

POLITECNICO DI TORINO



Master's Thesis in
Civil Engineering

Generative Modelling and Artificial Intelligence for Structural Optimization of a Large Span Structure

Topology, Size and Shape Optimization

Supervisors:

Prof. Giuseppe Carlo Marano

Laura Sardone

Company:

One Works S.p.A.

Candidate:

Utku Pasin

Academic Year 2020/2021

Contents

ABSTRACT	1
1. OPTIMIZATION ISSUES	3
1.1 Introduction	3
1.2 Structural Optimization	4
1.2.1 Sizing Optimization.....	5
1.2.2 Shape Optimization	7
1.2.3 Topological Optimization.....	8
2. PARAMETRIC DESIGN: A NEW APPROACH TO GENERATIVE DESIGN	12
2.1 Introduction	12
2.2 Parametric Design	13
2.3 Computational Design	15
2.3.1 The Availability of New Technologies.....	16
3. PARAMETRIC SOFTWARE AND ALGORITHMS	18
3.1 Literature Review	18
3.2 Parametric Design Software	20
3.2.1 Grasshopper	24
3.3 The Concept of Algorithm.....	27
3.3.1 The Nature of the Generative Design	28
3.3.2 Introduction to Genetic Algorithms.....	31
3.4 Topology and Form Finding via Genetic Algorithms	36
3.4.1. Grasshopper and Karamba.....	36
3.4.2. Basic Structural Elements.....	37
3.4.3. Karamba Form Finding	38
3.4.4 Topology Process	38
3.4.5. Genetic Algorithms in Grasshopper	39
3.4.6. Galapagos Function	40
3.4.6. Examples of Optimization with Galapagos Function.....	42
3.5 Optimization vs. Adaptation: Multi-Parameter Optimization	43
3.5.1 Single Parameter Optimization.....	44
3.5.2 Multi Parameter Optimization	45
3.5.3 Octopus Function.....	47
4. TOPOLOGICAL, DIMENSIONAL AND SHAPE OPTIMIZATION: THE CASE STUDY	50
4.1 Case Study: A Long Span Grid-Roof Structure	51
4.1 Development of a Parametric Model Supported By Algorithm	52
4.2 FEM Analysis Using Karamba3D.....	57
4.3 Optimization problem: the numerical model.....	64

4.3.1 SPEA-2 and Hype Reduction Algorithms: Solving the Optimization Problem by Using Evolutionary Genetic Algorithms.....	66
4.4 Results	72
5. CONCLUSION.....	82
6. BIBLIOGRAPHY.....	83
7. ACKNOWLEDGEMENTS	85

ABSTRACT

In architectural and structural design, current modeling and analysis tools are extremely powerful and allow one to generate and analyze virtually any structural shape. However, most of them do not allow designers to integrate structural performance as an objective during conceptual design. As structural performance is highly linked to architectural geometry, there is a need for computational strategies allowing for performance-oriented structural design in architecture. Today more than ever, success of optimization is highly depended on Artificial Intelligence (AI) techniques. Thanks to these techniques, the calculation tools can actively interact with the design process by directing the designer to the development of alternatives that show more compliance with project requirements and to search for solutions increasingly oriented to an optimized behavior of the final configuration.

The aim of this thesis is the development of a long span, grid roof structure for a flexible generation and optimization framework for practical use in the sense of a continuously accompanying design explorer, in which parameterization is adaptable and objective functions are changeable at any time during the design process. The user is supported in his/her understanding of correlations by identifying a multiplicity of optimal solutions utilizing state-of-the-art multi-objective search algorithms within the core of the framework. Considering the tool as an interactive design aid, an intuitive interface allowing for extensive manual guidance and verification of the search process is featured. User-selection of preferred solutions support man-machine-dialogue and incorporation of non- or not-yet quantifiable measures.

The work will be presented in four chapters, first through the understanding of the concept of structural optimization and parametric design, supported by the enormous potential of computational and algorithm tools. Then optimization of the chosen case study will be examined by modifying design parameters to minimize the mass of the structure by using genetic algorithms.

In Chapter 1, theory behind the different types of optimization will be examined by discussing the methodologies and advantages offered by its application in the structural field.

In Chapter 2, parametric design will be discussed with its key advantages,

Chapter 3 discusses parametric design application tools, i.e. software and algorithms. Then the chosen parametric software, Rhino and Grasshopper, will be introduced.

Finally, Chapter 4 parametric modelling and structural optimization (with the implementation of an algorithm in Grasshopper) will be applied on a case study.

In this sense, the great advantage offered by parametric modeling and visual programming has allowed convergence to the final configuration of the grid roof structure as a result of a optimization, in terms of mass (as well as cost), implemented through a multi-objective algorithm, and simultaneous optimization of the size of the elements by varying the design parameters.

1. OPTIMIZATION ISSUES

Most problems in engineering, economics, or generally all exact sciences, can be represented and studied as optimization problems. Optimization is the discipline that deals with determining useful models in applications, using efficient methods to identify an optimal solution; this explains the great interest that today, both from a technical and scientific point of view, is placed in the study and development of such models capable of ensuring the complex resolution of a wide range of problems.

1.1 Introduction

Optimization means maximizing or minimizing some function relative to some set, often representing a range of choices available in a certain situation. The function allows comparison of the different choices for determining which might be “best.” In mathematical language, optimizing means determining the value of variables in a function so that that may give minimum or maximum result. In fact, at the design stage, it is always advisable, or in any case useful, that the designer tends to pursue an optimal solution through the use of specific models, called mathematical optimization models, which allow to find the values of a set of parameters that able to maximize (or minimize) the functions of interest. The formulation of an optimization model can be schematic through four main phases:

- (i) *Identifying the problem.* First of all, it is necessary to identify the problem that you want to solve, clearly understanding the objectives of the decision-making process and the constraints that the optimal solution must respect.
- (ii) *Definition of variables.* The decision variables of the model must be defined, corresponding to the choices that the decision-making process entails.
- (iii) *The representation of the objective.* It is necessary to define the objective function of the model, to be minimized or maximized, dependent on the design variables that are freely varied within the optimization process itself.
- (iv) *The representation of the constraints.* Finally, you must identify and represent a constraint set, if any, to the objective function. These constraints represent particular conditions which the solution to the optimization problem must meet

1.2 Structural Optimization

Solving optimization problems is an inherent part of engineering design where one seeks the best design to minimize or maximize an objective function subject to some constraints. In structural optimization, depending on the nature of the design variables, three different optimization categories can be recognized. Sizing optimization arises when the design variables are connected to the dimensions of the elements. It can be useful where the layout and the shapes of the members are known and it is desired to find the optimum dimensions. On another level, one can choose the design variables to control the shape of the boundaries of the members. Such selection will lead to shape optimization. If the overall layout of the members is known and it is already decided where to put each member, in order to find the best shapes of the members, one can use shape optimization. In order to optimize the topology, connectivity, or layout of a system, topology optimization techniques should be used. In topology optimization the design variables control the topology and connectivity of the design.

A degree of overlap exists between the three categories, and arguably a combination of two or more of the categories previously defined constitute a fourth category. The most succinct explanation of the three methods is pictorial. It can be seen that of the three methods topology optimization is the most general, yielding information on the number and shape of openings within a generalized material continua. Potential overlap between the three categories is also apparent since it may be possible to derive a structural topology based on sizing optimization, provided that the minimum size of a member is defined as zero. Structural optimization techniques are most commonly applied to the design of automotive and aerospace structures where weight savings are critical. The application of structural optimization techniques to the design of building and civil/structural engineering structures is, however, a more challenging proposition.

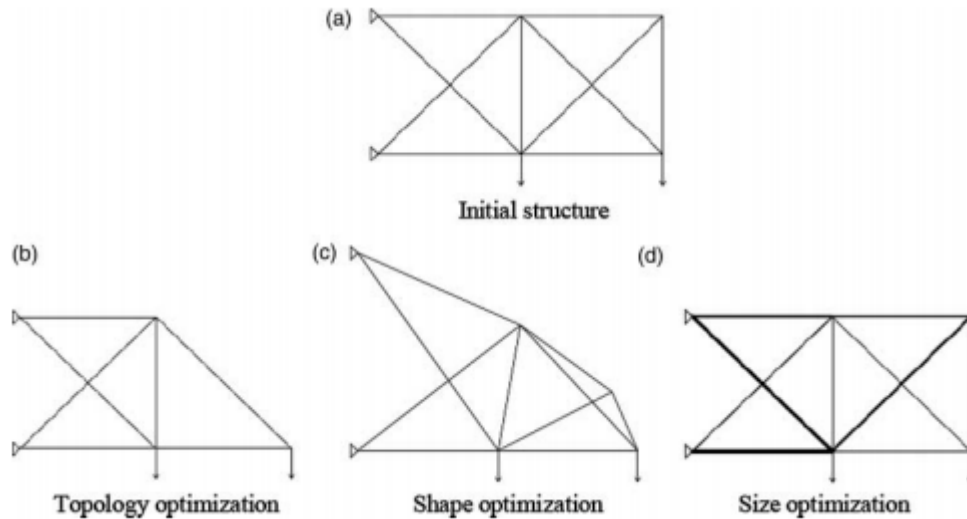


Figure 1 – Structural optimization types

1.2.1 Sizing Optimization

The first optimization class, generally used for truss structures, is the simplest one and involves varying the size of the structural component. For a given design domain and topology (shape and architecture), set of initial properties of the elements (area, thickness, inertia) that can be modified to determine the optimal combination of the properties of each element of the structure. The main feature of a sizing optimization problem is that the design domain is not only known a priori but remains fixed throughout the entire process. A typical problem *size* is illustrated in Figure 1. For the given truss system, constraint, loading and material conditions are known. It could be determined that "best" diameter for the beam elements, or the diameter that minimizes (or maximizes) a specific physical quantity such as compliance, deflection, or peak stress.

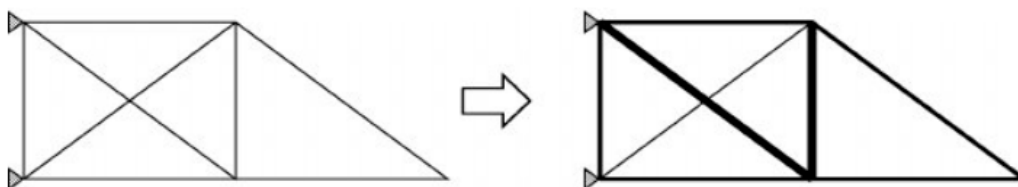


Figure 2 – Sizing optimization

Dimensional optimization is generally performed after shape optimization and topological optimization because it aims to find the optimal values of the parameters that define the cross-sections of the structural elements. In particular, in the case of optimization of plates or shell elements, thickness can be adopted as a set of design variables, as shown in the following figure:

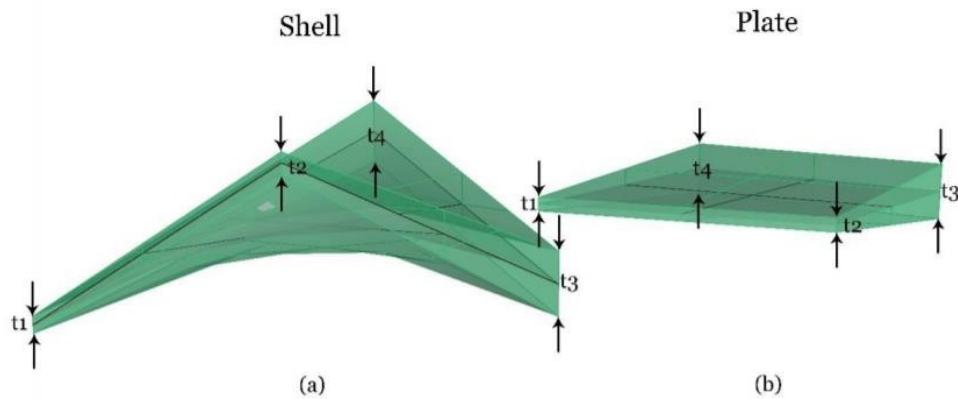


Figure 3 — Example of design variables for an element (a) shell and (b) plate

In this regard, it is common practice to optimize the variable thickness of thin shell covers to improve their structural behavior (mainly dependent on their shape) in terms of robustness, rigidity, and stability, ensuring the most uniform distribution of internal stresses possible to avoid unwanted flexural effects.

In the case of sizing optimization of frame structures, the cross-sectional areas of the members are generally taken as design variables. Instead, it is rarely decided to assume single parameters that characterize the cross-sections of the elements, such as the diameters of the circular cross-sections or the thicknesses of the hollow sections (Figure 4), since they would determine a significant increase in the design variables themselves.

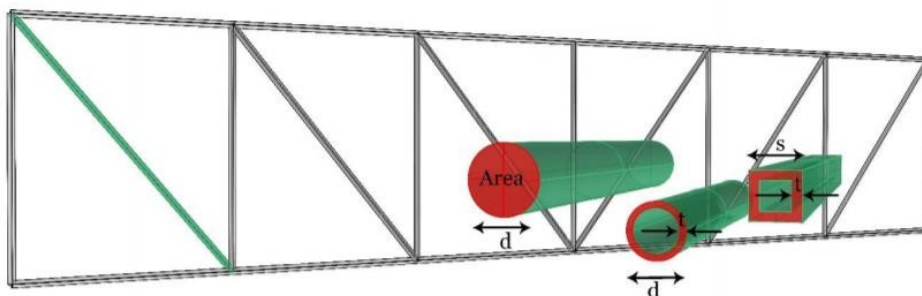


Figure 4 - Example of dimensional optimization variables for truss elements or frame structures

In addition, to reduce the necessary number of design variables, it is possible to group the elements of frame structures into several groups according to their structural functions or some geometric considerations (symmetrical conditions). On the other hand, in the optimization of discrete structures, such as frame structures, it may be convenient to adopt discrete design variables, such as parameters taken from a list of commercial cross-sections. Because solving an optimization problem with discrete design variables is more complex than solving similar problems with continuous variables, variables can be set as continuous later by rounding to the nearest integer.

1.2.2 Shape Optimization

Shape optimization problems employ the governing geometry variables of a shape parametrization as optimization variables, e.g. node coordinates of finite elements, control point coordinates of CAD models or morphing boxes or amplitudes of shape basis vectors. The topology of the structure (connectivity of elements) remains constant which prevents the generation of holes. The figure below motivates a simple shape optimization problem of a truss beam structure. It can be easily observed that the topology (connectivity) of all three design is equal although the geometry and therefore the load carrying behavior changes completely.

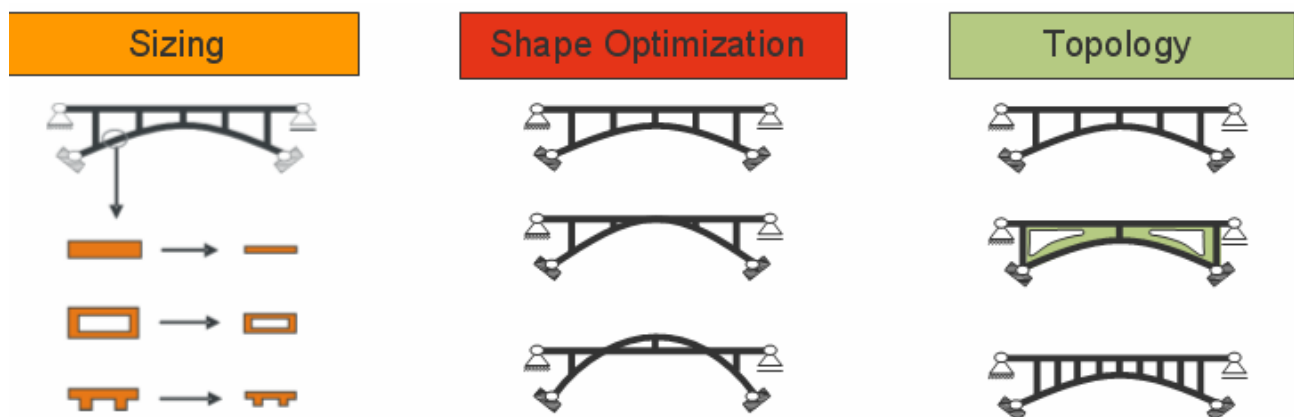


Figure 5 – Structural optimization types

Formulation of shape derivatives results in complex and therefore time consuming algorithms compared to material or sizing variables whereby algorithmic complexity is strongly related to the applied finite elements. In general response functions of shape optimization problems are highly non-convex especially for thin and lightweight structures caused by large differences in efficiency of load carrying mechanisms.

In general, design variables to consider can be the thickness distribution at an element of the structure, the diameter of a hole, the radius of curvature, or any other characteristic measure of the domain. It can then start from an initial geometric configuration and arrive, through procedures of swelling or thinning of the elements, to an optimized geometry, as shown in the following figure:

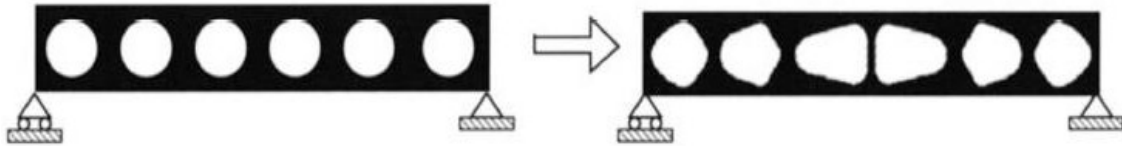


Figure 6 – Shape Optimization

The nodal coordinates of a continuous or discrete structure (appropriately discretized into lines or surface elements) can also be directly assumed as design variables, as shown in Figure 7:

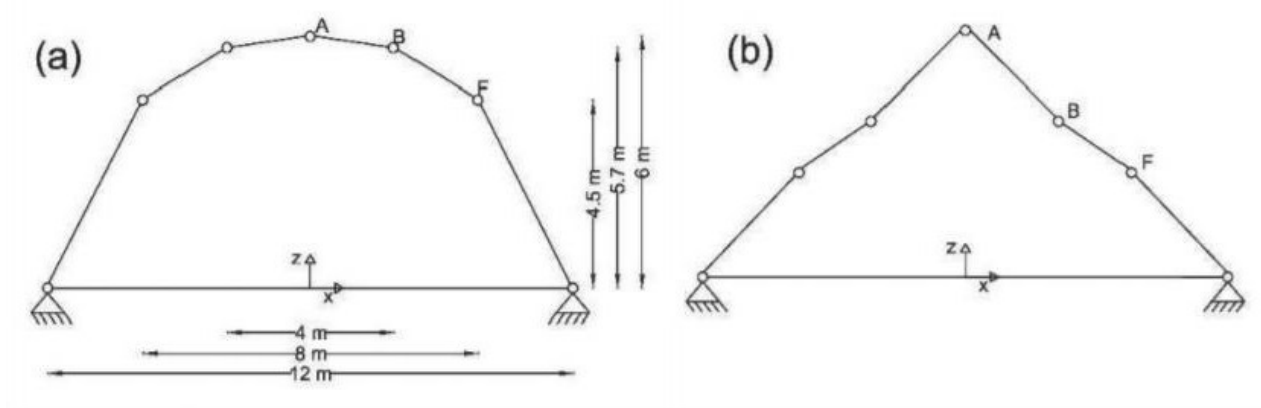


Figure 7 - Optimizing the shape of a discrete structure

1.2.3 Topological Optimization

Topology Optimization (TO) offers much more freedom for a designer to create totally innovative and efficient conceptual designs.

There are two types of TO, namely discrete or continuous types, which mainly depends on the type of the structure. For discrete structures like trusses and frames, TO is to figure out the optimal spatial order and connectivity of the structural members. This area has been largely developed by Prager and

Rozvany. On the other hand, TO of continuum structures is to search the optimal designs through pinpointing the best locations and geometries of cavities in the design space (Huang X and Xie M, 2010).

Topological optimization is the most general form of structural optimization whose application, in the various fields of engineering, can lead to significant improvements both in terms of cost, design and quality. For a design domain (volume), topology optimization seeks for the best spatial distribution of the material within the design space, thus determining the position, number, and interconnection of the voids and solids inside as shown in the following figure:

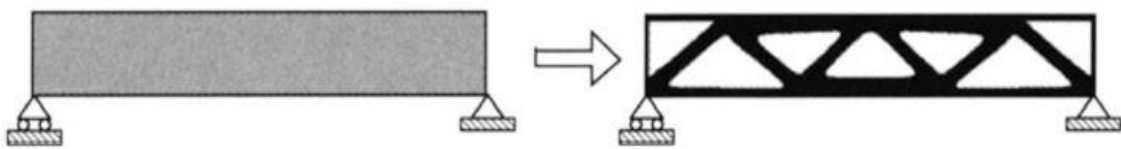


Figure 8- Topological Optimization (OT)

The material within the design space, thus determining the position, number, and interconnection of the voids and solids inside as shown in the following figure:

As stated above, TO have great Optimization potential since it is used in the initial phase of the design. During the past years, substantial progress has been achieved in the development of TO, especially by means of FEM, which is introduced by Richard Vourant in 1943 to figure out solutions for partial differential equations. It is also referred as finite element analysis (FEA). In this context, the geometry of the design domain is subdivided into smaller elements that are interconnected at nodes. Therefore, the entire domain, which is filled with elements without overlaps, is analyzed for functional performance. After applying boundary conditions and loads to the part, the resulting finite element equations are solved accordingly (Ramana M. Pidaparti, 2017). FEA is widely used in many areas of research, including biomedical, mechanical, civil, aerospace researches, etc. When it comes to TO, “The typical approach to plane or volumetric structures Topology Optimization is the discretization of the problem domain in a number finite elements and the assignment of full material, partial material or lack of material to each element, in an iterative scheme converging to the optimal material distribution inside the domain.” (Razvan CAZACU and Lucian GRAMA, 2014).

Because of their complexity, TO problems are often solved numerically using algorithms and iterative procedures. There are several approaches including the use of genetic algorithms, which allow the

removal of inefficient material between iterations during the optimization process. Without entering into the specific treatment, iteration after iteration, the densities of the materials associated with each finished element, modeled through FEM analysis, are updated through different methods such as MMA, Method of Moving Asymptotes, based on the results obtained from sensitivity analysis.

For example, Figure 9 shows a computational flow of topological optimization:

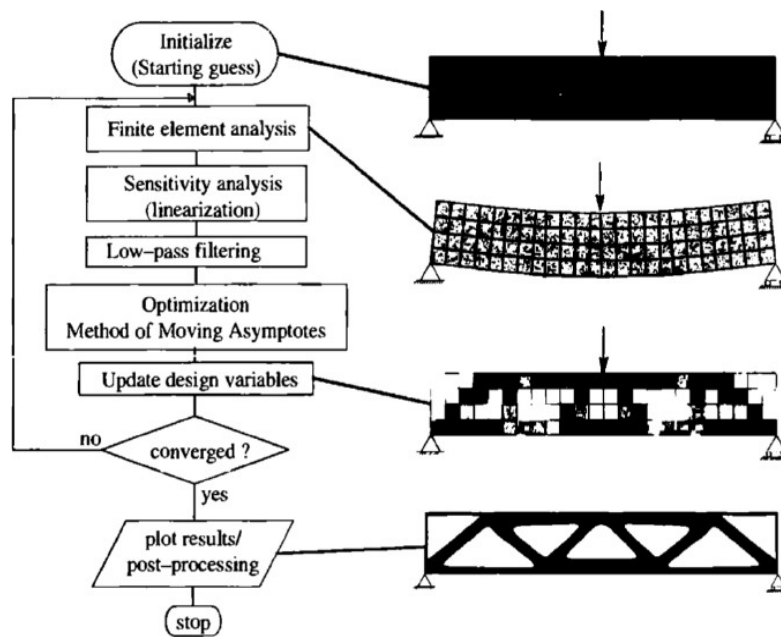


Figure 9 - Topological Optimization Flow

The flow is divided into five steps: first of all the initial design is defined characterized by a homogeneous distribution of the material through the FEM method each element is modeled by calculating displacements and compliance of the structure, the sensitivity analysis is performed, the design variables are updated using the MMA method, and finally, the convergence of the result is checked by evaluating whether, between the last step and the previous one, the variation in compliance is negligible or lower than a fixed limit.

In the case of trusses, the topological optimization takes a discrete form (also known as the topological optimization of the truss, or TTO) and the goal becomes to optimize the connectivity between the nodes, whose coordinates are known and fixed, of a grid. This typology of optimization problems can be conveniently formulated using the so-called “ground structure method” (MP Bendsoe and

Sigmund 2003); in this approach, the layout of a lattice structure can be found allowing for a certain set of connections between the nodal points of a fixed grid.

Finally, it is worth emphasizing that topological structural optimization can be seen as a category of shape optimization, even if it is subject to design constraints (for example, fixed nodes in the case of TTO).

2. PARAMETRIC DESIGN: A NEW APPROACH TO GENERATIVE DESIGN

2.1 Introduction

In the previous chapter, it was discussed how optimization techniques and technological innovations can support the design and how the searching for optimal solutions of a structure is the basis of the design phase.

Within an optimization process, a parametric definition of the design problem is often indispensable. For instance, objective function and the constraint functions is mainly depend on set of design variables that called parameters. (Figure 10).

Especially for the shape optimization of large structures (continuous or discrete) characterized by a large number of nodes may require a large number of project variables; in this sense, it could be advantageous to adopt parametric shape functions, depending on a small number of parameters that can be assumed as shape optimization variables for the considered problem.

The following chapter aims to introduce the concept of parametric design and the high-performance design tools that play a fundamental role in the preliminary phase of a structural optimization process. They are implemented through software platforms in which the designer is asked to insert a series of parameters that are subsequently processed by the software itself in a form created based on these parameters. That is why computational design is generally considered to be the tool that makes it possible to use parametric methods within design processes. From this definition, it is possible to understand that computational design, compared to traditional methods, offers passage from a logic of representation of the architectural object to a 3D simulation.

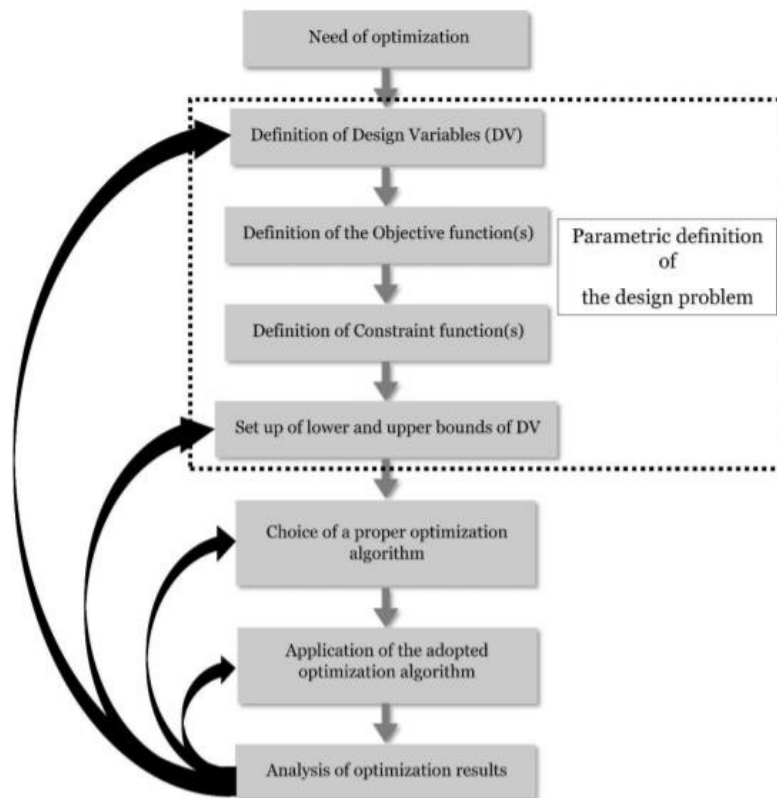


Figure 10 – "Parameterization" within a standard structural optimization process

2.2 Parametric Design

The inexorable refinement of design processes, fueled by prototyping in an exclusively digital space, has allowed engineers and designers to rethink their approach to many different application areas. Architectural prototyping, 3D modeling, and design, for example, have seen new, daring approaches in the past few decades that are today characterized by "parametric design."

The phrase "what's old is new again" has its place in architecture. However, today's groundbreaking designs have bucked tradition. Parametric design has particularly rebelled against long-standing guidelines. The term 'parametricism' was coined by Patrik Schumacher, who was a partner at Zaha Hadid Architects at the time.

Straight lines, sharp corners, and acute angles were the lifeblood of former styles. Conversely, Parametricism centers on free-form architectural concepts. Sweeping lines, curves, and irregular shapes give each building character. Such designs might look futuristic or even otherworldly.

Parametric architecture is defined by the following:

- Blending complexity and variety, thus rejecting homogenous utilitarianism
- Shared priorities involving urbanism, interior design, an architectural wonder, and even fashion
- The idea that all design elements are interdependent and adaptable
- A skew towards computerized, algorithmic design processes

Because parametric tools use algorithms, it becomes much easier to produce intricate designs. Design teams can devise sets of parameters before experimentation. Applications can churn out design candidates with degrees of variance.

French architect Jean Nouvel has designed many buildings using parametric design, one of the most notable being the Louvre Abu Dhabi. The building is much more subtle than its skyscraper counterparts, but its intricate dome holds its own. (Figure 11.)



Figure 11 – Louvre Abu Dhabi by Jean Nouvel.

Santiago Calatrava's otherworldly design for the World Trade Center Transportation Hub (also known as the Oculus) in New York City is another example of parametric architecture. Both its interior and exterior push the boundaries of what architecture can be. (Figure 12.)



Figure 12 – One World Trade Center Transportation Hub/Oculus by Santiago Calatrava.

2.3 Computational Design

Computational design, as the word itself says, is based on the use of computer tools within the design process and offers to designer the possibility to manipulate ideas, concepts, and processes via computer, as well as to control the conversion of object and input properties into different solutions and alternatives with the aim of optimizing the functional requirements of the design problem.

Therefore it makes possible to apply computational strategies to a compositional process and it can be understood as the product of the union of two disciplines: design and computation. A symbiosis, capable of combining those creative, functional, symbolic and productive factors that lead to the creation of a design object, and on the algorithms underlying computation, which translate the complexity of reality into a succession of simple elements and transform them into data that can be reworked in the form of algorithms that capable of managing and solving specific problems.

The great potential of computational design lies in the ability to make automatic changes to a chain of series of data that would normally take lots of time, energy and money. This speedup of operation makes it possible to study several solutions for a single request.

Once the object has been defined, such as a cover of a pavilion, the parameters should be defined such as, specifying dimensional, quantitative, material parameters, shape management, etc.

This operation with common CAD instruments, would be almost impossible, or at least very time and energy-consuming, to represent in 2D or 3D.

The role of an engineer is to realize the ambition of his customers by using innovative technologies that capable of meeting with growing demand more efficiently and offering higher standards. Within a generic design process, the key factors in using digital tools are:

- The need for cost/time efficient design process;
- The availability of new technology;
- The growing demands for project result performance

As a result, it is easy to guess how the use of computational-based methods not only dominates the way of design it also determines how we collaborate, share information, and organize the construction process itself.

2.3.1 The Availability of New Technologies

Today, digital tools, from machine learning to manufacturing technologies, from artificial intelligence to Big Data, are becoming increasingly ubiquitous and pervasive within our daily lives. This new technologies allows to overcome the previous limits and therefore improve the manageability of the designer. Moreover it inspires engineers, and architects to find solutions for problems that used to be considered as impossible to solve.

History teaches us that those new technologies are often needed from an engineering point of view to solve a specific technical problem. For example, the mathematical descriptions (algorithms) that allow Renault Citroen engineers to define the curved shapes of their new car designs, have eventually lead to the widespread use of such parametric functions in cad(computer-aided-design) tools that

inspired architects such as Frank O'Gehry, Zaha Hadid or many others to design their freeform buildings.

In past years, for some complex geometries, engineers had to devote their lives for that single project, but now those problems can be easily solved in couple of minutes by new technologies. Advanced computational design tools such as Rhinoceros and Autodesk3D, allows to design three-dimensional structures of complex shape that were considered too difficult or time-consuming to communicate or modify with traditional two-dimensional drawings or small-scale models.

The availability of sufficient computational and advanced analysis software, such as finite element analysis (FEA), allows us to use digital models to also obtain information on the structural behavior of any structure.

In the field of civil engineering, for example, a renovation project of a building of the Delft University of Technology faculty, a slit structure between the cantilevered classrooms, designed using specific structural optimization algorithms capable of allocating the material only to the places where it is most effective.

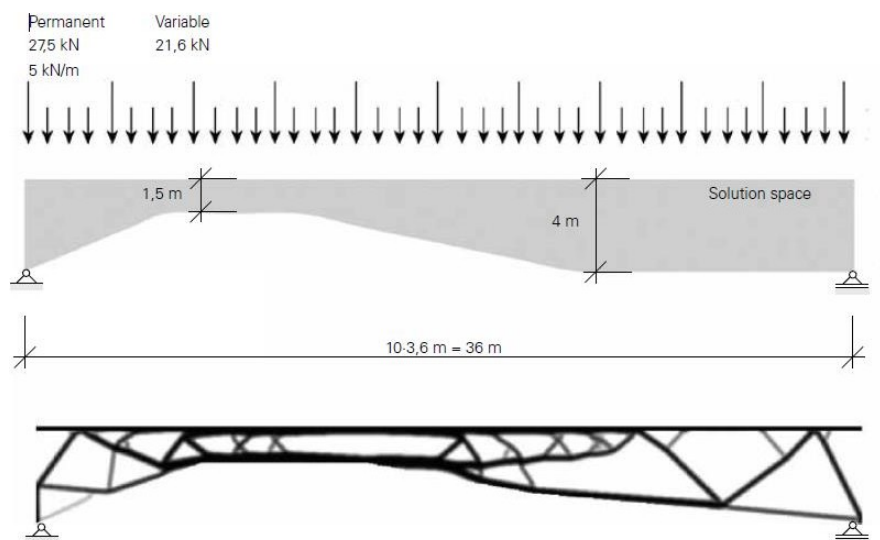


Figure 13 – Generating the layout of a slit structure using specific algorithms

The growing demands for performance of buildings and built environments, such as sustainability ambitions, also require an integrated process. As a result, the complexity of design workflows increases significantly and creates the need for advanced simulations to gain an insight into the numerous input variables and thus the consequences of project alterations on the overall result.

3. PARAMETRIC SOFTWARE AND ALGORITHMS

3.1 Literature Review

Engineers are under growing pressure to be more productive and, at the same time, structural design is becoming increasingly complex. The architectural design tools of today have expanded the possibilities for more curves and amazing structural details, all of which end up on the desktop of structural engineers, who today are working within smaller budgets to meet tighter deadlines. These constraints have driven structural engineers to explore so-called algorithm enabled parametric modeling, also called parametric design.

First, there is “parametric design” and “parametric modeling.” Parametric design is based on defining parameters, or in other words, data inputs that are connected with modeled objects. When adjusting one parameter, such as changing the number of columns or extending the width of a deck, all of the model objects affected by that change are automatically updated. When that parametric design is combined with a proper parametric BIM tool, the parameters can simultaneously be used to drive information-rich BIM data beyond simple geometry, thus the benefits of parametric BIM are far-reaching.

In practice, algorithm enabled parametric BIM allows structural engineers to easily and visually create data input schema using an algorithm-based editor and then output objects to a parametric BIM tool.

Today, structural engineers are leveraging this workflow without prior knowledge of programming through direct links between BIM software and visual programming tools such as Grasshopper, which is a pre-installed plugin for Rhinoceros 6, a 3D computer graphics and computer-aided design (CAD) application. This is especially beneficial for creating complex shapes like curved structures and architecturally challenging design intent.

By adding a visual programming editor, such as Grasshopper, to your design workflow, you can define input parameters such as coordinates, dimensions, curves or even complex NURBS, and then visually script rules that act on these parameters in order to generate the desired geometry or other output, which can then be applied directly to live objects in parametric BIM software which contains all of the necessary attributes to meet industry requirements.

Based on the attributes you define, the effects of any change to the design are automatically populated throughout the model. Essentially, Grasshopper takes your inputs, does the calculations and produces

an output that is applied to the model. This eliminates the need to manually apply changes across the model and allows you to quickly generate and visualize multiple iterations of complex designs in 3D by simply adjusting the attributes. This workflow is especially beneficial when modeling structures, such as bridges, that have complex geometries and curved surfaces.

Parametric design linked with parametric modeling, that is, algorithm enabled parametric BIM can help you meet all of these demands with a workflow that:

- Simplifies modeling complicated geometry
- Enables fast structural form iterations and design option investigations
- Significantly increases productivity by reducing the time it takes to modify designs
- Brings significant efficiency gains and benefits to the design of complex bridges and junctions
- Encourages and enables collaboration in the design process with better visualization and simulation across disciplines
- Allows you to simultaneously create calculations and see them populate the model in real-time
- Allows you to design repetitive geometries with less work, such as similar connections or geometries that follow a new alignment
- Is a step toward generative design which combines artificial intelligence with parametric BIM

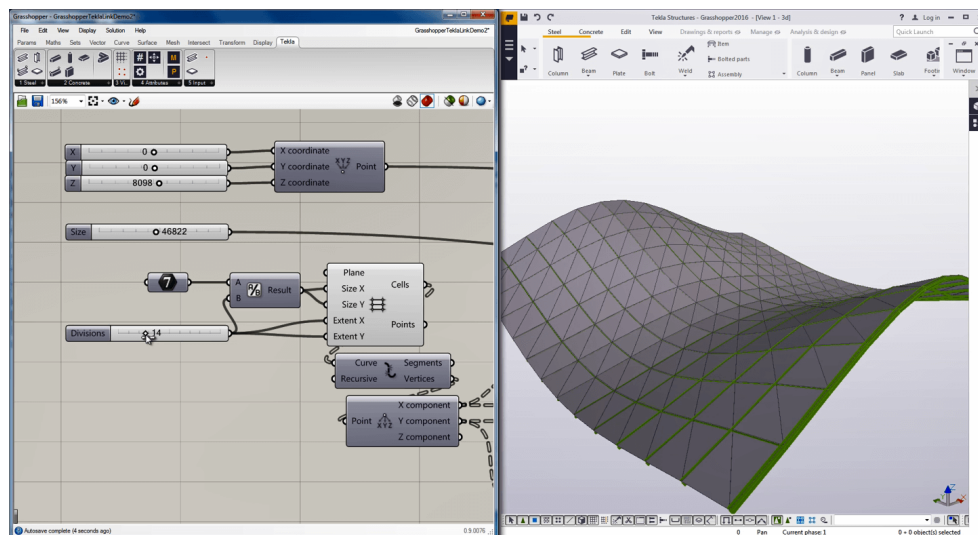


Figure 14 – A view from Grasshopper

3.2 Parametric Design Software

As we enter a new emerging era in the field of contemporary architecture and design, there is a tremendous demand for highly customizable convoluted geometry playing vital roles in the overall form, shape and size of the buildings. The direct outcome of such an increasingly persistent requirement for “modern” & “futuristic” forms, has paved the way for innovative new techniques and tools for today’s architect – Computer Aided Design (CAD). It deals with the use of computing devices to abet +one’s odds of better perceiving and visualizing a design.

It is generally used to create, analyze, modify and finally present the design, due to the accuracy and quality, it is capable of consistently administering. Parametric design on the other hand refers to the use of parameters such as constraints, the relationships between geometric entities, dimensions, the shape and size of the entities, etc. These parameters are substantially responsible in determining the relationship in between the design intent and the design response and overall, to algorithmically generate the desired framework and form of the model, which is to be visualized.

Among the most common software used by designers, interested in developing parametric design projects, the main ones are as follows:

Grasshopper 3D

Grasshopper 3d (originally Explicit History) is a plug-in for Rhinoceros 3D that presents the users with a visual programming language interface to create and edit geometry.

Components or nodes are dragged onto a canvas in order to build a grasshopper definition. Grasshopper is based on graphs (see Graph (discrete mathematics)) that map the flow of relations from parameters through user-defined functions (nodes), resulting in the generation of geometry. Changing parameters or geometry causes the changes to propagate throughout all functions, and the geometry to be redrawn.

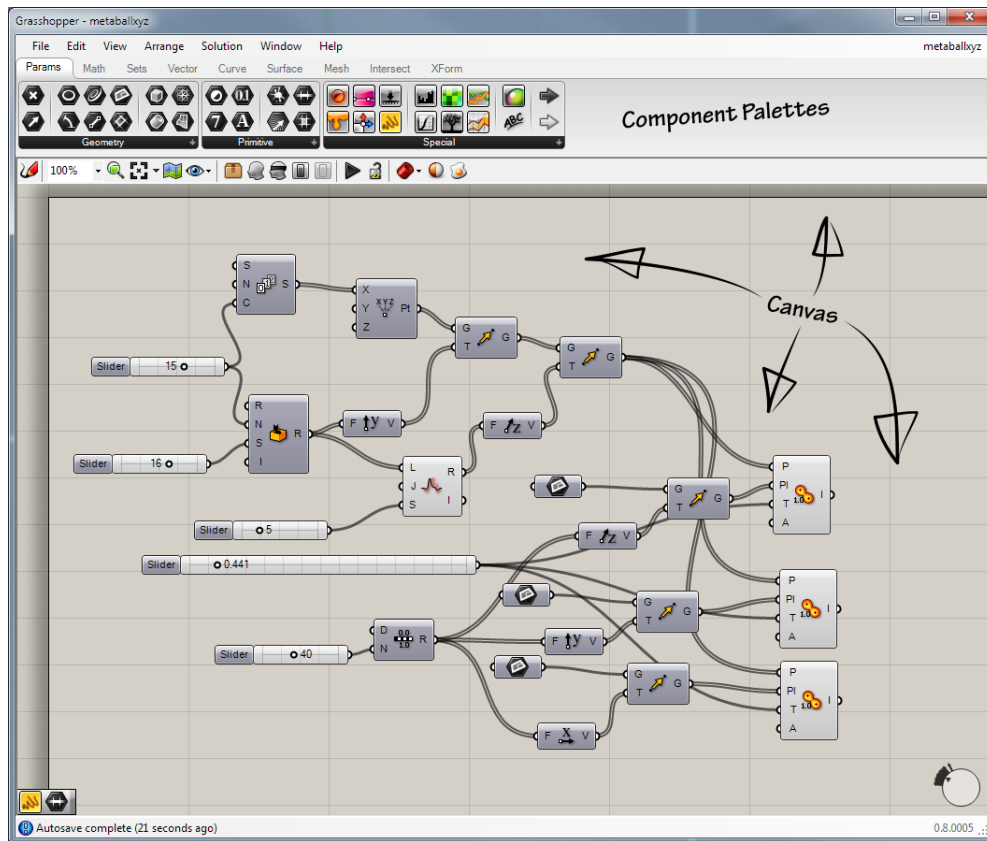


Figure 15 – The Grasshopper canvas with some nodes

Autodesk Revit

Autodesk Revit is building information modeling (BIM) software used by architects and other building professionals. Revit was developed in response to the need for software that could create three-dimensional parametric models that include both geometry and non-geometric design and construction information. Every change made to an element in Revit is automatically propagated through the model to keep all components, views and annotations consistent. This eases collaboration between teams and ensures that all information (floor areas, schedules, etc.) are updated dynamically when changes in the model are made.

Autodesk Dynamo

Dynamo is an open source graphical programming environment for design. Dynamo extends building information modeling with the data and logic environment of a graphical algorithm editor.

Autodesk 3DS Max

Autodesk 3ds Max is a parametric 3D modeling software which provides modeling, animation, simulation, and rendering functions for games, film, and motion graphics. 3ds Max uses the concept of modifiers and wired parameters to control its geometry and gives the user the ability to script its functionality. Max Creation Graph is a visual programming node-based tool creation environment in 3ds Max 2016 that is similar to Grasshopper and Dynamo.

Catia

CATIA (Computer Aided three-dimensional Interactive Application) was used by architect Frank Gehry to design some of his award-winning curvilinear buildings such as the Guggenheim Museum Bilbao. Gehry Technologies, the technology arm of his firm, have since created Digital Project, their own parametric design software based on their experience with CATIA.

Power Surfacing

Power Surfacing is a SolidWorks application for industrial design / freeform organic surface/solids modeling. Tightly integrated with SolidWorks, it works with all SolidWorks commands. Reverse Engineer scanned meshes with Power Surfacing RE.

GenerativeComponents

GenerativeComponents, parametric CAD software developed by Bentley Systems, was first introduced in 2003, became increasingly used in practice (especially by the London architectural community) by early 2005, and was commercially released in November 2007. GenerativeComponents has a strong traditional base of users in academia and at technologically advanced design firms.[citation needed] GenerativeComponents is often referred to by the nickname of 'GC'. GC epitomizes the quest to bring parametric modeling capabilities of 3D solid modeling into architectural design, seeking to provide greater fluidity and fluency than mechanical 3D solid modeling.

Users can interact with the software by either dynamically modeling or directly manipulating geometry, or by applying rules and capturing relationships among model elements, or by defining complex forms and systems through concisely expressed algorithms. The software supports many industry standard file input and outputs including DGN by Bentley Systems, DWG by Autodesk, STL (Stereo Lithography), Rhino, and others. The software can also integrate with Building Information Modeling systems.

The software has a published API and uses a simple scripting language, both allowing the integration with many different software tools, and the creation of custom programs by users

This software is primarily used by architects and engineers in the design of buildings, but has also been used to model natural and biological structures and mathematical systems. Generative Components runs exclusively on Microsoft Windows operating systems.

VIKTOR

VIKTOR is an application development platform that enables engineers and other domain experts to rapidly build their own online applications using Python. It is used to create parametric design models and integrates with many software packages. It enables users to make intuitive user interfaces (GUI), which include different form of visualizing results like 3D models, drawings, map or satellite views, and interactive graphs. This makes it possible to make the applications available to persons without programming affinity.

Applications made with VIKTOR are online, meaning data is update automatically and everyone works with the same information and the latest models. It includes a user management system, allowing to give different rights to users.

Modelur

Modelur is a parametric urban design software plug-in for Trimble SketchUp, developed by Agilicity d.o.o. (LLC).. Its primary goal is to help the users create conceptual urban massing. In contrast to common CAD applications, where the user designs buildings with usual dimensions such as width, depth and height, Modelur offers design of built environment through key urban parameters such as number of stories and gross floor area of a building.

Modelur calculates key urban control parameters on the fly (e.g. floor area ratio or required number of parking lots), delivering urban design information while the development is still evolving. This way it helps taking well-informed decision during the earliest stages, when design decisions have the highest impact.

Archimatix

Archimatix is a node-based parametric modeler extension for Unity 3D. It enables visual modeling of 3D models within the Unity 3D editor.

3.2.1 Grasshopper

In the content of this thesis, Grasshopper will be used as a main software for generative design and optimization of the chosen case study. For this reason, main futures of Grasshopper will be introduced.

The interface layout and its elements are as shown below.

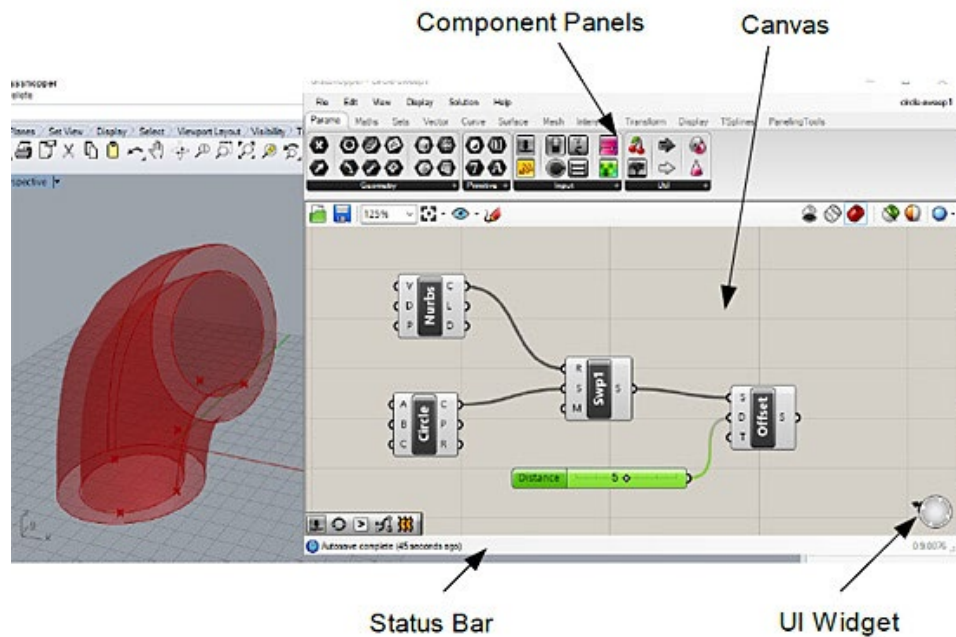


Figure 16 – The Grasshopper Interface

Anatomy of a Node

The left side sockets are for receiving inputs; whereas the right side sockets are for down streaming outputs.

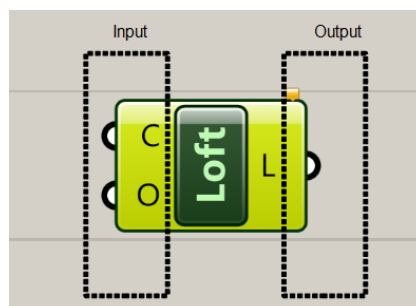


Figure 17 – The Grasshopper – Node

Colours of node and their corresponding meanings:

Orange – with warnings

Green – is selected

Light Grey – no warning or errors

Darker grey- preview is disable

Greyed out – has been disabled

Red – has errors

Basic Operations

There are 2 methods of adding components onto the canvas:

Method 1: Drag a new component onto the canvas

Method 2: Double click on the canvas to bring up the keyword search box.

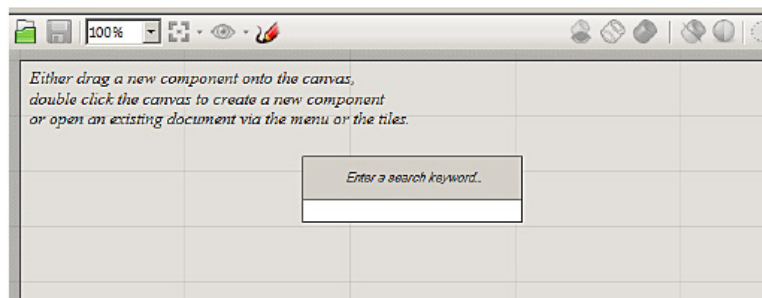


Figure 18 – The Grasshopper – Adding a component

To disconnect from a socket, right-mouse click, select **Disconnect**.

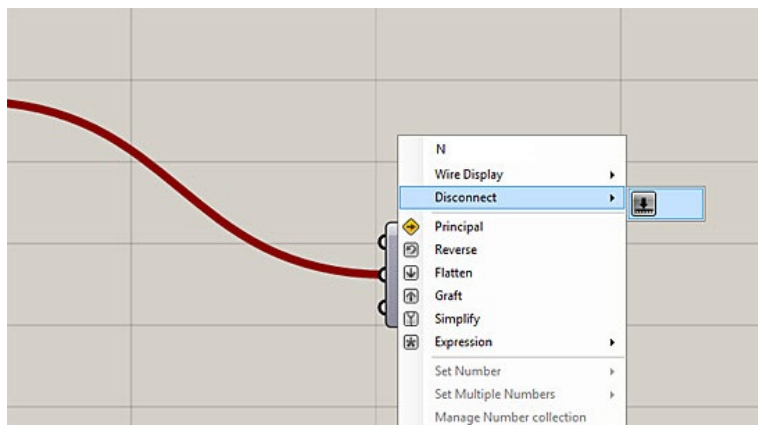


Figure 19 – The Grasshopper – Disconnect

To connect to a socket which already has an existing connection, press and hold the SHIFT key.

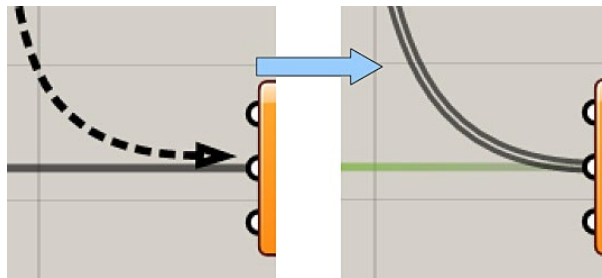


Figure 20 – The Grasshopper – Disconnect

Data Input methods

Generally, data can be from the following sources:

1. Referenced from Rhino objects
2. Assign values for component

Saving Grasshopper Definition

The algorithms that are built in Grasshopper are referred to generally as definitions. To save a definition in Grasshopper, go to File > Save or File Save As commands. A Grasshopper definition can be saved in 2 formats:

1. .gh format is for Grasshopper binary
2. .ghx format for Grasshopper XML.

Converting to Rhino surfaces – Bake surfaces

To convert Grasshopper previews into geometry that's editable in Rhino, we need to use the bake command. To do that, hover cursor over component whose geometry you'd like to convert in to editable geometry. Right click and select Bake.

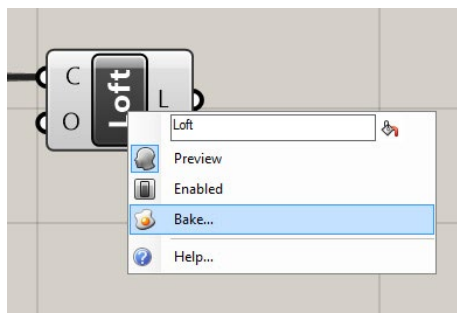


Figure 21 – The Grasshopper – Bake function

3.3 The Concept of Algorithm

In mathematics and computer science, an algorithm is a finite sequence of well-defined, computer-implementable instructions, typically to solve a class of specific problems or to perform a computation. Algorithms are always unambiguous and are used as specifications for performing calculations, data processing, automated reasoning, and other tasks. In contrast, a heuristic is a technique used in problem solving that uses practical methods and/or various estimates in order to produce solutions that may not be optimal but are sufficient given the circumstances.

As an effective method, an algorithm can be expressed within a finite amount of space and time, and in a well-defined formal language for calculating a function. Starting from an initial state and initial input (perhaps empty), the instructions describe a computation that, when executed, proceeds through a finite number of well-defined successive states, eventually producing "output" and terminating at a final ending state. The transition from one state to the next is not necessarily deterministic; some algorithms, known as randomized algorithms, incorporate random input.

The concept of algorithm has existed since antiquity. Arithmetic algorithms, such as a division algorithm, was used by ancient Babylonian mathematicians c. 2500 BC and Egyptian mathematicians c. 1550 BC. Greek mathematicians later used algorithms in 240 BC in the sieve of Eratosthenes for finding prime numbers, and the Euclidean algorithm for finding the greatest common divisor of two numbers. Arabic mathematicians such as al-Kindi in the 9th century used cryptographic algorithms for code-breaking, based on frequency analysis.

The word algorithm itself is derived from the name of the 9th-century mathematician Muḥammad ibn Mūsā al-Khwārizmī, whose nisba (identifying him as from Khwarazm) was Latinized as Algoritmi. A partial formalization of what would become the modern concept of algorithm began with attempts to solve the Entscheidungsproblem (decision problem) posed by David Hilbert in 1928. Later formalizations were framed as attempts to define "effective calculability" or "effective method". Those formalizations included the Gödel–Herbrand–Kleene recursive functions of 1930, 1934 and 1935, Alonzo Church's lambda calculus of 1936, Emil Post's Formulation 1 of 1936, and Alan Turing's Turing machines of 1936–37 and 1939.

In the Figure 22 it can be seen that flowchart of an algorithm (Euclid's algorithm) for calculating the greatest common divisor (g.c.d.) of two numbers a and b in locations named A and B . The algorithm proceeds by successive subtractions in two loops: IF the test $B \geq A$ yields "yes" or "true" (more

accurately, the number b in location B is greater than or equal to the number a in location A) THEN, the algorithm specifies $B \leftarrow B - A$ (meaning the number $b - a$ replaces the old b). Similarly, IF $A > B$, THEN $A \leftarrow A - B$. The process terminates when (the contents of) B is 0, yielding the g.c.d. in A . (Algorithm derived from Scott 2009:13; symbols and drawing style from Tausworthe 1977).

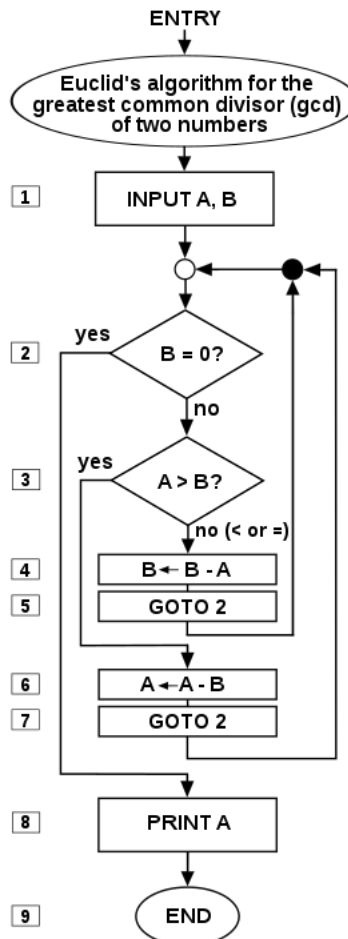


Figure 22 – Euclid's Algorithms

3.3.1 The Nature of the Generative Design

If we look at architecture as an object, represented in space, we always deal with geometry and a bit of math to understand and design this object. In the History of architecture, different architectural styles have presented multiple types of geometry and logic of articulation and each period has found a way to deal with its geometrical problems and questions. Since computers have started to help architects, simulate space and geometrical articulations, it became an integral tool in the design process. Computational Geometry became an interesting subject to study and combination of

programming algorithms with geometry, yielded algorithmic geometries known as Generative Algorithms. Although 3D softwares helped to simulate almost any space visualized, it is the Generative Algorithm notion that brings the current possibilities of design, like ‘parametric design’ in the realm of architecture. Architects started to use free form curves and surfaces to design and investigate spaces beyond the limitations of conventional geometries of the “Euclidian space”. It was combination of Architecture and Digital that brought ‘Blobs’ on the table and pushed it further. Although the progress of the computation is extremely fast, architecture has been tried to keep track with this digital fast pace progress. Contemporary architecture after the age of “Blob” seems to be more precise about these subjects. Architectural design is being affected by potentials of algorithmic computational geometries with multiple hierarchies and high level of complexity. Designing and modelling free-form surfaces and curves as building elements which are associated with different components and have multiple patterns is not an easy job to do with traditional methods. This is the power of algorithms and scripts which are forward pushing the limits. It is obvious that even to think about a complex geometry, we need appropriate tools, especially softwares, which are capable of simulating these geometries and controlling their properties. As a result, architects feel interested to use Swarms or Cellular Automata or Genetic Algorithms to generate algorithmic designs and go beyond the current pallet of available forms and spaces. The horizon is a full catalogue of complexity and multiplicity that combines creativity and ambition together.

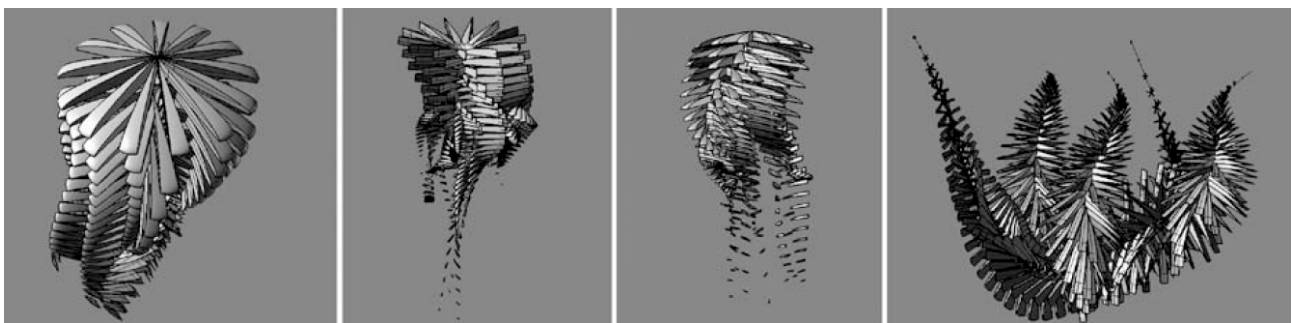


Figure 23 – Parametric Modelling for Evolutionary Computation and Genetic Algorithm, Zubin Mohamad khabazi, Emergence Seminar, AA, conducted by Michael Weinstock, fall 2008.

A step even forward, now embedding the properties of material systems in design algorithms seems to be more possible in this parametric notion. Looking at material effects and their responses to the hosting environment in the design phase, now the inherent potentials of the components and systems should be applied to the parametric models of design. Not only these generative algorithms deal with form generation, but also there is a great potential to embed the logic of material systems in them. “The underlying logic of the parametric design can be instrumentalised here as an alternative design

method, one in which the geometric rigour of parametric modelling can be deployed first to integrate manufacturing constraints, assembly logics and material characteristics in the definition of simple components, and then to proliferate the components into larger systems and assemblies. This approach employs the exploration of parametric variables to understand the behavior of such a system and then uses this understanding to strategize the system's response to environmental conditions and external forces" To work with complex objects, a design process usually starts from a very simple first level and then other layers are added; complex forms are comprised of different hierarchies, each associated with its own logic and details. These levels are also interconnected and their members affect each other and in that sense this method called 'Associative'. Generally speaking, Associative Modelling relates to a method in which elements of design being built gradually in multiple hierarchies and at each level, some parameters of these elements being extracted to be the generator for other elements in the next level and this goes on, step by step to produce the whole geometry. So basically the end point of one curve could be the center point of another circle and any change in the curve would change the circle accordingly. Basically this method of design deals with the huge amount of data and calculations and happens through the flow of algorithms.

The point is that all these geometries are easily adjustable after the process. Designer always has access to the elements of design product from start point up to details. Actually, since the design product is the result of an algorithm, inputs of the algorithm could be changed and the result would also be updated accordingly. It is now possible to digitally sketch a model and generate hundreds of variations of project by adjusting very basic geometrical parameters. It is also viable to embed the properties of material systems, fabrication constraints and assembly logics in parameters. It is also possible to respond to the environment and be associative in larger sense. "... Parametric design enables the recognition of patterns of geometric behavior and related performative capacities and tendencies of the system. In continued feedback with the external environment, these behavioral tendencies can then inform the ontogenetic development of one specific system through the parametric differentiation of its sub-locations".

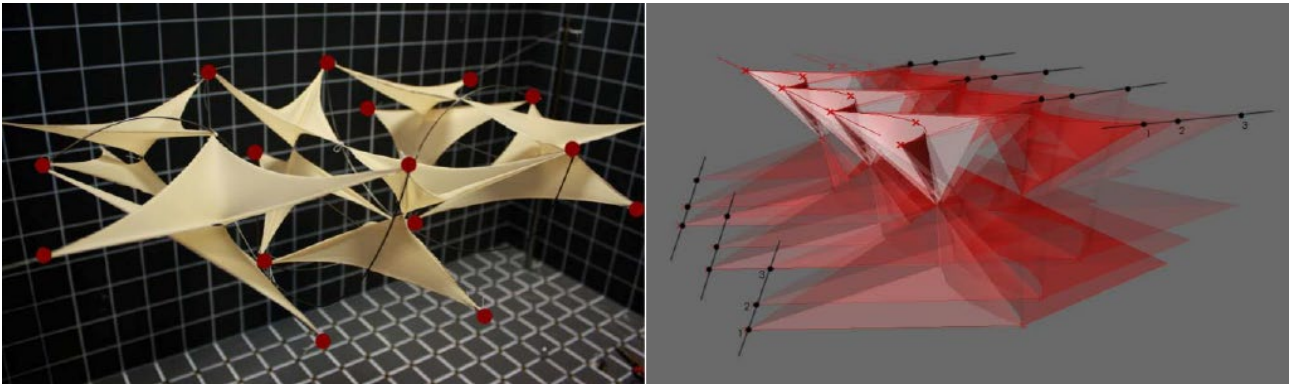


Figure 24 – A. form-finding in membranes and minimal surfaces, physical model, B. membrane's movement modelled with Grasshopper, Zubin Mohamad Khabazi, EmTech Core-Studio, AA, Conducted by Michael Hensel and Achim Menges, fall 2008.

Grasshopper is a platform in Rhino to deal with these Generative Algorithms and Associative modelling techniques. The following chapter are designed in order to combine geometrical subjects with algorithms to address some design issues in architecture in an ‘Algorithmic’ method. The idea is to broaden subjects of geometry and use more commands and examples are designed to do so.

3.3.2 Introduction to Genetic Algorithms

GA was inspired from the Darwinian theory of evolutionary in which the survival of fitter creature and their genes were simulated. GA is a population-based algorithm. Every solution corresponds to a chromosome and each parameter represents a gene. GA evaluates the fitness of each individual in the population using a fitness (objective) function. For improving poor solutions, the best solutions are chosen randomly with a selection (e.g. roulette wheel) mechanism. This operator is more likely to choose the best solutions since the probability is proportional to the fitness (objective value). What increases local optima avoidance is the probability of choosing poor solutions as well. This means that if good solutions be trapped in a local solution, they can be pulled out with other solutions.

The GA algorithm is stochastic, so one might ask how reliable it is. What makes this algorithm reliable and able to estimate the global optimum for a given problem is the process of maintaining the best solutions in each generation and using them to improve other solutions. As such, the entire population becomes better generation by generation. The crossover between individuals results in exploiting the ‘area’ between the given two parent solutions. This algorithm also benefits from

mutation. This operator randomly changes the genes in the chromosomes, which maintains the diversity of the individuals in the population and increases the exploratory behavior of GA. Similar to the nature, the mutation operator might result in a substantially better solution and lead other solutions towards the global optimum.

Initial Population

The GA algorithm starts with a random population. This population can be generated from a Gaussian random distribution to increase the diversity. This population includes multiple solutions, which represent chromosomes of individuals. Each chromosome has a set of variables, which simulates the genes. The main objective in the initialization step is to spread the solutions around the search space as uniformly as possible to increase the diversity of population and have a better chance of finding promising regions. The next sections discuss the steps to improve the chromosomes in the first population.

Selection

Natural selection is the main inspiration of this component for the GA algorithm. In nature, the fittest individuals have a higher chance of getting food and mating. This causes their genes to contribute more in the production of the next generation of the same species. Inspiring from this simple idea, the GA algorithm employs a roulette wheel to assign probabilities to individuals and select them for creating the next generation proportional to their fitness (objective) values. Figure 25 illustrates an example of a roulette wheel for six individuals. The details of these individuals are presented in Table 1.

It can be seen that the best individual (#5) has the largest share of the roulette wheel, while the worst individual (#4) has the lowest share. This mechanism simulates the natural selection of the fittest individual in nature. Since a roulette wheel is a stochastic operator, poor individuals have a small probability of participating in the creation of the next generation. If a poor solution is 'lucky', its genes move to the next generation. Discarding such solutions will reduce the diversity of the population and should be avoided.

It should be noted that the roulette wheel is one of the many selection operators in the literature. Some of the other selection operators are:

- Boltzmann selection
- Tournament selection

- Rank selection
- Steady state selection
- Truncation selection
- Local selection
- Fuzzy selection
- Fitness uniform selection
- Proportional selection
- Linear rank selection
- Steady-state reproduction

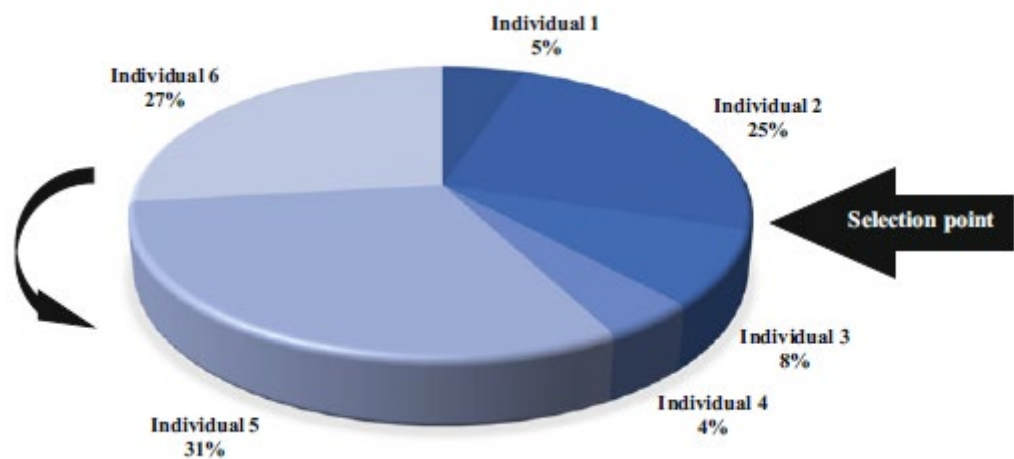


Figure 25– Mechanism of the roulette wheel in GA. The best individual (#5) has the largest share of the roulette wheel, while the worst individual (#4) has the lowest share

Individual number	Fitness value	% of Total
1	12	5
2	55	24
3	20	8
4	10	4
5	70	30
6	60	26
Total	227	100

Table 1– Details of the individuals in Fig. 31.. The fittest individual is Individual #5

Crossover (Recombination)

After selecting the individuals using a selection operator, they have to be employed to create the new generation. In nature, the chromosomes in the genes of a male and a female are combined to produce a new chromosome. This is simulated by combining two solutions (parent solutions) selected by the roulette wheel to produce two new solutions (children solutions) in the GA algorithm. There are different techniques for the crossover operator in the literature of which two (single-point and double-point) are shown in Figure 26.

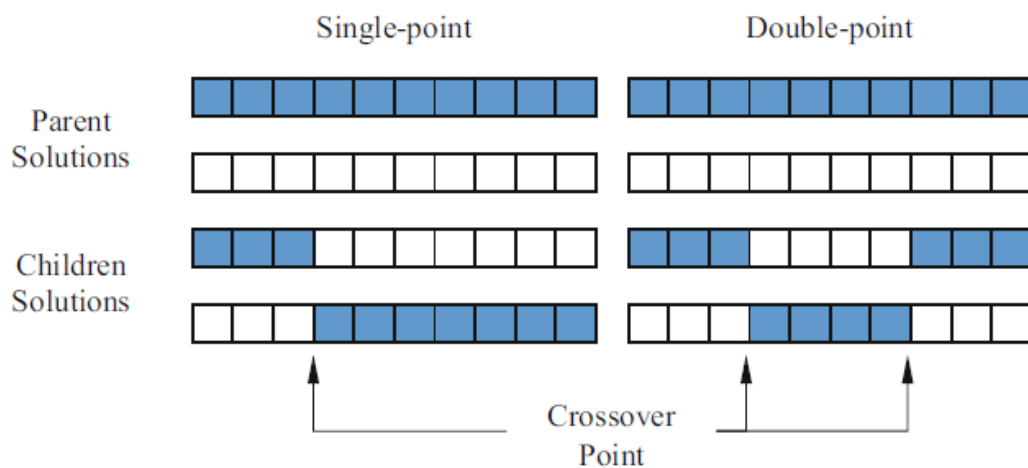


Figure 26– Two popular crossover techniques in GA: single-point and double point.

In the single-point cross over, the chromosomes of two parent solutions are swapped before and after a single point. In the double-point crossover, however, there are two cross over points and the chromosomes between the points are swapped only.

Other crossover techniques in the literature are:

- Uniform crossover
- Half uniform crossover
- Three parents crossover
- Partially matched crossover
- Cycle crossover
- Order crossover
- Position-based crossover
- Heuristic cross over

- Masked crossover
- Multi-point crossover

Mutation

The last evolutionary operator, in which one or multiple genes are altered after creating children solutions. The mutation rate is set to low in GA because high mutation rates convert GA to a primitive random search. The mutation operator maintains the diversity of population by introducing another level of randomness. In fact, this operator prevents solutions to become similar and increase the probability of avoiding local solutions in the GA algorithm. A conceptual example of this operator is visualized in Figure 27. It can be seen in this figure that slight changes in the some of the randomly selected genes occur after the crossover (recombination) phase.

Some of the popular mutation techniques in the literature are:

- Power mutation
- Uniform
- Non-uniform
- Gaussian
- Shrink
- Supervised mutation
- Uniqueness mutation
- Varying probability of mutation

Taken together, most of EAs use the three evolutionary operators: selection, crossover, and mutation. These operators are applied to each generation to improve the quality of genes in the next generation. Another popular evolutionary operator is elitism, in which one or multiple best solutions are maintained and transferred without modification to the next generation. The main objective is to prevent such solutions (elites) from being degraded when applying the crossover or mutation operators.

The GA algorithm starts with a random population of individuals. Until the end of the end criterion, this algorithm improves the population using the above-mentioned three operators. The best solution in the last population is returned as the best approximation of the global optimum for a given problem. The rate of selection, crossover, and mutation can be changed or set to fix numbers during the

optimization. The next sections investigate the impact of changing such rates on the performance of GA.

3.4 Topology and Form Finding via Genetic Algorithms

The following presents an approach to early applications of the Galapagos program as a means to optimize structural forms. The process was conducted with Rhino's Grasshopper program, the structural analysis plug-in, Karamba, and the genetic algorithm solver, Galapagos. This topological form finding process was based on flexible parameters that modified brace and column locations, and diaphragm size and positions.

This process worked by having Galapagos modify a parametric model which had initial randomly generated variables for the genomes. After structural analysis, Galapagos was tasked with changing the form in order to minimize overall displacement of the structure. Being an evolutionary solver, Galapagos creates a "population" of solutions and eliminates non-effective offspring to continue breeding effective offspring through multiple generations. This means that solutions found through Galapagos were best fit to the program, but were not necessarily an absolute perfect solution, as that could take hundreds of generations to find. This also means solutions vary based on the beginning placement of genomes before populations are created. However, after comparing Galapagos to what was intuited and what are known structural solutions, there is a strong case to be made that Grasshopper, Karamba, and Galapagos can be used effectively in engineering practice to create both beautiful and efficient structures.

3.4.1. Grasshopper and Karamba

Grasshopper is a visual coding environment within Rhino3D composed of pre-coded components placed on a canvas that interact with the Rhino modeling space. Grasshopper is unique in that there is no traditional code writing and there is no code to "run." Instead all components are constantly running, so any changes can be seen in real time within Rhino.

Karamba is a structural analysis plugin for Grasshopper that can perform many of the same tasks as a traditional analysis program. The main benefit of Karamba is that it can be used to find and create more optimal forms and material placement.

3.4.2. Basic Structural Elements

A simply supported beam and a portal frame modeled with Grasshopper and analyzed using Karamba. The results are what we would expect to see based on given loads. Both the geometry and load placement of both elements can be changed and results will be shown in real time.

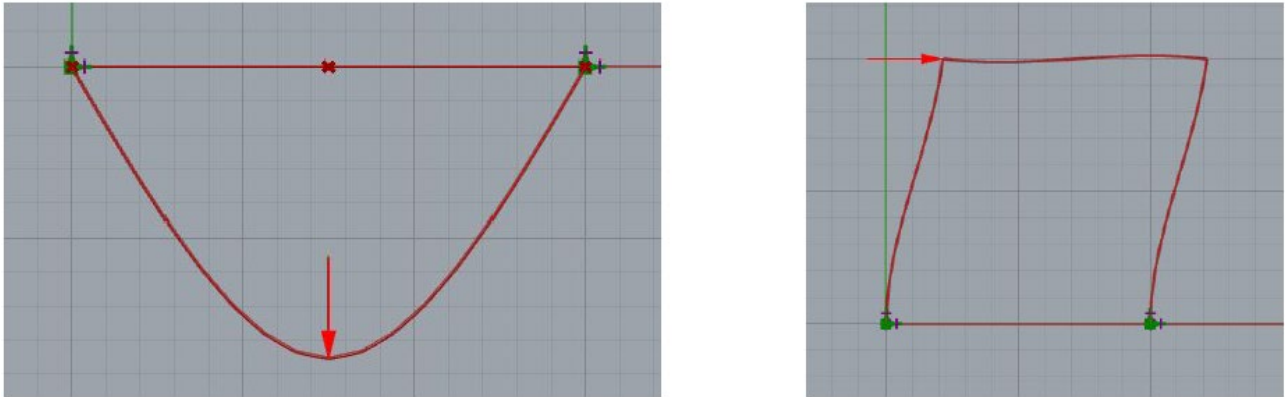


Figure 27– Simple Structures Analyzed with Karamba

A truss and a shear wall are shown here. Dimensions and the number of bays for the truss can be changed instantly. Karamba was used to paint the principal stress lines shown on the shear wall; blue for compression and green for tension.

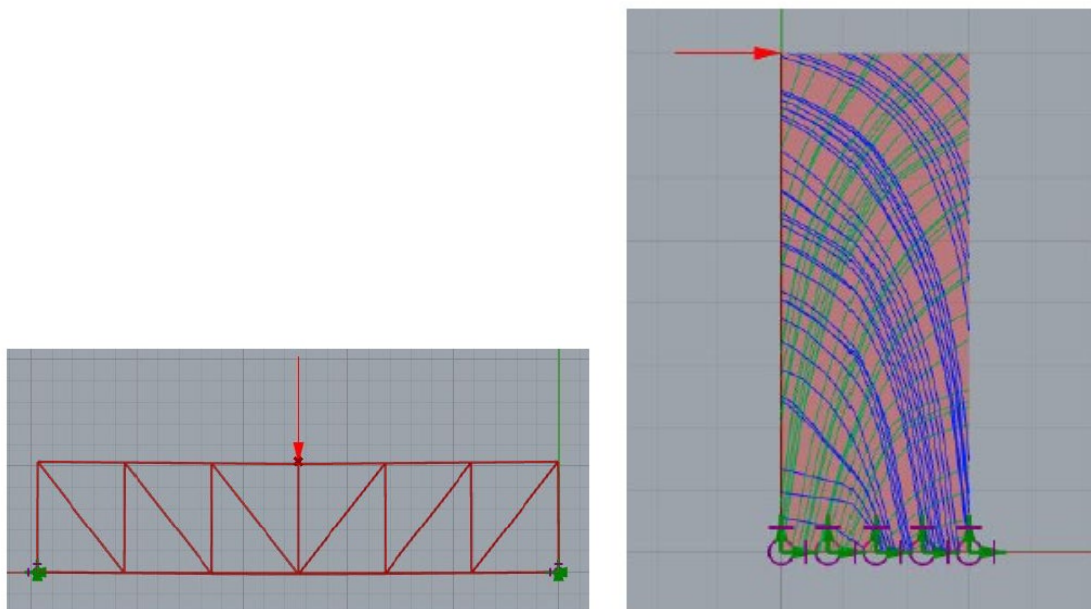


Figure 28– Simple Structures Analyzed with Karamba – 2

3.4.3. Karamba Form Finding

In Figure 29 it can be seen that a grid of beams supported at the green points shown and subjected to a simply gravity load across the whole structure. Using Karamba, the moment at each point along the beam was calculated. From the moment diagram, rectangles were extruded along the length of the beam. This can be translated to the idea that the most material should be placed where the rectangles are the tallest. This is a basic example of form finding and topology optimization, and moving forward the goal was to automate this process.

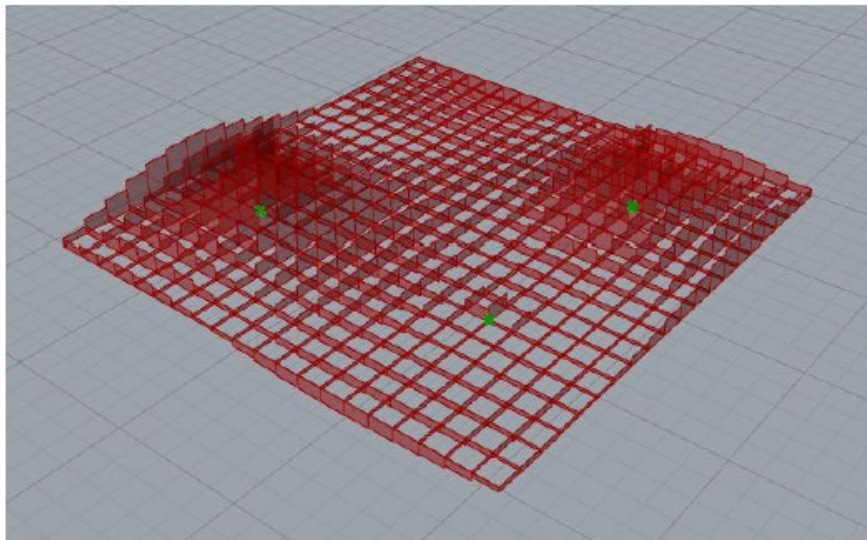


Figure 29– Basic form finding with Karamba

3.4.4 Topology Process

In the Figure 30, it can be seen that the graphic analysis process within the Grasshopper. This process can be summarized as step by step:

- Grasshopper Input creates a visual model in Rhino space, changes seen in real time
- Karamba plugin allows model to be structurally analyzed
- Galapagos uses Karamba output and geometric input to change parameters and find what it deems the most fit solution

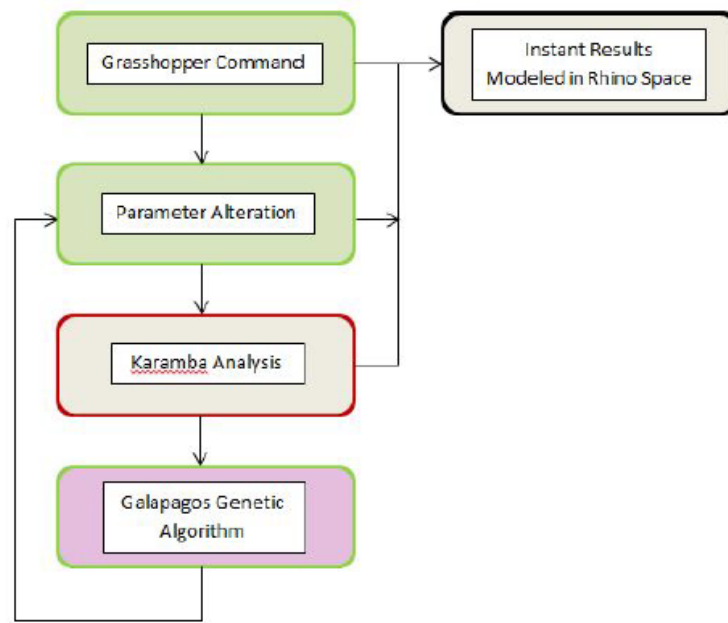


Figure 30– Graphic Analysis Process of Grasshopper

3.4.5. Genetic Algorithms in Grasshopper

A genetic algorithm is solver that uses “evolutionary techniques.” This is done by generating a population of solutions based on genomes (variable subject to change) reacting to a fitness (desired parameter to be minimized or maximized). An effective solution is found, keeping “fit” genomes in a generation and breeding them with other favorable genomes in the following generation, as well as eliminating non-favorable solutions. This process is very similar to natural selection in the real world since Galapagos iterates, or breeds, multiple generations of solutions until it finds what it believes to be the fittest solution. The image to the left shows the basic genetic algorithm process while Pugnale’s flowchart in the Figure 31 delves more into Galapagos’ specific process.

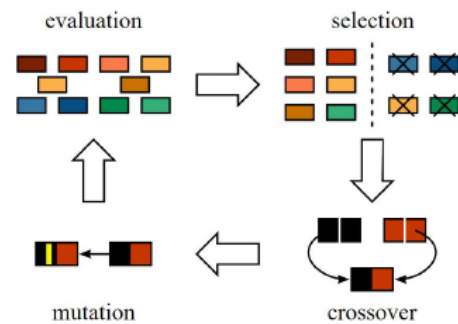
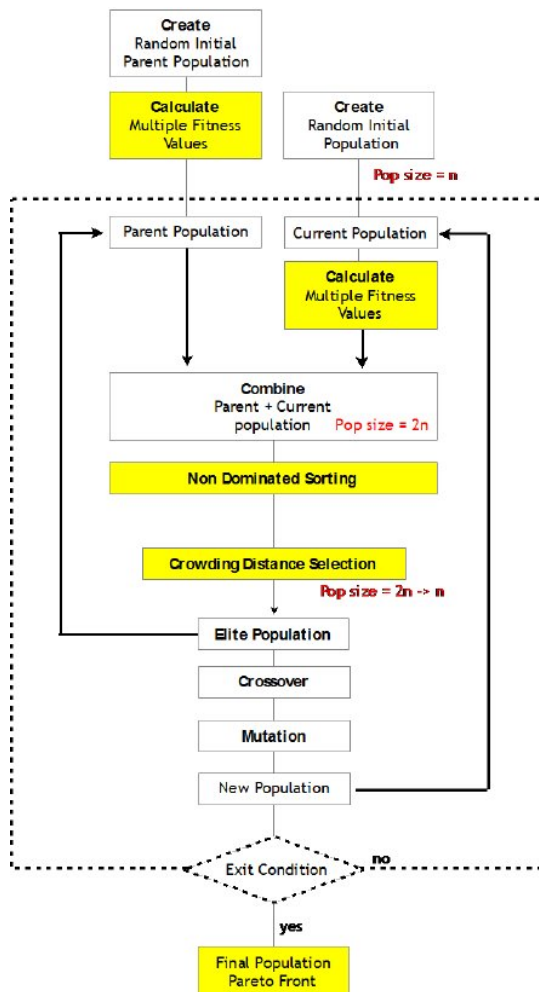


Figure 31– Genetic Algorithm, Pugnale's Flowchart

3.4.6. Galapagos Function

In the Figure 32, first graph shows the total number of generations being bred (x-axis) vs the total spread of genome solutions being tested (yellow region/y-axis). The red line shows the average solution in each generation and the orange region is the standard deviations away from the average solution. The bottom left image shows the total spread of solutions in relation to each other and the bottom middle similarly is the disparity between values on sliders compared to other solutions.

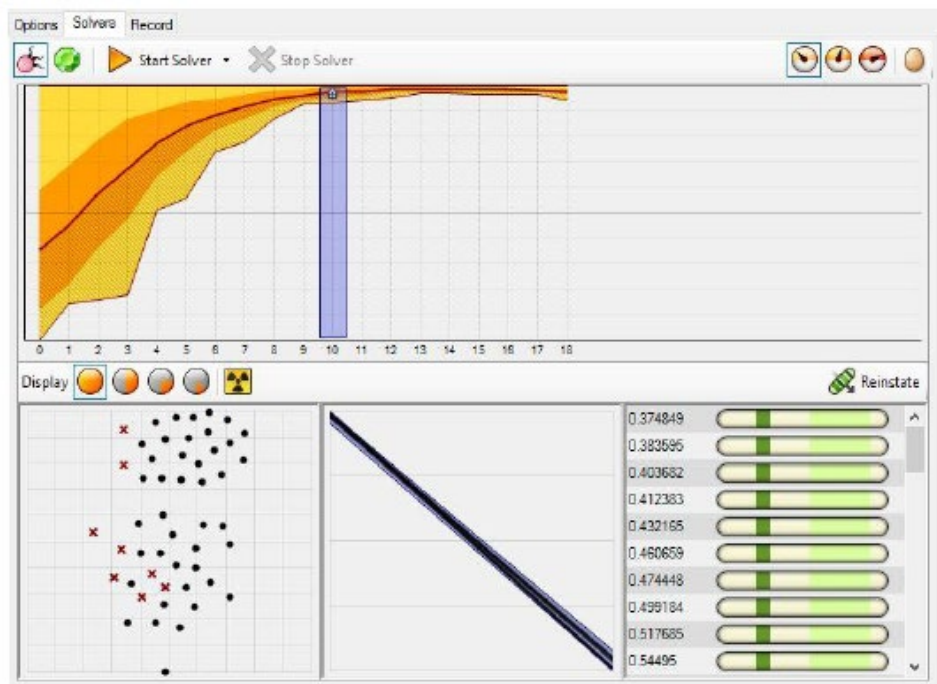


Figure 32– Galapagos Function

The Figure 33 shows Galapagos connecting to sliders (maroon arrows) which alter parameters in this file. The fitness connection (green arrow) decides what parameter should be minimized or maximized. In our case, fitness connected to deflection of the structure.

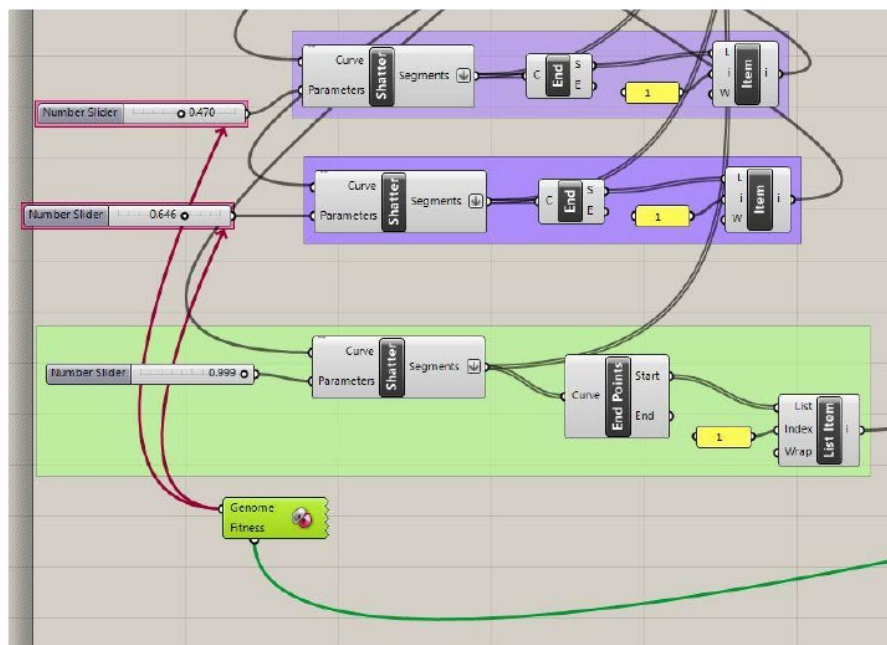


Figure 33– Galapagos Connections

3.4.6. Examples of Optimization with Galapagos Function

In Figure 34 example of Galapagos with moving point loads to find a maximum moment in a beam by combining Galapagos with Karamba. At the end of operation Galapagos ended up placing all loads on top of each other at the center of the beam, so we felt comfortable moving forward.

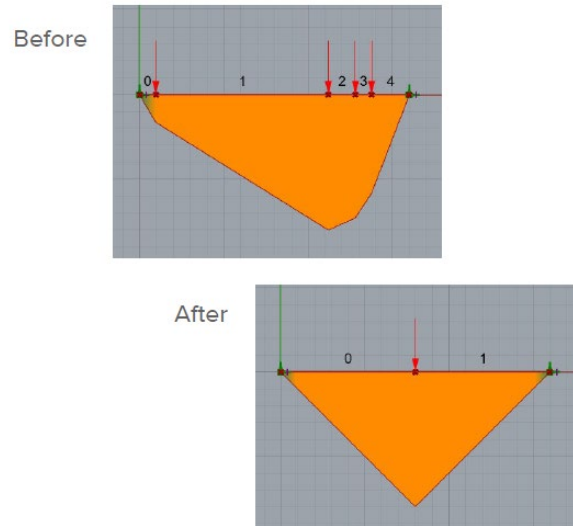


Figure 34– Galapagos application for Influence Line theory

In next example (Figure 35.) it can be seen a test on a simple braced frame, now with two adjustable variables: the location of either end of the brace along the frame. Fitness is minimization of deflection. Galapagos created the basic braced frame we see every day.

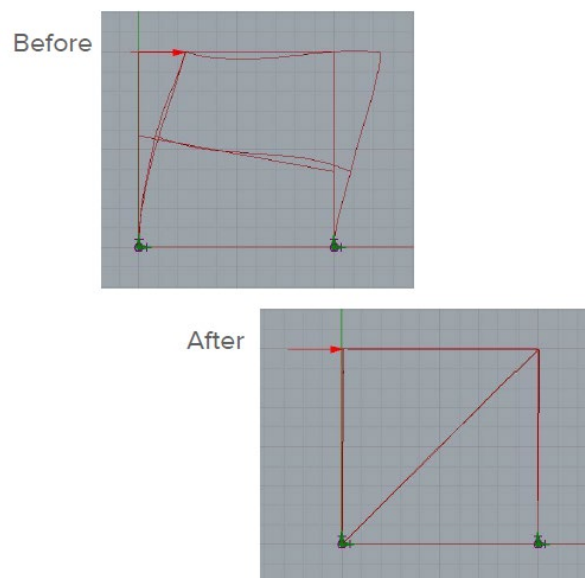


Figure 35– Galapagos application for a basic braced frame

In Figure 36, a more complex structural form: 2 stories, 5 editable braces, a central opening, as well as moving interior supports and columns can be seen. This type of form was to expand on the original, simple form while still utilizing increased amount of genomes on top of incorporating a hypothetical opening an architect/designer might task an engineer to design around. This shows us how many more iterations it takes for Galapagos to find a reasonable solution when given a multitude of variables to adjust. This was also example of seeing how an automated, genetic solver found a solution that doesn't seem logical if an engineer was tasked with reducing deflection in this system with the same parameters.

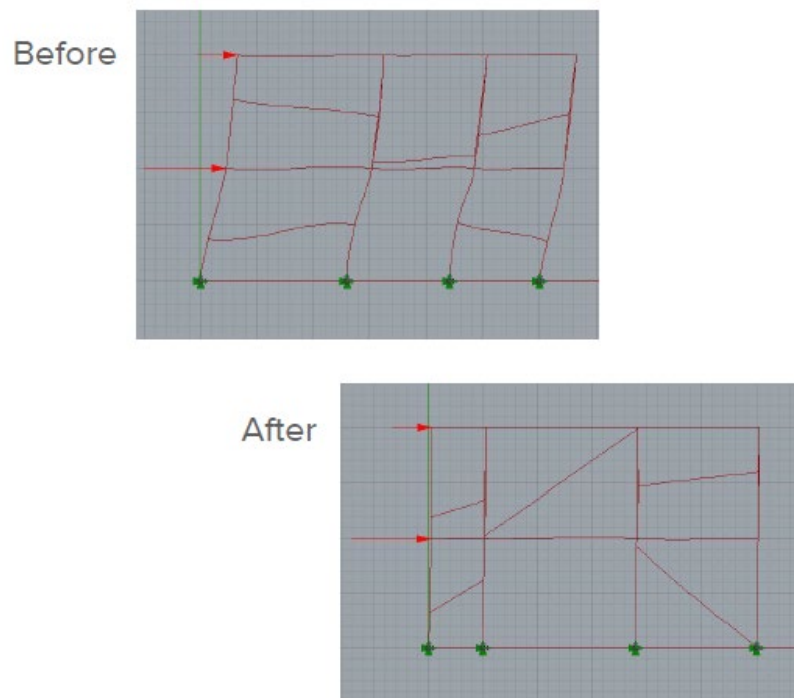


Figure 36– Galapagos application for a complex braced frame

3.5 Optimization vs. Adaptation: Multi-Parameter Optimization

This analysis of multi-parameter optimization systems hypothesizes their potential roles in the development and implementation of architectural designs. Following a look into how single-parameter optimization operates in Galapagos, the ‘evolutionary solver’ plug-in for the programme. Grasshopper, this post delves into the complex operations Galapagos was created to solve: those that must balance competing objectives. This chapter also aims to provide insight into another, lesser-known Grasshopper plug-in called Octopus as a means of simplifying and explaining systems of

multi-parameter optimization. A follow-up analysis discusses the capacity for adaptation within these systems of optimization.

Over the past few decades, facade design and building forms have evolved significantly through the application of environmental simulations and genetic algorithms. Derived from parametric studies of building performance, optimized systems have become commonplace and indicate the growing importance of sustainable considerations in facade design. Environmental performance analytics like Ladybug and Honeybee for Grasshopper have aided immensely in this development. Facade solutions offer efficient and pragmatic solutions that mitigate harsh climatic conditions affecting buildings and their occupants. Facade optimization, though calculated for efficiency, fails to address the variety of external conditions, such as sunlight, precipitation and changing occupancy, that affect both a building and its users. Parametric design in architecture can go beyond facade optimization by combining multi-objective optimization and efficient adaptability. Systems that achieve this combination can continuously self-optimize by account for fluctuations in performance parameters that affect their ability to compensate for climatic conditions.

3.5.1 Single Parameter Optimization

From the onset of the sustainable design movement, designers have favored optimized design solutions for their ability to distil information and provide efficient, cost-effective and pragmatic solutions for design scenarios. Facade optimization is a means by which a designer can establish a single design solution that will allow the system to perform best under the full range of exposures it will endure. It comprises static systems that rely on a convergence of data points to highlight the best formation possible based on concrete objectives. The plug-in Galapagos can analyses only one parameter in a given run, and though that run generates many numeric outputs, it will gradually approach a single best solution. This type of analysis is useful for systems of optimization as it can test thousands of options and filter through even the most subtle changes in input data.

In this example of single-parameter optimization, a simple script intends to produce an output with the smallest possible volume. Given three blocks with varying origin points and heights, Grasshopper continuously adjusts the input sliders to slowly create an output that best meets, or optimises, the goal outcome.(Figure 37.)

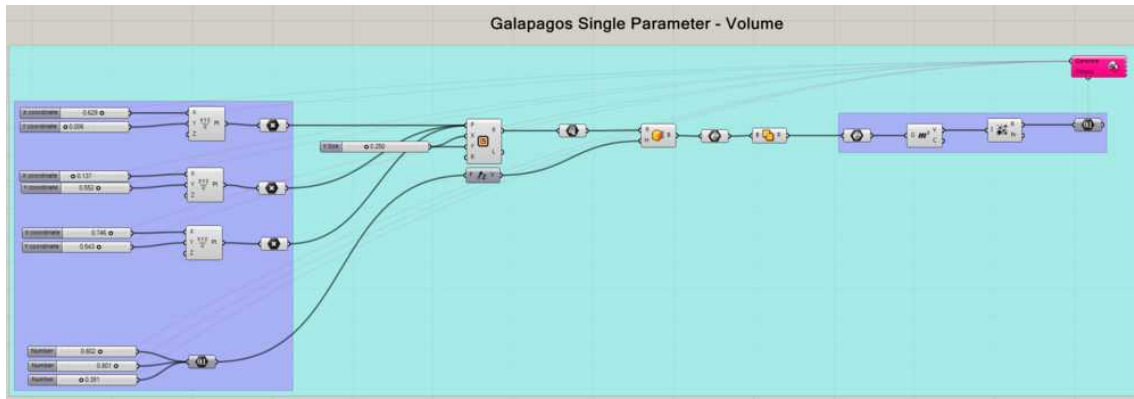


Figure 37– Single Parameter Optimization

In chronological order, these three generations of solutions illustrate Galapagos' ability to iteratively minimize total volume.(Figure 38.)

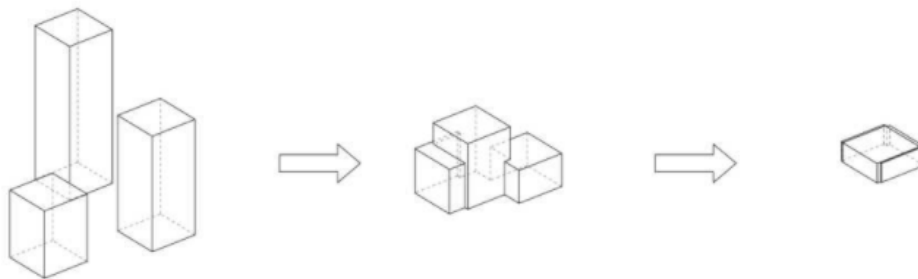


Figure 38– Single Parameter Optimization Illustration

Under restrictive parameters, optimized facade systems function as a single application. They act as broad-stroke techniques, attempting to provide an effective solution to as much of the problem as possible. These facades, though optimized to meet specific criteria when first designed, fail to adapt to changing environmental conditions or changing user requirements. Once finalized, they are locked into their configurations and any fluctuations in environmental impact need to be compensated for with internal conditioning systems. One way to manage multiple and changing variables is through multi-parameter optimization.

3.5.2 Multi Parameter Optimization

It is impossible to reduce the environmental factors that impact a building to a single parametric relationship. In reality, the factors that affect a building, its envelope and its ability to serve as an

effective system of climate control are multi-dimensional. Parametric simulators can aid in understanding variation in environmental performances of a single parameter, but are unable to optimize adequately for all factors in play. Multi-parameter optimization can achieve median-optimality across a variety of conditions while ensuring a sufficient minimum performance for each condition. When coordinating a building's programme or designing its facade, multi-parameter optimization can act as an instrumental tool to mediate conflicting goals of environmental control, financial costs, client needs and occupant comfort. When designing an effective building system, there is much to take into account, and optimization in some areas could mean unfortunate compromise in others. Multi-parameter optimization is a simple concept. Consider the scenario shown in Figure 39; a series of blocks of varying heights and base points must produce the minimum possible volume and maximum possible surface area simultaneously. In this case, neither parameter can achieve its maximum potential without hindering the maximum performance of the other objective. Example 1 optimized for minimum volume, but in so doing significantly reduced its maximum surface area; the opposite occurred in Example 3. Through multi-parameter optimization, a constant cross-referencing of parameters creates a desirable median optimality for all objectives, thus adequately compensating for each performance measure simultaneously. Multi-parameter optimization can resolve far more than two criteria simultaneously to create an equally dispersed favorability among competing parameters; note the 'favorable median' in Examples 2a, 2b and 2c.

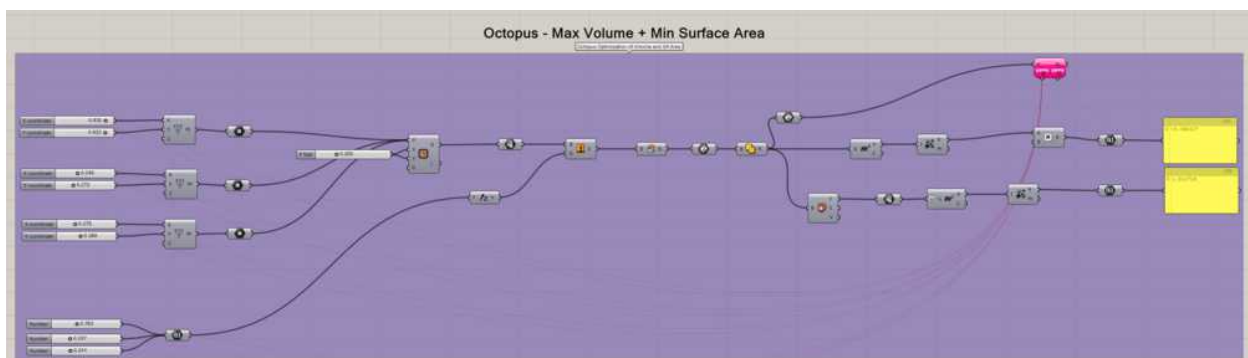


Figure 38– Unlike the previous scenario of single-parameter optimization, multi-parameter optimization in this example allows for two competing objectives to be optimized proportionally.

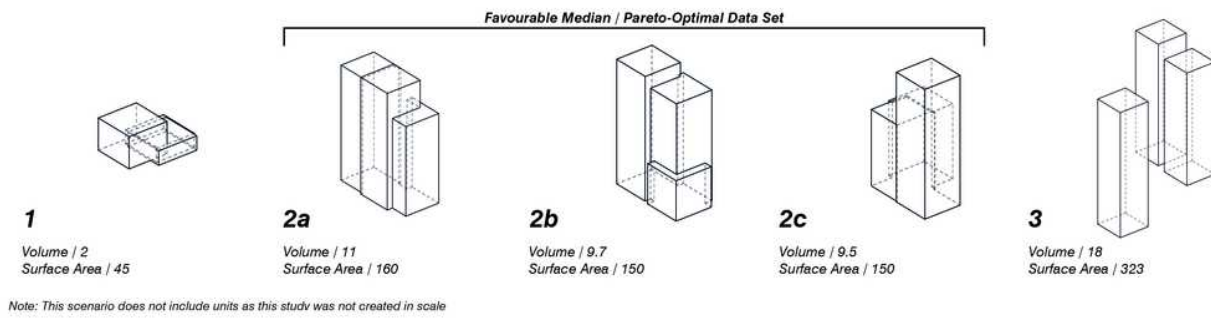


Figure 39– Multi Parameter Optimization Illustration

While the script and analysis are essentially identical to the prior example, this scenario uses the plug-in Octopus instead of Galapagos. Octopus allows two objectives to be analyzed simultaneously.

3.5.3 Octopus Function

Similar to the plug-in Galapagos, Octopus is an evolutionary simulator that can approach optimal solution sets through iterative tests and constant self-adaptation. Unlike Galapagos, however, Octopus possesses the ability to cross-reference multiple parameters simultaneously, whereas Galapagos is limited to a single parametric input. The script below has been broken down into three key components that both inform the analysis and allow the system to run effectively. The pink component to the right is the Octopus plug-in. Similar to the Galapagos plug-in, Octopus requires the same inputs, but as mentioned above, allows the flexibility to input multiple objectives instead of just one. Once Octopus has collected data, it begins to map the information on a coordinate grid that is setup based on the objectives set. Here, one can access the full range of data and separate the solutions that fall in the favorable median from those that do not. After sorting, one can ‘re-instate’ the favorable solutions, or in other words, select their specific data points to appear in Grasshopper, as shown in Figure 40.

Multi-parameter optimization with Octopus can generate far more advanced solution sets than simple box scenarios. This script creates a sophisticated analysis to balance the goals of maximizing volume and minimizing surface area and direct sunlight. With each iteration of the analysis, the script develops a potential building scheme and notes its outcome relative to these three parameters.

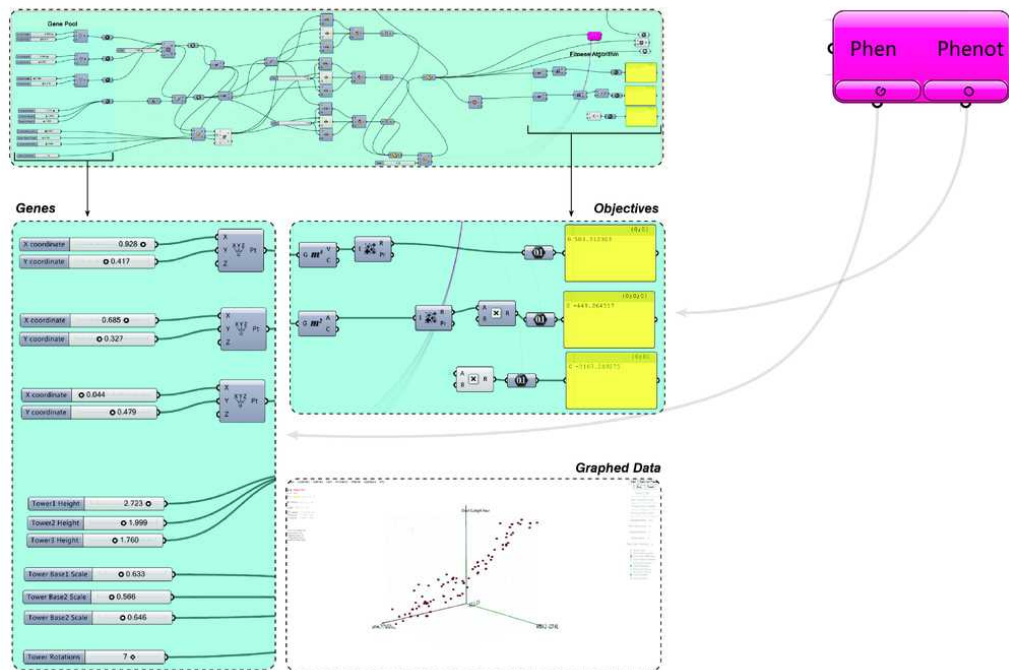


Figure 40– Multi Parameter Optimization with Octopus

Similar to the block example above, the same data trend appears in the tower example below. The towers in Category 1 could be considered the optimal solutions as they adequately balance all three parameters. Note that there is a narrow but distinct range of options amongst these optimal solutions. Each of these solutions falls within the most desirable range of outcomes, but individually possesses its own advantages and disadvantages that would make it more or less favorable for further design development. Categories 2 and 3 represent extremes in the data. Notice in Category 2 how the minimization of surface area and direct sunlight is ideal, but the volume is compromised greatly.

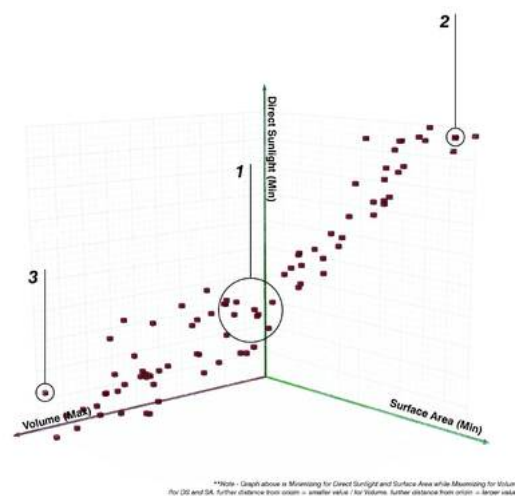


Figure 41– Octopus Optimisation

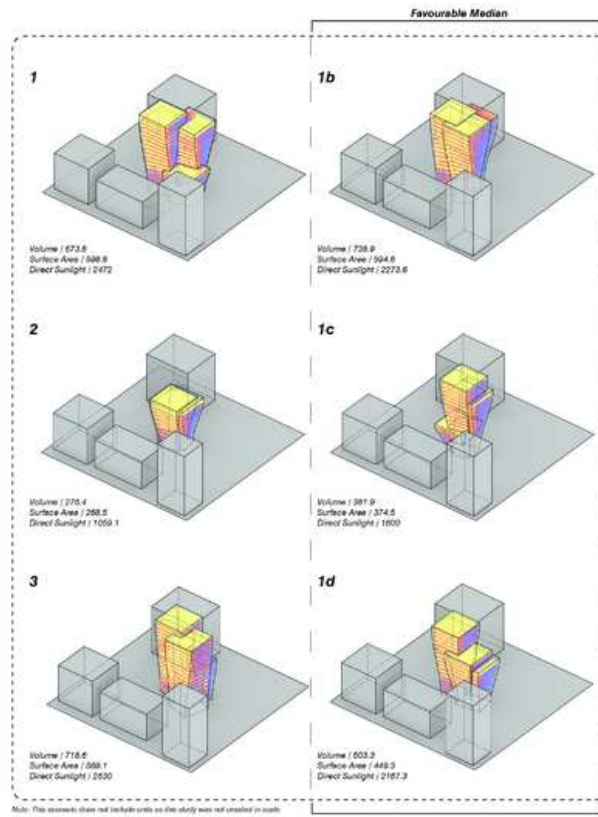


Figure 42– Multi Parameter Optimisation Illustration

4. TOPOLOGICAL, DIMENSIONAL AND SHAPE

OPTIMIZATION: THE CASE STUDY

The aim of this thesis is to create a parametric model of an existing long span grid-roof structure and optimize it in terms of mass, maximum displacement (topological optimization) and cross-section of the elements (dimensional and shape optimization).

A key step to find optimal design is to run a reliable "initial" solution that summarizes all the design criteria that will be executed until the final stage. This is one of the most important phases of the design, called "Conceptual Design", which represents precisely the preliminary phase of the architectural project to obtain the final product. Conceptual Design phase is an early phase of the design process, in which the broad outlines of function and form of something are articulated. It includes the design of interactions, experiences, processes, and strategies. It involves an understanding of people's needs - and how to meet them with products, services, and processes.

For the optimization process, a specific algorithm has been implemented in the thesis work (Figure 43), which will be described in detail. Following software/plugin have been used for the optimization of the case study.

- 1) *Grasshopper*: for creation of parametric model by visual programming;
- 2) *Karamba 3D*: plug-in used to perform FEM analysis;
- 3) *Octopus* : Estimation algorithm for multi-objective optimization.

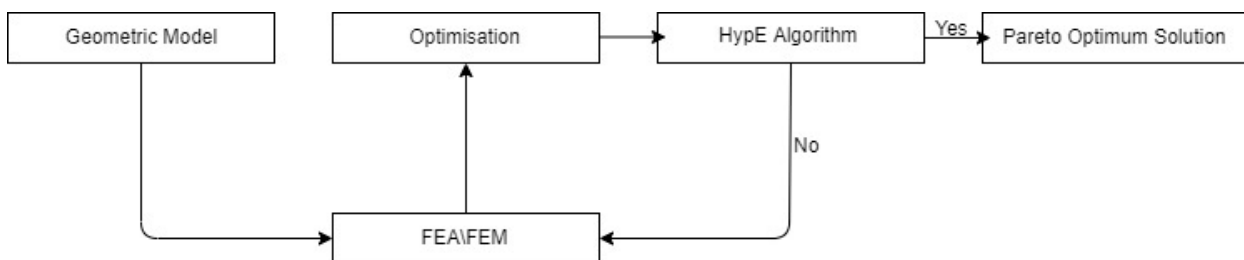


Figure 43– Flowchart of the used algorithm

4.1 Case Study: A Long Span Grid-Roof Structure

Chosen case study is a 120 years old, long span grid roof structure located in Vyksa, Moscow which is an industrial town. This structure also known as the world's first double curvature steel diagrid structure. In 1897, the Russian engineer Vladimir Shukhov, who is known with his famous design of Shukhov Tower, designed this building for a metallurgical industry in Vyksa. He was inspired by his previous project which he designed few years earlier in Grozny.

Already in the Grozny project, Shukhov managed to conceive a very light-weight roof, built by assembling very thin Z-section rods, all identically curved to circular arcs. That innovative structural scheme was perfect for prefabrication because it was made up of identical elements, was renamed with the explanatory name of “roof without trusses”. After this design this elements had been used to build almost all the industrial buildings of the late 19th century.



Figure 44– Photo of the structure during the construction

Grozny's truss-less roof scheme, could not be replicated identically to Vyksa because: it was, in fact, necessary that each rod of the trellis had lateral support. In other words, Grozny's barrel vault needed a series of columns placed on the sides of the roof.

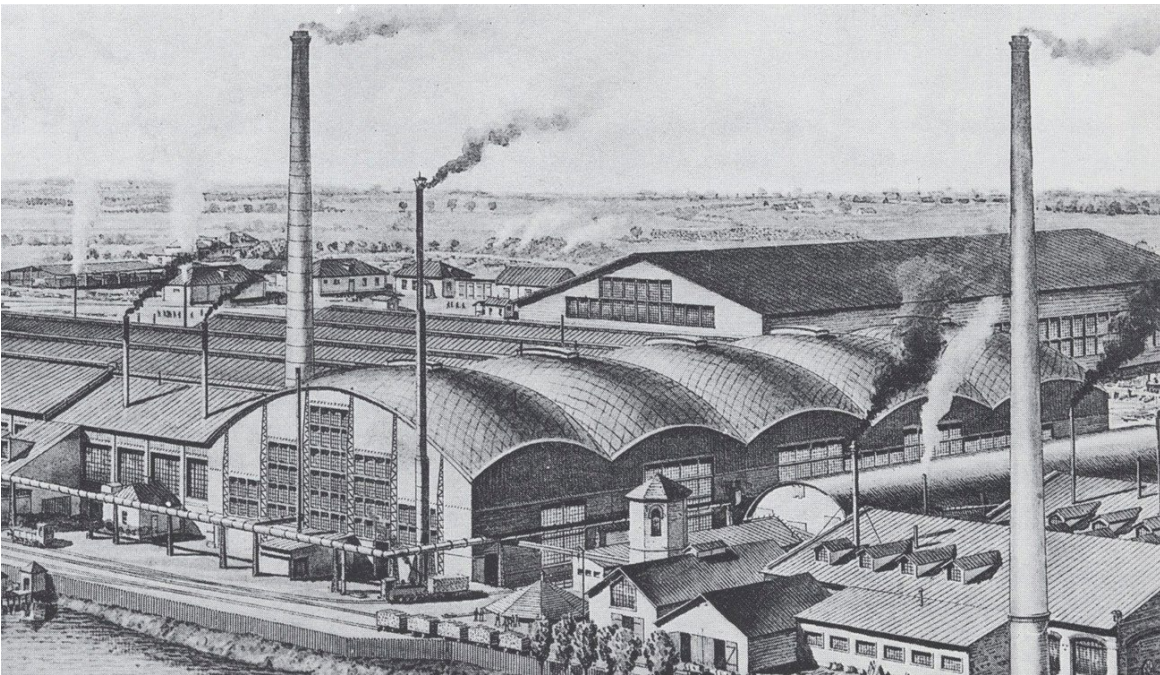


Figure 45– Current Photos of the roof of structure and image of the structure in function

4.1 Development of a Parametric Model Supported By Algorithm

To be able to make optimization a parametric model of the case study have been created by using visual programming with the Grasshopper by setting historical design documents as a reference. Historical documents of the design can be seen in the Figure 46, 47 and 48.

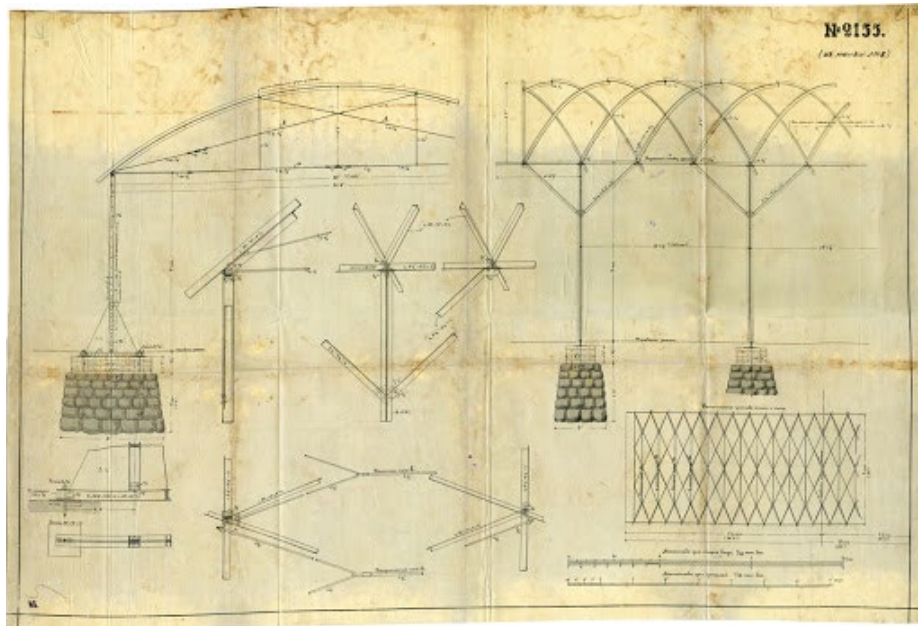


Figure 46— Original document of the design of the structure, a

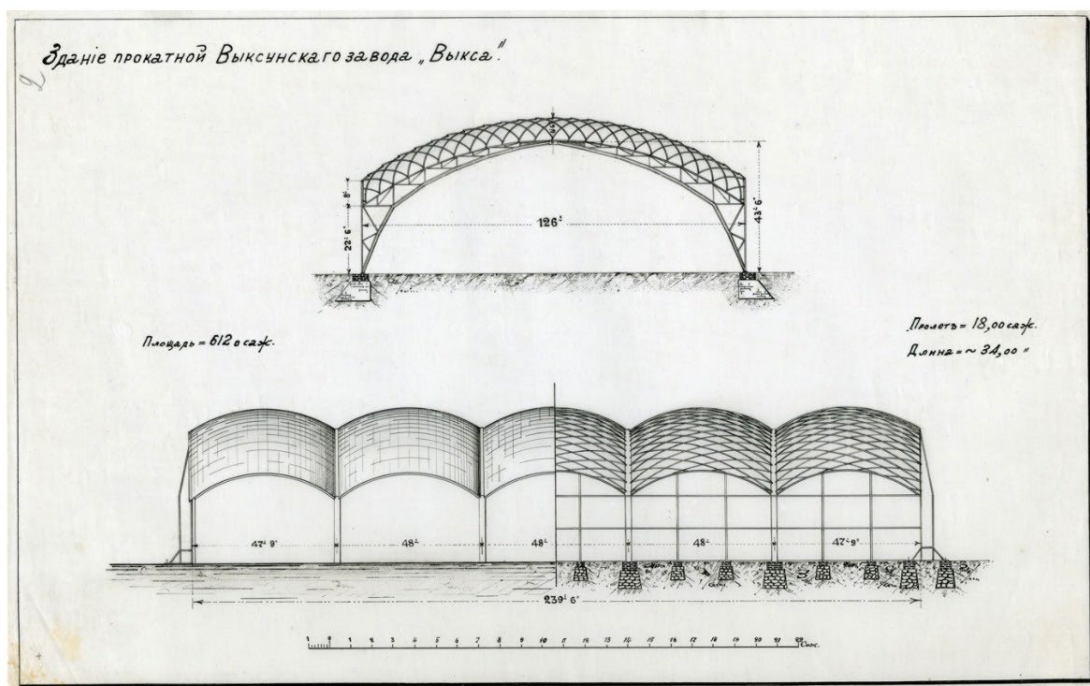


Figure 47— Original document of the design of the structure, b

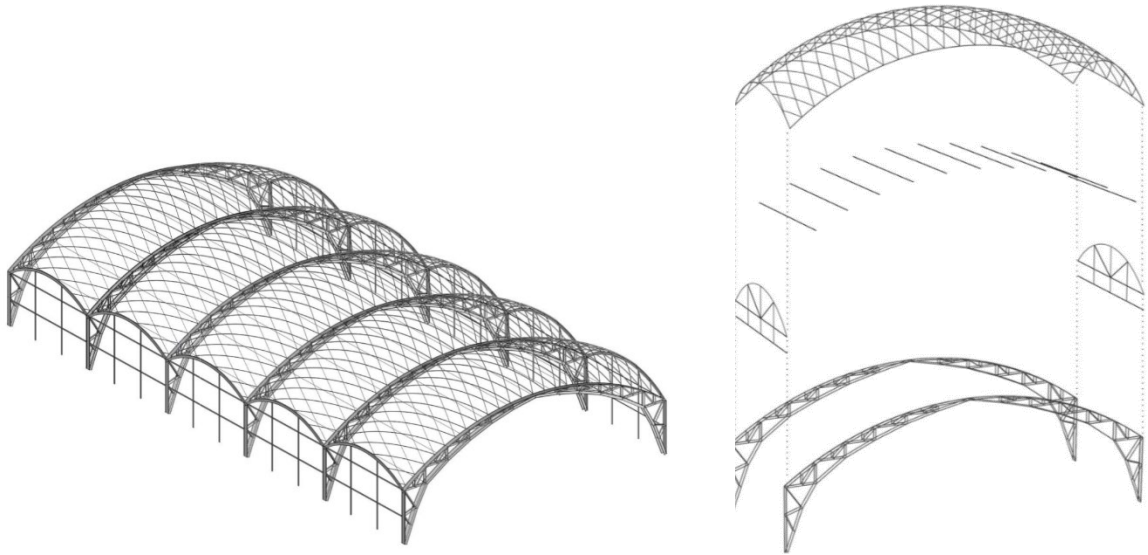


Figure 48– Original design of the structure

A complete parametric model of the structure have been created in order to be used in the FEA and optimization (Figure 49).

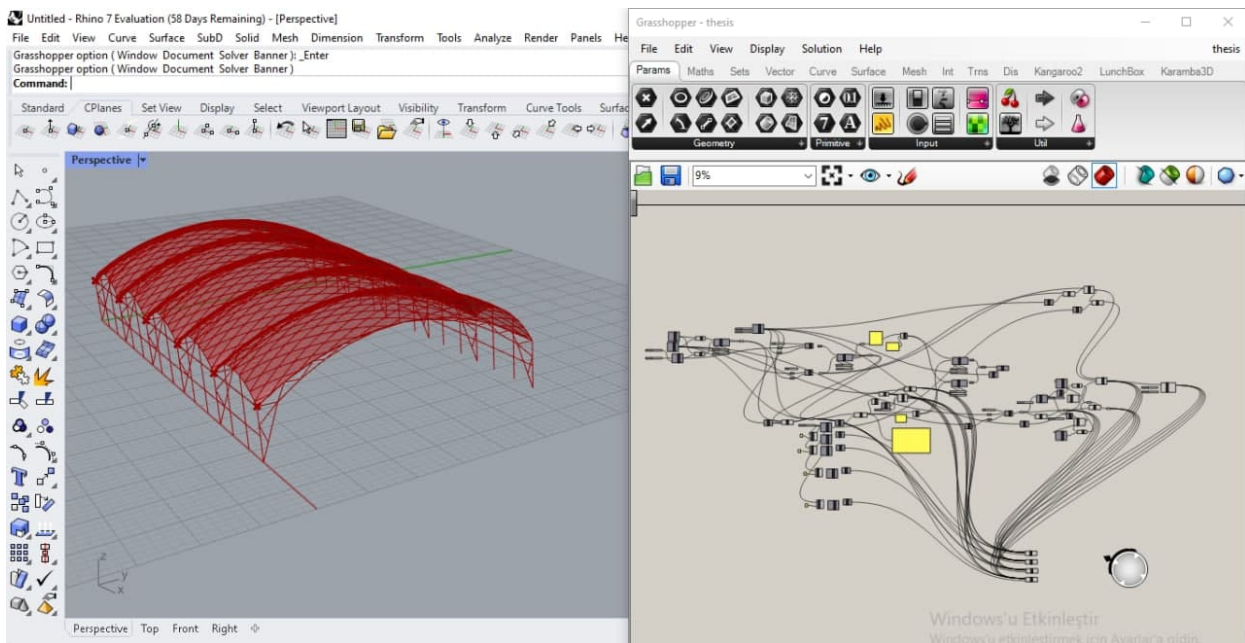


Figure 49– A complete parametric model of the structure

Then to optimize the structure in terms of different parameters, the model has been reduced to a single roof structure. (Figure 50)

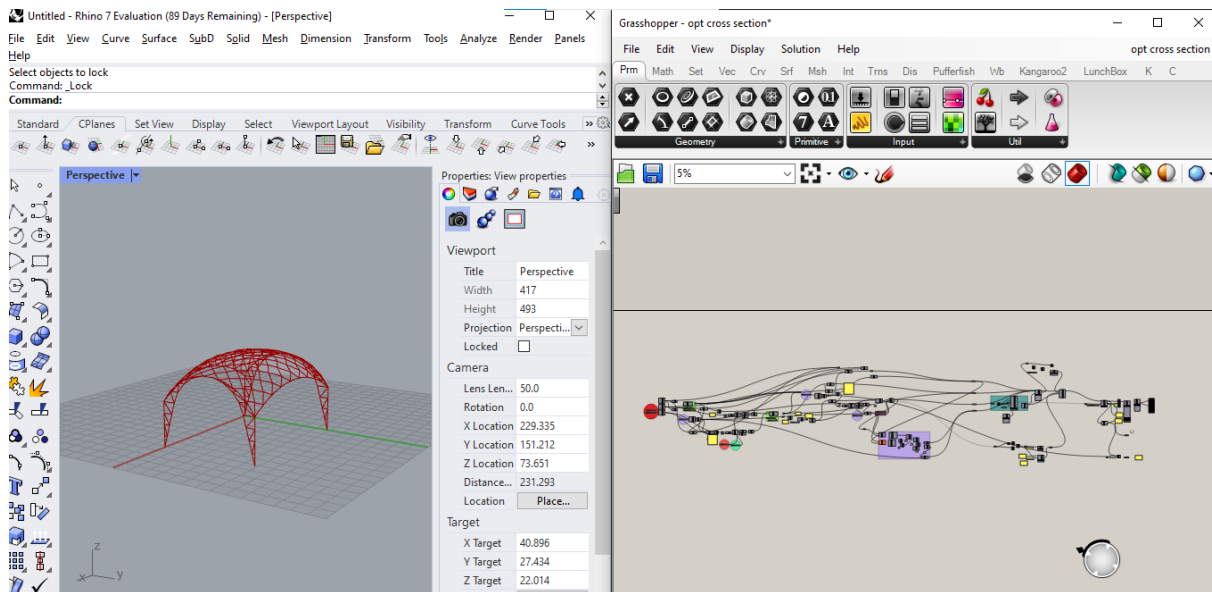


Figure 50– Reduced parametric model of the structure

Thanks to parametric modelling, it is possible to modify and optimize the structure in terms of chosen parameters. Parameters of the structure have been defined in following figures, some of these parameters have been also used in optimization process (Will be discussed in following sections).

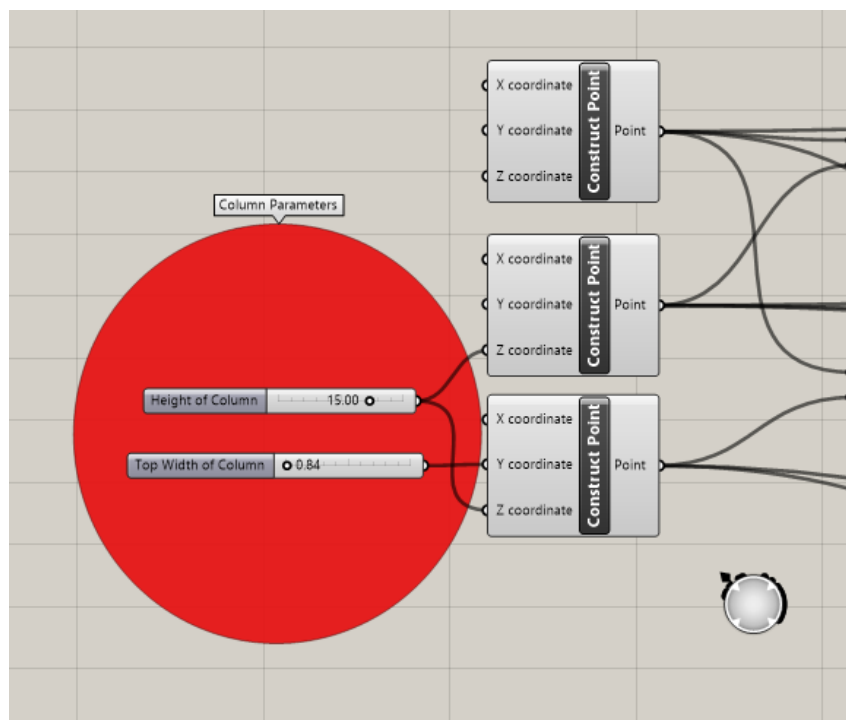


Figure 51– Column Parameters

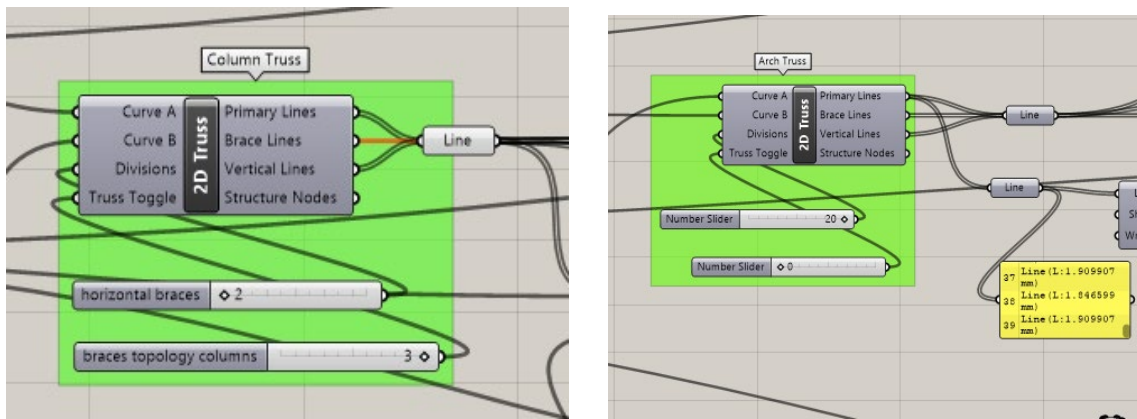


Figure 52– Truss Parameters

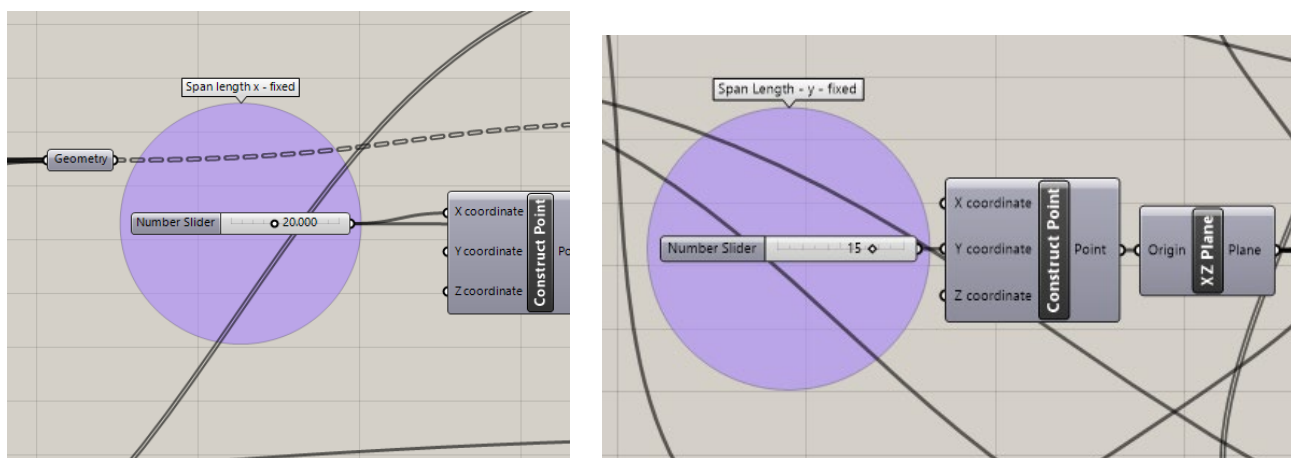


Figure 53– Length Parameters

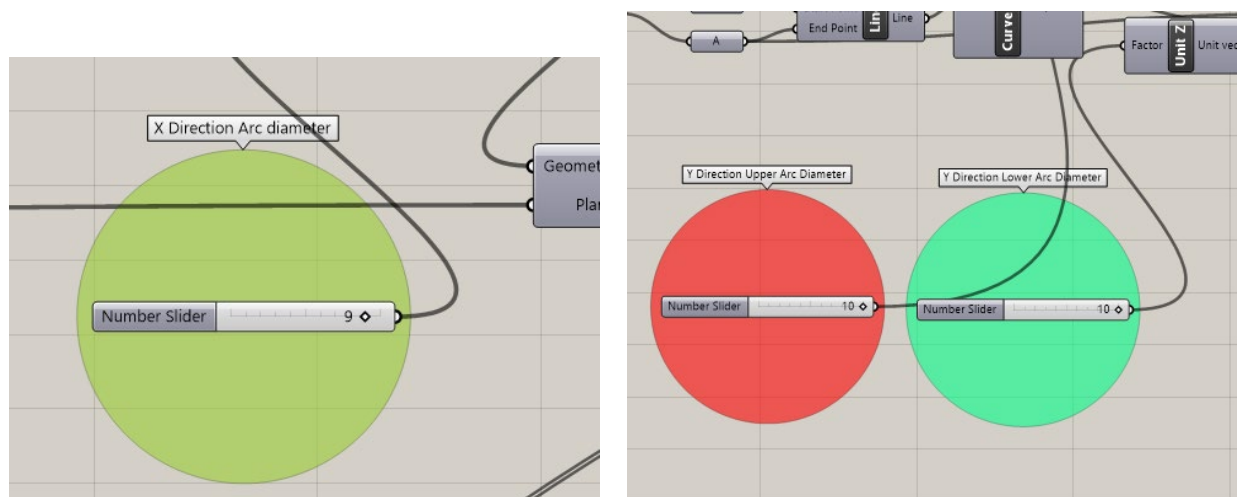


Figure 54– Arc Parameters

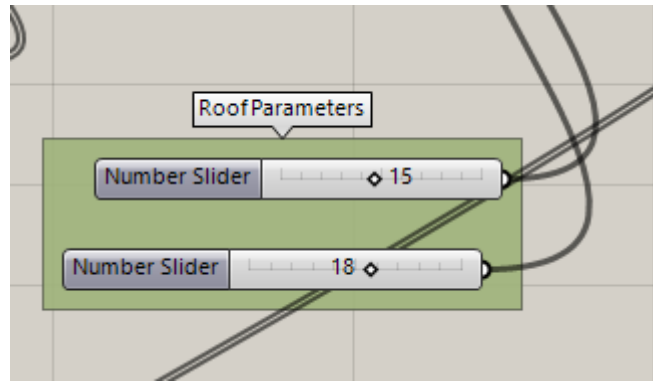


Figure 55 – Diagrid Roof Parameters

4.2 FEM Analysis Using Karamba3D

Karamba3D is a Finite Element program like many others. However it has advantages over these programs in several important respects: It is easy to use for non-experts, has been tailored to the needs of architects and engineers in the early design phase, works interactively and costs slightly less than the rest. Karamba3D is fully embedded in the parametric environment of Grasshopper which is a plug-in for the 3d modeling tool Rhinoceros. This makes it easy to combine parameterized geometric models, finite element calculations and optimization algorithms like Octopus or Galapagos.

To prepare the model for FEM with Karamba3D, all elements of the model should be collected to define as beams, supports, springs, load application positions etc.



Figure 56 – Collection of Elements for FEM Modelling

In order to assign steel profiles to curved parts of structure which designed with “Arc” tool before, should be discretized to “Line” elements. (Figure 57)

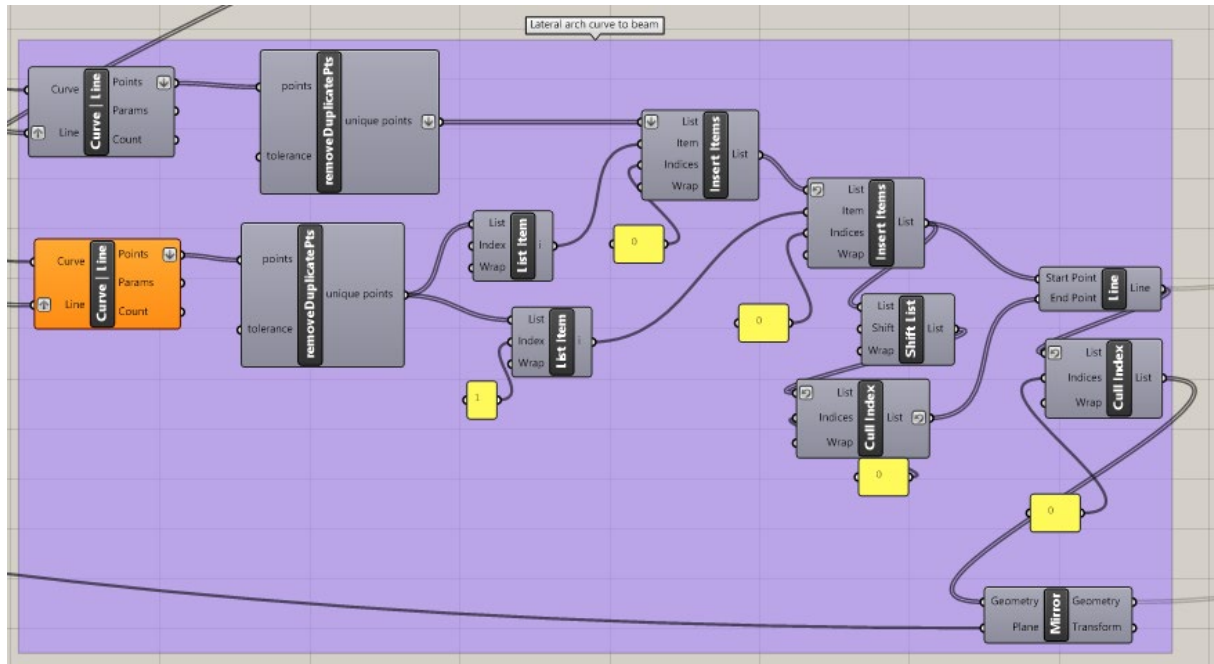


Figure 57– Discretization of Arc Elements

Assemble Model

In order to calculate the behavior of a real world structure one needs to define its geometry, loads and supports. The component “*Assemble*” gathers all the necessary information and creates a statical model from it (see Figure 58).

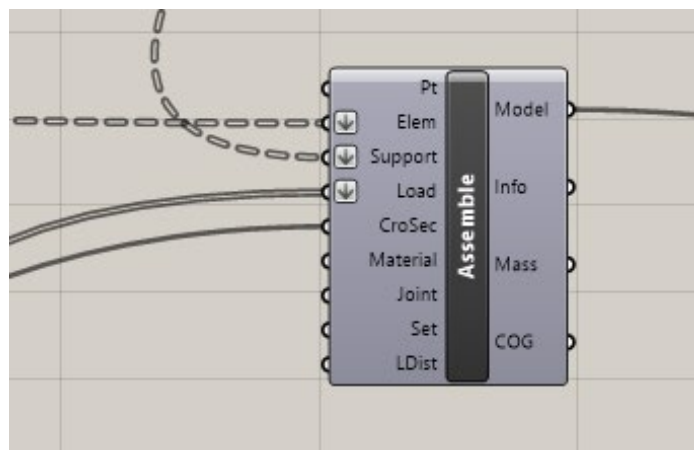


Figure 58– The “*Assemble*”-component gathers data and creates a model from it

Cross sections of elements and materials can be defined either upon creating an element or at the “*Assemble*”-component. The latter option overrides the former and assigns cross sections and

materials via element identifiers. Using regular expressions for selecting identifiers of elements provides a flexible means of attaching cross sections and materials to different parts of a model. The output-plug “Mass” renders the mass of the structure in kilogram, “COG” the position of its center of gravity. When being plugged into a panel the model prints basic information about itself: number of nodes, elements, and so on. At the start of the list the characteristic length of the model is given, which is calculated as the distance between opposing corners of its bounding box.

Line to Beam

Figure 59 shows how the “*LineToBeam*”-component takes two lines as input, finds out how they connect and outputs beams as well as a set of unique points which are their end-points. Lines with end-points of identical index get automatically removed. Unless lines get removed, there is a one to one correspondence between the list of input lines and output beams. The “*LineToBeam*”-component accepts only straight lines as geometric input. Therefore poly-lines and the like need to be exploded into segments first. All coordinates are in meters (or feet in case of Imperial Units). The “Color”-input lets one define a color for the rendering of elements. In order to display the colors activate the “Elements”-button in submenu “Colors” of the “*ModelView*”-component. In addition the option “Cross section” of the submenu “Render Settings” of the “*BeamView*”-component needs to be on.

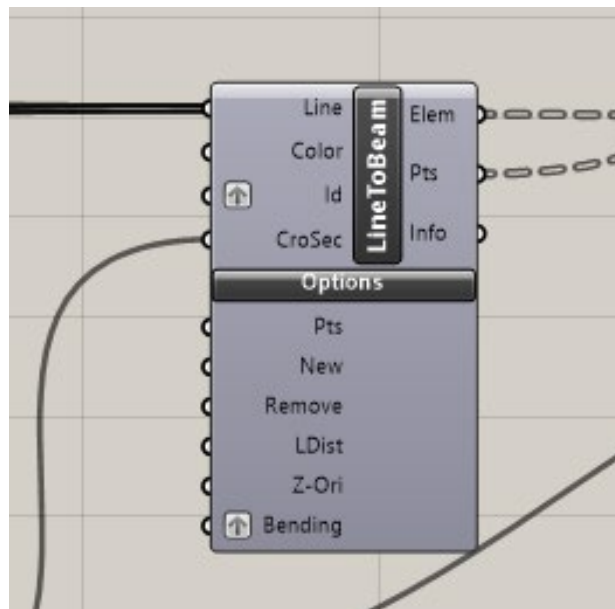


Figure 59 – The “*LineToBeam*”-component that turns two lines into beams

Elements can be given non-unique names via the “Id”-input . It takes a list of strings as identifiers for beams. The default value is an empty string. Each beam has a name by default: its zero based index

in the model. Identifiers provide a useful means to group the beams in order to modify or display them.

Cross Sections

Karamba3D offers cross section definitions for beams, shells and springs. They can be generated with the “*Cross Sections*” multi-component. Use the drop-down list on the bottom to choose the cross section type. The dimensions of each cross section may be defined manually or by reference to a list of cross sections.

In figure 60, it can be seen that the chosen material is S235 steel with I profile family. Within all possible I section profiles (for a given range of height and width, 10 to 200), it will be possible to do our first optimization which is “Size optimization” thanks to “*Optimize Cross Section*” tool of Karamba3D (will be discussed in next section). Moreover, the “Cross Section Selector” has been used to choose cross sections among all possibilities.

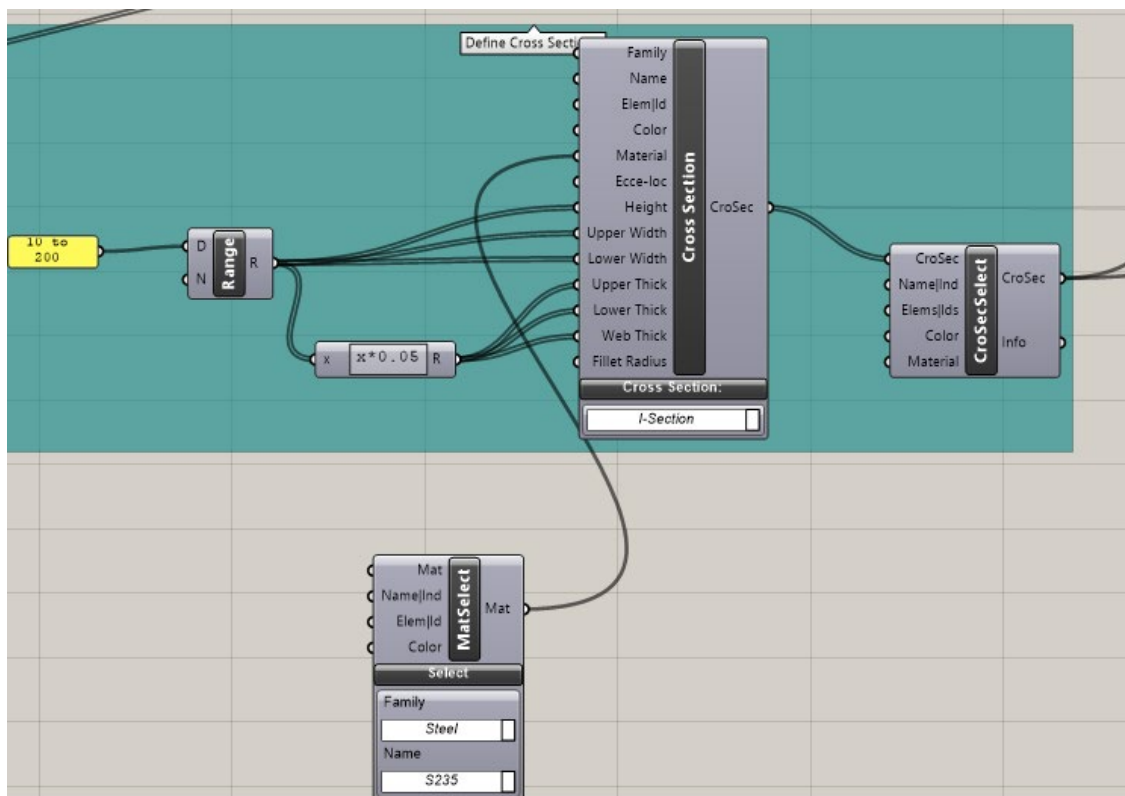


Figure 60 – “Cross Section” and “Cross Section Selector” Components

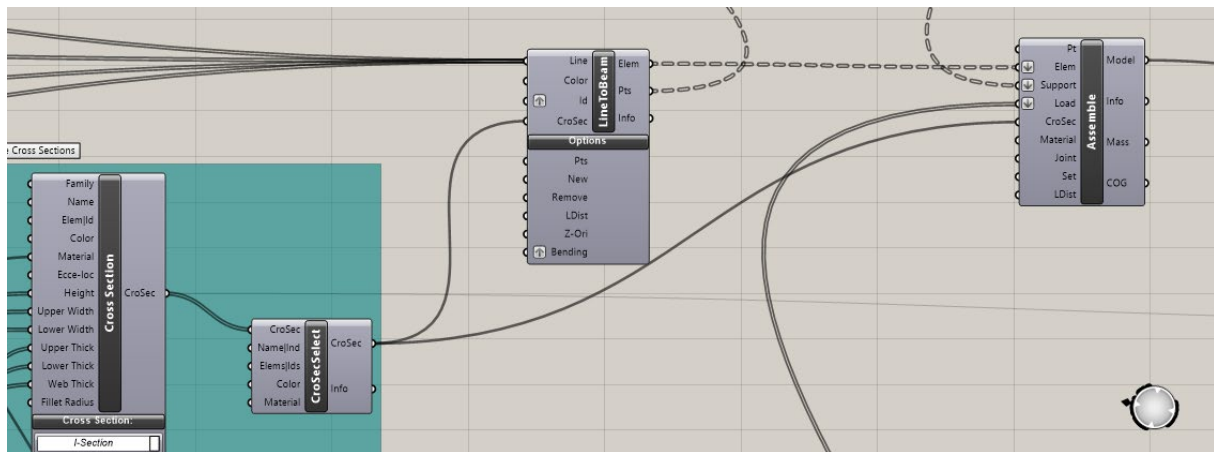


Figure 61 – “Cross Section”, “LineToBeam” and “Assamble” Components

Definition of Loads

Currently Karamba3d supports these types of loads: gravity-, point-, imperfection-, pretension-, temperature-loads, constant mesh-loads, variable mesh-loads and prescribed displacements at supports. An arbitrary number of point-, mesh-, etc.-loads and one gravity-load may be combined to form a load-case of which again an arbitrary number may exist. Figure 62 shows the definition of loads with the help of the “Loads” multi-component.

For the analysis of case study three representative loads have been chosen for the sake of simplicity. Those loads are, Self-Weight and 10 KN of point loads in direction of x and y that represent Wind-Load.

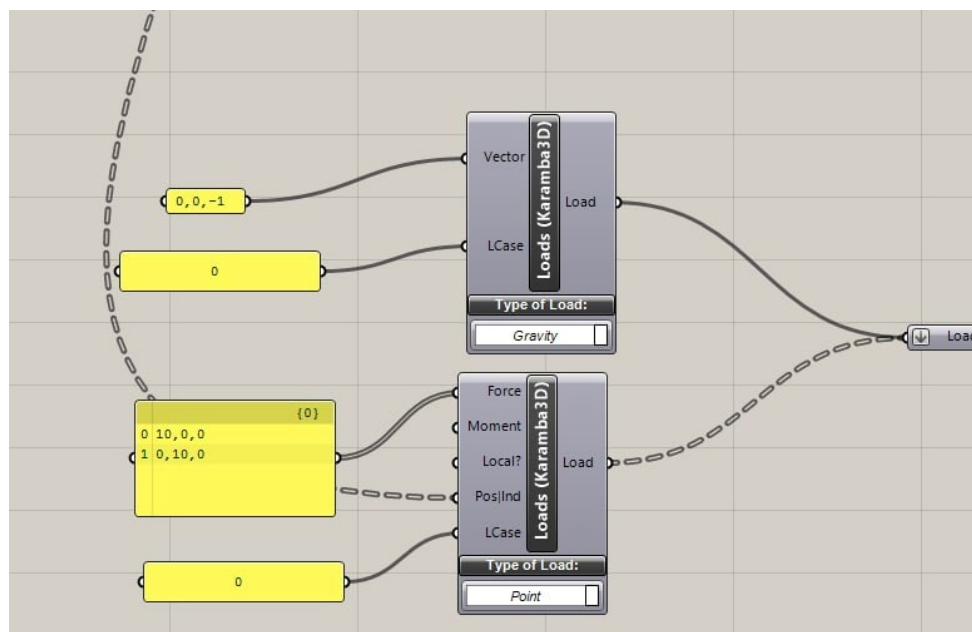


Figure 62 – Algorithm for defining and applying loads to the simulation model

Defining support and joints

Without supports a structure would have the potential to freely move around in space. This is not desirable in case of most buildings. Thus there should always be enough supports so that the structure to be calculated cannot move without deforming i.e. exhibits no rigid body modes.

For our case study, contact points of four columns are taken from the model and defined as a fixed support.

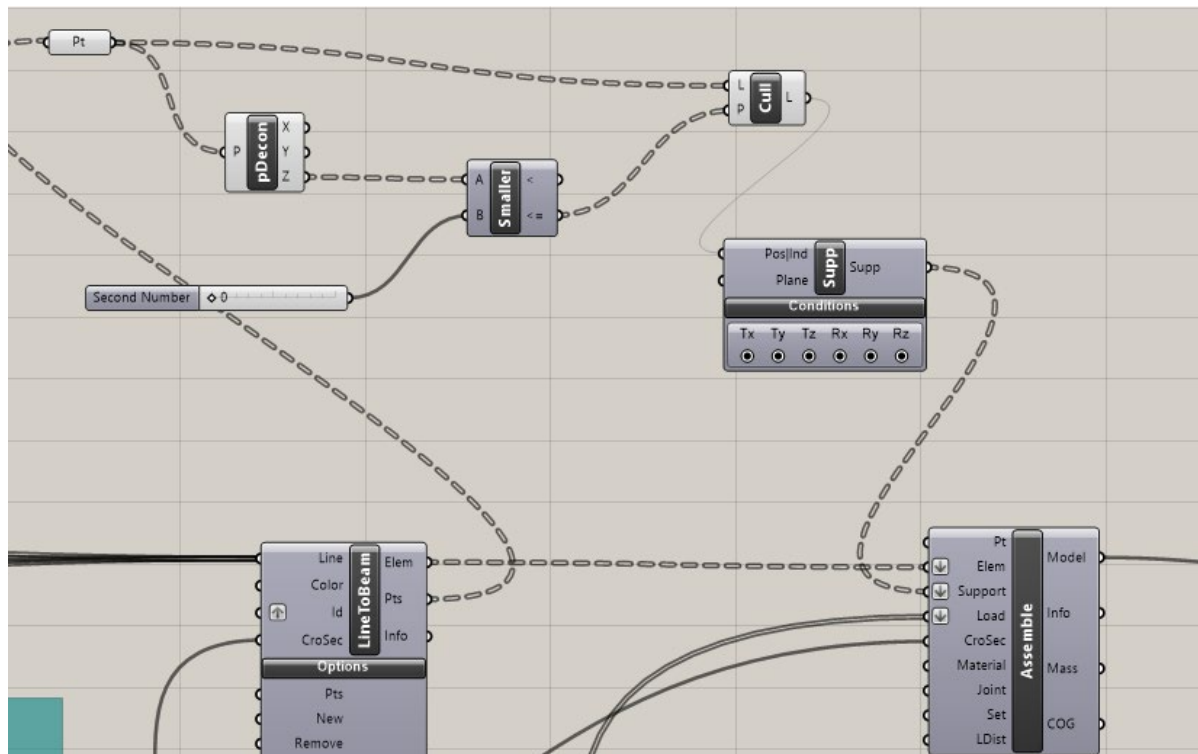


Figure 63 – Support Definitions

Analyze the Model

With geometry, supports and loads defined, the model is ready for processing. The “Analyze”-component computes the deflection for each load case and adds this information to the model. The algorithm behind the “*Analyze*”-component neglects the change of length in axial or in-plane direction which accompanies lateral deformations. This is justified in case of displacements which are small with respect to the dimensions of a beam or shell. For dealing with situations where this condition does not hold, geometric non-linear calculations need to be used.

The analysis component not only computes the model deflections but also outputs the maximum nodal displacement (in centimeter), the maximum total force of gravity (in kilo Newton, if gravity is

set) and the structures internal deformation energy of each load case. These values can be used to rank structures in the course of a structural optimization procedure: the more efficient a structure the smaller the maximum deflection, the amount of material used and the value of the internal elastic energy.

In order to view the deflected model use the “*ModelView*”-component and select the desired load case in the menu “*Result Case*”.

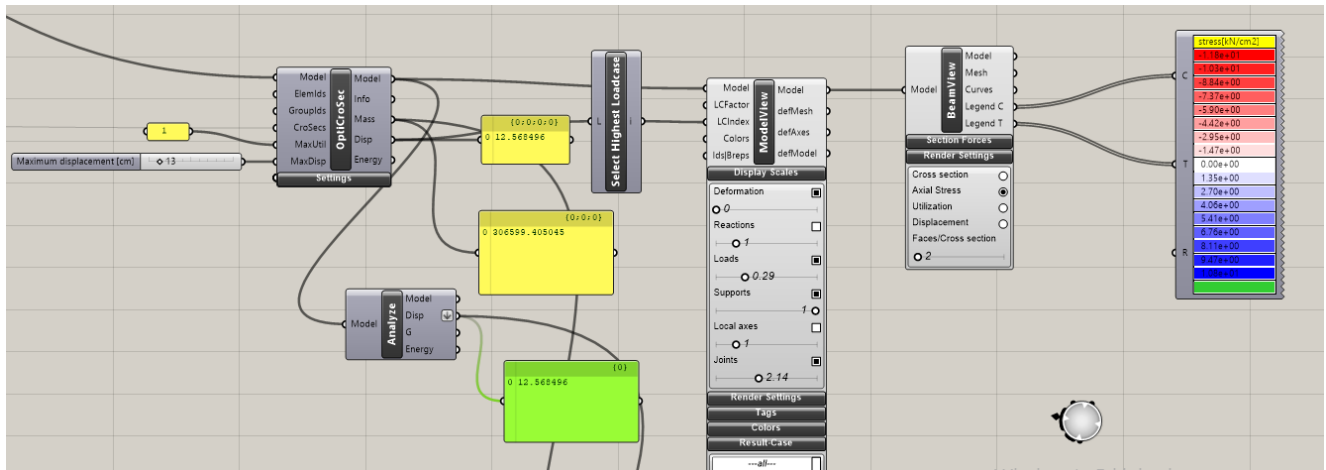


Figure 64 – “*Analyze*”, “*Model View*” and “*Beam View*” Tools

In Figure 65, FE model with all defined conditions can be seen.

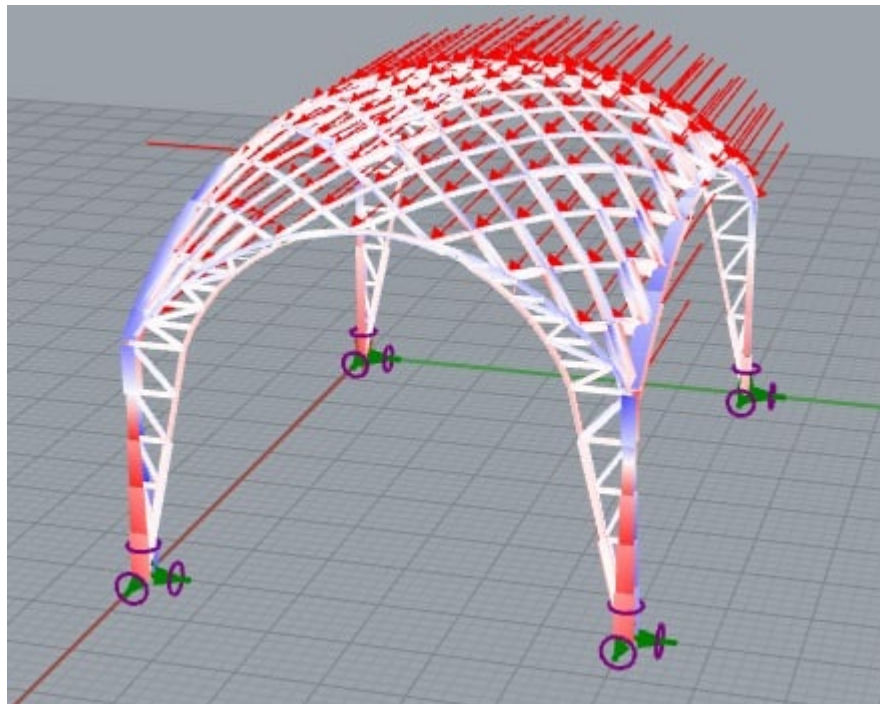


Figure 65 – FE Model of Case Study

After the FE analysis of the case study under chosen loads and support conditions results are as follows:

Initial Mass [kg]: 147198.146047

Initial displacement [cm] 45.315146

4.3 Optimization problem: the numerical model

As already discussed, the aim of the thesis work is to create a parametric model which will be analyzed by FEA and will be optimized in terms of weight with a fixed maximum displacements of each element due to the loads applied. Chosen parameters to be optimized are:

1. Top width of columns
2. Number of bracing elements of column trusses
3. Topology of columns trusses
4. Diameter of arc in X direction
5. Diameter of upper arc in Y direction
6. Diameter of lower arc in Y direction
7. Number of bracing elements of arc trusses
8. Topology of arc trusses

Span Length in directions X and Y are fixed at 20 and 15m in order to simulate and respect to available construction area.

For this case study it was aimed to apply three kinds of structural optimization; size, topology and shape optimization.

As for the “Size Optimization” achieved by using component of Karamba3D called Optimize Cross Section “OptiCrosSec”.

By taking these input parameters, the optimization component chooses the best section and assigns it to all beam elements of the model by varying it in thickness.

For this case study, minimizing the total mass of structure has been chosen as the objective. Moreover, as a restrain condition, maximum displacement have been limited with:

$$\delta_{max} \leq \frac{1}{150} \cdot L_{Tot} = 13 \text{ cm}$$

Where L_{Tot} represents the maximum span length of structure which is 20 m.

4.3.1 SPEA-2 and Hype Reduction Algorithms: Solving the Optimization Problem by Using Evolutionary Genetic Algorithms

"Octopus" function is specialized precisely in multi-objective evolutionary optimization, which allows the search for many objectives at the same time, producing a range of optimized solutions between the extremes of each goal based on the use of two types of algorithms:

- SPEA2 ("Strength Pareto Evolutionary Algorithm 2")
- HypE Reduction ("Hypervolume Reduction Algorithm")

These two algorithms perform the same function and, in general, the choice of preferring the use of one over the other is based only on the number of objectives to be optimized; if these are less than five, the Hype algorithm is used, vice versa the SPEA2.

In both cases, it is a type of genetic algorithms that are randomized search algorithms have been developed in an effort to imitate the mechanics of natural selection and natural genetics. Genetic algorithms operate on string structures, like biological structures, which are evolving in time according to the rule of survival of the fittest by using a randomized yet structured information exchange. Thus, in every generation, a new set of strings is created, using parts of the fittest members of the old set. These two algorithms will be briefly introduced below, describing their basic principle of operation.

Theory of SPEA2 algorithms

SPEA-2, an acknowledged algorithm published in 2001 (Zitzler, 2001), was chosen for implementation within this work, mainly to establish the basic functions of the interface, test its limitations and possibilities in practice and obtain deepened knowledge in the field. SPEA-2 does not represent the latest findings in evolutionary multi-objective problem solving, but still is a good

starting point for further work. Sean Luke's 'Essentials of Metaheuristics' (Luke, 2011) provides comprehensive, yet compact insight into strategies of evolutionary problem-solving. Together with the original documentation (Zitzler, 2001) SPEA-2 can be easily understood and reconstructed.

Figure 68 shows a flow-diagram of SPEA-2's main loop, which will be explained in the following. At its initialization, a random population is generated, and the archive of course is empty, since no elite was formed yet. The evaluation pool is assembled of both of them.

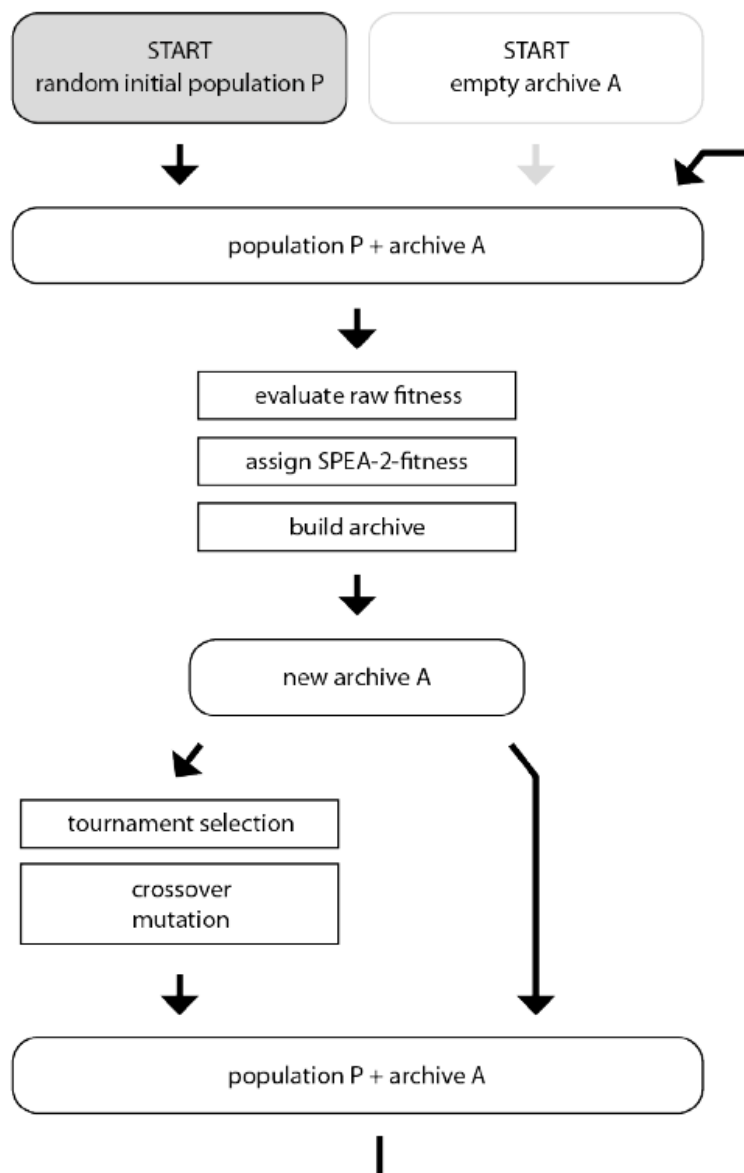


Figure 4.2: Flow diagram of SPEA-2 main loop

Figure 68 – Flow diagram of SPEA-2 main loop

- *Evaluate raw fitness*

The objective values of each solution within the pool are calculated, which are called 'raw fitness'. In Octopus, either an internally defined academic test problem, or an explicit problem formulation in the form of a Grasshopper-definition is evaluated. For practical problems, this is the most time-consuming part of the loop.

- *Assign SPEA-2-fitness*

Out of the multiple objective values calculated for each solution, a single fitness value has to be calculated for the genetic algorithm. This is where the Pareto-principle, resp. a sophistication, comes to action. In the multi-dimensional objective space, where every solution is represented as a (possibly four- or five-dimensional) point, the fitness value is summed up from two parts:

1. The solution's Pareto-strength, in principle a lexical comparison of multiple objectives, and
2. its sparsity, which is a measure forcing the even exploitation of the solution space.

- *Build archive or 'Environmental Selection'*

SPEA-2 maintains a constant archive size and makes sure that boundary solutions are not thrown away. This is a form of elitism, meaning that the solutions considered best are always copied from one generation to the next. For this purpose, as a first step, all Pareto non-dominated individuals are copied from the evaluation pool into the archive of the next generation. The source code of splitting a list of solutions into non-dominated and dominated according to the Pareto-principle. It then depends:

- Desired archive size exactly fits the number of non-dominated individuals, environmental selection is complete.
- If the archive is overfull, some individuals out of it have to be truncated. An iterative truncation-procedure is applied, which is exemplified in a two-dimensional solution space in Figure 69. This technique ensures diversity and uniform spread of individuals in the multi-dimensional solutions space: One of the two solutions lying closest to each other in solution space is removed. The ties between the two are broken by removing the one with the closer second-nearest neighbor, which, amongst others, ensures to maintain solutions at the outer boundaries of the Pareto-front approximated so far.

– If the archive is underfull, the dominated individuals with best fitness also are copied into the new archive (from the old evaluation pool).

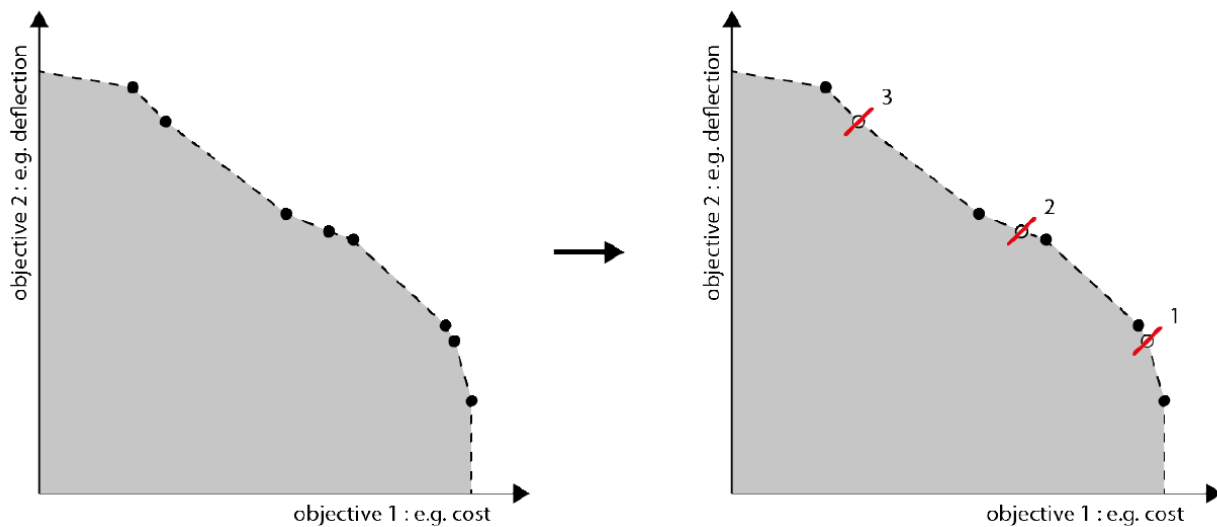


Figure 69 – Archive truncation in SPEA-2

- *Breeding a new population*

Here, a new set of individuals is produced to be evaluated in the next generation. For this purpose, two basic steps are performed in a GA, for which a number of different strategies exist for each of them. Here, only the ones used in SPEA-2 are presented:

1. Selection of individuals or 'Mating Selection': This is done by Tournament selection using Pareto principles. The size of the tournament can be varied, but two rounds proved to be practicable for most problems. The basic principle is the random selection of an initial candidate, and another random individual in each round of the tournament. Only the Pareto-dominating solution is being kept for the next round, so a basic average quality niveau of the mating candidates is ensured. By increasing the tournament size, this quality can be increased, but can also lead to premature convergence.

2. Variation: The individuals selected are varied by a two-stage process:

- Recombination or 'Crossover': The selected individuals mate by exchanging parts of their genomes. First of all, selected individuals get chosen for recombination with a certain probability, in case of the original SPEA-2 with 90%. 'Simulated Binary Crossover' (Deb, 1994) simulates important properties of binary crossover procedures that were used in the beginning of evolutionary computing, but actually come close to nature's evolutionary principles. There, the exchange of gene-information

happens on the basis of single bits and groups of them. The first important one of those properties is that the average of the two parents' decoded parameters is the same before and after the operation. The second one describes the probability occurrence of a spread factor $\beta \approx 1$ is more likely than any other value of β . Here, $\beta < 1$ and $\beta > 1$ mean 'contracting crossover' and 'expanding crossover', a measure for the exploratory character of a crossover operation.

- Mutation: The mutation rate set by the user firstly determines to which probability a single individual gets mutated, and, if, secondly to which extent. For a continuous variable, the current value is changed to a neighboring value using a polynomial probability distribution. This 'Polynomial mutation' procedure was introduced by (Deb, 1996).

Theory of HypE Algorithm

HypE is one of the most common hyper volume estimation algorithms (Zitzler and Thiele, 1998a, 1999), it is a specific indicator based on the concept of dominance used as a measure of the quality of a set of multi-objective optimization solutions; it is in fact the only quality indicator known to be completely sensitive to the dominance of Pareto.

A multi-objective optimization problem can typically be formulated as follows:

$$\min f(x) = [f1(x), f2(x), f3(x), \dots fd(x)]$$

Where $x = [x1, x2, x3, \dots xm]$ is the vector of decision variables, $f(x)$ is the vector of objective functions (relative to x), and d is the number of objective functions. The ideal solution to the optimization problem would be the one that has the lowest value of all objective functions but, since in most cases this solution does not exist, the definition of excellent must be extended by introducing the concept of dominated solution.

As already said, a solution is not to dominated if no solution dominates it and the set of non-dominated solutions represents the set of optimal according to Pareto while the corresponding set of points in space (which correspond to compromise solutions, among which the designer can choose) of the objective functions defines the Pareto front.

The Pareto Front - Optimal in multi-objective optimization (MOOP) problems therefore represents a set of solutions that are not dominated by each other, being the best considering the rest of the possible solutions. It means that a single solution cannot be superior to all other solutions compared to all objectives.

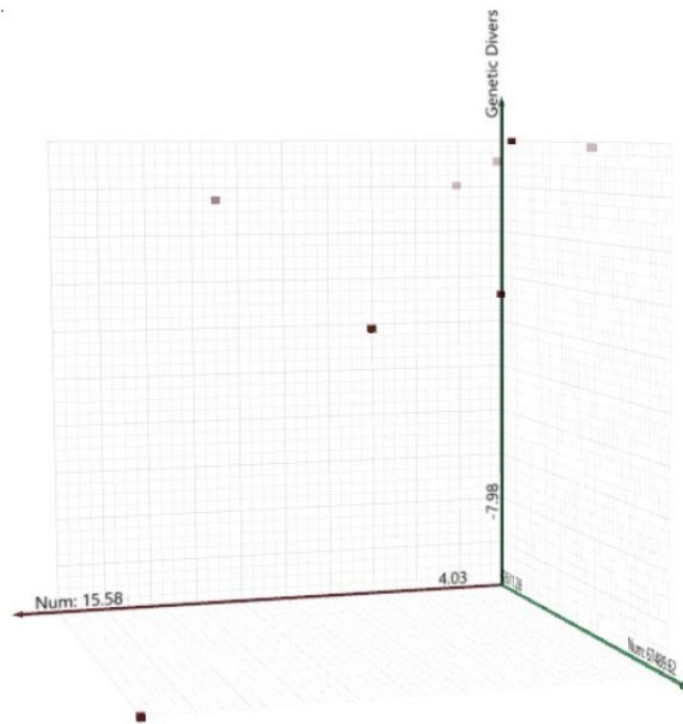


Figure 70 – Pareto Front for optimal solutions

For the application within chosen case study. Maximum number of generations have been chosen as 200 where each generation contains 50 population.

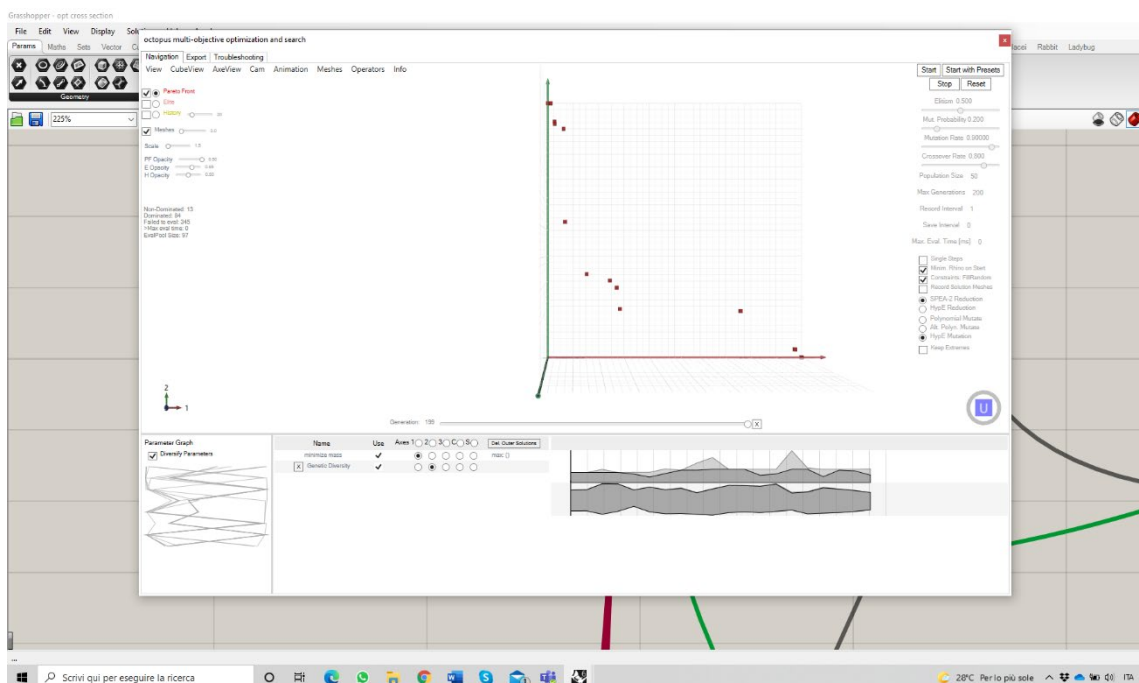


Figure 71– Pareto Front of case study optimization

4.4 Results

Once all the initial parameters have been set, multi-objective optimization has been performed using the HypE Reduction algorithm in which, as already mentioned, the function to be minimized is:

- $f(x)$: Mass (m)

By imposing:

- Constraint of max displacement of 13cm.
- Maximum number of generations $N_{\max, \text{gen}} = 200$
- Maximum number of populations $N_{\text{pop}} = 50$

At the end of the optimization process, carried out with the Octopus solver, the algorithm provides a set of solutions included in the Optimal Pareto Front representing the best solutions (which are not dominated by anyone else) among all possible ones in the optimization problem.

5 different solutions of optimization with Octopus function will be introduced as: Upper Solution, Middle Solution, Lower Solution, Mid-Up Solution, Mid-Low Solution from Pareto front. Those solutions are optimized solutions in terms of topology, shape and size.

Upper Solution

Position of solution in the Pareto Front can be seen in Figure 72.

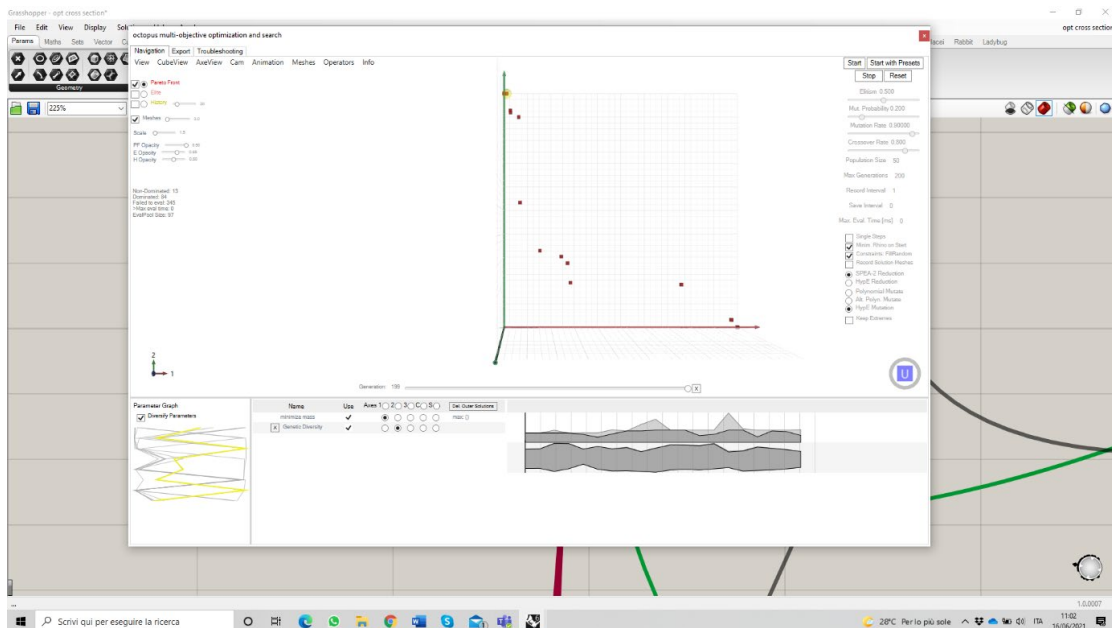


Figure 72– Upper solution in Pareto Front

Optimized geometry of case study can be seen in Figure 73.

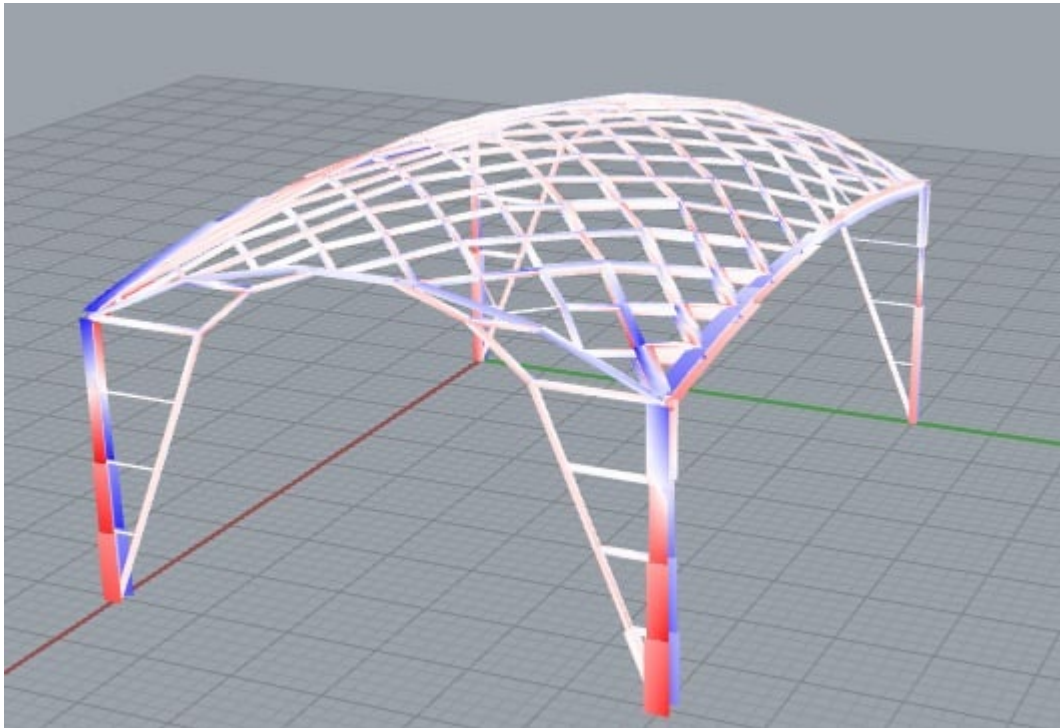


Figure 73– Geometry of Upper solution

Results in terms of mass and displacement are as follows:

Mass [kg]: 194394.54

Displacement [cm] 12.86

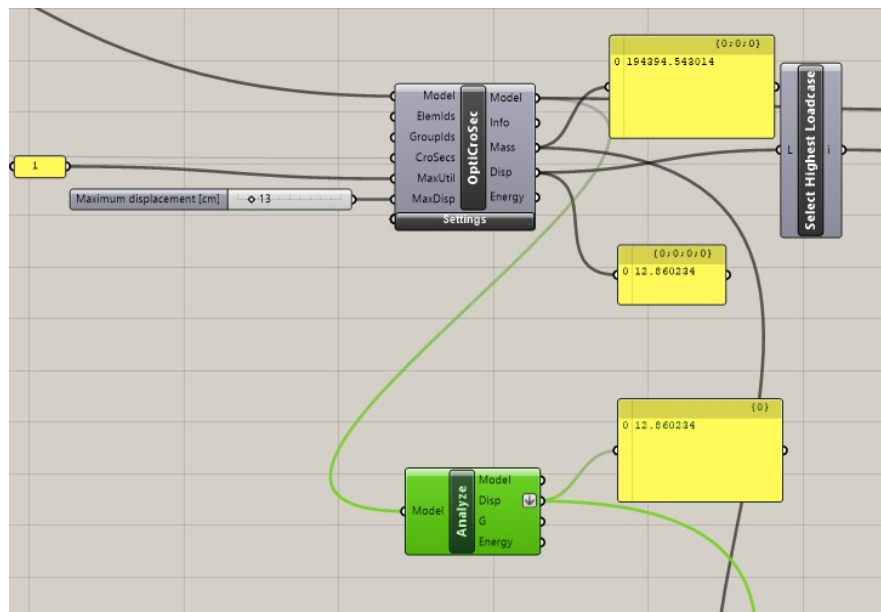


Figure 74– Results of Upper solution

Middle Solution

Position of middle solution in the Pareto Front can be seen in Figure 75.

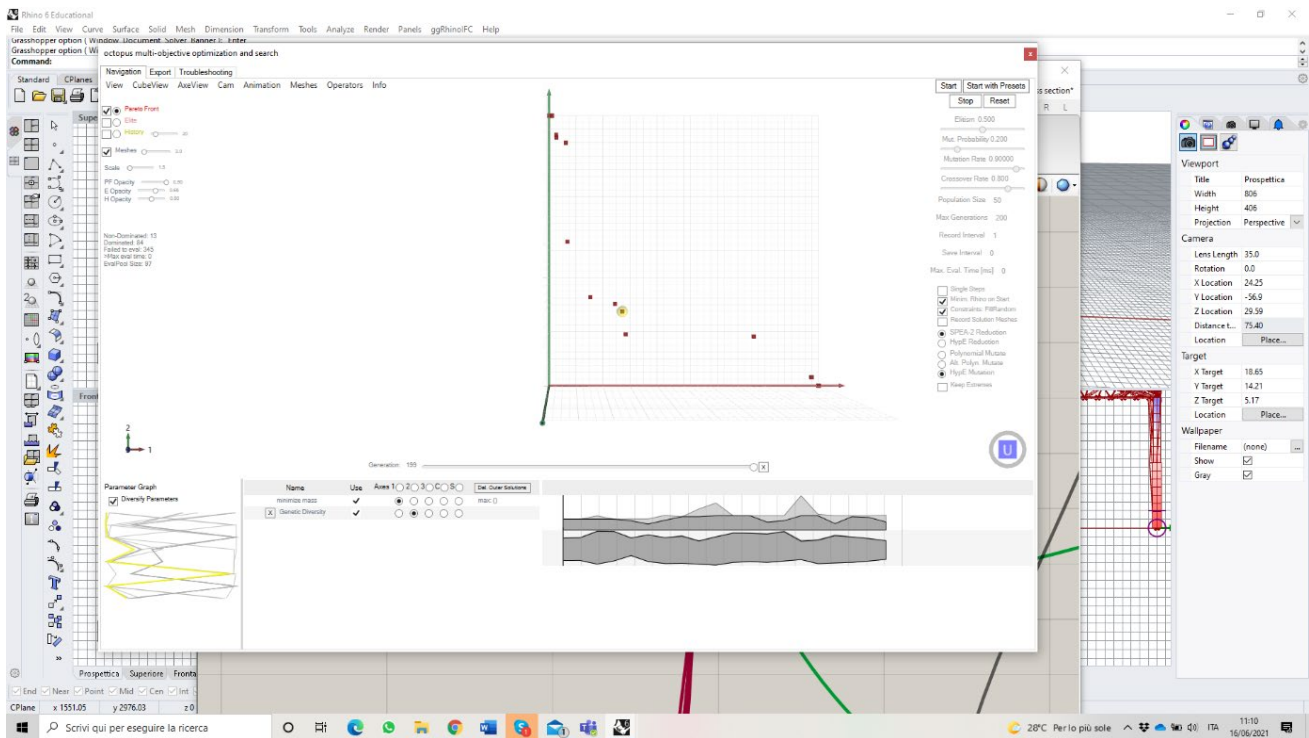


Figure 75– Middle solution in Pareto Front

Optimized geometry of case study can be seen in Figure 76.

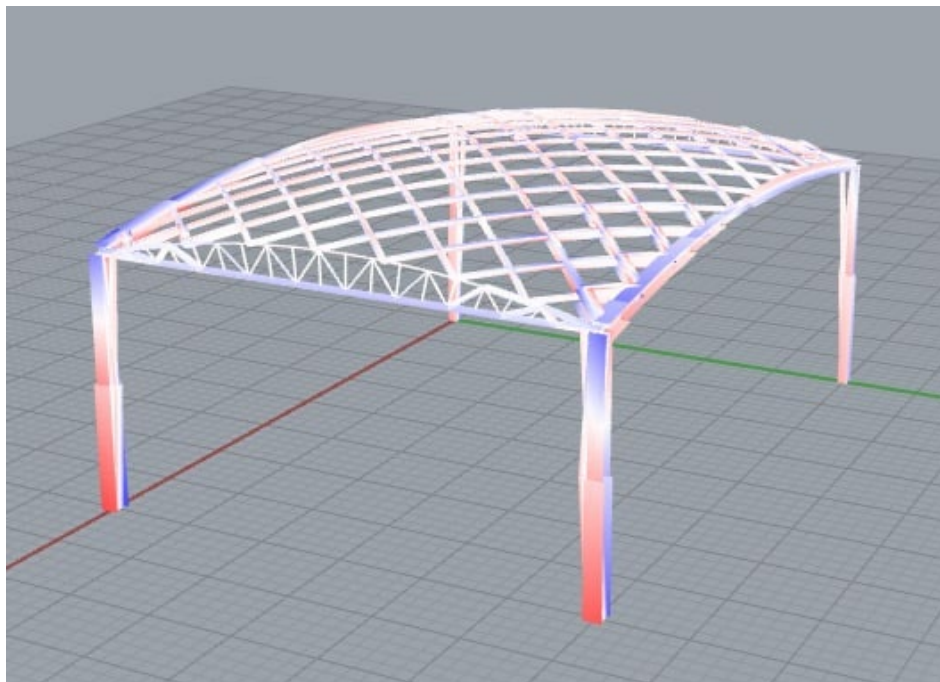


Figure 76– Geometry of Middle Solution

Results in terms of mass and displacement are as follows:

Mass [kg]: 226208.7

Displacement [cm] 12.91

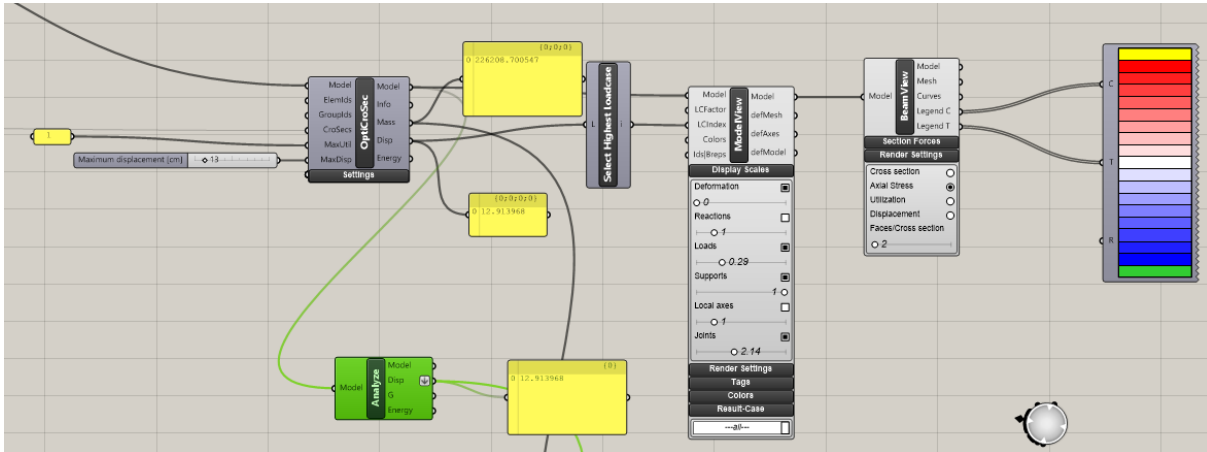


Figure 77– Results of Upper solution

Lower Solution

Position of lower solution in the Pareto Front can be seen in Figure 78.

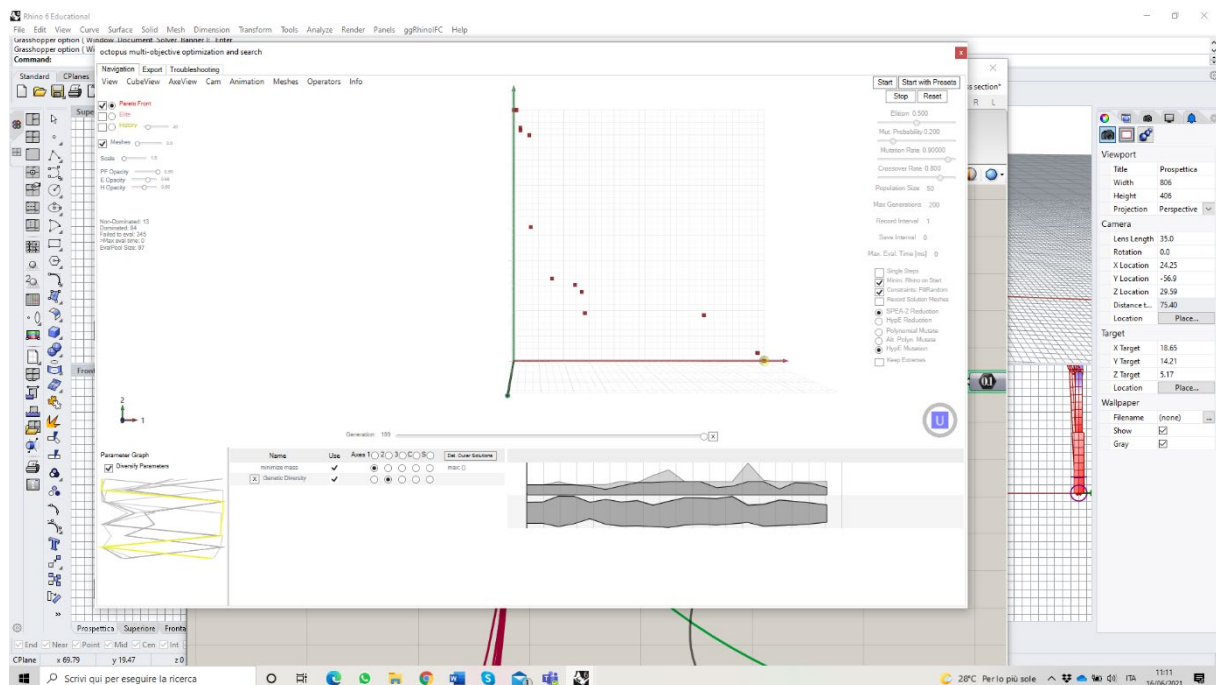


Figure 78 – Lower solution in Pareto Front

Optimized geometry of case study can be seen in Figure 79.

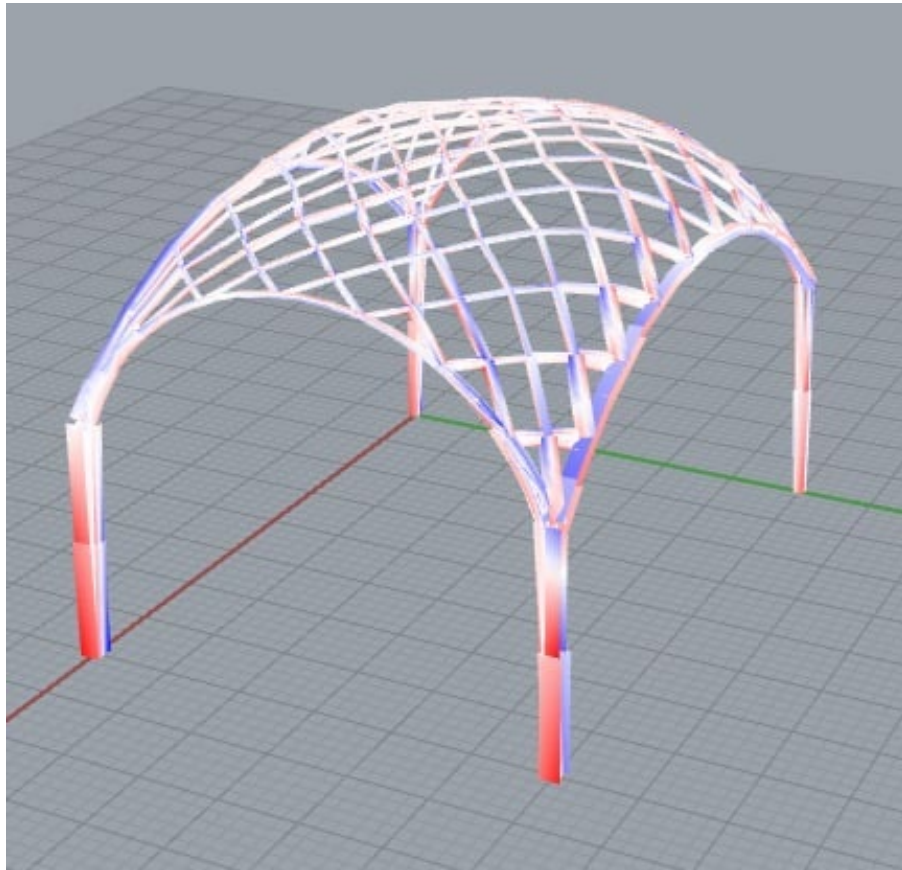


Figure 79– Geometry of Lower Solution

Results in terms of mass and displacement are as follows:

Mass [kg]: 306599.41

Displacement [cm] 12.57

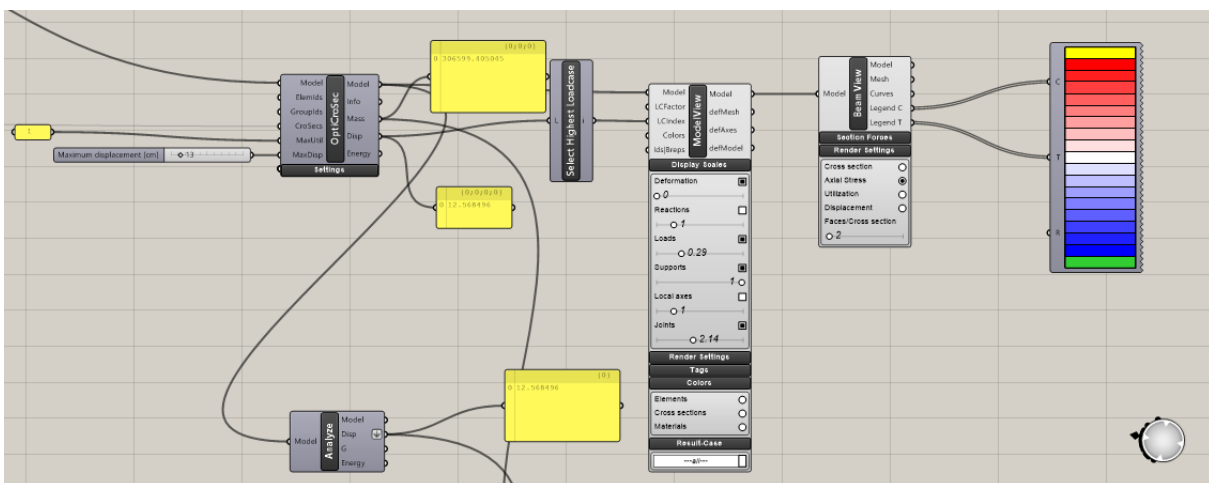


Figure 80– Results of Lower solution

Mid-Up Solution

Position of mid-up solution in the Pareto Front can be seen in Figure 81.

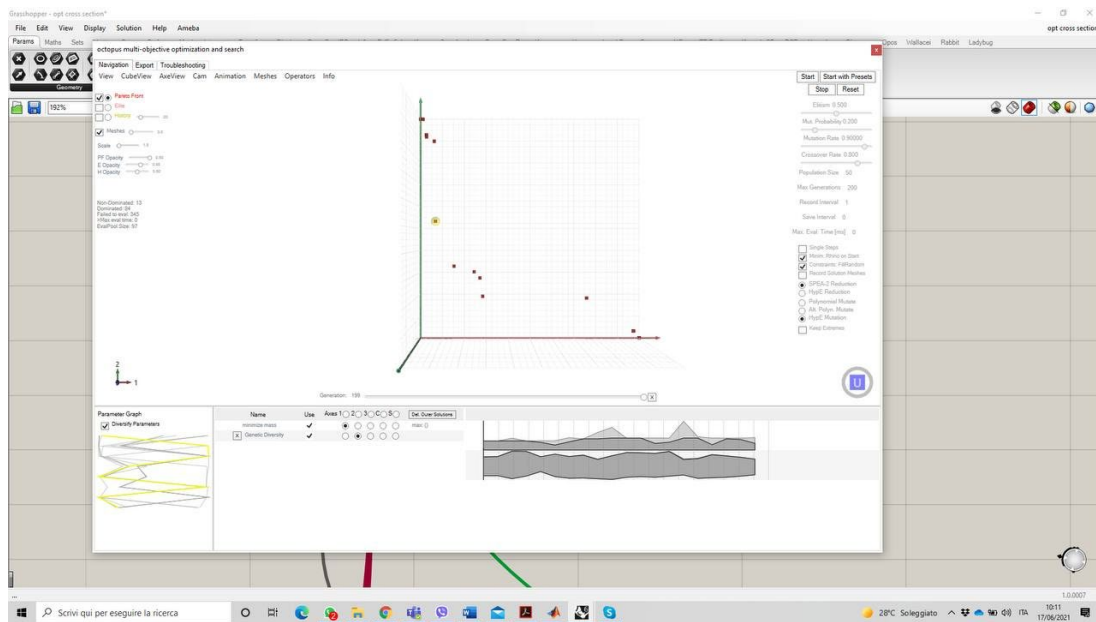


Figure 81– Mid-up solution in Pareto Front

Optimized geometry of case study can be seen in Figure 82.

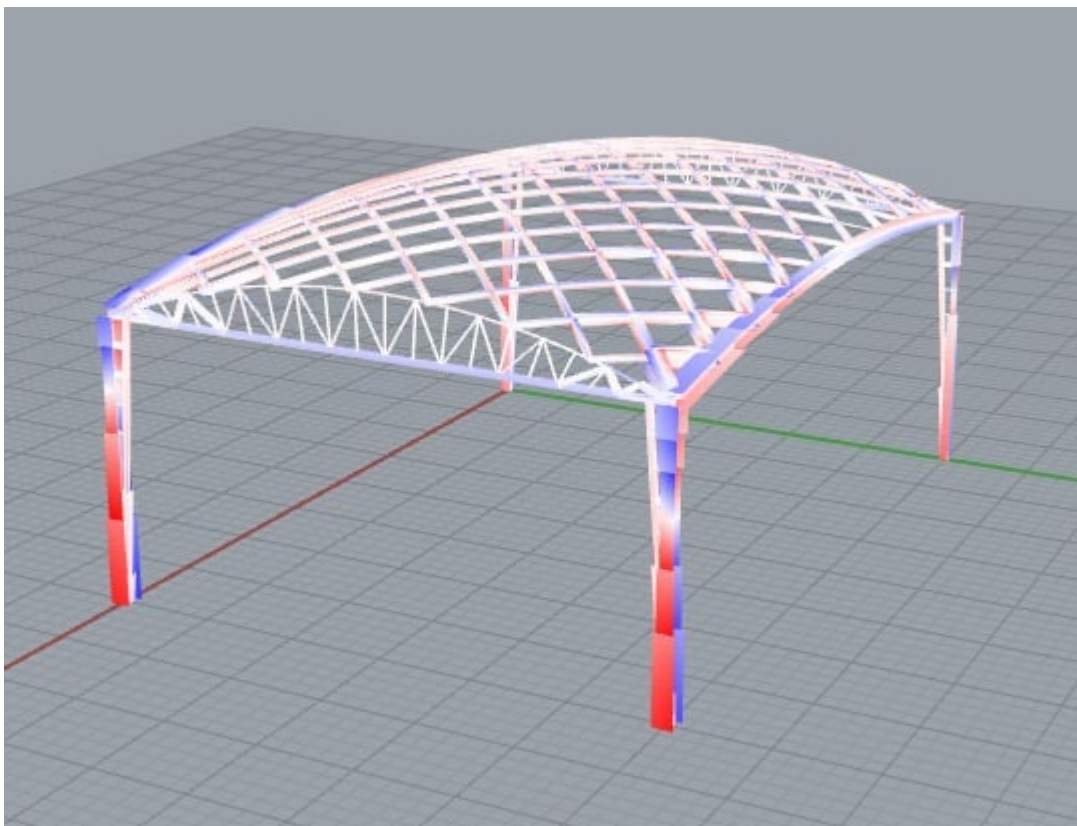


Figure 82– Geometry of Mid-up Solution

Results in terms of mass and displacement are as follows:

Mass [kg]: 201431.2

Displacement [cm]: 12.98

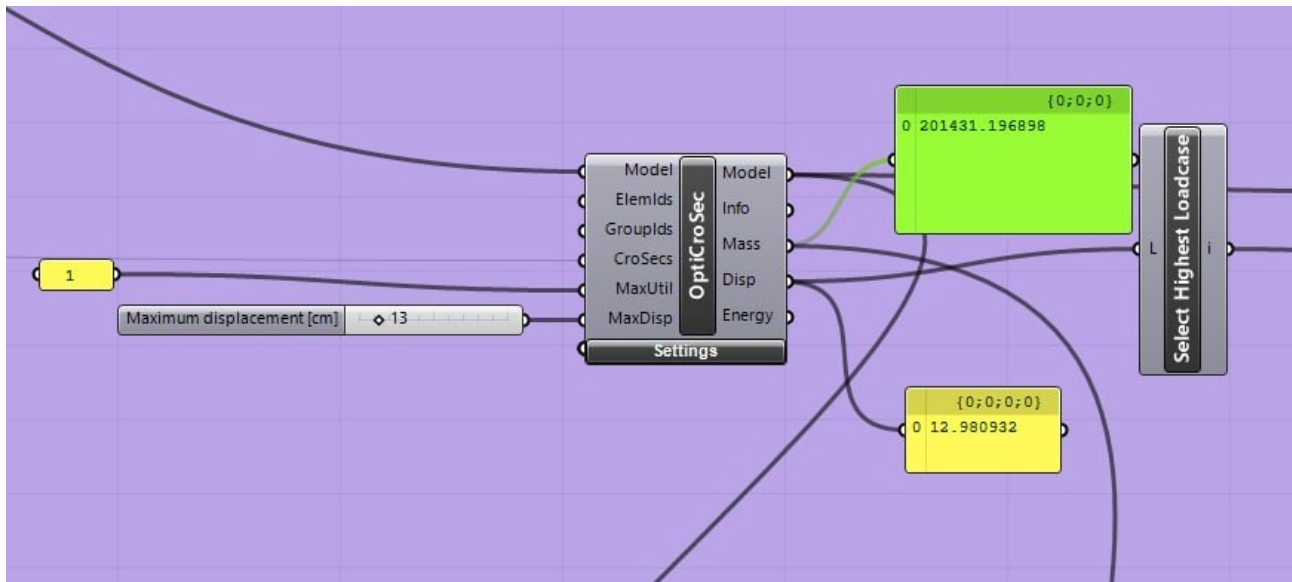


Figure 83– Results of Mid-up solution

Mid-Low Solution

Position of mid-low solution in the Pareto Front can be seen in Figure 84.

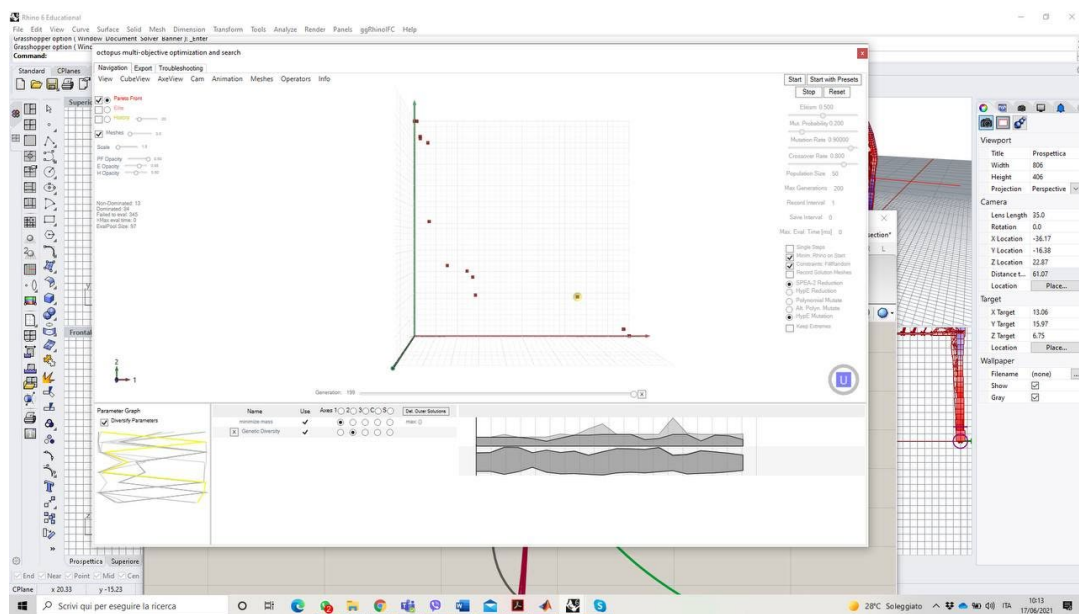


Figure 84– Mid-low solution in Pareto Front

Optimized geometry of case study can be seen in Figure 85.

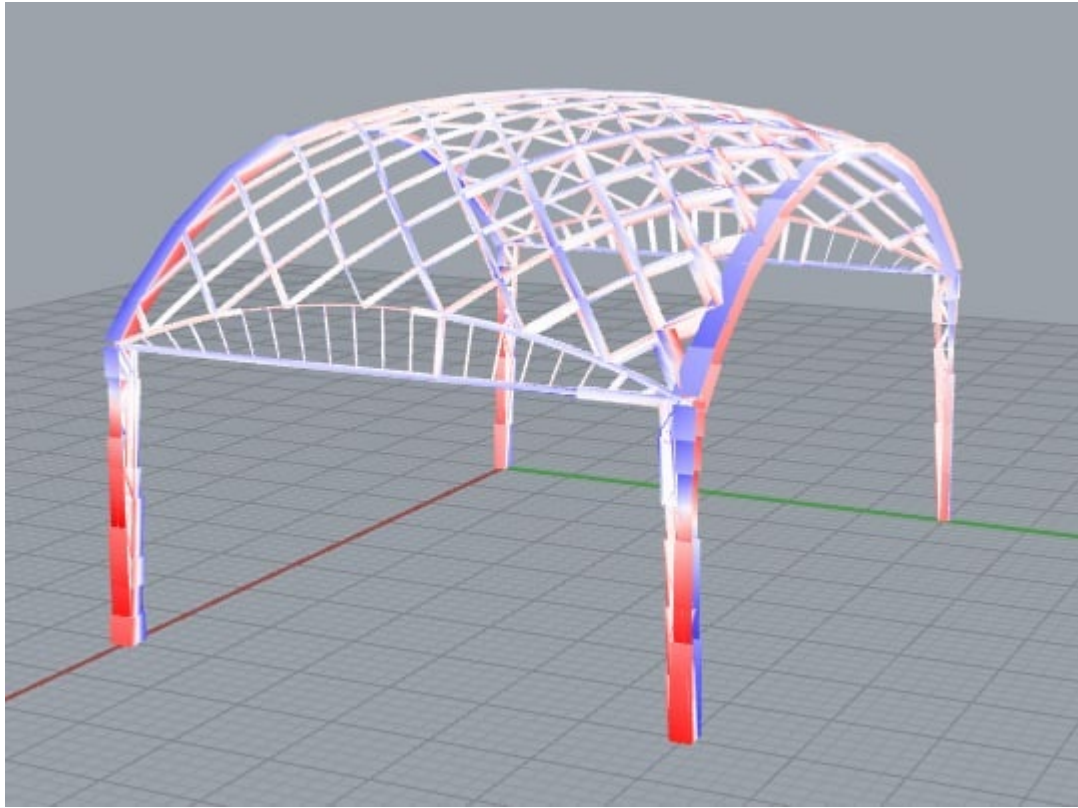


Figure 85– Geometry of Mid-low

Solution Results in terms of mass and displacement are as follows:

Mass [kg]: 296220.98

Displacement [cm] 12.996247

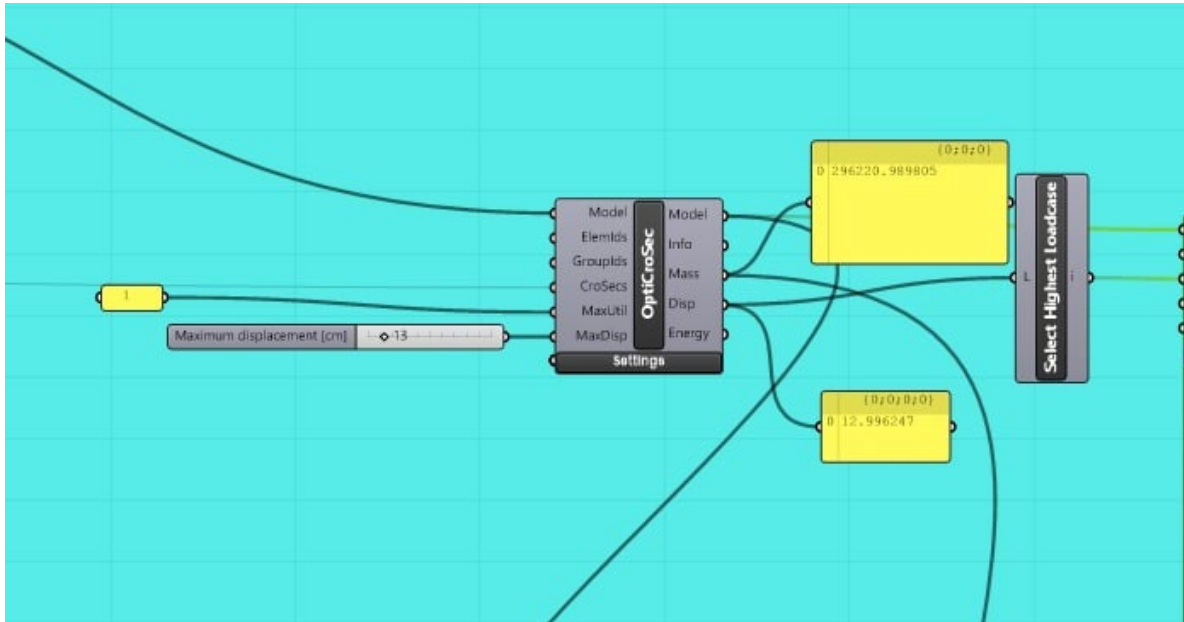


Figure 86– Results of Mid-low solution

In Table 2, it is showed that by using genetic algorithms we may optimize our structures in terms of different size, shape and topology. The value that wanted to be optimized was the mass of the structure. However, it can be seen that all values regarding the mass of optimized solutions are bigger that the mass of original shape. The reason behind that is, we have tried to optimize mass by imposing a constraint of maximum displacement of 13cm and, all five optimized solutions have respected that constraint. If we look from this viewpoint, the increase on the mass of structure is admissible since the decrease of the maximum displacement is very significant.

This results are showing us that using generative design and AI, can be very effective in the stage of conceptual design. Big range of possible solutions can be obtained by introducing values that aimed to be optimized, constraint that has to be respected, and parameters that can be modified.

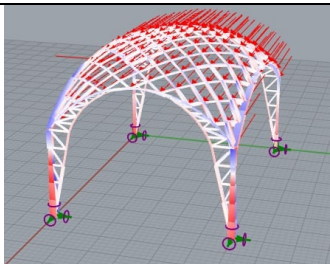
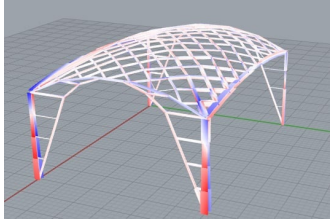
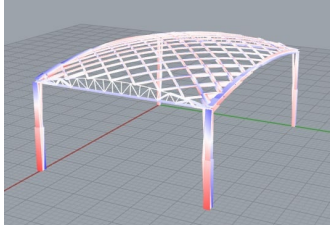
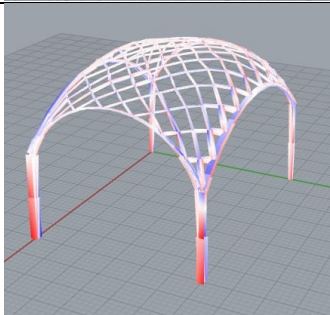
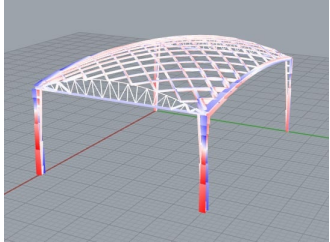
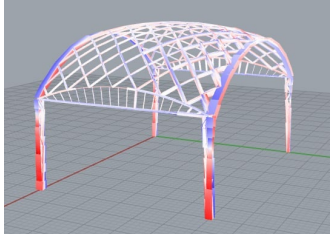
	Geometry	Mass(kg)	Displacement(cm)
Initial Geometry		147198.15	45.32
Upper Solution		194394.54	12.86
Middle Solution		226208.70	12.91
Lower Solution		306599.41	12.57
Mid-Up Solution		201431.20	12.98
Mid-Low Solution		296220.98	12.99

Table 2– Summary of optimization results

5. CONCLUSION

Structural optimization, or the use of numerical optimization techniques to design material-efficient or cost-effective structures, has great potential for the construction industry. The construction industry is responsible for a large share of the worldwide consumption of natural resources, and structural optimization can help to reduce this, so improving the sustainability of the sector. In addition, structural optimization has the potential to reduce not only the construction cost, but also the engineering cost, by automating the repetitive task of sizing structural members. Finally, structural optimization can lead to innovative design solutions for specific structural components or materials.

Usually, a distinction is made between three structural optimization strategies: (1) size optimization, where the aim is to find the optimal dimensions of the structural components, (2) shape optimization, where the shape of the structure is parameterized and these parameters are optimized, and (3) topology optimization, where the optimal spatial distribution of structural material or structural components is determined.

Real-world structural engineering problems are usually characterized by a number of uncertainties. The actual loads, material properties, geometry, etc., may deviate from the values assumed in the design phase. Such uncertainties can have a detrimental impact on performance. As a consequence, in structural optimization, it is important not only to optimize the performance of the blueprint design, but also to ensure that its sensitivity with respect to uncertainties remains limited. Robust and reliability-based design optimization techniques allow for the incorporation of uncertainties in the optimization process.

This Thesis Topic encompasses (but is not limited to) size, shape, and topology optimization in the context of structural engineering, applied to the design of a large span steel diagrid roof structure. Generative design and artificial intelligence (AI) tools have been used by combining computational and parametric design in order to obtain optimized solutions.

The geometry has been created parametrically by using visual programming software Grasshopper. The advantage of this software is it makes possible to implement FE analysis and some genetic algorithm tools that to be used in optimization process without having complex programming techniques but with some plug-ins that called Karamba3D and Octopus.

After FE analysis, size, typology and shape optimization have been achieved and total mass of the structure has been minimized by respecting the maximum displacement of 13 cm.

6. BIBLIOGRAPHY

- Danhaive, R. A., & Mueller, C. T. (2015, August). Combining parametric modeling and interactive optimization for high-performance and creative structural design. In *Proceedings of IASS Annual Symposia* (Vol. 2015, No. 20, pp. 1-11). International Association for Shell and Spatial Structures (IASS).
- Ghabraie, K. (2012). Applications of topology optimization techniques in seismic design of structure. In *Structural Seismic Design Optimization and Earthquake Engineering: Formulations and Applications* (pp. 232-268). IGI Global.
- Khabazi, Z. (2011). Generative Algorithms. *Accessed on*, 3.
- Hensel, M., Sunguroglu, D., & Menges, A. (2008). Material Performance. *Architectural Design*, 78(2), 34-41.
- Mirjalili, S. (2019). Evolutionary algorithms and neural networks. In *Studies in Computational Intelligence* (Vol. 780). Springer.
- Goldenberg, M., & Coburn, N. (2019). Topology and Form Finding via Genetic Algorithms.
- Preisinger, C., & Heimrath, M. (2014). Karamba—A toolkit for parametric structural design. *Structural Engineering International*, 24(2), 217-221.
- Huang, X., & Xie, M. (2010). *Evolutionary topology optimization of continuum structures: methods and applications*. John Wiley & Sons.
- Rozvany, G. I. (2012). *Structural design via optimality criteria: the Prager approach to structural optimization* (Vol. 8). Springer Science & Business Media.
- Pidaparti, R. M. (2017). Engineering finite element analysis. *Synthesis Lectures on Mechanical Engineering*, 1(1), 1-267.
- Cazacu, R., & Grama, L. (2014). Steel truss optimization using genetic algorithms and FEA. *Procedia Technology*, 12, 339-346.
- Bendsøe, M. P., Guedes, J., Neves, M. M., Rodrigues, H., Sigmund, O., Carbone, L., & De Arcangelis, R. (2003). Aspects of the design of microstructures by computational means. In *The First HMS2000 International School and Conference on Homogenization* (pp. 99-112). GAKUTO Int. Series in Math. Sci, Appl..

East, A. N., Post, E., Turing, A., Rosser, J. B., & Kleene, S. C. History: Development of the notion of" algorithm.

Zitzler, E., Laumanns, M., & Thiele, L. (2001). SPEA2: Improving the strength Pareto evolutionary algorithm. TIK-report, 103.

Luke, A. (2011). Generalizing across borders: Policy and the limits of educational science. Educational researcher, 40(8), 367-377.

Srinivas, N., & Deb, K. (1994). Muultiobjective optimization using nondominated sorting in genetic algorithms. Evolutionary computation, 2(3), 221-248.

Marano, G. C., Sardone, L., & Licari, A. IL DESIGN PARAMETRICO PER
L'OTTIMIZZAZIONE DI UNA COPERTURA RETICOLARE.

Zitzler, E., & Thiele, L. (1998, September). Multiobjective optimization using evolutionary algorithms—a comparative case study. In International conference on parallel problem solving from nature (pp. 292-301). Springer, Berlin, Heidelberg.

Vierlinger, R., & Hofmann, A. (2013, October). A framework for flexible search and optimization in parametric design. In Rethinking Prototyping-Proceedings of the Design Modelling Symposium Berlin, Berlin.

7. ACKNOWLEDGEMENTS

Throughout the writing of this thesis I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Giuseppe Marano, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I would also like to thank, Laura Sardone, for her valuable guidance throughout my studies. You provided me with the tools that I needed to choose the right direction and successfully complete my dissertation.

I am also very grateful to all those at the One Works office, especially Mr. Riccardo Pauletto and others who were always so helpful and provided me with their assistance throughout my dissertation.

In addition, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, I could not have completed this dissertation without the support of my girlfriend, Sinem Sisman, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.