Genetic Algorithms in Computer-Aided Design

Gábor Renner

Hungarian Academy of Sciences, renner@sztaki.hu

ABSTRACT

Genetic algorithms constitute a class of search algorithms especially suited to solving complex optimization problems in engineering. In addition to parameter optimization, genetic algorithms are also suggested for solving problems in creative design, such as combining components in a novel, creative way. Genetic algorithms (GA) transpose the notions of evolution in Nature to computers and imitate natural evolution. Basically, they find solution(s) to a problem by maintaining a population of possible solutions according to the 'survival of the fittest' principle. We present here the main features of GAs and several ways in which they can solve difficult design problems. We briefly introduce the basic notions of GAs, and discuss how GAs work. We then give an overview of applications of GAs to different domains of engineering design.

Keywords: Genetic algorithms, CAD, Optimization, Geometric design

1. INTRODUCTION

Designing a new product, i.e. creating new technical structures characterized by new parameters, consists of several phases. They differ in several details such as depth of the design, kind of input data, design strategy and procedures, and results. The design steps can often be interpreted as solving optimization problems. In this case a structure and/or a set of parameters is sought, which results in the best value of some attribute characterizing the quality of the design.

Analytical or numerical methods for calculating the extremes of a function have been applied to engineering computations for a long time. Although these methods may perform well in many practical cases, they may fail in more complex design situations. In real design problems the number of design parameters can be very large, and their influence on the value to be optimized (the goal function) can be very complicated, having nonlinear character. The goal function may have many local extrema, whereas the designer is interested in the global extremum. Such problems cannot be handled by classical methods (e.g. gradient methods) at all, or they only compute local extrema. In these complex cases stochastic optimization techniques including evolutionary algorithms such as genetic algorithms may offer solutions to the problem; they may find a solution (a design) near to the global optimum within reasonable time and computational costs.

Gradient methods start from a single point in the search space (a solution to the design problem), and search for a better solution in the direction of the gradient of the goal function (hill climbing). The method is efficient, because it requires just a few evaluations of potential solutions, which may be crucial in complex engineering problems. However, gradient methods have several difficulties. The basic problem is that gradient methods find only a local optimum, and no information is available on how good it is compared to the global one. Moreover, the local optimum found depends on the starting point; to improve results the computation is usually repeated for a number of starting points. Gradients of the goal function must be computed (analytically, or at least numerically), which implies that the goal function must be smooth. In real design problems — with complicated or possibly discontinuous goal functions, and discrete variables — these conditions are in general not fulfilled.

The simulated annealing method eliminates some of the disadvantages of the gradient method. In this stochastic search method a new solution is obtained by perturbing the current solution. If the goal function value of the new solution is better than that of the previous solution, then it is accepted. A solution may also be accepted, however, which produces a worse value of the goal function. The probability of accepting a worse solution is reflected in the temperature of the system. The temperature is gradually lowered as the search proceeds through an annealing process (e.g. following

Boltzmann's law), thus allowing acceptance of worse solutions with greater probability at the beginning and with smaller probability later. The advantage of simulated annealing is that there is a good chance of finding the global optimum and that the solution does not depend on the starting point. However, simulated annealing method requires higher computational effort than the gradient method.

Genetic algorithms strongly differ in conception from other analytic and stochastic search methods, including gradient and simulated annealing methods. The basic difference is that while other methods always process single points in the search space, GAs maintain a population of potential solutions.

GAs constitute a class of search methods especially suited for solving complex optimization problems [3],[18],[22],[34]. They transpose the notions of natural evolution to the world of computers, and imitate natural evolution. They were initially introduced by J. Holland [22] for explaining the adaptive processes of natural systems and for creating new artificial systems that work on similar bases. In Nature new organisms adapted to their environment develop through evolution. Genetic algorithms *evolve* solutions to the given problem in a similar way. They maintain a collection of solutions — a *population* of individuals — and perform a multidirectional search. The individuals are represented by chromosomes composed of *genes*. Genetic algorithms operate on the chromosomes, which represent the inheritable properties of the individuals. By analogy with Nature, through selection the fit individuals —potential solutions to the optimization problem — live to reproduce, and the weak individuals, which are not so fit, die off. New individuals are created from one or two parents by mutation and crossover, respectively. They replace old individuals in the population and they are usually similar to their parents. As a consequence, in a new generation there will appear individuals that resemble the fit individuals from the previous generation.

2. THE GENETIC ALGORITHM

Genetic algorithms – as representatives of artificial evolutionary systems (1) maintain a population of solutions, (2) allow the fitter individuals to reproduce, and (3) let the less fit individuals die off. The new individuals inherit the properties of their parents, and the fitter ones survive for the next generation. The final solutions will be much better than their ancestors from the previous generations.

The process of evolution is directed by fitness. The evolutionary search is conducted towards better regions of the search space on the basis of the fitness measure. Each solution in a population is evaluated based on how well it solves the given problem.

GAs use a separate *search space* and *solution space*. The search space is the space of coded solutions, i.e., *genotypes* or *chromosomes* consisting of genes. The solution space is the space of actual solutions, i.e., *phenotypes*. The *genotype* must be transformed into the corresponding *phenotype* before its fitness is evaluated.

2.1 The Genetic Process

Solving a problem with GA starts with designing a proper representation, fitness measure and termination criterion. Many representations are possible for a given problem, some are better than the others, however. The termination criterion usually allows at most some predefined number of generations and checks whether an acceptable solution has been found. The genetic algorithm then works as follows (Fig 1):

```
Create initial population
Evaluate initial population
repeat
        Create new population
          •  select individuals for mating
          •  create offspring by crossover
          •  mutate selected individuals
          •  keep selected individuals from
               previous population
        Evaluate new individuals
until termination criteria satisfied
```

Fig 1. Life cycle of the genetic process

1. The initial population is filled with individuals that are generally created at random.
2. Individuals in the initial population are evaluated using the fitness measure.
3. From the current population individuals are selected for reproduction, based on the fitness values of the individuals. Different types of selection mechanisms can be used (e.g. fitness proportional, ranked, tournament selection).
4. New individuals (offspring) are created by applying the genetic operators to parent individuals. Reproduction copies selected individuals from the current population. Crossover combines the genetic code of two parent individuals. Local changes are introduced into the genetic code of one individual by mutation. Different types of crossover and mutation operators can be used according to the features of the specific problem (some of them are shown in Fig. 2).
5. New individuals are evaluated using the fitness measure. New population is created by extending the current population with the new individuals and then omitting the least fit individuals.
6. If the termination criterion is met, the best solution is returned.

7. Steps starting from 3. are repeated until the termination criterion is satisfied. An iteration is called *generation*.

The above process can easily be transferred into an algorithm and a computer code. To predict the behavior of a GA, especially on a specific problem in a complex, highly nonlinear domain, is very difficult — if not impossible. However, there are theoretical results highlighting why and how GAs work for idealized settings. The so-called *schema theorem* [18], [22] states that previously evolved good parts of solutions (schemata) appear at exponentially increasing rates in consecutive generations.

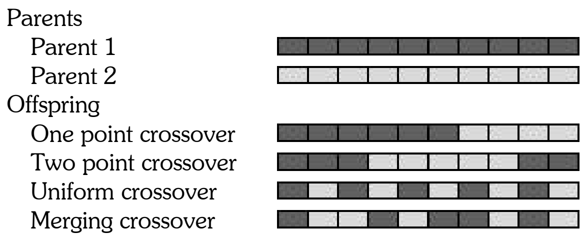

Fig. 2. Crossover operations

A more detailed discussion of the genetic process and its components can be found in [34] and [40].

2.2 Constraints in Genetic Algorithms

Problems are of particular interest in design where optimization and constraint satisfaction are coupled. In many cases the constraints can be expressed as well-defined intervals for the design parameters, but sometimes it is quite difficult to specify them (e.g. forbidden regions in robot path design). Several techniques have been developed to introduce constraint handling into different components of GAs [35].

A popular method of constraint satisfaction in GAs is to reject individuals that violate constraints, i.e., the infeasible individuals. Infeasible individuals that appear as the result of the genetic operators are not admitted to the new generation.

If the initial population consists of infeasible individuals only, they could be repaired instead of being rejected. The disadvantage of this method is that for each problem a specific repair algorithm must be devised. Meanwhile, evaluation can be tuned in such a way that individuals slightly violating the constraints are still accepted.

A frequently applied technique for handling constraints is to apply a penalty term in the fitness function. Individuals that do not fulfill the constraints are given penalties that depend on the extent of violating the constraints. Selection is based on a weighted sum of fitness and penalty. Thus, the infeasible individuals participate in the genetic process, as they are still considered capable of delivering useful offspring. However, careful adjustment of the penalty weight is needed. If the penalty weight is too low, "very" infeasible individuals could be preferred to slightly less fit but much more feasible individuals. On the other hand, application of high penalty weight may push promising individuals out of the population, and the process may converge to feasible but unfit individuals. In many cases a good strategy is to start with relaxed constraints, i.e., low penalty weights, and then continue with strengthened constraints, i.e., higher penalty weights, as the GA proceeds, thus ensuring a path to promising solutions.

Another approach is to incorporate all constraints into the genetic representation, i.e., to construct a representation which does not allow any individual to violate any constraints. However, incorporating too much problem specific knowledge into the representation largely limits the size of the search space and may require the careful definition of specific crossover and mutation operators.

2.3 Advanced Genetic Algorithms

Genetic algorithms work well for many practical problems. In their application to complex design problems, however, simple GAs may converge slowly, evaluations may be computationally intensive, or GAs may fail because of convergence to an unacceptable local optimum. Considerable research effort has been made to improve the efficiency of GAs, which has resulted in advanced genetic algorithms. The most important extensions shown to be advantageous in the application of GAs to design problems are multiobjective GAs, parallel GAs (including injection island GAs) and methods for preventing the population from converging too early to some local optimum (niching methods, fitness sharing, speciation). Discussion of these methods can be found in [40]. An overview of other useful extensions of GAs is given by Bentley [3].

3. GENETIC ALGORITHMS IN DESIGN

Engineering design — as an intelligent activity — can be characterized as a goal oriented, constrained, decision making process [16], which is aimed at creating artifacts (products) that satisfy well-defined human needs. Expectations and requirements concerning the product are described in design specifications. The design process can be seen as the transformation of the specifications into design descriptions. The design description must contain sufficient information (numerical, graphical, and symbolic) for manufacturing the product. Functionality and manufacturability impose constraints on the structure and parameters of the product.

Engineering design typically involves exploration and learning. While exploration is needed to identify what kinds of structures and variables are appropriate to fulfill the requirements, learning attempts to use experience gained from previous design processes and from emerging solutions.

Design can be conceived as a search for a suitable or optimal construction, where the term *search* is used in a technical sense. A search problem consists of a desired state (goal state), a search space and a search process. In design the goal state represents the characteristics of the final design. The goal state is consistent and complete: its characteristics are non-conflicting and fully specify the final design. The search space is the set of all designs characterized by all possible (or allowable) values of the design parameters. The search process (deterministic or heuristic) consists of searching for the goal state in the search space, in this case searching for the optimal design in the space of all designs.

The relationship between the functional requirements and the structure needed to satisfy those requirements is known in many design situations. In this case — referred to as routine design — the parameters allowing variation in the design are also known. The design task consists of defining appropriate values for the parameters, which frequently means searching for their optimal values.

The concepts of routine design can be directly mapped to genetic algorithms: the parameters are encoded as genes that form chromosomes, which are evolved by the genetic algorithm. There is a close relation — usually a one to one mapping — between the chromosomes (genotype) and the parameters (phenotype). Addition and deletion of genes (parameters) is generally not performed during evolution.

In routine design, the search space is fully determined by the structure and range of the design parameters. The fitness function evaluates all states, and the goal state is determined by the optimum of the fitness function. The search space is defined by the chromosomes, and the search process is the artificial evolution. The high efficiency of applying GAs to parameter optimization can be explained by the intensive exploration and exploitation of the search space through selection, crossover and mutation.

Genetic algorithms have also been suggested for creative design, i.e. to generate new forms, or to combine components in a novel way, guided purely by functional performance criteria.

In creative design GA techniques are applied non-traditionally: only the tools for constructing the solution are made available to the system, not possible solutions [4]. One way of doing this is to relax constraints in order to explore more potential solutions [4],[36]. By allowing modification of the representation, we could expect the GA system to be able to solve problems beyond optimization.

In the new approach, the parameters do not represent the solution itself, but the components from which the solution is constructed. The genetic representation consists of a set of rules for the construction of a solution. These rules are mapped into a solution through so-called *embryogeny* [4]. The phenotype is then evaluated for fitness. Embryogenies are used for exploring the search space, as they allow the construction of solutions from components, contrary to genetic optimization where the parameters of a fixed system are optimized.

4. APPLICATION OF GENETIC ALGORITHMS TO DESIGN

Genetic algorithms are being applied to many areas of engineering design in mechanical engineering, electrical engineering, aerospace engineering, architecture and civil engineering, etc. It is practically impossible to give a comprehensive overview of all existing applications even for one such area. Instead, we discuss branches of engineering design in which GAs are extensively used: conceptual design, shape optimization, data fitting, reverse engineering. The common feature of all these areas is their strong geometric nature, which is also important in most engineering design problems. This also indicates that genetic algorithms can be efficient in solving problems with very different engineering content within a similar framework and by using similar procedures.

4.1 Conceptual Design

Conceptual design of a product takes place in an early stage of design and usually requires the designer to act creatively. The designer either uses novel components or combines known components in a novel way. The design parameters to be optimized are decided at this stage of the design process. There could be several ways of constructing good conceptual designs, but there is no fixed methodology to follow.

Bentley and Wakefield [5] describe a GA-based system that evolves new conceptual designs from scratch. They apply the system to designing geometries of optical prisms such that light is directed through the prisms according to the design specifications. Only the *function* of the design is pre-specified, the shape of the prism is not. For more complicated problems they start from previously found good components and optimize for their positions and for the choice of components. Cvetković and Parmee [12] use a genetic algorithm as a component in a hybrid system for conceptual airframe design. Rasheed et al. [39] have devised a GA for continuous design space search that uses new genetic operators corresponding to the structures and properties of the engineering design domains. They apply the new

GA to conceptual supersonic aircraft design. Gero and Kazakov [17] use GAs to enlarge the state space, so that the set of possible designs changes. They generalize crossover in such a way that it can move the population outside the original state space. This strategy supports creative design, and therefore can be used in the conceptual stage of the design.

4.2 Shape Optimization

One of the most important characteristics of technical objects is their shape; functionality, and production costs strongly depend on shape. Considerable efforts are continuously exerted in engineering science to find better shapes, or to optimize the shape of a component subject to engineering constraints.

Shapes can be described by a great variety of different representations; a structured set of shape parameters (scalars, vectors), or discrete representations such as pixels or voxels may be appropriate. In shape optimization, values of the shape variables have to be determined which result in an optimal value of a target parameter. This latter characterizes the object from some technical, economical or aesthetic aspect or some combination of these. In engineering applications, the relation between the target value and the shape variables — the fitness function for GAs — may be very complex, highly nonlinear, having many local extrema, and even discontinuities. While classical methods for optimization often fail under such complicated conditions, genetic algorithms may offer solutions in many practical situations. In addition to this, design constraints that are hard to handle by analytic methods (e.g. forbidden regions of the space) can be directly incorporated into a genetic optimization.

Below, we discuss typical genetic solutions to shape optimization problems. Because of the rapid expansion of genetic applications, our overview may not be complete, and other ideas may be relevant as well.

4.2.1 Parametric GA

Different types of technical problems can be characterized by different kinds of shape parameters. Sometimes we are able to characterize the technical problem just by one parameter. In [11], for example, the optimum diameter of a rotor shaft is sought by a GA, to yield a critical speed as far as possible from the operating speed.

To describe more complex shapes, more parameters are needed. Eby et al. [15] optimize by GA the shape and material placement for a flywheel to give the maximum specific energy density when having an upper bound on the maximum allowable angular velocity. An injection island genetic algorithm is applied that searches at various levels of model resolution. The shape of a disk on which turbine blades are mounted in a jet engine is optimized for minimal stress and mass by Smith et al. [42].

Deb [13] demonstrates the robustness and efficiency of GA optimization by solving mechanical component design problems in pressure vessel design, spring design, and hydrostatic thrust bearing design. The genetic representations include geometric parameters of the components, which can be discrete or continuous. The results show that even if the initial population is away from the optimal solution, the GA can find a nearby optimal solution, which is better than the results obtained by traditional methods.

A large number of publications address the optimization of truss and bridge structures, including geometrical parameters such as size and cross sections [1],[25],[38],[43],[47]. Grierson and Pak [19] present optimal design of skeletal building structures accounting for discrete sizing as well as geometrical and topological variables. Several papers evaluate different types of crossover operations, with respect to the exploration and exploitation aspects of the search process, and relative effectiveness [25],[43],[47]. New types of crossover techniques have also been suggested. The mixed crossover technique of Hasancebi and Erbatur [20] provides a tool for controlling the exploration of the design space (at the beginning of optimization), then its exploitation (in later stages of the genetic process).

In many cases, complex shapes can be optimized by optimizing the shape of characteristic curves. Whenever the ordering of the defining curve data has a close geometric relation to consecutive parts of the curve (e.g., points of interpolation, control points of a Bèzier or B-spline representation), they provide a natural transformation of curve characteristics into a sequential genetic code; segments of the curve correspond to segments of the chromosome. Accordingly, good segments of the curve correspond to good substructures of the genetic code, and the probability of disrupting a promising short code segment by crossover may be kept small.

Mäkinen et al. [29] use as shape parameters the control points of a planar Bèzier curve which describes the profile of an airfoil. The cost function which must be minimized depends on the scattered electromagnetic wave and the pressure distribution of the airfoil. Marco and Lanteri [31] use the same two-dimensional curve representation for the optimum shape design of aerodynamic configurations.

Different tasks in designing characteristic curves can also be supported by genetic algorithms. Márkus et al. [32] present a GA based method for degree reduction of Bèzier curves. The approximating curve is defined to lie in a strip around the given high degree curve, as a constraint, and to have an optimal shape characterized by its minimal curvature integral. The fitness of a curve is composed of two parts: one is the curvature integral, and the other is a penalty. The penalty depends upon the degree of constraint violation, i.e., the length of curve segment that leaves the predefined strip. Strategies are developed to use different types of crossover operations and varying weights of the two components of fitness during the genetic process.

A GA based method for the computation of curve-curve intersections is presented in [21], which is independent of the parametrization of the curve and of its topological properties (e.g. cusps, loops, zero curvature spans, etc.). Pairs of randomly selected points on each curve are represented in the chromosome. A genetic algorithm works on the generations of point pairs, and by minimizing the distance between them, it converges to one of the points of intersection.

Complex shapes can be represented by two or three-dimensional grids of control points of Bèzier, B-spline, NURBS surfaces or volumes. Following the discussion for curves, a grid structure of control points may be suitable for genetic representation. Simple genetic codes are not applicable directly to these structures, and new genetic operators must also be developed. For example, crossover operating on grids of the same size can be defined by first randomly selecting subgrids at the same position in the two individuals and then exchanging them in the offspring. A frequently occurring problem is the appearance of self-intersecting grids, which may result in self-intersecting shapes. To avoid this, intersecting grids may be locally deformed or they may be strongly penalized in the fitness calculation. Variants of the control point representations can also be used as genetic representation of shapes. For example, Wataba and Okino [46] use the free form deformation lattice (deformation of a grid of Bèzier control points) as shape chromosomes, and solve a minimum weight design problem with constraint conditions, and limitation of maximum stress.

In many practical shape optimization problems, evaluation of the fitness function for a given set of shape parameters is computationally demanding; it frequently requires the solution of nonlinear state equations in two or three dimensions. A potential problem is that GA search may exploit weaknesses of the evaluation process; e.g. finite element evaluations may give completely misleading results outside the normal design space. If this results in good — but incorrect — fitness values, the whole search will be corrupted. The lesson is that careful coordination of the search and the evaluation process is necessary for a successful optimization.

4.2.2 Cell GA

An alternative way of representing the shape of an object to be optimized is to subdivide the space into small rectangular domains (pixels in 2D, voxels in 3D), and assign them a binary full value (1) for material or empty value (0) for void (see Fig. 3.), or integers for different materials.

The cellular representation has the advantage that any shape can be represented with a certain accuracy, which can be increased by increasing the resolution. At the same time, the cellular representation can be mapped directly into a two or three-dimensional binary genetic representation, resulting in a two or three-dimensional array chromosome. By applying this representation, domain specific knowledge and geometric constraints can easily be built into the genetic process.

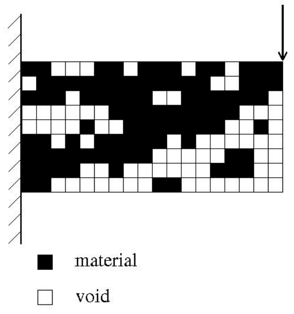

Fig. 3. The cell representation

It is a great advantage compared to a parametric GA, that the structure and topology of the object need not be fixed in advance, but develop through the optimization process. As a result, a properly designed genetic process can work in an extended search space and give better results. However, unwanted small holes may appear in the final shape, and lack of smoothness of the boundary may not be acceptable in certain engineering problems.

When a genetic algorithm for shape optimization is designed based on two or three dimensional arrays as a genetic representation, specific two or three dimensional crossover operators must be developed. One practical solution — a direct generalization of one point crossover — is first to select randomly a point in the middle of the array. Then complementary parts of the parents' chromosomes in blocks determined by the selected point are used to form the chromosomes of the children (see Fig. 4.).

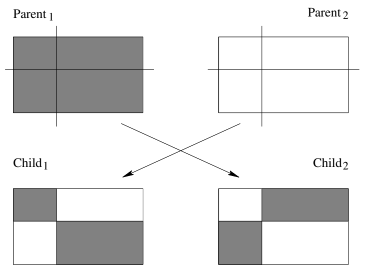

Fig. 4. The diagonal crossover for 2D arrays

For this type of genetic representation, fitness evaluation is more complicated and time consuming than in the case of a simple sequential genetic code. Typically, planar or spatial distribution of geometrical or physical quantities need to be evaluated, often by using demanding analysis programs (e.g., finite element programs).

Baron et al. [2] study the shape optimization of a beam and an annulus. In order to reduce the number of small holes, they apply a smoothness operator; the most common value for cells in a rectangular region with random position is applied to all the cells in that region. It is rather interesting that the design produced by the cell representation compares well with that produced by a parametric GA; however the cell GA requires more time to find a solution

Duda and Jakiela [14] describe a speciating genetic algorithm for shape optimization. The speciation is based on a sharing function computed from the distance in the genotype and phenotype space. The sharing function is also used for mating restriction during evolution. As a result of the genetic process, the speciated final population contains not only one solution, but good designs with different topology and shape. The method has been shown to be efficient for solving planar strain problems based on cell shape representation.

Two-dimensional cell (pixel) representation is also used for shape optimization by Kane and Schoenauer [26]. The inadequacy of one dimensional (bitstring) representation is emphasized and an evolutionary choice made among different two dimensional genetic operators.

Chapman and Jakiela [7] apply cell representation for the geometric and topological design of truss structures having a maximum stiffness-to-volume ratio. In order to obtain realistic structures a connectivity analysis is performed; all material elements in the topology which are not connected to another one through an edge are set to void. Chen and Rajan [8] consider simultaneous sizing, shape and topology optimization of structural framed systems subject to static and dynamic loads. Raich and Ghaboussi [37] develop an implicit redundant representation genetic algorithm, which allows the representation of a variable number of location independent parameters. In this way the fixed parameter limitations of usual genetic algorithms can be overcome.

A special type of cell representation called cell division model is introduced by Taura et al. [44]. The shape of a free form object is represented by dots (cells) on the surface of a sphere; the distance between a point on the surface of the free-form object and the center of the sphere is proportional to the local density of cells on the sphere. Shape evolves through a series of cell divisions, which are governed by a set of rules. These rules indirectly hold the features of the shapes, and are encoded as bit strings. A GA is applied for adaptive generation of new rules and for testing the effectiveness of existing ones. The whole mechanism resembles the early development of a living creature. In nature, cell divisions divide the fertilized egg (a single cell) into a population of smaller cells, which form the early embryo. Shape features of the evolving multicellular system are determined by rules of cell division. Although the complexity and accuracy of shapes which can be handled by the model is questionable, first experiments show that shape features can be represented, held, combined and manipulated by the method, which is useful and especially valuable in the early phase of design.

4.3 Data Fitting

Fitting of continuous curves and surfaces to discrete data points is often needed in solving a number of engineering tasks. Well established fitting methods (usually variants of the least-squares technique) are known, but they perform well only if several parameters are defined in advance. E.g., for spline fitting, the process is computationally feasible (linear) if the degree, knot distribution, and parametrization of the data points are given and fixed for the computation. In practical situations, however, these are not known to the designer. At the same time, these parameters strongly influence the quality of the fit, because they determine the structure and the basic geometric properties of the approximating function. If they are not defined properly, either the accuracy will be poor or the shape quality will not be satisfactory. The complicated and strongly nonlinear interdependence of the parameters and their influence on the fitting process is hard to keep under control, but is relatively easy to include into a GA. Successful experiments have been made to replace the complicated and unreliable process of determining the fitting parameters or a subset of them with genetic algorithms.

A genetic algorithm was developed by Márkus et al. [33] for the computation of parameter values, associated with given data points, as knots of an interpolating cubic spline curve. The curve has optimal shape in the sense that its curvature integral is minimal. An additional constraint can also be handled; namely, the curve must not pass outside a strip around a predefined curve (which is here the interpolating curve with chord length parameter).

Limaiem et al. [28] propose the application of a statistical technique called kriging for curve and surface fitting. The kriging model consists of a drift function describing the average shape and fluctuations, derived from the general covariance function. The model parameters are set by a GA to yield the minimum average error between the kriged (continuous) curve or surface and the data points.

The special case of data fitting when a polygonal approximation of an object curve is sought is fundamental in computer graphics, image processing and pattern recognition, and also plays an important role in engineering design (e.g. NC tool path calculation). The GA based technique developed by Huang and Sun [23] significantly reduces the integral square error between the polygon and the curve, and is insensitive to the selection of starting point for a closed curve and initial solutions.

Manela et al. [30] apply genetic search to finding the best combination of the number of knots and the factor that balances the interpolating and smoothness capabilities of a univariate cubic spline function fitted to noisy data points.

The difficulties in defining the fitting parameters are even more serious if not just a low dimensional shape, but a large set of high dimensional data with a large number of variables (say 20 or 30) must be fitted. Rogers [41] describes a multidimensional (multivariate) adaptive linear regression algorithm, which determines linear spline parameters using a GA. The fitness function is based on the least squared error with an additional penalty term related to the size of the model.

Classical least squares methods can be quite sensitive to noise and errors in the data, and especially to outliers (points that are not consistent with the rest of the data set). Robust statistical procedures, such as median least squares, have been developed for these cases; they are superior to least squares techniques in a strongly noisy environment. Their basic characteristic is that they are powerful, if they are combined with an efficient search algorithm. Karr et al. [27] utilize the search capability of a GA to develop a least median square curve-fitting algorithm.

4.4 Reverse Engineering

Reverse engineering is the process of creating a geometric or a CAD model on the basis of a set of data points generated by a measurement of a physically existing object [45]. It starts with registration, i.e., matching and fusing of point sets from separate views. Then a topology (usually a triangulation) is built over the unstructured point cloud, which reflects neighborhood relations. The main parts of the reconstruction are to separate subsets of data that belong to the constituting geometrical elements, and to determine types and parameters of the extracted elements. For each phase, the application of GA techniques has been investigated.

In registration, a 3D transformation is to be determined, which brings a pair of point sets describing the same shape into correspondence. GAs were used for registration of 3D images in medical applications by Jacq and Roux [24], and for free form surface matching by Brunnström and Stoddart [6]. Although accurate registration is usually based on gradient descent algorithms (e.g. iterative closest point, ICP), they work only if an initial guess reasonably close to the actual solution is known. In [6] chromosomes represent a correspondence between the two point sets. The fitness reflects the quality of the match by evaluating quantities that are invariant under translation and rotation (distances and relative orientation of normal vectors between point pairs). In Yamany et al. [48] registration is based on a grid closest point technique. The genetic code is composed of the parameters of the transformation matrix, and the genetic search minimizes the distances between associated points.

Chen and Wang [10] discuss the construction of an optimized triangulation of a point set. Although the method was developed for creating optimized STL files (a standard for rapid prototyping), it seems to be suitable for preparing surface reconstruction as well. Starting from an initial triangulation, triangles which are coplanar or nearly coplanar are removed. Then blank regions, bounded by a set of straight-line segments, are re-triangulated by a genetic algorithm. To obtain a valid triangulation, constraints are built into the initial population, the crossover and the mutation operators (a geometry constraint and a crossing constraint). Fitness is evaluated according to the smoothness of the triangulation in the blank region, the smoothness of its connection to the surroundings, and the shape of the individual triangles (equiangularity).

A GA based algorithm which is capable of extracting quadric surfaces from a set of data points is given by Chen and Liu [9] The genetic representation is a string containing the indexes of a set of data points (gene points) as elements. For the fitness evaluation, the (pseudo geometric) distances of data points to a quadric — least-squares fitted to the gene points — are evaluated. The evolution of the surface representation is achieved through the genetic improvement of the gene points.

5. CONCLUSION

Genetic algorithms have increasingly been applied in engineering in the past decade. GAs have been considered mainly as tools for optimization and parameter tuning, but they are also used for creative design.

Genetic algorithms conduct a search through the space of potential solutions to the problem. They provide a balance between two search strategies — exploration and exploitation —: they explore the search space and in the mean time exploit the good features of already found promising solutions. GAs perform an independent sampling on a population of design solutions, then select

members of the population, i.e., highly fit designs, for survival, and create new designs by crossover (combination of building blocks from different individuals) and by mutation. Crossover ensures the inheritance and dominance of valuable features, whereas mutation introduces variability. Together with the selection mechanism, they drive the artificial evolution process towards generating better and better solutions.
The strength of genetic algorithms in design can be attributed to several factors. They are flexible and can be adapted to different design problems. Realization of GA procedures results in robust and stable algorithms and computer codes. Complexity of a problem can be handled by their ability to work with many parameters simultaneously in a search space of complicated structure. In design applications it is very important that there is a built-in tendency to find global optima. GAs can also be used beyond parameter optimization, for creative design.

6. ACKNOWLEDGEMENTS

The author is grateful to Anikó Ekárt for numerous comments and suggestions.

7. REFERENCES

[1] Adeli, H. and Cheng, N.-T., Concurrent genetic algorithms for optimization of large structures, *Journal of Aerospace Engineering*, Vol. 7, No. 3, 1994, pp 276–296.

[2] Baron, P., Fisher, R., Tuson, A., Mill, F. and Sherlock, A., A voxel-based representation for evolutionary shape optimization, *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 13., 1999, pp 145–156.

[3] Bentley, P., An introduction to evolutionary design by computers, *In Evolutionary Design by Computers,* Morgan Kaufmann, 1999.

[4] Bentley, P. J. and Corne, D. W., An introduction to creative evolutionary systems, *In P. J. Bentley and D. W. Corne, editors, Creative Evolutionary Systems,* Academic Press, 2002, pp 1–75.

[5] Bentley, P. J. and Wakefield, J. P., Conceptual evolutionary design by a genetic algorithm, *Engineering Design and Automation*, Vol. 3, No. 2, 1997, pp 119–131.

[6] Brunnström, K. and Stoddart, A. J., Genetic algorithms for free-form surface matching, *In Proceedings of Int. Conf. on Pattern Recognition*, Vol. IV, 1996, pp 689–693.

[7] Chapman, C. D. and Jakiela, M. J., Genetic algorithm-based structural topology design with compliance and topology simplification considerations, *Journal of Mechanical Design*, 118., 1996, pp 89–98.

[8] Chen, S. Y. and Rajan, S. D., A robust genetic algorithm for structural optimization, *Structural Engineering and Mechanics*, Vol. 10, No. 4, 2000, pp 313–336.

[9] Chen, Y. H. and Liu, C. Y., Quadric surface extraction using genetic algorithms, *Computer-Aided Design*, Vol. 31, No. 2, 1999, pp 101–110.

[10] Chen, Y. H. and Wang, Y. Z., Genetic algorithms for optimized retriangulation in the context of reverse engineering, *Computer-Aided Design*, Vol. 31, No. 4, 1999, pp 261–271.

[11] Choi, B. G. and Tang, B. S., Optimum shape design of rotor shafts using genetic algorithm, *Journal of Vibration and Control*, Vol. 6., 2000, pp 207–222.

[12] Cvetković, D. and Parmee, I. C., Genetic algorithms based systems for conceptual engineering design, *In International Conference on Engineering Design*, 1999.

[13] Deb, K., GeneAS: A robust optimal design technique for mechanical component design, *In Evolutionary Algorithms in Engineering applications,* Springer, 1997, pp 497–514.

[14] Duda, J. W. and Jakiela, M., Generation and classification of structural topologies with genetic algorithm speciation, *Journal of Mechanical Design*, Vol. 119, 1997, pp 127–131.

[15] Eby, D., Averill, R. C., Punch, W. F. and Goodman, E. D., Optimal design of flywheels using an injection island genetic algorithm, *Artificial Intelligence for Engn. Des., Analysis and Manufacturing*, 99., 1999, pp 327–339.

[16] Gero, J., Design prototypes: a knowledge representation schema for design, *AI Magazine*, Vol. 11, No. 4, 1990, pp 26–36.

[17] Gero, J. and Kazakov, V., Adaptive enlargement of state spaces in evolutionary designing, *Artificial Intelligence for Engn. Des., Analysis and Manufacturing*, 14., 2000, pp 31–38.

[18] Goldberg, D. E., Genetic Algorithms in Search, Optimization, and Machine Learning, *Addison-Wesley*, 1989.

[19] Grierson, D. E. and Pak, W. H., Optimal sizing, geometrical and topological design using a genetic algorithm, *Structural Optimization*, Vol. 6, No. 3, 1993, pp 151–159.

[20] Hasancebi, O. and Erbatur, F., Evaluation of crossover techniques in genetic algorithms based optimum structural design, *Computers & Structures*, Vol. 78, No. 1–3, 2000, pp 435–448.

[21] Hawat, R. N. and Piegl, L. A., Genetic algorithm approach to curve-curve intersection, *Math. Engn. in Ind.,* Vol. 7, No. 2, 1998, pp 269–282.

[22] Holland, J. H., Adaptation in Natural and Artificial Systems, *Ann Arbor: The University of Michigan Press*, 1975.

[23] Huang, S. C. and Sun, Y. N., Polygonal approximation using genetic algorithms, *Pattern Recognition*, 32., 1999, pp 1409–1420.

[24] Jacq, J. J. and Roux, C., Registration of 3D images by genetic optimization, *Pattern Recognition*, Vol. 16, No. 8, 1995, pp 823–856.

[25] Jenkins, W. M., On the application of natural algorithms to structural design optimization, *Engn. Struct.*, Vol. 19, No. 4, 1997, pp 302–308.

[26] Kane, C. and Schoenauer, M., Genetic algorithms for two dimensional shape optimization, *In Artificial Evolution*, 1996, pp 355–369.

[27] Karr, C. L., Weck, B., Massart, D. L. and Vankeerberghen, P., Least median squares curve fitting using a genetic algorithm, *Engineering Applications of Artificial Intelligence*, Vol. 8, No. 2, 1995, pp 177–189.

[28] Limaiem, A., Nassef, A. and El-Maraghy, H. A., Data fitting using dual kriging and genetic algorithms, *Annals of the CIRP*, Vol. 45, No. 1, 1996, pp 129–134.

[29] Mäkinen, R., Periaux, J. and Toivanen, J., Shape design optimization in 2D aerodynamics using genetic algorithms on parallel computers, *In Parallel Computational Fluid Dynamics: Implementations and Results Using Parallel Computers, Proceedings of the Parallel CFD'95 conference*, 1996, pp 395–402.

[30] Manela, M., Thornhill, N. and Campbell, J. A., Fitting spline functions to noisy data using a genetic algorithm, *In Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp 549–553.

[31] Marco, N. and Lanteri, S., A two level parallelization strategy for genetic algorithms applied to optimum shape design, *Parallel Computing*, Vol. 26, No. 4, 2000, pp 377–397.

[32] Márkus, A., Renner, G. and Váncza, J., Genetic algorithms in free form curve design, *In Mathematical Methods for Curves and Surfaces*, 1995, pp 343–354.

[33] Márkus, A., Renner, G. and Váncza, J., Spline interpolation with genetic algorithms, *In Shape Modeling and Applications*, 1997, pp 47–54.

[34] Michalewicz, Z., Genetic Algorithms + Data Structures = Evolution Programs, *Springer-Verlag*, 1992.

[35] Michalewicz, Z., Dasgupta, D., Riche, R. G. L. and Schoenauer, M., Evolutionary algorithms for constrained engineering problems, *Computers and Industrial Engineering*, Vol. 30, No. 4, 1996, pp 851–870.

[36] Parmee, I., Exploring the design potential of evolutionary search, exploration and optimization, *In P. J. Bentley, editor, Evolutionary Design by Computers*, Morgan Kaufmann, 1999, pp 119–143.

[37] Raich, A. M. and Ghaboussi, J., Evolving structural design solutions using an implicit redundant genetic algorithm, *Structural and Multidisciplinary Optimization*, Vol. 20, No. 3., 2000, pp 222–231.

[38] Rajeev, S. and Krishnamoorty, C. S., Discrete optimization of structures using genetic algorithms, *Journal of Structural Engineering*, Vol. 118, No.5, 1992, pp 1233–1250.

[39] Rasheed, K., Hirsh, H. and Gelsey, A., A genetic algorithm for continuous design space search, *Artificial Intelligence in Engineering*, 11., 1997, pp 295–305.

[40] Renner, G. and Ekárt, A., Genetic algorithms in computer-aided design, *Computer-Aided Design*, Vol. 35, No. 8, 2003, pp 709-726.

[41] Rogers, D., G-splines: a hybrid of Friedman's multivariate adaptive regression splines (MARS) algorithm with Holland's genetic algorithm, *In Proceedings of the Fourth International Conference on Genetic Algorithms*, 1991, pp 384–391.

[42] Smith, R., Warrington, S. and Mill, F., Shape representation for optimization, *In Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1995, pp 112–117.

[43] Syswerda, G., Uniform crossover in genetic algorithms, *In Proceedings of the Third International Conference on Genetic Algorithms*, 1989, pp 2–9.

[44] Taura, T., Nagasaka, I. and Yamagishi, A., Application of evolutionary programming to shape design, *Computer-Aided Design*, Vol. 30, No. 1, 1998, pp 29–35.

[45] Várady, T., Martin, R. and Cox, J., Reverse engineering of geometric models - an introduction, *Computer-Aided Design*, Vol. 29, No. 4, 1997, pp 255–268.

[46] Wataba, H. and Okino, N., A study on genetic shape design, *In Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp 445–450.

[47] Wu, S.-J. and Chow, P.-T., Steady-state genetic algorithms for discrete optimization of trusses, *Computers & Structures*, Vol. 56, No. 6, 1995, pp 979–991.

[48] Yamany, S M., Ahmed, M. N. and Farag, A. A., A new genetic-based technique for matching 3D curves and surfaces, *Pattern Recognition*, 32., 1999, pp 1817–1820.