Using Evolutionary Methods with a Heterogeneous Genotype Representation for Design Optimization of a Tubular Steel Trellis Motorbike-Frame

U. M. Fasel, O. König *, M. Wintermantel and P. Ermanni

Centre of Structure Technologies, Institute of Mechanical Systems, Swiss Federal Institute of Technology, Zurich, Switzerland

Abstract

We present an optimization approach for a tubular steel trellis motorbike-frame based on evolutionary methods. The optimization objective is to minimize the mass of the frame subject to a given torsional stiffness and a strength constraint occurring during braking of the motorbike. Further we take into account different constraints on available tube sizes, what leads to continuous, discrete or even mixed optimization variables. For stiffness and strength calculations we use our own Finite Element code FELyX (Finite Element Library experiment). As optimization engine we use Evolutionary Algorithms within the framework of the Evolving Objects (EO) library. There we implemented a new heterogeneous genotype that allows to combine different parameter types, represented by according genes into a single genotype. Further we discuss the impact of smart parameterization schemes onto the convergence of such methods. Finally we show how they can be linked naturally and efficiently to modern Evolutionary Methods that distinguish between phenotype and genotype representation.

Key words: Evolutionary Algorithms, heterogeneous genotype, mixed parameter optimization, discrete parameter optimization, motorcycle main frame

* Corresponding author. Address: Leonhardstr. 27, LEO C4, CH-8092 Zurich
Email addresses: okoenig@imes.mavt.ethz.ch (O. König),
wintermantel@imes.mavt.ethz.ch (M. Wintermantel),
ermanni@imes.mavt.ethz.ch (P. Ermanni).

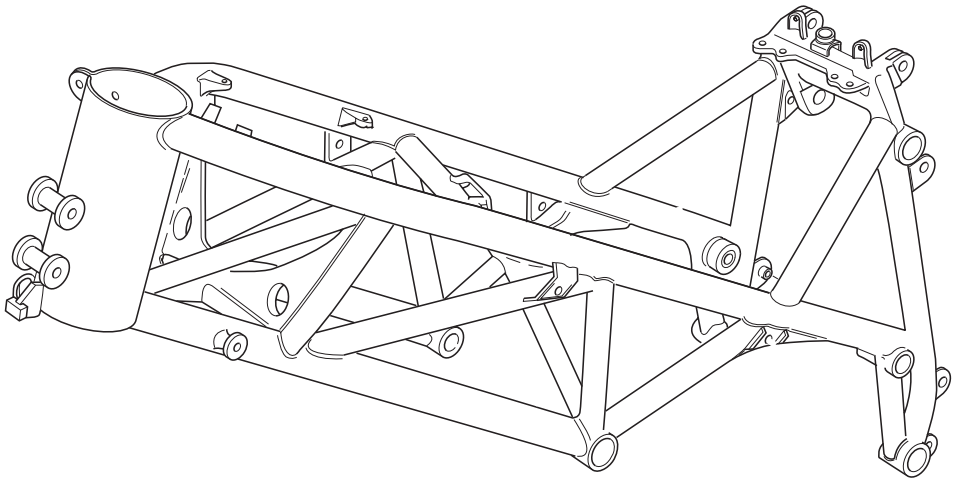

Fig. 1. Tubular steel-trellis frame of Ducati 998S.

1 Introduction

For the race-track performance of a motorcycle the main frame, connecting the front wheel via front fork and steering shaft with the swing arm, is one of the most crucial parts. Today as in the last decade, there were two successfully applied concepts for such frames. One is an aluminum-box frame, typically applied in Japanese motorcycles. The other one, mainly used by the Italian builder Ducati, is a tubular steel trellis frame. Both concepts showed to perform at an equal level during the last decade of motorbike racing. In our work we will focus on the optimization of a tubular steel trellis frame. Therefore, as a starting point, we use a frame of the Ducati 916/996/998 (see Fig. 1), that has been continously improved and already won several world championships during the last decade.

To optimize such a frame, one has first to be aware of what the design objectives are. Considering only race application, as we do within this work, weight is always a central issue. Further the stiffness of the frame between the steering shaft and the swing arm is a crucial parameter. But it is not maximum stiffness that designers want to achieve; in fact they want to have a certain compliance. This compliance depends on various motorcycle parameters such as the geometry of the bike or the power deconvolution of the engine. For our work, we will not discuss this parameter further, we just assume that a certain stiffness/flexibility has to be achieved. Experience shows that this stiffness values are so high, that maximum stresses induced by a torsional load case are far below critical values.

A second relevant load case is braking. Considering sport or race bikes, principally the front brake produces the whole deceleration. Therefore via front fork and steering shaft a momentum and a compressive force are introduced into the chassis. For this load case not stiffness but strength is crucial.

Table 1
Four important forms of Evolutionary Algorithms

Algorithm	Abbreviation	References
Genetic Algorithms	**GA**	[1–3]
Evolutionary Programming	**EP**	[4]
Evolution Strategies	**ES**	[5–8]
Genetic Programming	**GP**	[9]

Looking at the Ducati tubular-steel-trellis frame the most straight-forward approach for optimization is to adjust the dimensions of the tubes in an optimal way. But because of manufacturing costs, one has to be aware that only a limited number of different tube dimensions or even standard dimensions should be used.

Thus the global aim is to achieve minimum weight of the frame under the equality constraint of preserving the specified structural torsional compliance and the upper limit constraint of providing sufficient strength for the critical braking load case by adjusting and combining a limited number of available tubes. This obviously is a highly heterogenous optimization problem. That is why we consider Evolutionary Algorithms as especially suited in this context. They can deal in a quite natural way with all kinds of constrains as well as with discrete variables.

2 Basic ideas of Evolutionary Algorithms in Structural Optimization

Evolutionary Algorithms use an analogy with natural evolution to perform search by evolving solutions to problems, usually working with large collection of solutions at a time. The common underlying idea behind the techniques is to have a given population of individuals (solutions); the environmental pressure causes natural selection and hereby the fitness of the population is growing. From the point of view of classical optimization, EAs represent a powerful stochastic zeroth order method and can find the global optimum of very rough functions.

In Structural Optimization, EAs are used in many different forms. Commonly they are divided into four categories (see Table 1). However, they are all based on similar evolutionary principles. Therefore we will use a more modern terminology also used by [10] and [11]: we generally speak just of Evolutionary Algorithms. All the above listed strategies can be seen as specializations of

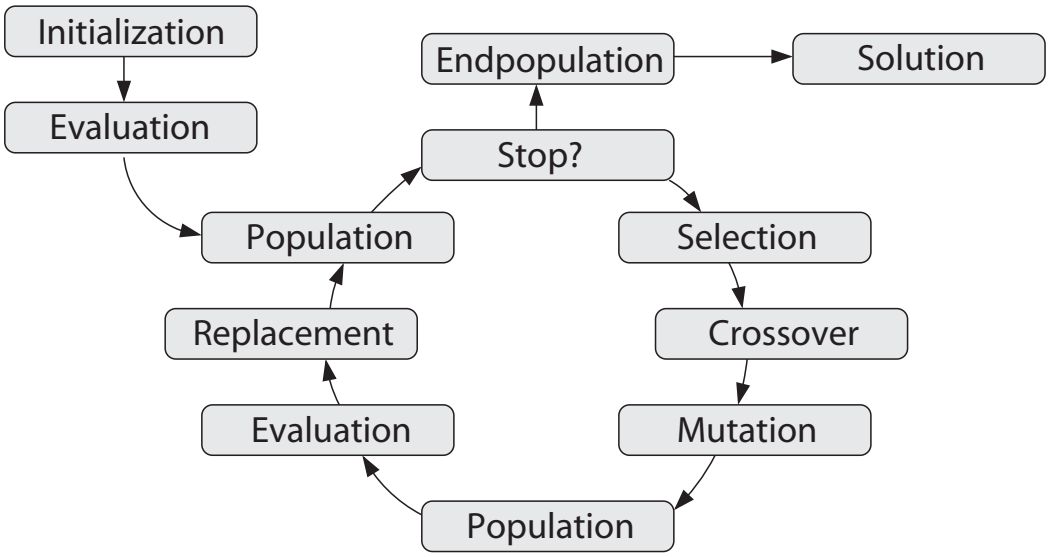

Fig. 2. Basic scheme of an Evolutionary Algorithm

general EAs which we will describe below.

EAs use two separate spaces: the search space and the solution space. The search space is a space of coded solutions (*genotype*) to the problem. The solution space is the space of actual solutions (*phenotype*). EAs maintain a population of $P \in \mathbb{N}$ individuals. Each individual consists of a genotype and a corresponding phenotype.

A simple EA works as outlined in Figure 2. A population of a given number of individuals is initialized randomly. The fitness of the whole population is evaluated *(Evaluation)*. A fitness value can be assigned to every individual by applying the objective function as an abstract fitness measure. The fitness from the evaluation is then used to determine how many copies of each individual are placed into a temporary area often termed the mating pool. For reproduction parents are picked from the mating pool by giving some preference to individuals with better fitness values *(Selection)*. Offspring are generated by the use of the crossover operator, which randomly allocates genes from each parent's genotype to each offspring's genotype. Mutation is then occasionally applied to the offspring. A new population is built by deterministic or stochastic choice of parents and offspring (*Replacement*). Then the new individuals of the population are evaluated. The entire process of evaluation, reproduction and replacement continues until a given *termination criterion* is reached. This can for example be triggered by with the values of a time or generation counter or the best fitness of the population.

3 Programming Tools

Within our work, we use different C++ software libraries freely available under a public license as open source. The evolutionary strategies in this work are implemented using the EO (Evolving Objects) library [1]. Fitness evaluations are accomplished using the FELyX (Finite Element Library eXperiment) library [2] developed at our chair. Everything was done under Linux/Unix operating systems and compiled with GNU gcc compilers [3]. As pre- and postprocessor for the Finite Element model of the frame, we used the commercial Finite Element software ANSYS [4].

3.1 EO - The Evolvable Objects Library

EO - the Evolvable Objects library allows to evolve any data structure (object) that provides the necessary operators for the according evolutionary programming paradigm. Basically EO just provides the building blocks to perform any kind of evolutionary strategy within the field of evolutionary computing. Nevertheless common paradigms like Genetic Algorithms, Evolutionary Strategies, Evolutionary Programming, or Genetic Programming are already implemented in the actual release of EO. The adaption of EO for the actual work is discussed in Sec. 4.

3.2 FELyX - The Finite Element Library eXperiment

FELyX is an object-oriented Finite Element library. It is especially designed for the needs in the field of structural optimization, independent of what kind of numerical optimizers are used. Therefore it primarily has to be fast, but it should also allow to access/modify easily all the data of a Finite Element calculation. Considering speed it compares favorably with commercial Finite Element programs like ANSYS without relying on a lot of machine- and compiler-specific code optimization [13]. The second point is addressed through the object-oriented programming approach resulting in one single Finite Element object that allows to manage all of the data of a Finite Element calculation. Further on, FELyX provides an interface to ANSYS allowing to use the graphical pre/post processor of this commercial program. When using Evolutionary Algorithms, the Finite Element object can directly be accessed

[1] http://eodev.sourceforge.net [12]
[2] http://felyx.sourceforge.net [13]
[3] http://gcc.gnu.org
[4] http://www.ansys.com

to simulate the needed mechanical properties of a structure for fitness evaluation.

4 eoUniGene - A Universal Genotype for Heterogeneous Parameter Lists

In the following we present the concept and implementation of a new genotype able to handle heterogeneous lists of parameters in a very natural way. First some general ideas of genotype design are discussed in Section 4.1, then the implementation of the eoUniGene genotype is detailed in Section 4.2, and appropriate operators are presented in Section 4.3.

4.1 Genotype Design

A well-designed genotype is able to code any feasible and only feasible solutions for a specific problem. Ideally this projection from the genotype space to the phenotype space of feasible solutions $f : G \rightarrow P$ is bijective. Therefore f has to be surjective

$$f(G) = P \qquad \text{or} \qquad \forall x' \in P \; \exists \; x \in G : f(x) = x' \tag{1}$$

as well as injective

$$\forall x, y \in G : x \neq y \Rightarrow f(x) \neq f(y). \tag{2}$$

But when looking at real-world problems like our steel-trellis frame already condition (1) can not be achieved in any direct way. To do so it would be necessary to have a genotype that implicitly only codes solutions that fulfill e.g. the stress constraints. Obviously this is not easily achieved. Actually in most cases it only holds

$$f^{-1}(P) \subset G. \tag{3}$$

Also condition (2) is far from being always fulfilled. Often there are may individuals in the genotype space that are rated with an equal fitness and therefore identical in the phenotype space. The more this occurs the more it can harm the convergence behavior of the EA.

Even though it is not possible to achieve conditions (1) and (2) in a strict mathematical sense, it must be the goal of genotype design to approximate

them as close as possible. For condition (1) this means the space of infeasible designs

$$D_{infeas} = f(G) - P \tag{4}$$

should be small, since in the evolutionary optimization process solutions within D_{infeas} are evaluated as well and require additional computational resources. But they have little to no further impact on the evolutionary optimization process since they are excluded by either direct elimination or by the application of some sort of penalty mechanisms. For condition (2)

$$D_{sur} = \{x, y \in G : f(x) = f(y) \wedge x \neq y\} \tag{5}$$

should be kept as small as possible.

4.2 Implementation of a Universal Genotype

For the motorbike frame, we want to investigate optimal solutions for different constraints on available tube dimensions, as introduced in Section 1. In terms of genotype design this would mean to either implement dedicated genotypes and according operators for each case or to treat all constraints on available tube dimensions with penalty mechanisms. While the first approach can not be considered very efficient, the second proposal is undesirable because of the ideas presented in the previous Section.

Therefore we introduce an universal genotype for parameter optimization. This genotype is based upon generic programming paradigms as they are widely used within modern object-oriented programming languages. It consists of a collection of different gene types, where each type represents a common parameter type such as *float* or *int*. Therefore any genotype that is representable by a collection of the available gene types can be realized by just composing a heterogenous/polymorphic list of the appropriate genes. The genes implemented at the moment are:

- **Float-gene** represents an arbitrary floating point parameter. Upper and/or lower limits can be provided.
- **Integer-gene** represents an integer parameter. Again upper and/or lower limits can be provided.
- **Bool-gene** represents a binary parameter that can be true or false.
- **Float-list-gene** is a list of arbitrary floating point values. The parameter always has to represent one of these values.
- **Const-float-list-gene** are equally distributed floating point values i.e. they have a constant distance between two neighboring values. Therefore this

gene is quite similar to the integer gene.
- **String-list-gene** is a list of arbitrary discrete values upon which no norm or ordering can be applied. This is in contrast to the integer-gene and the float-list-gene and the const-float-list-gene.

Further float-gene, integer-gene, float-list-gene, and const-float-list-gene can be provided with so called cyclic properties. This is suitable when between two possible gene values distance but no absolute order can be defined, as e.g. for angle values.

This so called eoUniGene genotype is implemented very directly by means of inheritance capabilities offered by C++. The different genes all offer a common basic interface and inherit general properties from a basic gene. To handle the different genes in a list and to access them through the common interface, smart pointer techniques are used.

4.3 Evolutionary Operators

Representing the information in the genotype space is just half of the work to be done when defining a new genotype. Appropriate operators, i.e. initialization, crossover and mutation operators, have to be defined. They must be capable to handle the new genotype. At this point the structure of our heterogeneous genotype proves to be advantageous since all different genes offer a common interface. For example to initialize the whole genotype uniformly, the global operator only calls the uniform initialization operator of every gene through the common interface. The specific initialization routine, implemented for every gene type, is then executed automatically. Everything that is specific to a certain gene type is hidden behind this interface and implemented for every gene type separately. With this approach our heterogenous genotype can be expanded with new gene types by simply providing a new data object and implementing the basic functionality required from the common interface. It is not necessary to make any changes to the genotype itself.

Now it becomes evident why we also implemented cyclic gene properties. From a point of pure information representation it would not have been necessary. But defining specialized operators for cyclic genes can improve the performance of EAs quite drastically. For example, a specific arithmetic crossover operator applied to angle values of 350° and 20° can result in a new value of 0°. Treating the angle as normal float value would result in a new value of 185°, which is not a very reasonable combination of the two parent values.

The current version of the eoUniGene genotype provides the following operators:

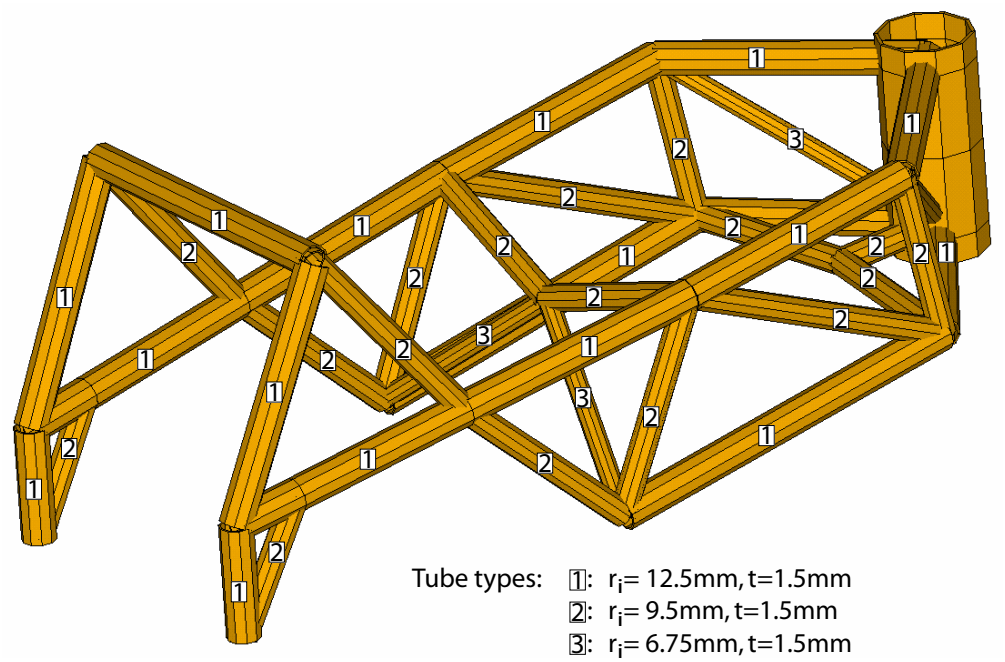

Fig. 3. Beam model of an original frame consisting of 3 different tube types (measured dimensions).

- Uniform and Gaussian initialization.
- Classical one-point, two-point, and n-point crossover.
- Arithmetic crossover (superposition of two individuals with given weights).
- Intermediate and uniform crossover (swapping genes with certain probabilities).
- Uniform and Gaussian mutation.

For further information about the definitions of the different operators we refer to the EO documentation [5].

5 Parameterization of the Tubular Steel Trellis Motorbike-Frame

Only tube dimensions were chosen to be variable for the optimization. This prevents violating any functional or manufacturing constraints of the frame, and results from optimization can directly be compared with the original frame. Due to functional requirements, dimensions of the steering-shaft tube can not be varied. A geometric model of the frame was created in ANSYS and meshed with ordinary 2-node beam elements as shown in Figure 3. The beam elements are of annulus shape defined through the inner radius and the thickness. These parameters represent the first layer of our parameterization concept (see Figure 4). For parameterization, the most simple approach could

[5] http://eodev.sourceforge.net [12]

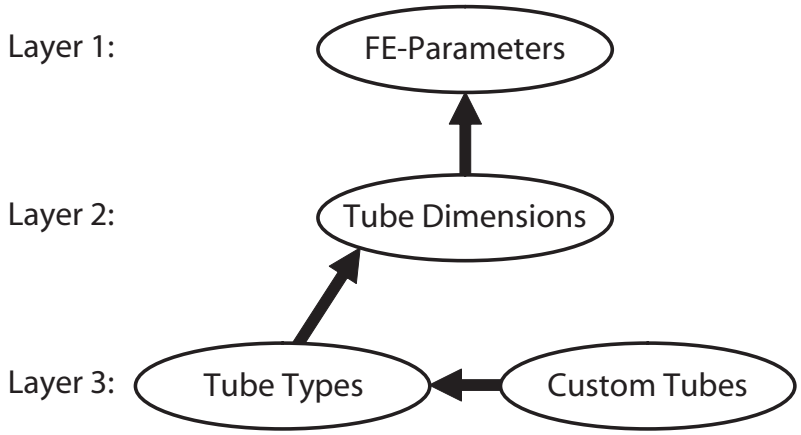

Fig. 4. Different layers for parameterization and how they are linked together.

be to place radius and thickness of every finite element into a long list of float-genes. But then we would have to implement, via penalty mechanism, the constraint that all finite elements of a tube must have the same dimensions in the fitness-function formulation. Because of the reasons outlined in Section 4.1, this would not be very elegant.

The constraint of constant tube dimensions can very easily be included in the parameterization. The first layer is mapped onto a second one, by mapping the dimensions of all finite elements of a single tube to only two parameters of the second level. Further on, symmetry conditions of the frame are included in the parameterization so that identical tubes on each side of the frame are defined through one parameter set. This results in a genotype consisting of 30 float-genes represented by the second layer of Figure 4. The number of evaluations needed in optimization are drastically reduced. First optimization runs are done with this genotype, named the *free-tubes parameterization*. Results are discussed in Section 7.

But as already outlined in Section 1 we want also take into account that for productional reasons it is not possible to choose arbitrary dimensions for each tube. We will consider two possibilities. The first is to only use certain standard tube dimensions. We introduce a third layer, in terms of a list of standard-tube dimensions, into the parameterization. Their dimensions are then mapped to layer two. 13 standard tube dimensions from $R = 8$mm to $R = 14$mm with a step width of $\Delta R = 0.5$mm are given, and for each of them 4 thickness values ranging from $t = 1$mm to $t = 2.5$mm with a step size of $\Delta t = 0.5$mm can be chosen. All together we offer the optimization 52 different tube types arranged in order of their cross section area. The corresponding genotype consists of 15 integer-genes with 1 and 52 as lower and upper limits. Every gene represents therefore a variable tube in the frame. Results for this possible approach named *fixed-tubes parameterization* are presented in Section 7.

Finally, we also consider a parameterization where three custom tube types

are given that could still be produced at reasonable costs. In this case the third layer of parameterization, and therefore our genotype, consists of the dimensions of the custom-tube types represented by 6 float-genes, and for each tube in the frame there is a string-gene specifying which custom tube should be placed at this location. The string-gene type is chosen because we do not know in advance the dimensions of the three custom tube types, and therefore no ordering or norm is given. This configuration is named *limited free-tubes parameterization.*

6 Fitness Evaluation

In the previous section we have presented different parameterization approaches implicitly including the given constraints on tube types to be used. Now, an adequate rating for all the different solutions has to be established. As outlined in Section 1, further demands are to minimize the weight of the frame, to preserve a given torsional flexibility, and to adhere to a strength constraint for braking. In the following we will first detail how these demands can be simulated using FEM. Then we present how we implemented an adequate fitness function consisting of a weighted sum of the different demands.

6.1 Finite Element Analysis of the Motorbike Frame

To analyze the mechanical behavior of the motorbike frame, a Finite Element model is implemented as shown in Figure 5. As outlined before, the frame is modeled in ANSYS using standard 2-node beam elements. Since the engine is clamped rigidly to the frame with 3 bolts, an adequate simulation of the stiffness behavior of the frame has to take into account the stiffness of the engine. But since no geometry data of the engine was available, we first built a rough shell model of the basic engine block. For computational reasons a beam model of the engine block, as shown in Figure 5, was then implemented and tuned to reflect the stiffness behavior of the more accurate shell model. Since the bolts do not carry torsional moments, the torsional stiffness of the appropriate beam elements is set to very small values. The frame is clamped at the connection points to the swing arm and an additional support at the headset. The complete Finite Element model built in ANSYS is then imported into FELyX to evaluate the required properties for all the different solutions in the optimization process.

From this model the weight W of the frame can directly be evaluated by summing up the weight of the finite elements. The torsion stiffness is evaluated by solving a linear static analysis with a torsional moment M_T applied to the

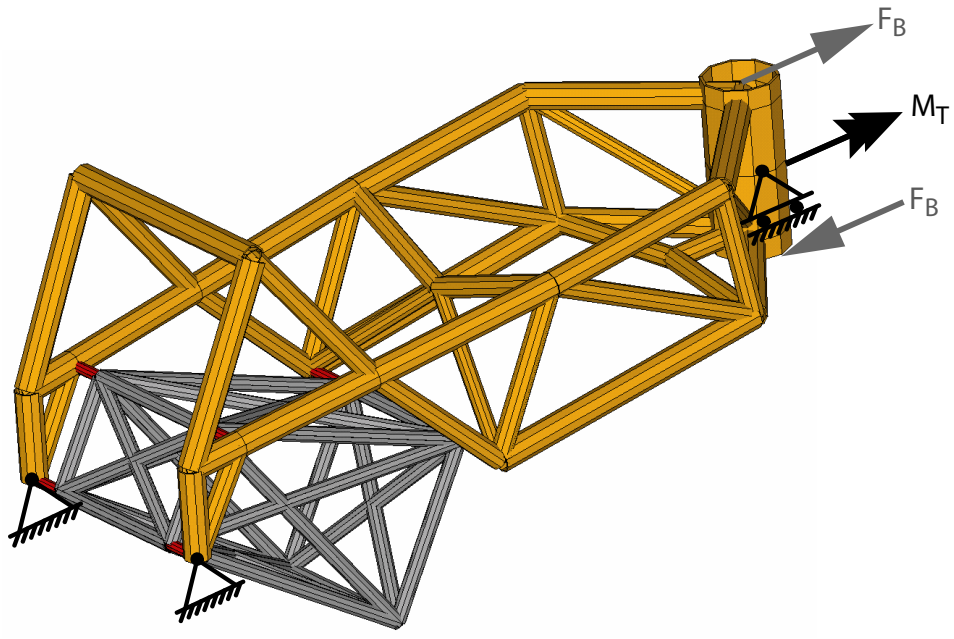

Fig. 5. Finite Element model of frame with engine block and boundary conditions.

headset. A stiffness measure S is then composed of the applied moment M_T and the resulting twist λ_T between swing arm axis and headset axis:

$$S = \frac{M_T}{\lambda_T} \quad \left[\frac{Nm}{\circ}\right] \tag{6}$$

Mechanical stresses in the frame for a braking load case are also needed. This load case can be applied by a pair of forces F_B acting at the headset. After a second linear static analysis the maximum von Mises stress σ_{max} in the frame is then evaluated based on nodal stress values.

6.2 Fitness Function

The fitness function $F(\vec{p})$ depending on the phenotype $\vec{p}$ should give an adequate rating for each possible design with respect to the demands. Therefore the fitness function typically is a weighted sum

$$F(\vec{p}) = \sum_i w_i D_i(\vec{p}) \tag{7}$$

where the $D_i(\vec{p})$ represent the rating for the specific demands and the w_i are the relative weights. The fitness function $F(\vec{p})$ shall be minimized. In order to avoid that one of these terms gets very large and therefore becomes dominant we use only functions $D_i\,(\vec{p})$ that are bounded and we scale them to the interval $[0, 1]$. This also facilitates the adjustment of weight coefficients w_i.

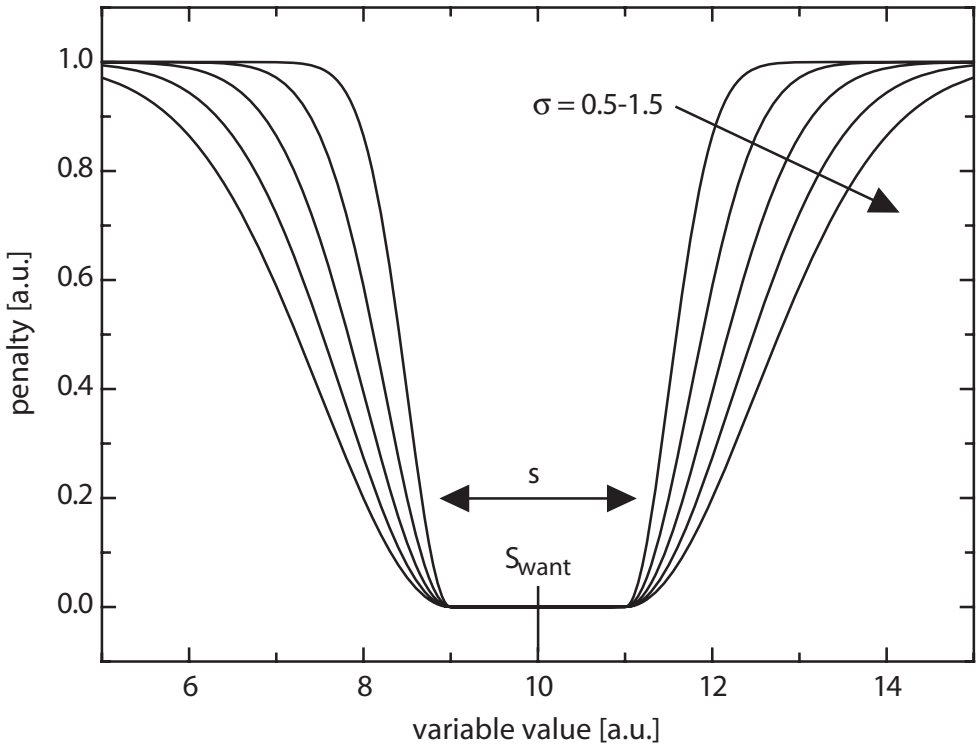

Fig. 6. General penalty function for a target value constraint.

The first demand is to minimize the weight. Normalizing the actual weight with the initial weight we obtain

$$D_{weight}(\vec{p}) = \frac{W(\vec{p})}{W_{init}} \tag{8}$$

where $W(\vec{p})$ and W_{init} are the actual and the initial weights of the chassis.

The second demand is to achieve a given torsional stiffness. It is implemented using a penalty mechanism, i.e. the bigger the difference of the designs torsional stiffness from the demanded one, the larger the penalty should be. For this we use a modified Gaussian function

$$D_{stiff}(\vec{p}) = \begin{cases} 0 & : |S - S_{want}| < s \\ 1 - e^{-\frac{(|S(\vec{p}) - S_{want}| - s)^2}{2\sigma^2}} & : |S - S_{want}| \geq s \end{cases} \tag{9}$$

where S_{want} is the wanted and $S(\vec{p})$ the actual stiffness. The parameter s defines a feasible range for the stiffness and σ determines how fast the penalty increases when the feasible range is left (see Fig. 6). For the frame optimization, a stiffness $S_{want} = 1330\ \frac{Nm}{\circ}$ has to be achieved, and we set $s = 10$ and $\sigma = 70$.

A further demand is that the maximum stress value $\sigma_{max}(\vec{p})$ be under a critical value σ_{crit}. Therefore we formulate the penalty function as

$$D_\sigma(\vec{p}) = \frac{1}{1 + e^{-\gamma(\sigma_{max}(\vec{p}) - \sigma_{crit})}} \tag{10}$$

where $\sigma_{max}(\vec{p})$ is the actual maximum stress and σ_{crit} defines the critical stress for the material. Further γ defines how fast the penalty applies when $\sigma_{max}(\vec{p})$ comes into the range of σ_{crit} (see Figure 7). For the problem at hand, the critical stress was set to $\sigma_{crit} = 470\ \frac{N}{mm^2}$ which value occurs as maximum stress in the original frame. The penalty steepness parameter was set to $\gamma = 0.5$.

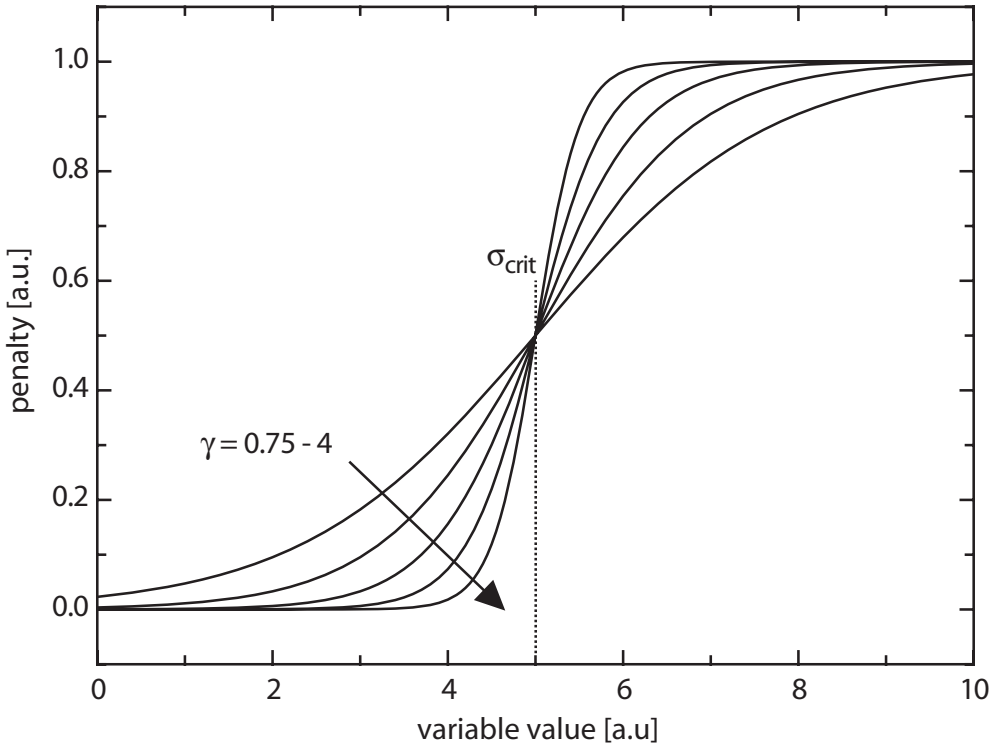

Fig. 7. General penalty function for a upper limit constraint.

Since the chassis is a space-truss construction, most tubes only undergo tension or compression loads. Thereby the fitness of the design is just sensible to the cross-section of the tubes. It is mechanically equivalent whether the EA selects a tube with a big radius and a thin wall or vice versa. But for productional reasons it is favorable to use tubes with thicker walls. Therefore we introduce a further term to our fitness function giving tubes with thin walls a slight penalty as follows

$$D_{thick}(\vec{p}) = \frac{\sum_{i=1}^{N_{tubes}} \frac{1}{(\alpha_i(T_i(\vec{p}) - T_i^{min}) + 1)^{\beta_i}}}{N_{tubes}} \tag{11}$$

where $T_i(\vec{p})$ is the actual thickness and T_i^{min} the minimum allowed thickness for the i^{th} tube. Further α_i allows to adjust the scale and β_i influences the steepness of the penalty (see Figure 8). For our purpose we have chosen $\alpha_i = 1$ and $\beta_i = 3$.

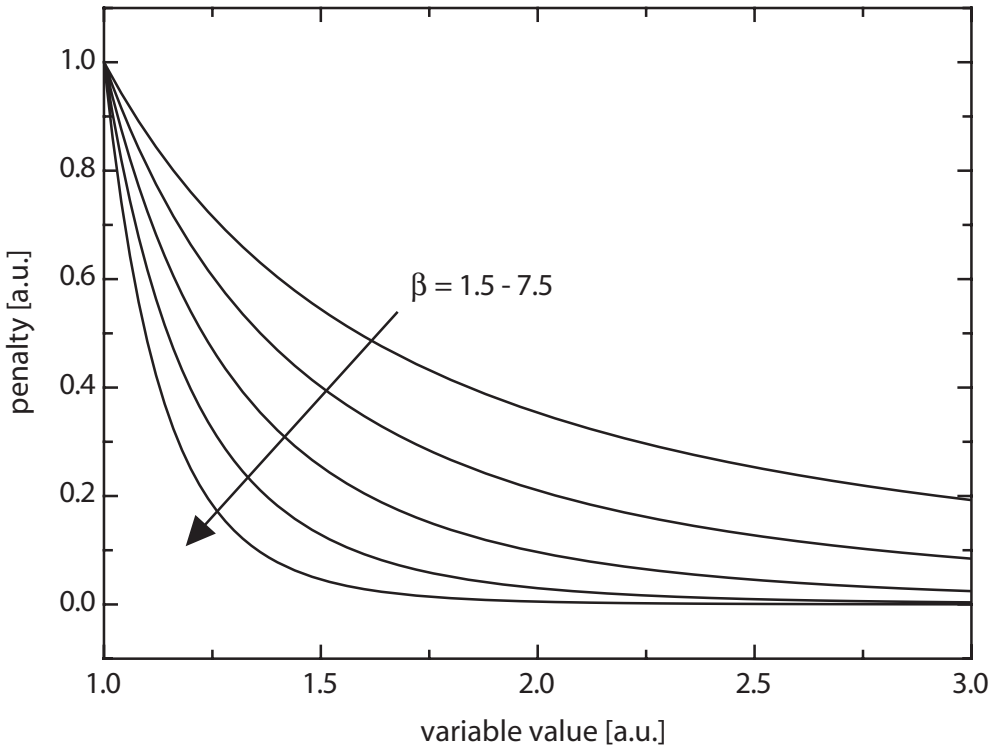

Fig. 8. Slight penalty function for low tube thickness values.

Finally, weight coefficients w_i for the different demands have to be chosen. For the weight, stiffness, and strength demands, the coefficients were set to 1. For the thickness penalty we varied the weight coefficient to investigate its influence on the optimization process.

7 Results and Discussion

For all optimizations presented in this Section the population size was set to 500 and the optimization was run over 300 generations. To reduce the stochastic influence on the results all computations were carried out 7 times and averaged values were taken. In Table 2 the results of the optimization runs for the different parameterizations regarding allowed tube dimensions are presented. The table lists the achieved weight reductions compared to the original weight of the frame. Constraints on torsional stiffness and strength

Table 2
Optimization results for the different parameterization types (averaged values from 7 runs).

parameterization	mass [kg]	improvement	t_0	C [kg]
free tubes ($w_{thick} = 0.0$)	5.89	20.4%	73.3	5.86
free tubes ($w_{thick} = 0.0133$)	5.86	20.8%	43.7	5.88
fixed tubes ($w_{thick} = 0.0$)	6.30	14.9%	16.9	6.29
limited free tubes ($w_{thick} = 0.0$)	5.86	20.8%	50.6	5.86

for the braking loadcase are fulfilled for all optimization results presented. Typical, but not averaged, convergence curves are shown in Figure 9.

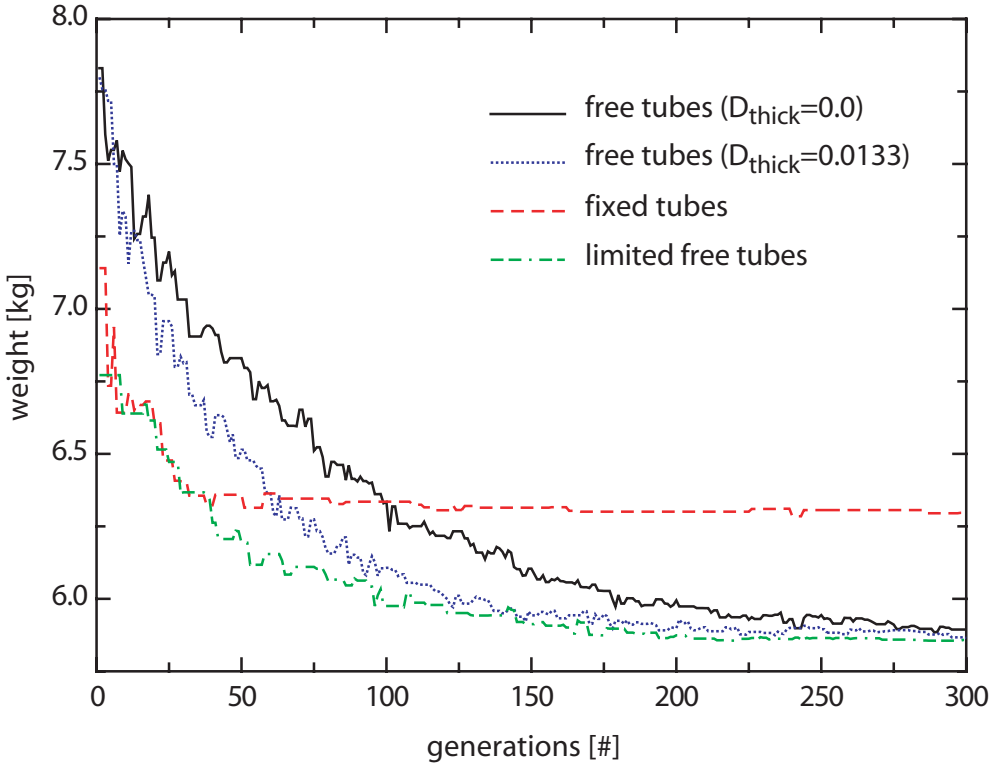

Fig. 9. Convergence curves for the different parameterization types.

The first run (free-tubes parameterization), allowing arbitrary tube dimensions within some limits, reduces the mass of the frame by 20.4% compared to the original frame that has a weight of 7.4kg. Surprisingly, this parameterization with the fewest constraints on tube dimensions does not yield the very best frame that could be achieved. The second optimization run (free-tubes parameterization, $w_{thick} = 0.0133$) with a very small thickness penalty w_{thick} performs slightly better. But from the convergence curves in Figure 9 one obtains that mainly the convergence behavior of the second run is better and

probably not the optimal solution that could be achieved. To quantify this we fitted the curves with the exponential decay function

$$y = A \cdot e^{-\frac{x-x_0}{t_0}} + C \tag{12}$$

and listed the fitted exponential decay times t_0 as well as the theoretically fitted asymptotic target values C where the optimization should converge to in Table 2. From these values we can see that the target value of the first optimization with $w_{thick} = 0.0$ is slightly lower than the one of the second optimization with $w_{thick} = 0.0133$. This is quite expected. Also expected is the far worse decay time t_0 of the run with no thickness penalty. This is due to the fact that most tubes are only under tension/compression loads as already mentioned. Tubes with different radii and wall thicknesses perform equally well as long as their cross-sections are identical. Therefore we obtain a very surjective mapping function in the sense of Equation 5 and the expected worse convergence behavior of the optimization as outlined in Section 4.1.

Looking at the optimization with fixed-tubes parameterization we notice a very fast convergence fostered by the relatively limited genotype space. But the weight is reduced by only 14.9% for this parameterization, a smaller reduction, compared with other approaches.

The genotype allowing three arbitrary custom tube types (limited free tubes parameterization) performs outstandingly well as can be seen in Table 2. For this motorbike frame, producing three well chosen custom tube types should really be considered. The best individual ever found for this parameterization approach is shown in Figure 10. Since no thickness penalty was given for this run the optimization converges to tubes with small thicknesses.

Finally we investigated how far towards thicker tubes we can go without increasing the weight of the frame significant. Therefore a Pareto set was calculated with thickness penalty weights w_{thick} between 0 and 3.16 equally distributed on a logarithmic scale. Again, to reduce stochastic influence, every point of the Pareto set is the averaged result of 7 identical optimization runs. Figure 11 shows that the weight does not increase significantly up to $w_{thick} = 0.1$ although this corresponds to increasing the medium wall thickness by 12.5%.

8 Conclusion and Outlook

We introduced an universal genotype for optimization of heterogeneous parameter lists and applied it successfully to different optimization problem

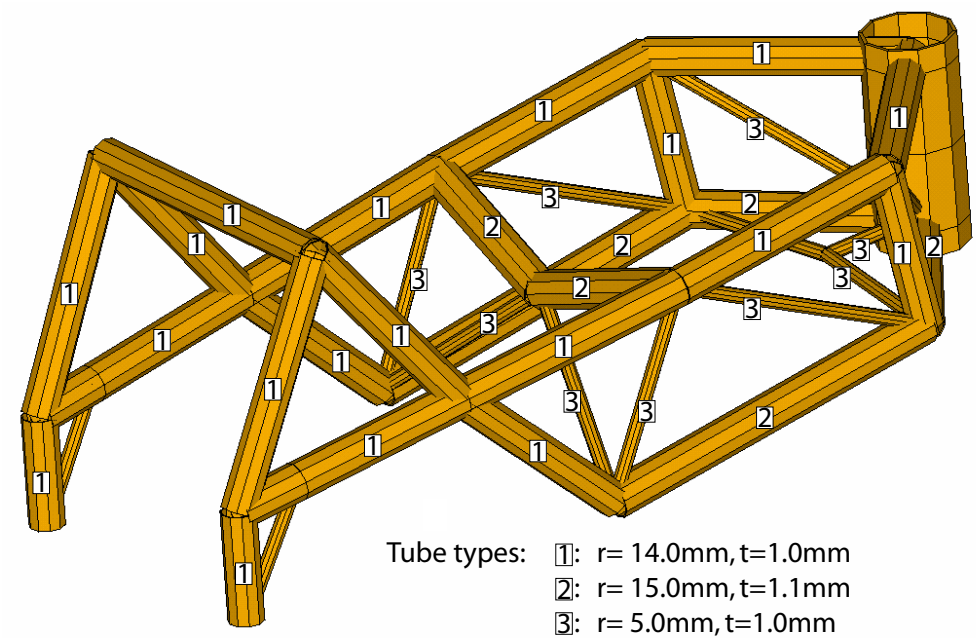

Fig. 10. Best individual ever found for *limited free tubes parameterization*.

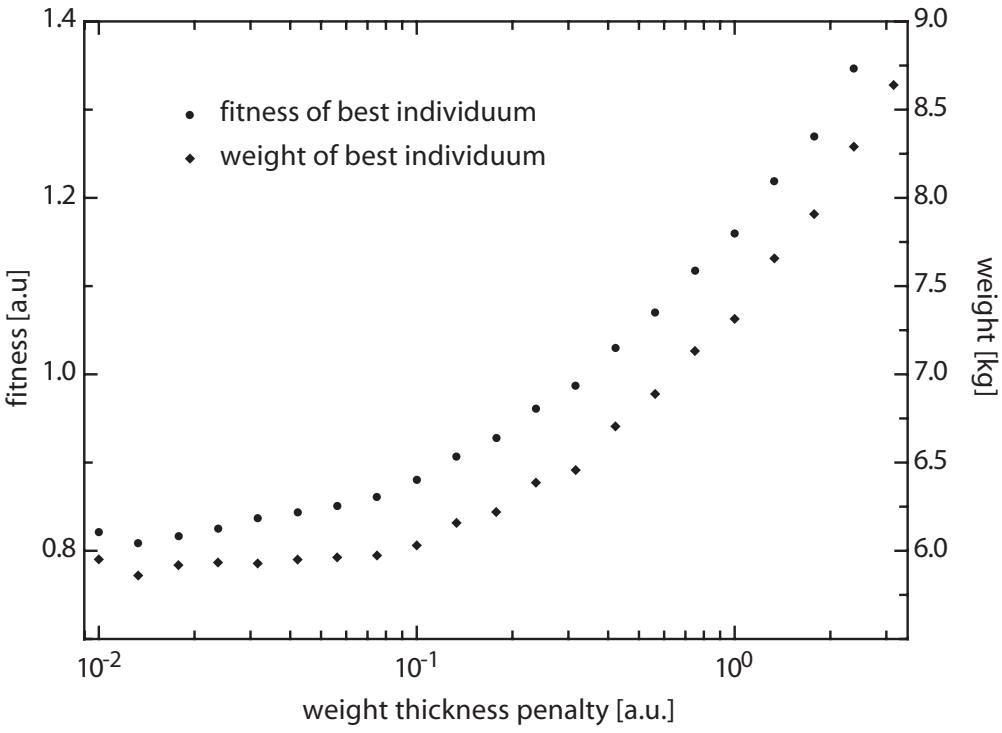

Fig. 11. Pareto set of fitness and weight versus thickness penalty.

formulations in context of a steel-trellis motorbike frame. Further we used normalized function formulation for the different terms contributing to the fitness function. This made it quite easy and understandable to find adequate coefficients for the different terms of the fitness function. Surprisingly, this

frame design, intensively developed during one decade, still shows quite some potential for improvement. We could reduce the weight by 20.8% while preserving the originally desired mechanical properties. This also demonstrates the potential of using this optimization not only for weight reduction, but to easily realize specific mechanical properties of a structure in a very efficient way.

Acknowledgements

This work was supported by the Swiss National Foundation (SNF-project 21-66879.01).

References

[1] J. Holland, Adaptation in natural and artificial systems, University Michigen Press, Ann Arbor, MI, 1975.

[2] D. Goldberg, Genetic algorithms in search, optimization and machine learning, Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1989.

[3] P. Hajela, Stochastic search in structural optimization: Genetic algorithms and simulated annealing, in: Structural Optimization: Status and Promise, Vol. 150, Progress in Astronautics and Aeronautics, 1992, pp. 611–637.

[4] L. Fogel, A. Owens, M. Walsh, Artificial intelligence through simulated evolution, John Wiley, New York, 1966.

[5] I. Rechenberg, Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, Fromann-Holzboog, Stuttgart-Bad Cannstatt, 1973.

[6] I. Rechenberg, Evolutionsstrategie '94, Fromann-Holzboog, Stuttgart-Bad Cannstatt, 1994.

[7] H.-P. Schwefel, Evolutionsstrategie und numerische Optimierung, Ph.D. thesis, TU Berlin (1975).

[8] T. Bäck, Evolutionary algorithms in theory and practice, Oxford University Press, 1995.

[9] J. Koza, Genetic programming: on the programming of computers by means of natural evolution, MIT Press, 1994.

[10] P. J. Bentley (Ed.), Evolutionary design by computers, Morgan Kaufmann Publishers, Inc, San Francisco California, 1999.

[11] M. Schoenauer, Z. Michalewicz, Evolutionary computation: an introduction, Control and Cybernetics 26 (3) (1997) 307–338.

[12] M. Keijzer, J. Merelo, G. Romero, M. Schoenauer, Evolving objects: a general purpose evolutionary computation library, in: EA-01, Evolution Artificielle, 5th International Conference in Evolutionary Algorithms, 2001, pp. 231–244.

[13] O. König, M. Wintermantel, N. Zehnder, P. Ermanni, FELyX - The Finite Element Library eXperiment, Advances in Engineering Software. Submitted for publication.