International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:3, No:11, 2009

Using Memetic Algorithms for the Solution of Technical Problems

Ulrike Völlinger, Erik Lehmann and Rainer Stark
Technical University Berlin and Fraunhofer Institute IPK Berlin

Abstract—The intention of this paper is, to help the user of evolutionary algorithms to adapt them easier to their problem at hand. For a lot of problems in the technical field it is not necessary to reach an optimum solution, but to reach a good solution in time. In many cases the solution is undetermined or there doesn't exist a method to determine the solution. For these cases an evolutionary algorithm can be useful. This paper intents to give the user rules of thumb with which it is easier to decide if the problem is suitable for an evolutionary algorithm and how to design them.

Keywords—Multi criteria optimization; Memetic algorithms

I. Introduction

The use of evolutionary algorithms (EAs) has entered many areas in engineering disciplines beyond computer science. EAs use the benefits of the biological evolutionary process to successively generate an optimal solution. The basic idea behind this is to create a set of potential solution candidates, which undergo a simulated evolutionary process. Terms used in nature like individual, population, selection, mutation, recombination and fitness have different characteristics depending on the design of the EA. In the last couple of years new developments in this areas merge different EAs to hybrid or memetic algorithms. This is done by combining the advantages of the different approaches.
The simulation group of the Virtual Reality lab of the Fraunhofer Institute IPK in Berlin started with the work on EAs in 2006 with a concrete optimization problem, the assignment of stiffness values in mass-spring models. On one hand a lot of papers which describe a specific optimization problem and the use of EAs for the specific adaption to this problem can be found. On the other hand there can be found a lot of books on specialized topics about different kinds of implementations or parameter settings within specific evolutionary operators. But it is difficult to find a paper describing a procedural method, how to compose an EA from scratch and adopt it to a problem. This paper is intended to fill this gap and give the reader some rules of thumb, how to proceed when designing an EA.

The main scope is to help the reader, to find an easier decision about the following issues:

U. Völlinger is with the Institute of Industrial Information Technology, Technical University Berlin, (phone: +49 30 39006111, fax: +49 30 3930246, email:ulrike.voellinger@tu-berlin.de)
E. Lehmann is with the department of Product Modelling, Fraunhofer IPK, Berlin, (email:erik.lehmann@ipk.fraunhofer.de)
R. Stark is head of division Virtual Product Creation, Fraunhofer IPK, Berlin, email:(rainer.stark@ipk.fraunhofer.de)

- Is the problem suitable for EAs
- Choice of algorithm type
- Choice of design
- Choice of parameters and evolutionary operators

At this point it has to be stated clearly, that EAs are not generic problem solvers. If the optimization problem can be solved by a specialized method, than in general with this method better results can be achieved. Therefore evolutionary algorithms should only be used in cases where no specialized algorithms exist. For a lot of problems in the engineering domain it is not necessary to achieve the best solution, but to receive a good solution in an acceptable time. Typical areas of application are:

- Technical design of circuits
- Shortest path selection routing
- Parameter optimization of machine tools
- Forming optimization of mechanical components

Further information about areas of application can be found in [1]. In the next section follows a description of the class of optimization problems for which evolutionary optimization is suitable. Afterward the description of evolutionary algorithms and usable operators and with instructions how to design an EA is given. In section 4 follows a description of a concrete implementation for the earlier mentioned example of the optimization of spring constants in a mass-spring system. Finally a set of rules of thumb for the reader which should be useful when adapting an EA for their own optimization problem is given.

II. Optimization Problems

For the formulation of an optimization problem a search space, an objective function and possible constraints are needed. Whereas the search space S consists of a set of feasible solutions, the objective function f assigns to every element of S a real number, $f : S \rightarrow \mathbb{R}$. In case of minimization of the objective function f the problem is to find $x^* \in S$ such that

$$f(x^*) = \min \{f(x) | x \in S\} . \quad (1)$$

A. Classification

The optimization problem have to be classified according to specified criteria, because of the problem dependency of the various optimization methods. The criteria have to comply with the type of the objective function, the search space and the constraints.

Criteria for classification of optimization problems:

1) Continuous or discrete: depending on whether the optimization parameters consists of of real or discrete values.
2) Constrained or unconstrained: depending on whether the search space is reduced by constraints.
3) Linear or nonlinear: depending on whether the objective function and the corresponding constraints are linear.
4) Multi-criteria optimization: depending on whether the objective function is composed of multiple objectives. Most problems in practice with conflicting objective functions fall into this class, for example the weight minimization of a structure with simultaneously minimization of the deformation. In general a set of possible solutions, the so called Pareto set, is achieved. In solving methods which ascertain only one element of the Pareto set of the entire objective function is constituted over the weighted sum of the individual objective functions.

B. Nonlinear Optimization Methods

Before going into detail about EAs, a short survey of nonlinear optimization methods is given.
The general structure of an iterative optimization method consists of:

1) Initialization: During the initialization process the optimization variables $x_j, j = 1, \ldots, n$ get an adequate initial value.
2) Iteration code: By the iteration rules a sequence $x^i, i = 1, 2, \ldots$ of the variable vector is generated, which cause an improvement in respect to the objective function in each iteration step i.
3) Abort criteria: The abort criteria exist in form of an inequation which is used to decide about the termination or further iteration steps.

The different methods of nonlinear optimization algorithms consists of two classes deterministic and stochastic methods. Furthermore deterministic methods are subdivided into direct and indirect methods. While direct methods use heuristics to determine the search direction, whereas indirect methods work on the basis of partial derivatives of the objective function and therefore require differentiability of the objective function. Stochastic methods are based on the usage of random numbers and probabilities. The benefit of these methods is, that they don't demand an objective function which is continuous and differentiable. In figure 1 the most important nonlinear optimization methods are shown.

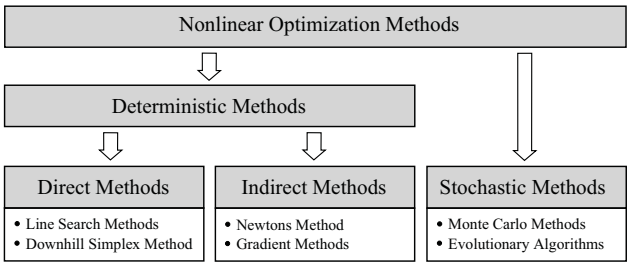

Fig. 1. Nonlinear optimization methods

III. Evolutionary Algorithms

EAs are an optimization method which are orientated on the evolution of living organisms by copying the behavior of the nature in abstract form. A set of solution candidates are created and undergo a simulated evolution. Through variation and selection the solution candidates get better and better until the optimal solution is reached.

EAs have many advantages which justify their usage:

- The objective function does not require continuity or differentiability
- Strong tendency for locating the global optima
- Flexible adaptation possibilities thereby universally applicable
- Handling of complex search spaces (i.e. with a lot of optimization parameters)
- Handling of complex objective functions
- Possibility of parallelization

A. Concept of Evolution

To understand the basic idea of EAs, it is necessary to become familiar with the original terms and concepts of the theory of evolution.
A gene is part of a chromosome, which is the smallest unit of genetic information. Every gene is able to assume different values, each called allele. All genes of an organism form a genome, which affects the appearance of an organism, also called phenotype. The genome and the phenotype combined describe an individual. A set of individuals is called population. The population of the evolution at a point in time is called generation. New individuals can be created by a recombination mechanism. Thereby two individuals are combined by permutation of their alleles to produce one or two new individuals. Additionally to the recombination, mutations can occur, these are caused by a failure during the reproduction of the genotype. Thereby a few alleles of an individual get slightly modified. In principle only those organisms are able to survive, whose characteristics are the most advantageous to organize food, to exploit their habitat and to find mating partners. The weaker organisms get replaced and become extinct, according to the principle of: "survival of the fittest".
Assigning the principles of the biological evolution to an optimization problem, a solution candidate is considered as individual, whose genes are the parameters to be optimized. A set of possible solution candidates compose a population, which is subject to a simulated evolutionary cycle. Therefore each individual is reproduced, varied, and evaluated through the usage of evolutionary operators. By the use of evolutionary operators like recombination and mutation combined with selection, as site-directed evolutionary operator, an evolutionary interplay can be received.

B. Types of Algorithms

Since the middle of the 60s emerged three types of algorithms of EAs independent of each other: genetic algorithms (GAs), evolutionary strategies (ESs) and evolutionary programming (EP). At the beginning of the 90s a further type of algorithm, genetic programming (GP) was established. Although all methods are orientated on the biological evolution, there exists big differences in the realization of the principles of evolution. The difference lies especially in the functioning of the evolutionary operators. Whereas the initialization, the evaluation and the abort criteria are the same in all methods. The genetic algorithms are based on the work of HOLLAND and DE JONG [2], [3]. In the foreground of these algorithms stands the recombination operator, whereas the mutation operator is inferior. This can be seen like in the biological evolution where mutations also do appear with minor probability. Evolutionary strategies were developed by RECHENBERG and SCHWEFEL for optimization problems in the technical physical field [4], [5]. In these methods the optimization process is brought forward by mutation, while the recombination plays a minor role. The evolutionary programming is based on the work FOGEL, OWENS and WALSH and further developed by FOGEL [6], [7]. As biological variation mechanism only the mutation operator is used. Genetic programming was developed in combination with genetic algorithms mainly by KOZA [8]. In this method the recombination is the primary and the mutation the secondary operator, just like in genetic algorithms.

In the last couple of years new developments in this research area dealt mainly with the combination of different EAs to hybrid or memetic algorithms (MAs). The aim of this research is to resolve the deficits of classical genetic algorithms like slow convergence by combination of the advantages of diverse methods. The name memetic algorithm leads back to the concept of meme developed by DAWKINS [9]. The meme are regarded as units of cultural evolution and have the ability of local improvement unlike the genes in the genetic evolution. Through this concept further evolutionary parameters like learning for lifetime, learning through imitation and cultural transmission of knowledge are taken into account.

C. Memetic Algorithms

In this section we will discuss memetic algorithms (MAs) and their properties in more detail. The population size μ, with common values between 5 and 100 individuals, has a great influence on the run of the MA. The population size decides how many genes are provided in the gene pool for recombination and mutation. Another important parameter is the number of generated children λ. In MAs $\mu = \lambda$ is true for most cases. In figure 2 the entire evolutionary cycle of a MA is shown.

At the beginning of the optimization a start population with solution candidates is generated through an appropriate initialization. The individuals are evaluated by an fitness function which assigns every individual a particular quality. It is necessary that the fitness function is reasonable and well-defined. Therewith, we are able to compare two individuals and to decide if an individual is better than another one.

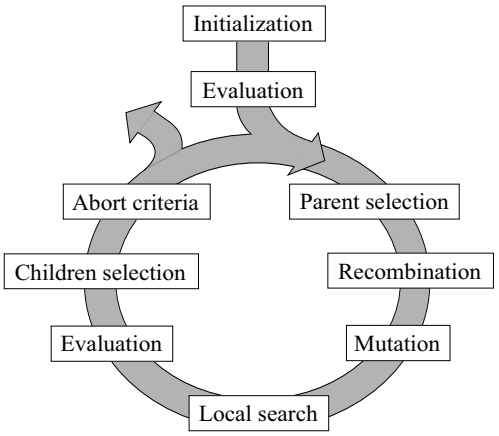

Fig. 2. Evolutionary cycle of a MA

During parent selection the search for an optimum gets a definite direction, whereby the optimization is gradually focused on the relevant areas of the search space. In the course of parent selection some individuals are selected from the current population for recombination. The selection probability of each individual should depend on their fitness. Well-known methods for parent selection are roulette wheel selection, stochastic universal sampling and tournament selection.

The recombination is the primary operator, its task is to cross the search space efficiently. By adequate methods regions of the search space with upper average quality will be reached faster than by random search. All recombination methods is common that at least two parent individuals are crossed with a certain probability, they produce at least one child individual. Well-known methods are multipoint crossover, uniform crossover and arithmetic crossover. In addition to the recombination method, the recombination probability is of significance with frequent used values between 0.7 and 1.0.

By the mutation, as secondary operator, the alleles of the produced child individuals are modified slightly, which enables further exploration of the search space. An often used mutation probability is 1/n, with n as amount of the genes of an individual. The most popular mutation methods are: shuffle mutation, uniform mutation and non uniform mutation.

By introducing local search a faster convergence of the EAs is reached in most cases. The local search is based on the concept of learning for lifetime. With local search the best child individuals are selected and locally optimized. For the selected individuals optimization steps are accomplished by usage of Hill Climbing or Simulated Annealing. Thereby the alleles of the individuals become slightly modified by a new usage of the mutation operator, whereas an improvement of the quality of the individual shall be gained.

After the creation of the children, it has to be decided which individuals should be taken into the next generation. The methods of child selection are not based on random numbers, but exclusively on the fitness of the individuals, in contrast to the parent selection. According to this, individuals with good quality will be preferable taken to the next generation.

The most popular methods are: generational replacement and steady-state.
At the end of the evolutionary cycle an abort criteria decides if the evolution process should be stopped. Typical abort criteria are the achievement of a certain quality value or of a fixed number of generations.
Detailed descriptions of the methods for selection, recombination and mutation are provided in [10], [11], [12] and [1].

D. Design of Algorithms

In case of an optimization problem, the use of EAs for finding a global optimum is a good choice, if a traditional optimization method only deliver a local solution or if a specialized method doesn't exist at all. Prior to the use of an EA, an adaption to the optimization problem has to be done. Therefore it is necessary to set up a pilot survey with serial tests to find the optimal settings for the specific problem like population size or evolutionary operators. By ascertaining adequate and inadequate methods the effort of pilot surveys could be reduced. The design of an EA requires the following steps:

1) Classification of the optimization problem
2) Assignment of an algorithm type
3) Pilot survey
   a) Creation of a test model
   b) Finding constraints of the search space
   c) Creation of the evaluation function (the objective function of the optimization problem)
   d) Creation of an adequate initialization of the start population
   e) Choice of an adequate fitness function for the parent selection
   f) Choice of adequate mutation operators (e.g. physical or topological)
   g) Exclusion of inadequate methods for variation and selection
4) Parameter detection through setting up of serial tests

The serial test for parameter detection are evaluated by the quality of the best individual, the speed of convergence and the computing time. The quality of the best individual is measured by its objective function value. For the speed of convergence the number of required generations to reach a given quality can be used. In the following section we describe the set up of pilot surveys for a concrete problem at hand in order to reduce the effort to a minimum by ascertaining adequate and inadequate methods.

IV. Optimization of Stiffness Constants in Mass-Spring Models

Below the practical adoption of memetic algorithms is shown by using the example of the optimization of stiffness constants in mass-spring models (MSM). MSM are often used in Virtual Reality applications, because of their conceptual simplicity and their computational speed. They approximate a continuous body by a finite set of mass points, which are connected by massless spring elements, usually modeled as damped springs. The motion of the point masses are described by a differential equation system which are solved by methods such as the Runge-Kutta method. Whereas the elastic deformation characteristics are controlled by the chosen stiffness constants.
The main problem during the creation of the MSM consists in choosing the right stiffness constants for the spring elements. The basic material parameters young modulus and mass can not be taken directly into the model. Assigning all spring elements with the same stiffness constant leads in general to a wrong deformation simulation. Furthermore there exists no analytic solution to this problem in general as it was shown by van Gelder [13]. In the last couple of years therefore a lot of researchers used optimization algorithms to solve this problem [14], [15], [16].

A. Classification of the Optimization Problem

The problem at hand is a continuous multi-criteria optimization problem, for which a memetic algorithm can be adapted. In the developed approach an individual consists of the spring constants $k_i \in \mathbb{R}, i = 1, \ldots, m$ of an MSM. Therefore we receive $S = \mathbb{R}^m$ as search space. In order to build the objective function, nine different natures of load have to be considered to describe all possible deformations of a body in 3 D space, [17]. Therefore we generated FEM reference deformations for all cases by using NX Nastran.
Let $k \in S$ be the vector of the spring constants. With k nine deformation calculations for each specific kind of load has to be carried out. After a fixed number of simulation steps $N > 0$ the MSM is in an equilibrium state and it is possible to compare the simulated point positions $x_i, i = 1, \ldots, n$ with compatible positions $x_i^{ref}, i = 1, \ldots, n$ in the reference model.
In [18] it is recommended to use the maximal distance of a point as factor of the standard deviation, in order to distinguish the quality of two simulation results with the same standard deviation.
Which leads to the following objective function f for the nine reference deformations:

$$f(k) = \sum_{j=1}^{9} \left(r_{max}^{j} \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left\| x_i^{ref_j} - x_i^{j} \right\|_2^2} \right). \qquad (2)$$

B. Classification of the Algorithm Type

From the four different main classes of EAs, genetic algorithms, evolution strategies, evolutionary programming and genetic programming, only genetic algorithms and evolution strategies are suitable for continuous optimization problems. Therefore the other algorithms were excluded from further investigation.

C. Pilot Survey

First an workpiece was chosen, modeled as MSM and a corresponding FEM model was created. The material properties for the chosen workpiece and the characteristics of the corresponding MSM model can be seen in tables II and III

in appendix A. Ahead of the start of the optimization process the number N of simulation steps, until the MSM reach an equilibrium state, have to be determined. This test has to be done for each MSM.
Constraints to the underlying problem are the range of the stiffness constants, due to experience it can be limited to $0 = k_{min} \leq k_i \leq k_{max} = 1000$. In general it can be stated that as much as possible constraints should be defined to reduce the search space. For the evaluation of each individual the above objective function is used.
The next step is to create a start population, in the present case a population with approximately 20 individuals is sufficient. Two different initializations for the individuals have been tested. First all spring constants are set to the same value and second all spring constants are set to randomized values within the above range.
For recombination three selection methods, roulette method, stochastic universal sampling and tournament selection, have been tried, whereas tournament selection delivers the best results. The selected individuals are then used to create new individuals by the uniform crossover method. Another important issue is the choice of the mutation method. In the present case this method enables further exploration of the search space by two operators. The first operator picks a spring at random of an individual and assigns a new value to them. The second operator creates or deletes springs at random within an individual. This leads to the development of new topologies. Whereas this is especially important if an individual consists of a MSM, which is coarsely meshed or contains coarsely meshed parts. In this case it is difficult to adjust the deformation behavior by applying new stiffness constants. The described operators are furthermore used by the local search algorithm.

D. Determination of parameters using tests

In order to find the best evolutionary algorithm for the problem, a serial test for every adjustment of the parameters has to be done. For evaluation of the serial test, three criteria, quality of the solution, speed of convergence and calculating time have been used. After the determination of the best parameters for three different types of evolutionary algorithms, genetic algorithm with and without local search and evolutionary strategies, could be tested.

E. Results of the Optimization Process

With genetic algorithms individuals with higher quality could be received whereas with evolution strategies the speed of convergence was better. With the adapted algorithms tetrahedron and hexahedron meshes could be optimized. The simulation behavior of a MSM is besides the choice of stiffness constants dependent on the mesh resolution and the topology. It is not possible to reach a physical plausible behavior according to reference deformations for every topology. Therefore especially for tetrahedron meshes it is important to optimize the topology as well inside the optimization algorithm. This was done by the mutation operator. Furthermore the convergence to a good solution could be fasten up by inclusion of local search. The quality of the individuals could be optimized of approximately 90 % with both types of meshes. In figure 3 an hexahedron MSM after the optimisation process can be seen.

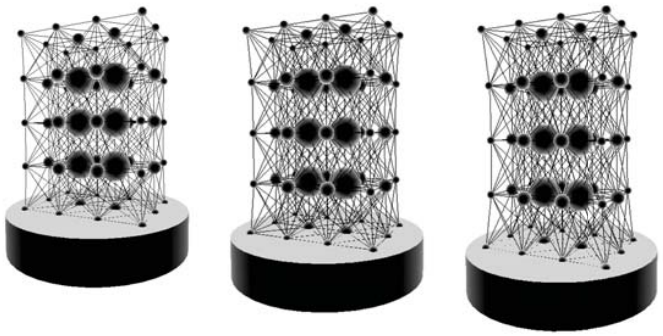

Fig. 3. Optimized MSM with different loads

V. Conclusion

In this paper, an overview of EAs has been given and MAs has been discussed in detail according to continuous optimization problems. The most popular methods for selection, recombination and mutation have been mentioned. Furthermore the important steps for developing an EA for a concrete technical problem has been shown and has been demonstrated on the optimization of stiffness constants of mass-spring models. In table I the parameter settings of the adapted algorithm are shown. These parameter settings were successful at the present problem, but they are not necessarily the best settings in general. In another optimization problem other settings might be better. According to this a concrete EA is not valid in general. An adaptation to the concrete optimization problem is essential.

TABLE I
Parameter settings of our algorithm

Parameter	Setting
Population size	20
Parent selection	Tournament selection
Recombination	Uniform crossover
Mutation	Non uniform mutation
Local search	Simulated annealing
Child selection	Generational replacement

Nevertheless a set of rules of thumb should help the reader to adapt an EA for a concrete optimization problem:

1) If the optimization problem is continuous genetic algorithms, evolutionary strategies or memetic algorithms should be used. Otherwise in case of discrete optimization evolutionary or genetic programming should be used.
2) In case of continuous optimization a memetic algorithm with parameter settings according to table I could be a useful start configuration for problem adaptation.

3) The population size should be as low as possible and as high as necessary. With upper population size a better solution and a faster speed of convergence is expected, but more computing time is required (linear increase).
4) The choice of the mutation operator should be considered very carefully. Dependent on the optimization problem the mutation operator can have different shapes. At the present problem two different types have been used: a topological and a physical mutation operator. In addition to this the mutation step size should be monotonically decreasing in the course of the evolution. At the beginning of the optimization a large step size is profitable to make large steps through the search space. At the end of the optimization a small mutation step size is better.

In general EAs can be used for many problems, but they might not always be the best choice. The adaptation of an EA to the concrete problem can cost a lot of time. In case the objective function is continuous and differentiable derivative based methods such as the Newton method or the Gradient method might provide a faster solution. In case the objective function does not fulfill this requirements or a complex objection function is given, the EAs may reach good optimization results according to a global optimization.

ACKNOWLEDGMENT

Part of this work is funded by the German Research Society (DFG) during the project 'memetic algorithms for efficient model optimization for realtime deformation simulations'.

APPENDIX A

TABLE II
CHARACTERISTICS OF THE OBJECT

Characteristic	Values
Dimensions (x/y/z)	0.06m/0.08m/0.04m
Total mass	76.8g
Material consistence	homogeneous isotropic
stress-strain behavior	linear elastic with $E = 7 \cdot 10^4 N/m^2$ and $\nu = 0.32$

TABLE III
CHARACTERISTICS OF THE MSM

Characteristics	Values
mesh resolution	0.02m
Number of mass points	60
Number of spring elements	425

REFERENCES

[1] K. Weicker, *Evolutionäre Algorithmen*, 1st ed. Teubner Verlag, 2002.
[2] J. H. Holland, *Adaption in natural and artifical systems*. University of Michigan Press, Ann Arbor, 1975.
[3] K. A. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems," Ph.D. dissertation, University of Michigan, 1975.
[4] I. Rechenberg, *Evolutionsstrategie. Optimierung technischer Systeme nach Prinzipien der biologischen Evolution*, 1st ed. Frommann-Holzboog, 1973.
[5] H.-P. Schwefel, "Evolutionsstrategie und numerische optimierung," Ph.D. dissertation, Technische Universitt Berlin, 1975.
[6] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artifical intelligence through simulated evolution*. Wiley, New York, 1966.
[7] D. B. Fogel, "Evolving artifical intelligence," Ph.D. dissertation, University of California, San Diego, 1992.
[8] J. R. Koza, *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University Computer Science Department, 1990.
[9] R. Dawkins, *The selfish gene*, new ed. Oxford New York: Oxford University Press, 1989.
[10] . E. Eiben, R. Hinterding, and Z. Michalewicz, *Parameter Control in Evolutionary Algorithms*. IEEE Computer Society, 1999.
[11] I. Gerdes, F. Klawonn, and R. Kruse, *Evolutionre Algorithmen*, 1st ed. Vieweg Verlag, 2004.
[12] E. Schneburg, F. Heinzmann, and S. Feddersen, *Genetische Algorithmen und Evolutionsstrategien*, 1st ed. Addison-Wesley, 1994.
[13] A. van Gelder, *Approximate simulation of elastic membranes by triangulated spring meshes*. J. Graph. Tools, 3(2):21-42, 1998.
[14] A. Nealen, M. Müller, R. Keiser, E. Boxermann, M. Carlson, D. W., and C. Helmberg, "Physically based deformable models in computer graphics," *cgforum*, pp. 71–94, Jul 2005.
[15] G. Bianchi, B. Solenthaler, G. Szkely, and M. Harders, "Simultaneous topology and stiffness identification for mass-spring models based on fem reference deformations," 2004.
[16] U. Völlinger, H. Stier, J. Priesnitz, and F.-L. Krause, "Evolutionary optimization of mass-spring models," *CIRP Journal of Manufacturing Science and Technology*, vol. 1, pp. 137–141, December 2008.
[17] O. Deussen, *Untersuchung effizienter Verfahren zur Bewegungssimulation deformierbarer Körper*, fortschritt-berichte vdi reihe 20 nr. 215 ed. VDI Verlag, 1996.
[18] V. D. I. VDI, *Konstruktionsmethodik - Technisch-wirtschaftliche Bewertung*, vdi richtlinie 2225, blatt 3 ed., 1998.

Ulrike Völlinger Ulrike Völlinger received a diploma in Computer Science in 2005. She is currently working as research assistant at the chair for Industrial Information Technology and the division Virtual Product Creation, Fraunhofer IPK, Berlin. Besides she is teaching at Technical University Berlin and doing her PhD. Her research interests include physics based simulation of deformations, optimization and numerical mathematics.

Erik Lehmann Erik Lehmann received a degree in Mathematics from the Beuth University of Applied Sciences in Berlin, Germany, in 2008. He is currently pursuing the Master's degree in Computational Engineering at the same University. His research interests include numerical mathematics, optimization, CAD and FEM.

Rainer Stark Prof. Rainer Stark is director of the chair for Industrial Information Technology at Technical University Berlin and of the division Virtual Product Creation, Fraunhofer IPK, Berlin since 02/2008. After his studies in mechanical engineering (Ruhr University Bochum and Texas A&M University), he worked as research assistant at the chair of design engineering of the University of Saarland (from 1989 to 1994) where he received his PhD in 1994. From 1994 to 2002 he worked as system engineer in the department of body design at Ford AG Cologne. In the following he focused on methods in virtual product creation and was a technical expert for CAD, modeling of product data and product information management. In 2002 he was nominated Technical Manager of the Department of Virtual Product Development and Methods of Ford Motor Company, where he worked until January 2008.