Genetic Algorithms for the Optimization of Catalysts in Chemical Engineering

Martin Holena*

Abstract **The paper addresses key problems pertaining to the commonly used evolutionary approach to the search for optimal catalysts in chemical engineering. These are on the one hand the insufficient dealing in existing implementations of genetic algorithms with mixed optimization, which plays a crucial role in catalysis, on the other hand the narrow scope of genetic algorithms developed specifically for searching optimal catalyst. The paper proposes an approach to constrained mixed optimization based on formulating a separate linearly-constrained continuous optimization task for each combination of values of the discrete variables. Then, discrete optimization on the set of nonempty polyhedra describing the feasible solutions of those tasks is performed, followed by solving those tasks for each individual of the resulting population of polyhedra. To avoid computationally expensive checking of the non-emptiness of individual polyhedra, the set of polyhedra is first partitioned into equivalence classes such that only one representative from each class needs to be checked. Finally, the paper outlines a program generator automatically generating problem-tailored genetic algorithms from descriptions of optimization tasks in a specific description language, which employs the proposed approach to constrained mixed optimization.**

Keywords: genetic algorithms, chemical engineering, constrained optimization, mixed optimization, program generator

1 Introduction

In chemical engineering, 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 up the reaction or to improve its selectivity to the desired products. Catalysts are materials that decrease the energy needed to activate a chemical reaction without being themselves consumed in it. Catalytic materials typically consist of several components with different purpose 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 chemical process. Consequently, the search for catalysts leading to optimal 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.*

Commonly used optimization methods, such as the 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 measurement error. 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, and stochastic ones, such as simulated annealing, or genetic and other evolutionary algorithms. Especially *genetic algorithms (GA)* have become quite popular as to the search for optimal catalysts in chemical engineering, mainly due to the possibility to establish a straightforward correspondence between multiple optimization paths followed by the algorithm and the channels of a high-throughput reactor in which the materials proposed by the algorithm are subsequently tested.

Nevertheless, a lack of appropriate implementations still hinders genetic algorithms to be routinely used to this end. Generic GA implementations, such as the Genetic Algorithms and Direct Search Toolbox of Matlab [11], do not sufficiently address all the above features (i.)-(iv.), in particular mixed optimization is always addressed only

*Institute of Computer Science, Academy of Sciences of the Czech Republic, Pod vodárenskou věží 2, 18207 Praha 8, Czech Republic, E-mail: martin@cs.cas.cz, and Leibniz Institute for Catalysis, Branch Berlin, Richard-Willstätter-Straße 12, 12489 Berlin, Germany, E-mail: martin.holena@catalysis.de. The research reported in this paper has been supported by the grants Nos. 201/08/0802 and ICC/08/E018 of the Grant Agency of the Czech Republic

quite superficially. In addition, they use a low-level coding of input variables by means of bit strings and real numbers, which is tedious, error prone, and difficult to understand for chemical engineers. This disadvantage can be avoided with GA developed specifically for searching optimal catalysts, and several of them have really appeared in recent years [14, 18, 20, 22]. However, none of those specific algorithms attempts to seriously tackle constrained mixed optimization, and the experience gathered with them so far shows that they bring a difficulty of another kind: they are usable only for a narrow spectrum of particular optimization tasks and have to be reimplemented each time when different tasks emerge.

This paper proposes a quite general approach to employing GA for the constrained mixed optimization tasks entailed by the search for catalysts. However, to avoid the disadvantages of generic GA implementations, and at the same time not to suffer from the narrow scope of specific GA, this approach has not been implemented in any particular algorithm. Instead, it has been employed in a program generator that generates problem-tailored GA from descriptions of optimization tasks in a specific description language developed to this end [6].

In the next section, theoretical principles of GA, and main approaches to using them for constrained optimization are recalled. The approach proposed for the constrained mixed optimization tasks entailed by the search for catalysts is explained in Section 3. Finally, Section 4 sketches a prototype program generator that generates problem-tailored GA based on this approach.

2 Genetic algorithms and their modifications for constraints

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* [10, 16, 19]. Basically, that way consists in:

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

In the context of the search for optimal catalysts in chemical engineering, it is useful to differentiate between quantitative mutation, which modifies merely the proportions of substances already present in the catalytic material, and qualitative mutation, which enters new substances or removes present ones (Figure 1).

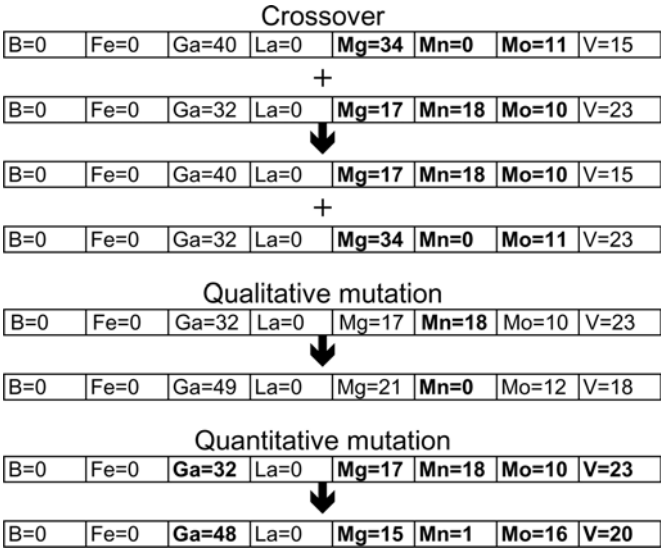

Figure 1: Illustration of genetic operations used in the search for catalysts, the values in the examples are molar proportions (in %) of oxides of the indicated elements in the catalytic material

Genetic algorithms have been originally introduced for unconstrained optimization. However, there have been repeated attempts to modify them for *constrains*. Basically, those attempts belong to one of (or combine several of) the following approaches [1, 4, 5, 9, 12, 17]:

(i) To *ignore* offsprings infeasible with respect to constraints and not include them into the new population. Because in constrained optimization, the global optimum frequently lies on a boundary determined by the constraints, ignoring infeasible offsprings may lead to discharging information on offsprings very close to that optimum. Moreover, for some genetic algorithms this approach can lead to the deadlock of not being able to find a whole population of feasible offsprings.

(ii) To modify the objective function by a superposition of a *penalty for infeasibility*. This works well if theoretical considerations allow an appropriate choice of the penalty function. If on the other hand some heuristic penalty function is used, then its values typically turn out to be either too small, which can allow the optimization paths to stay forever outside the feasibility area, or too large, thus suppressing the information on the value of the objective function for infeasible offsprings.

(iii) To *repair the infeasible offspring* through modifying it so that all constraints get fulfilled. This again can discard information on some offsprings close to the

global optimum. Therefore, various modifications of the repair approach have been proposed that give up full feasibility, but preserve some information on the original offsprings, e.g., repairing only randomly selected offsprings (with a prescribed probability), or using the original offspring together with the value of the objective function of its repaired version.

(iv) To add the feasibility/infeasibility as *another objective function*, thus transforming the constrained optimization task into a task of *multiobjective optimization*. Unfortunately, this new task is similarly far from the original purpose of genetic algorithms and similarly difficult for solving with them as the original one.

(v) To *modify the recombination and/or mutation operator* in such a way that it gets closed with respect to the set of feasible solutions. Hence, also a mutation of a point fulfilling all the constraints or a recombination of two such points has to fulfil them. An example of such an operator is the edge-recombination operator used for the traveling salesman problem [21]. Unfortunately, such a modification always requires enough knowledge about the particular constraints, therefore this approach can not be elaborated in a general setting.

This brief survey indicates that each approach has its specific difficulties, and none of them is an ultimate way of dealing with constraints, generally better than the others.

3 Proposed approach to constrained mixed optimization

The proposed way of solving constrained mixed optimization tasks follows the above recalled approach (v) and is based on two specific features of such tasks pertaining to the search for optimal catalysts:

(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

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

then its feasibility (i.e., $P \neq \emptyset$) is invariant with respect to any permutation of columns of A, as well as respect to any permutation of rows of (Ab). Moreover, since:

- identity is also a permutation,
- each permutation has a unique inverse permutation,
- 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

$$\begin{aligned} P \approx P' \text{ iff } & (A'b') \text{ can be obtained from} \\ & (Ab) \text{ through some permutation} \\ & \text{of columns of } A, \text{ followed by} \\ & \text{some permutation of rows} \\ & \text{of the result and of } b \end{aligned} \tag{2}$$

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 real-world tasks on which the approach has been tested so far, the number of solution polyhedra ranges from hundreds of thousands to hundreds of millions, but the number of their equivalence classes ranges only from several dozens to several hundreds. An example is documented in Figure 2, using the graphical interface with status information of the generated algorithm (cf. Section 4). In that example, there are 874848240 solution polyhedra, which form 480 equivalence classes. Whereas a separate check of the nonemptiness of each polyhedron could prohibitively increase the computing time of the GA, forming the equivalence classes is fast, and then only one representative from each class needs to be checked.

Those features together with the fact that the constraints determine which combinations of values of discrete variables to consider, suggest the following procedure for dealing with constrained mixed optimization:

1. A separate continuous optimization task 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 tasks formulated in step 1, is partitioned according to the equivalence (2).

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

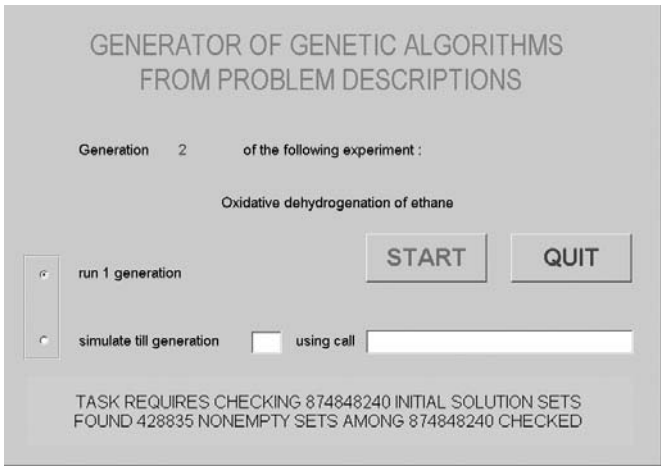

Figure 2: Simple graphical interface with status information of the GA generated according to Section 4, after all the polyhedral solutions sets for individual continuous optimization tasks have been checked for non-emptiness

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:

   - In the first generation, all polyhedra are selected according to the uniform distribution.
   - 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 is selected according to the uniform distribution.
   - 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, 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 of the mixed optimization task.

```
⋮
FeedbackTable GlobalParameter vam
FeedbackField GlobalParameter fitness
vam_catalyst ComposedOf support InProportion support_fraction,
& support_dopants InProportion support_dopants_fraction, Pd InProportion
& Pd_fraction, Au InProportion Au_fraction, dopants InProportion
& dopants_fraction PreparedUsing dopants_preparation
support_fraction FromInterval 87,100 WithPrecision 1
support_dopants_fraction FromInterval 0,10 WithPrecision 0.1
Pd_fraction OneOf 0.7,1
Au_fraction OneOf 0,0.2,0.4,0.6,0.8
dopants_fraction FromInterval 0,1.2 WithPrecision 0.01
dopants_preparation OneOf sequential, intermediate_drying
support_fraction + support_dopants_fraction + Pd_fraction + Au_fraction
& + dopants_fraction = 1
support OneOf siliperl, aerosil200, aerosil300, basisP25, zirkonix, TiO2BET
support_dopants ComposedOf number_of_support_dopants FromAmong
& B InProportion B_fraction, Ti InProportion Ti_fraction, Ce InProportion
& Ce_fraction, Fe InProportion Fe_fraction, Y InProportion Y_fraction,
& La InProportion La_fraction, Mo InProportion Mo_fraction, Cr InProportion
& Cr_fraction, Nb InProportion Nb_fraction
number_of_support_dopants OneOf 0,1,2
B_fraction FromInterval 1,10 WithPrecision 0.1
Ti_fraction FromInterval 1,10 WithPrecision 0.1
Ce_fraction FromInterval 1,10 WithPrecision 0.1
Fe_fraction FromInterval 1,10 WithPrecision 0.1
Y_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
Nb_fraction FromInterval 1,10 WithPrecision 0.1
B_fraction + Ti_fraction + Ce_fraction + Fe_fraction + Y_fraction +
& La_fraction + Mo_fraction + Cr_fraction + Nb_fraction =
& support_dopants_fraction
siliperl IsPrimitive
aerosil200 IsPrimitive
aerosil300 IsPrimitive
basisP25 IsPrimitive
zirkonix IsPrimitive
TiO2BET IsPrimitive
⋮
```

Figure 3: Example fragment of a task description in the catalyst description language proposed in [6]

4 Implementation by means of a program generator

To be available for a possibly broad spectrum of optimization tasks entailed by the search of optimal catalysts in chemical engineering, the proposed approach has not been incorporated into a particular GA implementation, but has been combined with a *program generator* that transforms a description of the optimization task to an executable program. A first prototype of such a generator has been developed at the Leibniz Institute for Catalysis (LIKat) in Berlin and is currently in the testing phase. Differently to a human programmer, a program generator needs the description to be expressed in a rigorously formal way, i.e., with some kind of a task description language. For catalysis, a formal catalyst description language has been proposed in [6]. It allows expressing a broad variety of user requirements on the catalytic materials to be sought by the genetic algorithm, as well as on the algorithm itself (Figure 3).

An overall scheme of this approach is depicted in Figure 4. The program generator accepts text files with task

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. In the prototype implementation of the program generator at LIKat Berlin, the generated GA consists of a persistent part, called *skeleton*, and of a variable part, generated as a binary code, which is input to the GA skeleton. The skeleton has been implemented in Matlab, and for the implementations of individual genetic operations makes calls to its Genetic Algorithm and Direct Search Toolbox [11], as well as to the Multi-Parametric Toolbox from ETH Zurich [8]

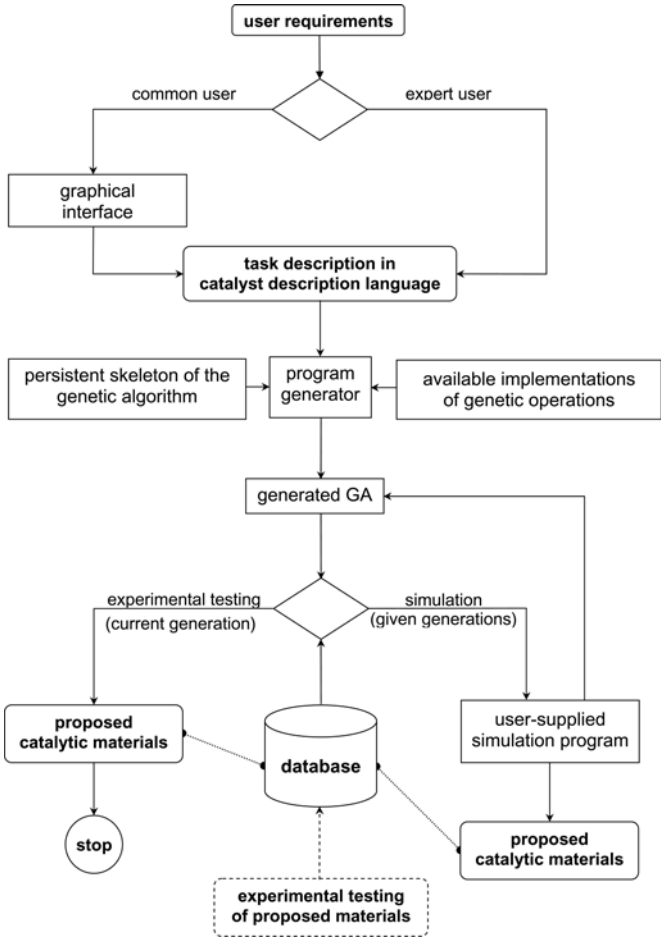

Figure 4: Scheme of the proposed approach to generating problem-tailored genetic algorithms according to descriptions of catalytic optimization tasks

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. This possibility is intended for the case of *surrogate modeling* (also known as *metamodeling*) [3, 13, 15] – i.e., for the case that optimization is combined with some regression model of the relationship between the objective function (in particular, fitness) and its inputs. In the context of the GA-based search for optimal catalysts, the fitness function is some measure of catalyst performance, such as yield and conversion. Applications of surrogate modeling have been reported several times in this context, always combining a specifically developed genetic algorithms with regression by means of artificial neural networks [2, 7, 18]. In the case of surrogate modeling, the generated GA implementation alternates with the employed simulation program for as many generations as desired.

Figure 5: Main window of the graphical interface allowing users to enter the information needed to create a task description

Finally, the implementation places a graphical interface between the user and the program generator, the purpose of which is to remove from the users the necessity to write files with task descriptions manually, and the necessity to understand the description 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 task description (Figure 5). In addition, also the generated GA implementations contain a simple graphical interface, which shows the progress of the performed optimization, and allows deciding whether to run the implementation only once or which external program to use for simulation (Figure 2).

5 Conclusion

The paper has presented solutions to two key problems encountered when genetic algorithms are used for searching optimal catalytic materials in chemical engineering. First, it has proposed an approach to constrained mixed optimization tasks based on specific properties of the search for optimal catalysts. Second, it has shown that it is possible to avoid repeated reimplementations of genetic

algorithms developed specifically for searching optimal catalysts without having to resort to generic software. To this end, a program generator has been used that generates problem-tailored genetic algorithms, based on the proposed approach to constrained mixed optimization, from descriptions of optimization tasks. A first prototype of such a generator has been developed at LIKat Berlin and is currently in the testing phase.

References

[1] P.C. Chu and J.E. Beasley. Constraint handling in genetic algorithms: The set partitioning problem. *Journal of Heuristic*, 4:323–358, 1998.

[2] F. Clerc, M. Lengliz, D. Farrusseng, C. Mirodatos, S.R.M. Pereira, and R. Rakotomata. Library design using genetic algorithms for catalysts discovery and optimization. *Review of Scientific Instruments*, 76:062208, 2005.

[3] M.A. El-Beltagy, P.B. Nair, and A.J. Keane. Metamodeling techniques for evolutionary optimization of computaitonally expensive problems: Promises and limitations. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 196–203. Morgan Kaufmann Publishers, San Francisco, 1999.

[4] C.M. Fonseca and P.J. Fleming. An overview of evolutionary algorithms in multiobjective optimization. *Evolutionary Computation*, 3:1–16, 1995.

[5] R. Fung, J. Tang, and D. Wang. Extension of a hybrid genetic algorithm for nonlinear programming problems with equality and inequality constraints. *Computers and Operations Research*, 29:261–274, 2002.

[6] M. Holena. Present trends in the application of genetic algorithms to heterogeneous catalysis. In A. Hagemeyer, P. Strasser, and A.F. Volpe, editors, *High-Throughput Screening in Chemical Catalysis*, pages 153–172. Wiley-VCH, Weinheim, 2004.

[7] M. Holena. A new approach to tuning heuristic parameters of genetic algorithms. *WSEAS Transactions on Information Science and Applications*, 3:562–569, 2006.

[8] M. Kvasnica, P. Grieder, M. Baotić, and M. Morari. Multi-parametric toolbox (MPT). In *Hybrid Systems: Computation and Control*, pages 449–462. Springer Verlag, Berlin, 2004.

[9] H. Li, Y.-C. Jiao, and Y. Wang. Integrating the simplified interpolation into the genetic algorithm for constrained optimization problems. In *Computational Intelligence and Security*, pages 247–254. Springer Verlag, Berlin, 2005.

[10] K.F. Man, K.S. Tang, and S. Kwong. *Genetic Algorithms. Concepts and Design*. Springer Verlag, London, 1999.

[11] The Mathworks, Inc. *Genetic Algorithm and Direct Search Toolbox*, 2001.

[12] Z. Michalewicz and M. Schonauer. Evolutionary algorithms for constrained parameter optimization problems. *Evolutionary Computation*, 4:1–32, 1996.

[13] Y.S. Ong, P.B. Nair, A.J. Keane, and K.W. Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In Y. Jin, editor, *Knowledge Incorporation in Evolutionary Computation*, pages 307–331. Springer Verlag, Berlin, 2005.

[14] R.M. Pereira, F. Clerc, D. Farusseng, J.C. Waal, and T. Maschmeyer. Effect of genetic algorithm parameters on the optimization of heterogeneous catalysts. *(QSAR) and Combinatorial Science*, 24:45–57, 2005.

[15] A. Ratle. Kriging as a surrogate fitness landscape in evolutionary optimization. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 15:37–49, 2001.

[16] C.R. Reeves and J.E. Rowe. *Genetic Algorithms: Principles and Perspectives*. Kluwer Academic Publishers, Boston, 2003.

[17] D.J. Reid. Genetic algorithms in constrained optimization. *Mathematical and Computer Modelling*, 23:87–111, 1996.

[18] U. Rodemerck, M. Baerns, and M. Holena. Application of a genetic algorithm and a neural network for the discovery and optimization of new solid catalytic materials. *Applied Surface Science*, 223:168–174, 2004.

[19] M.D. Vose. *The Simple Genetic Algorithm. Foundations and Theory*. MIT Press, Cambridge, 1999.

[20] Y. Watanabe, Umegaki, T., Hashimoto, M., Omata, K., and M. Yamada. Optimization of Cu oxide catalysts for methanol synthesis by combinatorial tools using 96 well microplates, artificial neural network and genetic algorithm. *Catalysis Today*, 89:455–464, 2004.

[21] D. Whitley, T. Starkweather, and D. Shanner. The traveling salesman and sequence scheduling: Quality solutions using genetic edge recombination. In L. Davis, editor, *Handbook of Genetic Algorithms*, pages 350–372. Van Nostrand Reinold, New York, 1991.

[22] D. Wolf, O.V. Buyevskaya, and M. Baerns. An evolutionary approach in the combinatorial selection and optimization of catalytic materials. *Applied Catalyst A: General*, 200:63–77, 2000.