Optimization of Catalysts Using Specific, Description-Based Genetic Algorithms

Martin Holena,* Tatjana Cukic, Uwe Rodemerck, and David Linke

Leibniz Institute for Catalysis, Branch Berlin, Richard-Willstätter-Strasse 12, 12489 Berlin, Germany

Received June 21, 2007

This paper deals with the key optimization task that has to be solved when improving the performance of many chemical processes—optimization of the catalysts used in the reaction via the optimization of its composition and preparation. A novel approach is presented that allows for the preservation of the advantages of genetic algorithms developed specifically for the optimization of catalytic materials but avoids the disadvantageous necessity to reimplement the algorithm when the scope of the optimized materials changes. Its main idea is to automatically generate problem-tailored implementations from requirements concerning the materials with a program generator. For the specification of such requirements, a formal description language, called catalyst description language, has been developed.

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 specrum of different preparation methods are tested. Consequently, the optimum has to be searched in a high-dimensional descriptor space; nowadays, 20−30 descriptors are not exceptional. For high-dimensional spaces, the traditional methods of design of experiments 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 (Figure 1). 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 optimized performance indicator of the catalyst (such as yield, activity, or conversion) 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,[2,3] and stochastic ones, such as simulated annealing[4,5] or genetic and other evolutionary algorithms.[6−10] Especially *genetic algorithms (GA*) have become quite popular in the search for optimal catalytic materials, mainly due to the possibility of establishing 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. Nevertheless, a lack of appropriate implementations still hinders them to be routinely used. Generic GA implementations do not address all of the above features (i)−(iv), and the low-level coding of input variables makes them unacceptable for chemists. On the other hand, the first experience with GA implemented specifically for the optimization of catalytic materials shows that such algorithms are usable only for a narrow spectrum

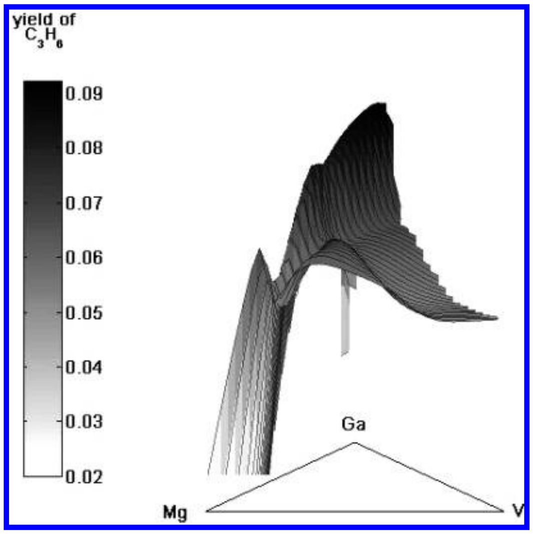


Figure 1. Example performance function in the field of catalytic materials. Yield as a function of the fractions of Mg, V, and Ga in the oxidative dehydrogenation of propane. Reprinted with permission from Holena, M.; Baerns, M. In *Handbook of Heterogeneous Catalysis*; YEAR, VOLUME, PAGES. Copyright YEAR Wiley-WCH.

* Corresponding author e-mail: martin@cs.cas.cz. Additional address: Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic.

10.1021/ci700218p CCC: $40.75 © 2008 American Chemical Society
Published on Web 02/07/2008

of problems and have to be reimplemented for other problems.[11]

In this paper, we present an approach that can solve those difficulties: *automatically generated problem-tailored GA* for the optimization of catalytic materials from user requirements. For a formal specification of such requirements, we use the Catalyst Description Language developed at the Leibniz Institute for Catalysis (LIKAT) in Berlin. Automatic generation of problem-tailored GA requires first implementing 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 in the testing phase.

In the next section, theoretical principles of GA are recalled, and their use in the optimization of catalytic materials is discussed. The following, core section of the paper, proposes to automatically generate problem-tailored GA from catalyst descriptions. It also explains the syntax and scope of the Catalyst Description Language. The paper then concludes with sections describing the functionality of such a program generator and its prototype implementation.

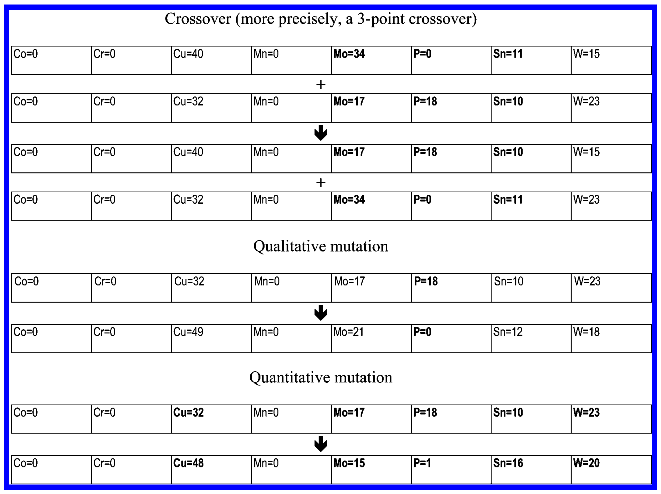
A preliminary presentation of the approach, corresponding to an early stage of its development, has been published in the chapter[11] of the Wiley-VCH book "High-Throughput Screening in Chemical Catalysis". A technically oriented presentation of the approach from the point of view of computer scientists has been recently given at the Applied Computer Science 2006 conference.[12] Compared to those publications, the present paper describes the final state of the approach and from the point of view of its ultimate objective—the search for optimal catalytic materials.

GENETIC ALGORITHMS AS AN OPTIMIZATION METHOD AND THEIR SPECIFICITY IN CATALYST OPTIMIZATION

GA Principles. Genetic algorithms are a *stochastic optimization method*. This means that when searching for maxima or minima of the objective function, the available information about its response surface is complemented with random influences. 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*. Basically, that way consists of the following: (i) randomly exchanging coordinates of points between two particular points in the input space of the objective function (*recombination*, *crossover*), (ii) randomly modifying coordinates of a particular point in the input space of the objective function (*mutation*), and (iii) *selecting* the points for crossover and mutation according to a probability distribution, either uniform or skewed toward points at which the objective function takes high values (the latter being a probabilistic expression of the survival-of-the-fittest principle from biological evolution).

GA Features Important for Catalysis. Detailed treatment of various kinds of genetic algorithms as well as of other evolutionary optimization methods can be found in specialized monographs.[13−17] Here, only those features of GA will be pointed out that are particularly important for the optimization of catalytic materials:

1. What is evolved with a genetic algorithm or, equivalently, what is the meaning of the individual coordinates of points in the input space of the objective function is *strongly problem-dependent*. In catalysis, the coordinates typically convey some of the following meanings: (*i*) *Qualitative composition* of the catalyst, i.e., which active components does it contain, whether it contains dopants, and which ones, whether it is supported, and what is its support. (*ii*) *Quantitative composition* of the catalyst, i.e., the proportion of active components, support and dopants within the catalyst, the proportions of individual active components among all active components (or alternatively, their proportions within the whole catalyst), etc. (*iii*) *Procedure by which the catalyst is prepared*, its individual steps, and their quantitative specification. (*iv*) *Reaction conditions* of the catalyzed reaction.



Crossover (more precisely, a 3-point crossover)

Co=0	Cr=0	Cu=40	Mn=0	**Mo=34**	**P=0**	**Sn=11**	W=15

+

Co=0	Cr=0	Cu=32	Mn=0	**Mo=17**	**P=18**	**Sn=10**	W=23

🡻

Co=0	Cr=0	Cu=40	Mn=0	**Mo=17**	**P=18**	**Sn=10**	W=15

+

Co=0	Cr=0	Cu=32	Mn=0	**Mo=34**	**P=0**	**Sn=11**	W=23

Qualitative mutation

Co=0	Cr=0	Cu=32	Mn=0	Mo=17	**P=18**	Sn=10	W=23

🡻

Co=0	Cr=0	Cu=49	Mn=0	Mo=21	**P=0**	Sn=12	W=18

Quantitative mutation

Co=0	Cr=0	**Cu=32**	Mn=0	**Mo=17**	**P=18**	**Sn=10**	**W=23**

🡻

Co=0	Cr=0	**Cu=48**	Mn=0	**Mo=15**	**P=1**	**Sn=16**	**W=20**

Figure 2. 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 part of the catalyst. The elements affected by the respective operation are in bold (following ref 11).

There is an intimate connection between (i) and (ii): The presence of a particular component in the catalyst is equivalent to the proportion of that component being nonzero. That has deep consequences for the algorithms accomplishing the operations of recombination and mutation. Namely, those algorithms have to guarantee that this equivalence cannot get invalidated through the respective operation. For example, if a coordinate is exchanged during recombination, and that coordinate conveys the presence of a particular component in one of the parent catalysts and the absence of that component in the other one, then at the same time also the coordinate stating the proportion of that component has to be exchanged. Similarly, if a mutation of a particular coordinate takes some component off the catalyst, then at the same time the coordinate stating the proportion of that component has to be set to zero. 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 catalyst, and *qualitative mutation*, which enters new substances or removes present ones (Figure 2).

2. The crossover and mutation operations provide an excellent possibility for the genetic algorithm to be applied to more individuals at each iteration, thus to *follow many optimization paths in parallel*, a synchronization among which is needed only after each iteration. Because of the biological inspiration of genetic algorithms, individual itera-

tions of a genetic algorithm are called *generations*, and all points in which the value of the objective function is considered in a generation (in particular, all catalysts the performance of which has been measured in that generation) are denoted as *population* of that generation. The fact that genetic algorithms follow many optimization paths in parallel is actually the main reason for their attractiveness in high-throughput catalysis because it is then possible to establish a straightforward correspondence between those optimization paths and the technical realization of the experiment (e.g., channels of the high-throughput reactor in which the catalysts are experimentally tested).

3. Since they do not use derivatives, GA are not attracted to local optima. On the contrary, the random variables incorporated into recombination, mutation, and selection enable the optimization paths to *leave the attraction area of the nearest local optimum* and to continue *searching for a global one*.

4. Due to the biological inspiration of GA, a particular distribution of the incorporated random variables cannot be justified mathematically, but its choice is a heuristic task. The most important heuristic parameters describing that distribution are the *overall probability of any modification* of an individual, the *ratio between the conditional probabilities* of crossover and qualitative or quantitative mutation, conditioned on any modification, and the *distribution of the intensity of quantitative mutation*. Also the population size is sometimes a matter of heuristic choice, though in catalysis it is usually determined by the number of channels in the reactor in which the catalysts are tested (taking into account the possibility of using the reactor for a single population several times).

Motivation for our Research. The difference between qualitative and quantitative mutation pertains 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 general GA software, such as the Genetic Algorithm and Direct Search Toolbox of Matlab.[18] Indeed, such generic GA software encodes coordinates of points by means of low-level data types, typically by means of real numbers or bit-strings. And to encode the composition of catalytic materials, their preparation and reaction conditions solely with low-level data types is tedious, error-prone, and requires a great deal of mathematical erudition.

For this reason and due to the fact that generic GA software insufficiently addresses especially the features (ii) and (iv) of the above optimization task, it is not surprising that, apart from early attempts to use generic GA software to optimize the distribution of active sites of two catalytic components,[19,20] the application of GA in this area has followed the way of developing specific algorithms for the optimization of catalytic materials. The first such algorithm for solid catalyst development was implemented by D. Wolf et al. in the late 1990s,[6,21,22] followed by a modified version in 2002;[8,23] similar algorithms were later developed at other institutions.[9,10,24−28]

However, experience gained with those algorithms shows that they bring another kind of problems. Namely, any design decision made when a specific genetic algorithm is implemented restricts the class 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. To keep the usefulness of the implementation high for a long time would require to preview, during its development, all reactions, catalytic materials, and their preparation methods for which the implemented algorithm could need to be employed in the future. Nevertheless, there is no guarantee that all of them really can be previewed at the development time. In addition, the more possibilities the implementation of a genetic algorithm attempts to cover, the broader the class of possible distributions of the incorporated random variables, thus according to 4., the higher the number of possible combinations of parameters of those distributions that need to be heuristically chosen.

That is why we started, 5 years ago, to work on an approach attempting to tackle the limited usefulness of specific GA without having to resort to generic GA software. This approach consists of replacing aspecific GA developed in advance with a GA 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*. The approach is presented in the following section.

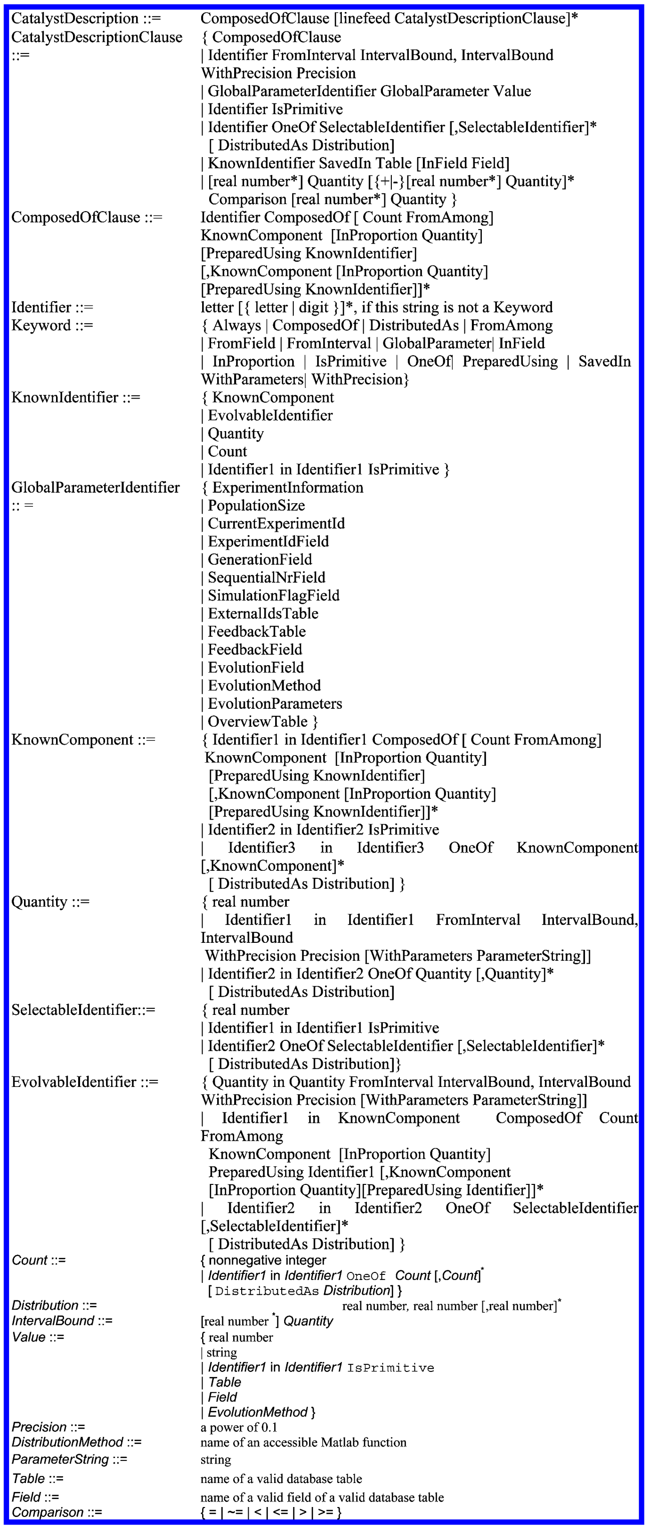
PROPOSED APPROACH, BASED ON A FORMAL DESCRIPTION LANGUAGE

Principles of the Proposed Approach. As was already mentioned in the previous section, the difficulty pertaining to specific GA in catalysis is with time decreasing usefulness of any specific genetic algorithm implementation. This difficulty inspired the basic idea of the proposed approach—to postpone the implementation of the algorithm as much as possible, i.e., to implement the algorithm just 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 GA to be precisely *problem-tailored*. In addition, it allows for the restriction of the class of possible distributions of the incorporated random variables hence also the number of possible combinations of parameters of those distributions that need to be heuristically chosen.

However, the traditional *implementation by humans is not feasible* in such a situation since it is error prone, expensive, and too *slow*. Therefore, we propose to generate problem-tailored GA implementations *automatically*. To this end, we need a *program generator*, i.e., a software system that transforms given requirements to executable programs. Different than a human programmer, a program generator needs the requirements to be expressed in a rigorously formal way. For this purpose, we use a *Catalyst Description Language (CDL)*, developed at LIKAT Berlin[11] (Figures 3 and 4).

CDL Scope. The language allows for the expression of a broad variety of user requirements on the catalytic materials to be sought by the genetic algorithm as well as on the algorithm itself. Most important among them are the following requirements:

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



```
CatalystDescription ::=        ComposedOfClause [linefeed CatalystDescriptionClause]*
CatalystDescriptionClause      { ComposedOfClause
::=                            | Identifier FromInterval IntervalBound, IntervalBound
                               WithPrecision Precision
                               | GlobalParameterIdentifier GlobalParameter Value
                               | Identifier IsPrimitive
                               | Identifier OneOf SelectableIdentifier [,SelectableIdentifier]*
                                 [ DistributedAs Distribution]
                               | KnownIdentifier SavedIn Table [InField Field]
                               | [real number*] Quantity [{+|-}[real number*] Quantity]*
                                 Comparison [real number*] Quantity }
ComposedOfClause ::=           Identifier ComposedOf [ Count FromAmong]
                               KnownComponent  [InProportion Quantity]
                               [PreparedUsing KnownIdentifier]
                               [,KnownComponent [InProportion Quantity]
                               [PreparedUsing KnownIdentifier]]*
Identifier ::=                 letter [{ letter | digit }]*, if this string is not a Keyword
Keyword ::=                    { Always | ComposedOf | DistributedAs | FromAmong
                               | FromField | FromInterval | GlobalParameter| InField
                               | InProportion | IsPrimitive | OneOf| PreparedUsing | SavedIn
                               WithParameters| WithPrecision}
KnownIdentifier ::=            { KnownComponent
                               | EvolvableIdentifier
                               | Quantity
                               | Count
                               | Identifier1 in Identifier1 IsPrimitive }
GlobalParameterIdentifier      { ExperimentInformation
:: =                           | PopulationSize
                               | CurrentExperimentId
                               | ExperimentIdField
                               | GenerationField
                               | SequentialNrField
                               | SimulationFlagField
                               | ExternalIdsTable
                               | FeedbackTable
                               | FeedbackField
                               | EvolutionField
                               | EvolutionMethod
                               | EvolutionParameters
                               | OverviewTable }
KnownComponent ::=             { Identifier1 in Identifier1 ComposedOf [ Count FromAmong]
                                KnownComponent  [InProportion Quantity]
                                 [PreparedUsing KnownIdentifier]
                                 [,KnownComponent [InProportion Quantity]
                                 [PreparedUsing KnownIdentifier]]*
                               | Identifier2 in Identifier2 IsPrimitive
                               |   Identifier3   in   Identifier3   OneOf   KnownComponent
                               [,KnownComponent]*
                                 [ DistributedAs Distribution] }
Quantity ::=                   { real number
                               |   Identifier1   in   Identifier1   FromInterval   IntervalBound,
                               IntervalBound
                                WithPrecision Precision [WithParameters ParameterString]]
                               | Identifier2 in Identifier2 OneOf Quantity [,Quantity]*
                                 [ DistributedAs Distribution]
SelectableIdentifier::=        { real number
                               | Identifier1 in Identifier1 IsPrimitive
                               | Identifier2 OneOf SelectableIdentifier [,SelectableIdentifier]*
                                 [ DistributedAs Distribution]}
EvolvableIdentifier ::=        { Quantity in Quantity FromInterval IntervalBound, IntervalBound
                               WithPrecision Precision [WithParameters ParameterString]]
                               |   Identifier1   in   KnownComponent    ComposedOf   Count
                               FromAmong
                                 KnownComponent  [InProportion Quantity]
                                 PreparedUsing Identifier1 [,KnownComponent
                                 [InProportion Quantity][PreparedUsing Identifier]]*
                               |   Identifier2   in   Identifier2   OneOf   SelectableIdentifier
                               [,SelectableIdentifier]*
                                 [ DistributedAs Distribution] }
Count ::=                      { nonnegative integer
                               | Identifier1 in Identifier1 OneOf  Count [,Count]*
                                 [ DistributedAs Distribution] }
Distribution ::=                                real number, real number [,real number]*
IntervalBound ::=              [real number *] Quantity
Value ::=                      { real number
                               | string
                               | Identifier1 in Identifier1 IsPrimitive
                               | Table
                               | Field
                               | EvolutionMethod }
Precision ::=                  a power of 0.1
DistributionMethod ::=         name of an accessible Matlab function
ParameterString ::=            string
Table ::=                      name of a valid database table
Field ::=                      name of a valid field of a valid database table
Comparison ::=                 { = | ~= | < | <= | > | >= }
```

Figure 3. Syntax of the proposed Catalyst Description Language (CDL).

(b) In which hierarchy should the components from the pool be organized (Figure 5). 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 material as well as the number of simultaneously present components of a particular type from the component types hierarchy (e.g., the material should contain 5 components altogether, 3 of which are active components).

(d) The choice among several possibilities for the *number of simultaneously present components* (e.g., the material should contain 4−6 components altogether, 2−4 of which should belong to active components and 0−2 to dopants). By default, all possibilities occur with the same probability, but this can be changed through specifying their probability distribution.

(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) In addition to the quantities introduced so far, i.e., proportions, numbers of simultaneously present elements, and lower or upper bounds, also *any additional quantities*, to describe various chemical or physical properties of the catalysts, their preparation procedure, reaction conditions, but also the behavior of the genetic algorithm itself. Also for those additional quantities, lower and upper bounds or probability distribution of the values of the quantity may be specified.

(h) *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.

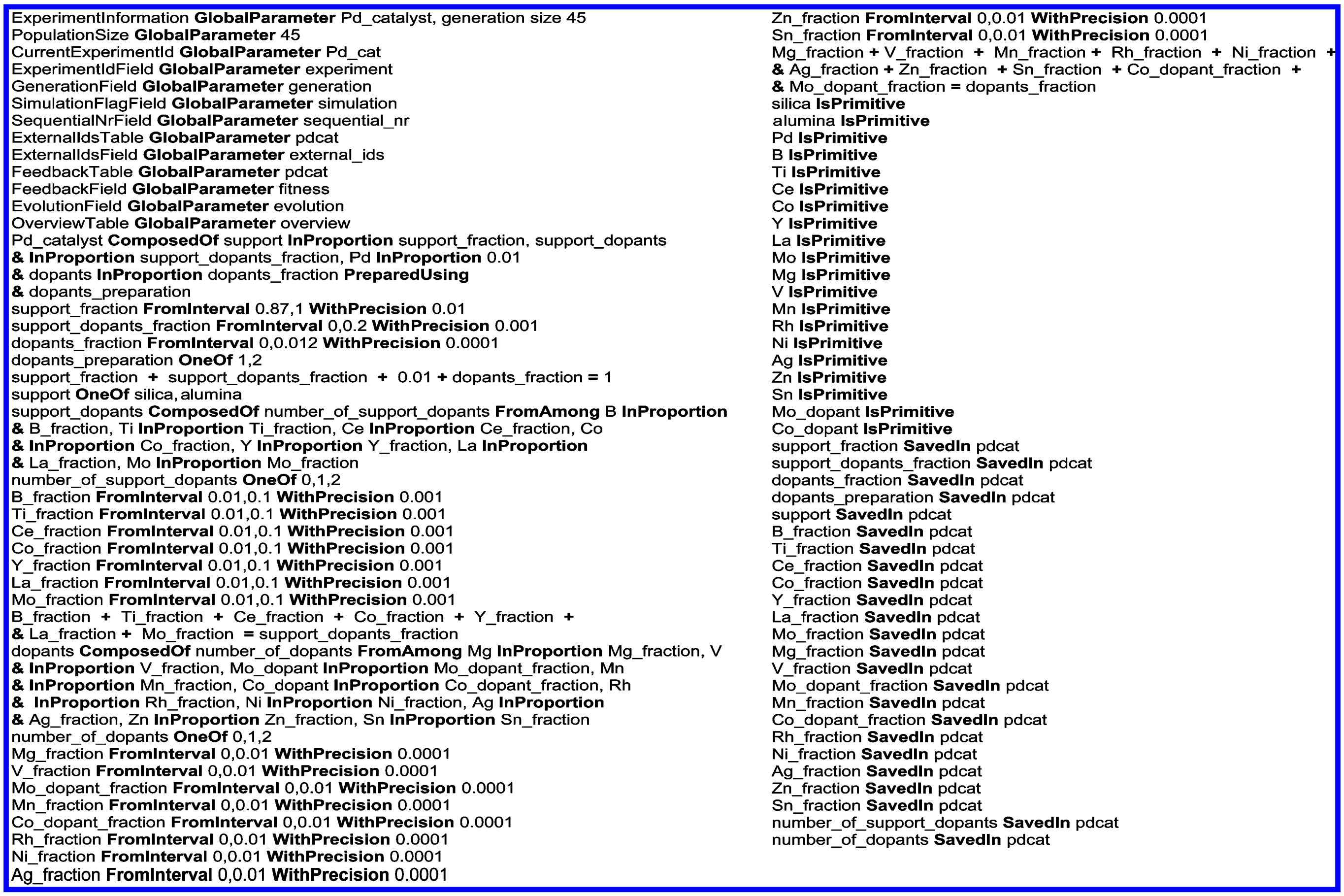
(i) The *choice among several possibilities* for the preparation of the catalyst, of any component, and of any type from the component types hierarchy. In addition, any step of the preparation (e.g., precipitation, calcination) and any of its features can be again chosen among several possibilities. 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 (cf. d).

(j) Population size of the current generation.

(k) Descriptive information about the current experiment provided by the user.

(l) In which particular way the *genetic operations* selection, crossover, and mutation should be performed.

(m) Which particular parts of the algorithm output should be *stored in the database*, and in which *tables and fields* should they be stored. However, those parts of the algorithm output that are vital for a correct function of the algorithm in subsequent generations have to be stored in the database (and the corresponding tables and fields have to be specified) anyway. These include in particular the following: (i) all output obtained using *any random choices*, such as the choice of the specified number of components from the pool, the choice among several possibilities, or the choice of the value of a quantity from the interval between a lower and an upper bound, since such output cannot be reconstructed if it has not been stored; (ii) the information needed to *uniquely*



```
ExperimentInformation GlobalParameter Pd_catalyst, generation size 45
PopulationSize GlobalParameter 45
CurrentExperimentId GlobalParameter Pd_cat
ExperimentIdField GlobalParameter experiment
GenerationField GlobalParameter generation
SimulationFlagField GlobalParameter simulation
SequentialNrField GlobalParameter sequential_nr
ExternalIdsTable GlobalParameter pdcat
ExternalIdsField GlobalParameter external_ids
FeedbackTable GlobalParameter pdcat
FeedbackField GlobalParameter fitness
EvolutionField GlobalParameter evolution
OverviewTable GlobalParameter overview
Pd_catalyst ComposedOf support InProportion support_fraction, support_dopants
& InProportion support_dopants_fraction, Pd InProportion 0.01
& dopants InProportion dopants_fraction PreparedUsing
& dopants_preparation
support_fraction FromInterval 0.87,1 WithPrecision 0.01
support_dopants_fraction FromInterval 0,0.2 WithPrecision 0.001
dopants_fraction FromInterval 0,0.012 WithPrecision 0.0001
dopants_preparation OneOf 1,2
support_fraction + support_dopants_fraction + 0.01 + dopants_fraction = 1
support OneOf silica, alumina
support_dopants ComposedOf number_of_support_dopants FromAmong B InProportion
& B_fraction, Ti InProportion Ti_fraction, Ce InProportion Ce_fraction, Co
& InProportion Co_fraction, Y InProportion Y_fraction, La InProportion
& La_fraction, Mo InProportion Mo_fraction
number_of_support_dopants OneOf 0,1,2
B_fraction FromInterval 0.01,0.1 WithPrecision 0.001
Ti_fraction FromInterval 0.01,0.1 WithPrecision 0.001
Ce_fraction FromInterval 0.01,0.1 WithPrecision 0.001
Co_fraction FromInterval 0.01,0.1 WithPrecision 0.001
Y_fraction FromInterval 0.01,0.1 WithPrecision 0.001
La_fraction FromInterval 0.01,0.1 WithPrecision 0.001
Mo_fraction FromInterval 0.01,0.1 WithPrecision 0.001
B_fraction + Ti_fraction + Ce_fraction + Co_fraction + Y_fraction +
& La_fraction + Mo_fraction = support_dopants_fraction
dopants ComposedOf number_of_dopants FromAmong Mg InProportion Mg_fraction, V
& InProportion V_fraction, Mo_dopant InProportion Mo_dopant_fraction, Mn
& InProportion Mn_fraction, Co_dopant InProportion Co_dopant_fraction, Rh
& InProportion Rh_fraction, Ni InProportion Ni_fraction, Ag InProportion
& Ag_fraction, Zn InProportion Zn_fraction, Sn InProportion Sn_fraction
number_of_dopants OneOf 0,1,2
Mg_fraction FromInterval 0,0.01 WithPrecision 0.0001
V_fraction FromInterval 0,0.01 WithPrecision 0.0001
Mo_dopant_fraction FromInterval 0,0.01 WithPrecision 0.0001
Mn_fraction FromInterval 0,0.01 WithPrecision 0.0001
Co_dopant_fraction FromInterval 0,0.01 WithPrecision 0.0001
Rh_fraction FromInterval 0,0.01 WithPrecision 0.0001
Ni_fraction FromInterval 0,0.01 WithPrecision 0.0001
Ag_fraction FromInterval 0,0.01 WithPrecision 0.0001
Zn_fraction FromInterval 0,0.01 WithPrecision 0.0001
Sn_fraction FromInterval 0,0.01 WithPrecision 0.0001
Mg_fraction + V_fraction + Mn_fraction + Rh_fraction + Ni_fraction +
& Ag_fraction + Zn_fraction + Sn_fraction + Co_dopant_fraction +
& Mo_dopant_fraction = dopants_fraction
silica IsPrimitive
alumina IsPrimitive
Pd IsPrimitive
B IsPrimitive
Ti IsPrimitive
Ce IsPrimitive
Co IsPrimitive
Y IsPrimitive
La IsPrimitive
Mo IsPrimitive
Mg IsPrimitive
V IsPrimitive
Mn IsPrimitive
Rh IsPrimitive
Ni IsPrimitive
Ag IsPrimitive
Zn IsPrimitive
Sn IsPrimitive
Mo_dopant IsPrimitive
Co_dopant IsPrimitive
support_fraction SavedIn pdcat
support_dopants_fraction SavedIn pdcat
dopants_fraction SavedIn pdcat
dopants_preparation SavedIn pdcat
support SavedIn pdcat
B_fraction SavedIn pdcat
Ti_fraction SavedIn pdcat
Ce_fraction SavedIn pdcat
Co_fraction SavedIn pdcat
Y_fraction SavedIn pdcat
La_fraction SavedIn pdcat
Mo_fraction SavedIn pdcat
Mg_fraction SavedIn pdcat
V_fraction SavedIn pdcat
Mo_dopant_fraction SavedIn pdcat
Mn_fraction SavedIn pdcat
Co_dopant_fraction SavedIn pdcat
Rh_fraction SavedIn pdcat
Ni_fraction SavedIn pdcat
Ag_fraction SavedIn pdcat
Zn_fraction SavedIn pdcat
Sn_fraction SavedIn pdcat
number_of_support_dopants SavedIn pdcat
number_of_dopants SavedIn pdcat
```

Figure 4. Simple example of a complete CDL-description. Description keywords are in boldface.

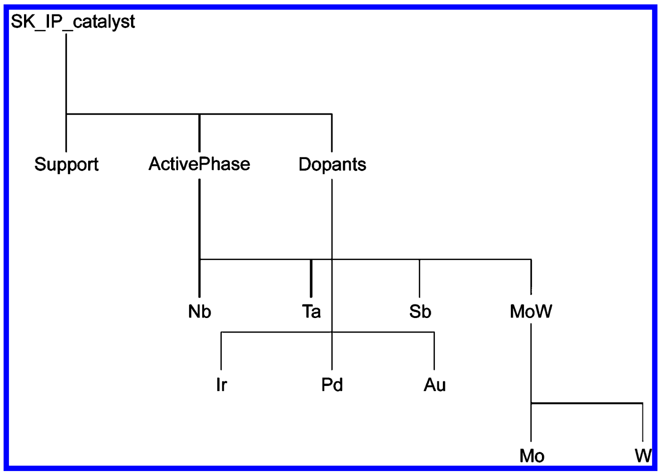


Figure 5. Example hierarchy of component types, as depicted in the graphical user interface for creating CDL descriptions.

identify a catalytic material in the database, which consists of a unique database identifier of the considered experiment, the number of the current generation, and the number of the catalyst within that generation.

(n) *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 exist in the database.

(o) Which table and field of the database contain the *actual values of the objective function*.

The Optimization Task in the Context of a CDL Description. Once the objective function has been chosen (e.g., yield, conversion), then the optimization task is fully determined as soon as we specify the set on which the optimum of that function should be searched. To specify this set in turn means to specify the following:

(i) The *meaning of the individual coordinates* of points in the set, i.e., the meaning of coordinates in the input space of the objective function, in which this set lies. Recall from the previous section that in catalysis, the coordinates convey the qualitative and quantitative composition of the catalyst, its preparation, and reaction conditions. The correspondence between them and the requirements expressible in CDL is established in Table 1.

(ii) *Delimitation of the set* on which the optimum should be searched within the space where it lies. The above overview of requirements expressible in CDL showed several possibilities how to delimit that set, in particular (ii.a) linear *equality or inequality constraints* for quantities; (ii.b) *the number of components simultaneously present* in the catalyst as well as numbers of simultaneously present components of particular types, possibly together with the choice of their values among several possibilities and sometimes also together with their probability distribution on the set of possible values; and (ii.c) *lower and upper bounds for proportions*.

We can see that the proposed language CDL is very powerful for defining optimization tasks to be solved with the generated GA. The transformation from a CDL description to a genetic algorithm inevitably requires to be generated by some program generator for that algorithm. The functionality of such a generator from the point of view of the proposed approach will be described in the following section.

Table 1. Correspondence between the Meaning Conveyed by Individual Coordinates of Points in the Input Space of the Objective Function and the Requirements Expressible in CDL

meaning conveyed by coordinates	requirement expressible in CDL
qualitative composition of the catalyst	pool of catalyst components, component types hierarchy
quantitative composition of the catalyst	proportion of a component or a type from the component types hierarchy in the catalytic material, or within a component type
catalyst preparation	choice among several possibilities for the preparation of the catalyst, of a component or a type, or choice among several possibilities for a preparation step or feature
reaction conditions of the catalyzed reaction	additional quantities, together with a choice among several possibilities for their values, optionally with their lower and upper bounds

FUNCTIONALITY OF A GENERATOR FOR PROBLEM-TAILORED GA

Overview of the Program Generator. An overall scheme of the proposed approach is depicted in Figure 6. The program generator parses text files with CDL descriptions as input and produces GA implementations as output. The generated algorithm, on the other hand, uses only a database for input and output. As was described in the preceding section, the CDL language can specify the tables and fields needed for the input and output of the generated GA. However, it cannot specify other tables and fields in the database nor the information needed to build the database. Therefore, the database has to already exist when the GA is generated and to contain the specified tables and fields, though they may be empty.

For the approach, it is immaterial what the program generator 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 of obtaining those values from some simulation program instead. Then the GA implementation alternates with that program for as many generations as desired.

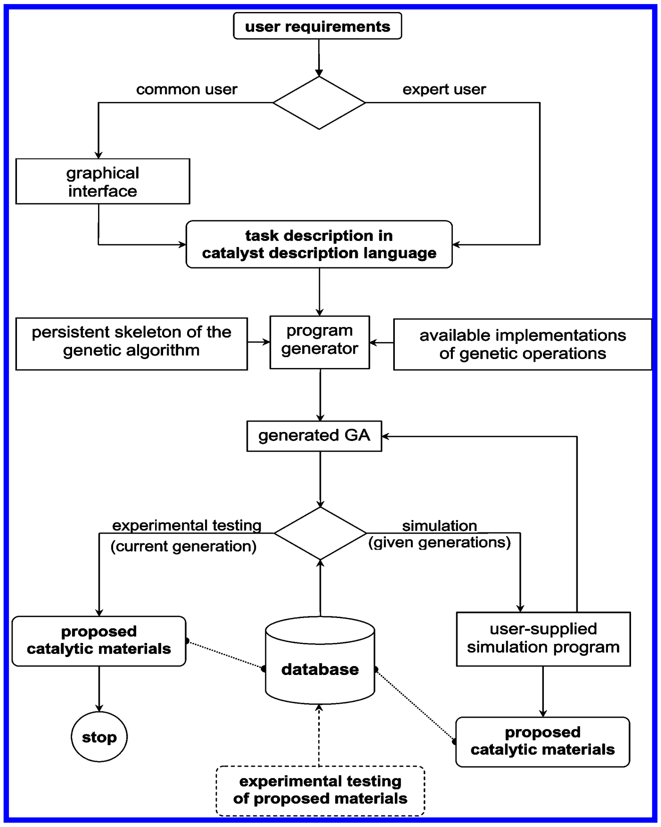

user requirements
common user
expert user
graphical interface
task description in catalyst description language
persistent skeleton of the genetic algorithm
program generator
available implementations of genetic operations
generated GA
experimental testing (current generation)
simulation (given generations)
proposed catalytic materials
database
user-supplied simulation program
stop
proposed catalytic materials
experimental testing of proposed materials

Figure 6. Schema of the functionality of a program generator generating problem-tailored GA according to CDL descriptions.

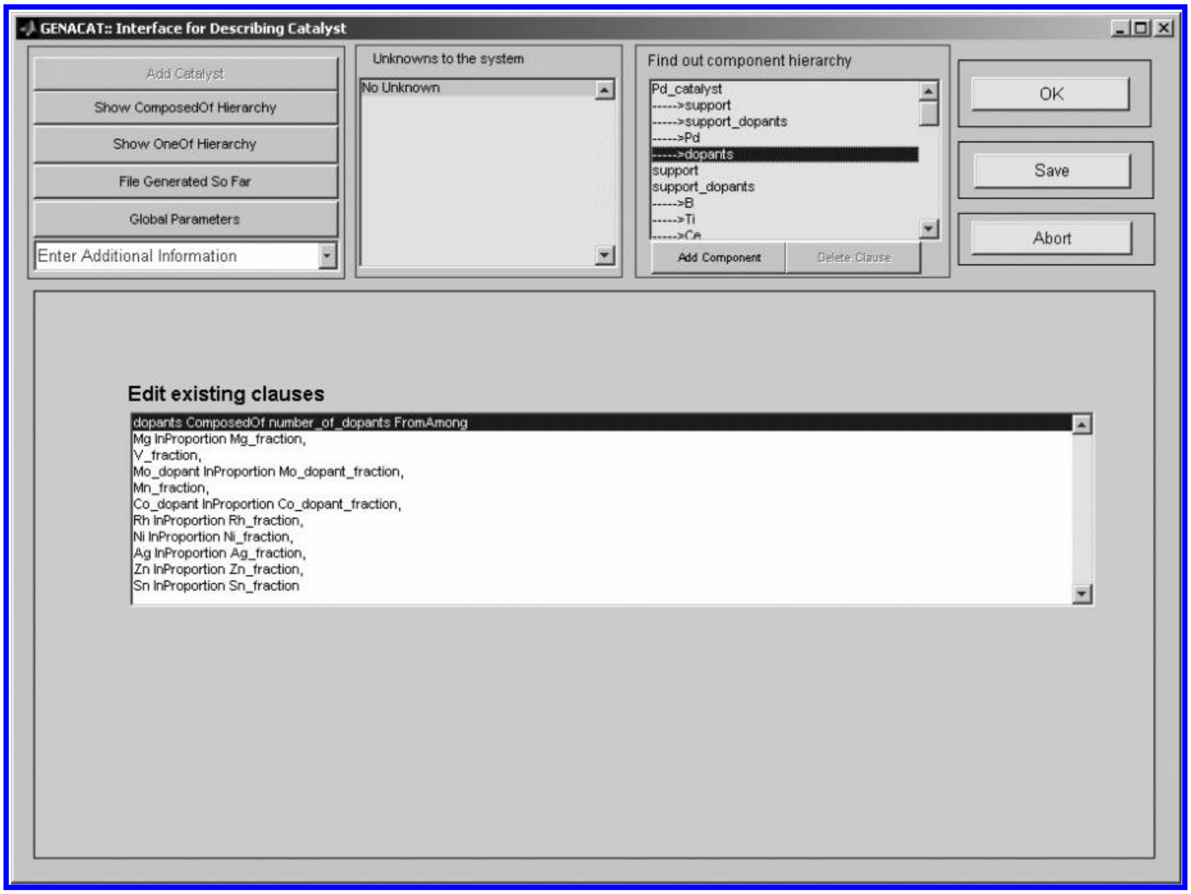
Our Approach to Constrained Mixed Optimization. To address mixed constrained optimization problems, we make use of two specific features of such problems in the area of catalytic materials:

(i) It is sufficient to consider only *linear constraints*. Even if the set of feasible solutions is not constrained linearly in reality, the finite measurement precision of the involved continuous variables always allows to constrain it piecewise linearly and to indicate the relevant linear piece with an additional discrete variable. Consequently, the set of feasible values of the continuous variables that form a solution together with a particular combination of values of the discrete variables is a *polyhedron*, though each such polyhedron can be empty, and each has its specific dimension, ranging from 1 (closed interval) to the number of continuous variables.

(ii) If a solution polyhedron is described with an inequality

$$P = \{x : Ax \leq b\}$$

then its feasibility, i.e., the property $P \neq \emptyset$, is invariant with respect to any permutation of columns of A as well as with respect to any permutation of rows of (A b). Moreover, since (ii.a) identity is also a permutation, (ii.b) each permutation has a unique inverse permutation, and (ii.c) the composition of permutations is again a permutation, the relation $\approx$ between solution polyhedra, defined for P and $P' = \{x : A'x \leq b'\}$ by means of $P \approx P'$iff (A' b') can be obtained from (A b) through some permutation of columns of A, followed by some permutation of rows of the result and of b, is an *equivalence* on the set of solution polyhedra. Consequently, it partitions that set into disjoint *equivalence classes*, the number of which can be much lower than the number of polyhedra. In the optimization tasks faced during the testing of the approach, the number of solution polyhedra ranges between thousands and hundreds of thousands, but the number of their equivalence classes ranges only between several and several dozens. As an example, the simple catalyst optimization task described in Figure 4 leads to 6494 solution polyhedra, which form 9 equivalence classes (cf. the generated code for this example, see the Supporting Information). Whereas a separate check of the nonemptiness of each polyhedron could prohibitively increase the computing time of the generated GA, forming the equivalence classes is fast, and then only one representative from each class needs to be checked.



No Unknown

```
Pd_catalyst
----->support
----->support_dopants
----->Pd
----->dopants
support
support_dopants
----->B
----->Ti
----->Ce
```

```
dopants ComposedOf number_of_dopants FromAmong
Mg InProportion Mg_fraction,
V_fraction,
Mo_dopant InProportion Mo_dopant_fraction,
Mn_fraction,
Co_dopant InProportion Co_dopant_fraction,
Rh InProportion Rh_fraction,
Ni InProportion Ni_fraction,
Ag InProportion Ag_fraction,
Zn InProportion Zn_fraction,
Sn InProportion Sn_fraction
```

Figure 7. Main window of the graphical interface allowing users to enter the information needed to create a CDL description.

Algorithm for Constrained Mixed Optimization. Combining the above approach with the fact that the constraints specified in a CDL description determine which combinations of values of discrete variables to consider suggests the following algorithm for dealing with constrained mixed optimization in the generated GA:

1. A separate continuous optimization problem is formulated for each combination of values of discrete variables that can be for some combination of continuous variables feasible with respect to the specified constraints.

2. The set of all solution polyhedra corresponding to the continuous optimization problems formulated in step 1 is partitioned according to the above equivalence.

3. One polyhedron from each partition class is checked for nonemptiness, taking into account the discernibility (measurement precision) of considered variables.

4. On the set of nonempty polyhedra, discrete optimization is performed, using operations selection and mutation developed specifically to this end. In particular, selection is performed in the following way: (i) In the first generation, all polyhedra are selected according to the uniform distribution. (ii) In the second and later generations, a prescribed proportion is selected according to an indicator of their importance due to the points from the earlier generations, and the rest are selected according to the uniform distribution. (iii) As an indicator of the importance of a polyhedron due to the points from the earlier generations, the difference between the fitness of the point and the minimal fitness of points in the earlier generations is taken and summed over all points for which the discrete variables assume the combination of values corresponding to the polyhedron. Each of the polyhedra forming the population obtained in this way corresponds to a subpopulation of combinations of values of continuous variables.

5. In each of the polyhedra found through the discrete optimization, a continuous optimization is performed. The combinations of values of continuous variables found in this way, combined with the combinations of values of discrete variables corresponding to the respective polyhedra, form together the final population of solutions to the mixed optimization problem.

PROTOTYPE IMPLEMENTATION

Based on the approach presented in the previous section, a software system GENACAT has been developed at LIKAT Berlin, allowing for the generation of genetic algorithms tailored to provided descriptions in the proposed catalyst description language (Figure 7). It has been developed in Matlab, and also the genetic operations were implemented in Matlab as well as the skeleton, i.e., the persistent part of the generated GA. In particular, they make use of Matlab Genetic Algorithm and Direct Search Toolbox[18] as well as of the Multi-Parametric Toolbox from ETH Zurich.[29] On the other hand, the variable part of the GA is generated as a binary code, which is input to the GA skeleton. As an example, the generated part corresponding to the simple CDL-description from Figure 4 transformed for readibility to a text form is available as a part of the Supporting Information. In addition, the Supporting Information also contains the Matlab code of the skeleton.

To *switch between a single run of the genetic algorithm and simulation* and to set the name of the simulation program and its parameters, the GA generator provides a simple graphical inerface. In addition, that interface continuously shows the number of catalysts proposed so far and the current phase of the whole optimization process as well as other status information.

Much more complex is the graphical interface allowing users to enter the information needed to create a CDL description. Its main window is shown in Figure 7. In addition, the interface provides also the possibility to visualize the hierarchy of component types and the hierarchy of choices (Figure 5). The decision whether to use this interface or to write the CDL description manually is made in the introductory window of GENACAT. Moreover, that introductory window also allows for employing an existing description and introducing only the necessary changes in it as well as to store fragmental descriptions for later reopening and completing. These features have a twofold importance. First, much time in preparing the description can be saved if the intended experiment is similar to an earlier one. Second, the user is not bored with the necessity to deal with information that lies outside his or her area of competence and that can be provided by some other colleague. This is intended at the first place for information concerning database and simulations, which is typically provided by a database administrator or a data analyst, but it allows also, for example, the information about catalyst composition and the information about preparation methods to be provided by two independent experimenters.

CONCLUSIONS AND DISCUSSION

The paper presents a novel approach to the optimization of catalytic materials. The objective of that approach is, on the one hand, to preserve the advantage of genetic algorithms developed specifically to this end, namely a chemically meaningful and mathematically undemanding way of formulating the optimization task, and, on the other hand, to tackle the disadvantage of specific GA—the narrow spectrum of problems for which they 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 problem-tailored GA. A prototype of such a program generator has been developed at LIKAT Berlin and is now in the testing phase. Not surprisingly, its development has been much more demanding and time-consuming than the development of any particular specific GA. From a long-time point of view, however, it actually saves development efforts. In addition, it tackles a much broader variety of problems than any specific GA. Moreover, the presented approach could be principally used also for other kinds of optimization problems in chemistry. Though they would need different description languages, the functionality of the program generator would remain the same.

In this context, the possibility of using some general description language instead of specific description languages for particular kinds of problems should be discussed. Examples of well-known languages that could be feasible for this purpose are (i) unified modeling language (UML); (ii) extensible markup language (XML); and (iii) ontology languages, such as OWL.

In the area of catalysis, there has been some interest in the XML language, though in a very general context, not in connection with the specification of optimization tasks.[30,31]

Our decision, not to use a general language but to design a specific Catalyst Description Language for the optimization of catalytic materials, has the following reasons:

(i) A general language is similarly difficult to use for chemists as generic GA software. Consequently, if chemists would have to describe problems in such a language, the efforts to replace generic GA with GA generated specifically for those problems would lose their justification.

(ii) If the description of a problem is entered through a graphical interface, then it is actually immaterial to which language the interface translates that description. However, a simple description language such as CDL preserves its advantage of an easier understandability for end users if they subsequently read the description or if they subsequently perform only small changes in it, which can be more quickly and more flexibly done through direct editing of the description than through the graphical interface.

(iii) A GA generator based on a simple description language is easier to develop and to implement than a generator based on a general language.

(iv) Similarly, a simpler language allows an easier development and implementation of an interface for entering problem descriptions.

ACKNOWLEDGMENT

The research reported in this paper has been supported by the Federal Ministry for Education and Research of the Federal Republic of Germany (03X2003B) and partially supported by the Grant Agency of Czech Republic (grant no. 201/08/1744).

Supporting Information Available: Matlab code of the GA skeleton (file "gaskeleton.txt"); generated variable part of the GA corresponding to the CDL-description from Figure 4, transformed for readibility from the original binary form to a text form (file "parsedexample.txt"). This material is available free of charge via the Internet at http://pubs.acs.org.

REFERENCES AND NOTES

(1) Holena, M.; Baerns, M. Computer-Aided Strategies for Catalyst Development. To appear in *Handbook of Heterogeneous Catalysis*; Ertl, G. et al., Eds.; Wiley-WCH: Weinheim, Germany.

(2) Holzwarth, A.; Denton, P.; Zanthoff, H.; Mirodatos, C. Combinatorial approaches to heterogeneous catalysis: strategies and perspectives for academic research. *Catal. Today* **2001**, *67*, 309−318.

(3) Végvári, L.; Tompos, A.; Göbölös, S.; Margitfalvi, J. Holographic research strategy for catalyst library design. Description of a new powerful optimization method. *Catal. Today* **2003**, *81*, 517−527.

(4) Li, B.; Sun, P.; Jin, Q.; Wang, J.; Ding, D. A simulated annealing study of Si, Al distribution in the omega framework. *J. Mol. Catal. A: Chem.* **1999**, *148*, 189−195.

(5) Eftaxias, A.; Font, J.; Fortuny, A.; Giralt, J.; Fabregat, A.; Stüber, F. Kinetic modelling of catalytic wet air oxidation of phenol by simulated annealing. *Appl. Catal. B: Environ.* **2001**, *33*, 175−190.

(6) Wolf, D.; Buyevskaya, O. V.; Baerns, M. An evolutionary approach in the combinatorial selection and optimization of catalytic materials. *Appl. Catal. A: General* **2000**, *200*, 63−77.

(7) Huang, K.; Zhan, X. L.; Chen, F. Q.; Lü, D. W. Catalyst design for methane oxidative coupling by using artificial neural network and hybrid genetic algorithm. *Chem. Eng. Sci.* **2003**, *58*, 81−87.

(8) Rodemerck, U.; Baerns, M. M.; Holena, M.; Wolf, D. Application of a genetic algorithm and a neural network for the discovery and optimization of new solid catalytic materials. *Appl. Surf. Sci.* **2004**, *223*, 168−174.

(9) Watanabe, Y.; Umegaki, T.; Hashimoto, M.; Omata, K.; Yamada, M. Optimization of Cu oxide catalysts for methanol synthesis by combinatorial tools using 96 well microplates, artificial neural network and genetic algorithm. *Catal. Today* **2004**, *89*, 455−464.

(10) Pereira, R. M.; Clerc, F.; Farrusseng, D.; Waal, J. C.; Maschmeyer, T. Effect of the Genetic Algorithm Parameters on the Optimisation of Heterogeneous Catalysts. *QSAR Comb. Sci.* **2005**, *24*, 45−57.

(11) Holena, M. Present Trends in the Application of Genetic Algorithms to Heterogeneous Catalysis. In *High-Throughput Screening in Chemical Catalysis;* Hagemeyer, A., Strasser, P., Volpe, A. F., Eds.; Wiley-WCH: Weinheim, Germany, 2004; pp 153−172.

(12) Holena, M.; Rodemerck, U.; Cukic, T.; Dingerdissen, U.; Genetic Algorithms for the Optimization of Chemical Processes Based on Problem Descriptions. *WSEAS Trans. Math.* **2007**, *6* (4), 615−621.

(13) Man, K. F.; Tang, K. S.; Kwong, S. *Genetic Algorithms. Concepts and Designs;* Springer: London, Great Britain, 1999.

(14) Vose, M. D. *The Simple Genetic Algorithm. Foundations and Theory;* MIT Press: Cambridge MA, 1999.

(15) Wong, M. L.; Leung, K. S. *Data Mining Using Grammar Based Genetic Programming and Applications;* Kluwer: Dordrecht, The Netherlands, 2000.

(16) Freitas, A. A. *Data Mining and Knowledge Discovery with Evolutionary Algorithms;* Springer: Berlin, Germany, 2002.

(17) Reeves, C. R.; Rowe, J. E. *Genetic Algorithms: Principles and Perspectives*; Kluwer, Boston, MA, 2003.

(18) *Genetic Algorithm and Direct Search Toolbox*, *version 2*; The MathWorks, Inc.: Natick, MA, 2004.

(19) McLeod, A. S.; Johnston, M. E.; Gladden, L. F. Development of a Genetic Algorithm for a Molecular Scale Catalyst Design. *J. Catal.* **1997**, *167*, 279−285.

(20) McLeod, A. S.; Gladden, L. F. Heterogeneous Catalyst Design Using Stochastic Optimization Algorithms. *J. Chem. Inf. Comput. Sci.* **2000**, *40*, 981−987.

(21) Buyevskaya, O. V.; Wolf, D.; Baerns, M. Ethylene and propene by oxidative dehydrogenation of ethane and propane − Performance of rare-earth oxide-based catalysts and development of redox-type catalytic materials by combinatorial methods. *Catal. Today* **2000**, *62*, 91−99.

(22) Rodemerck, U.; Wolf, D.; Buyevskaya, O. V.; Claus, P.; Senkan, S.; Baerns, M. High-throughput synthesis and screening of catalytic materials: Case study on the search for a low-temperature catalyst for the oxidation of low-concentration propane. *Chem. Eng. J.* **2001**, *82*, 3−11.

(23) Grubert, G.; Kondratenko, E.; Kolf, S.; Baerns, M.; van Geem, P.; Parton, R. Fundamental insights into the oxidative dehydrogenation of ethane to ethylene over catalytic materials discovered by an evolutionary approach. *Catal. Today* **2003**, *81*, 337−345.

(24) Caruthers, J. M.; Lauterbach, J. A.; Thomson, K. T.; Venkatasubramanian, V.; Snively, C. M.; Bhan, A.; Katare, S.; Oskarsdottir, G. Catalyst design: knowledge extraction from high-throughput experimentation. *J. Catal.* **2003**, *216*, 98−109.

(25) Corma, A.; Serra, J. M.; Chica A. Discovery of new paraffin isomerization catalysts based on SO_4^{2-}/ZrO_2 and WO_x/ZrO_2 applying combinatorial techniques *Catal. Today* **2003**, *81*, 495−506.

(26) Serra, J. M.; Chica, A.; Corma, A. Development of a low temperature light paraffin isomerization catalysts with improved resistance to water and sulphur by combinatorial methods *Appl. Catal. A: General* **2003**, *239*, 35−42.

(27) Corma, A.; Serra, J. M. Heterogeneous combinatorial catalysis applied to oil refining, petrochemistry and fine chemistry. *Catal. Today* **2005**, *107−108*, 3−11.

(28) Corma, A.; Serra, J. M.; Serna, P.; Valero, S.; Argente, E.; Botti, V. Optimisation of olefin epoxidation catalysts with the application of high-throughput and genetic algorithms assisted by artificial neural networks (softcomputing techniques). *J. Catal.* **2005**, *229*, 513−524.

(29) Kvasnica, M.; Grieder, P.; Baotic, M.; Christophersen, F. J. *Multi-Parametric Toolbox (MPT)*; ETH: Zurich, Switzerland, 2005.

(30) Gilardoni, F.; Yakovlev, A. CatML: A catalyst markup language. Presented at the 226th ACS National Meeting [Online], New York, September 7−11, 2003; J. K. Javits Convention Center. http://acscinf.org/docs/meetings/226nm/226cinfabstracts.htm (accessed September 25, 2007)

(31) Warr, W. *IUPAC project meetings: Extensible markup language (XML) data dicitionaries and chemical identifier*; Wendy Warr & Associates: Holmes Chapel, Great Britain, 2004.

CI700218P