Politecnico di Torino

Politecnico di Torino

Master's degree in Civil Engineering
A.y. 2021/2022
Graduation session March 2022

Combination of evolutionary algorithms and form-finding methods for the structural and shape optimization of a roof structure

Supervisors:
Prof. A. Manuello Bertetto
Prof. G. C. Marano
Ing. J. Melchiorre

Candidate:
F. Deiana

Summary

1. Introduction

The objective of this thesis is the coding of an algorithm to define and optimize a roof structure to reduce the employed quantity of material and maximize the elements that are equal to each other.

The optimization is a natural process; indeed, every form of life tries to reduce the energy needed to do survive, and many natural shapes are made to optimize the material used, from the shell to the hive shapes.

Also, humans try to optimize their activity, but this typology of optimization is not scientifically defined but more impulsive. One of the first examples of scientific optimization is the assembly line invented by H. Ford; where each step is scrupulously defined to reduce the time needed for the construction of a vehicle. In the building field, the application of the assembly line is really difficult, overall because the main part of the building is a unique, prototype, and not produced in series; therefore the optimization should be done on the quantity of material. The first example of this typology of optimization was done by A.Gaudì at the start of the 20th century, with the help of some physical models, in a process called form-finding; but the goal of his experiment is the design of the structure with really complex shape and not the material reduction. The merit of Gaudì's experiment is which one to invent the form-finding, this procedure allows to define a structural shape with really low bending moments and so to reduce the material quantity. This method to define the structural shape was abandoned for fifty years until designers like F. Otto and H. Isler reuse Gaudì's method to design their structures. These new structures are light and have a good material use, like Isler's concrete vaults with a thickness of only 4,0 cm. these can be the first example of structural optimization, but this practice spreads only at the start of the 21st century thanks to the huge use of computers in the structural design, and it melts with the parametric design used by Zhah Hadid.

This new technic of design is based on an algorithm to define the virtual model of the structure which has different parameters to modify the characteristics of the model. It exploits the big computational capacity of the computer to generate different models, and the computer graphic to see the results. These algorithms can be inside a design software or be coded in the programming software.

Therefore, in this thesis a structural optimization algorithm was coded, to define different roof structures with different characteristics; this algorithm was coded in *Grasshopper,* a visual coding software inside *Rhinoceros* that allows the parametric definition and the analysis of a structure. The first part of the algorithm is the definition of the shape, thanks to a virtual form-finding after the shape is transformed in a grid-shell and its parameter evaluated; the last part is the structural analysis.

To reduce the time needed to research the best structure, this algorithm is commanded by an evolutionary algorithm (EA), that modifies the parameters to reach an objective. Also, these algorithms are quite new technology, the first example was in the seventies, and they exploit the natural evolution theory to select the best parameter.

The application of the evolutionary algorithms to structural optimization allows a huge reduction of the time and computational effort to define a structure; its parameters are collected in a number,

the objective, and the EA modifies the algorithm's parameters to maximize or minimize the objective.

1.1 Structure of the Thesis

The thesis is composed of four chapters, that follow the iter of the thesis.

The first chapter deals with the theory behind structural optimization, from the most generic mathematical optimization to the different optimization processes, focusing on the different form-finding processes.

The second chapter is about the software used in the thesis, starting with *Rhinoceros* and *Grasshopper,* and explains the plug-ins used in this thesis; their inputs, and their outputs. A focus is done on *Galapagos* to explain how an evolutionary algorithm works.

The third chapter speaks about the coding of the algorithm and how it works. Each paragraph is about a different step and explains the flow of the code; to simplify the comprehension a flow-chart is shown at the end of each paragraph.

The fourth and last chapter deals with the analyses made with the algorithm and the different structures defined with the different objectives; all the analyses process is explained with the reason that takes to an analysis. At the end of the chapter, all the structures are compared and their pros and cons are exposed.

2. Concepts

2.1 Parametric Design

Parametric design, or computational design, is a design process based on the relation, like algorithm, between some data, like numbers, to define the object's model.

These relations are made between some design parameters, like geometric ones, and defined by some functions, like mathematical ones. This allows modifying the project, its shape, only with the modification of a parameter; while in the "classical" design, if a parameter changes, the project should be made from zero. In other words, in the "classical" design the model is defined by the designer, while in the parametric design the model is defined by software, with the relationship and the dates given in input by the designer. Therefore, the designer always has the "mind" behind the project, but his works become more conceptual because the computer generates the model.

Thanks to the power of this method, parametric design is used in many design fields, from manufactory to architecture, because allows the generation of different possible models in a short time. Different models mean a bigger possibility to find the best structures.

2.1.1 The Concept of Parametric Design

Every design should respect some parameters like the maximum height, the maximum surface, or the ratio between the windows surfaces and the room surfaces; the idea behind the parametric design is to use these parameters to define the model. Therefore, the model is defined by an algorithm that generates a structure with the given parameters; so, the algorithm's core is the parametric design.

The algorithm is based on a mathematical relationship between the elements of the model that link the parameters with the results; furthermore, the mathematical relation allows to define of a complex shape, like in the designs of Zaha Hadid, which are difficult to define with a "classical" CAD. Therefore, parametric design explicitly links architecture and mathematics; allows the use of the unction to have better use of material. An example is the use of the catenary, which is defined with the hyperbolic cosine, for design arch without bending moments. Thanks to the possibility of using algorithms, in parametric design, mathematical optimization, and evolutionary algorithms are also used, which will be explained in the following chapters.

The possibility to modify the parameters modifies the model fast because is done by the software, increasing productivity. For example, we can image the columns of a building; if in the "classical" design I should change the shape of them, I should re-drawing all the columns; while, in parametric design, I change the parameters of the columns and the model is modified by the software.

Therefore, the role of the designer is condensed in the definition of the characteristics of the building, with some mathematical relationships between the elements and the respects of some parameters, while the model is defined by the software.

2.1.2 Software for Parametric Design

Parametric design is achievable only with computers, so a lot of different software were born like Revit, ArchiCAD, and Rhinoceros.

The first two are specifics for architecture, they have a lot of specific functionality for the building, like wall definition and the easy insert of doors and windows, but they are "statics" because is really difficult to define elements different from the pre-defined ones and it is really difficult to realize complex shape.

On the opposite, Rhinoceros is not specific for the architecture, but it is for "general" design, from mechanics to interior decoration, thus it is more "dynamic" because is easier to modify the shape of a structure, but it is not all.

One of the powerful plug-ins of Rhino is Grasshopper, which is a visual-coding software for the definition of the shape in Rhinoceros. Therefore, it is possible to code an algorithm for parametric design in grasshopper and visualize the model in Rhinoceros. Grasshopper has a lot of different commands, from the most general like basic geometric elements to the most specific commands to define wall and windows like Revit; but the most powerful commands are the "scientific" one, like a mathematical function, physics solvers, and evolutionary algorithms. This allows a bigger editability of the models, indeed, Grasshopper is used by many big architecture studios, like ZHA or Foster and Partners.

In this thesis, Grasshopper is used for the design, shape optimization, and structural optimization of a roof, so in the following many aspects of this software will be explained.

2.1.3 Examples

Although the parametric design has developed with computer science, the first examples that can be brought back to parametric design were realized in the star of the 20th century from Antoni Gaudì in the design of the church in the Colonia Güell or for the Sagrada Familia. The artist built a scaled model with some chains and loads where chain length and the load were the parameters.

Fig. 1.1 An upside downforce model of the Colònia Güell [1]

After the second world war, the concept of parametric design was developed by a designer like Frei Otto, Félix Candela, and Luigi Moretti; which built bring back the concept used by Gaudì to define

structural shapes. Some examples are the Olympiastadion in Munch from Frei Otto and L'Oceanogàfic from Félix Candela.

Fig. 1.2 L'Oceanogàfic from Félix Candela [2]

In all these projects, the parametric design coincides with the form-finding, the definition of the shape that minimizes the use of structural material exploiting the shape of the structure. With the development of computer science, parametric design disconnects itself by the form-finding for be used for the definition of new and complex shapes, that not necessarily are more efficient. It becomes an instrument for the research of new shapes in architecture and the born of o new architectural current called *Parametricism. S*ome examples are the designs of Zaha Hadid and Frank Gary like the MAAXI museum in Roma and the Guggenheim Museum in Bilbao.

Fig. 1.3 The Guggenheim Museum in Bilbao [3]

2.2 Mathematical Optimization

As said before, the parametric design is based on the use of the mathematical function to define a model; between the most used function there are the maximum or minimum ones used to manage some characteristics, like the volume or the number of roof panels, it is also possible to speak of *Mathematical Optimization.*

***"Mathematical optimization** is the selection of the best element, about some criteria, from some sets of available alternatives"; For that reason, it is more and more used in the industries.*

Mathematical optimizations, also called mathematical programming, use the maximization or minimization of a function to determine the best solution, so a mathematical model, composed of a set of parameters and a mathematical function, should be determinate; the function is also called the objective function. An optimization model is composed of the following phases:

1- Identification of the problem: the problem should be analyzed, all the constraints and variables identified and the goal defined;
2- Representation of the objective: the objective function should be defined in a way that gives the best fit of the problem, it depends on the variables and the goal, and usually is a maximum or minimum function;
3- Definition of the constrain, the constrain defined in the first step should be represented in a mathematical way like the condition that the function should be respected.

Going into the detail, the objective function *f* is a function with domain $A \in R^n$ and codomain *R*, that is maximized or minimized for search an element $x_0 \in A$ such that $f(x_0) \leq f(x)$ if a minimum is searched; or such that $f(x_0) \geq f(x)$ if a maximum is searched.

Usually, *x* is a vector of *n* dimension, expressible like $x = (x_1, x_2, \ldots, x_n)$ and it is known as vector project variable, so the function *f* has *n* variable. It is also considered a vector of the admissible solution, a subset of A, that contains the elements that are acceptable for a certain application field, that return a solution with a tolerance. Thus x_0 is inside the S set and returns the best solution, $f(x_0)$ called *global minimum (or maximum).*

In a simplify notation we can write, for maximum and minimum, respectively:

$$\max_{x \in S} f(x)$$

$$\min_{x \in S} f(x)$$

If the mathematical optimization is composed of different characteristics and some should be maximized and some minimized, like in this thesis where the number of similar elements and the glass used should be compared, the minimized characteristics can be maximized thanks to a sign inversion. The global maximum of $f(x)\ (x \in S)$ concedes with the global minimum of $-f(x)\ (x \in S)$, so the following expression is valid:

$$\max_{x \in S} f(x) = -\min_{x \in S} \big(-f(x)\big)$$

The explained problem is called *non-constrained optimization*, while, if the solution set do not belong to R^n , the problem is called *constrained optimization* and the solution set is described by a finite set of constraints of equality and inequality, like:

$$S = \{x \in R^n : g_i(x) = 0, h_j(x) \leq 0\}$$

Where *g(x)* and *h(x)* are functions that represent the constrain and there are vectors of assigned function with domain and codomain in R^n. Thus, the solution of the objective function x should be respecting the condition of the constrain; if *g(x)=1 x* can not be a solution. In the application, constrained optimization is more used than non-constrain one.

In real mathematical optimization is not always possible to define the global minimum, or maximum, for the following situation:

1- The solution set S is empty;
2- The solution set is unlimited below, so is not impossible to determinate a minimum of *f in S;*
3- Global minimums does not exist

Therefore, a local minimum x* should be used. This value respects the conditions of optimum, as the first derivative of *f(x*)* should be zero, but there are some other points with an equal or minor value. It is also possible to define *convex* optimization problem, where is always possible to determine the global minimum; and *concave* optimization problem, where is not always possible to define the global minimum because there are a lot of local minimums.

The optimum solution can be chosen for non-mathematical criteria, like in structural optimization, where the shape and the architecture of the structure can be more important than other parameters.

2.3 Structural Optimization

The application of mathematical optimization in structural engineering is called “structural optimization”; which has the goal to reduce the solicitation on the structural elements, the reduction of used material and the waste material, generated by non-optimized shapes. Because it is about different fields of building design, it is possible to define three different typologies of structural optimization:

- shape optimization;

- size optimization;

- topology optimization;

Although there are big conceptual differences between the methods, the procedure is similar for all three:

- definition of an initial geometry;

- definition of the boundary condition;

- definition of an algorithm and an objective function;

- execution of the algorithm and definition of the final shape.

2.3.1 Shape Optimization

In this typology of optimization, the goal is to define the more convenient shape of the building or its elements. More convenient has multiple meanings: it can be the shape of which the bending moments on the elements are minimum, the shape with the least number of different elements, or the shape with the minimum request of material.

To determine the best shape an objective function that evaluates the searched characteristics, like minimum bending moments, is defined and the parameters are varied with an iterative algorithm. This variation can be randomly or made with logic, like the evolutionary algorithms, and proceed until a “good” result is found. Indeed, all the shape and relative parameters are recorded and those that have the best characteristics value are evaluated, because not only the value is important, but the “beauty” is too.

The most common example of an algorithm is composed of an initial model of the structure on which some analyses, like static ones, are performed. From the analyzed model some characteristics are changed and a new model is defined and analyzed; this procedure is performed until the optimized structure is found. Said before, is really difficult to obtain the “perfect” structure, but some structures close to it are found, so the different structures are analyzed in a back-analysis to choose one.

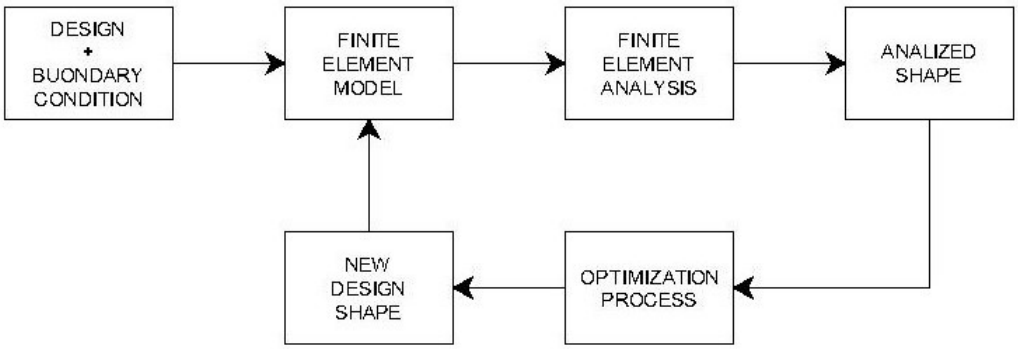


Fig. 1.4 An example of shape optimization algorithm [4]

A particular example of shape optimization is the *form-finding* that is the of this thesis and will be extensively discussed below.

2.3.2 Size Optimization

This is the most common typology of optimization and it is usually performed on the structural elements. It consists of the variation of the size of an element to obtain its maximum exploit, the size should be related to geometric parameters like thickness, area, or moment of inertia.

The objective function should variate the size until is not reach the most efficient; for example, in the case of structural elements, it should research the size with a ratio between the applied stress and the maximum stress between 80% and 90%.

Therefore, structural optimization is the process performed by every engineer when designing a structural element.

2.3.3 Topology Optimization

“Topology is the study of the relationship between geometric parts undergoing deformation” AAD

This is the most general method of optimization and it is based on the comparison of figures, not geometric measures, derived from a Finite Element Analysis. It can be seen as the union of the two previous methods the goal is to find a shape of a structure where the element dimensions are optimized for the stress that they should reach.

It is an iterative process where the starting point is an element with a simple shape, like a parallelepiped, and where loads and constraints are defined. An FEA is performed and the graphical representation of the stress is used to define the non-agent parts of the elements. These parts are subtracted in the new model, the FEA is re-performed and other materials are subtracted; this process continues until the elements are statically verified.

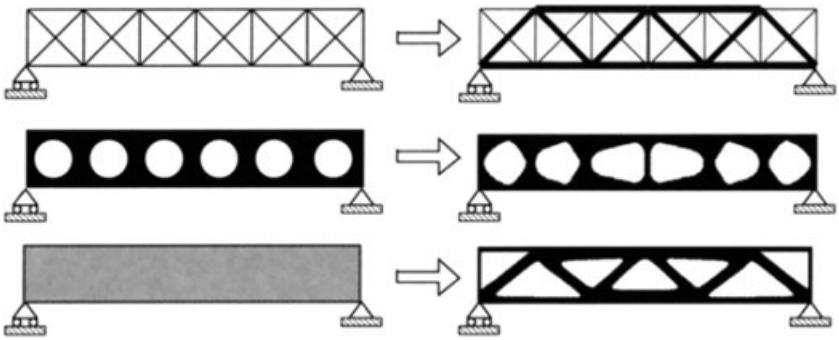


Fig. 1.5 An example of topology optimization [5]

A virtual example could be a reinforced concrete beam subjected only to shear where the concrete is subtracted until it reaches the Morsch truss, obviously is not possible to realize it. There are two main strategies to perform topology optimization:

- the uniformity of the stress, the goal is to define a shape that gives the same in all the elements, usually stress close to the maximum stress of the material;

- the reduction of the compliance, the goal is to define a shape that maximizes the stiffness of the structure and does not overcome the maximum stress of the material.

This process allows to obtain a really efficient structure but due to its complexity is usually performed by software for the FEA.

2.4 Form-finding

A particular case of shape optimization is *form-finding*, an optimization strategy where is search the most efficient shape of a long-span building. Indeed, a shape that minimizes the bending moments in the backbone structure minimizes the partialization of the section of the structural elements, in such a way the biggest part of the resistant area reacts at only one kind of solicitation, traction, or compression.

This process is called *form-finding* and it is based on an easy and ingenious concept: if I take a net and I apply some loads at the nodes, the net assumes an equilibrium shape where all the ropes are in tension because the rope does not resist to compression; but if I take the form and loads before defined realized with a material that resists to compression and I overturn it, I obtain a shape with only compression solicitation.

This stratagem is used to define the structural shape whit the minimum material request. Indeed, some models, real or computerized, are realized with a net that represents the backbone structure and some loads applied to the net node to define the shape. After the definition of the net-load system, the gravity is applied and the system shape is modified to determinate an equilibrium condition. The coordinates of some characteristic points of the new shapes are taken to define the form of the structure, usually, the model has negative Z coordinates and the structure has positive ones, furthermore, the coordinates should be scaled.

2.4.1 The Origins of the Form-finding

The first examples of form-finding were developed with some physicals model of the structure, where the gravity force determines the shape. These models were made with some nets or tensile membranes at which some loads were applied, where the net represents the backbone structure and the weight of the solicitation. These models were subjected to the gravity force that defines the shape in equilibrium, the idea behind this model is to find the shape with the minimum energy, so in an equilibrium condition. Many building designers use these models, from Antonio Gaudì for the Sagrada Famìlia in Barcelona to Frei Otto for the Multihalle in Mannheim.

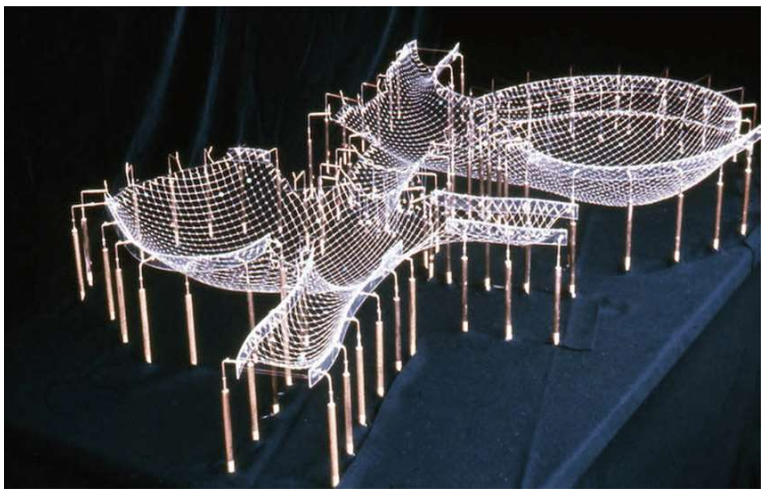


Fig. 1.6 Final hanging chain model for Mannheim [6]

Gravity models are not the only way to define a surface with the minimum energy, another possibility is to use a mix of water and soap, like a bubble blower; indeed this mix has the same behavior of a membrane which tends to assume the minimum surfaces or energy. The main developer of this idea was Frei Otto that use it to define the shape of his buildings, like the new train station in Stuttgart.

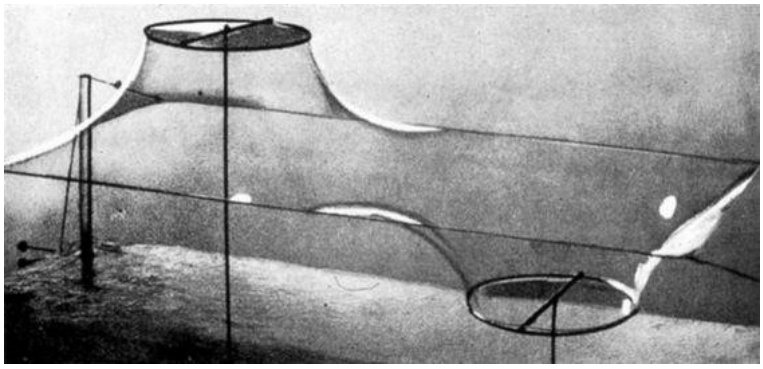


Fig. 1.7 Frei Otto Experimenting with Soap Bubbles [7]

Known the shape of the model, a geometric grid was defied and projected on the surface, so the coordinates of the projection on the shape were evaluated; these were scaled for building the real structure before a structural analysis. This process is also parametric design because the shapes of the models were defined by some simple parameters, like the length of the net or the module of the weight.

2.4.2 Form-finding with Computers

Nowadays the models are not more realized by hand but with some physic solvers, which are software that analyzed the physical behavior of one or more objects. Indeed, the components of the models, like string, membrane, and loads, are defined in the software together to the boundary condition like the anchor. The physic solver analyzed the model and defined the equilibrium configuration, with or without the intermediate step. The shape defined is then analyzed with finite element software and imported into CAD software.

This process allows interoperability between the software and easy management of the model, thus bigger productivity. Indeed, the model parameters are easier to modify, the strength of springs is modified by a number-slider when in the reality the spring should be changed.

But the easy management of the model is not the biggest advantage; the last evolution is the use of evolutionary algorithms associated with the form-finding technique to obtain the best shape. These algorithms exploit the evolution theory to define some parameters that managed an objective, this argument will better explain before, thus they are used to manage the model's parameter to obtain a better solution. This method allows the analysis of a really big number of different shapes in a short time.

An example is the design of the Morpheus Hotel by Zaha Hadid Architects in Macau or the 30th st. Mary axe by Foster and Partners in London.

Since 1965, some computer algorithms were developed for the form-finding with different methods of shape definition:

- The *Force Density Method* by H.-J. SCHEK (S.-J. Schek, 1974), is a matrix method for defining the equilibrium shape of a net structure.

- The *Dynamic Relaxation Method* by A. Day (A. S. Day, 1965)., researches the steady-state of a system, composed of lumped mass linked with spring, under the gravity force with a little displacement of the mass.

- The *Multi-Body Rope* Approach by A. Manuello Bertetto (A. Manuello Bertetto,2020), researches the steady-state of a system, composed of lumped mass connected by inextensible ropes characterized by a certain slack coefficient, under the gravity force with a little displacement of the mass.

These methods are only the computer version of the real models used by Gaudì Otto and the other designer, indeed all of them simulate the behavior of a net with some loads under the gravity force. The pros of this computer method are the easier editability of the model's parameters, the fast realization of the models, and the possibility to shear the model with other software like CAD and FEM ones.

2.4.3 Force Density Method

This method was invented to define all the possible equilibrium shapes of a pin-joint network, of cables or bars, analytically. In such a way, became possible to perform the form-finding of a structure with a computer, overcoming the physical models (S.-J. Schek, 1974).

The force density method is based on a system of linear equations solved in a matrix way based on the concept of *Force-Density*, which is the ratio between the forces on the branches and their lengths. The first step is the definition of the *Branch-node* matrix C_S that is composed of two different matrices: one related to the "free" nodes *C* and one related to the fixed nodes C_F, so that $C_S=[C\ C_F]$. These matrices are composed of zeros and ones, and they are filler between C_S, considering the columns related to the nodes and the row related to the elements. For each element the first node is signed with 1 and the second one with a -1; if the node is fixed it belongs to C_F, else it belongs to *C*

Known the *Branch-node* matrix it is possible to write this system (bold letters are vectors, while capital ones are matrices):

$$\begin{cases} u = Cx + C_F x_F \\ v = Cy + C_F y_F \\ w = Cz + C_F z_F \end{cases}$$

To have an equilibrium condition the forces at each node should be zero, and with the diagonal matrices *U, V* and *W* about the displacement and *L* about the branch length is possible to write:

$$\begin{cases} C^t U L^{-1} s = p_x \\ C^t V L^{-1} s = p_y \\ C^t W L^{-1} s = p_z \end{cases}$$

Where p_x , p_y and p_z are the load vectors and *s* are the vector with forces over the branches.

It is also possible to define the *force-density* as $q=L^{-1}s$, where s is the force over the branches so an axial load, and the relative matrix *Q.* It is also possible to define the matrices $D=C^tQC$ and $D_F= C^tQC_F$.

With some passages reported on Schek's paper is possible to obtain this system of equations for the position of the nodes:

$$\begin{cases} x = D^{-1}(p_x - D_F x_F) \\ y = D^{-1}(p_y - D_F y_F) \\ z = D^{-1}(p_z - D_F z_F) \end{cases}$$

And the relative forces on the branches: ***s=Lq***

Therefore, the definition of a shape is related to the constrain of the node with C_S, the load on the net with p_x , p_y and p_z and the force density with *Q*. Changing just only one of these parameters a new equilibrium shape is defined.

The method explained below is the more general, and some additional constraints are defined, like:

- Fixed branches length;
- Fixed branches forces;
- Fixed node distances.

2.4.4 Dynamic Relaxation

This is an iterative method based on the body motion, indeed a network of lumped mass linked with springs and under a gravity force is considered (A. S. Day, 1965).

This method is considered a network of lumped mass linked with springs and under a gravity force, the goal is to find the shape where the network is in a steady-state. The *Dynamic Relaxation* method is based on the body-motion equation, Newton's second law of motion primarily; and it is an iterative process because the evaluation of the motion-state of the network is done every step of time Δt.

In this method the velocity, the displacement and the forces are evaluated for every lumped mass. Being an iterative method, the coordinates, the velocity and the residual forces on each node should be evaluated for every step.

The first step is to set kinetic energy and nodal velocity at zeros at time *t=0*; because the initial condition is a steady-state where the network is not deformed; since the first step the network is in a motion-state until a new steady-state is found.

The second step is the evaluation of the nodal coordinates and the applied load at *t=0*. In such a way the initial and undeformed condition is evaluated and in the next steps the network will be deformed.

In the third step the tension and the residual forces over each branch are evaluated, these are generated by the loads over the network and depend on the branch disposition. The residual forces are the causes of the variation of the velocity of the lumped mass, so there are the causes of the network's deformation. The residual force in x-direction on the *i* node with a P_{ix} load, link with the *j* node with the branch *m* with a length l_m and a tension T_m is evaluated with the following equation:

$$R_{ix}(t+\Delta t) = P_{ix}(t+\Delta t) + \sum \frac{T_m(t+\Delta t)}{l_m(t+\Delta t)} * \Big(X_j(t+\Delta t) - X_i(t+\Delta t)\Big)$$

The residual forces should be set to zero for the constrained nodes at the fourth step

In the fifth step the velocity and coordinates of each node is updated from once *t* to once *t+Δt* because of the residual forces. Always in the x-direction for the *i* node with a nodal mass M_i, the velocity and the displacement are evaluated with the following equation:

$$V_{ix}\left(t+\frac{\Delta t}{2}\right) = V_{ix}\left(t-\frac{\Delta t}{2}\right) + \frac{\Delta t}{M_i} * R_{ix}(t)$$

$$X_i(t+\Delta t) = X_i(t) + \Delta t * V_{ix}\left(t+\frac{\Delta t}{2}\right)$$

Step three, four and five are repeated until the network is in a steady-state, so the velocity between the two updates does not change.

This method is conceptually different from the force density method, but the results are similar; indeed the parameter that manages the two methods is the same: load, branches' length, and tension over the branches; the ratio T_m / l_m is the definition of *force density* in the previous method (Veenendaal and Block, 2012).

This method is the one used for this thesis, because a plug-in of *Grasshopper, Kangaroo2* uses this method for the form-finding.

2.4.5 Multi Body Rope Approach

This method was developed for the form-finding of quadrangular grid shell and it is always based on a dynamic model of a network of lumped mass; indeed the static configuration is reached with an iterative process based on the equilibrium equations of the nodes defined as once's function. Another peculiarity of this method is that all the branches of the final configuration are in tension, so in compression, if they are overturned.

The equilibrium equations are written for each node, and the equilibrium condition needs that the resultant of each node should be zero; in this case the node i is considered, linked with the node *j, k, l, and m*. This force is composed by the tension on the branches s_a, s_b, s_c, and s_d , the load on the node p_i , the inertial force *F'* and the dissipative force due to the dumping *F''*; the sum of all this component can be written as a system of non-linear differential equation that can be solved in time domine with the Runge-Kutta's method. The equation in the x-direction is:

$$\frac{x_i - x_j}{l_a} s_a + \frac{x_i - x_k}{l_b} s_b + \frac{x_i - x_l}{l_c} s_c + \frac{x_i - x_m}{l_d} s_d + p_{ix} - m_i \ddot{x}_\iota - c_i \dot{x}_\iota = 0$$

Where l_a , l_b ,l_c and l_d indicate the length of the branches is obtained with the Pythagorean theorem between the nodes of the branch, m_i the mass of the i node and c_i the dumping.

To solve the system of the differential equation, that is related to a hypostatic system, the degree of freedom condition should be considered. These conditions are about the congruence of displacement of the branches converging in the same node; for example, in the case of the i node just before considered, it is possible to write the following condition:

$$\{ds_i\}_a^T\{n\} = \{ds_i\}_b^T\{n\} = \{ds_i\}_c^T\{n\} = \{ds_i\}_d^T\{n\}$$

$$\{ds_i\}_a^T\{p\} = \{ds_i\}_b^T\{p\} = \{ds_i\}_c^T\{p\} = \{ds_i\}_d^T\{p\}$$

$$\{ds_i\}_a^T\{t\} = \{ds_i\}_b^T\{t\} = \{ds_i\}_c^T\{t\} = \{ds_i\}_d^T\{t\}$$

Where $\{ds_i\}$ is the elementary displacements of the node considered in the *{n}* orthogonal, *{p}* parallel and *{t}* transversal direction. Node i is an internal node, so it has nine converging ropes and nine constraints, but this value decreases at three and two ropes for borders and corners node while the constrain decrease to six for both.

With the resolution of the differential equation system in the time domain, with Runge-Kutta's method, it is possible to define the equilibrium and steady-state condition of the network.

This method introduces a new parameter, the slack coefficient *sc*, which is the ratio between the distance of two adjacent nodes in the initial grid and the length of the rope that unite them. This coefficient influences the deep of the net Δ_n, which can be evaluated as a function of the orthogonal displacement and the slack coefficient:

$$\Delta_n = f\left(\sum_{i=0}^{n} n_i, \sum_{j=0}^{n} n_j, \sum_{k=0}^{n} n_k, \sum_{m=0}^{n} n_m\right) + g(sc)$$

3. Software

3.1 Grasshopper e Rhino

Rhino is a CAD software for the three-dimensional modeling of complex surfaces, and it has many fields of application, from machine design to architecture; an example is the pavilion of China in expo 2015. This software was chosen for a different reason, like the easy management of complex shapes and its plug-in: *Grasshopper*, *Kangaroo2,* and *Karamba3D.*

Grasshoppers allow the definition of the shape and the characteristics of the object (a roof in this thesis) between different parameters, like maximum height or mesh shape, and mathematical function. This plug-in is a visual coding software; thus, the code is composed of different blocks that explain different actions like generating curves, surfaces, meshes, or solving mathematics equations or Boolean operations. Usually, the commands have some input parameters, like a point coordinate, and some outputs, like a mesh or a shape.

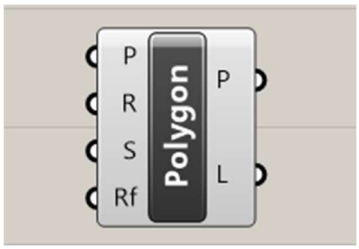


Fig. 2.1 A Grasshopper Command

The flow of the code is given by the links between different blocks. The software interfaces remember an electrical circuit, where the links are represented with cable and the command like electric components. Although, there is the possibility to use some "coding blocks" that allow writing a code in Python or C#. These blocks are useful for list management or for simplifying the code; and this thesis is used to evaluate the number of similar panels, as will be seen later.

Thanks to the interoperability between *Grasshopper* and Rhino the graphical representation of the geometric command is visible in the windows of rhino;, for this thesis, *Rhinoceros* was used for representing the results of *Grasshopper* and not for generating shape.

Grasshopper is vector software, so every element is defined by its coordinates; this allows easy management of the elements and the generation of data lists. For example, the edges of a surface can be a list of lines.

The list of data is a fundamental part of this software and there are two different typologies: *Lists and Trees.* Lists are a simple collection of data, like an array. Trees are more complex because are like a list inside a list, every list is called *branch*, and all the branches create a tree. For example, we can consider a cube subdivided by faces, all the surfaces of faces create a list; while the edges of faces create a tree, where we have six branches (one for faces) and every branch have four elements, the edges of a face.

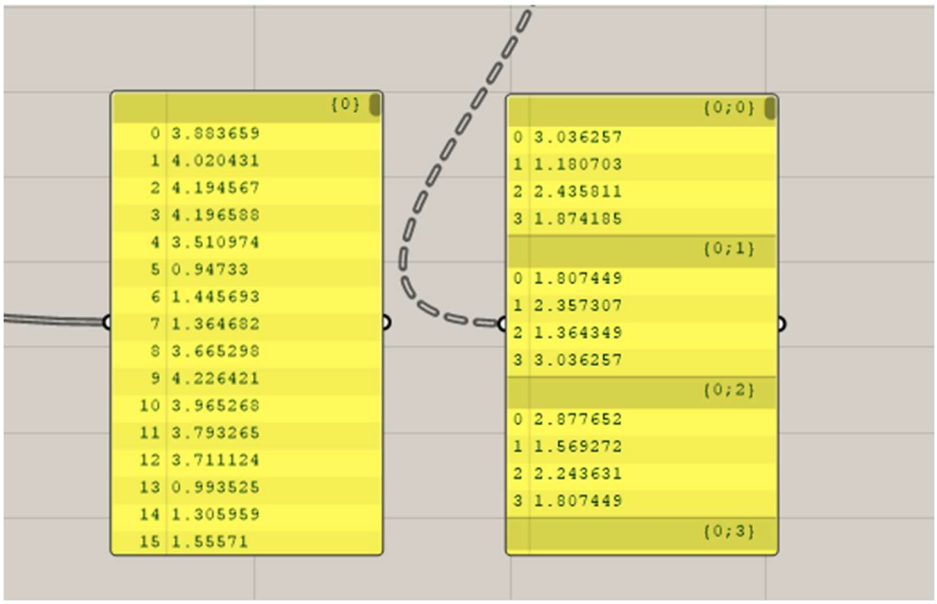


Fig. 2.2 Data inside a list on the left and inside a tree on the right

The choice of *Grasshopper* for coding the thesis algorithm is taken because this software has a lot of plug-ins for the parametric design, indeed it is used by many famous architects like ZHA. Although it needs some coding skills, *Grasshopper* is more editable than software like Revit or ArchiCAD; this allows to define the more complex shape and perform more complex analysis. It is also possible thanks to its plug-in, which is about different aspects of the project, from shape definition to finite element analysis; someone is about machine learning, evolutionary algorithm, and electronic robot control. Is easy to understand the big power of this software. In this thesis the following plug-ins were used:

- *Pufferfish*, for shape and mesh definition;

- *Weaverbird*, for mesh definition;

- *Kangaroo2*, a physic solver for form-finding;

- *Karamba3D*, for the finite element analysis of the structure;

- *Galapagos*, an evolutionary algorithm of defining the shape.

Except for the first two plug-ins which are easy to understand, the other will explain before.

3.1.1 Parametric Design and Form-Finding with Grasshopper

Every project is defined by different elements, like points, lines, and surfaces, that are considered joints, beams, panels, etc. parametric design a relation between elements is defined, usually, a mathematics one, that modifies the design if some parameters are modified, from here the adjective parametric.

A really easy example of parametric design is the definition of a surface with four vertices, where the parameters are the position of they. An algorithm is defined, at the start, there are the point's

coordinates followed by the four elements *points*, they generate the border of the shape with a polyline and the surface is generated using the command *Boundary Surfaces*.

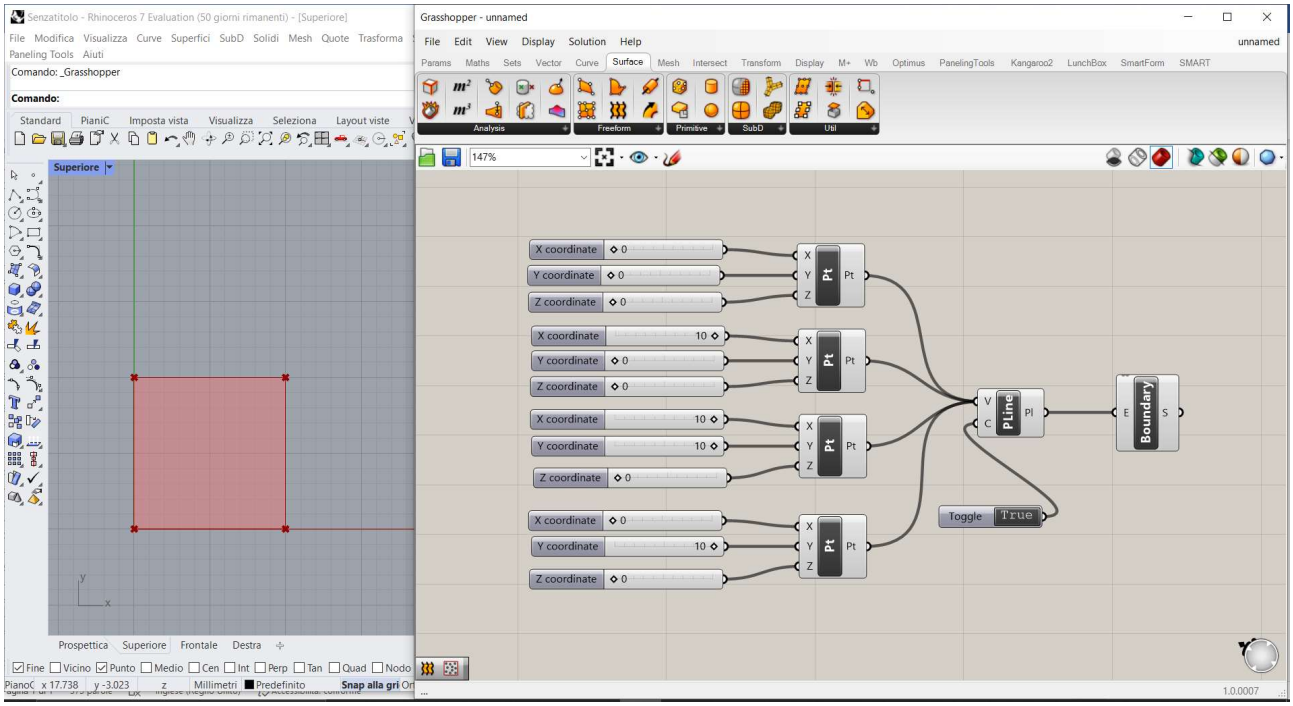


Fig. 2.3 Algorithm and figure representation

If the coordinates of a point are modified, obviously, the shape change.

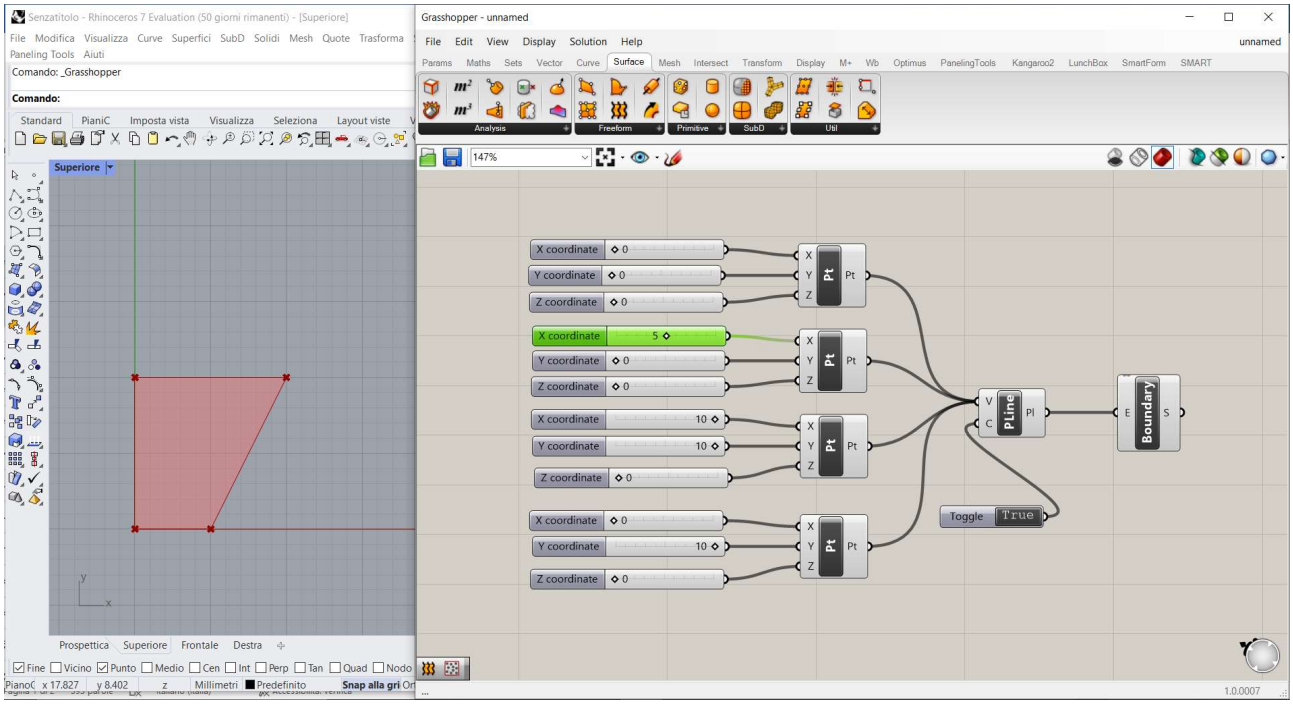


Fig. 2.4 New parameter and figure

This is an easy example of parametric design, but it demonstrates the power of parametric design. Indeed, parametric design is used with more complex functions, like maximum function, and the parameter, like the area of a surface or a length of a curve, is modified to obtain the best solution.

In the thesis a similar process is used:

- a mesh of the base surface is defined, by the number of elements, that is a parameter;

- the mesh is given in input to *Kangaroo2*, the physic solver, that is commanded by some parameters and define the shape of the structure;

- a remeshing of the structure is done to define the panels, also this remesh is commanded by other parameters;

- an analysis of the panel is done and some characteristics are evaluated;

- the parameters are modified to reach the best characteristics.

The algorithm is more complex, but the logic is the same, change the parameters to modify the structure and obtain the searched characteristics.

3.2 Kangaroo2

"Kangaroo2 is a Live Physics engine for interactive simulation, form-finding, optimization and constraint solving"; as reported in food 4 rhino.

3.2.1 Theory behind Kangaroo2

Kangaroo is a plug-in of Grasshopper made by Daniel Piker, by Foster and Partners, to perform the form-finding of structures and simulate the physical behavior of a system of lumped mass and springs under certain conditions. It is based on a solving method called *Dynamic Relaxation*, a dynamic method to define the steady-state solution, equivalent to the static equilibrium of a multi-body system subjected to action, the gravity. The physic solver moves the point by small steps until the equilibrium of the multi-body is reached, that is the configuration with the minimum energy. Therefore, the lumped mass, linked together by springs, is moved in space until the energy is zero. Also, dumping is considered, like the masses are linked together by a spring-dumper system, and it is used to decrease the energy inside the system and ensure the convergence of the system.

This method is the informatic version of the method used by A. Gaudì and F. Otto, with the exception that they do not use springs but lanyards, because also the real models move under the gravity force until they reach the steady-state.

The *Dynamic Relaxation Method* is quite different from the most known *Force Density Method,* which is a "static" method based on the stiffness matrix, but the solution is about the same, the steady-state model is close to the solution of the *FDM* (Veenendaal and Block, 2012).

To define the behavior of our structures there are some *Goals*, which are physical or geometric. The first ones are like spring, anchor points, and load; they have different behavior, but it is always

described by physics law. The second one is like circular tangency or planarity of a face. When the goal is reached the output is zero.

The kangaroo workflow is:

1- Discretization of the base surface of the structures: the surface is simplified in lines and nodes, whit a mesh, and all the edges become lines and the vertices points.
2- Definition of the Particle-Spring System (PSS): the lines are transformed in spring, with *Length(Line)* or *EdgesLengths, with a given stiffness and length ratio;* and at the points can be added a load, with a given module and a gravity direction, or they can be transformed in anchor points.
3- Application of the Dynamic relaxation method at the PSS and definition of the new shape: all the goals are linked with the solver, which moves a load of little steps until every iteration
4- When the steady-state is found, the solver stops.

The solvers give in output vertices and edges of every iteration, which are visible in the Rhinoceros windows until the steady-state is found. The structure also defined is a vectorial model that can be analyzed with the FEM or imported into CAD software.

3.2.2 Main Commands

The simplest command to transform a line in the spring is *Length(Line)*, where in the input is given the line, the length of the relaxed spring (pre-set to the line length), and the strength of the spring, also known as k, in N/m.

When the structure is discretized with a mesh *EdgesLengths* is used. This command is similar to the previous one, in input, there are: *mesh* that needs the mesh of the structure; *length factor* that is a multiplier of the initial length of the mesh (if it is set equal one the original dimension is considered) and *strength* that it is the strength of the spring.

To define the lumped mass kangaroo has the command *Load* where are required the points, the direction of the gravity by a vector, and the weight in N.

Anchor is the command used to fix the point to the floor (XY plane); it is also possible to set the strength and a target point that the anchor can reach during the solution.

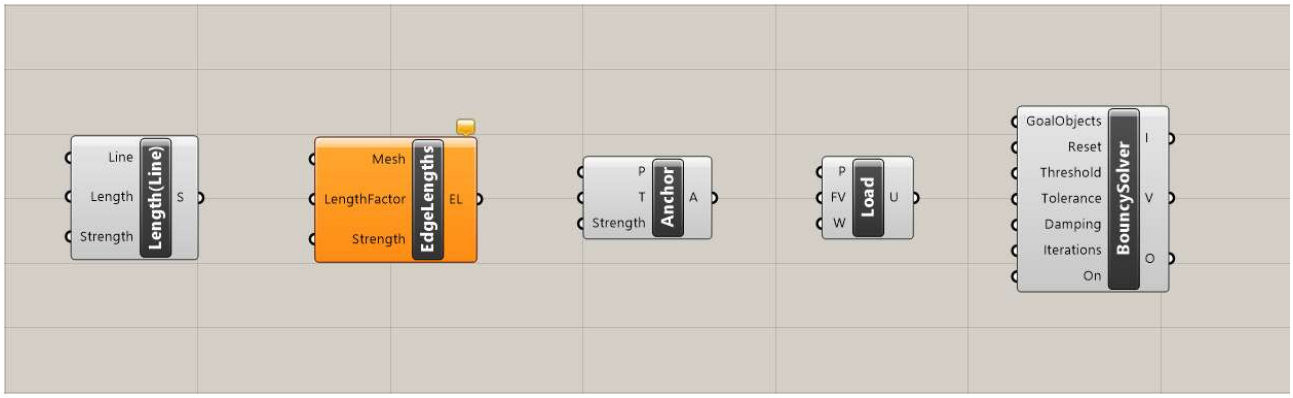


Fig. 2.5 the main Kangaroo2 commands

The core of *Kangaroo2* is the solver, this block collects all the goals and modifies the structure until the total energy is zero, when it is possible. Sometimes, the energy can be found equal to zero and the system of spring and mass keeps moving. Kangaroo has five different solvers, but the most used is the bouncy solver, which has five inputs:

- *GoalObject,* like *Length(Line), EdgesLengths, Load, and Anchor;*
- *Reset*, can be true or false;
- *Threshold*, the solver stops when is reached;
- *Tolerance* of the solver
- *Dumping*, is a value between 0 and 1, if it is 1 all the velocity is preserved between solver iteration, if it is 0 the velocity is not preserved.
- *Iteration*, is the number of iterations performed;
- *On*, if it is false the solver is reset.

The commands reported below are the most common and used for this thesis, but kangaroo has a lot of different commands and analyses; it is possible to consider the action of the wind, pressure over the faces, or the collision between different bodies.

The solution has three different outputs:

- number of iterations done by the solvers to obtain the solution, equilibrium condition, (I);

- the vertices of the solution mesh (V);

- the lines by which the mesh is composed (O).

The outputs lines are disjoint, so a command, *Weaverbird's Mesh From Lines,* is needed to define the mesh.

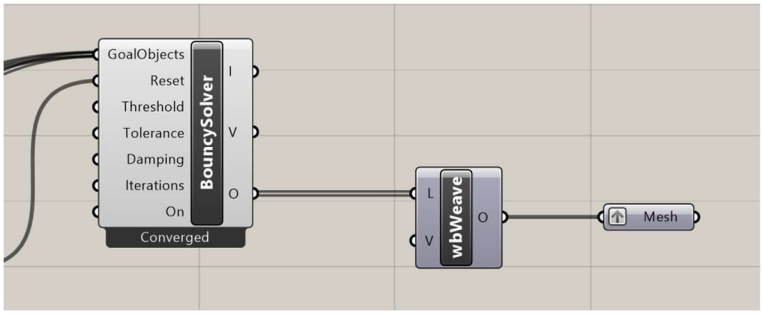


Fig. 2.6 From BouncySolver to Mesh

In the thesis, the *Bouncy solver* is replaced by the *Zombie solver*, which returns only the definitive solution and not the intermediate one. This is a big difference if the output mesh of kangaroo is the input of another command or an algorithm, like in the thesis, because the command should elaborate the mesh only one time when a parameter of kangaroo is changed. This solver has less input than the *Bouncy solvers*, so is less editable, but has the same physics solver inside.

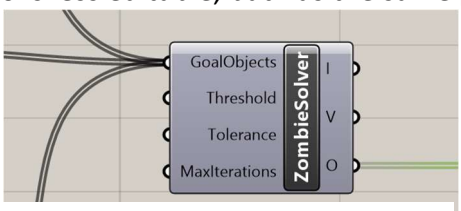


Fig. 2.7 The ZombieSolvers

3.2.3 An easy Example

To give a better explanation a simple example is proposed below. A basic system composed of a line, with a length of 10 m, and two points is considered. The upper point is given as input for the anchor point, while the lower is given in input for the load and the line is given in input at *Length(Line)*.

In the first case, the load is set at 10 N, the strength at 10 N/m, and the length is the same as the line; for the Hook's low the final length is:

$$l = l_0 + \frac{W}{k} = 11m$$

As results by rhino window:

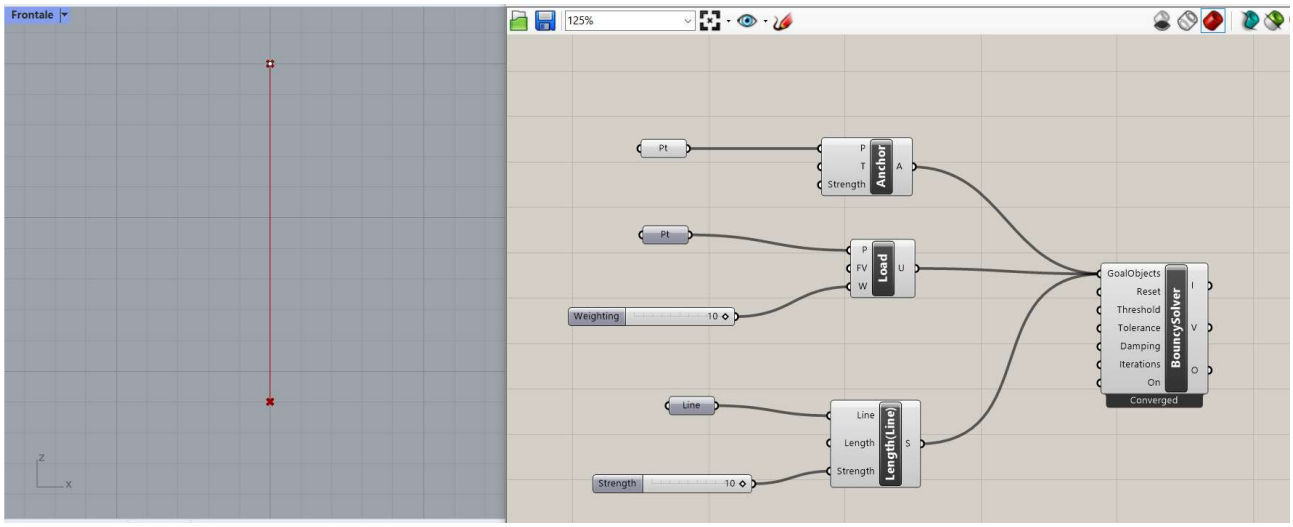


Fig. 2.8 The results of the analysis, each grid-square has 1,0 m edge length

In the specific case of this thesis, the structure is composed of a net of spring, as said in the form-finding paragraph, that can be realized with a net of springs; for obtain it, Kangaroo 2 has a command called *Edge Lengths* that allow generating the net directly by mesh. This command needs to input a mesh; the strength and a length factor, that is preset to 1.0 for do not modify the mesh dimension.

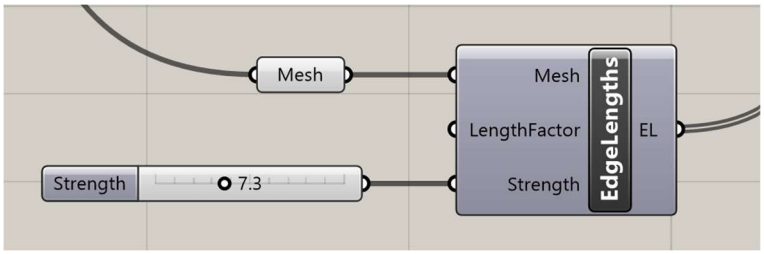


Fig. 2.9 EdgeLengths and its inputs

3.3 Karamaba3D

"Karamba3D is a parametric structural engineering tool which provides accurate analysis of spatial trusses, frames and shells." as reported on the official website of *Karamba3D.*

Karamba3D is a software for the Finite Elements Analysis (FEA) of building structural elements developed by Clemens Preisinger in cooperation with the architectural studio Bollinger und Grohmann ZT GmbH of Vienna; that I should thank for the trial license. This software was developed for work inside *Grasshopper* like a plug-in, allowing to combine parametric design and FEA, generating an algorithm that returns not only the shape of the structure but also the stress and displacement over there.

Like all the grasshopper plug-ins, Karamba 3D has different commands to generate an algorithm that performs the FEA of the shape, indeed the input of the code are geometric elements that are transformed into structural elements by Karamba 3D.

The toolbar of Karamba 3D is composed of different blocks that contain different elements, the most important of them will be explained below.

3.3.1 Model

In this block, there are the commands to transform the geometric elements into structural elements and generate the model. Below will explain the most common commands like *Line to Beam, Support,* and *Assemble Model.*

Line to Beam is the command that generates the beams in the finite element model from the grasshopper lines. It has in input:

- the lines from which generate the beams in the structural model;

- the color of that beam;

- an identifier for these beams, indeed it is possible to generate a different typology of beams with an assigned identifier that the software recognizes allow an operation only over a typology of beams;

- the cross-section of the beams, given by a specific command, but set by default like a circular hollow profile of diameter 114mm with a wall thickness of 4mm with a steel S235 as default material.

It is also possible to set the behavior of the beam whit some options, like the bending stiffness of the elements that can be set to zero for the trusses or set the buckling length.

Supports, define the supports of the structures in given points, insert like geometric points or points identifier and the reference plane. The Degree of freedom of the support can be given in input like a list or set with the appropriate button.

Assemble Model is the command that transform/unites all the elements in a finite element model, in input, has all the elements and their characteristics and return the *Model*, that will be analyzed by other commands, some information about their, the *Mass* of the model in kilograms, its *Center Of Gravity* and the load cases presents in the model.

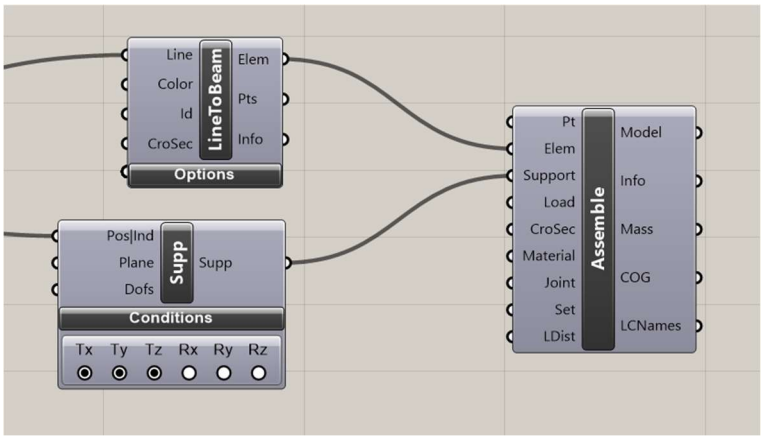


Fig. 2.10 Definition of a model in Karamba3D

3.3.2 Load

Here it is possible to find the commands to define the loads over the structure; unthought there are four different possibilities, the main command is *Loads*. It is possible to define the different kind of load with a drop-down menu that also modify the structure of the command; the different option will explain below.

Gravity defines the action of the gravity acceleration on the elements because they have a mass, defining the weight force. The command has an acceleration of 10 m/s^2 downward that can be modified insert a vector; the presets (0; 0; -1). it is also possible to set different gravity for different load cases or consider the gravity for different load cases by inserting a panel with the load-case

Point (Load) define the point loads on the structure, they can be force or moments inserted like a vector. The position is inserted like geometric points or points identifier and it is also possible to define the load case give in input a text like a name.

Initial Strain and *Temperature* allows considering these stresses in the elements using the characteristics parameters, initial axial strain and initial curvature for the first one and linear temperature change and uniform temperature change for the second one. Also, load-case and elements identifiers are possible to be set.

MeshLoad allows to define of a load over the mesh, that the software transforms from a distributed load in kN/m^2 in point and linear distributed load. The input is the vector with the direction and value of the load, the mesh on which place the load, the elements and node on which place the load if not all of they are loaded, and the load-case. It is also possible to define the direction of the load, which can be:

- *Local to mesh* if it is perpendicular to each face of the mesh;

- *Global* if the load is oriented according to the global coordinates system;

- *Global project* if the load is oriented according to the global coordinates system and it is in the projection of the elements on the global coordinates system;

Is also possible to decide if the load is only applied on the nodes, on the elements, or in both.

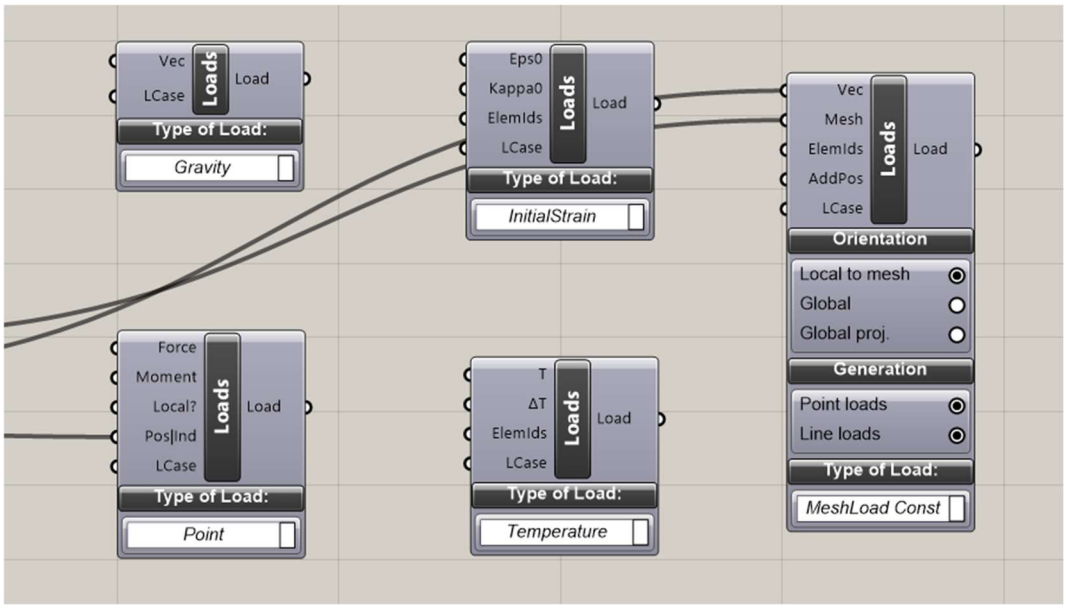


Fig. 2.11 Different typology of loads

3.3.3 Cross Section

In this box is possible to define the cross-section of the elements, it can be selected from an international library of sections, with *CroSecRSelect,* or defined by some parameters, with *Cross Section*.

Cross Section allows to define a cross-section by its geometric parameters, with a drop-down menu is possible to select the typology of the section from which depends on the parameters.

CroSecRSelect is a cross-section range selector, that allows defining a range of cross-sections from a library where the profile is divided by country, shape, and family. It is also possible to insert maximum height and width to shrink the range.

It is possible to give this section range to *CroSecselect* that is used for select a section from the range defined by *CroSecRSelect* using the name of a section or its position on the range given in input at *Elems|Ids*; it is also possible to define the name or the identifier of the elements with this section, with *Name|Ind*, the color and the material. The output is the cross-section for the command *Model*.

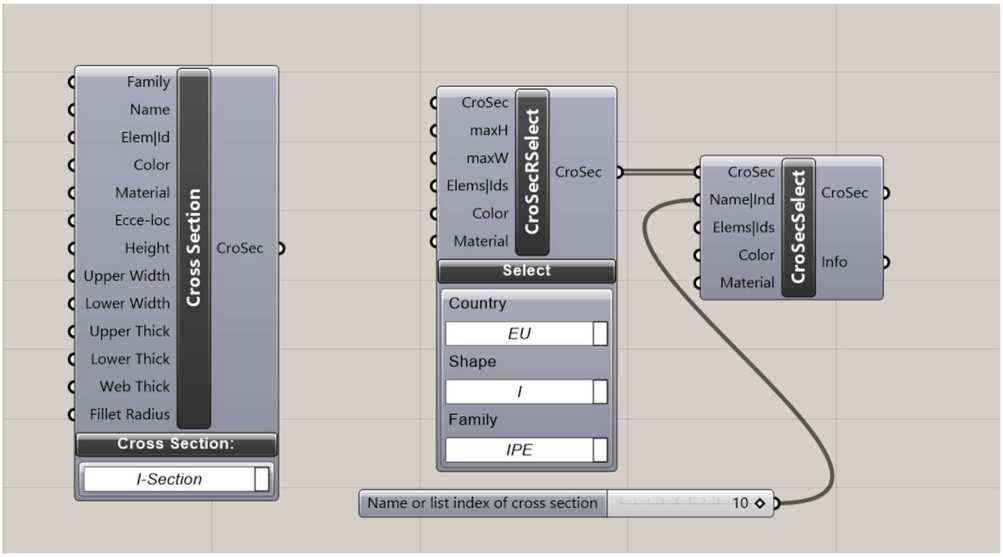


Fig. 2.12 Different Cross-section command

3.3.4 Joint

These elements are made to define the joint between elements, and the most used is *Beam-Joints.* In this command can be set the six DOF of the start and the end of the elements, by default all they are fixed and can be free by the appropriate buttons or with an input list.

For every DOF is possible to set a stiffness, translational [kN/m] or rotational [kNm/rad], for define an elastic joint. Like all the other commands is possible to define the elements with the defined joint thanks to an identifier inserted in *AtElemIds.*

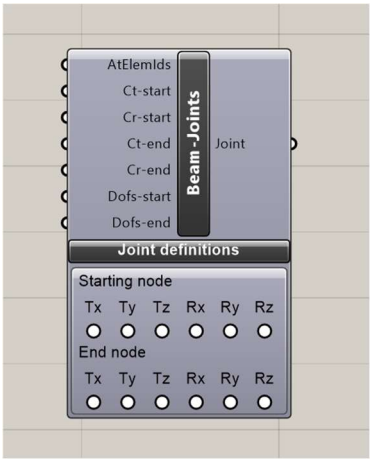


Fig. 2.13 Joint command

3.3.5 Material

The last element characteristic to define is the material, the software allows to define a material from its properties like Young's modulus, share modulus, density, and if it is isotropic or orthotropic with M*atProps.*

For the most common materials is present *MatSelect* where the material is chosen from his family (steel, concrete, etc.) and name (S235, C25/30, etc.)

Also for the material is possible to define the elements which are present by an identifier.

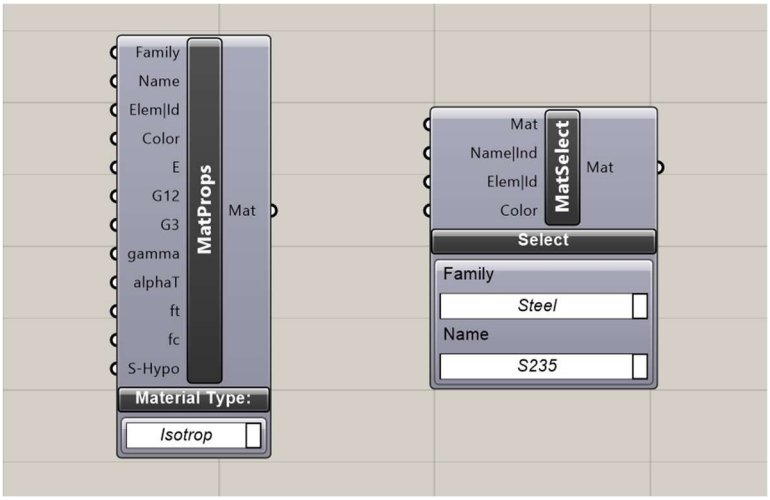


Fig. 2.14 Material commands

3.3.6 Algorithms

The command inside this box is the core of Karamba 3D because they are the algorithms that evaluate the solicitation, the displacement, and the behavior of the model; the most significant are:

- *Analyze,* for the basic structural analysis of the model;

-*AnalyzeThll,* for consider the second-order effects on the analysis;

-*Analyze Nonlinear WIP,* for the analysis of a model with a non-linear behavior, like when the material leaves the elastic branch or with big rotation;

-*Buckling Modes*, this algorithm considers the buckling failure of the elements, not considered in the previous ones;

-*Eigen Modes,* evaluate eigenmodes and eigenvalues of the model;

-*Natural Vibrations,* evaluate the natural frequencies of the model;

-*Optimize Cross Section,* is an algorithm that determines the best cross-sections for the elements of the model.

To perform the form-finding algorithm the command *AnalyzeThII* was chosen; because a general stability analysis is needed, while the most accurate analysis will perform on the structures returned by the algorithm. The input of this command is the model generated by *Assemble Model* while the

output is the *Calculated Model,* the *Maximum displacement* [cm], the *Resulting force of gravity [kN*], and the internal *Deformation Energy* [kNm] for each load-case.

While the *Calculated Model* is an input for the *Results* command, the other parameters are useful for ranking the models; indeed an efficient model has a little displacement, a little force of gravity (weight of the structure) and so a little dissipated energy.

The structure research in the form-finding is mainly subjected to axial load, so the buckling failure should be considered; to do this *Buckling Modes* are used. This command needs to input the output models of *AnalyzeThII* because the second-order effects are evaluated, and return the Buckling load factor *BLFacs*; which is the "scale factor" of which the load of the structure should be multiplied to obtain a buckling failure. This value should be bigger than one to avoid failure.

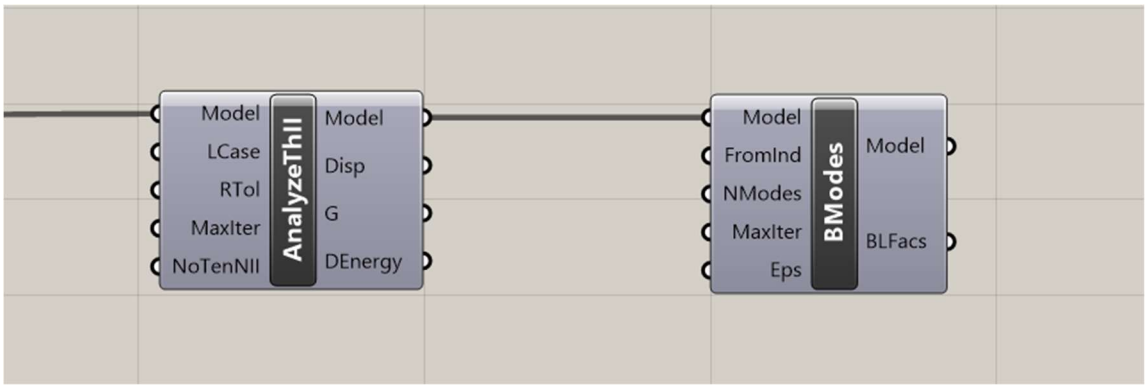


Fig. 2.15 AnalyzeThII and BModes

3.3.7 Results

This typology of command allows obtaining and visualizing the results of the calculated model, like tensions and displacement. It is possible to obtain the displacement of the elements, the reaction forces, the deformation energy, the stress, the displacement of the beam, etc. But the most important for this thesis are *ModelView* and *Utilization (of elements).*

The first one allows visualizing the model, after or before the analysis, and all its components. It is really useful for understanding if the model is correctly defined before the analysis and understanding the global behavior after it. The drop menus allow selecting the shown elements, their scale, their tags and their representation; for all or just one result case, that are the solution of the load-case. *ModelView* returns the input model, the deformed one, only the mesh of the deformed model, and only the deformed axis. This command returns a different graphical representation of the model useful for understanding the global behavior of the structure, for more specific data there are other commands.

One of there is *Utilization,* which returns the stress in the cross-section and its utilization. In the input, in addition to the model, there are

-*Elems|Ids,* is the elements to consider given by its identifier;

-*LCase*, is the load-case to consider;

-*nSamples,* is the number of points element points on which evaluate the returned value;

-*Elast,* is the option of designing the section only in the elastic range, preset true;

-*gammaM0*, is a safety factor for the element failure not initiated by buckling;

-*gammaM1,* is a safety factor for the element failure initiated by buckling;

-*swayFrame,* a flag that considers the sideways due to buckling, preset true for safety reasons.

In the output, there are all the data about the tension on the cross-sections and their utilization, where the utilization is none other than the ratio between the stress, due to all the solicitation or only by one, and the yield stress considering the safety coefficient. All the values for the steel beams are determined according to Eurocode 3 (EN 1993-1-1) from which arrive also the safety factors gammaM0 and gammaM1.

These values are fundamental for determining the static verification of the structure.

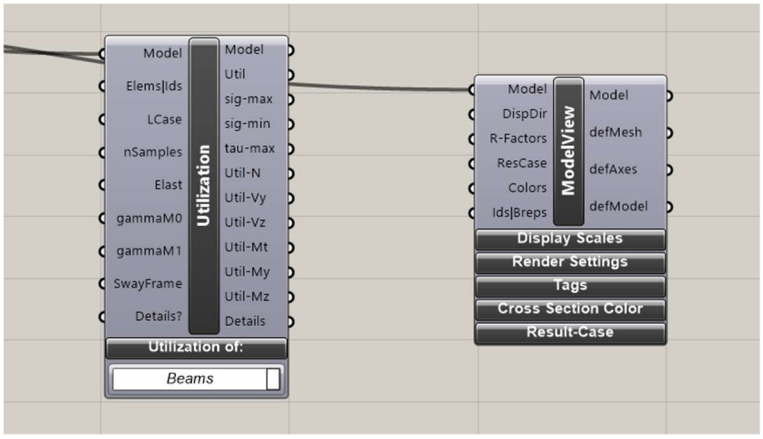


Fig. 2.16 Utilization and ModelView

3.4 Evolutionary Algorithm and Form-finding

At this point of the thesis all the elements for the form-finding, physic solver, and counting algorithm for similar elements are known and now different forms should be generated and analyzed. For generating different roof shapes, parameters of the mesh generator and the physic solver can be changed to achieve a better form; these parameters are the number of the mesh elements, the strength, and the weight in the physic solver.

For modifying the parameters there are only two possibilities: use a random parameter or use an evolutionary algorithm. The first option is not so efficient, the parameters are generated without any logic or correlation and a lot of them are useless, so there is a big waste of time and energy.

The second option is more interesting, the parameters are generated by an evolutionary algorithm that uses the evolution low to generate new parameters. After the first series of values, randomly generated, the following series are generated using the biological evolution lows.

3.4.1 How Evolutionary Algorithms Works

As said before, evolutionary algorithms use the biological evolution theory to obtain the best parameters knowing some input conditions. Therefore, the parameters can also be called genes, different genes generate a genome (like a population) and every iteration of the algorithm generates a population composed of different genomes.

Every combination of genes (a genome) has a different fitness of the solution, if the genes are imaged like the X and Y coordinates and the fitness the Z coordinate it is possible to define a “fitness landscape” where the peaks are the best solution.

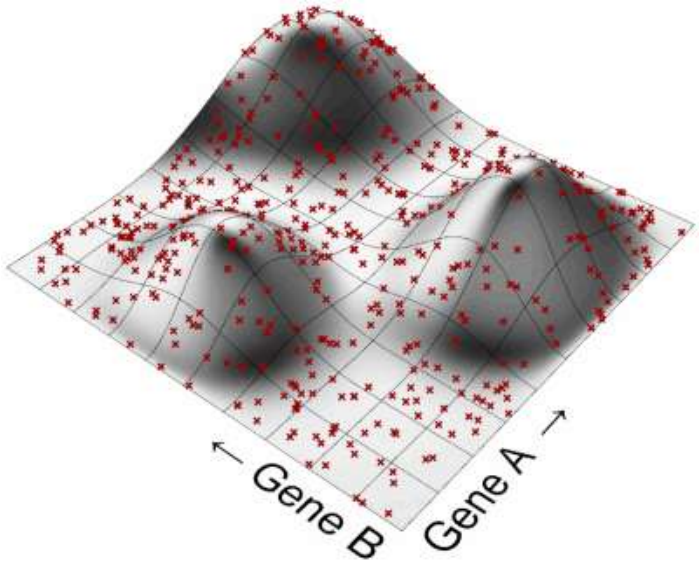


Fig. 2.17 Population on the fitness landscape

The landscape depends on the fitness of the model of the genomes; for each genome composed of two genes X and Y, is possible to define the fitness of the model that is the Z coordinates. In such a way is possible to define a point in the tridimensional space and “draw” the landscape like a set of points.

That landscape is not evaluated a priori, but it defines itself in the algorithm iteration. The first generation has a random value of genes and the following generation tries to achieve the best fitting (solution, top of the peak) following the biological evolution lows.

The algorithm is composed of the following steps:

1- fitness function,

2- selection mechanism,

3- coupling algorithm,

4- coalescence algorithm,

5- mutation factor.

3.4.1.1 Fitness Function

"At least in Evolutionary Computation, fitness is a very easy concept. Fitness is whatever we want it to be. " (David Rutten)

A fitness function is a function that describes the problem; indeed it is the function that describes the better way to reach the best fit of a given genome. If we consider the landscape before and an ancestral genome at the base of a pick, the fitness function is represented by the steepest slope of the peak, which is the more efficient way to reach the top and the best fit; and the steepest slope represents the most successful offspring. Every peak has a basin of attraction, the genomes inside this basin have the highest fitness in this peak. The fitness of the genome does not depend on the basin surface but only on the height of the peak.

Usually, evolutionary algorithms are used for solving maximum or minimum problems, so the fitness function gives the genome with maximum or minimum value.

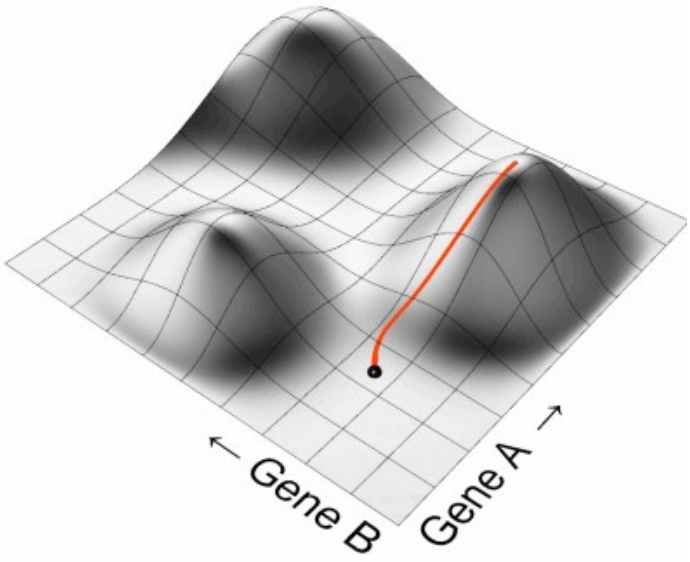


Fig. 2.18 Fitness landscape and fitness function

3.4.1.2 Selection Mechanism

As the biological evolution theory said, in a population, not all individuals get to mate; for this reason, in the evolution algorithm a selection mechanism should be considered for determinate genomes that will create a mating couple. In computer science the selection is given by some simple mechanisms, the most used are:

1- isotropic selection, all the individuals have the same possibility to procreate;

2- exclusive selection, only the top N% of the population with the better fit get to mate;

3- biased selection, when the chance of mate is directly proportional to the fitness.

Therefore, one of these mechanisms should be chosen to define which genome will mating

3.4.1.3 Coupling Algorithm

One time that the genomes are selected, the couple will be defined. A way to select the mate of a genome is the genomic distance, while the mate is chosen through his distance by the genome.

For a better explanation, we can image a population with only two genes, it is possible to generate a genome map, like a cartesian graph with the two genes in the axis, where the genome position is determined by its genes.

In this simple example, the distance is the geometric distance calculated with the Pythagorean theorem.

If we choose a mate close to a selected genome, we obtain offspring like the genome, and the diversity of the population rapidly decreases, so it is really difficult to find a good solution.

On the other hand, if we choose a mate far from the selected genome, so far that the mate belongs to a different sub-species, we will obtain an offspring really different from the parents that need a different fitness function, two far genes can belong to two different peak basins of attraction.

Therefore, the mate for the selected genome should be chosen not so close and not so far from it.

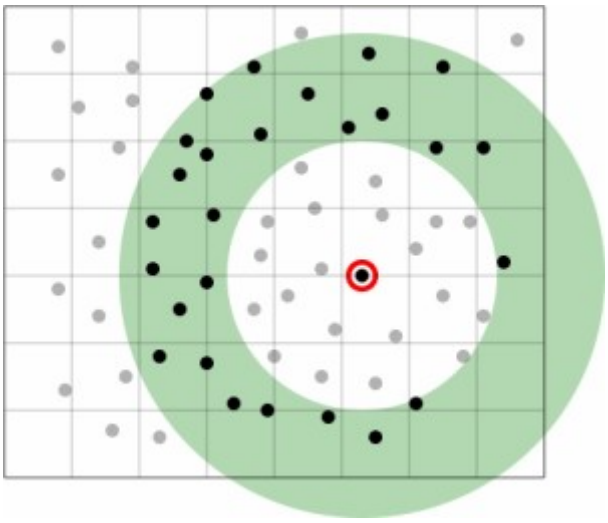


Fig. 2.19 Genome map with the selected genomes [2]

3.4.1.4 Coalescence Algorithm

Once the couples are created, an algorithm for generating the offspring should be defined, or better a way to generate the gene of the offspring. In the digital world there are two main possibilities:

1- Crossover coalescence, if an individual has four genes two arrive from parent A and two from parent B;

2- Blend coalescence, the genes are an average of the parents' genes, a weighted average can be used if a parent has a better fit than the other.

3.4.1.5 Mutation Factor

The mechanism preview described has the goal to reduce the diversity of a population, but that can take to a low resilient population. In our case, low resilient means low diversity of solution and rapid convergence to a solution that cannot be the one with the high fitness.

To have a more resilient population a mutation factor is considered. The "mutant" genome can give a wick solution, and its gene disappears in a few generations or a strong genome that becomes predominant after a few generations thus allowing to obtain a better solution.

There are different behaviors of the mutant factor, like:

- modify the value of a gene by a given percentage;
- invert the value of two genes;
- a gene's value can be set with the interpolation of two adjacent genes.

3.4.2 Galapagos

In *Grasshopper* the main evolutionary solver is G*alapagos* which is based on the concept before explained. The solves has two links, one with the genes of the fitness function and one with the objective; indeed, they are not like input and output but two links with some monitored parts. The objective is only one, differently from other evolutionary solvers.

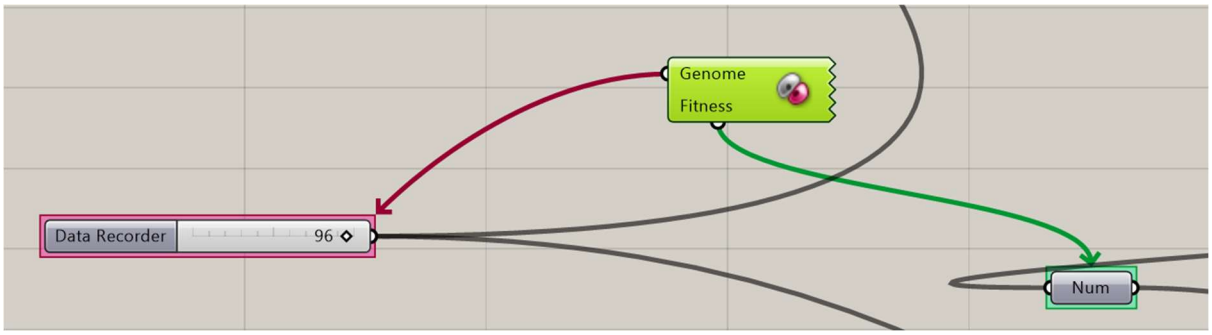


Fig. 2.20 Galapagos command

Click over the command a command window is open, which has three different forms.

The first form is for the settings on the population and the selection mechanism; from top to bottom we have:

1- *Fitness:* the function can be maximized or minimized;

2- Threshold*,* acceptable fitness;

3- Max Duration, enabled with *Runtime Limits*, is the possibility to set a maximum time of running;

4- M*ax stagnant,* maximum number of generations before the solver aborts;

5- Population, number of the genomes of each generation exceed the first;

6- *Initial Boost, is the* value for which *Population* is multiplied to obtain the first generation;

7- *Maintain*, is the percentage of better genomes that are copied in every new generation;

8- *Inbreeding*, is the percentage of elements of every new generation that is obtained by the maintained elements, the other elements are randomly generated.

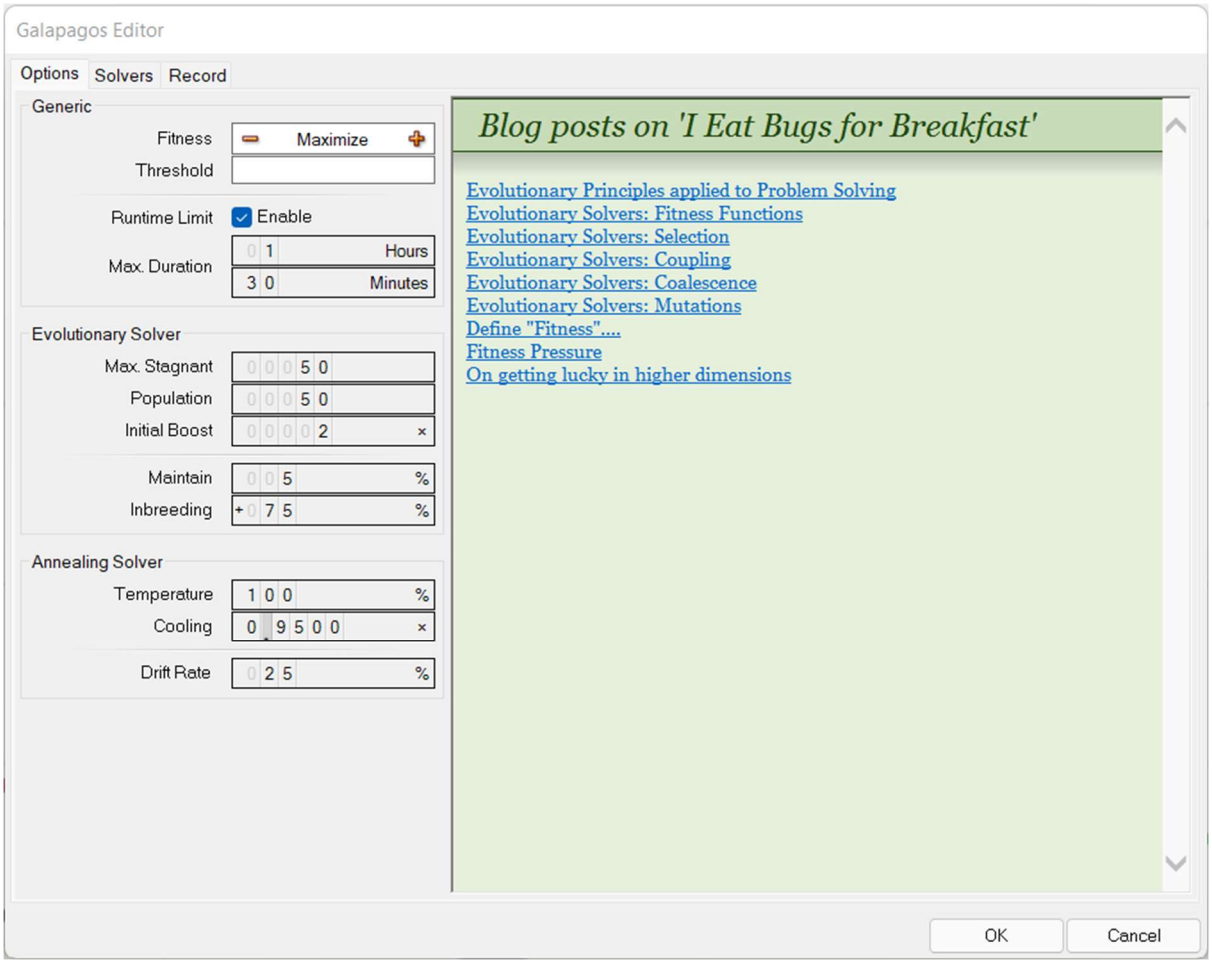


Fig. 2.21 Galapagos command windows-option

The second form is the solver, from there is possible to run the solver and there are four windows with different elements. The first from the top show the trend of the objective to vary of the generation, so is possible to choose the best generation.

The window lower left shows the genome maps and changes every generation.

The window on the middle is a graph with n parallel axis, where n is the number of genes, and for every gene is reported the value on the corrective axis and all the genes of a genome are linked by a polyline; in the image below the genes are six.

The windows on the right contain all the genome of a generation, sorted in descending order, with the objective value; it is also possible to *Reinstate* every genome with the appropriate button.

All the other buttons are to set the windows.

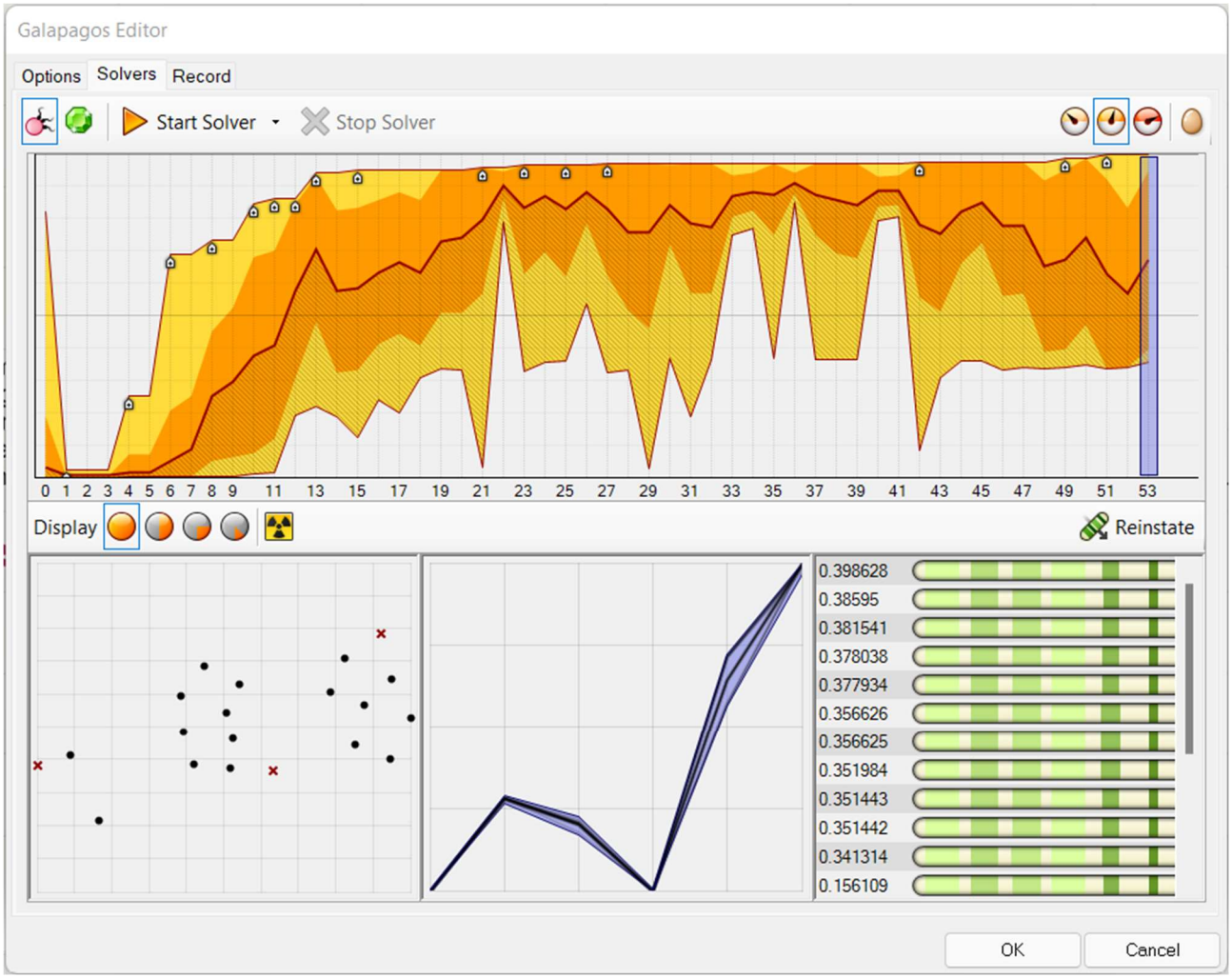


Fig. 2.22 Galapagos command windows- solver

The third form is about the recorded data, that is reported for every generation.

3.4.3 Application

The evolutionary algorithms are used to research an object with the best characteristic, so they are part of a “research algorithm”. Usually, the first part of these algorithms defines the parameters of the objective, like the area or the weight, which is the input of the fitness-function. Indeed, the second part of a “research algorithm” is the fitness function, which generates a relationship between these parameters and returns a value, the objective; this relation can be really easy. For example, if the structure with the minimum weight is researched, the parameters are the weights of the different components, returned by the first part of the “research algorithm”, and the objective is the total weight obtained as the sum of the parameters. If the researched structure should be statically verified, over than the lighter, a penalty factor for the non-verified structures can be introduced. The purpose of these parameters is to increase exponentially the weight of the non-verified structures, in such a way they are excluded by the evolutionary algorithm. If the weight components are:

- the weight of the wall (W);
- the weight of the floor (F);
- the weight of the roof (R);

And a penalty factor P = 1000 if the structure is not statically verified, and an objective O, the fitness function can be:

$$(W+F+R)*P = O$$

The evolutionary algorithm should modify the structure parameters, like height and surface, from which derive the weight components to obtain the lightest structure.

This optimization method, with the use of a penalty factors, is called “constrained optimization”; where P is the big difference between the usual optimizations. The objective of the penalty factor is to obtain a value of O so big that the evolutionary solvers exclude the parameters by the next generation.

3.5 Algorithms for panel counting

A fundamental element to define the shape of the structures is the number of different typologies of panels. Few different shapes allow the industrialization of the building process with a reduction of time and cost.

For this reason, a Grasshopper command was implemented to determine the different typologies of elements and their number; to do it a *Python* component was used. The different typologies are determined by similarity, and not equality because tolerance is always considered in the construction operation; indeed a tolerance parameter is considered in the inputs.

The *Python* component is a component that allows running a Python code inside Grasshopper it needs some input data and a code, it returns some data. In this case, the input is the angles and the edges dimension of the face panels, given as tree, and the tolerance as integer-like percentage; while the output is a list with the numbers of similar panels for every panel.

The output of this component, a list with the number of similar panels for each panel, is fundamental for the choice of the building shape; indeed a genetic algorithm is used to define the different form, where the goal to reach is that all the elements have the same shape with some given dimensions. The number of panels similar to a panel is divided by the number of panels, the objective of G*alapagos* is a ratio of one; the algorithm will better explain in the following.

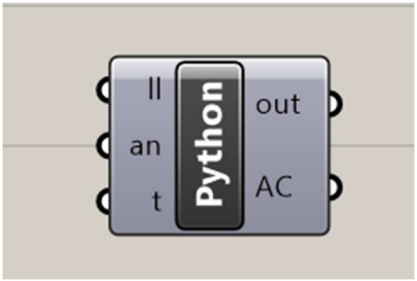


Fig. 2.23 Python command

3.5.1 Input Data

The algorithm was coded only for four-side polygons, but no other geometric conditions are imposed; for this reason, all the dimensions of angles (*an)* and edges (*ll)* should be given as input as lists. These lists should be given as Tree lists sorted in ascending order, this condition is necessary to simplify the code in the researching part, the core of the algorithm. Indeed, in place of performing research of similar elements of the considered panel, angle or side, in the four elements of the analyzed panels, every element of the considered panels analyze its corresponding for the analyzed panel; reducing the analysis from sixteen to four. For better understanding, if we consider a panel with elements *x, y, z, and u* and an analyzed panel with elements *a, b, c and, d*, all sorted in ascending order, only the similarity between *x and a, y and b, z and c, u and d* should be considered instead of the similarity between *x and a, x and b, x and c, … , z and u;* is easy to understand the big reduction of computational effort.

The other input is the tolerance in centesimal, in place to consider only the elements with the same values, a range of value is considered because in the manufacture of the element a certain error is physiologic and is considered. To consider the tolerance, in place of equality between the value of the first branch (V1) and the value of the second one (V2), a range is used:

$$V1 * (1 - t) \leq V2 \leq V1 * (1 + t)$$

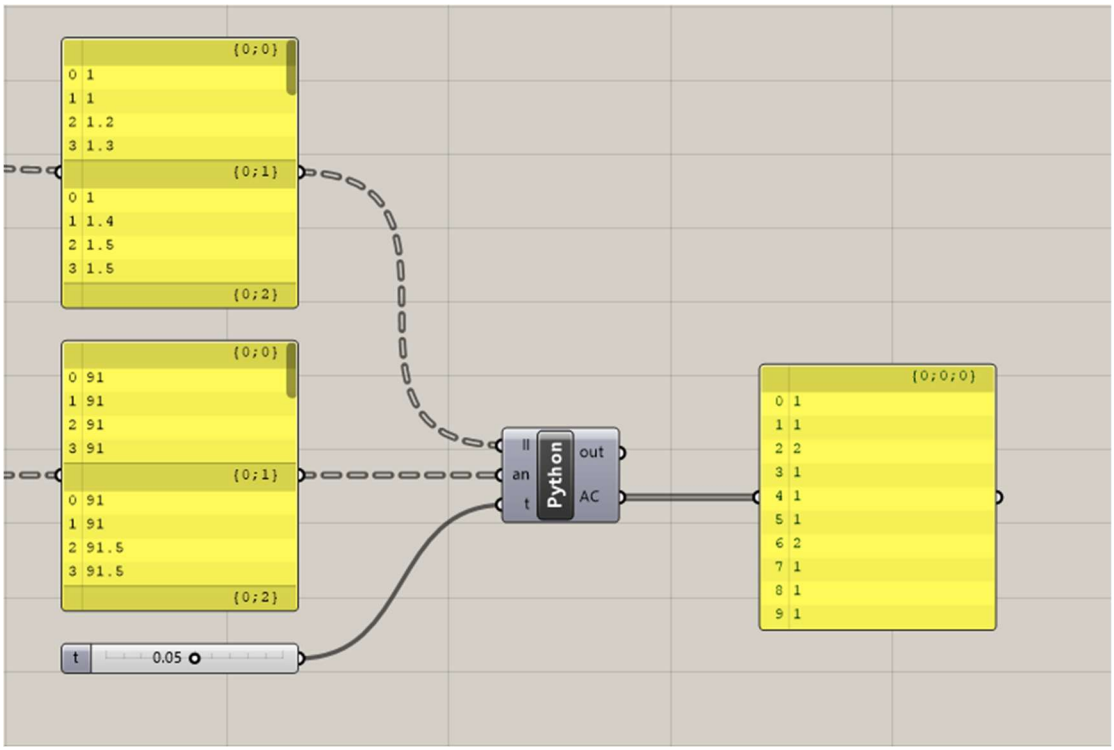


Fig. 2.24 Inputs and outputs of Python command

3.5.2 Analysis Code

As said before, the code is in Python language and use *for* cycles to manage some lists and trees. It is possible to subdivide the code into three different parts:

1- Input and initialization;
2- Determination of similar elements;
3- Definition of the output vector.

In the first part, there is the initialization at zero of a one-dimensional list, AC, with a length equal to the number of the panels, and a bi-dimensional list, C, with a dimension of several panels per number of panels.

The one-dimensional list is filled by the number of similar elements to every panel, while the bi-dimensional list is used to sign the position of a similar panel; if the panel in the x-panel have some similar panels in position a, f, and g, the cells with x-row and position a, f and g is set 1.

The initialization is made by two nested *for* loop; the first set to zero AC and change the row of C. The second *for* change the columns of C, thus to fill the bidimensional list a row is set to zero in the nested cycle and after the row is changed by the first *for* cycle.

After the initialization there is the core of the code, that is the determination of the similar elements and the substitution at 1 of the C cells; also this part is made by some nested *for* cycles.

The first *for* cycle is used to move between the tree branches of which similar elements are searched, it can be called “base branch”, at every step the branch change that is like a changing

panel. Inside it, there is another *for* cycle that changes the examinated panel. The first two *for* cycle are used to change the branch.

Inside the second cycle, there is the first Boolean condition: the number of elements of the base branch and the number of elements of the examination panel should be equal to four; in such a way the code analyses only the branch that represents four-side panels.

If the examination branch respects the first condition there are two other conditions:

- all the angles of the base panels should have similar elements in the analyzed panel, if true a flag call CAN is set to one;

- all the edges of the base panels should have similar elements in the analyzed panel, if true a flag called CLL is set to one;

If both the conditions are respected the element i, j of the bi-dimensional list C is set to one; where i is related to the row and the first *for* cycle and the j is related to the columns and the second *for* cycle.

This procedure is performed for all the panels, and the bi-dimensional list is filled by ones to indicate the position of similar panels.

3.5.3 Output Data

The last part of the code has the scope to count the number of panels similar to each panel, this is performed with a nested *for* cycle that stores this value in the list AC. The first cycle is made to change the base panel and the position in the AC list, while the second change the column's position in the bidimensional list. When a similar panel is found, thus a one in C, a counter is incremented. At the end of the nested cycle, the number of elements reported in the counter is stored in AC. In such a way, the number of similar panels is determined for every base panel, and it is possible to determine which shape is more recurrent.

The command returns only the lists AC that contain the number of panels similar to each panel.

3.5.4 Algorithm for Square Panels Counting

The first optimization attempts were made with the code explained below, but they were too slow and, sometimes evolve to a computer crash; thus a new lite algorithm was coded to reduce the problem. The first algorithm was so slow because it is made to the count the number of panels similar to each panel, so there is a nested *for* cycle, the first for the base panel and the second to analyze the panel list and define the similar ones. The problem is that this procedure needs a big computational effort; indeed, if the list has 1000 panels the python command should analyze 1000000 panels, thousand for each panel. This big value of panel to analyze need a lot of time and can involve a computer crash; so another and lighter Python command to work with *Galapagos* was coded.

The new command is made to research the panels with a square shape and a given edges length, with a given tolerance. The code is always divided into three parts: input and initialization, searching algorithm, and outputs.

The inputs are always the trees of edge length and angular dimension, but in this case, can not be sorted in ascending order because should be all like a given value. The other inputs values are the

edges length *l* in meters and the tolerance *t* in centesimal. Due to the simplicity of the code, only a vector *P* to record the position of the similar panels is initialized at zeros with a length equal to the number of panels; also a counter *C* for the number of similar panels is initialized to zero.

The searching algorithm is composed of a *for* cycle to move between the branches of the edge length and the angular dimensions trees. Inside the cycle, the *i* branches of edges and angles are assigned at two vectors: *an1* for the angles and *ll1* for the length. After the vectors filling there is the first condition that the vectors should be respected, several elements equal to four; in such a way all the non-quadrangular elements are excluded. If the vectors respect this condition the vector's elements are analyzed; the edge lengths should be included in a range of *l±t* while the angles should be included in a range of 90°*±t.* If the two conditions are respected the position is saved in the *P* vector and the counter *C* is incremented of one.

The output of this command are the *P* vector and the number of similar panels *C*; the last one is divided for the total number of panels to obtain the *Galapagos* objective.

The best optimization process requests the maximization of the number of similar panels, not the maximization of a given shape panel; but this procedure becomes necessary to reduce the computational request of the algorithm that allows the computer to find some solutions in a reasonable time.

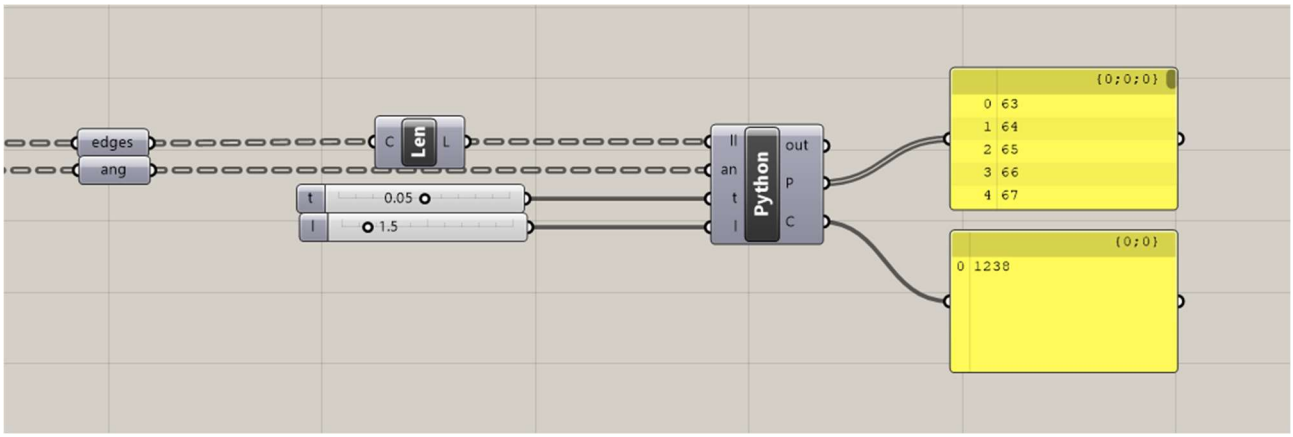


Fig. 2.25 New python command

4. Applications

4.1 Case Study

The form-finding process is performed for a big-span structure composed of three halls, one main, and two secondaries, linked together by four corridors. The target is to obtain a structure that reduces the use of pillars only at the main all, for this reason, a form-finding process was chosen.
The only data about the structure is the external planimetry, the building technology, and the building will build in the province of Napoli.
The structure is composed of a steel warp with some glass panels and some tree-shaped pillars, with four branches, in the main hall to reduce the structural engagement of the warp; the number of pillars is free and will be decided after some structural analysis. The bearing structure is made with a rectangular hollow profile RHS 250x100x10 mm, while the trees are made with a circular hollow profile CHS 406,4x25 mm, but this value can be changed if the structure results oversized. For the roof panels some glass sheets with a thickness of 20 mm and the planar dimension of 6000x3000 mm were chosen; to reduce the material waste they should have a dimension multiple of the glass sheet, so a target planar dimension of 1500x1500 m was chosen. Due to the complexity of the shape, the majority of panels should have this shape, indeed some panels should have a different shape in the junction between two linear parts.

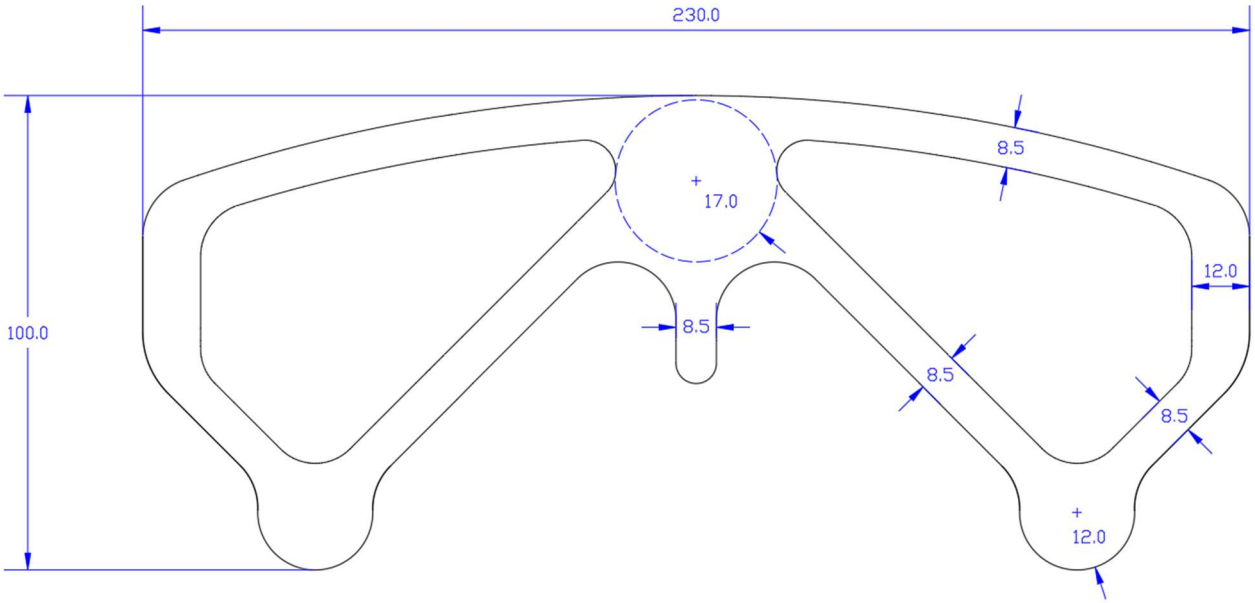


Fig. 3.1 Planimetry of the roof

4.1.1 Load Analysis

For the structural analysis, the acting loads are needed, so a structural analysis based on the Italian normative (Norme Tecniche sulle Costruzioni del 2018) is performed.

The own weight of the steel elements is given by the producer, and it is 57,0 kg/m for the RHS 250x100x10 and 235,0 kg/m for the CHS 406,4x25 mm
The glass panel is considered with a density of 2500 kg/m^3, thus a distributed load of 0,5 kN/m^2 for a thickness of 20 mm.

The overload for the roof, accessible for maintenance only, is 0,5 kN/m^2 from the NTC18
The snow load is evaluated by the normative, with this equation:

$$q_s = q_{sk} \times \mu_i \times C_E \times C_t = 0{,}5 \text{ kN/m}^2$$

where:

- q_{sk} = 0,6 is the reference weight of the snow for zone III where is situated Torre Annunziata.
- μ_i = 0,8 is the shape coefficient for the roof, due to the complexity of the shape the higher one was chosen.
- C_E = 1 is the exposition coefficient.
- C_t =1 is the thermal coefficient, one without a particular prescription.

The wind action is not evaluated due to the complexity of the roof shape and the necessity to perform more accurate analysis, as reported in the NTC18 §3.3.10 .

To perform the verification at the Ultimate Limit State (failure) of the structure, the Italian normative prescribe to use the *Fundamental Combination* of the action, expresses with this equation:

$$F_d = \Upsilon_{G1}G_1 + \Upsilon_{G2}G_2 + \Upsilon_{Q1}Q_{K1} + \Upsilon_{Q2}\psi_{02}Q_{K2} + \ldots$$

where:

- G_1 is the own weight of the backbone structure;
- Υ_{G1} is a partial coefficient for the weight of the backbone structure.
- G_2 is the own weight of the non-structural elements, like roof panels.
- Υ_{G2} is a partial coefficient for the weight of the non-structural elements.
- Q_{Ki} is the weight of a variable action on the structure, like snow and overload.
- Υ_{Qi} is a partial coefficient for a variable action
- ψ_{0i} is a combination factor for the variable action.

ψ_{0i} is a combination factor, to avoid the combination of action that hardly occur together, for the roof ψ_{0i}=0, so snow and overload are not summed together in the same combination.

LOAD TYPOLOGY	CAUSE	VALUE	COEFFICIENT	COMBINATION VALUE
G_1	RHS 250x100x10	0,57 kN/m	1,3	0,74 kN/m
G_1	CHS 406,4x25 mm	2,35 kN/m	1,3	3,06 kN/m
G_2	Glass Panels	0,50 kN/m^2	1,5	0,75 kN/m^2
Q_{k1}	Overload	0,50 kN/m^2	1,5	0,75 kN/m^2
Q_{k2}	Snow	0,50 kN/m^2	1,5	0,75 kN/m^2

Table 3.1 Load Components

The sum of this value is not performed, because they have a different definition in Karamba3D. The own weights are evaluated with *Gravity Loads* where a vector with a module of 1,3 and -Z direction is given in input. For the distributed loads the command *MeshLoad* is used with a vector with a module of 1,5 and -Z direction. All the vector is obtained with the multiplication of the load for the proper coefficient.

4.2 Thesis Algorithm

In this chapter the thesis algorithm will be explained, it was defined with grasshopper using the elements defined before, like kangaroo2 and karamba3d. The goal of the algorithm is the definition of the shape of the roof given some conditions; the definition of this condition is necessary for construction reasons, like the reduction of different elements to decrease the complexity of the structure or the reduction of the number of panels to decrease the material used. Therefore, the condition considered in the algorithm are:

1- Increasing of panels with a square shape with 1,5m edges length;
2- Reduction of the number of panels;
3- Reduction of the structure weight;
4- Reduction of structure cost;
5- Reduction of the Global Warming Pollution.

The first condition is obtained with the python command defined before, which counts the number of panels that have a square shape with 1,5m edges length. The biggest number of panels with this shape is divided by the total number of panels and this ratio should be as close as possible to one when all the panels are similar between there.

The second condition is defined by evaluating the difference between the highest number of panels admitted by the algorithm and the real number of panels; this value is divided to the maximum number of panels to obtain a ratio as in the first condition. In place that the value of panels, the difference is chosen because the fitness function research the maximum, while this objective is a minimum, so the difference should be maximized.

The third condition is evaluated only considering the weight of the single elements, without geometric restriction; while in the fourth the number of panels that have not a square shape with 1,5m edges length is evaluated. Indeed the cost of the structure is proportional to its weight, but an additional cost is considered for the panels without the defined shape because most waste is produced in the cut by the glass sheets.

The fifth condition is evaluated similarly to the fourth, with the same consideration.

Over these two conditions, two constraints are considered, the statical verification and the maximum height of the structure. The statical verification is evaluated with Karamba3D, considering the maximum utilization of the structure, the maximum displacement, and the buckling load factor. If just only one parameter is not verified the objective becomes zero and the Galapagos solver excludes the genes of the structure from the next generation. The maximum height verification follows the same reasoning, but the control parameter is evaluated on the roof mesh.

The two conditions and the two constraints converge in just only one parameter, that is the Galapagos objective. The evolutionary algorithm modifies some parameters, or genes, to maximize the value of the objective.

How the conditions and the constraints are defined in the algorithm and generate an objective; and how the Galapagos command works will be explained in the following step by step.

The algorithm can be divided into the first part, where the structure parameters are evaluated and a second part, where the fitness-function is defined and the objective calculated.

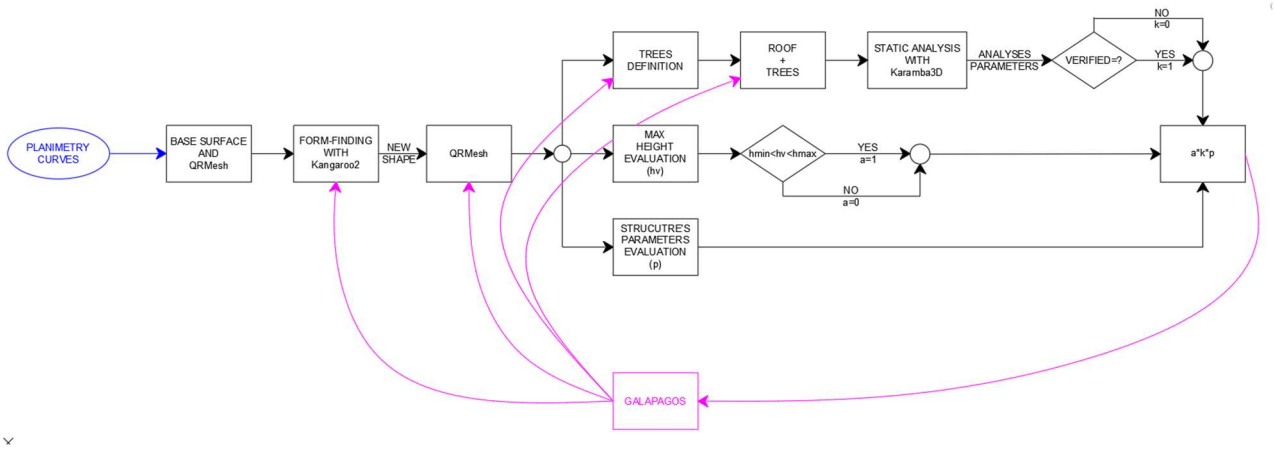


Fig. 3.2 Algorithm overview

4.2.1 From Perimeter Line to Surface Mesh

All the algorithm is based on the perimeter lines of the structure, that is given. From the perimeter, the surface inside it is defined using the *Boundary Surfaces* command, and the surface defined from the external line has subtracted the surfaces of the internal line, in such a way the surface between the lines is defined.

That surface is necessary for defining the mesh for the kangaroo solvers. This mesh is defined with Q*uad Remesh* considering some settings:

- The number of faces, defined with a slider between 600 and 2000, bigger it is smoother the roof is;
- The symmetry of the mesh, should be symmetric about the Y-axis;
- The mesh should coincide with some given lines, that is the local Y-axis of the hall, give in input to the command.

A note should be done on the number of panels in the *QRMesh* settings, this number is a target that the number of panels should reach, not the value of panels. Indeed, the mesh algorithm should respect different parameters over the number of panels.

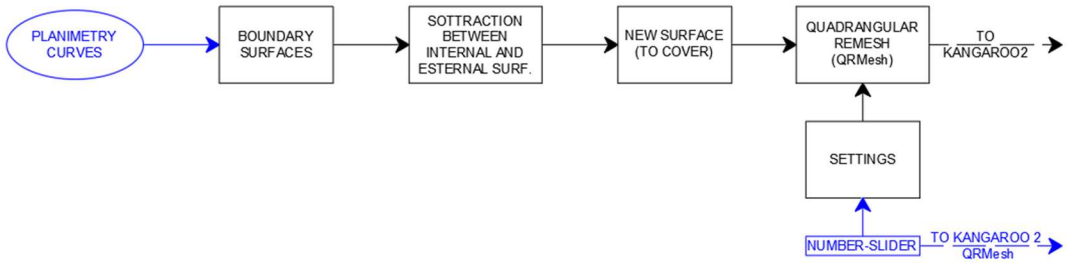


Fig. 3.3 Algorithm to define the Kangaroo2 mesh

This mesh is used by kangaroo2 to define the shape of the roof, as explained in the paragraph before.

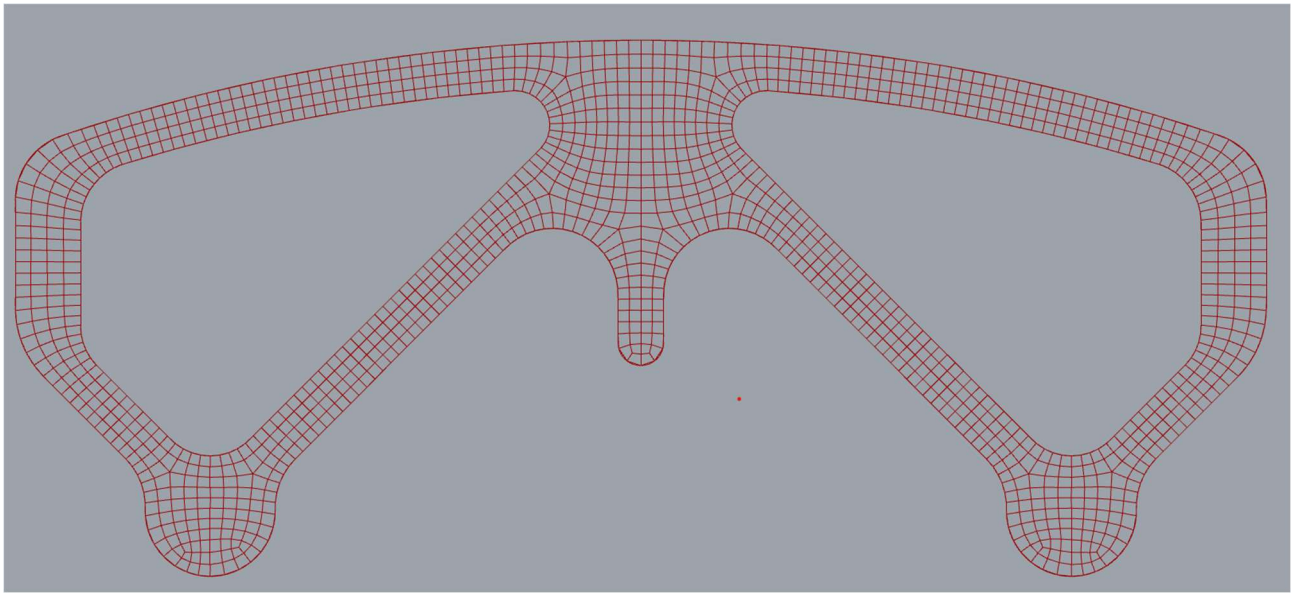


Fig. 3.4 Base mesh for Kangaroo2

4.2.2 Kangaroo2

As said before, to define the roof shape kangaroo2 is used. In the physics solver these elements were defined:

- The springs, defined with the mesh;
- The anchor with the soil, defined with the mesh vertex in the perimeter;
- The load in the node that deforms the spring net.

For the first goal, *Edgelenght* is used, the mesh defined before is given in input for defining the spring net; the length factor is left 1 for having a normal deformation of the spring while the strength is defined by a slider with a domain between 0.0 N/m and 10.0 N/m .

The A*nchors* are defined by the relative command in the point on the perimeter. To define these points the command *Naked Vertices* is used, this command sorts the vertices of the mesh in two lists, the first with the point surrounded by faces and the second one with the point on the perimeter (non-surrendered by faces); the second one is used for the anchor.

The *Loads* are applied to the vertices of the mesh, defined with *DeconstructMesh*, by the relative command. *A* slider between 0.0 N and 10.0 N defines the intensity and a vector in the Z direction with module 1.0 for defining the direction of the gravity. The direction is in the Z direction, and not in -Z direction to obtain a structure on the Z positive side and avoid problems with the load definition in the FEA software or the overturning for a CAD analysis.

Therefore, the shape of the structure is defined by three different parameters: the base surface mesh, the strength of the springs, and the load; all of these are managed by some sliders that can be modified by hand or by a genetic algorithm to obtain a structure that satisfies the condition, in this case only the kanagarro2 parameters are controlled by Galapagos.

As a solver a *ZombieSolver* is used; the characteristic of this solver is that return only the definitive solution and not every step; in such a way the subsequent part of the algorithm is performed only one time decreasing the computational time, the intermediate solutions are not in the steady-state so they are useless.

The time that the solver has to define the shape, it is transformed in a mesh with the command *WeaverBird*, which transforms the output line of the solver in a single mesh. While the output vertices are used to define the maximum height of the structure.

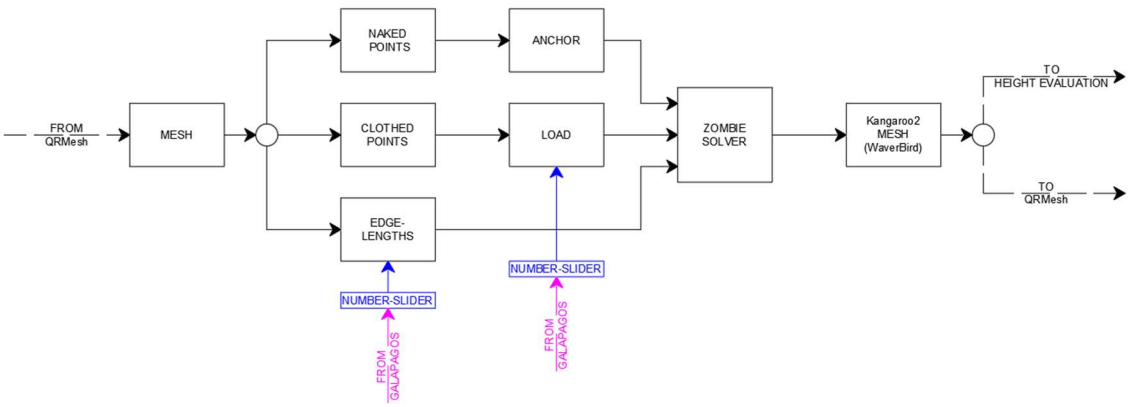


Fig. 3.5 Algorithm for the form-finding

4.2.3 Quadrangular Remesh and Count Of 1,5m Length Elements

At once that the shape of the roof is defined a remeshing of it is performed because in such a way is easier to define a structural grid that respects the given condition. To generate the new mesh *Quadremesh* is still used the settings like the symmetry condition and a slider to define the numbers of panels. Logically, the dimension of the panels depends on their numbers; thus the respect of the conditions depends mainly on this slider, which is a Galapagos gene.

To evaluate the number of edges with a length of 1,5m the edges of the mesh are selected with *MeshEdges* and put in a list. It is given in input to the similarity command for define the number of elements with a length similar to 1,5 m , is chosen to use similarity instead of equality because in the fabrication of the elements there is always a tolerance and S*imilarity* allows to consider there.

The output of this command is a list of true or false relative to each element, this list is given to *Member Index,* a command that researches a member/value inside a list and returns its position in the list and the number of members in the list. In such a way the number of edges with a length like 1,5 m is determinate.

This value is divided for the total number of roof edges, defined by the length of the list of edges returned by *MeshEdges,* and this ratio is used as a control parameter.

The mesh obtained by *Quadremesh* is also used to evaluate the maximum height of the roof; indeed the Z-coordinate of all the vertices is sorted in ascending order and the higher analyzed. If it is outside a defined range of height a flag with zero value is given in input to the ratio part of code; if not the flag is one. This flag is used as a multiplier of the ratio, so, if the maximum height is outside the range, the ratio is zero and the evolutionary solver discards the input values.

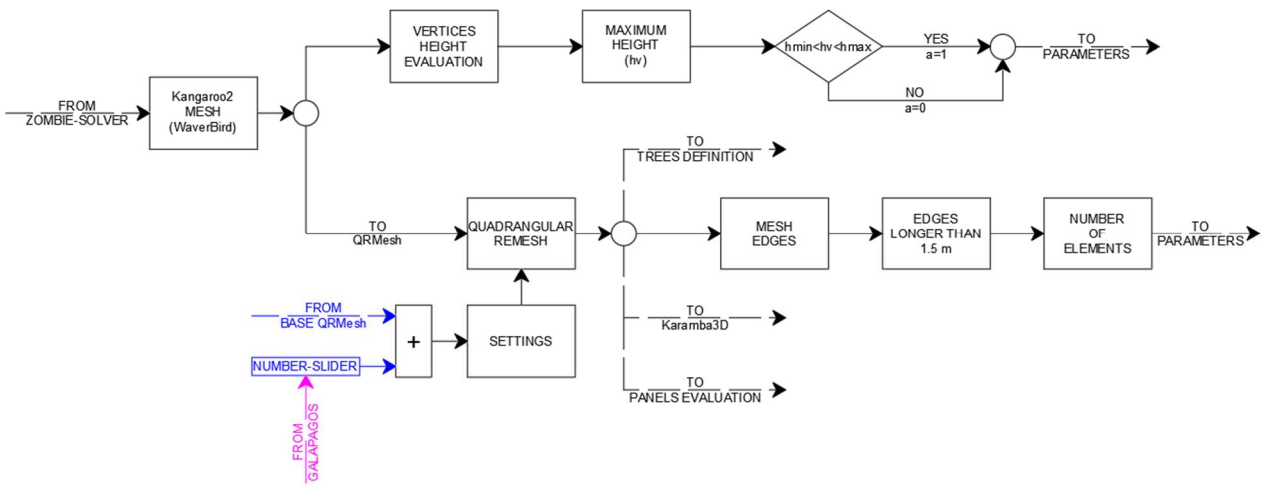


Fig. 3.6 Algorithm for the shape remeshing

4.2.4 Parametric Tree Definition

For a static reason, the roof of the main hall is helped by some tree-shaped pillars joined with the mesh, which is parametrically defined in the code with a little algorithm.

First of all, the positions of the trunks are defined; they are in the vertices of a polygon of which dimension, position, and the number of edges are variable to have the best position of the pillars. This polygon is defined with the namesake command, which has in input the position of the center, the radius dimension, and the number of edges, which are all variable parameters. The polygons can be also overturned by a given angle with *Rotate.* The number of edges and the radius is controlled by Galapagos between two number sliders, the number of trees is from three to six, while the radius is from ten to eighteen; the rotation angle is always 270° to have the symmetric disposition of the tree respect to the Y-axis.

In the algorithm also the panels are considered because at their vertices the branches are joined to define the roof surface. But before the bases of the pillars are projected on the roof surface to determine the distances from the base; that is multiplied for a coefficient, like 0,6, to define the heights of the trunks. Known the heights, some lines with these lengths parallel the Z-axis and with the origins in the polygon, vertices are defined as the trunk.

After the trunks definition, the algorithm determines the vertices of the panels closest to the projection of the vertices of the polygon and the nine panels vertices near them. The position on the panels' lists of these points is needed to define the group of nine panels close to the projection of the polygons; each group of panels is transformed into a unique quadrangular surface with *Brep Join* and the four vertices are united with lines to the top of the trunk.

In such a way the parametric pillars with a tree shape are defined, and all the characteristics are editable to modify the shape.

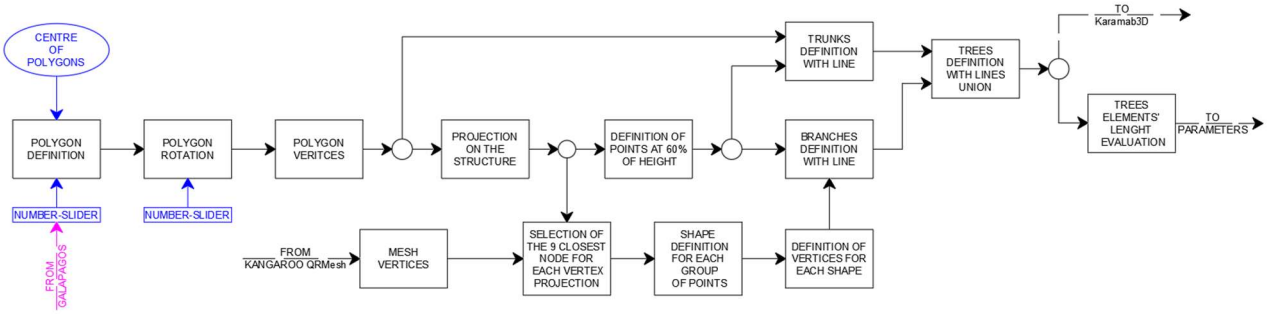


Fig. 3.7 Algorithm for the tree-pillars definition

4.2.5 Panels Evaluation

The conditions to define better paneling are related to the mesh, but it is composed of edges and faces that the software considers like *mesh face* that is different *geometry* than *surface* and does not contain the same data or is really difficult to extrapolate them. For that reason, should be defined a new surface for every face with an algorithm that defines a surface from four points, using *4Point surface*, which are the four vertices of every face. The vertices of the mesh are obtained from *Deconstruct mesh* and *Deconstruct face* and given to *4Point surface;* getting a surface for every face and a list of theirs.

From this list of surfaces are obtained:

1- The area of the roof like sums of the areas of the panels;
2- the number of the panels;
3- A tree of edge length, with several branches equal to the number of panels and every branch with four values;
4- A tree of the angular dimension of the faces with a number of branches equal to the number of panels and every branch with four values;

About the angular dimension, the definition of the values is not so easy because inside grasshopper is not present a dedicated command, thus a little algorithm should be defined. The input of this algorithm is the edges of a face, a branch, from which are defined two lists of edges; the first one is the content of the branch, the second one the content of the branch shifted of one position. For example, the first list has four elements with positions 0, 1, 2, 3 in the branch; while the second list has four elements with positions 1, 2, 3, 0; so is the list of the next edges. From these two lists is obtained the plane between an edge and its next. The two lists of edges and the plane is given in input to the *Angle* command that defines the angle between the two following edges.

At this point, there are two trees, the first one with the edges length and the second one with the angular dimension, that they are given in input to the python command to evaluate the number of similar elements.

The output of the *Python* command is the number of panels similar to a square with a 1,5 m edge length and the position of the similar panels. This number of panels is divided by the total number of panels before evaluation and this ratio represents the characteristics of maximization of similar panels.

The other objective of the algorithm is the reduction of the number of panels, to reduce the quantity of material needed for the structure, and so its cost. To insert this objective in the fitness function should find a value that can be maximized. For this reason, is evaluated the difference between the maximum number of panels admitted as a sum between the upper limit of base mesh elements (2000) and the upper limits of roof mesh (5000), that is 7000 panels; from this value is subtracted the real number of roof panels evaluated before. This odd should be maximized to minimize the number of panels, but it is divided to the maximum number of panels to have the same order of magnitude of the square panels' ratio.

These two ratios are considered in the fitness function together with the statical verification and the height verification.

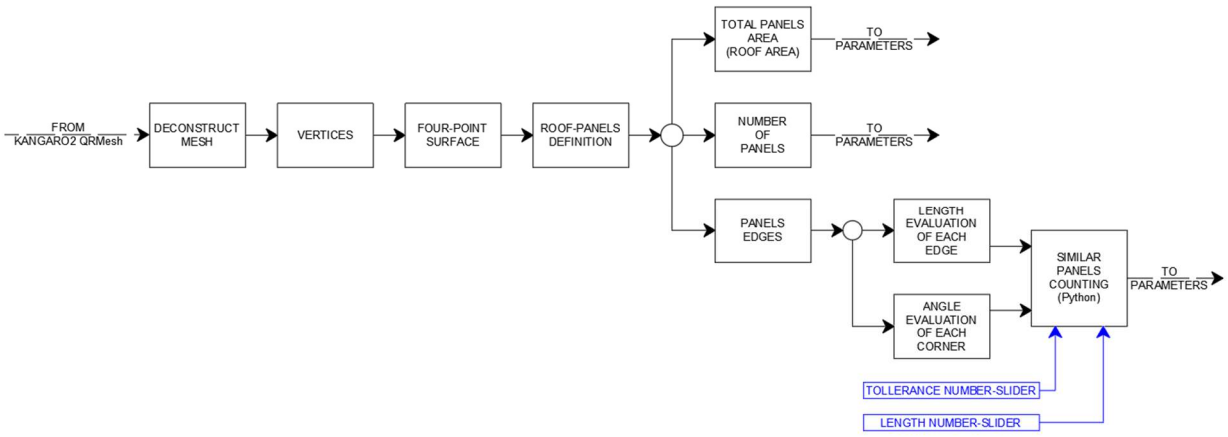


Fig 3.8 Algorithm for the panel's evaluation

4.2.6 Karamba3D

After the definition of the shape of the structure a structural analysis should be performed to evaluate the structural stability; to do this the FEM software karamba3D is used.

The first step is the model definition, so the line that compose the trees and the roof mesh is given in input at two different *LineToBeam,* in such a way is possible to define two different elements typology with different characteristics. Indeed, the trees are made with a circular hollow profile CHS 406,4x25 mm with a steel S275 and identify with "T"; while the roof structure is made with a rectangular hollow profile RHS with a steel S275 and identifies with "E". The edges cross-section can be chosen, with a number-slider, between all the RHS profiles smaller or equal to the RHS 250x100x16 and they are ordered in area ascending order. About the supports, there are defined some tridimensional hinges without stiffens for every edge vertex in the ground with the command *Supp.* The link between edges is not fixed but there are hinges. For the load definition there are *Gravity Loads,* to define the proper weight of the structural elements, with a safety coefficient of 1,3; while the carried and accidental loads are defined with *MeshLoad Const* with a value of 1,0 kN/m^2 and a safety coefficient of 1,5. All the load and coefficient derive from the structural analysis before being defined using the characteristic load combination as defined in the NTC18.

All the model characteristics are given in input to *Assemble Model* and after analysis is performed considering the second order effects due to axial load with *AnalizeThII*; which returns the analyzed model and the maximum displacement. This value is fundamental for the structure analysis; indeed the displacement should be reduced at the minimum value for aesthetic reasons and to avoid the glass panel cracking.

The calculated model is given in input at three different commands, *Beam View, Utilization, and Buckling Modes;* while the first one is only to visualize the result of the analysis and understanding the global behavior of the structure, the second one and third one are fundamental for the algorithm. Indeed, *Utilization* returns the utilization of each beam, using a safety coefficient gammaM0 of 1,1 as reported in the Eurocode 3 chapter 5, inside a list that is sorted in ascending order and the bigger value is analyzed. If it is less than one all the structure is statically verified; it is bigger, no. *Buckling Modes* make a buckling analysis of the structure and return the *buckling load factor*, which should be more than one because the structure is statically verified.

Therefore the tree structure's parameters, maximum displacement, maximum utilization, and buckling load factor, are given in input to the fitness function. Because of the different fitness-function that will be defined in the next analyses, not three fixed control values of the parameters can be defined but should be modified for each analysis.

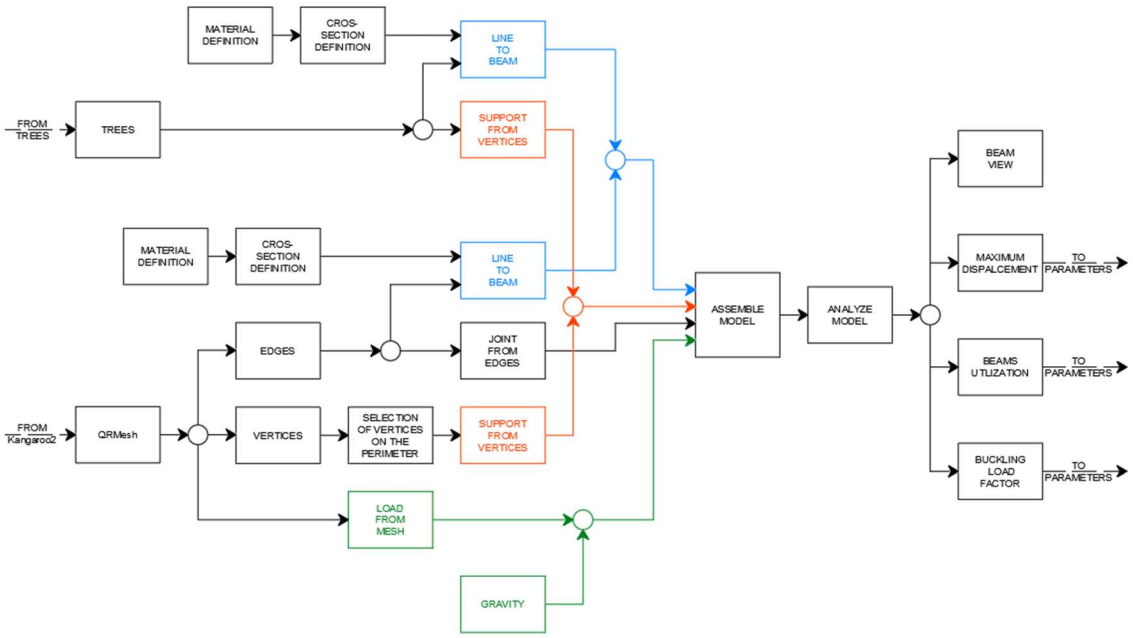


Fig. 3.9 Algorithms for the structural analyses

4.2.7 Fitness-function and Galapagos

In the previous paragraph are explained the part of the algorithm with the purpose of the definition of the structural parameters, therefore the first part of the "research algorithm". The second part of the algorithm is the definition of the fitness function given the structure parameters. In the following different analyses with a different objective will be performed, therefore in the first part of the algorithms only the geometric parameters are defined:

- the roof area;
- the number of panels;
- the number of panels with square shape and an edge-length of 1,5m;
- the total edge-length, that is the total length of the grid-shell elements;
- the total number of grid-shell elements;
- the number of grid-shell elements with a length of 1,5m
- the tree-pillars edge-length.

Four other parameters are defined as penalty factors:

- the maximum height of the strictures;
- the maximum displacement of the structure;
- the maximum utilization of the elements;
- the buckling load.

From these parameters the different fitness-function, with a different objective, will be defined; but all of them should have only one objective. Indeed, Galapagos is a mono-objective evolutionary solver, thus it can modify different parameters but analyze just only one objective; differently from other evolutionary solvers, like octopus.

At this point the “researching algorithms” is defined, thus it is possible to proceed to determine the structures that satisfy the different objective, related to the analyses. The parameters that Galapagos can modify, so the algorithm genes, are:

- the Kangaroo2 strength;
- the Kangaroo2 weighting;
- the *QRMesh* number of elements;
- the tree-pillars number;
- the tree-pillars distance;
- the cross-section.

Not all the parameters are considered in the analyses and the fitness function will change for all the analyses; but these will be explained in the relative paragraph in the next chapter.

5. Analysis

At this step of the thesis, the form-finding algorithm is ready to perform some analyses to define the different roofs given changing some parameters. Indeed, not just one structure is researched, but different ones with different characteristics: like the presence or not of the tree-pillars, different importance of the number of panels concerning the panel's ratio, the minimum weight, or the cheapest structure. These parameters are all related to the cost of the structure, indeed less material meaning a cheaper structure; but it should be related to the labor cost. Nowadays the most expensive voice in the realization cost of a structure is the labor and not the material, whereby if the grid elements are standard the labor request is less and so is the building cost. In a market different than this behavior can be overturned, so the complementary analysis is performed.

The biggest part of the analyses uses Galapagos to find the best solution in the different configurations; thus the different parameters are considered as genes, but sometimes some parameters are excluded by the analysis and are fixed, but this is specified in the analysis' paragraph.

To have a track of the analyses history that allows understanding how the solver reached the objective, all the parameters and partial objectives of the analyses are recorded with some DataRecorder commands linked with the genes and the objective.

Below there are described the analyses performed to understand the behavior of the algorithm and the structure.

5.1 First Analysis

In the first analysis, all the parameters are considered with the goal to find a benchmark structure to compare the other structure; indeed the fitness function researches the roof with the maximum number of panels with square shape and edge-length of 1,5m and the minimum number of panels. Thus the number of panels is subtracted by a maximum value of 7000 and the difference divided always by 7000; this ratio is summed by the ratio between the number of panels with square shape and an edge-length of 1,5m and the total number of panels. These two ratios are summed together to define the objective and Galapagos research the maximum.

The Galapagos solver was set with a population of 20 genes and let run until it converges to a solution at the 82nd generation.

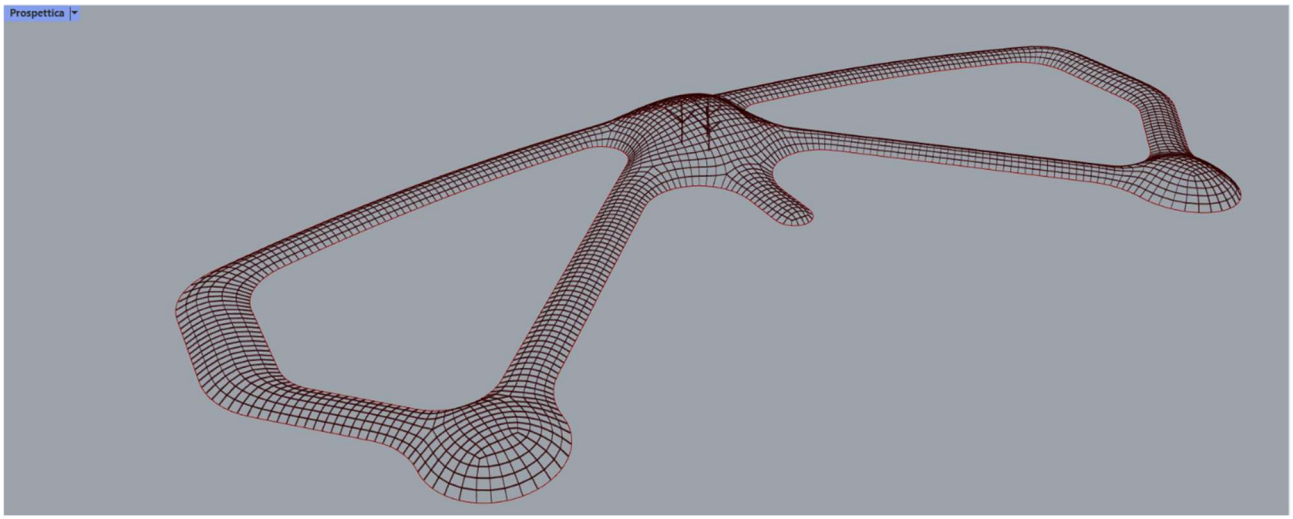


Fig. 4.1 Prospective view of the structure one with pillars

The result of the analysis is a structure with a quite good panels ratio, the 42,1% of panels have a squared shape with an edge-length of 1,5m mainly disposed over the corridors, where the roof have a more regular shape. Furthermore, the elements with a length similar to 1,5 m are 59,6 % of the total; this is a good result because the element has a quite constant length and the biggest number of panels can be cut with an edge-length multiple of the glass-sheets that have a dimension of 6,0x30 m.

The number of panels is 2761, that is the 39,4% of 7000, thus it is quite limited and takes the structure to have a weight of 851,4kN composed of 483,1 kN of steel and 368,3 kN of glass. The structure has a maximum height of 9,18m.

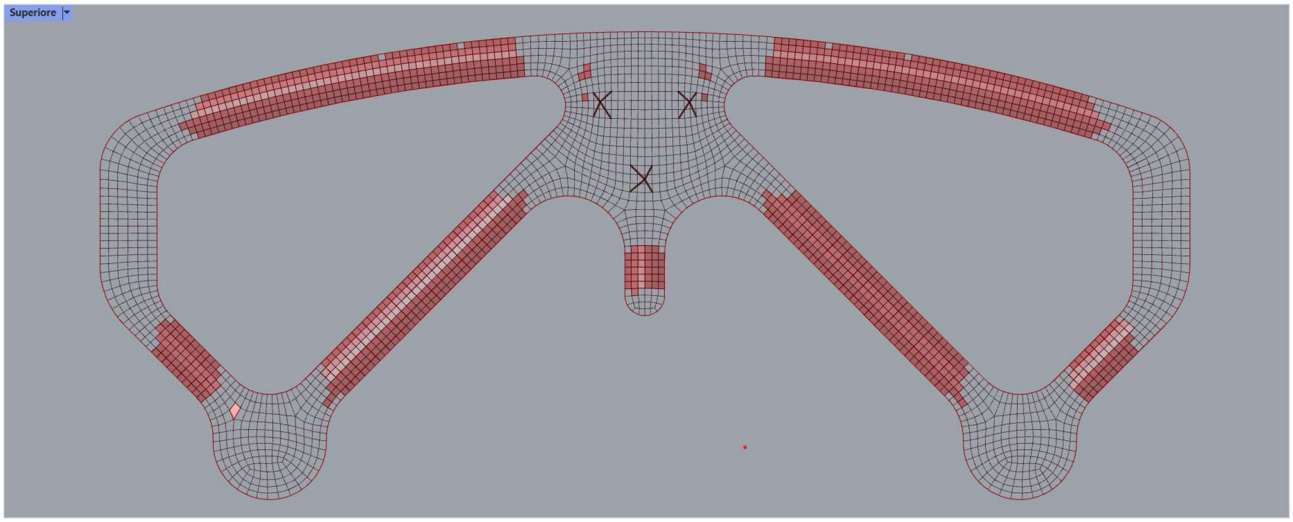


Fig. 4.2 Square panels disposition on the structure one

About the structural analysis, the maximum utilization is 21,8% which is a low value, usually, an element can reach an utilization of 90%; while the maximum displacement value is 8,28 mm, less than the 1,0 ‰ of the height. These two values are really good, and represent a structure under-exploited; therefore it is possible to reduce the cross-section of the elements, in such a way the cost of the grid-shell decrease. Another option is to delate the tree-pillars to try to increase the utilization of the structure.

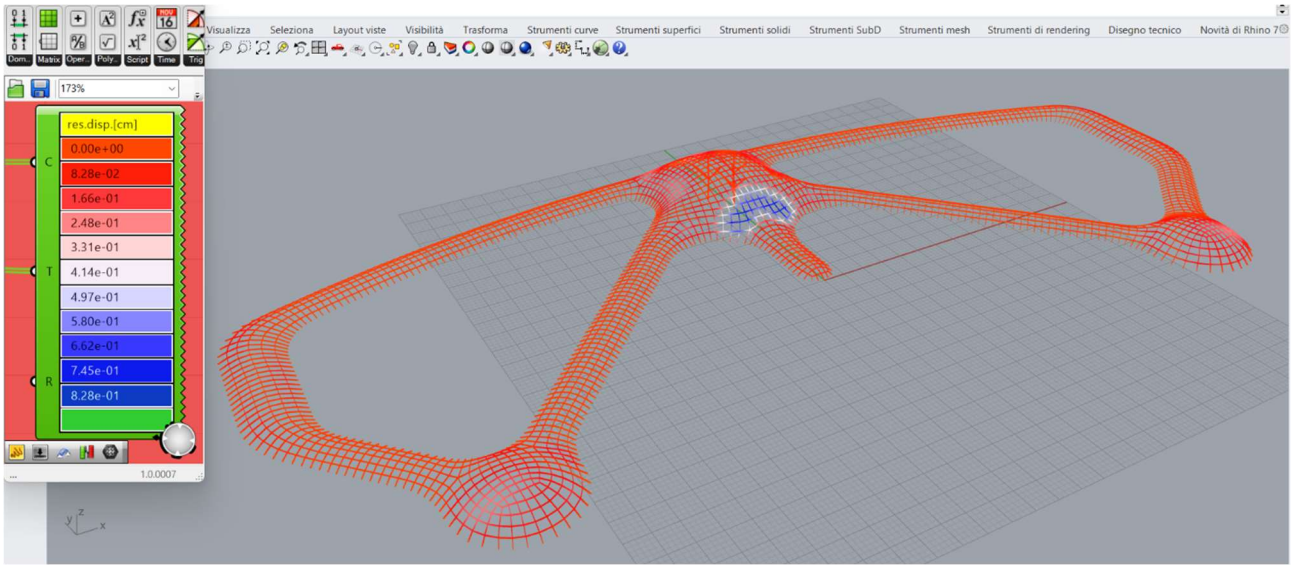


Fig 4.3 Prospective view of the displacement in the structure one

Since the cross-section is a parameter given by the client, in the second analysis the tree-pillars are delated to understand the behavior of the structure.

5.2 Second Analysis

To delate the tree-pillars by the model, the commands *LineToBeam* and *Support* concerning the pillars are disabled and a new analysis, with the same settings of the first is run.

The results are amazing, the solvers return the same structure defined with the tree-pillars; also the

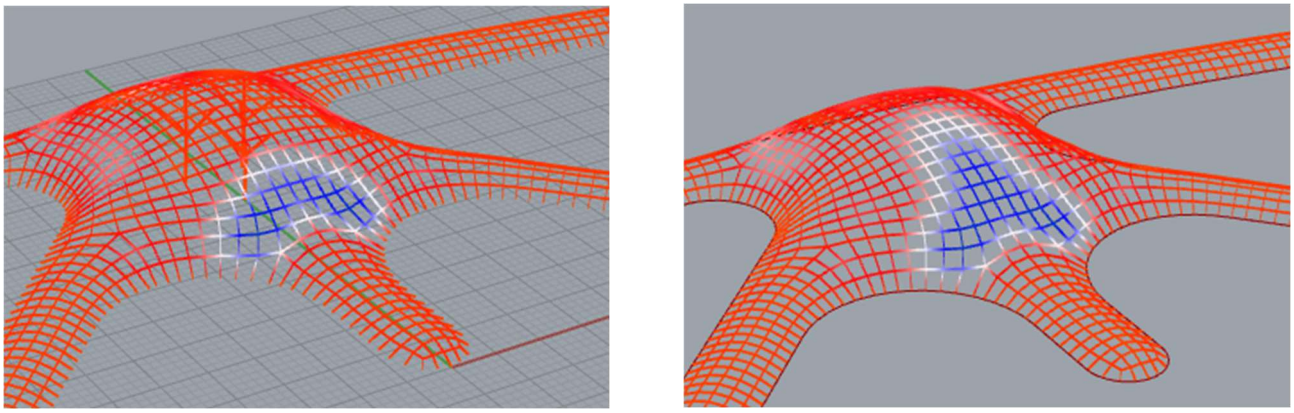


Fig. 4.4 Different displacement in the structure one with and without pillars

data of the structural analysis is the same. It is only possible to notice that the displacement has a slightly different value, from 8,28 mm to 8,235 mm, and a different configuration on the main hall; as it is possible to see in these images.

This behavior can be obtained for two different reasons. The first is that the cross-section is under-exploited, and so the structure is so stiff, that the support given by the tree-pillars is irrelevant. Therefore the behavior of the structure, with or without pillars, will change if a minor cross-section is used.

The other explanation is proper of the form-finding: the shape of the structure is defined without the tree-pillars to work only in compression, therefore the statical scheme of the structure is already hyperstatic. The addition of the pillars slightly increases the degree of constraints of the structure which is already more than the degree of freedom; so they affect the behavior of the structure only locally, and not globally. It is the same behavior of a hyperstatic beam with more DoC than DoF, if a constrain is added it affects the behavior of the beam locally and not globally.

To understand which the behavior of the structure is, are performed two analyses where the cross-sections are incremented from the smallest to the biggest, in the first one the tree-pillars are considered, in the second one not.

5.3 Third and Fourth Analyses

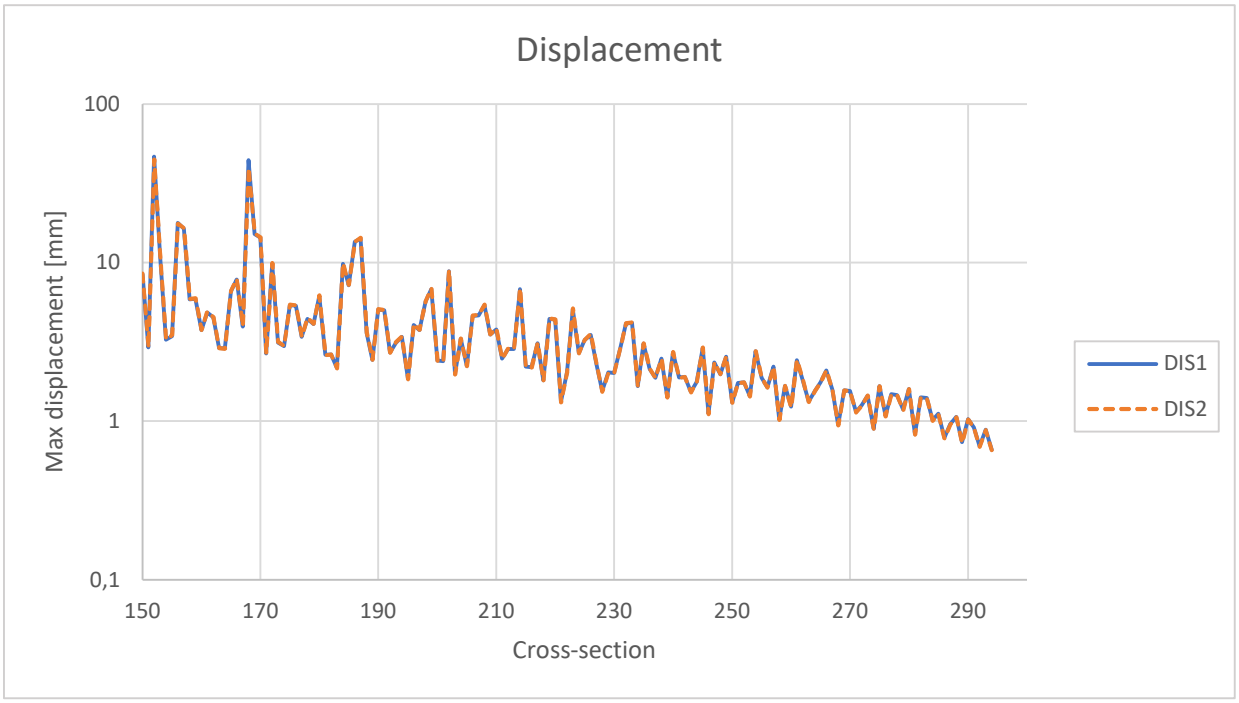
As said before, these analyses are performed to understand the importance of the tree-pillar in the static scheme through the cross-section increment in two different configurations, with or without pillars.

In these analyses, the structure configuration is not changed, only the cross-section, in such a way all the different parameters of the structural analyses are due to the different cross-section. Obviously, the used structure is the best-one defined in the previous analyses; and the cross-section is incremented from the smallest to the biggest with an “animate” number slider that changes the position in the cross-section list; indeed, the cross-section used in the model arrives from *CroSecRSelect*, which is a cross-section library.

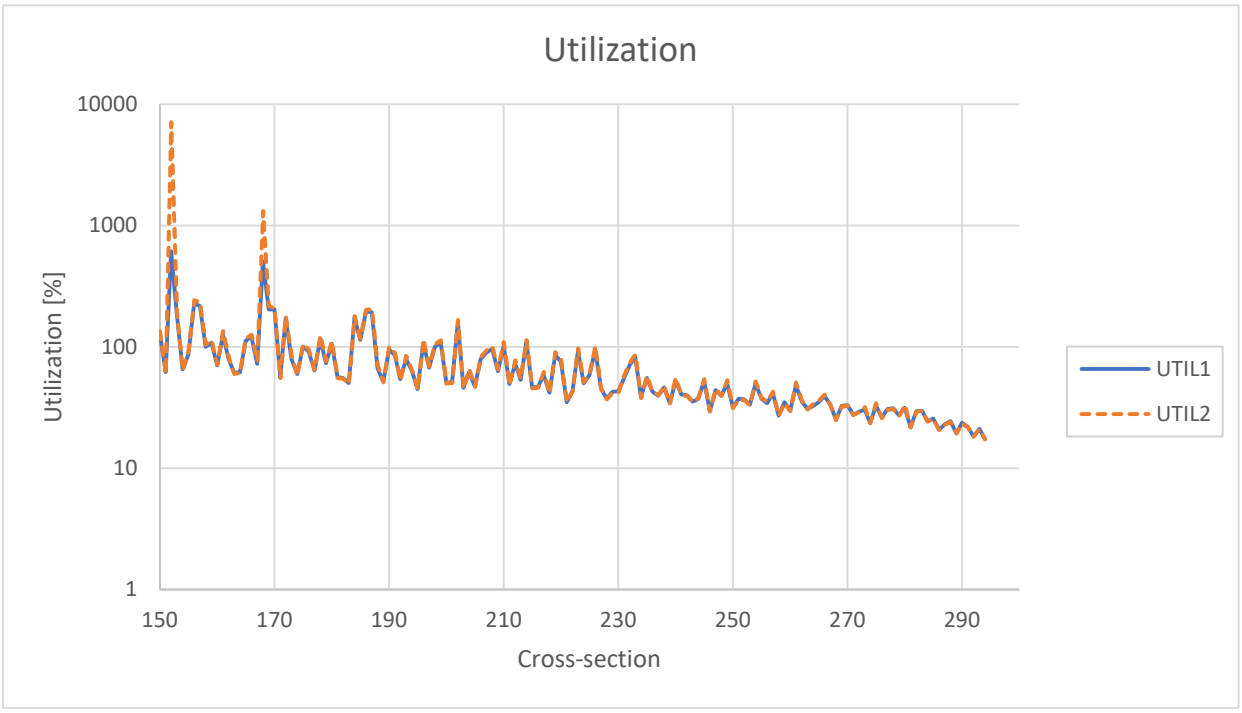
An “animate” number slider is a particular option of the command that allows to automatically increase the value from zero to the end; therefore the part of the algorithm managed by this number slider is run with increasing values. Was chosen to use this command in place of Galapagos because an objective is not searched, but all the values of the structural analyses are sorted following the cross-section.

To collect the data some data recorders are linked with the cross-section position, the maximum utilization of the structure, the displacement, the buckling load factor, and the boolean conditions of utilization and displacement to simplify the data reading.

The results of these analyses are represented in this graph.



Graph 4.1 Third and fourth analyses displacement



Graph 4.2 Third and fourth analyses utilization

It is really difficult to distinguish the two series of data because they are too close; except for the utilization of the cross-sections 152 and 168. For cross-section 152, the utilization is eleven times higher without tree-pillars; while for cross-section 168 it is 2,5 times higher, but it is always over 100%, so the statical verification is not satisfied. Furthermore, these cross-sections fail for buckling because they are thin, the cross-section 152 is an RHS 80x50x11 mm and the 168 is an RHS 80x50x12,5 mm.

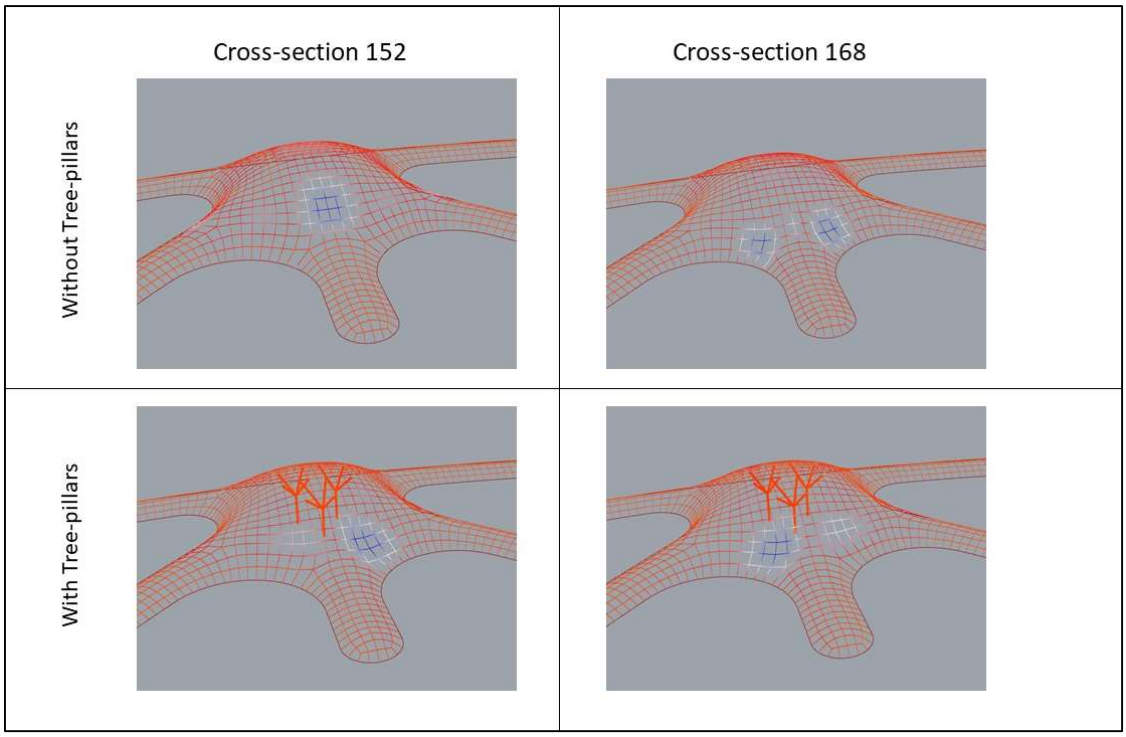


Fig. 4.5 Comparison of the four different configurations

Analyzing the displacement graphs of Karamba3D is easy to understand how the tree-pillars modify the statical scheme of the structure. They reduce the displacement on the center of the hall dome and move it on the side; because the pillars support the dome center but they do not reduce the load on the structure, which fails for buckling on the most stressed parts.

This behavior confirms that the pillars act on the structure, but there are not so useful to reduce the displacement in this configuration.

To find a new configuration of the tree-pillars that allow to obtain a lighter structure a multi-step analysis is run. center

5.4 Fifth Analysis

This analysis uses multi-step optimization because is composed of three different processes of optimization on the same structure. The first is the definition of the structure, as done in the first and second analyses; indeed the structure defined in the first analysis is used. The second step is the optimization of the roof mesh, with the process performed only on the QRMesh command. The third step is the optimization of the tree-pillars configuration and the cross-section, to find the lightest one.

Whereby, in the second step only the QRMesh number-slider's is the gene of Galapagos, while the objective is always the same while the number of panels and squared panel ratio have the same weight. This analysis returns the same number of panels as the first one, therefore the multi-parameters analysis just obtains the best remeshing of the roof shape.

The third step is the optimization of the cross-section and the pillars configuration in such a way that the lightest cross-section, which satisfies the statical verification, is found. To perform it, a new

fitness-function that rewards the cross-section with the smallest area is made; indeed the cross-section list used by Karamba 3D was sorted in area ascending order.

The new fitness function has the number of pillars, the radius, and the cross-section as a parameter; while the objective is the list's positions of the cross-section that should be minimized, to obtain the lightest cross-section. The cross-section position is multiplied by a thousand if the statical verifications are not satisfied; in such a way the objective is too, height and Galapagos exclude these genes from the following analyses. As in the previous analyses, the statical verifications are performed with Karamba3D with the same command; but in this case, the displacement limits are not considered, because the target of this analysis is to understand the function of the pillars inside the statical scheme, not define a new typology of the roof structure.

First of all, this analysis is performed with the tree-pillars, to define the lightest cross-section and the better disposition of the pillars; indeed the results are really interesting. The lightest one that satisfies the statical analyses is the RHS 120x60x8 mm with maximum utilization of 95,2% and a maximum displacement of 50,8 mm (around the 0,5% of the structure height); while the unitary weight is 20,6 kg/m against the 50,9 kg/m of the RHS 250x100x10. The analysis determined five pillars with a radius of 8,0 m.

Also, the pillars should be sized, whereby the number slider that selects the cross-section from the Karamba3D library is animated and the utilization recorded. By the results, a cross-section CHS 76,1x3,2 is found with the utilization of 94,0% and a buckling load factor of 2,2. The images below show the different displacement in the two structures after and before the optimization; it is also possible to see the influences of the tree-pillars disposition.

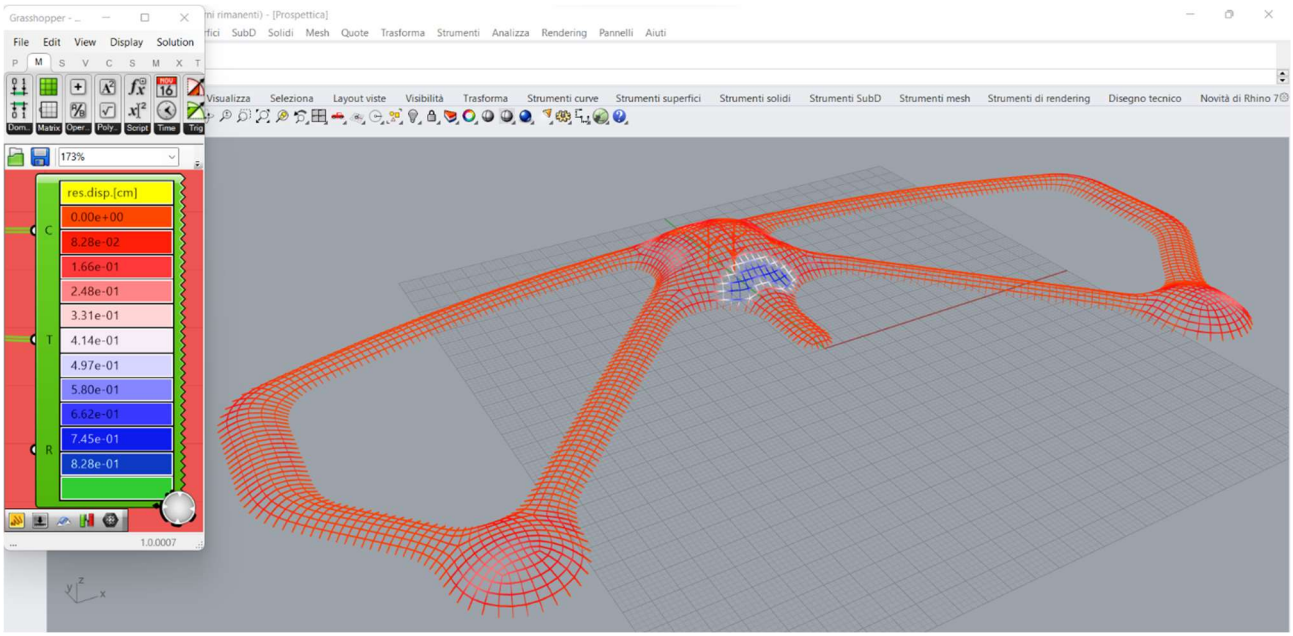


Fig. 4.6 Structure one after the optimization

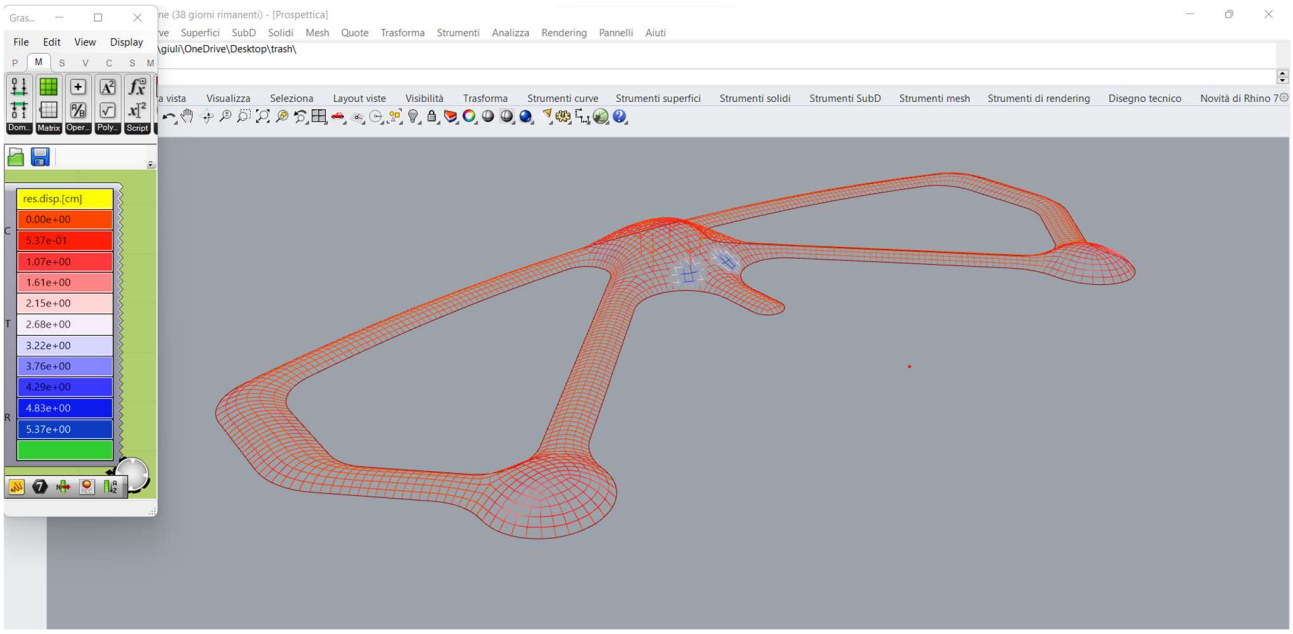


Fig. 4.7 Structure one before the optimization

In the next analysis, the pillars are removed, and the lightest cross-section that satisfies the statical verification is the RHS 120x60x10 mm with maximum utilization of 94,2% , a linear weight of 24,3 kg/m and a maximum displacement of 47,0 mm.

Therefore the results of the two analyses are really close, so the evaluation of the best solution from an economic point of view is not immediate. The structure is the same for the two analyses, with a total edge length of 8475,0 m; while the tree-pillars have a tola length of 116,0 m made with a cross-section CHS 76,1x3,2 mm that has a linear weight of 5,8 kg/m . The total steel weight of the two structures is:

$$With\ pillars:\ \ 8475{,}0 * 20{,}6 + 116{,}0 * 5{,}8 = 175258\ kg$$

$$Without\ pillars:\ \ 8475{,}0 * 24{,}3 = 205942\ kg$$

With the tree-pillars, the steel saving is 30684 kg, which is 15% of the weight. This result is really interesting because the use of the pillars allows the saving of 30684 kg of steel, which corresponds to 148817 euro.

About the first cross-section used, the RHS 250x100x10 the steel saving is obvious, and around 60%.

5.5 Sixth Analysis

This analysis has the goal to define the structure with a smaller number of panes, that can be matched to the lightest structure because the panel's area and the total edge length will be minimized. Therefore the fitness function is the one used in the first and second analyses, with several panels' weight of 90% and a square panels ratio's weight of 10%. But to determine the lightest structure also the lightest cross-section should be evaluated, thus a second analysis will perform if the RHS 120x60x8 a the RHS 120x60x10 mm do not satisfy the static verification.

The structure defined by this analysis has a shape not so different from the previous one, indeed the kangaroo weighting is the same and the strength is 9,2 N/m in place of 6,4 N/m; but the new mesh is wider and no one panels have the square dimension researched. Indeed, the most common panel shape is a square with an edge-length of 2,0m. In the following images, it is possible to see the two different structures, the first on the right and the second on the left.

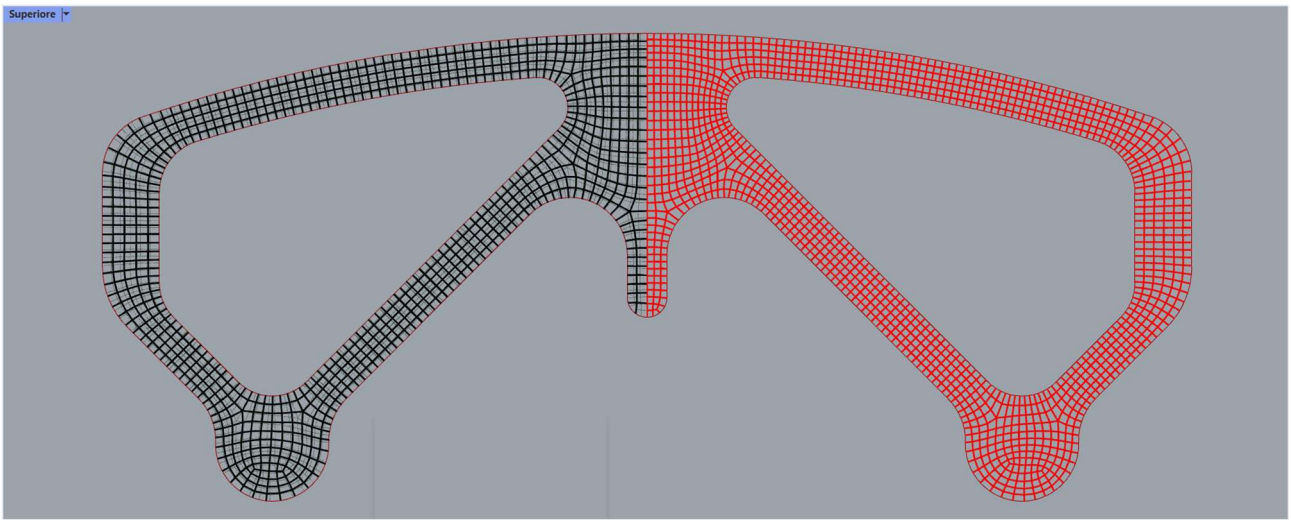


Fig 4.8 Top view of the second (right) and third (left) structures

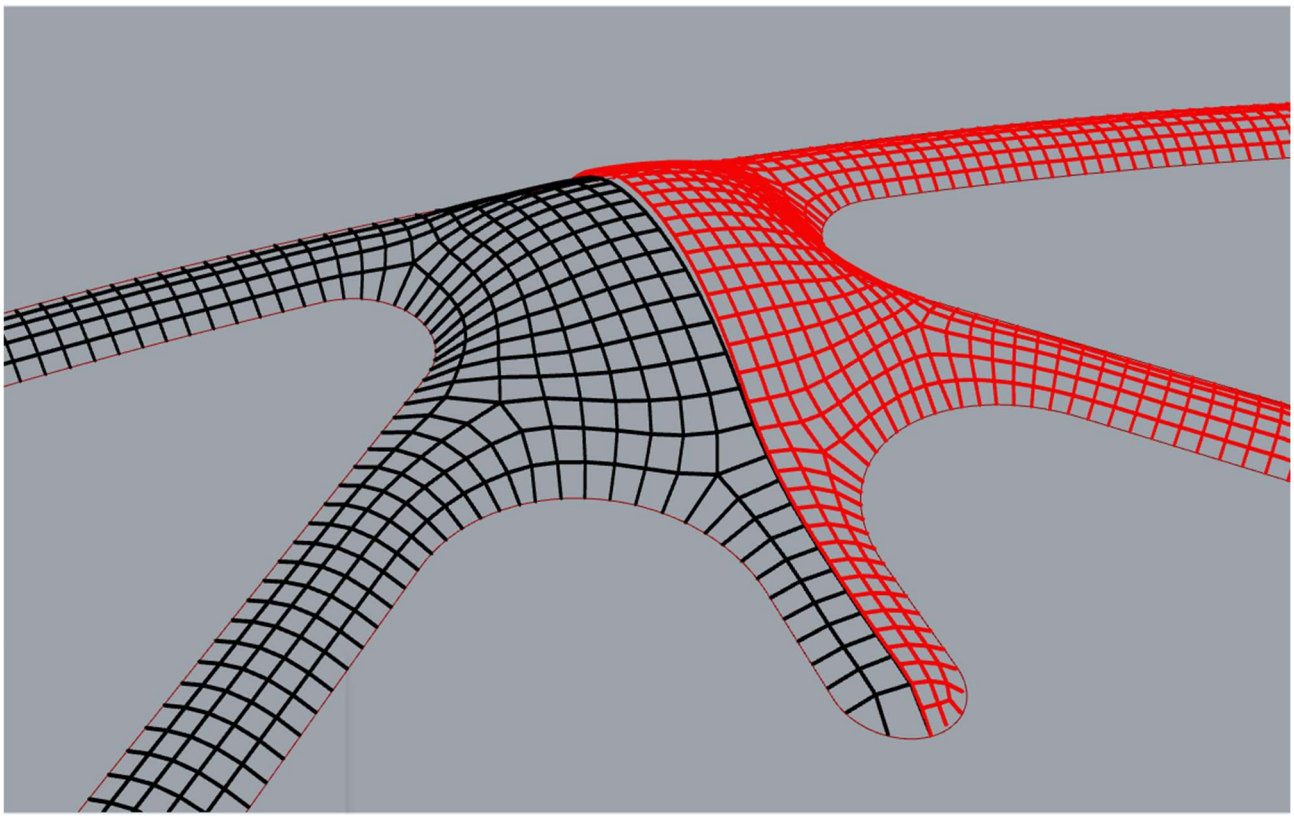


Fig 4.9 Prospective focus on the main hall with the second (right) and third (left) structures

By the images it is possible to see the different number of panels for the two structures and the little shape difference, that is focused on the rise of the main dome which is 8,0 m in place of 10,0 m.

The problem with the new structure is that do not satisfy the statical verification with the predefined section; so new cross-section research should be performed, with the same algorithm used in the previous one.

The algorithm to research the lightest cross-section is the same as the previous analysis. For the case without tree-pillars, it defines a cross-section RHS 120x80x10 with maximum utilization of 95,5% and a maximum displacement of 37,8 mm; while with the pillars the cross-section is an RHS 120x80x8 with maximum utilization of 91,4% and a maximum displacement of 39,9 mm. The pillars can have a cross-section CHS 76,1x4,0 with a unitary weight of 7,1 kg/m

Like in the previous analysis, the total weight of the two structures is evaluated:

$$With\ pillars:\ \ 6644 * 22{,}6 + 125{,}6 * 7{,}1 = 151046\ kg$$

$$Without\ pillars:\ \ 6644{,}0 * 27{,}4 = 182046\ kg$$

Also for this structure, the version with the pillars is lighter than the version without; with a difference of 17% . Also for this structure the pillars give an important steel saving; considering the unitary cost of the metallic carpentry is 4,85 euro/kg, the cost difference is around 150350 euro.

These structures are lighter than the previous ones; also because the area of these roof structures is 7203 m^2 in place of 7366 m^2 for the first ones, this means a lower glass weight. The weight difference between the two structures is:

		STRUCTURE 1	STRUCTURE 2	DIFFERENCE
WITH PILLARS	GRID WEIGHT	175258	151046	24212
	GLASS WEIGHT	368300	360150	8150
WITHOUT PILLARS	GRID WEIGHT	205942	182046	23896
	GLASS WEIGHT	368300	360150	8150

Table 4.1 Weight composition and saving structure 1 and 2

Only with the definition of a new shape, optimized to reduce the number of panels, it is possible to reduce the steel weight around 12% independently using the tree-pillars; these allow an interesting money-saving, that is discussed in the 4.6 paragraph.

These structures are not found with a weight analysis, but with a geometric one; whereby a new analysis where the structure weight is the objective will be performed.

5.6 Seventh Analysis

The goal of these analyses is to define the lightest structure with and without tree-pillars; therefore a new fitness function should be defined where the objective is the total weight of the structure and all the parameters are considered. Indeed, not only the shape of the structure should be modified, but also the cross-section; so the Galapagos genes are:

- kangaroo weighting;
- kangaroo strength;
- number of panels;
- tree-pillars radius;
- tree-pillars number;
- cross-section dimension.

As said before, the objective is the minimum weight, which is evaluated as a sum between the weight of the grid-shell weight, the tree-pillars, and the glass panels. The first one is evaluated as the product between the cross-section area, the total edge length, and the steel density of 7860 kg/m^3. The same is done for the tree-pillars but considering a fixed cross-section CHS 406,4x25 that weights 235,1 kg/m; this cross-section will be optimized in a second step, to reduce the complexity of the fitness function. The glass weight is evaluated as the product between the area and a weight of 50 kg/m^2 considering a glass thickness of 20 mm.

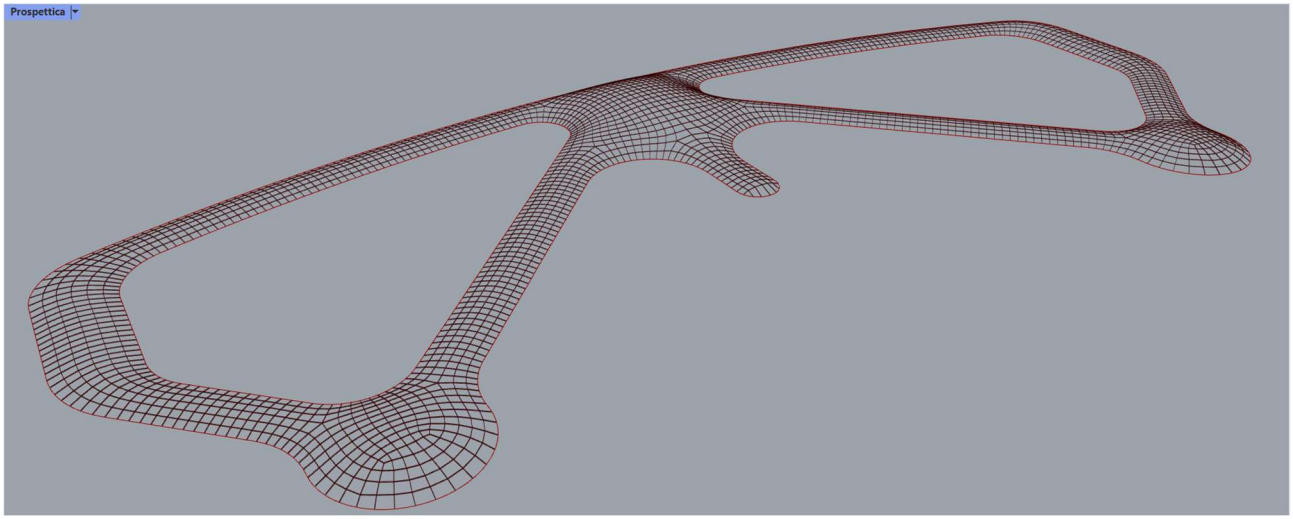


Fig 4.10 Prospective of the structure four without pillars

The new structures found with these analyses are really different from the previous one because they have a maximum height between 4,0 m and 4,87 m; therefore the roof is flatter. This shape is not so convenient for the grid-shell, because it has a bigger axial load than a curved one, but it is more convenient for the glass weight because a flat structure has a lower surface than a curved one; these two opposite mechanisms find an equilibrium in this new shape. Indeed, the new cross-section is an RHS 140x80x5 for the structure with the tree pillars and an RHS 160x80x4 for the structure without them, which are heavier than in the previous structure. The number of tree-pillars is three, with a radius of 9,0 m and a cross-section CHS 76,1x2,6 obtained with the second optimization; also these elements allow a weight reduction. In this table it is possible to confront

the grid-weight of this structure with the grid-weight of the structure two and three, the structure one is not considered because is the non-optimized version of structure two

STRUCTURE	CROSS-SECTION	CS WEIGHT (kg/m)	EDGE-LENGTH (m)	GRID WEIGHT (kg)	DIFFERENCE FROM THE NEW STRUCTURE (kg)	DIFFERENCE FROM THE NEW STRUCTURE (%)
2 TREES	120x60x8	20,6	8475,5	174595	-38083	21,8
2 NO TREES	120x60x10	24,3	8475,5	205955	-86011	41,7
3 TREES	120x80x8	22,6	6644	150154	-13642	9,0
3 NO TREES	120x80x10	27,5	6644	182710	-62766	34,3
4 TREE	140x80x5	16,3	8375	136512	0	0,0
4 NO TREES	160x80x4	14,5	8272	119944	0	0,0

Table 4.2 Steel weight composition and saving of structures 1,2 and 3

Although the cross-sections of the new structures have bigger external dimensions, their thickness is lower, this allows to hold the bigger axial load without a buckling failure and in the meantime to have lighter structures; indeed the edge-length of the new structures is quite big.

The other weight component is the weight of the glass panels, which is considered 50 kg/m^2 considering a glass-thickness of 20 mm, and a less surface takes to a less weight; as it is possible to see in this table.

STRUCTURE	SURFACE (m^2)	GLASS-WEIGHT (kg)	DIFFERENCE FROM THE NEW STRUCTURE (kg)	DIFFERENCE FROM THE NEW STRUCTURE (%)
2	7366	368300	-26650	7,2%
3	7203	360150	-18500	5,1%
4	6833	341650	0	0

Table 4.3 Glass weight composition and saving of structures 1,2 and 3

The surface reduction is not so big, but it helps to decrease the total weight of the structures, also because low permanent load means a lighter cross-section. It is possible to see this effect in the following table:

STRUCTURE	GRID WEIGHT (kg)	TREE WEIGHT (kg)	GLASS WEIGHT (kg)	TOTAL WEIGHT (kg)	DIFFERENCE FROM THE NEW STRUCTURE (kg)	DIFFERENCE FROM THE NEW STRUCTURE (%)
2 TREES	174595	673	368300	543569	65133	12,0
2 NO TREES	205955	0	368300	574255	112661	19,6
3 TREES	150154	892	360150	511196	32760	6,4
3 NO TREES	182710	0	360150	542860	81266	15,0
4 TREE	136512	274	341650	478436	0	0
4 NO TREES	119944	0	341650	461594	0	0

Table 4.4 Weight composition and saving of structures 1,2 and 3

From the table, it is possible to say that the new structures are lighter than the previous one, with a really interesting reduction until 19,6%. Whereby it is possible to say that the new shapes are lighter although they have a flat shape.

A reflection about the tree-pillars is mandatory; although the cross-section reduction due to their presence appears not so significant, only some millimeters of thickness, the total weight of the structure drastically change, with a reduction of around 5% between structures with and without pillars.

5.7 Eighth Analysis

A fundamental criterion to select the structures is their cost; therefore a new typology of analysis with the minimum cost as the objective is performed.

To define the cost of a structure it is necessary to know the unitary cost of its component, labor included, that can be found in the pricing for the public work edited by each Italian region. For these analyses is used the unitary price find in the pricing of the Piemonte region, where the steel carpentry is 4,85 euro/kg while the glass panels are 100, 00 euro/m^2 which should d be included a labor cost of 40 euro/m^2. These two values are used to define the final cost of the structure.

To obtain the cheapest structure a new fitness-function should be defined; this function defines the total steel weight, as in the previous one, and the panels' area as the sum of the area of each panel and it apply the unitary cost to define the total cost. The cheapest panel to make is each one with a squared shape and edge-length of 1,5 m, as explained before; to consider the cost increasing for different shape panels their area is increasing by 50%, this value is indicative and can be easily changed. In such a way the cost of the different panels is increasing because it depends on their area. About the steel, its weight is defined with the same algorithm defined in the previous analysis and multiplied for the unitary cost. The Galapagos solver was run whit the cost thus defined and with the same six genes of the previous analysis.

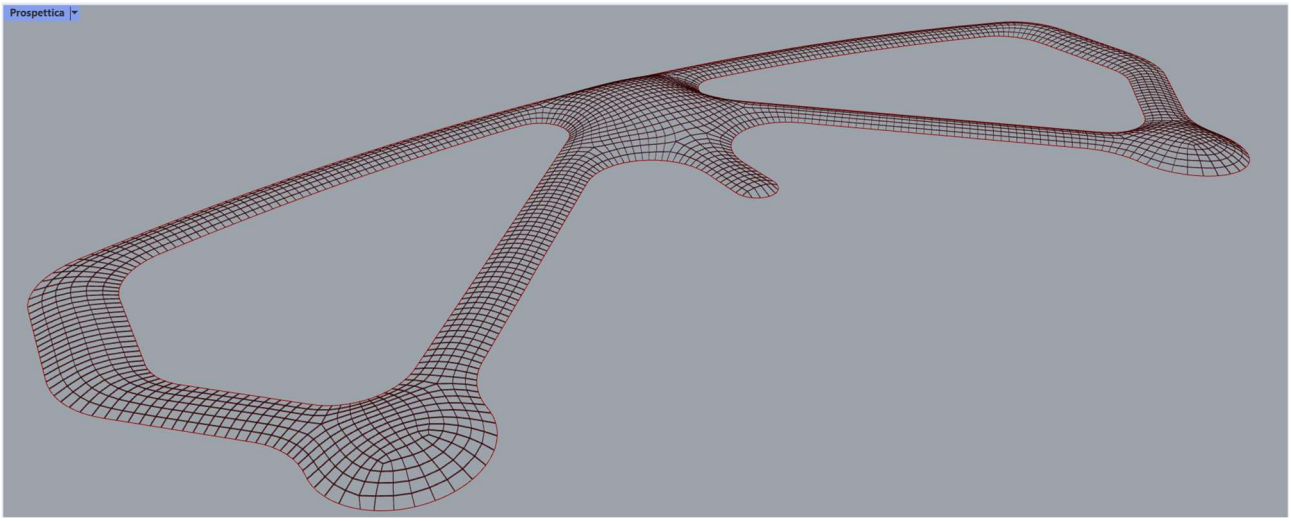


Fig 4.11 Prospective of the structure five without pillars

The results of these analyses are two structures similar to the minimum weight structures, therefore a direct relationship between the weight and the cost is not so wrong, but with some differences in the grid-shell. Whereby the Kangaroo2 parameters are slightly different, the structures have a

similar shape, with a height between 4,0m and 5,0 m and several panels around 2900; also the roof area of the four structures is similar and around 6900 m^2 . About the cost of the structures, it is possible to see a big difference between the first four structures and the last four; but not a big difference between the research of the structure with the minimum weight and the research of the structure with the minimum cost, as it is possible to see in this table.

STRUCTURE	TOTAL WEIGHT (kg)	TOTAL COST (€)	PANELS RATIO (%)	WEIGHT REDUCTION (kg)	MONEY-SAVING (€)	MONEY-SAVING (%)
1 TREES	817531	3536208	42,1	-365717	-1745194	-49,35%
1 NO TREES	799703	3449744	42,1	-333427	-1593591	-46,19%
2 TREES	543569	2215499	42,1	-93405	-424484	-19,16%
2 NO TREES	574255	2356320	42,1	-107979	-500166	-21,23%
3 TREE	511196	2244994	0	-59382	-453979	-20,22%
3 NO TREES	542860	2398564	0	-76584	-542410	-22,61%
4 TREES	478436	1938393	37,4	-26622	-147379	-7,60%
4 NO TREES	464844	1853208	37,4	1432	2945	0,16%
5 TREES	451814	1791015	39,8	0	0	0,00%
5 NO TREES	466276	1856154	39,8	0	0	0,00%

Table 4.5 Weight and cost comparison of the structures

The cost reported in this table is evaluated with the money increasing due to the different panels. It is quite interesting to see how the different characteristics affect the cost of the structures. The first four structures were defined with the objective to the research the maximum number of similar panels and the minimum total number of panels; the objective was reached, but the cost is quite expensive, so it is not possible to define a relation between the objective and the cost. The first two structures are a particular case because they have a pre-defined cross-section RHS 250x100x10, that is oversized for the load, so it is not optimized.

The fourth and fifth structures are defined with as objective the minimum number of panels, without constrain about the dimension, therefore the biggest part of the panel have an edge-length of 2,0 m which produce a lot of glass waste; for this reason, the cost of these structures are so hight.

About the sixth and seventh structures, the reasoning is always the same, less weight meaning less cost. Whereby in the low weight objective the number of similar panels was not considered, this number is quite high, between the 37% and the 40% , and this factor help in the cost reduction. Indeed the structure find with the lowest cost objective, where the panel's ratio is considered with the cost increment for the different panels, the number of similar panels is quite the same. Therefore the cost of a structure depends on its weight and the number of similar panels.

Another value that is unexpectedly minimized is the weight of the structure with pillars, a low weight was expected, but not the minimum, which should be found in the minimum weight research in paragraph 4.5. The reason because the weight is the minimum is due to the shape of the roof, because this new shape is more curved than the previous one, so the structural elements are less stressed and so the cross-section can be lighter; a cross-section RHS 140x70x4 mm, in place of an RHS 140x80x5 mm, is used. While for the structures without pillars the cheapest is not the lights but

it is really close; indeed the weight difference is 1432 kg which corresponds to 0,3% of the total weight. Therefore these structures have an edge-length and a roof area close.

In the last four structures, the failure is due to the buckling instability and not for the stress; indeed the buckling load factor is close to 2 and the utilization is around 60%. Therefore, with an optimized cross-section, different from the commercial ones, it is possible to obtain a lighter structure.

5.8 Ninth Analysis

In the last years, a new parameter to select goods is made, the Global Warming Potential (GWP); This parameter expresses the quantity of CO_2 , in kilograms, made for the production of each good. Also, the structural elements are evaluated; for the steel hollow profile, it is expressed in kg/kg, while for the glass panels in kg/m^2. This parameter is proper of each element and it is evaluated with an analysis of the production cycle reported in an official document called Environmental Product Declaration. This document considers only the CO_2 made to produce the elements in the factory, and not its transport to the building site and the assembly because this parameter depends on many factors like the distance from the factory and the assembly technology. Therefore the CO_2 quantity reported in this thesis is only the one to produce the elements, to which it is necessary to add the ones for the transport and the assembly.

The roof is made mainly by two typologies of elements, the steel profiles, and the glass panels. For these elements is really easy to find the GWP because the main factory made it; therefore it is possible to consider a GWP for the steel hollow profile of 2,27 kg/kg , while for the glass panels it is 73 kg/m^2 considering panels with a thickness of 20 mm, so 1,46 kg/kg.

To research the two structures, with and without pillars, with the lowest GWP a new fitness-function which minimizes the $CO_{2eq.}$ produced for each structure is coded. In this new algorithm, the weight of the steel and the glass is defined, as in the previous analyses, and it is multiplied for the corresponding GWP; if the structure does not satisfy the utilization limit or the buckling load, the $CO_{2eq.}$ is multiplied for thousand. As for the cost evaluation, the glass waste due to the out-of-size panels is considered, considering an increase of the roof area of the 50% of the out-of-size panels. The logic behind the fitness-function is the same in all the analyses, only the objective change.

The results of these analyses are two structures close to the minimum weight structure, therefore these two structures are not added to the list of the structure, the reason is quite clear being the GWP directly proportional to the weight of the materials. As in the previous cases, a table to summarize the cases is reported.

STRUCTURE	STEEL WEIGHT (kg)	GLASS WEIGHT (kg)	ADD GLASS WEIGHT (kg)	$CO_{2eq.}$ (kg)	SAVING OF $CO_{2eq.}$ (kg)	SAVING OF $CO_{2eq.}$ (%)
1 TREES	449231	368300	116500	1727561	-821061	-47,53%
1 NO TREES	431403	368300	116500	1687093	-751834	-44,56%
2 TREES	176919	368300	116500	1109415	-202914	-18,29%
2 NO TREES	205955	368300	116500	1175325	-240066	-20,43%
3 TREE	151046	360150	180000	1131494	-224993	-19,88%
3 NO TREES	182710	360150	180000	1203371	-268112	-22,28%
4 TREES	136786	341650	113700	975316	-68815	-7,06%
4 NO TREES	119944	344900	109200	935259	0	0,00%
5 TREES	105764	346050	110400	906501	0	0,00%
5 NO TREES	120176	346100	108650	936735	-1476	-0,16%

Table 4.6 Weigh and GWP comparison of the structures

By this table is possible to say that there is a linear correlation between the weight, the cost, and the GWP due to the construction material whereby the lightest structure is also the cheapest and the less polluting.

5.9 Structure Overview

It is now possible to do an overview of all the structures defined by the analyses, in such a way it is possible to compare all of them for their characteristics and understand the effects of the parameters on them. The next table summarizes the input, the characteristics, and the objectives of the structures defined in the analysis.

STRUCTURE	Input Parameters	Cross-section optimization	Pillars	Objective
1-TREE	Kangaroo2 weighting, Kangaroo2 strength, number of panels, number of pillars, and radius	NO	YES	Minimum number of panels, the maximum number of square panels with 1,5 m edge-length
1-NO TREE		NO	NO	
2-TREE	The shape of structure 1, number of pillars, and radius	YES	YES	Define the lightest structure with the shape 1
2-NO TREE		YES	NO	
3-TREE	Kangaroo2 weighting, Kangaroo2 strength, number of panels, number of pillars, and radius	YES	YES	Minimum number of panels
3-NO TREE		YES	NO	
4-TREE	Kangaroo2 weighting, Kangaroo2 strength, number of panels, number of pillars, and radius	YES	YES	Minimum structure's weight
4-NO TREE		YES	NO	
5-TREE	Kangaroo2 weighting, Kangaroo2 strength, number of panels, number of pillars, and radius	YES	YES	The less expensive structure
5-NO TREE		YES	NO	

Table 4.7 Overview of the analyzed structures

First of are showed the parameters, or better, the genes of the structures; these values have a physical meaning but are not possible to compare the structures with these values because the roofs depend on too many factors.

STRUCTURE	Kangaroo2 Weighting	Kangaroo2 Strenght	Number-slider	Number of Pillars	Radius of Pillars	GRID		TREE	
						CROSS-SECTION (RHS)	C-S WEIGHT (kg/m)	CROSS-SECTION (CHS)	C-S WEIGHT (kg/m)
1-TREE	0,4	6,4	1633	3	4	250x100x10	50,9	406,4X25	235,1
1-NO TREE	0,4	6,4	1633	-	-	250x100x11	50,9	-	0
2-TREE	0,4	6,4	1633	5	8	120x60x8	20,6	139,7X12,5	20
2-NO TREE	0,4	6,4	1633	-	-	120x60x10	24,3	-	0
3-TREE	0,4	9,2	500	5	8	120x80x8	22,6	76,1X4,0	7,1
3-NO TREE	0,4	9,2	500	-	-	120x80x10	27,5	-	0
4-TREE	0,1	15,8	1794	3	9	140x80x5	16,3	76,1x2,6	5,4
4-NO TREE	0,2	18,4	1564	-	-	160x80x4	14,5	-	0
5-TREE	0,2	15,8	1797	3	10	140x70x4	12,4	76,1x2,6	5,4
5-NO TREE	0,2	15,7	1570	-	-	160x80x4	14,5	-	0

Table 4.8 Algorithm parameters of the structures

5.9.1 Geometric Parameters

Some more interesting parameters are the geometric and tensional ones, which are reported in the following table.

STRUCTURE	PANEL-RATIO (%)	NUMBER OF PANELS	EDGE-RATIO (%)	MAX HEIGHT (m)	MAX UTILI-ZATION (%)	MAX DISPLA-CEMENT (mm)	ROOF-AREA (mq)	WASTE FOR OUT OF SIZE PANELS (mq)	TOTAL EDGE-LENGHT	TREE EDGE-LENGHT
1-TREE	42,1	2761	59,6	9,18	21,8	8,28	7366	2330	8475,5	75,83
1-NO TREE	42,1	2761	59,6	9,18	21,8	8,235	7366	2330	8475,5	0,0
2-TREE	42,1	2761	59,6	9,18	99,5	53,7	7366	2330	8475,5	116,2
2-NO TREE	42,1	2761	59,6	9,18	94,2	47	7366	2330	8475,5	0,0
3-TREE	0,0	1799	0,7	8,00	91,4	40	7203	3600	6644	125,6
3-NO TREE	0,0	1799	0,7	8,00	95,5	37,8	7203	3600	6644	0,0
4-TREE	37,4	2915	59,05	4,03	94,7	37,2	6833	2274	8375	50,7
4-NO TREE	40,9	2833	57,8	4,87	66	26,2	6898	2184	8272	0,0
5-TREE	39,8	2963	58,6	5,14	83,7	41,7	6921	2208	8507	51,3
5-NO TREE	41,5	2833	57,7	5,15	64,7	25,3	6922	2173	8288	0,0

Table 4.9 Geometric and structural parameters of the structures

With these parameters it is possible to compare the structures; remember that the first and the second structure have the same grid, but the first is made with a given cross-section, the second with an optimized cross-section.

Starting with the panel ratio and the edge-length ratio, it is possible to see how structures one and two have the biggest value of these proportional parameters; while the structure tree has this ratio practically nil. This is because structures one and two have the panel ratio as objective, while structure three has the lowest number of panels, which is respected. About structures four and five, it is possible to say that the number and the typology of panels do not affect the weight, the cost, and the $CO_{2eq.}$, but they depend on other factors.

About the maximum utilization of the elements, it is possible to say that it is too low for the structures one because it is oversized; the other structures have a correct utilization because the cross-sections are optimized. Also for structures four and five, the cross-sections are optimized, but the failure is done for the buckling; therefore the optimization should be done on the cross-section shape over than on the dimension.

The displacement depends on many factors, first of all, the cross-section; indeed in structure one, it is negligible. Also in structures two and three where it is around 0,5% of the height. This parameter grows around 1,0% for structures four and five because it is flatter than the previous one, so the roof's curvature affects the total displacement. Another factor that influences this parameter is the tree-pillars, for structures four and five it reduces the displacement of the 3therefore the presence of the pillars alloallowsmake the lighter structure.

The roof area logically depends on the roof's curvature, but the panel waste hardly depends on the panel ratio over the roof area; structures two and four have a roof area difference of 500 m^2, but a waste area close.

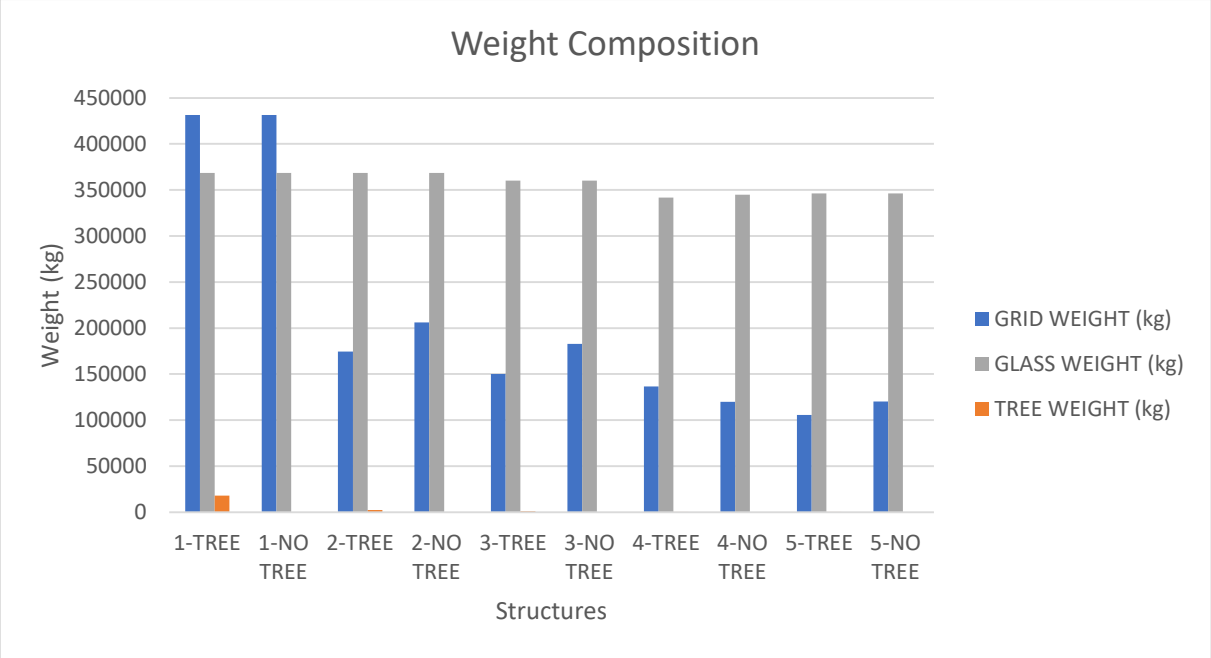
It is not possible to find a relation between the parameters on the table and the total edge length, probably it depends on the mesh dimension which is a parameter that is not possible to control in the *QRMesh* command. The tree edge length depends on their position and number, the more the radius is bigger their height is lower.

5.9.2 Weight Composition

Over the geometric and statical parameters, a structure typology was defined to have the lightest weight; therefore the structures should be compared for their weight, as reported in the table below and the relative bar chart.

STRUCTURE	GRID WEIGHT (kg)	TREE WEIGHT (kg)	GLASS WEIGHT (kg)	TOTAL WEIGHT (kg)
1-TREE	431403	17828	368300	817531
1-NO TREE	431403	0	368300	799703
2-TREE	174595	2324	368300	545219
2-NO TREE	205955	0	368300	574255
3-TREE	150154	892	360150	511196
3-NO TREE	182710	0	360150	542860
4-TREE	136513	274	341650	478436
4-NO TREE	119944	0	344900	464844
5-TREE	105487	277	346050	451814
5-NO TREE	120176	0	346100	466276

Table 4.10 Weight composition of the structures



Graph 4.3 Weight composition of the structures

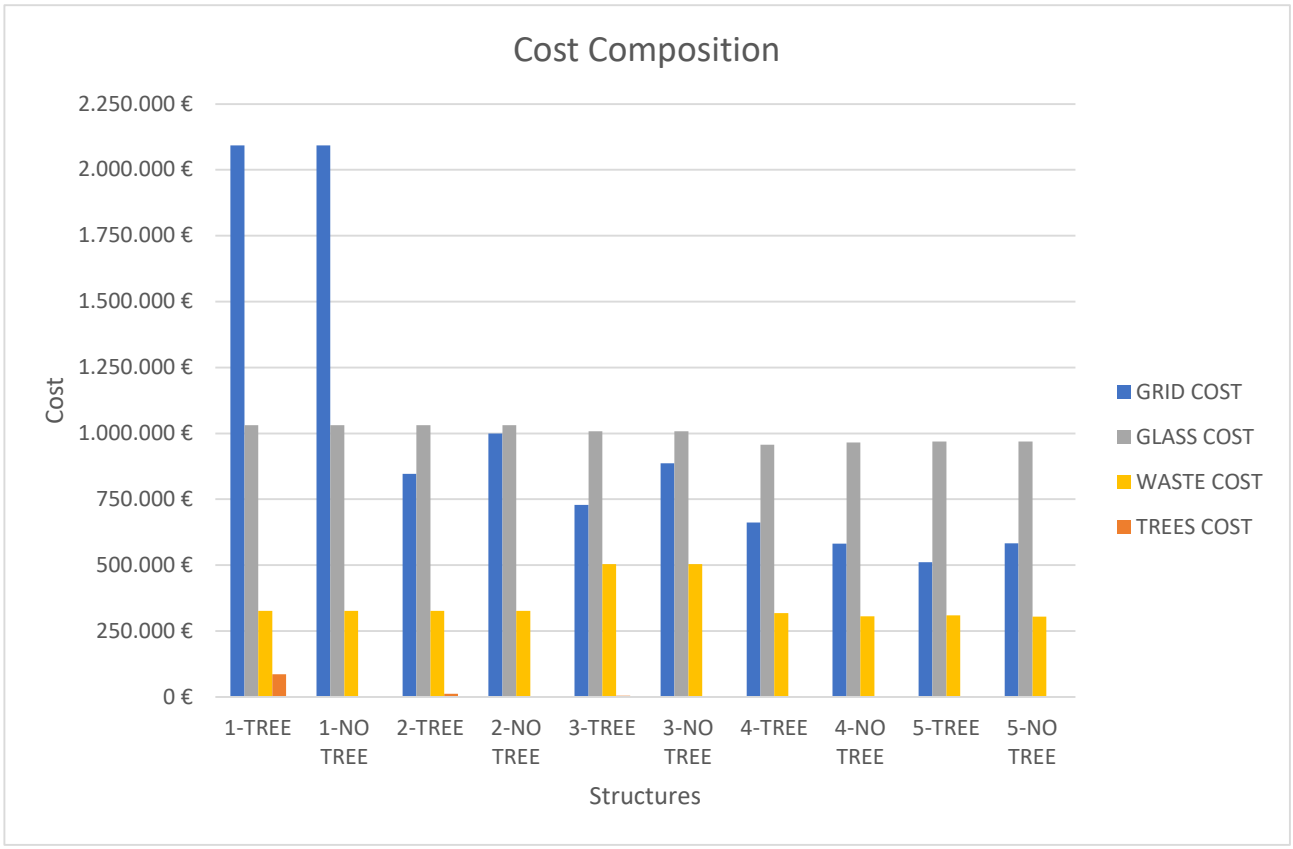
The first thing that appears is the huge weight of the grid in the structures one, disproportionate respect to the other structures, obviously, it indicated an oversized structure; the same thing can be said for the tree-pillars. Indeed, in the fully optimized structures, the glass weight is almost double concerning the grid weight, and it can be 3,5 times the grid weight in the weight-optimized structures. Another observation is that the glass weight has almost the same weight for all the structures, seems that the curvature does not affect it. The grid weight has a bigger variation, but it can be done to the density of the grid and not to the curvature of the roof. About the tree-pillars' effects can be said that it is positive because it reduces the weight of the structure, except for the structures four that were defined to be the lighter; but these two structure is quite different between their as the algorithm's parameters report (tab 4.8.1).

5.9.3 Cost Composition

The other parameters for which the structures are defined are the cost, summarized in the following table and the relative bar chart.

STRUCTURE	GRID WEIGHT (kg)	TREE WEIGHT (kg)	ROOF-AREA (mq)	WASTE FOR OUT OF SIZE PANELS (mq)	GRID COST (€)	TREES COST (€)	GLASS COST (€)	COST FOR OUT OF SIZE PANELS (€)	TOTAL COST (€)
1-TREE	431403	17828	7366	2330	2092304	86.464	1031240	326200	3536208
1-NO TREE	431403	0	7366	2330	2092304	0	1031240	326200	3449744
2-TREE	174595	2324	7366	2330	846787	11.271	1031240	326200	2215499
2-NO TREE	205955	0	7366	2330	998880	0	1031240	326200	2356320
3-TREE	150154	892	7203	3600	728249	4.325	1008420	504000	2244994
3-NO TREE	182710	0	7203	3600	886144	0	1008420	504000	2398564
4-TREE	136513	274	6833	2274	662086	1.328	956620	318360	1938393
4-NO TREE	119944	0	6898	2184	581728	0	965720	305760	1853208
5-TREE	105487	277	6921	2208	511611	1.344	968940	309120	1791015
5-NO TREE	120176	0	6922	2173	582854	0	969080	304220	1856154

Table 4.11 Cost composition of the structures



Graph 4.4 Cost composition of the structures

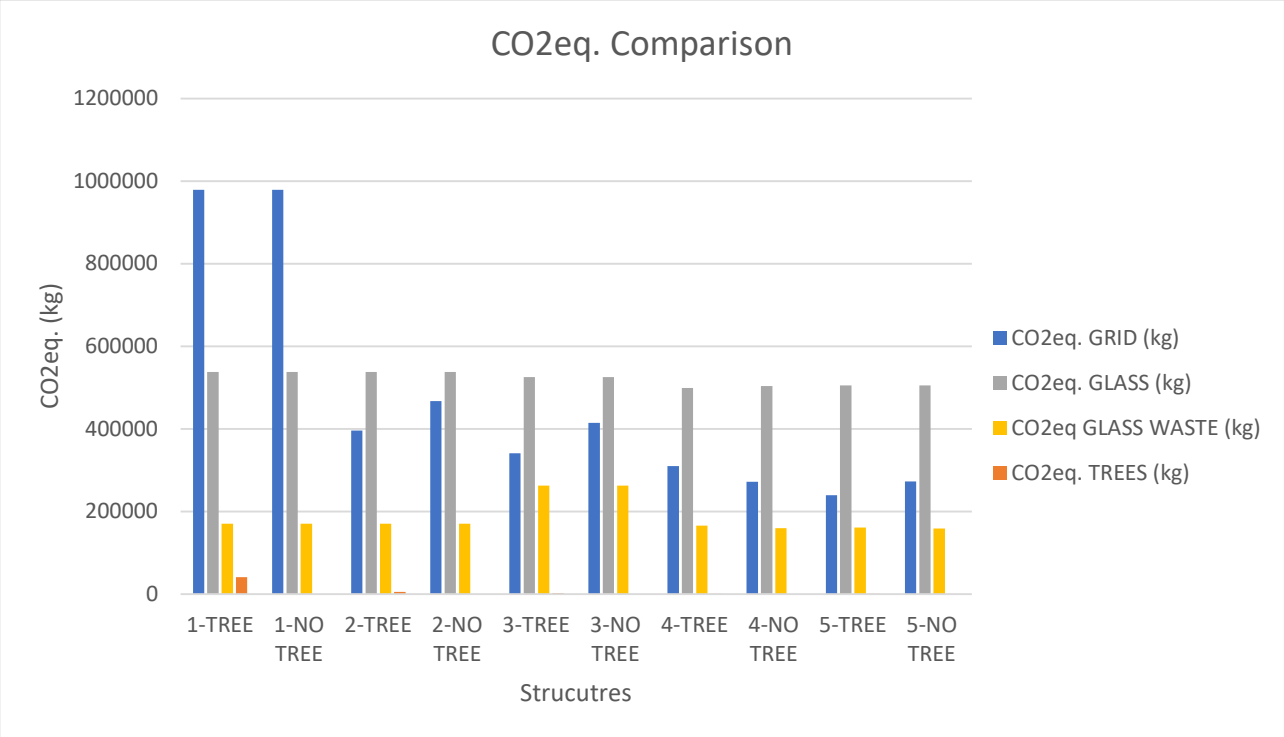
From this bar chart, it is possible to see that the difference between the steel cost and the glass cost is not so much bigger as for the weight. Indeed the steel carpentry is 4,85 €/kg, while the glass is 140,00 €/m^2 that become 2,80 €/kg considering a glass-thickness of 20 mm; this is the reason for these little differences between the different components of the cost. Therefore the cheapest structures have the lower steel weight. The third component is the cost of the waste of glass due to the out-of-size panels; this value is constant for all the structures, indeed they have the panel's ratio of around the 40%, except for the structures three, because all the panels are out of size. About the tree pillars, the cost is negligible, but they help to reduce the steel cost; therefore the cost-benefit ratio is positive.

5.9.4 Materials' Pollution

The last parameter used to define the structures is the GWP of the materials with a unit of measure the $CO_{2eq.}$. As explained in chapter 4.7 this value is proportional with the weight, and with a value of 2,27 kg/kg for the steel and 1,46 kg/kg for the glass. The value for each structure is reported in the table below.

STRUTTUTA	CO2eq. GRID (kg)	CO2eq. TREES (kg)	CO2eq. GLASS (kg)	CO2eq GLASS WASTE (kg)	CO2eq. TOT (kg)
1-TREE	979285	40469	537718	170090	1727561
1-NO TREE	979285	0	537718	170090	1687093
2-TREE	396331	5275	537718	170090	1109415
2-NO TREE	467517	0	537718	170090	1175325
3-TREE	340850	2024	525819	262800	1131494
3-NO TREE	414752	0	525819	262800	1203371
4-TREE	309883	621	498809	166002	975316
4-NO TREE	272273	0	503554	159432	935259
5-TREE	239455	629	505233	161184	906501
5-NO TREE	272800	0	505306	158629	936735

Table 4.12 GWP production of the structures



Graph 4.5 GWP Production of the structures

The bar chart is similar to the previous one, therefore the pollution reduction is given by the reduction of the steel quantity and the out-of-size panels. Comparing the three bar-chart is possible to see the linear relation between the quantity of material, cost, and pollution.

6. Conclusion

The argument of the thesis is the coding of an algorithm for the structural optimization of a grid-shell roof structure and the following analyses. This algorithm exploits the potentiality of *Grasshopper* to obtain an optimized shape, but also to obtain a statically verified structure with the finite element software *Karamba3D.* Therefore the optimization includes the shape and the structure, obtaining as input a structure that can be realized and does not need successive analyses.

Although the algorithm was coded to define a specific roof structure, with specific characteristics, it is quite simple to change the structure because all the parameters can be changed with respective number-slider and the planimetry only changes the planimetry line. Therefore this code is quite adaptable to different structures and can be used in the professional field to design optimized structures.

The evolutionary algorithm *Galapagos* is a real power instrument to research the correct shape; indeed a try and error process surely needs more time and maybe does not take to the same good results. The relation between the input parameters and the final objective is really complex because of the high number of parameters and the complex evaluation of the objective; thus defining the logic to vary the inputs is too complex. Therefore, although the algorithm also works without *Galapagos*, it is strongly recommended its use to define the structures because it reduces the computational time and the difficulty to find a good.

About the structure defined, they have some constrain, like the panels shape, and an objective, like the minimum cost. These parameters can be easily changed, for example fixing the cross-section and researching the structure with the maximum utilization, to have the structure with the maximum exploiting of the structural elements. This shows the big adaptability of the code to the professional necessity.

A problem encountered on the shape optimization is a mesh generator that allows results to define and control the shape of the mesh elements; algorithms constrain is the shape of the panel that should be square with an edge-length of 1,5m , but it is not possible to define it, therefore an algorithm to maximize the panel with this shape was coded. Whereby a future improvement can be the coding of a better inside *Grasshopper,* that allows the meshing algorithm to fix the shape of the mesh module.

7. Bibliography

2. Concepts

- Alessandra Licari, Il design parametrico per l'ottimizzazione di una copertura reticolare, Rel. Giuseppe Carlo Marano. Politecnico di Torino, Corso di laurea magistrale in Ingegneria Civile, 2021.
- AAD, Algorithms-aided design: parametric strategies using Grasshopper, Arturo Tedeschi, Le Penseur Publisher, 2014.
- H.-J. Schek, The force density method for form-finding and computation of general networks, Computer Methods in Applied Mechanics and Engineering, Volume 3, Issue 1, 1974, Pages 115-134, ISSN 0045-7825.
- A. S. Day, An introduction to dynamic relaxation. The Engineer 1965, 219:218–221
- Manuello Bertetto A. Multi-body rope approach for grid shells: form-finding and imperfection sensitivity. Eng Struct. 2020;221:111029.

Sitology:

- https://en.wikipedia.org/wiki/Parametric_design
- https://en.wikipedia.org/wiki/Parametricism
- https://www.domusweb.it/it/movimenti/architettura-parametrica.html
- https://www.fablabvenezia.org/2020/06/10/cose-il-design-parametrico-3-domande-e-risposte/
- https://www.architetturaecosostenibile.it/architettura/criteri-progettuali/design-computazionale-parametrica-292
- http://www.freiotto.com/
- https://it.wikipedia.org/wiki/Frei_Otto
- https://en.wikipedia.org/wiki/Mathematical_optimization
- https://www.researchgate.net/publication/258316734_An_Extended_Force_Density_Method_for_the_form_finding_of_cable_systems_with_new_forms/link/02e7e527c9c5f1c462000000/download
- https://en.wikipedia.org/wiki/Dynamic_relaxation
- https://advanceseng.com/multi-body-rope-approach-grid-shells-finding-
- imperfection-sensitivity/

3. Software

- AAD, Algorithms-aided design: parametric strategies using Grasshopper, Arturo Tedeschi, Le Penseur Publisher, 2014.
- D. Veenendaal, P. Block An overview and comparison of structural form-finding methods for general networks. International Journal of Solids and Structures, Volume 49, Issue 26, 2012; Pages 3741-3753, ISSN 0020-7683. As said by https://block.arch.ethz.ch/brg/files/2012-ijss-veenendaal-block_1380094819.pdf
- Preisinger, C. (2013), Linking Structure and Parametric Geometry. Architectural Design, 83: 110-113 DOI: 10.1002/ad.1564.
- Imparare Python, Mark Lutz, Tecniche Nuove, 2011.

Sitology:

- https://www.grasshopper3d.com
- https://www.food4rhino.com/en
- https://discourse.mcneel.com/t/how-does-kangaroo-solver-work/92075
- https://www.grasshopper3d.com/group/kangaroo/forum/topics/force-density-method-on-kangaroo
- https://www.grasshopper3d.com/profiles/blogs/evolutionary-principles
- https://ieatbugsforbreakfast.wordpress.com/2011/03/07/define-fitness/
- https://en.wikipedia.org/wiki/Evolutionary_algorithm
- https://en.wikipedia.org/wiki/Genetic_algorithm

4. Applications

- NTC 18, D.M. 17 gennaio 2018: Aggiornamento alle “Norme tecniche per le costruzioni”;
- CIRCOLARE 21 gennaio 2019, n. 7 C.S.LL.PP: Istruzioni per l’applicazione dell’«Aggiornamento delle “Norme tecniche per le costruzioni”» di cui al decreto ministeriale 17 gennaio 2018.
- EN 1993-1-1: Eurocode 3;

Sitology:

- https://it.saint-gobain-building-glass.com/it/proprieta-meccaniche-del-vetro;
- https://www.karamba3d.com/news/karamba3d-manual/

5. Analyses

- Prezzario per Opere e Lavori Pubblici della Regione Piemonte 2021, D.G.R. n. 19-3632 del 30/07/2021 (B.U. n. 31 s.o. n. 1 del 05/08/2021)

Sitology:

- www.ibu-epd.com | https://epd-online.com
- https://www.isover.it/sostenibilita/sostenibilita/dichiarazione-ambientale-di-prodotto-epd
- https://dlubal.com/en/cross-section-properties/

8. Index of Figures