Evolutionary Structural Optimization in Shell Design

Sassone, Mario; Pugnale, Alberto

Published in:
Advanced Numerical Analysis of Shell-like Structures

Publication date:
2007

Document Version
Accepteret manuscript, peer-review version

Evolutionary Structural Optimization in Shells Design

Mario SASSONE[1], **Alberto PUGNALE**[2]

[1]*Department of Structural and Geotechnical Engineering*
Politecnico di Torino
Viale Mattioli 39, I-10125 Torino, Italy

E-mail: mario.sassone@polito.it

[2]*Department of Architectural and Industrial Design*
Politecnico di Torino
Viale Mattioli 39, I-10125 Torino, Italy

E-mail: alberto.pugnale@polito.it

Abstract. In the paper the application of a GA evolutionary algorithm to the structural optimisation of free-form architectural shell structures, as large concrete roofs, is shown. The attention is focused on the need of different computational tools and on the problems involved in multidisciplinary design, as the communication and information exchange. The procedure herein described can be applied to more general problems, involving not only the structural performance, but even functional and constructive requirements. The proposed implementation of the algorithm is efficient, robust and particularly capable to avoid local minimum convergence.

1 Introduction

The interest architects show for free-form structures has had a unexpected growth in last years, as many new buildings testify, due mainly to their plastic capabilities and to their similarity with organic shapes (Figure 1).

From the structural point of view free-form shells belong to the typology of shape-resistant structures, in which the mechanical behavior is strictly related to the global structural spatial configuration, rather than to the properties of individual members (as for instance in the case of truss structures).

Figure 1: Example of free-form architecture

As a consequence, the search of the optimal shape requires a global, we could say multidisciplinary, optimization process in which the structural performance, as well as the architectural shape, the lighting control, the acoustic behavior, and all the architectural problems involved in the project can find a suitable answer. During the whole design process, starting from the early concept until the final construction in the building site, architects and engineers need to work interactively on the project.

The organization of the design process involves architects, engineers and other different professional figures into a continuous share of their specific skills, frequently generating problems in communication and information exchange, due to the specific education and professional approach to problems.

Nowadays architects usually work with the aid of three-dimensional CAD modelers, and with design tools featured by powerful and user-friendly graphic interfaces, so that creation and manipulation of architectural shapes becomes easy and intriguing. On the other hand, for structural engineers the global shape is just the first step of a more complex process of analysis of the structural behavior, usually performed with the aid of computational tools as finite elements solver.

Same things can be said for other specific problems, like acoustics or construction problems, in which the architectural shape needs to be analyzed, decomposed and transformed in something real. Thus, the trial from the architectural concept to the evaluation of the structural behavior and to construction is not a continuous, say ‘fluid’ process, but it involves many steps, interruptions and feedback, characterized by a reinterpretation, a kind of ‘translation’ of the information produced at each stage.

Starting from this conceptual frame, the paper describes a process of structural optimization of shell structures based on the application of an evolutionary algorithm. The optimal shape is obtained iteratively as a converging sequence of intermediate solutions. The attention is focused both on the efficiency of the algorithm and on the problems related to the communications and information exchange required at each iteration, pointing out the role of the architect in guiding the iterative process and in checking the solution from the formal and architectural point of view.

2 NURBS Representation of Free Form Shell Structures

In free-form structures the global architectural or structural shape can not generally described in terms of simple geometrical shapes. If one way to overcome this problem is represented by substituting the desired surface with a discrete geometrical triangular mesh, a more powerful and refined design tool consists in the approximation of the desired surface in terms of Non Uniform Rational B-Splines (NURBS). This mathematical formulation of surfaces has his antecedents in the Rational and Non-Rational B-Splines and in the Bézier curves and surfaces, that represented an interesting examples of intersection between different disciplines, if we consider that the engineering problem of tracing suitable curves for automotive design found his solution into an abstract topic of analysis, the so called Bernstein basis polynomials.

The NURBS belong to the category of parametric curves and surfaces, and they become the standard for describing and modelling free-form shapes in computer aided design (CAD) and in computer graphics. The surfaces represented by NURBS are mainly defined by a control polyhedron (Figure 2), which vertices are called control points, but there are also other parameters affecting the surface shape in its mathematical definition: the basis functions, the degree, and the knots matrix.

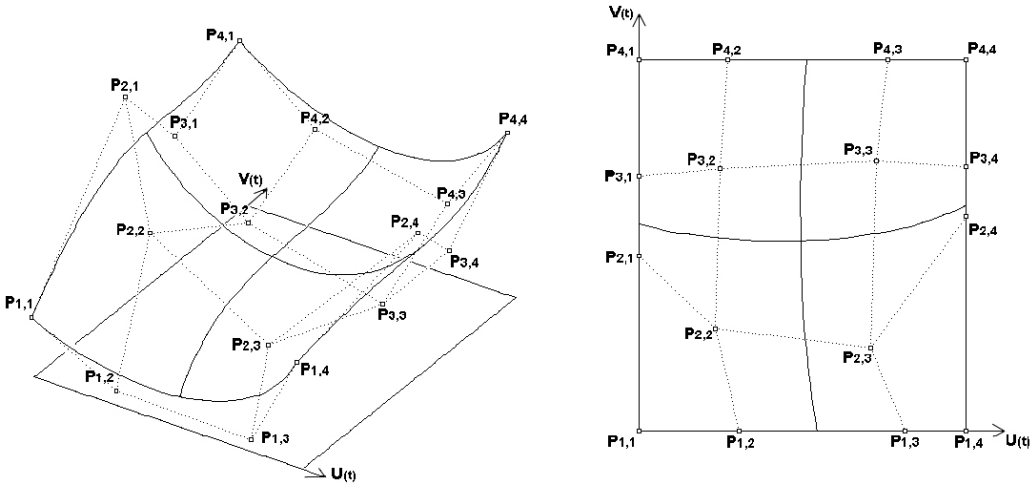


Figure 2: Domain and control points of a surface NURBS representation.

This method of mathematically describing shapes allows an excellent local and global control of free-form shapes, in all design problems where both aesthetic and functional requirement are indispensable and the resulting curves or surfaces need to agree with previously defined geometric boundary conditions.

What is of particular interest in this geometrical tool is the fact that a complex shape, with variable Gaussian curvature, imposed borders, even self-intersections and specific topological properties can be parametrically described by a finite set of real numbers. This set of numbers contains all the information necessary to completely describe the surface, even tough in a kind of coded form: the local and global geometrical properties of the surfaces can not be directly inferred from the parameters. From this point of view a parametric representation of a geometric entity is already analogous to the genotype of

a living being: more a rule from which the corresponding phenotype can be build rather than a simple synthetic description. As it will shown in the sequel, the translation from the shape to a parametrical representation (coding) and the translation from the code back to the shape (decoding), are basic operations in genetic algorithm theory.

3 GA Heuristic Optimisation

After almost thirty year they have been described for the first time [HOLLAND 1975], and after a large diffusion of theoretical studies and applications, the status of Genetic Algorithms (GA) in the scientific research still does not seem to be well defined. As many other mathematical models, starting from neural networks, going through fuzzy logic, simulated annealing techniques, and arriving to particle swarm optimization, ant colony optimization and the 'shuffled frog leaping' algorithm, all these approaches to the solution of mathematical problems were born as a tentative to explain, or at least to imitate, some natural phenomena related to the world of life. The history of neural networks is quite interesting from this point of view: when they were described for the first time [RUMELHART AND MC LELLAND, 1988] the contest was cognitive psychology and the interest to the parallel microstructure of thought was in opposition with a more synthetic "higher level" approach [MINSKY, 1985]. Only a few years after the enthusiasm fade away, when scientists understood that having a local model of neuronal behavior is quite different from being able to simulate the parallel interactions between many billions of neurons.

In scientific literature we find papers on genetic algorithms under different branches: in Operation Research (OP) they are classified as a heuristic or semi-heuristic technique for combinatory optimization; in biology they are considered as a suitable model for living beings natural evolution. In architecture and structural engineering we are mainly concerned with their capability to explore a dominion of possible configurations looking for the best one, form a specific point of view defined in the so called fitness function.

4 The Genetic Algorithm

The algorithm adopted in this work is based on a binary 8-bits coding of the control point coordinates of a NURBS surface. If the surface has ***n*** control points, his coded representation, the 'chromosome' is a binary vector of ***nx3x8*** bits, being ***3*** the spatial components of the control point position. A 8-bit representation of a real number, included into a specific interval, means that the interval is discretised into 256 discrete values. Each surface, assumed as the geometry of a free-form structure, is associated with a fitness value, representing the structural performance of the surface itself. The fitness can be evaluated by means of a structural analysis: depending on the kind of problem a local (max displacement, max stress) or integral (elastic potential energy, or some other measure of the global behaviour) value can be chosen.

If a coding criterion has been defined as well as a suitable fitness function, the genetic algorithm can be implemented. It is important to focus that genetic algorithms do not require any restrictions on the kind of problem they have to solve: the algorithms do not know what is the entity represented by the code, the ‘signification’ of the code, as well as they do not know what is the meaning of the fitness function. All that is required by the algorithm is a correspondence between each chromosome and its fitness value.

Initially a set of ***m*** geometrical shapes is randomly generated, as the starting point of the optimization process. Each shape is called 'individual' and the whole set is called 'population'. Starting from the initial random population the iterative procedure is made of these steps:

- each individual is coded, it means that from the geometric ***nx3*** coordinates representing the control point of the NURBS description of the shape, the ***nx3x8*** binary vector is build;
- the fitness off each individual is calculated: a finite element model of the structure represented by each shape is build automatically, the analysis is performed and the fitness value is calculated following the adopted criterion;
- staring from the population and the fitness values at step ***i***, a new set of shapes, it means a new population, is generated for step ***i+1***.

The procedure of creation of the new set of shapes at each step of the iterative process is the core of the algorithm: the classical techniques are based on the crossover procedure, the random mutation and the elitism method, but many combinations or variations of these criteria can be adopted, in order to improve the convergence of the algorithm.

It is important to know that the evolutionary optimization strategy is particularly suitable when the function to be maximised is not regular and it is necessary to avoid that the solution converges to a local maximum. This problem is very important in optimisation theory and the largest part of classical algorithms, as the ones based on gradient method, can not avoid this risk. At each iteration the evolutionary strategy selects the best solutions, recombines them together by means of a binary coding process (crossover and mutation), and creates a new generation of individuals. The whole domain of solutions is explored (at each iteration) and the performance of each generation of shapes improves, approaching to the optimal one.

5 Design Automation

The process of optimisation above described requires a set of computational tools: first the NURBS, representing the architectural end structural shape, need to be managed inside a graphic mathematical environment, as the one offered by a CAD software. The commercial application adopted in this work is the McNeel Rhinoceros™ three-dimensional graphic modeller, largely diffused both in the architects and designers communities. Secondly a Finite Element structural solver is necessary for the optimization process: the Ansys™ commercial application has been used. Finally a generic programming environment in which develop the genetic algorithm is required. The main task of this tool is to elaborate information and to exchange it with the commercial applications: the real time communication between different software applications is the most important problem to be solved. In order to simplify the problem, the built-in Visual Basic interpreter Rhinoscript™, which is included in the Rhinoceros™ CAD application, has been chosen. The interface between the visual basic code and the graphic database of the CAD is managed by an object oriented language included in the interpreter itself and integrated with the visual basic language. So that half of the problem is solved. The communication with Ansys™ is made possible by

using text files exchanging, even though that is not the most efficient way, and by a procedure of launching and synchronization that allow the VB code to call and to handle the FEM solver.

Operating in this way the core of the optimisation process is the VB code implemented in Rhinoscript language, that only deals with the parametric representation of shapes and generates the geometrical description of finite element models (the structural mesh and al the FEM input data), meanwhile the CAD operates with the NURBS, generating them from parameters and allowing to extract from them all the required information, and the FEM solver just receive at each iteration and for each individual in the population an input data file and a call from the code, producing the correspondent output result file.

At each iteration and for each individual all the information are available: the shapes generated by the algorithm are of interest for the architect, who can follow the process from his own aesthetic point of view; the engineer is concerned with structural behaviour and can check the solutions iteratively generated trying to understand the influence of the global shape on displacements, stress distributions, stability.

6 Tests, Applications and Results

The algorithm efficiency depends mainly on three factors: the fitness function, the recombining strategy of the individuals (elitism, roulette wheel, crossover) and the mutations ratio. No general calibration criteria are allowable for these parameters: but in each single case is opportune to test some times the whole algorithm and analyze its results to modify at best its configuration values. A bad definitions of them could cause a ‘genetic drift’, that means all the individuals have the same genetic features and it is not possible an evolution yet, a long time to find the optimal solution, a result that is not the optimal one but a little lower (stopped by a local maximum). The numbers of individuals, the number of generations and the encoding method are other important parameters.

After some launches of the genetic algorithm based on this first test configuration, it is possible to evaluate advantages and disadvantages of these parameters for this specific case. The fitness function is the most important parameter to be defined for a good evaluation and an efficient selection of the individuals. In this first test, the typology of the evaluated shape belongs to the form resistant structures, therefore it was chosen as fitness value the inverse of the maximum displacement among all the structural model nodes. The algorithm builds an input file for the finite elements analysis that meshes all the surface and defines an array of shell elements. Thus, the fitness function could be better represented in future tests by the elastic potential energy.

The random crossover operator, for recombining among them the best half of individuals, is not sufficient in order to obtain the optimal solution. In a quite number of generation, always between the fifth and the tenth, the procedure fall in a genetic drift. The crossover operator is indispensable to improve the medium performance of each generation, but it is difficult that it is able to increase the maximum fitness value at each step.

The efficiency of the computing time is not important for this architectonic application, on the contrary it could be interesting to follow the evolutionary procedure that generates a wider range of shapes, to select then visually the preferred one.

Keeping the best shape of each generation to build the next one is a good technique in order to improve the fitness value during the process without any regression course. In fact, in the worst case the maximum fitness value will remain the same of the previous iteration.

A calibrated numbers of genetic mutations, that means a local variation of the genetic structure of an individual, is necessary to avoid to fall in local maximums (genetic drift). It is important that they are in a little percentage of the population, otherwise the individuals will probably get worse than their parents. In nature, for example, the genetic evolution is done by little steps, greater ones often found the death.

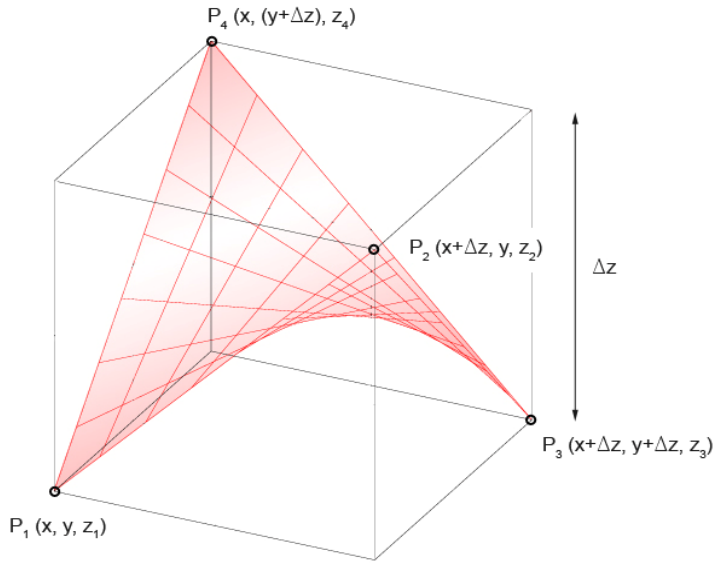
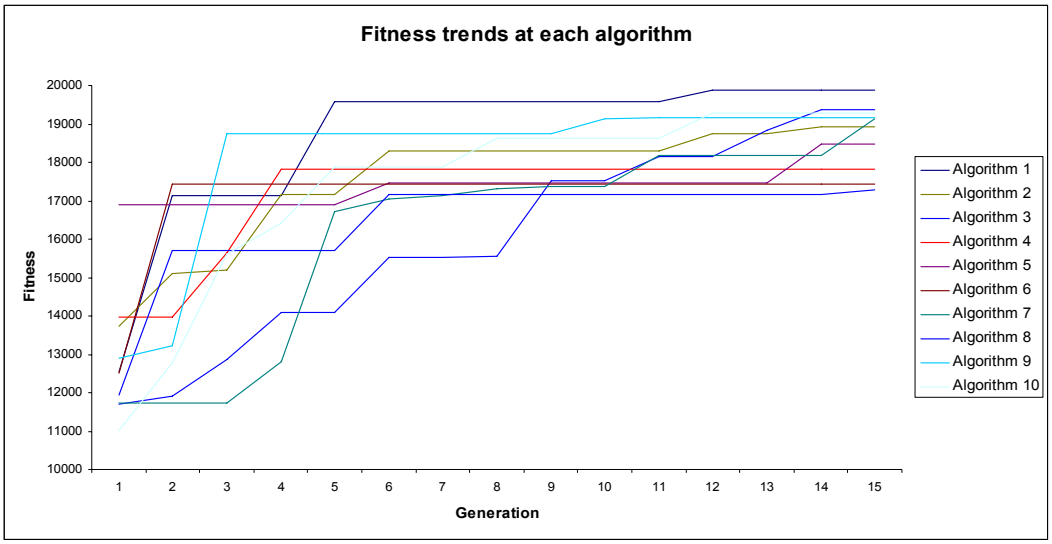


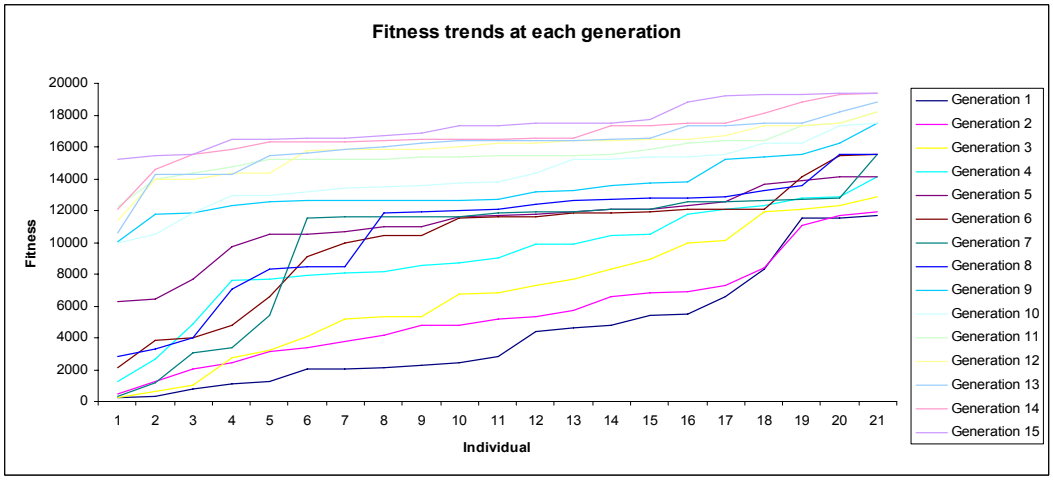
Figure 3: Domain and control points of a surface NURBS representation.

The secondary parameters, as the numbers of individuals and generations and the encoding method, have been found as less influent after many launches of different configurations of the same algorithm. The number of individuals have been established in a range of ten to twenty shapes. The number of generations is always not less than ten, however over this value there is not any significant improve of the maximum fitness in this case. The encoding method is an eight bit binary codification, that is good for every numeric variable case.

It is very important to launch many times the whole algorithm, because the first random set of individuals can evolve more or less just for the different structural behaviour of the starting phenotypes.

The algorithm searches an optimum solution by exploring a domain of potential configurations. From the computational point of view, the choice of the domain, in particular of its wideness, affects the direction of the solution research and the speed of convergence. From the architectural point of view the domain of the design variables represents the field of shapes, the range of spatial configurations from which the final solution will be elaborated. Therefore, the designer must known what are the implications of his choices, when he defines a specific domain .





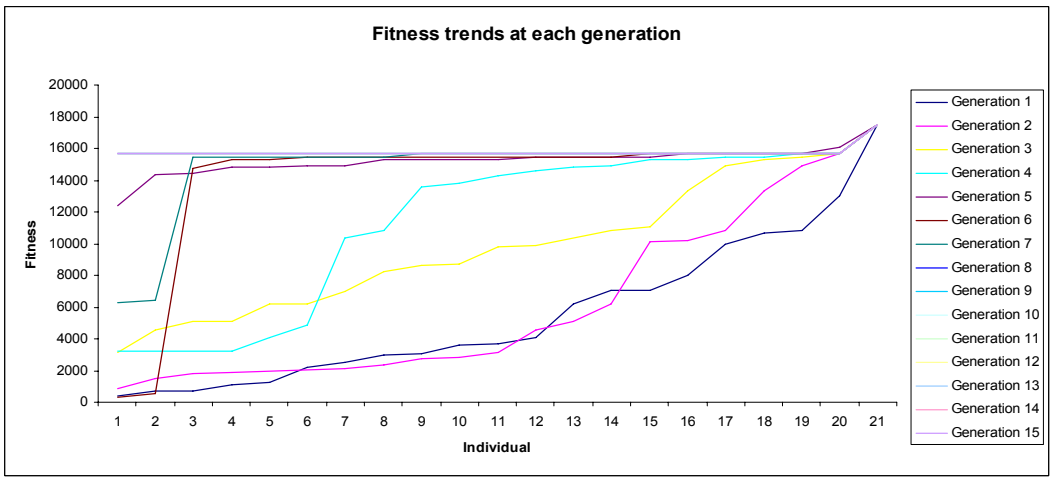


Figure 4 : Algorithm efficiency: best individual for each generation, crossover mutations, only crossover.

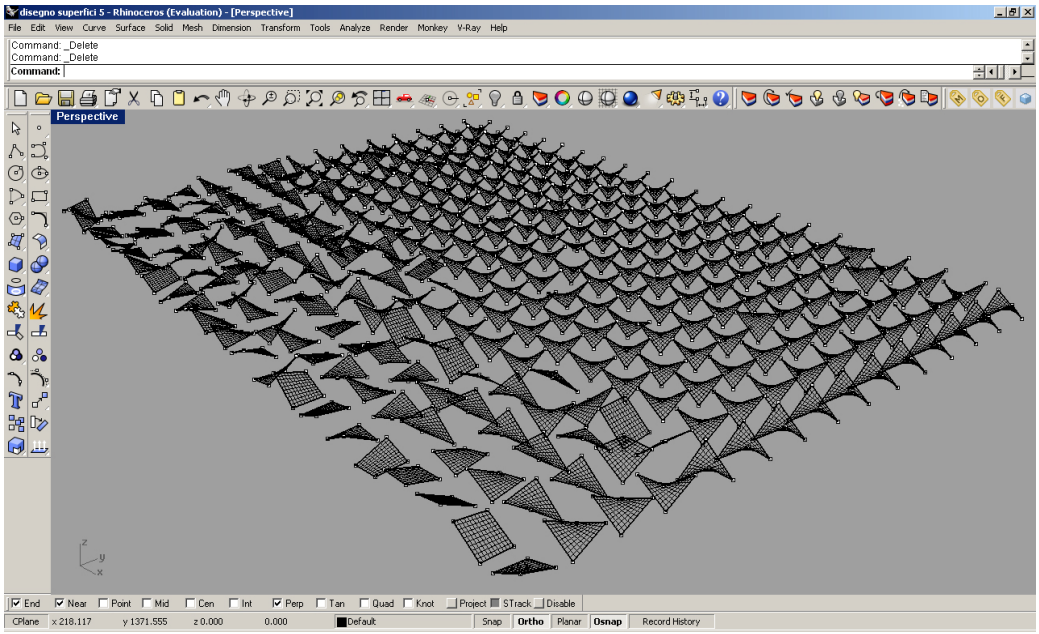


Figure 5: Rhinoceros™ screenshot: the shapes generated by the algorithm (each row is a generation of 21 individuals).

Two different approaches can be adopted. In the first, a small domain of the design variables is assumed : the optimal solution is then researched into a small range of potential configurations, all close to a reference configuration initially defined. From the architectural point of view the reference shape corresponds to the conceptual idea that has to be preserved in the design process . The genetic algorithm can improve the structural behaviour of the initial spatial configuration, by applying little variations on the coordinates of the original control points, but the optimising process is limited by the reduced extension of the domain. This approach can be called a form-improving process. A large part of the real design situations are like this, because the architects, or in general the designers, start the design process with by defining and drawing a conceptual reference shape. As an example, one can think to the Toyo Ito's crematorium or the Arata Isozaki's cultural centre in Japan, in order to cite only some recent important projects.

Following the second approach, a larger range of values is chosen, so that the design domain is not representative of a previously defined shape or configuration to be improved, but it only constitutes a wide field of potential solution for the algorithm. Thus, a true form-finding process is performed and unexpected new shapes can be obtained. The designer has to be careful in choosing the degrees of freedom of the design process, because in some cases the optimal solutions found by the algorithm are not interesting or appropriate. A situation where a larger domain can work well is in the design of the roof for a concert hall: the ground plan and all the perimeter walls are defined, so that unexpected ceiling shapes with optimal structural and acoustic behaviour are well accepted. This second approach can be considered as a form-finding process.

The sequence of the algorithm operations starts from the parametric definition of the domain of solutions, by means of a set of design variables. . At the beginning, the genetic algorithm has been tested with a very simple shape, of which it is well known the optimal configuration. It is a square defined by four vertices with the x and y coordinates fixed, and the z coordinate free to change on a small domain. In this case, the best form resistant shape is known and corresponds to the saddleback configuration.

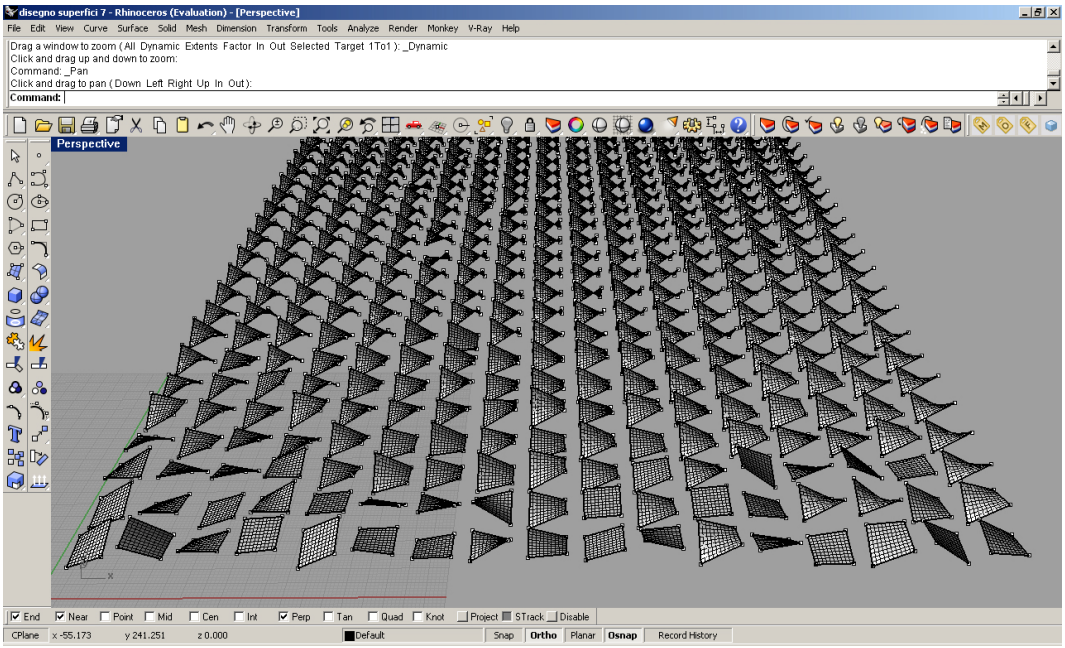


Figure 6: Rhinoceros™ screenshot: the shapes generated by the algorithm (each row is a generation of 17 individuals).

The optimization procedure searches the optimal solution inside the four-dimensional domain, of all the possible configurations. It starts from a set of random variations based on the defined domain and on the reference architectural shape. For this test, the domain is a range of z coordinates and the architectural reference is a shape, a cover, inscribed in a box.

This optimization method needs the synchronization of Rhinoceros and Ansys, for the coordinated definition and evaluation of every generated shape. Rhinoceros takes care of the form description, then the algorithm is able to read it and write an input file for Ansys, in order to launch the structural analysis. The mathematical definition of a NURBS surface is here translated from the Rhinoceros language to the correct syntax of the Ansys one, for the automatic reading and reconstruction of the model inside this software. Therefore the passage between the modelling and the evaluation do not require any user participation, and it is possible to repeat it for all the shapes that will come generated by the algorithm. At the end of this step, the code have only more to read the required result value from the output log file produced by the finite elements analysis, in order to assign a fitness coherent with the structural behaviour.

7 Conclusions

A general procedure for form-improvement and form-finding of architectural shell-based shapes is presented in this paper. The procedure is based on the application of a genetic algorithm as the optimizing algorithm and the performance required to the shell structures is the best mechanical behavior. In order to handle the optimization process three different software applications need to be linked: the genetic algorithm formulation, implemented by the authors in the VB programming language; a geometric NURBS based CAD environment, necessary to describe the structures from the geometric point of view; and finally a FEM solver, in order to evaluate the mechanical behavior of the solutions. The procedure has been tested on a simple case of study, revealing the robustness and the reliability of the algorithm. The differences between the form-improving and the form-finding optimization approaches have been described with particular attention to the implications on the architectural design process.

It is important to underline that this approach, can be applied in a quite general way to many architectural and engineering problems, enhancing the control and the check of the project during the design process, provided that the problems are well formulated in terms of the solution domain and the required performance.

References

[1] Cerchi L., Invece del corpo una nuvola. *Casabella*, **752**: 30-37, 2007.

[2] Ciammaichella M., *Architettura in NURBS: il disegno digitale della deformazione*. Testo&Immagine, Torino, 2002.

[3] Dawkins R., *L'orologiaio cieco: Creazione o evoluzione?*. Mondadori, Milano, 2003 (1986).

[4] Floreano D., Mattiussi C., *Manuale sulle reti neurali*. Il Mulino, Bologna, 2002 (1996).

[5] Maffei A., Razionalità delle forme organiche. *Casabella*, **752**: 40-45, 2007.

[6] Minsky M., *La società della mente*. Adeplhi, Milano, 1989 (1987).

[7] Piegl L., Tiller W., *The NURBS book, 2nd edition*. Springer, Berlin, 1997 (1966).

[8] Rogers D. F., *An introduction to NURBS: with historical perspective*. Academic Press, San Diego, San Francisco, 2001.

[9] Rumelhart D., McLelland J., *Parallel Distributed Processing. Explorations in the Microstructure of Cognition*. The MIT Press, Cambridge, 1986.

[10] Sasaki M., *Flux Structure*. TOTO, Tokyo, 2005.