support, or dopants. Each of them may have its own subtypes, those again subsubtypes, etc.

c) The *number of components that may be simultaneously present* in an individual catalyst, as well as the number of simultaneously present components of a particular type from the component types hierarchy (e.g., the catalyst should contain 5 components altogether, 3 of which should belong to active components).

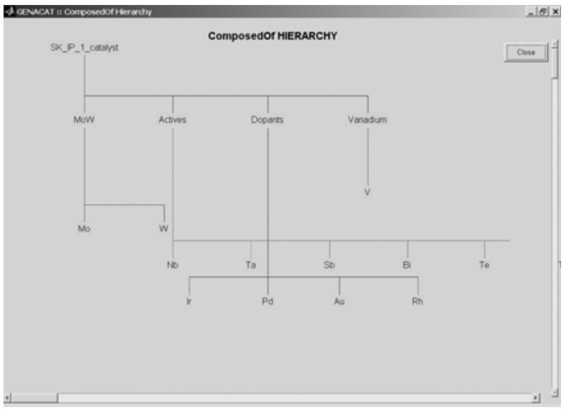


Fig. 3 Example hierarchy of component types, as depicted in the graphical user interface of the developed program generator

d) A *lower and an upper bound for the number of simultaneously present components* (e.g., the catalyst should contain 4-6 components altogether, 2-4 of which should belong to active components and 0-2 to dopants). By default, all numbers within those bounds occur with the same probability, but this can be changed through specifying ratios of probabilities of their occurrence.

e) *Proportion* of a component or a type from the component types hierarchy in the catalyst, or its proportion within (another) type from the hierarchy (e.g., proportion of the support in the catalyst, or proportion of a particular active component among all active components).

f) A *lower and an upper bound for the proportion* of a component (e.g., proportion of the support in the catalyst is 75 % - 80 %, or proportion of a particular active component among all active components is 20 % - 100 %).

g) *Linear equality or inequality constraints* for any quantities (e.g., proportion of Mg among all active components + proportion of Mn among all active components = 50 %, lower bound for the proportion of active components > 5*upper bound for the proportion of dopants). Each such constraint may contain an arbitrary number of quantities, and each quantity may occur in an arbitrary number of constraints.

h) The *choice among several possibilities* for the preparation of the catalyst, of any component, and of any type from the component types hierarchy. In addition, any step of the preparation (e.g., precipitation, calcination), and any of its features can be again chosen among several possibilities. In this way, the component types hierarchy is complemented with an arbitrarily complex hierarchy of choices. Moreover, that hierarchy can be applied also to quantities (e.g., the number of components simultaneously present in the material is 2, 4, or 6).

i) Population size of the current generation.

j) Descriptive information about the current experiment.

k) Which *particular implementation of the genetic operations* selection, crossover and mutation should be employed.

l) Which particular parts of the algorithm output should be *stored in the database*, and in which *tables and fields* should they be stored.

m) *Precision* with which any part of the algorithm output should be stored in the database. The generated algorithm then during its search for new catalytic materials avoids finding those that within the given precision already exists in the database.

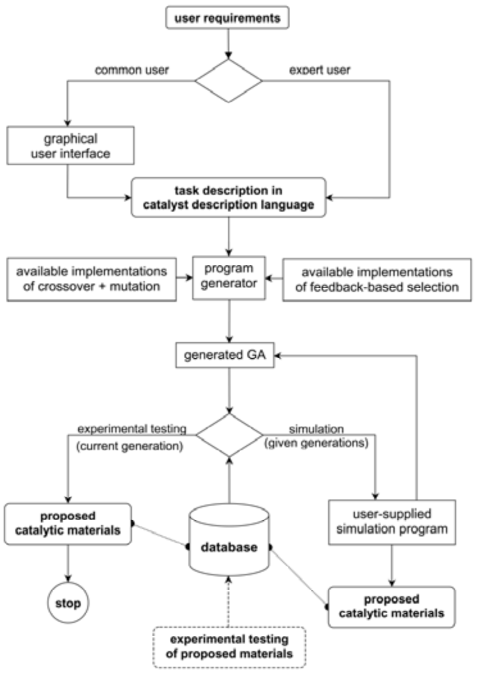

user requirements
common user
expert user
graphical user interface
task description in catalyst description language
available implementations of crossover + mutation
program generator
available implementations of feedback-based selection
generated GA
experimental testing (current generation)
simulation (given generations)
proposed catalytic materials
database
user-supplied simulation program
stop
proposed catalytic materials
experimental testing of proposed materials

Fig. 4 Scheme of the proposed approach to generating problem-tailored genetic algorithms according to CDL descriptions

An overall scheme of the proposed approach is depicted in Fig. 4. The program generator accepts text files with CDL descriptions as input, and produces GA implementations as output. For the approach, it is immaterial how such an implementation looks like. It can be programmed in various languages; it can be a stand-alone program or can combine calls to generic GA software with parts implementing the functionality that the generic software does not cover. If the values of the objective function have to be obtained through experimental testing, then the GA implementation runs only once and then it exits. However, the approach previews also the possibility to obtain those values from some simulation program instead. Then the GA implementation alternates with that program for as many generations as desired.

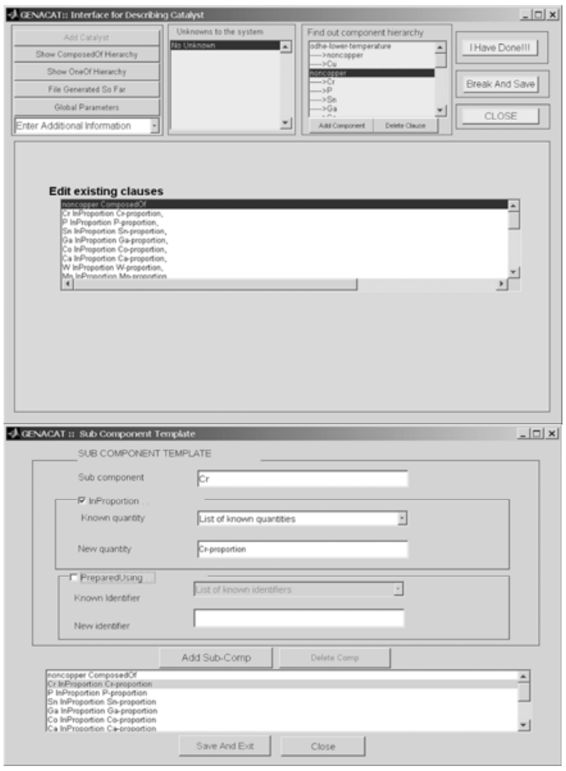
Finally, the approach previews the possibility to place between the user and the program generator a graphical interface, the purpose of which is to remove from the users the necessity to write files with CDL descriptions manually, and the necessity to understand the CDL language and its syntax. To this end, the interface provides a *series of windows* through which a user can enter all the information needed to create a complete CDL description.

4. Implementation

The first prototype of a system generating problem-tailored GA has just been developed at LIKat Berlin and currently enters the early testing phase. It has been developed in Matlab; also the generated GA implementations are in Matlab and make use of its Genetic Algorithm and Direct Search Toolbox. The system follows the approach outlined in the previous section, including the graphical interface between the user and the program generator (Fig. 5). In addition, also the generated GA implementations contain a simple graphical interface, which shows the progress of the performed optimization, and allows to decide whether to run the implementation only once or which external program to use for simulation (Fig. 6).

5. Conclusions

The paper presents a novel approach to the optimization of the main factor influencing the performance of chemical processes – the catalysts used in them. The objective of that approach is to preserve the advantage of specific genetic algorithms for the optimization of catalytic materials, namely a chemically meaningful and mathematically undemanding way of formulating the optimization task, while at the same time dealing with its disadvantage – the narrow spectrum of problems for which such an algorithm can be employed. To achieve this objective, we propose to automatically generate problem-tailored GA from requirements expressed in some formal description language. The feasibility of the proposed approach has been partially confirmed through developing such a language, though completely can it be confirmed only through experience with a system implementing the approach, i.e., with a program generator for specific GA implementations. A prototype of a program generator has been developed at LIKat Berlin and is entering the early testing phase. Not surprisingly, its development has been much more demanding and time-consuming than the development of any particular specific GA implementation. From a long-time point of view, however, it actually saves development efforts. In addition, it tackles a much broader variety of catalyst optimization tasks than any specific GA could ever do.



No Unknown

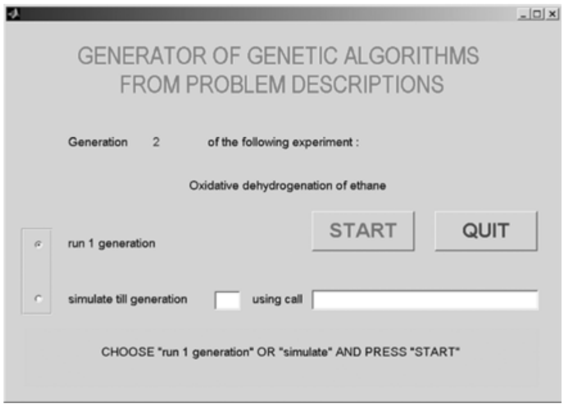
odhe-lower-temperature
----->noncopper
----->Cu
noncopper
----->Cr
----->P
----->Sn
----->Ga

noncopper ComposedOf
Cr InProportion Cr-proportion,
P InProportion P-proportion,
Sn InProportion Sn-proportion,
Ga InProportion Ga-proportion,
Co InProportion Co-proportion,
Ca InProportion Ca-proportion,
W InProportion W-proportion,
Mn InProportion Mn-proportion

Sub component: Cr
[✓ InProportion]
Known quantity: List of known quantities
New quantity: Cr-proportion
[ ] PreparedUsing
Known Identifier: List of known identifiers
New identifier:

noncopper ComposedOf
Cr InProportion Cr-proportion
P InProportion P-proportion
Sn InProportion Sn-proportion
Ga InProportion Ga-proportion
Co InProportion Co-proportion
Ca InProportion Ca-proportion

Fig. 5 Two example windows of the graphical interface allowing users to enter the information needed to create a CDL description



GENERATOR OF GENETIC ALGORITHMS FROM PROBLEM DESCRIPTIONS

Generation 2 of the following experiment :

Oxidative dehydrogenation of ethane

CHOOSE "run 1 generation" OR "simulate" AND PRESS "START"

Fig. 6 Simple graphical user interface of the generated GA implementations

Acknowledgement
Financial support by the Federal Ministry for Education and Research of the Federal Republic of Germany (project nr. 03X2003B) is gratefully acknowledged.

References:

1. A. Eftaxias, J. Font, A. Fortuny et al., Kinetic modelling of catalytic wet air oxidation of phenol by simulated annealing. *Applied Catalysis B: Environmental*, 33(2001), 175–190.
2. *Genetic Algorithm and Direct Search Toolbox*. The MathWorks, Inc., 2004.
3. D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.
4. M. Holeňa, *Description language for catalyst search with evolutionary methods. Version 2.0*, Leibniz Institute for Catalysis, Berlin, 2005.
5. A. Holzwarth, P. Denton, H. Zanthoff et al., Combinatorial approaches to heterogeneous catalysis: strategies and perspectives for academic research. *Catalysis Today*, 2001, 67 (2001), 309–318.
6. K. Huang, X.L. Zhan, F.Q. Chen et al., Catalyst design for methane oxidative coupling by using artificial neural network and hybrid genetic algorithm. *Chemical Engineering Science*, 58(2003), 81–87.
7. J.R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, 1992.
8. J.R. Koza, *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, 1994.
9. J.R. Koza, F.H. Bennett, D. Andre et al., *Genetic Programming III: Darwinian Invention and Problem Solving*. Academic Press, 1999.
10. J.R. Koza, M.A. Keane, M.J. Streeter et al., *Genetic Programming IV: Routine human-competitive machine intelligence*, Kluwer, 2003.
11. B. Li, P. Sun, Q. Jin et al., A simulated annealing study of Si, Al distribution in the omega framework. *Journal of Molecular Catalysis A: Chemical*, 148(1999), 189–195.
12. R.M. Pereira, F.Clerc, D. Farrusseng et al., Effect of the Genetic Algorithm Parameters on the Optimisation of Heterogeneous Catalysts. *QSAR and Combinatorial Science*, 24(2005), 45–57.
13. U. Rodemerck, M. Baerns, M. Holeňa et al., Application of a genetic algorithm and a neural network for the discovery and optimization of new solid catalytic materials. *Applied Surface Science*, 223 (2004), 168–174.
14. L. Végvári, A. Tompos, S. Göbölös et al., Holographic research strategy for catalyst library design. Description of a new powerful optimization method. *Catalysis Today*, 81(2003), 517–527.
15. Y. Watanabe, T. Umegaki, M. Hashimoto et al., Optimization of Cu oxide catalysts for methanol synthesis by combinatorial tools using 96 well microplates, artificial neural network and genetic algorithm. *Catalysis Today*, 89(2004), 455–464.
16. D. Wolf, O.V. Buyevskaya, M. Baerns, An evolutionary approach in the combinatorial selection and optimization of catalytic materials. *Applied Catalyst A: General*, 200 (2000), 63–77.