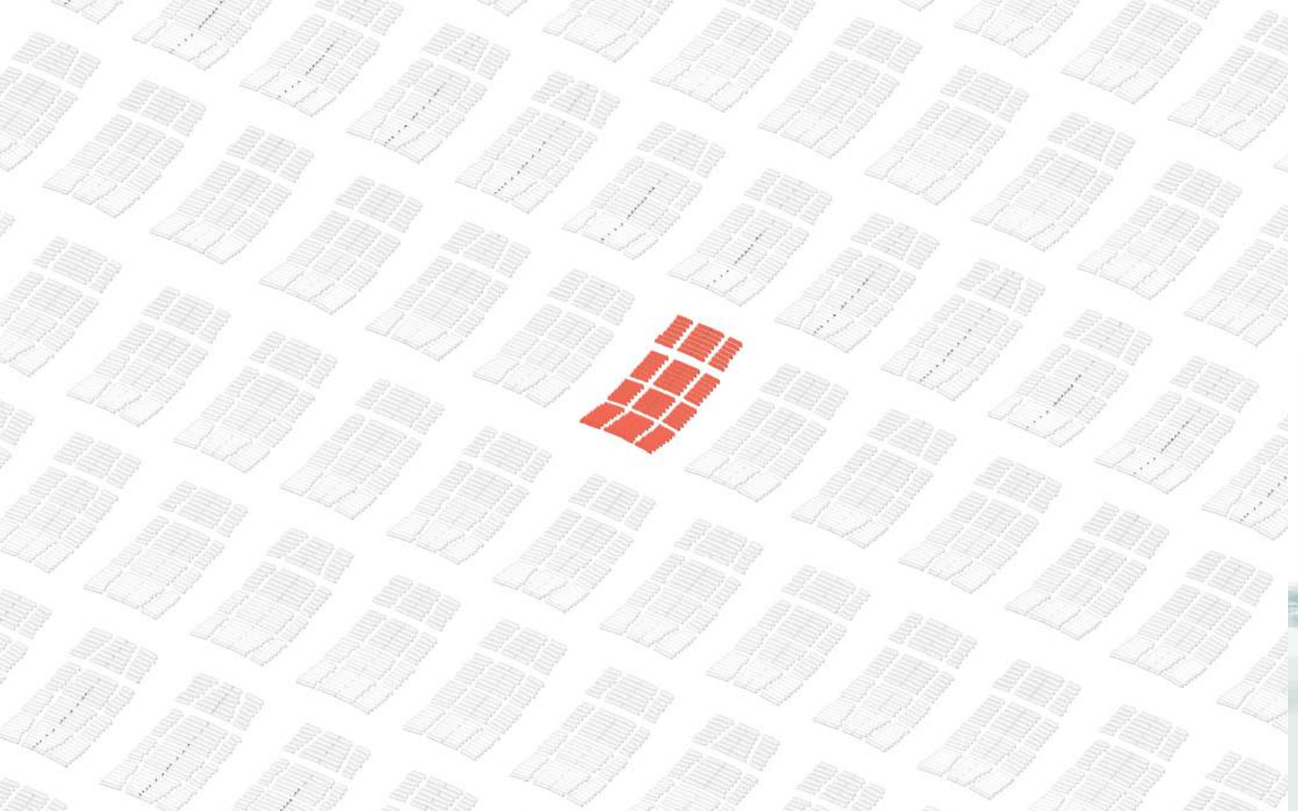


Computational and Algorithmic Design for Design Innovation in Architecture

Optimization Process in Architectural Design

H Architecture

This research paper was written and edited by Dongil Kim, Jake Jaekyung Han and John Hanghyun Cho at H Architecture. 2018.

Computational and Algorithmic Design for Design Innovation in Architecture

Optimization Process in Architectural Design

H Architecture

Computational and Algorithmic Design for Design Innovation in Architecture

Optimization Process in Architectural Design

Contents

Introduction

Introduction

Optimization Process in Architectural Design

The “Optimization Process in Architectural Design” is developed by H Architecture, New York, to identify various applications of information in architecture practices and fabrication processes.

As introduced in “Data, Information, and Architecture” (H Architecture, New York, NY, 2017), data is simply facts or figures - bits of information, but not the information itself. The “Computational Design in Practice” introduces architects and designers to the data as design parameters that could be utilized to optimize the building performances and the methods to achieve such results efficiently through computational design.

The “Optimization Process in Architectural Design” covers the applications of the repetitive problem-solving method known as the ‘Evolutionary Computing’ at different architectural design stages- from the strategic location of building masses to the development of modules to help the architects and designers to develop the optimal design proposals through iterations.

2014 – The "Domestic Hats" was patented for technical illustrations that became popular in the Industrial Revolution. It typically contained tools for accurate drawing like a protractor and compass.

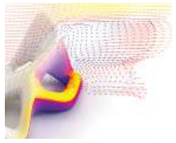
2009 – Phare Tower Option Study Models by Morphosis Architects

A History of Evolutionary Computing

Repetitive Problem Solving Technique

The Evolutionary Computing emanates from the "Theory of Evolution by Natural Selection" by Charles Darwin (Darwin, Charles. On the Origin of Species by Means of Natural Selection, Or, the Preservation of Favoured Races in the Struggle for Life. London: J. Murray, 1859.) The Theory of Evolution states: organisms with physical or behavioral traits that allow them to better adapt to their environments have higher chances of survival and have more offsprings. The process of Evolutionary Computing resembles such a process. Lawrence J. Fogel, a pioneer in Evolutionary Computing, first used the notion of Evolutionary Computation in the 1960s with a support from the U.S. Government. The application of the evolutionary computing in design processes emerged in the 1980s with the advent of personal computers. As the Computer Aided Design software such as AutoCAD, FOrm Z, Maya, and Rhinoceros were more frequently used, the 'Evolutionary Computing' became more accessible as an architectural design procedure. Despite its potential for practical uses, the evolutionary computing software is generally perceived as software for the programmers only due to its complexity and was only used in a few projects during the last few decades.

Top: Geco Grasshopper Simulation
ICD/ITKE Research Pavillion 2016 Structure Model
The Tower by CITA Interior
Lower Manhattan Modeling by NYC Opendata
ICD/ITKE Research Pavilion 2016 Robotic Sewing Process

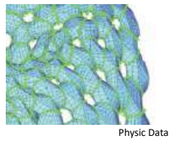


Environmental Data

Environmental Information Based Design

Rhinoceros + Grasshopper ——— Honeybee + Ladybug

Environmental Based Initial Massing in Pre-schematic
Facade system Analysis in Schematic and Design Development Phase



Physic Data

Structural Information Based Design

Rhinoceros + Grasshopper ——— KARAMBA 3D
Revit + Dynamo ———

Initial massing and Integrative Design System for parametric structure
in Pre-schematic & Schematic Phase

Material Data

Force Information Based Design

Rhinoceros + Grasshopper ——— Kangaroo

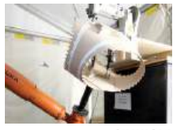
Column Free roof Structure, Stadium, Terminal, Airport,
Facade, Interior Feature in Pre-schematic & Schematic Phase

GIS Data

Urban and Geographical Information Based Design

Rhinoceros + Grasshopper ——— GIS, Local Code, Elk

Urban Design in Pre-schematic & Schematic Phase



Process Data

Advanced Geometry Analysis in Design

Rhinoceros + Grasshopper ——— Lunch Box
Revit + Dynamo ———

Optimization and Realization for Paneling of Complex Geometry in Design
Development & Construction Document Phase

Although the Evolutionary Computing is still rarely used, numerous other data analysis software became available to architectural designers through open sources. The prevalence of data analysis software allowed the Evolutionary Solvers to flourish in Rhinoceros + Grasshopper environment. The optimization in terms of "X" became available to architectural discipline through the evolutionary solvers and this new trend is called the "X" Driven Design. The Environmental Data Driven Design is a branch of this trend which is relatively common. An example of this process would be the extraction of view, radiation, glare, and daylight data through plug-ins such as Ladybug and HoneyBee and utilizing these data to simulate and evaluate for the optimal design solution through another plug-ins, Galapagos or Octopus. There are other simulation plug-ins such as Kangaroo Physics for the Force Driven Design and Karamba for Structure, Local Code, or Urban Data Driven Designs.

Research Objective

Application of Optimization in Multiple Design Phase

The objective of this research is to introduce the practical applications of the optimization software in architecture. The optimization process could be described as "Flexible" because the optimization results depend on the raw input data and setting of the user's objective.

This research will cover the methods to set appropriate objectives for the optimization software. The optimization process is similar to a conventional architectural design process that it simulates generations of similar options and selecting the better ones. The only difference is that a computer cannot distinguish the better or worse options. Thus, the user should prepare appropriate inputs for proper optimization, which cannot be done without the knowledge and experiences.

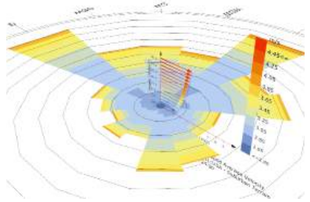
This research will go through the applications of evolutionary optimization at various architectural design stages to demonstrate its flexibility and validity.

Methods

Evolutionary Computing

Single-Objective Optimization

Multi-Objective Optimization



Environmental data

Environmental data is usually documented in numerical form, which can be simulated, evaluated, analyzed and visualized relatively easily. This characteristic is convenient for the evolutionary computing.

Geometric data

Geometric data in 3D modeling space is organized with numerical matrixes such as a point (x,y,z), a line (x,y,z *2), and a surface (x,y,z *4). In addition to the coordinates, the number of items is also numerical data.

GIS data

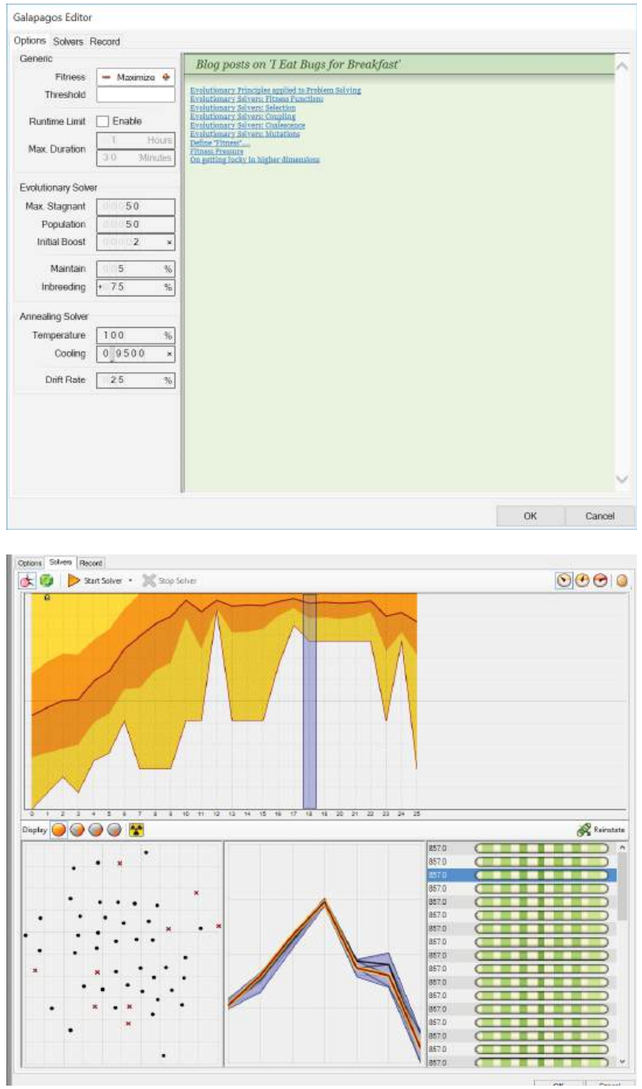
Not all but some of the GIS data, such as the number of water complaints, the price of the rent, or the population in a region, can be converted to numerical data and used in the design process.

Evolutionary Computing

The Evolutionary computing in Rhinoceros + Grasshopper can be utilized in conjunction with data analysis plug-ins to seek for the optimal design based on the environmental data. David Rutten, a software developer at McNeel & Associates, developed a plug-in called the “Galapagos” which facilitates the optimization process within the Grasshopper (A visual algorithm editor). In his lecture: ‘Computing Architectural Concepts’, Architectural Association, London, 2010, Rutten explained the basics of the Galapagos and its potential uses. The same material can be found in his personal blog - “Evolutionary Principles Applied to Problem Solving.”

“Octopus” is another plug-in for the Grasshopper that is similar to Galapagos. It is widely utilized for the application of evolutionary principles in parametric design and problem-solving. The Octopus produces a range of optimized solutions for multiple parameters. The critical difference between the two plug-ins is that the Galapagos can only process one parameter at a faster rate while the Octopus can process multiple parameters simultaneously.

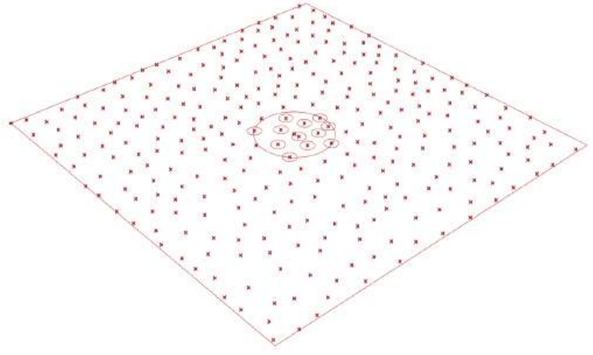
Evolutionary solvers are most effective when it is used with numeric data such as Environmental, Geometric, Structural and GIS data. The data types on the left are the examples of the appropriate numeric data, which can be extracted through several other plug-ins such as Ladybug, Honeybee, Kangaroo, Elk and many more.

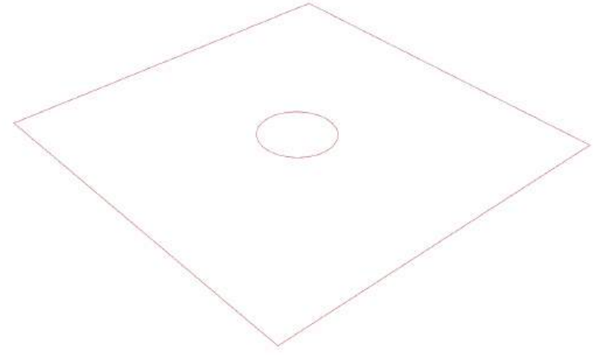


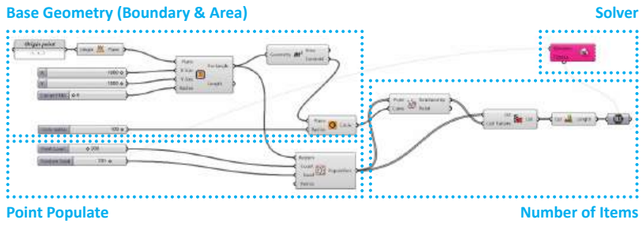
Galapagos Interface

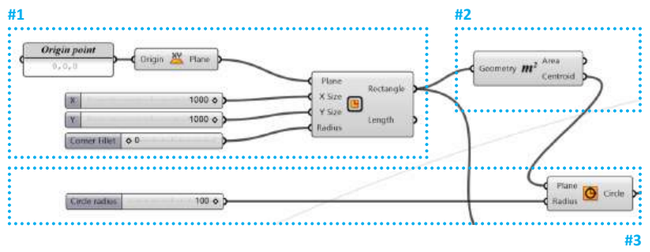
Single Objective Optimization

As its name suggests, the Single Objective Optimization can only process one-goal output. In cases that do not require multiple objectives, the Single Objective Optimization is preferred since it is faster to simulate and easier to interpret. Galapagos is one of the most prevalent single objective optimizers in the Grasshopper environment. Its interface is composed of three major categories ('Options,' 'Solvers,' and 'Record') located on the top-left corner of its window. The 'Options' panel is organized to set up the behavior of the Evolutionary computing. The Fitness (Goal) option decides whether to simulate for the maximum or the minimum value of a given parameter. In the Evolutionary solver tab, Max means the maximum number of the generations to be calculated. As mentioned in the last chapter, the Evolutionary solver creates a collection of options in several steps called 'generations.' The Evolutionary solver categorizes the superior data and its proximate data as a 'Family' in a generation and iterates the next generation with the family. The 'population,' just below the max tab, indicates the density of the data to be simulated in a generation. In the 'Solvers' window, the simulation results are displayed. The upper part is the generation and data tree, and the lower part is a list of the generation selected above. 'Record' window shows the simulated data as a text.







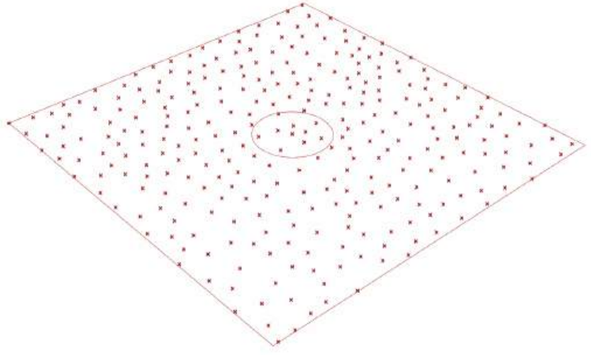


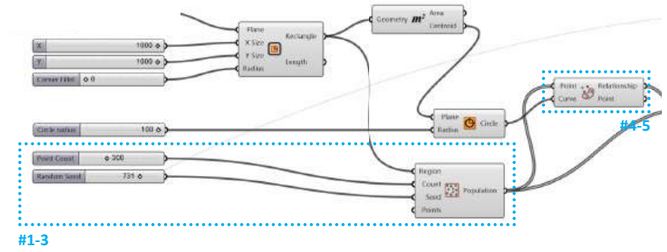
Single Objective Optimization Example

Above code is example of the fundamental application of the evolutionary computing in Rhinoceros & Grasshopper environment. The object of the overall coding is to find a specific random seed which maximizes or minimizes the number of the points. Base geometries are rectangular boundary and circular inner area. Points are populated randomly and according to the seed parameter, the location of the populated points are changed without changing the number of the total number of the points.

Base Geometries setting

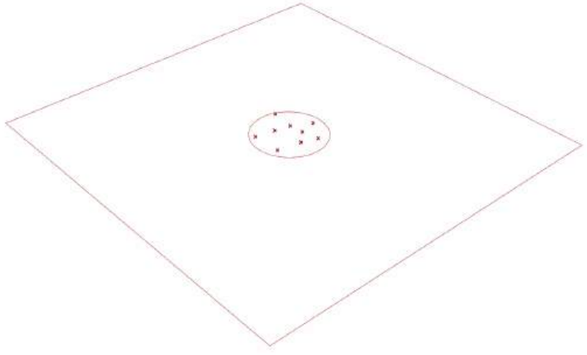
1. Create Rectangular boundary with given setting (plane, XY size, Radius)
2. Extract Center point of the rectangle via 'Area' Component
3. Create circle inner boundary with plane & any given number of radius

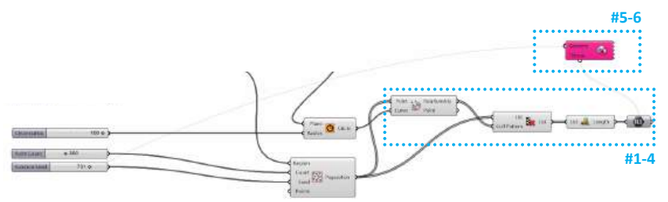




Populate points and Sorting

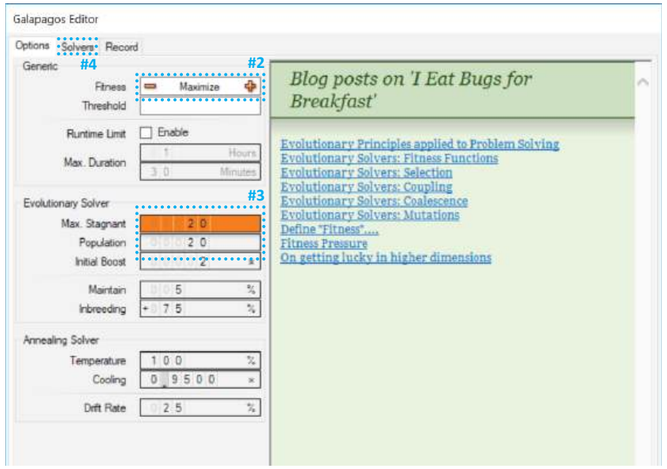
1. Connect the ‘Rectangle’ output to ‘Region’ input of ‘populate 2d’
2. Set the total number of the points at ‘Count’ input
3. Set the number slider with domain of 0 to 1000 and connect to ‘Seed’ input
4. Connect the ‘Circle’ output to ‘Curve’ input of ‘Point in Curve’
5. Connect the ‘Population’ output to ‘Point’ input of ‘Point in Curve’

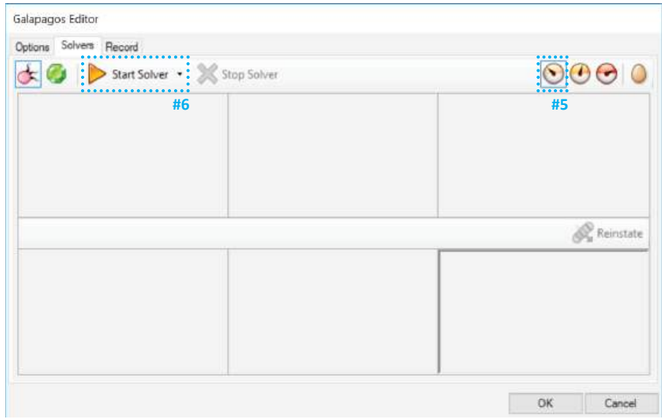




Connection to Solver

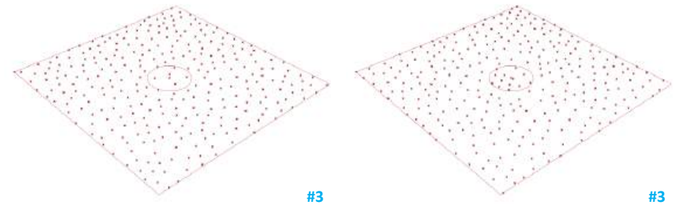
1. Connect ‘Relationship’ output to ‘Cull Pattern’ input of ‘Cull Pattern’
2. Connect ‘Population’ output to ‘List’ input if ‘Cull Pattern’
3. Connect ‘List’ output to ‘List’ input of list length
4. Converting the data type to ‘Number’
5. Connect ‘Genome’ to ‘Random seed’ number slider
6. Connect ‘Fitness’ to result ‘Number’

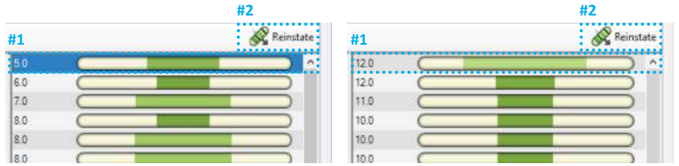


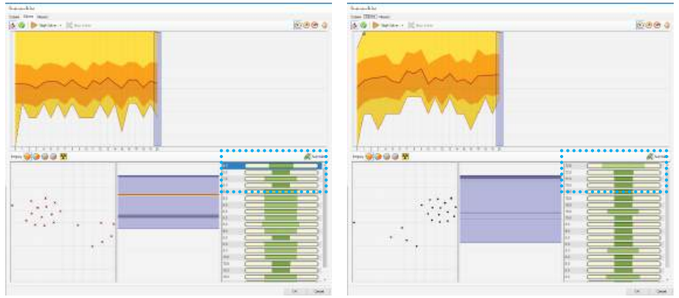


Galapagos Setting & Run

1. Double Click the Galapagos Solver to open up the pop-up window
2. Set 'Fitness' as 'Maximize' or 'Minimize' depending on the purpose
3. Set 'Max. Stagnant' and 'Population' to 20 for shorter simulation
4. Click 'Solver' tab for the Simulation
5. Click the left clock icon to visualize overall simluation in Rhino viewport
6. Start Solver

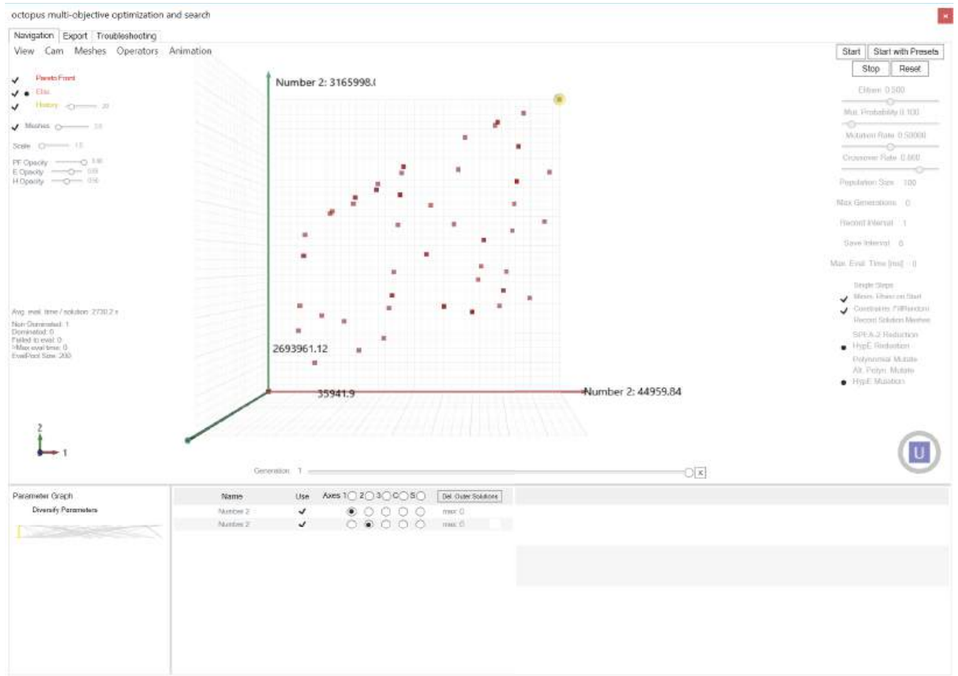






Optimization Results

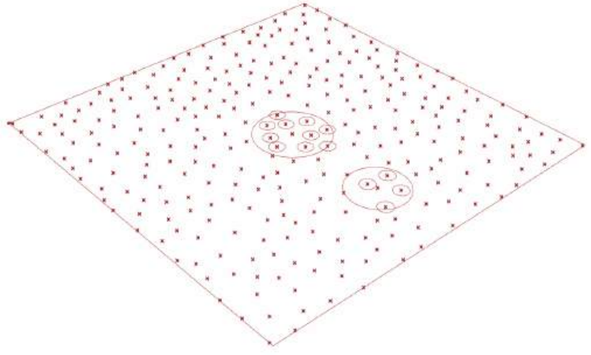
1. Click the optimized data
2. Reinstate the input params
3. Check the results (Maximized or Minimized)

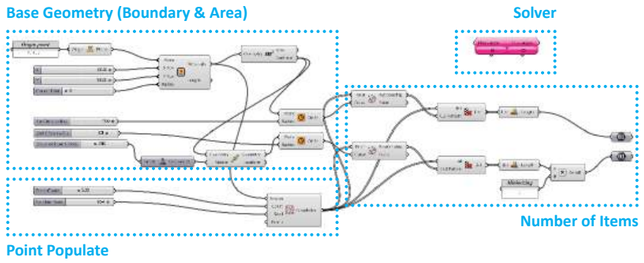


Octopus Interface

Multi Objective Optimization

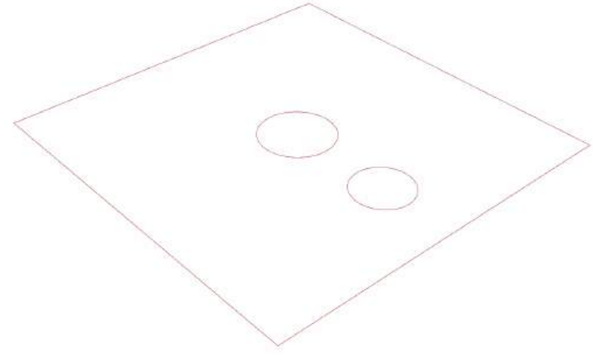
Unlike the Single Objective Optimization, The Multi Objective Optimization can have two or more controversial objectives as goals to achieve. The Multi Objective Optimization has a three-dimensional graph interface to display the complex simulation results. Octopus is the most prevalent plug-in for the Multi Objective Optimization. The 'Genome' tab of the Octopus undertakes the Genetic input of the Galapagos. The 'Octopus' tab, which is usually misunderstood as Goal, does the same function as the Fitness in Galapagos. While the Octopus can facilitate most of the Galapagos functions, it cannot prioritize for the minimum. To modify this issue, the users have to multiply the input data with a negative to convert the maximum into a minimum. The three-dimensional graph is another visualization method of the evolutionary computing, which is rather intuitive than logical. The main logic of the graph is that the distance between a point, which represents the simulation result of a specific option, and the origin point can be interpreted as the effectiveness of the option. This relationship could vary depending on the input data type, thus the result must be evaluated by a designer with the option's x,y,z values.

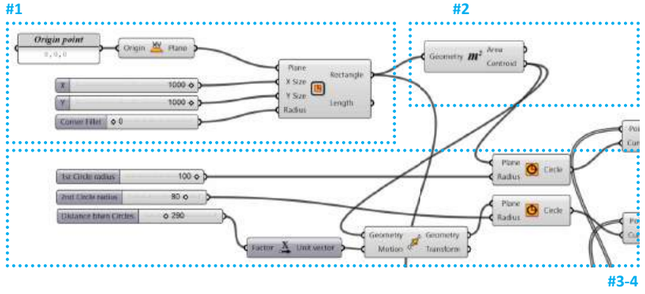




Multi Objective Optimization Example

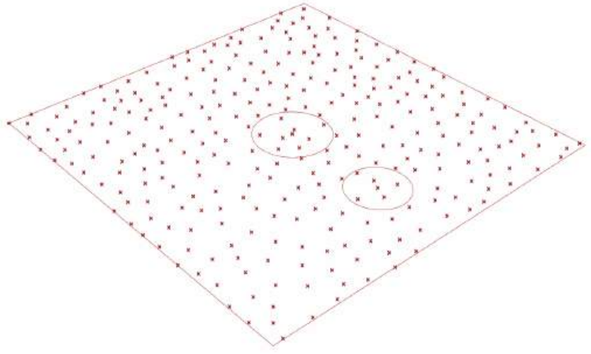
Above code is an example of the fundamental application of the multi objective optimization computing in Rhinoceros & Grasshopper environment. The object of the overall coding is to find a specific random seed which maximizes and minimizes the number of the points in a defined region. Base geometries are rectangular boundary and 2 circular inner areas. Points are populated according to the random seed as previous optimization coding.

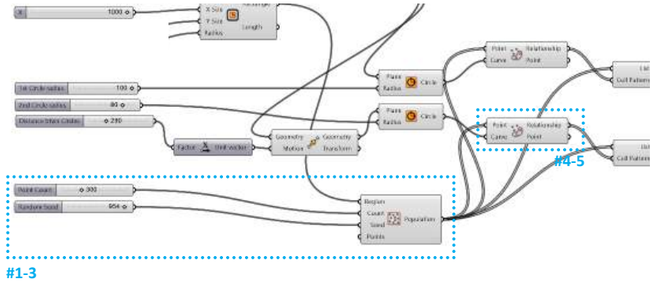




Base Geometries setting

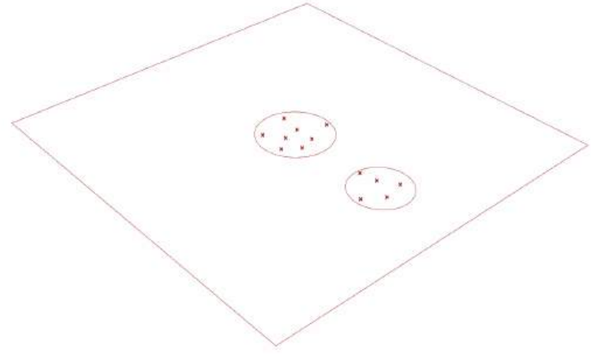
1. Create Rectangular boundary with given setting (plane, XY size, Radius)
2. Extract Center point of the rectangle via 'Area' Component
3. Create the first circle inner boundary with plane & any given number of radius
4. Create the second circle boundary in different location via 'move' component

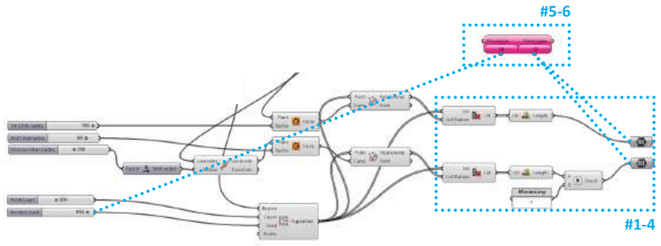




Populate points and Sorting

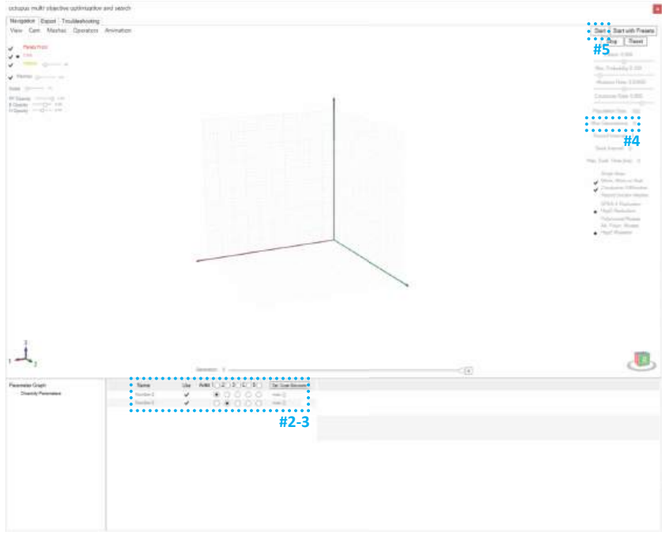
1. Connect the ‘Rectangle’ output to ‘Region’ input of ‘populate 2d’
2. Set the total number of the points at ‘Count’ input
3. Set the number slider with domain of 0 to 1000 and connect to ‘Seed’ input
4. Connect the ‘Circle’ output to ‘Curve’ input of ‘Point in Curve’
5. Connect the ‘Population’ output to ‘Point’ input of ‘Point in Curve’

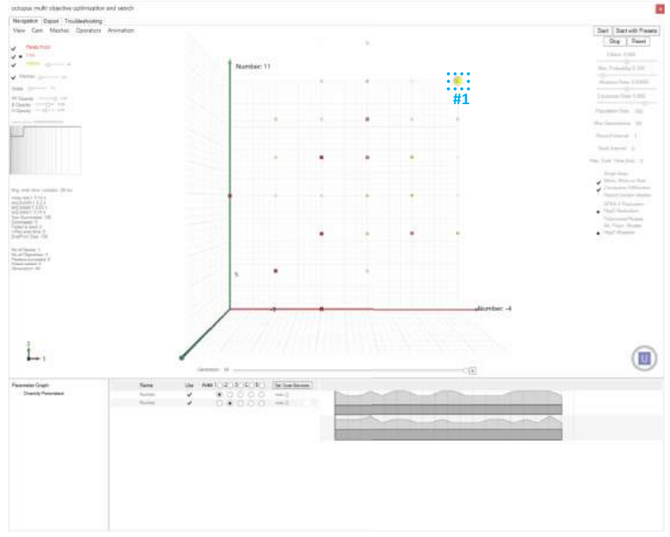


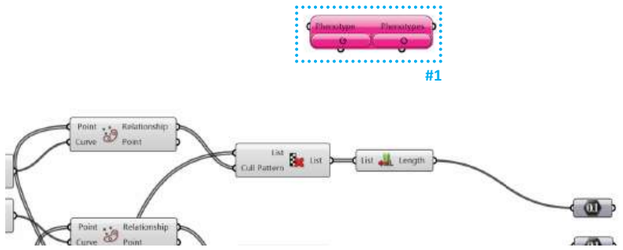


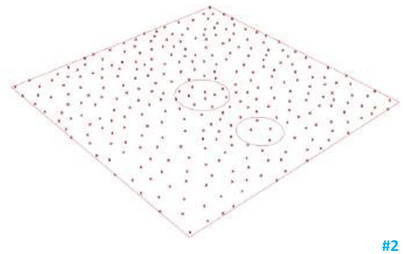
Connection to Solver

1. Connect ‘Relationship’ output to ‘Cull Pattern’ input of ‘Cull Pattern’
2. Connect ‘Population’ output to ‘List’ input if ‘Cull Pattern’
3. Connect ‘List’ output to ‘List’ input of list length
4. Converting the data type to ‘Number’ and Multiply ‘-1’ for minimize optimization
5. Connect ‘Genome’ to ‘Random seed’ number slider
6. Connect ‘Fitness’ to each result ‘Number’ components









Octopus Setting & Run

1. Double Click the Octopus Solver to open up the pop-up window
2. Check the parameters to be matched with the grasshopper coding
3. Set parameters to have individual axis by clicking the checkbox
4. Set Max generation (‘0’-unlimited / Typical setting is about ‘50’)
5. Start the solver

Optimization Results

1. Click the optimized option and select ‘reinstate’ in dropdown menu
2. Check the results (Maximized and Minimized)

Application

Evolutionary Optimization on Multiple Design Stages

Analysis Stage

Initial Design Stage

Developing Stage

Post Design Stage

Evolutionary Optimization on Multiple Design Stages

This research covers the applications of the Evolutionary Solver as a decision-making tool at various architectural design phases. An architectural design proposal is embedded with countless decisions as its process is a compromisation between various role players or parameters. Many decisions are made by definitive parameters-such as building codes or the cost. On the other hand, some parameters are more sophisticated to evaluate. The Evolutionary Optimization is an ideal procedure to iterate options through repetitive and complex numeric data.

Analysis Stage

The context analysis is typically the first step of the architectural design as most of the building design proposals are set in specific sites and environmental conditions.

Environmental data is a collection of numeric data. View, infrastructure, neighboring buildings, census, sun paths, wind roses, temperature, noise level to name a few are the examples of data types that can be collected as the raw material for the context analysis.

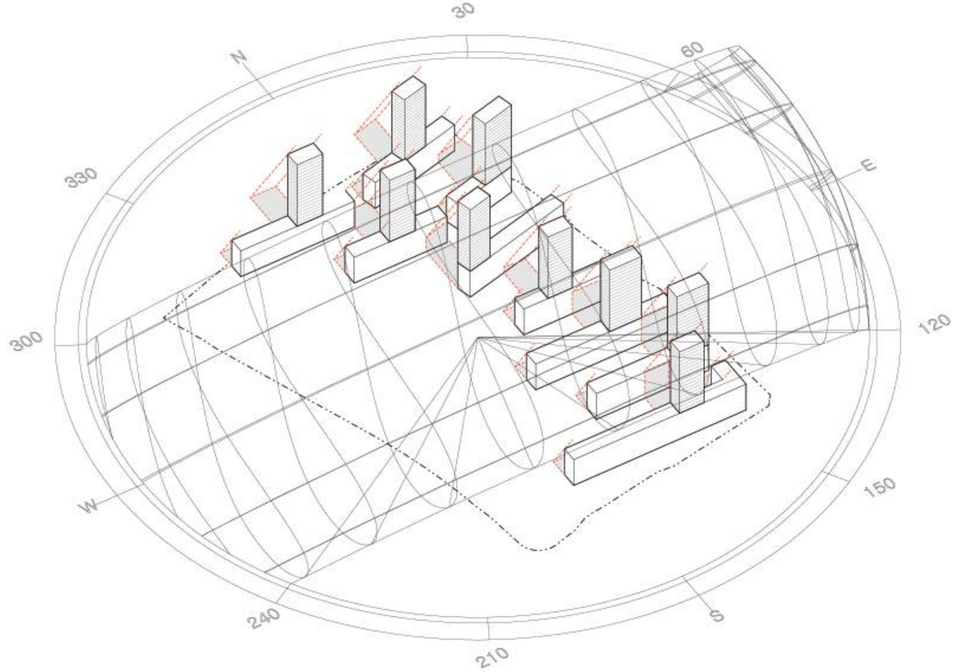
The computation converts the raw data into visual information for interpretation.

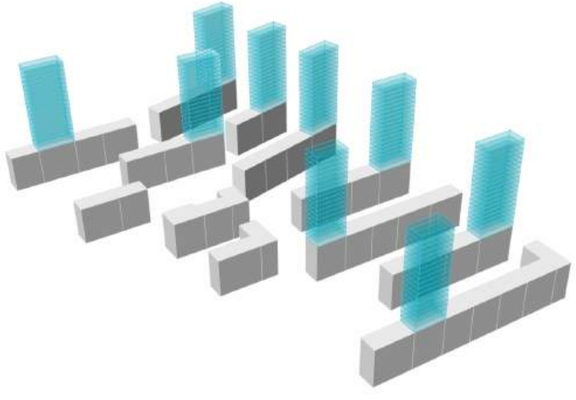
In the Analysis Stage, the 'Partial Building Massing Process' of "Sejong M2 Block project, South Korea, 2013" is used as a reference.

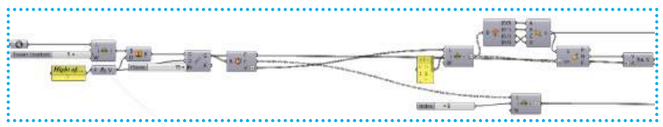
Sejong M2 Block project was done in 2013, South Korea. H-architecture participated in the project during the pre-schematic phase. This chapter will use the Sejong M2 Block project as a reference to discuss the partial application of the shadow range analysis and evolutionary computing at the early design stage.

Generally, in residential complex projects, one building's shadow on another building results in inferior units and negatively affects the value of the units in the shadow. To avoid this situation, the architects are paying more attention to guarantee as much daylighting as possible to all units.

The Sejong M2 Block project consists of bar type buildings below and tower type apartment buildings on top. The bars' locations are determined by regulations and knowledge of residential projects while the towers' locations on the bar are relatively flexible. The Evolutionary Optimization was used to simulate critical differences resulted from subtle differences of designs.


30
60
N
330
E
300
120
W
150
240
S
210

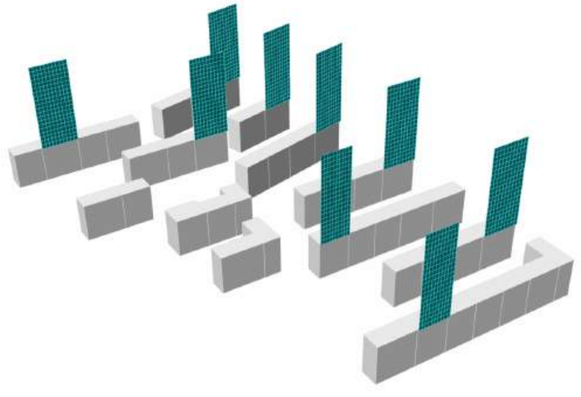


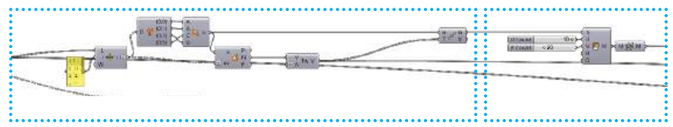


Base Tower Geometry
(x9)

Tower location setting

1. Divide the first podium massing according to the circumstaces(width, cores, columns)
2. Create tower massing with given height & number of the floors
3. Repeat the same process for other podiums (in this tutorial, total 9 podium massings)



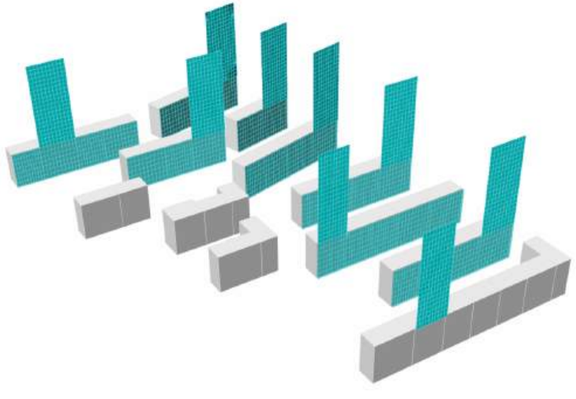


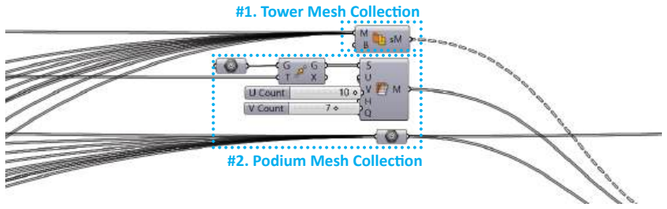
#1-2. Tower Front Surface Extraction
(x9)

Mesh Converting
(x9)

Analyzing mesh creation

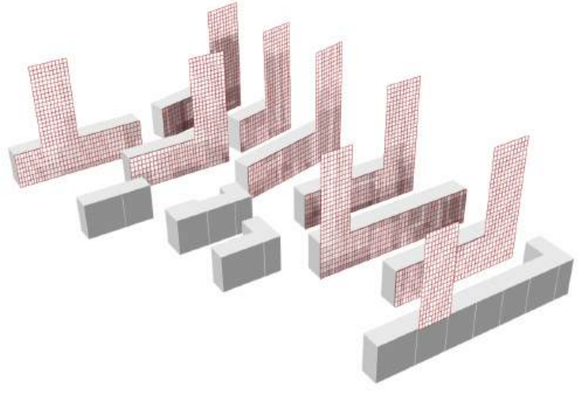
1. Extract front surface of the tower
2. Move the surface slightly above to avoid the surface overlapping
3. Convert the surface to mesh (U = n(unit) in a floor, V = n(floors))

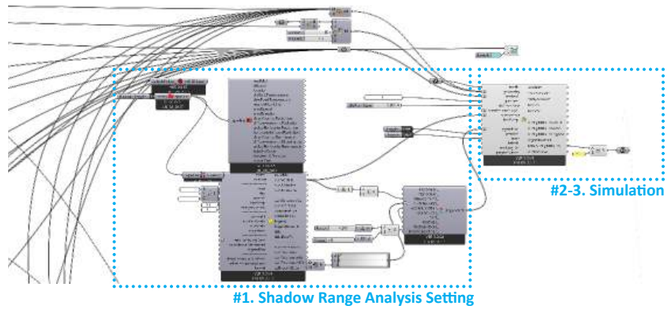




Podium surface extraction

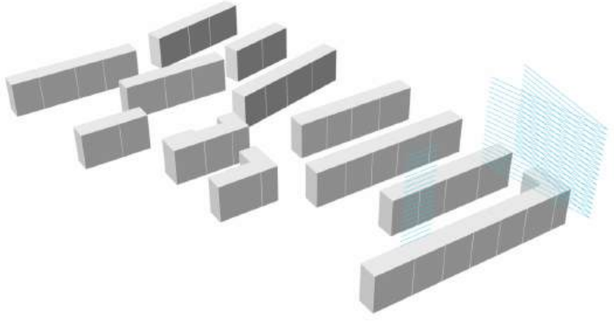
1. Collect the prepared tower front surface meshes
2. Collect the front surface of podiums (surfaces for analysis only)

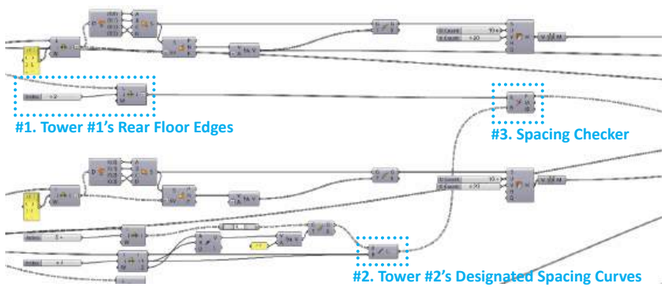




Shadow range (Minimizing Value)

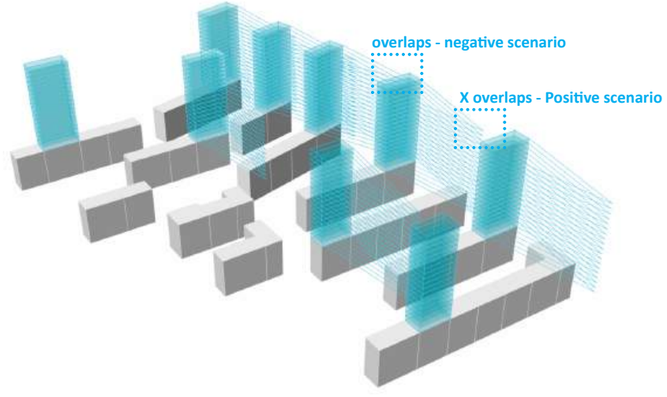
1. Set up the Base setting for shadow range analysis
2. Run the Simluation
3. Connect ‘Sunlight hours’ to ‘Number’ for data conversion

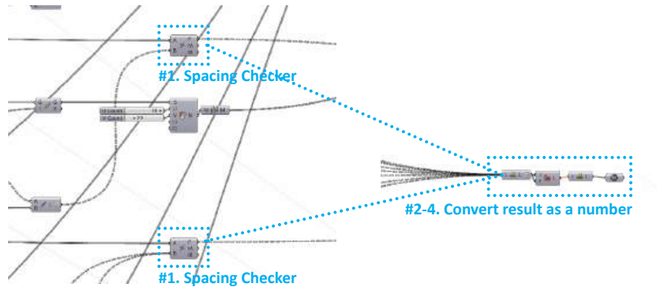




Tower spacing check

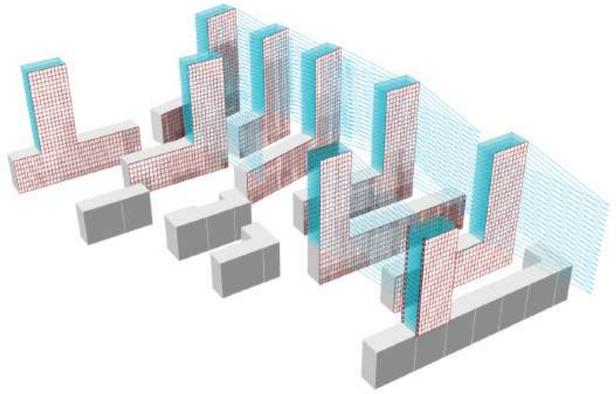
1. Extract rear edges of the front tower's floors
2. Extract spacing curves from backward tower's floor edges
3. Create 'Curve / Curve' component to check overlapping of the tower locations

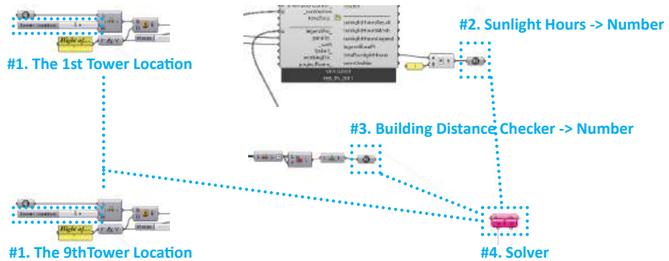




Relative spacing check (Minimizing Value)

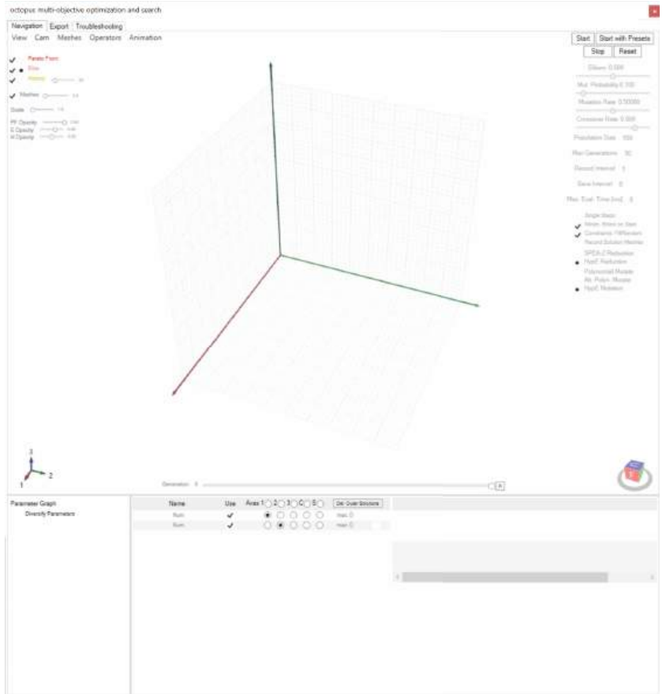
1. Collect the 'Point' output of the 'Curve/Curve' component to 'List length' Component
2. Cull the non-overlapping values with 'Cull pattern' component
3. Recalculate the length of the cull patterned list via 'List length' again
4. Convert the data type to numbers





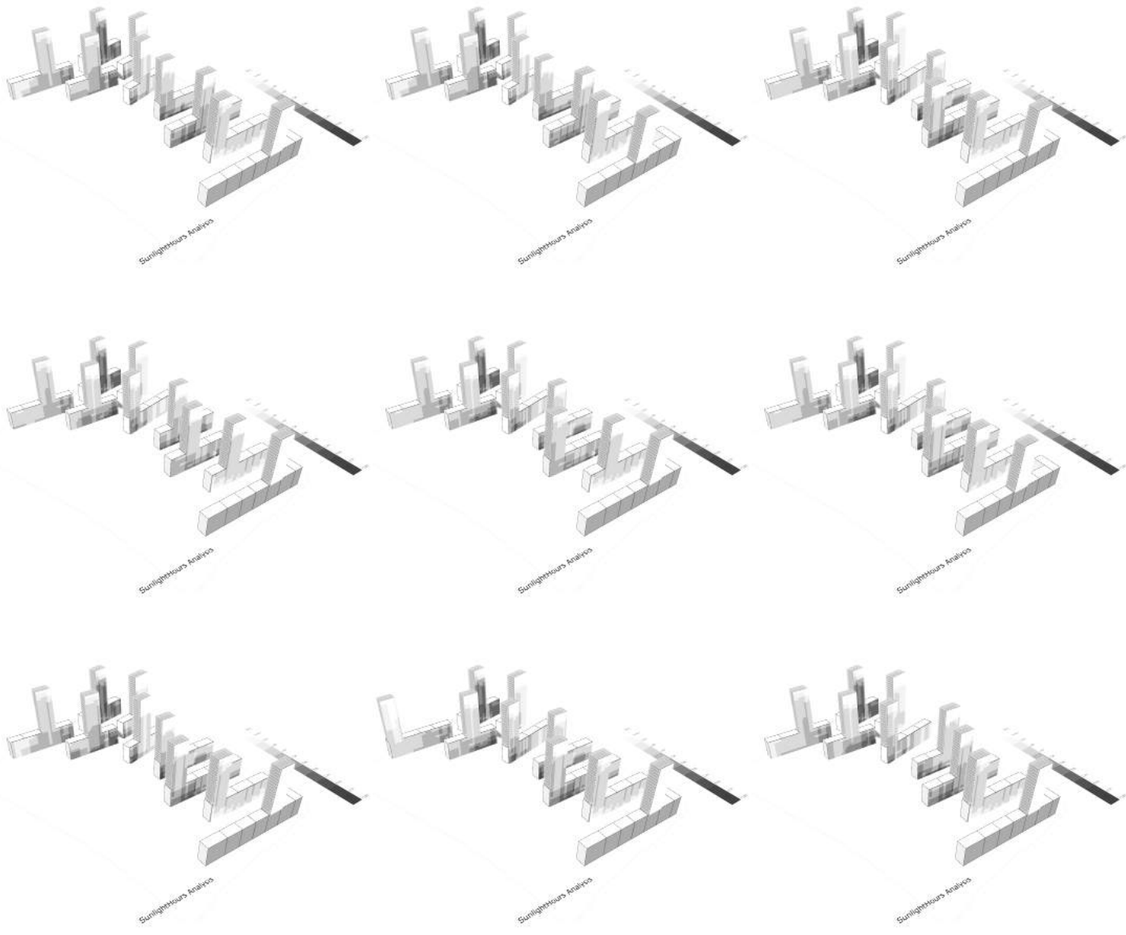
Octopus Connection

1. Connect the ‘Tower Location’ params(1-9) to ‘Genome’ of the Octopus solver
2. Connect converted numeric data of the sunlight hours to ‘Octopus’ of the solver
3. Connect converted numeric data of the distance checker to ‘Octopus’ of the solver
4. Double click the solver to open up the pop-up window

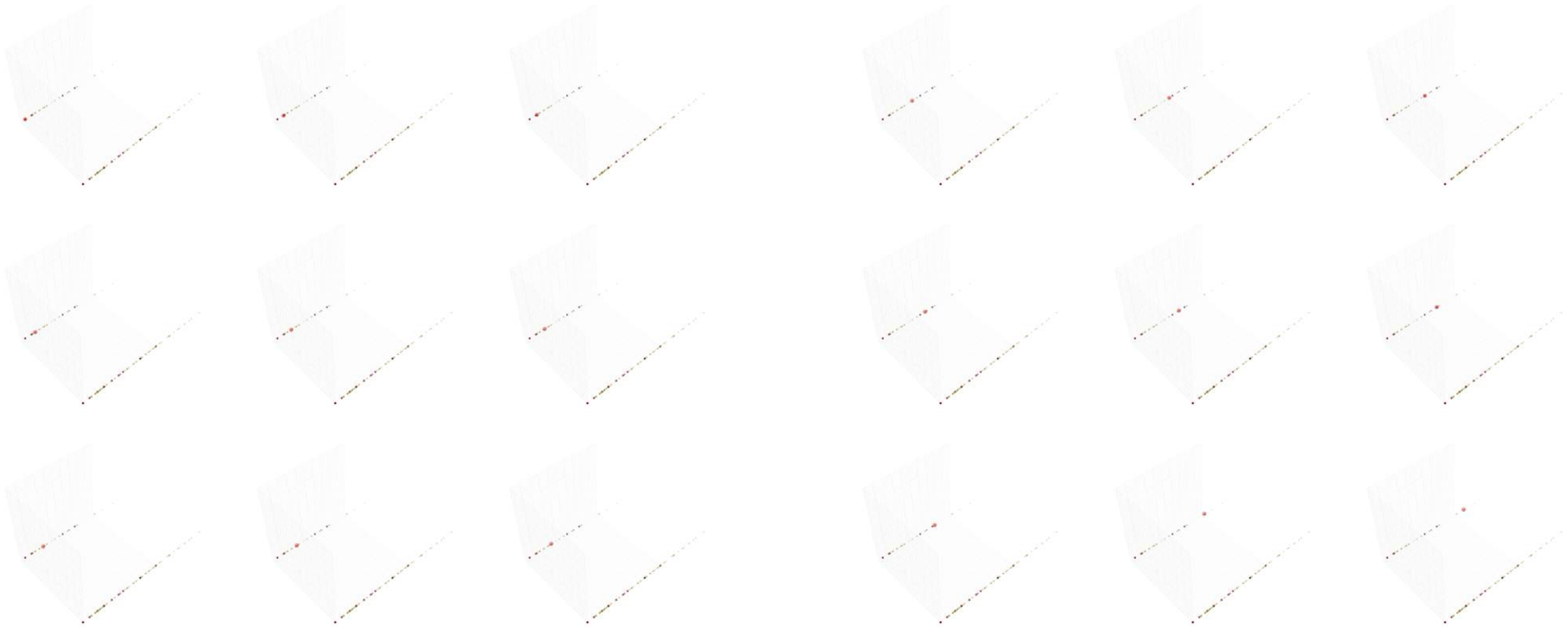


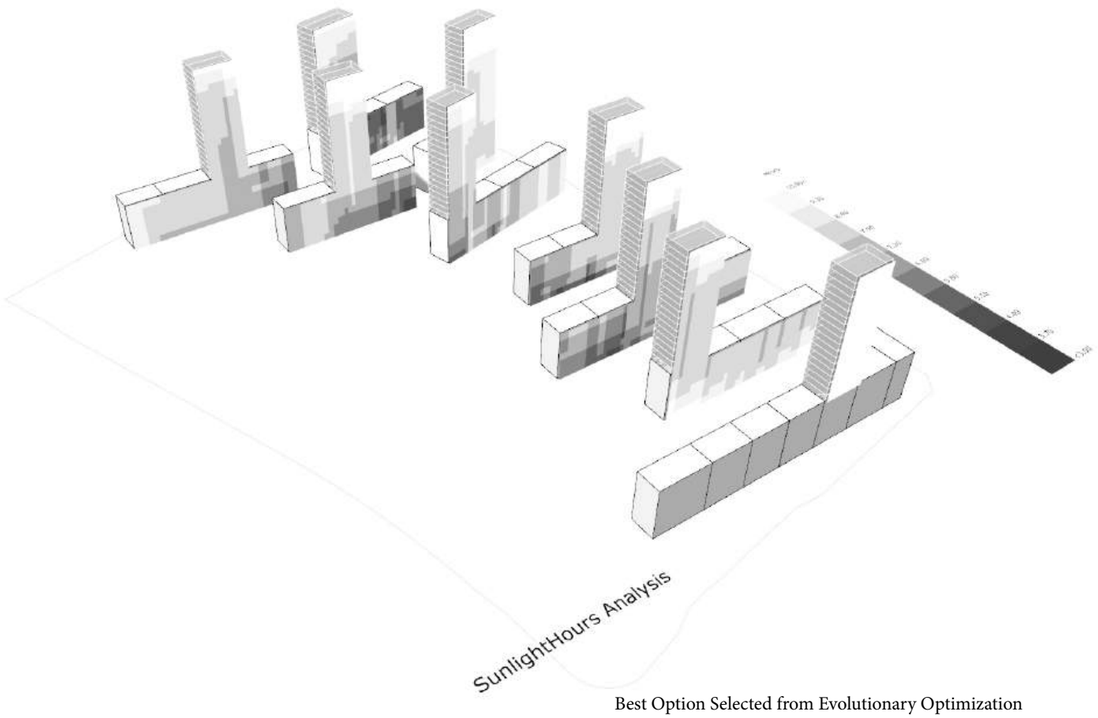
Run the simulation

1. Set up the max generation according to circumstances
2. Check the numbers are having individual axis
3. Click ‘Start’ to run the simulation



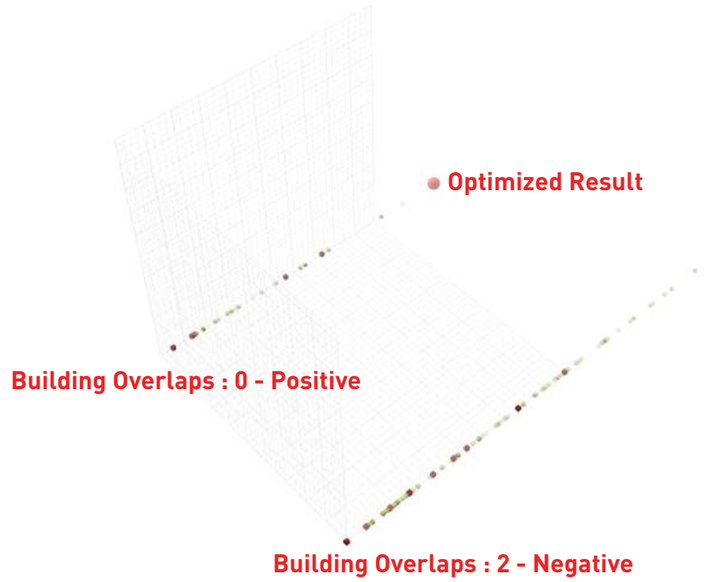
Options simulated via Evolutionary Optimization



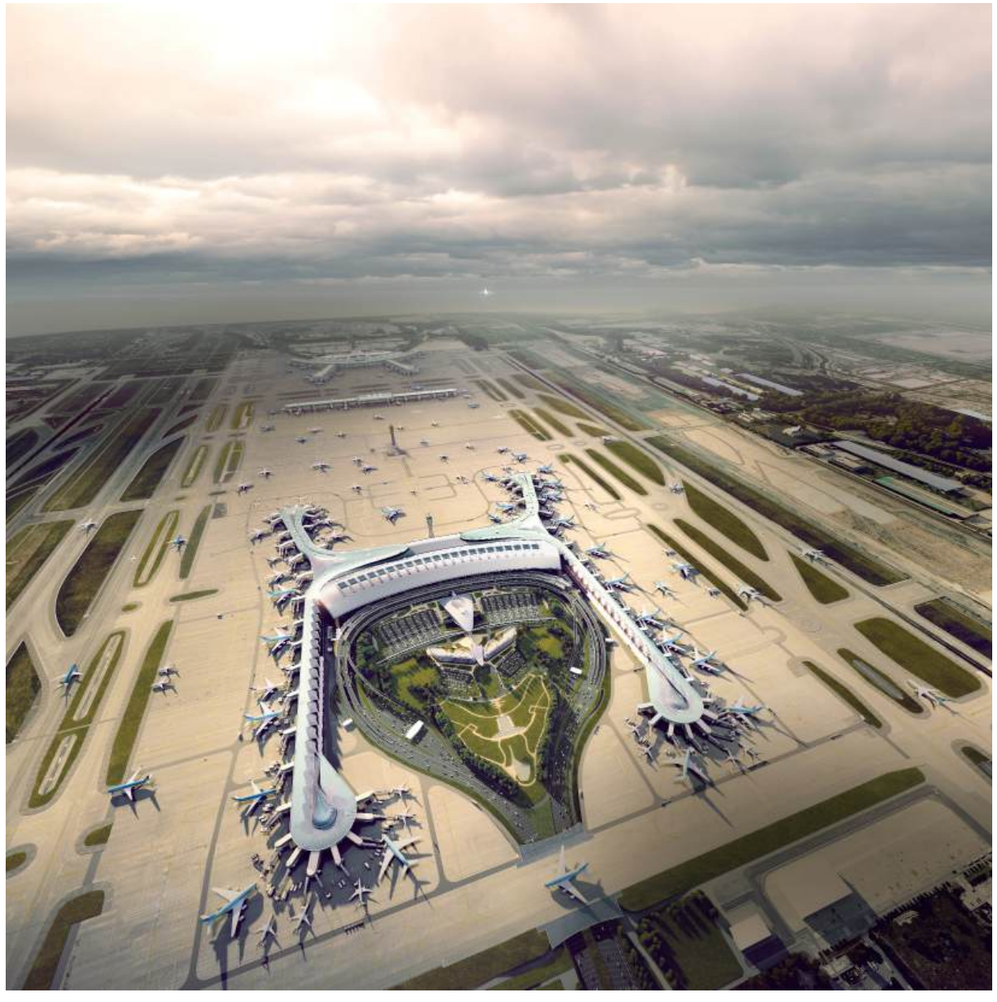


Best Option Selected from Evolutionary Optimization

Best Option Selected from Evolutionary Optimization



The tower casts less shadow to an adjacent building in this location and orientation. The Shadow Range Analysis of the Ladybug plug-in was used to measure the shadow range. The analysis code was plugged-in to the 'Octopus', a multi-objective evolutionary solver, to simulate various tower locations and evaluate whether the locations of the towers were feasible. The net shadowing hours on the building segments were used to evaluate the tower location options. The graph above shows all of the simulated options and each point corresponds to a certain amount of shadow range. The image on the left has the least amount of shadow range in the entire project, which is indicated as the 'Optimized Result' in the graph above.



Initial Design Stage

Conventionally, a larger scale building is composed of repetitive elements called the "Modules." The modules are relatively flexible design elements than the overall building form that is limited by strict regulations, building codes and so on. Therefore, in many cases, the overall form and the orientation of the building may not be in the ideal condition. However, there is still a chance to improve the building performance by proper optimization of the modules.

Typically, buildings with horizontally elongated proportions are composed of repetitive modules. The airports and factories are the clear examples.

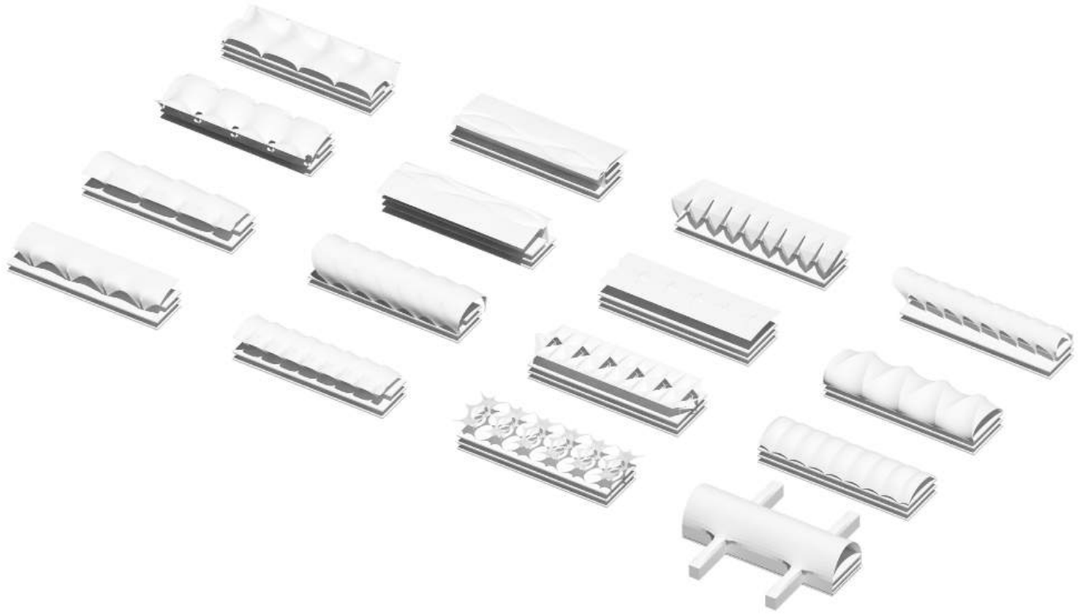
In the initial design Stage application, The 'Proto-type Form Finding Process of the Module Design' of "Incheon international airport Terminal 2 design, South Korea, 2017" is used as a reference.

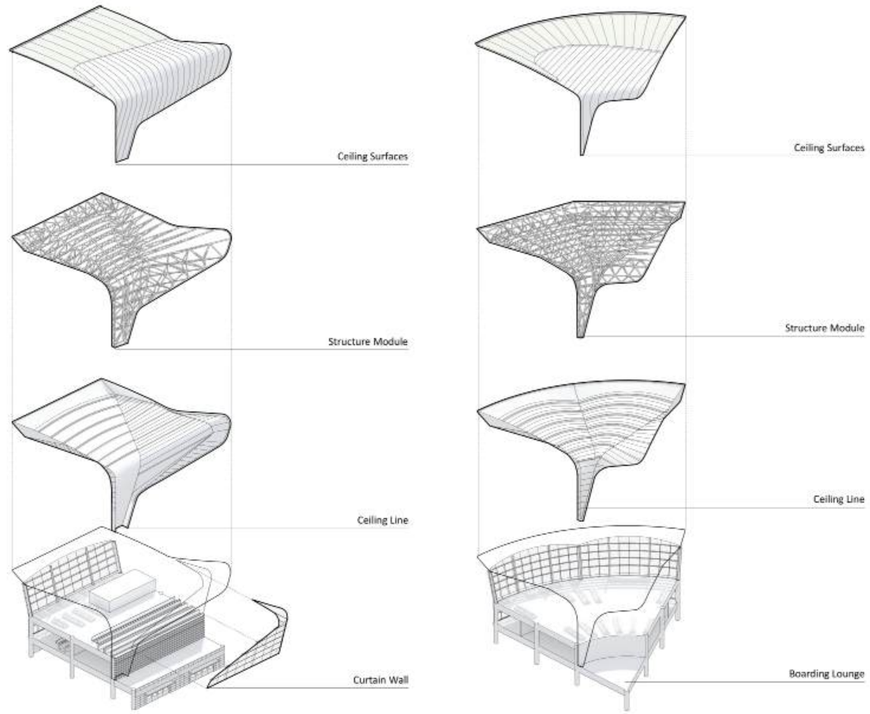
The Incheon International Airport Terminal 2 extension project was done in 2017 as a competition project. H-architecture participated in the entire process of the competition.

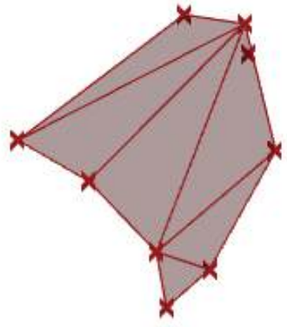
One of the most important parts of the project was to design a module that crosses the entire width of the airport lounge spaces to minimize the structural members obstructing the circulation.

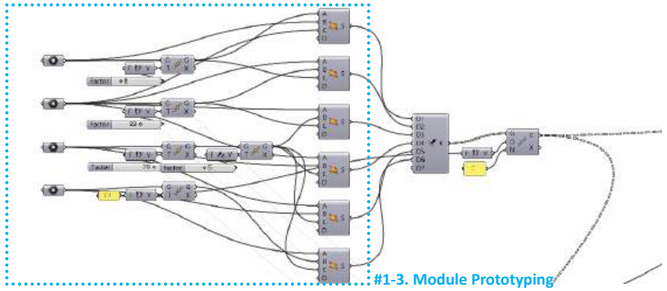
The view and daylight were the two design parameters of the modules. Ideally, a module would have a view to the airside and roadside from the interior and soft indirect daylight with minimal glare.

However, due to the orientation of the modules, the above criteria were conflicting elements. The evolutionary solver was utilized to simulate and evaluate for the optimal design option.



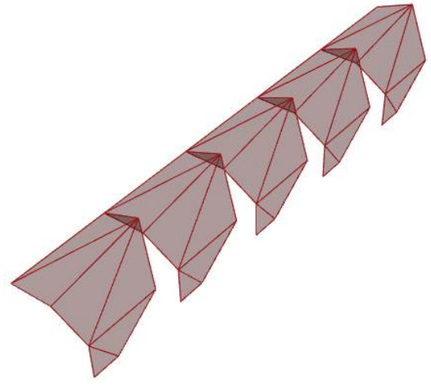

Ceiling Surfaces
Structure Module
Ceiling Line
Curtain Wall
Ceiling Surfaces
Structure Module
Ceiling Line
Boarding Lounge

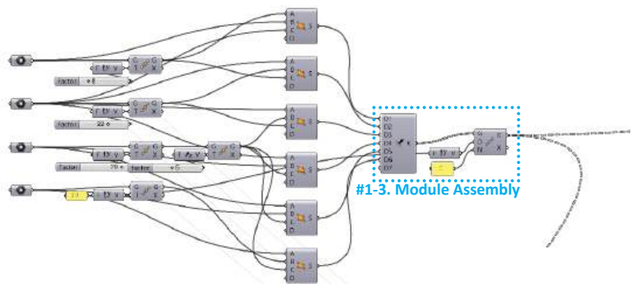




Module prototype creation

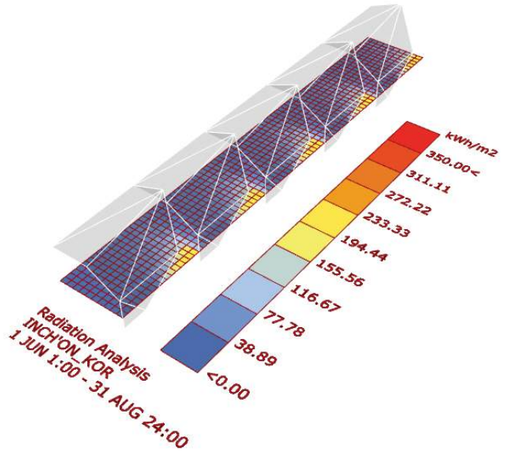
1. Create standard points and import it to Grasshopper via ‘Point’ component.
2. Set Movable parameters according to circumstances (Height limit, Columns, etc).
3. Connect points by creating triangulated surface for prototype geometry creation.

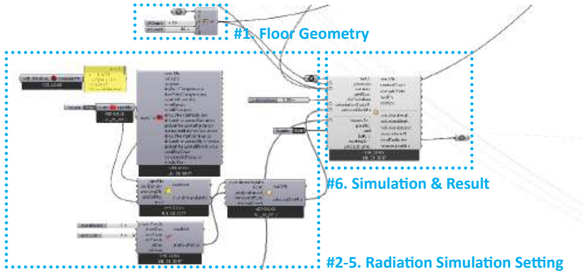




Array Modules for simulation

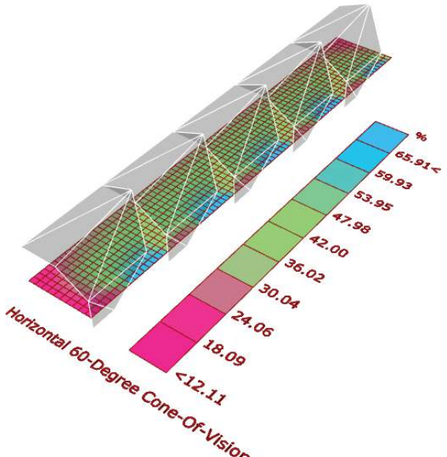
1. Assemble the surfaces via ‘Merge’ component.
2. Create repeated installation of the module via ‘Linear array’ component.
3. Numbers vary depending on each projects’ circumstances.

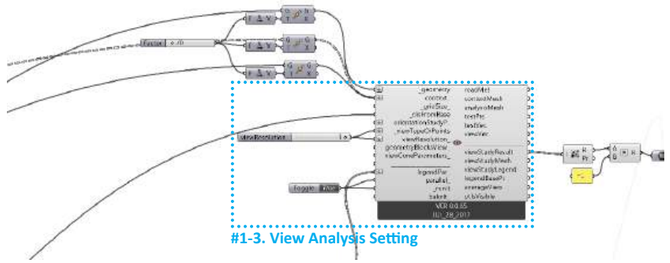




Radiation Analysis (Minimizing Value)

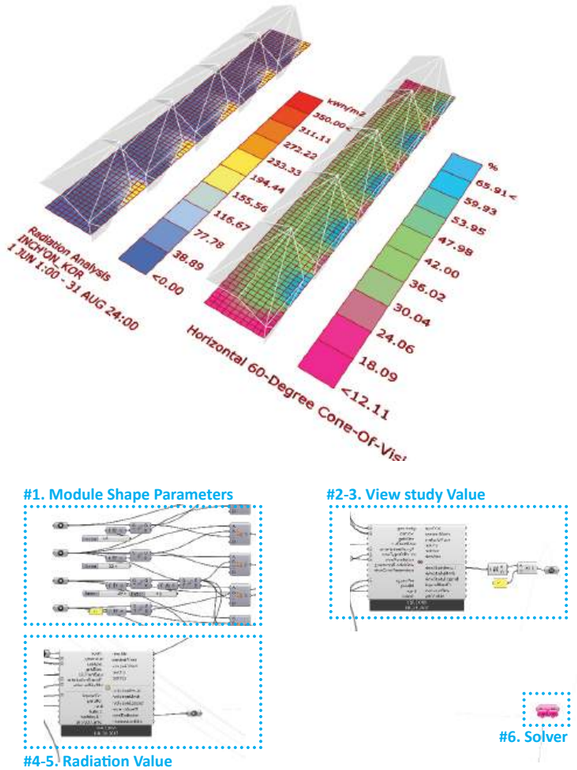
1. Create target floor geometry as a mesh with appropriate U,V value.
2. Create radiation simulation setting for the summer period.
3. Connect the module, simulation setting and floor mesh to radiation simulation.
4. Connect the target floor geometry to Geometry of Radiation Analysis.
5. Connect the arrayed modules to Context of Radiation Analysis.
6. Run the simulation.





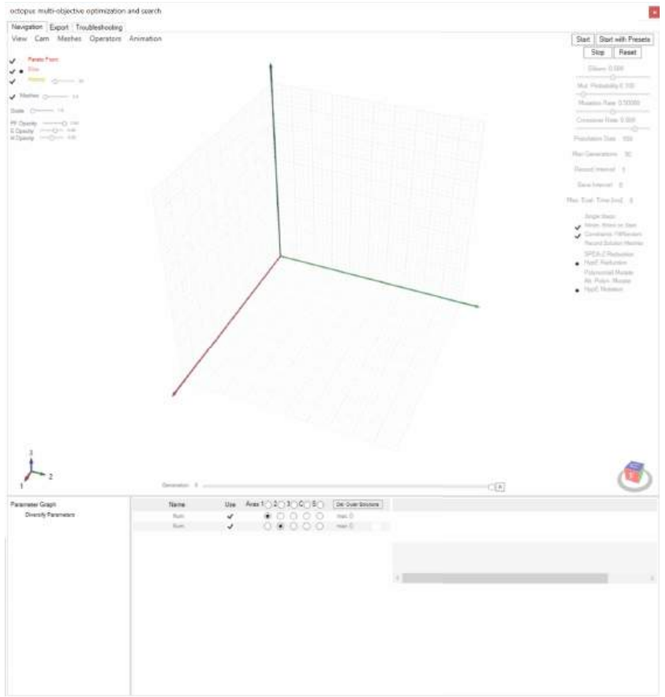
View Analysis (Maximizing Value)

1. Connect the target floor to Geometry of View Analysis.
2. Connect the arrayed modules to the Context of View Analysis.
3. Set appropriate integer (0-4) for View type and resolution.
4. Run the simulation.



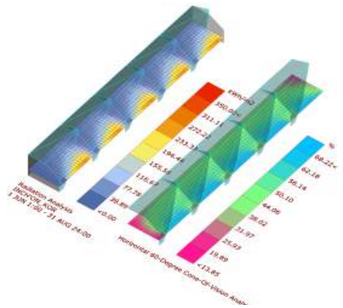
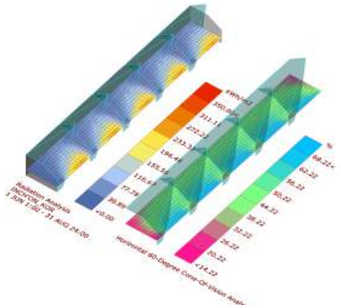
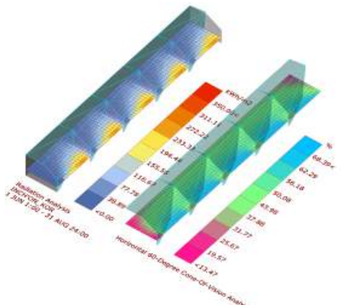
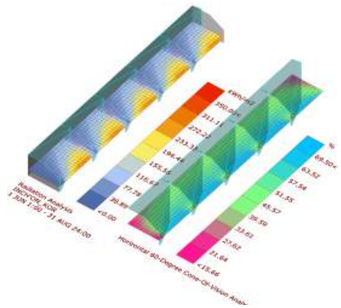
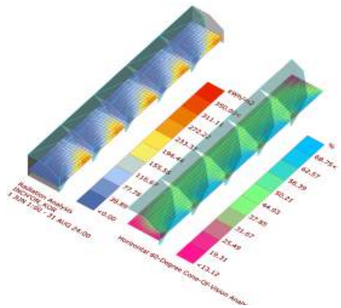
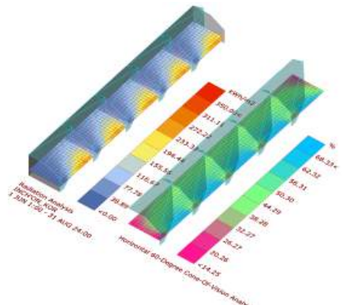
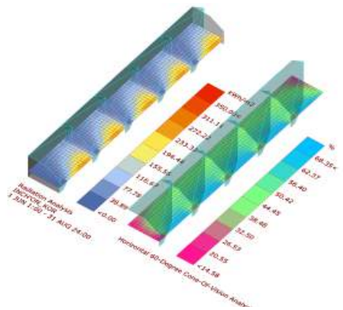
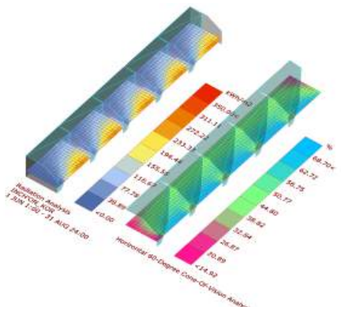
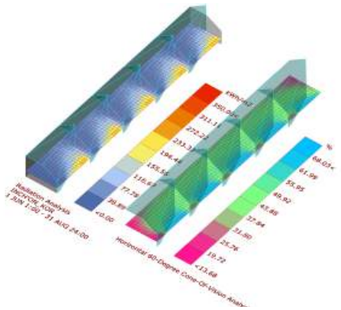
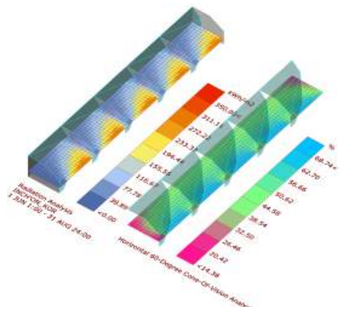
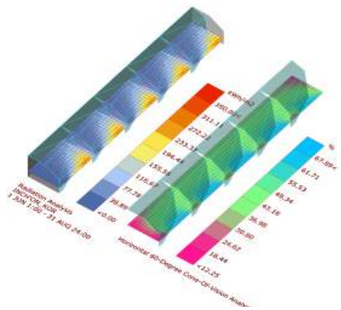
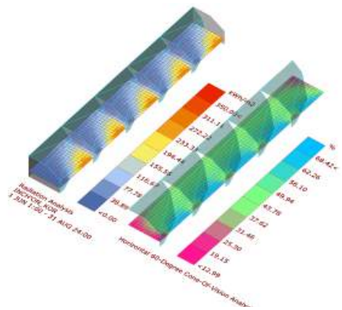
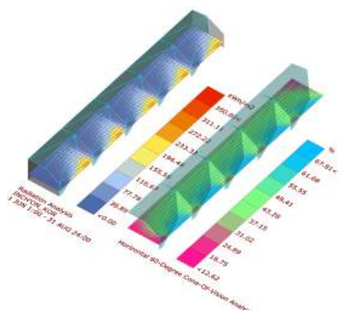
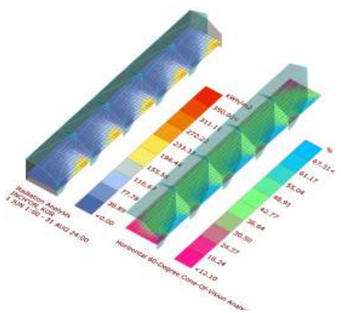
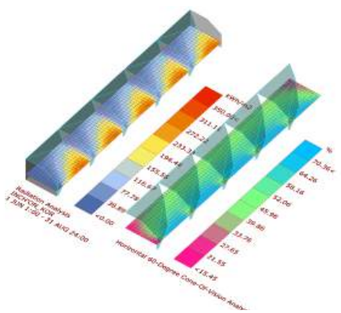
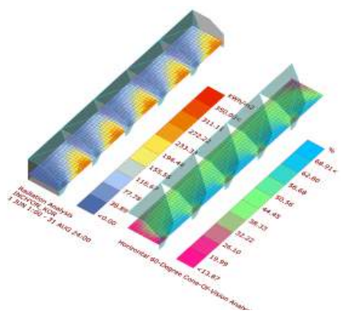
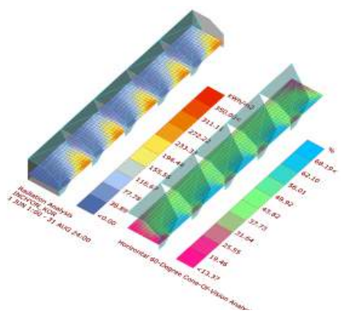
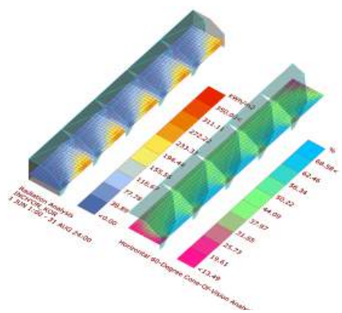
Solver

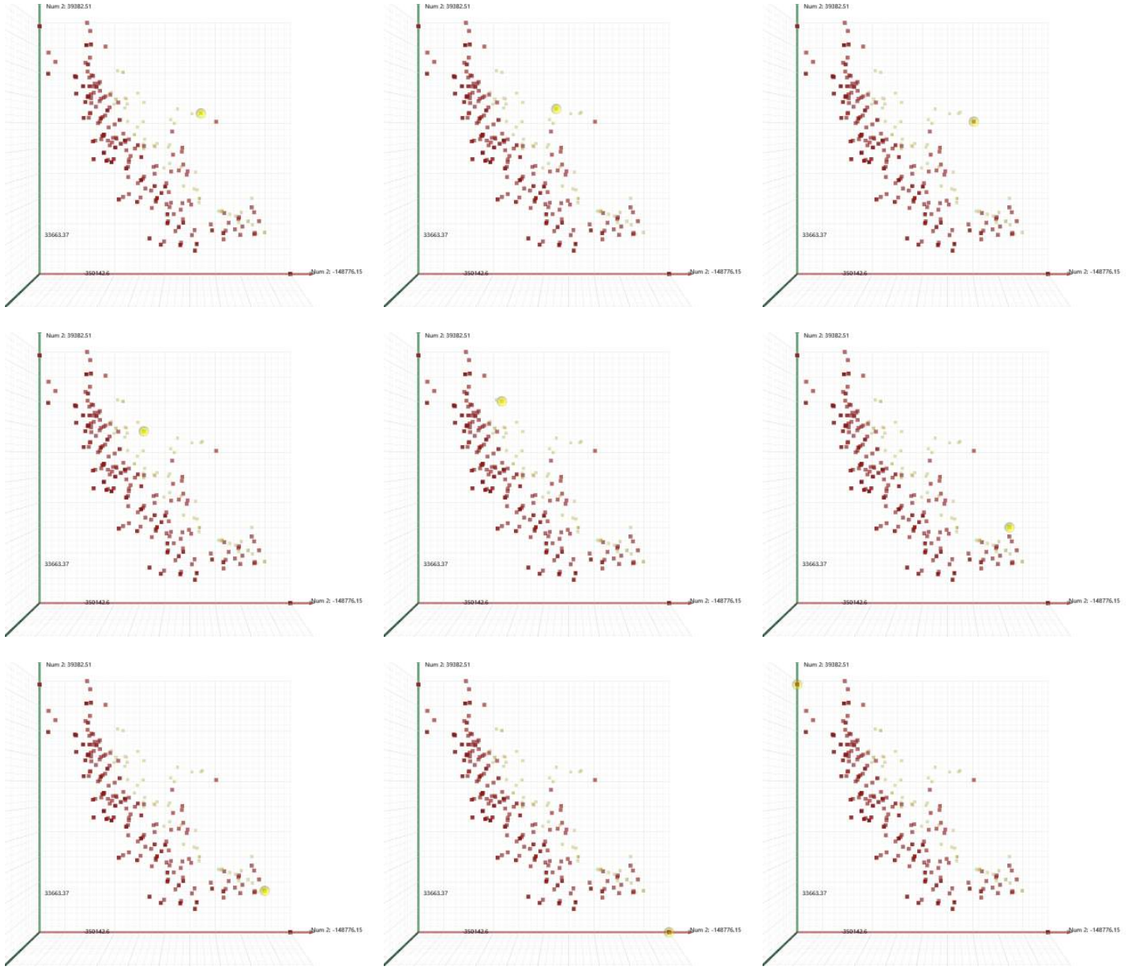
1. Connvet module shaping params to ‘Genome’ of the Solver
2. Convert ‘viewStudyResult’ output into ‘Number’ parameter
3. Connect the converted number to ‘Octopus’ of the Solver
4. Convert ‘totalRadiation’ ouput into ‘Number’ parameter
5. Connect the converted number to ‘Octopus’ of the Solver
6. Double click the solver to open up the pop-up window

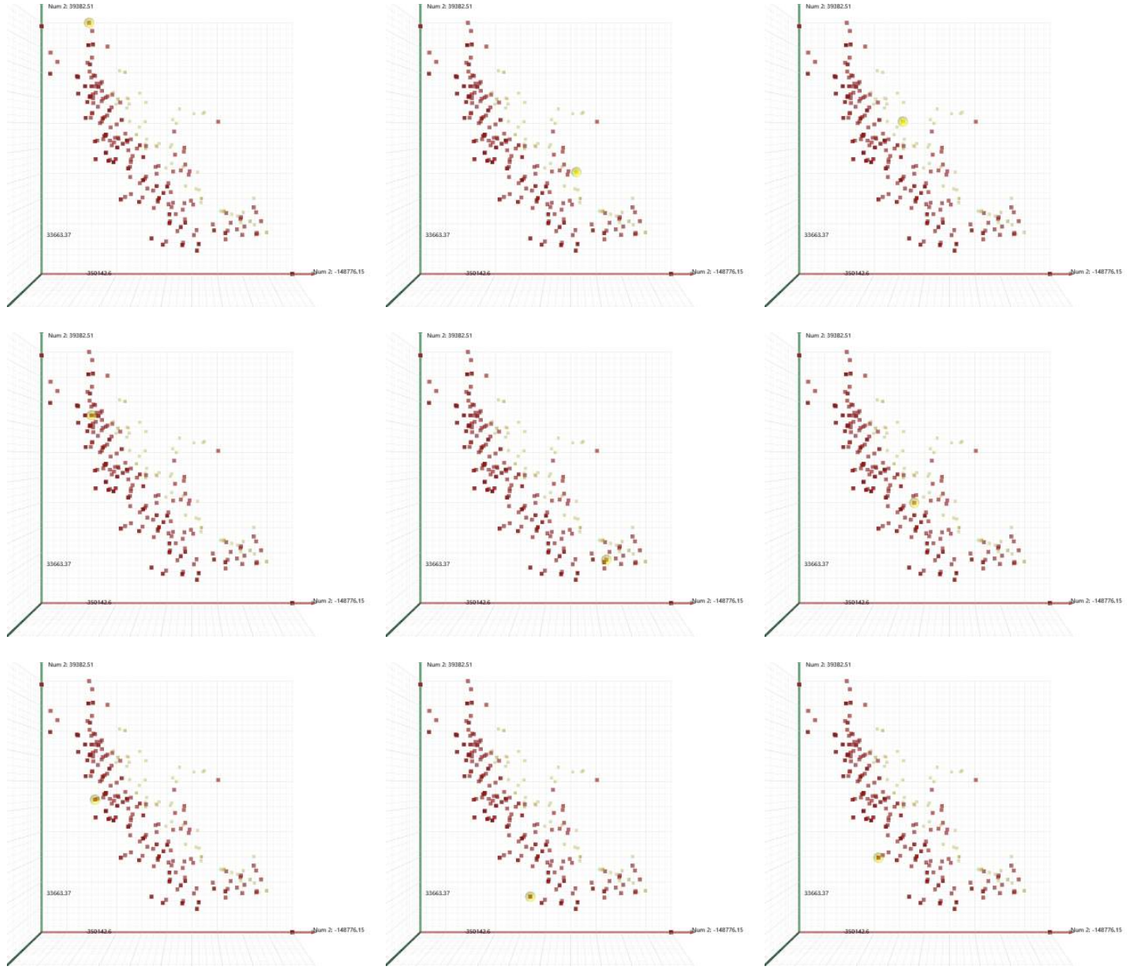


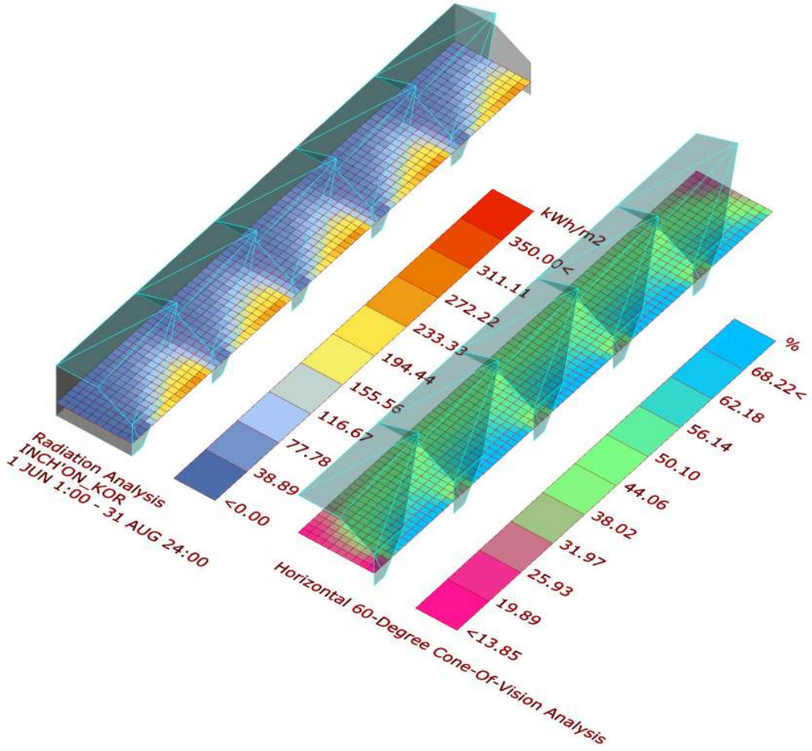
Run the simulation

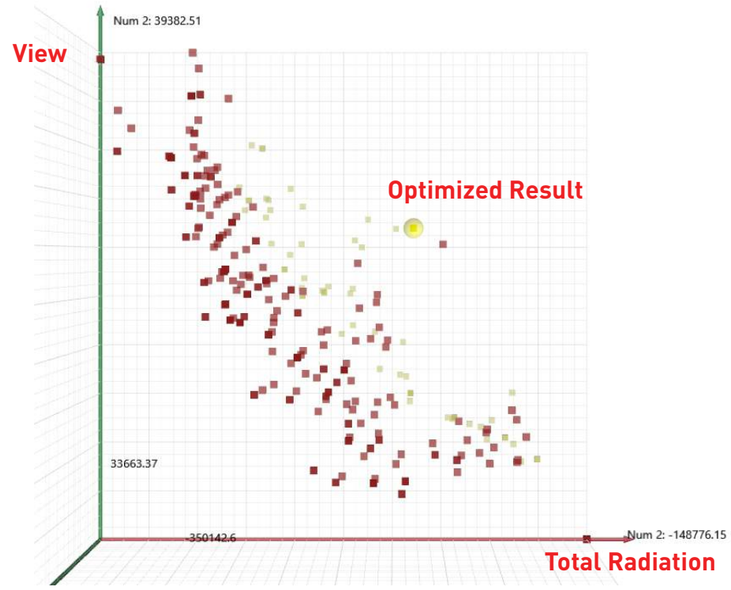
1. Set up the max generation according to circumstances
2. Check the numbers are having individual axis
3. Click ‘Start’ to run the simulation


Num 2: 39382.51
33663.37
3501426
Num 2: -148776.15


Num 2: 39382.51
33663.37
3501426
Num 2: -148776.15





The 'Octopus' is used instead of Galapagos for the multi-objective optimization. Each point in the graph above is a simulation result of complex relationships between the view, daylighting and glare.

The view and the sun direction coincided due to the orientation, which complicated the design process. An incresed opening meant a better view and the daylight but also larger the opening higher the change of the glaring effect due to an increased difference between the brighter and darker parts of the interior.

To resolve this complicated relationship, the genetic algorithm is plugged into the Octopus to evaluate several possible options that satisfy the above mentioned conflicting objectives.

Developing Stage

The Evolutionary solvers are not limited to only data-driven simulations. The major benefit of the Evolutionary solver is that it is super flexible with the condition. It only requires a numerical input parameter and a numerical objective. In a traditional design process, the optimizations were manually done by architectural designers. If the evolutionary solver could be used instead of manual optimization, some undiscovered opportunities could be found.

The Plan and Section studies are one of the most typical manual optimization processes. The designers had to manually draft in search for more seats, units, rooms, or leasing spaces. This process could be revolutionized by the Evolutionary Solvers with practical considerations.

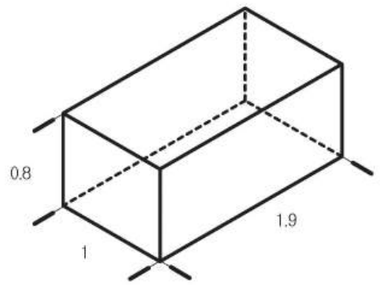
In the Developing Stage application, 'Seat layout optimization process' of "Bucheon Concert Hall project, South Korea, 2018" is used as a reference.

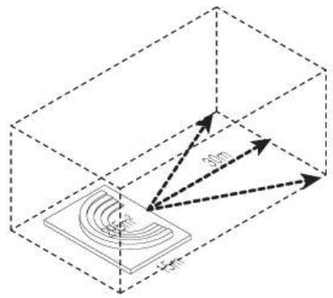
Bucheon Concert Hall project was done in 2018 as a competition project and H-architecture participated in main concert hall design.

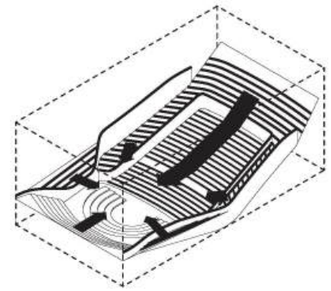
Due to acoustical reasons, the main concert hall had fixed dimensions.
The limited room volume resulted in an issue of the seat arrangement. Since the total number of the seat is directly connected to the total income, it had to be maximized in a limited volume.

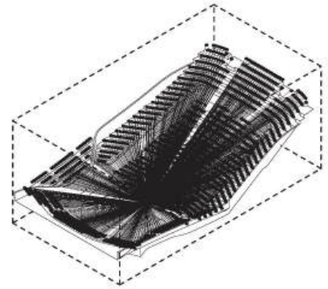
The location and the dimensions of the egress, defined by the building codes, and the visibilities to the stage from each seat are the two major factors had to be considered in the seat layout design.

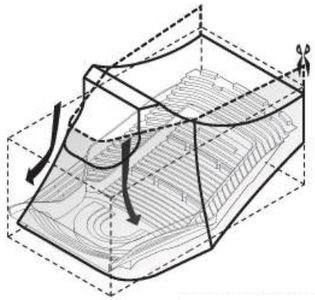
The two parameters-the egress and the view had to be engraved in the Grasshopper coding as a design generator.

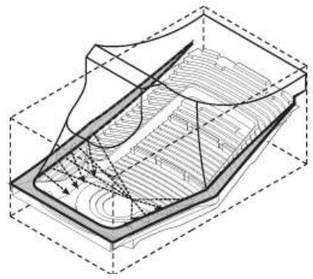

0.8
1.9
1

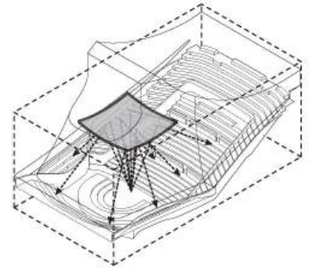


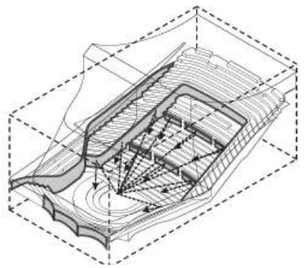


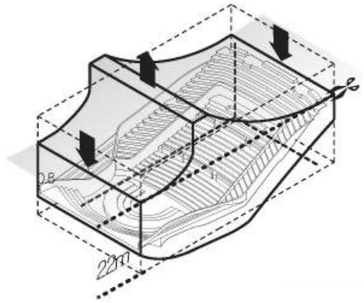


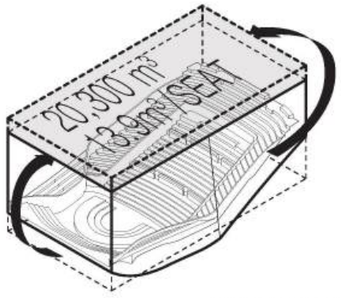


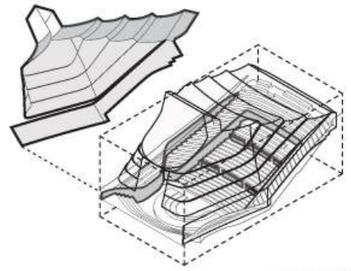


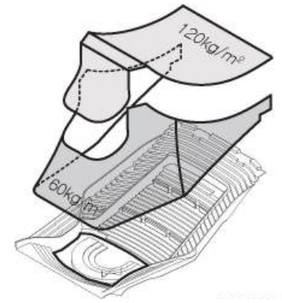


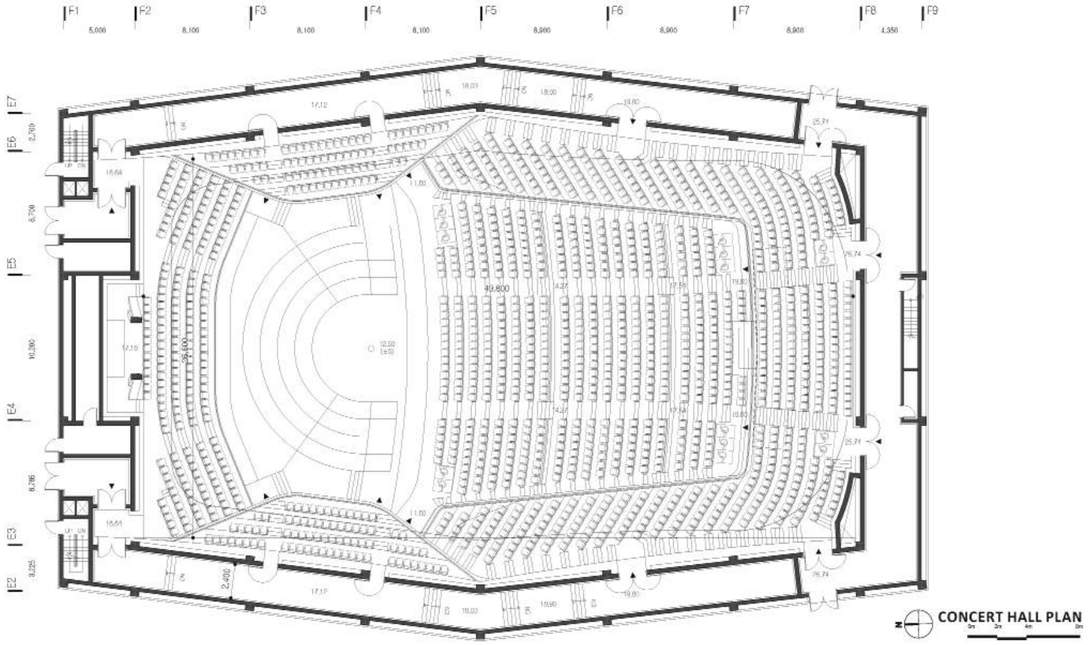


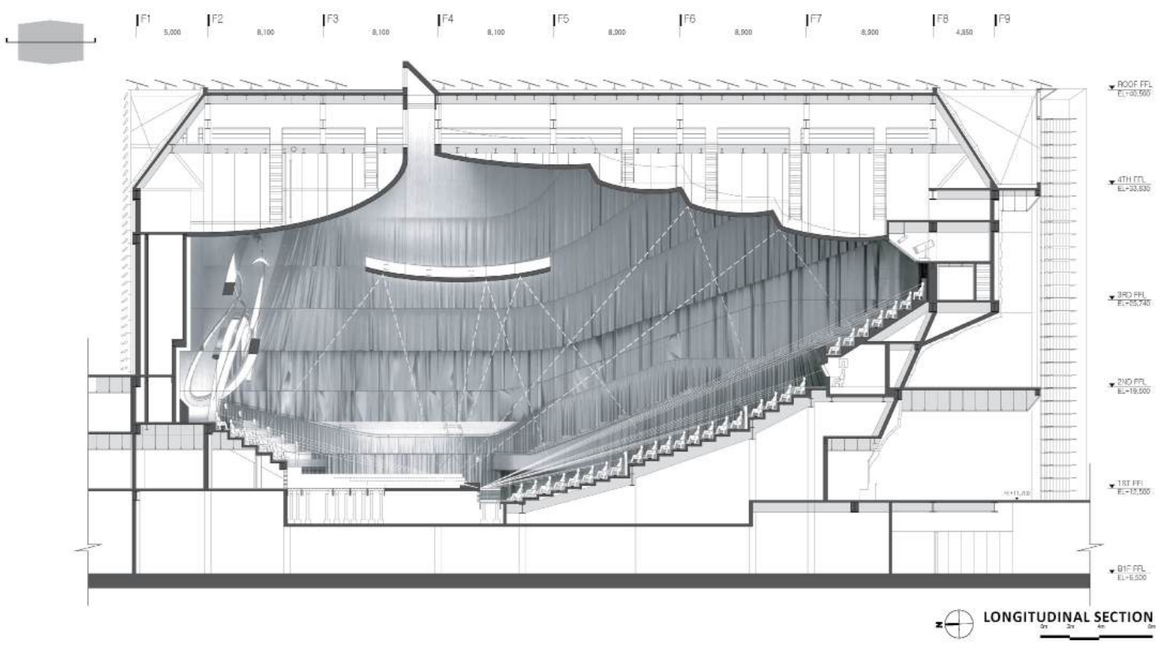


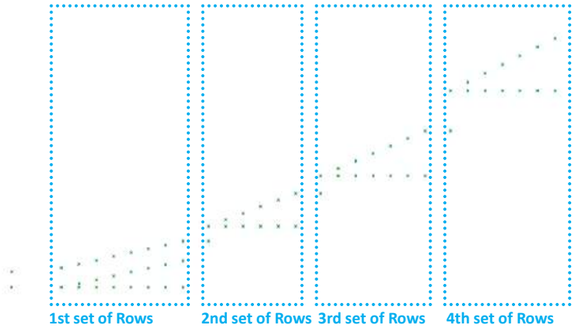

20.300 m²
13.9m²/SEAT

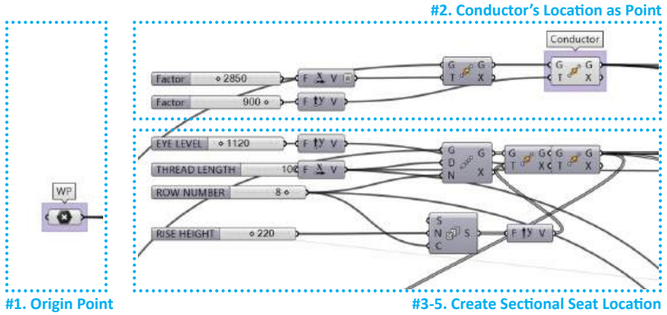



120kg/m²
60kg/m²


F1
F2
F3
F4
F5
F6
F7
F8
F9
5,000
8,100
8,100
8,100
8,000
8,000
8,000
4,350
E7
E6
E5
E4
E3
E2
2,700
6,700
10,300
8,700
9,225
CONCERT HALL PLAN

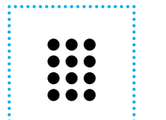

F1
F2
F3
F4
F5
F6
F7
F8
F9
5,000
8,100
8,100
8,100
8,000
8,000
8,000
4,350
ROOF FFL
EL.+30,500
4TH FFL
EL.+20,930
3RD FFL
EL.+25,740
2ND FFL
EL.+19,000
1ST FFL
EL.+12,500
B1F FFL
EL.+5,500
LONGITUDINAL SECTION

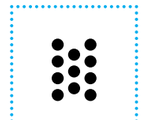


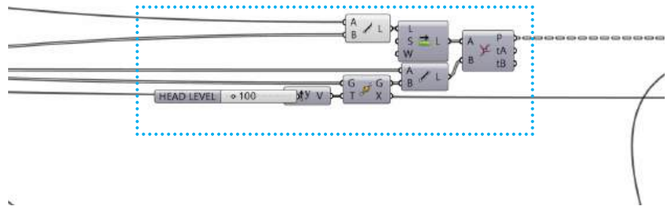


Sectional Seat Layout Creation

1. Create single point in Rhino environment and export it in Grasshopper
2. Move the point to Conductor's location on the stage condition in the project
3. Set up distances between the first series of the seat rows in section
4. Set up rise height of the first series of the seat rows in section
5. Applying the distances and the rise heights to each rows
6. repeat the #3 to #5 in each series of the seat rows in section

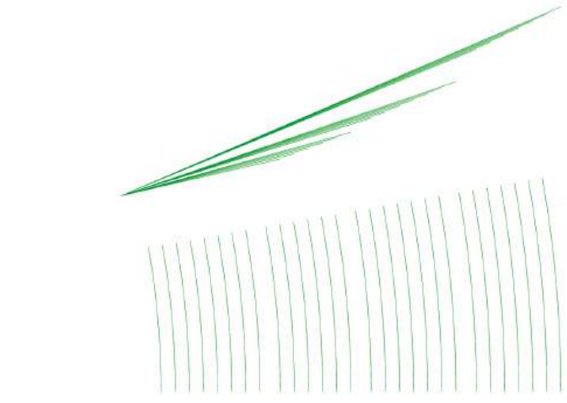

Plan view of the Orthodox Seat layout

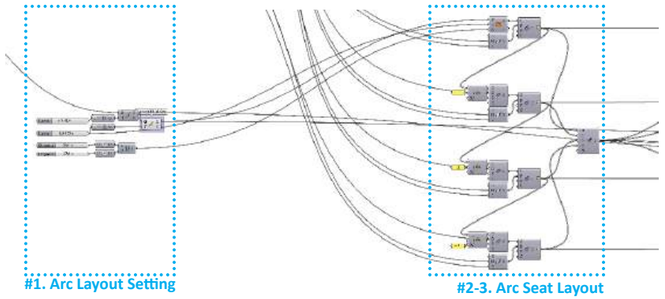

Plan view of the Twiggled Seat layout



Visibility Check

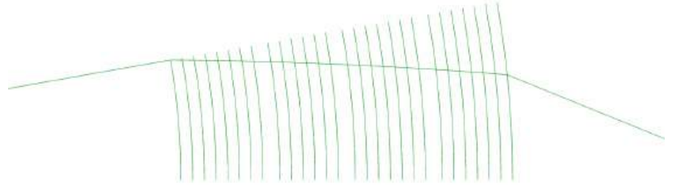
1. Create curve from audiences eye level's point to conductor's location point
2. Create curve from audiences' seat level to headtop
3. Check the corssing curve to determine the twiggling of the seats in plan

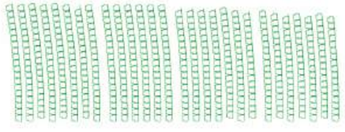


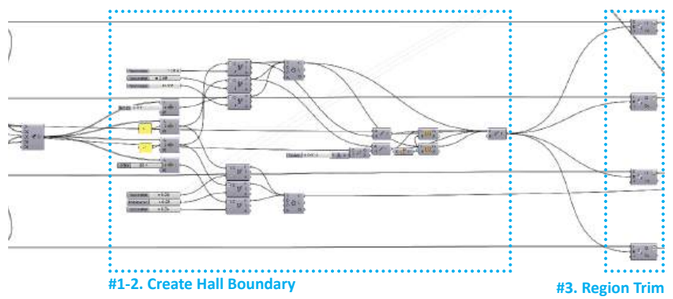


Radial Seat Layout in Plan

1. Set Center point and Radius of the Arc for the seat layout
2. Create Arc with a setting and offset the arc with designated distance
3. Merge all data into one data list via ‘Merge’ component

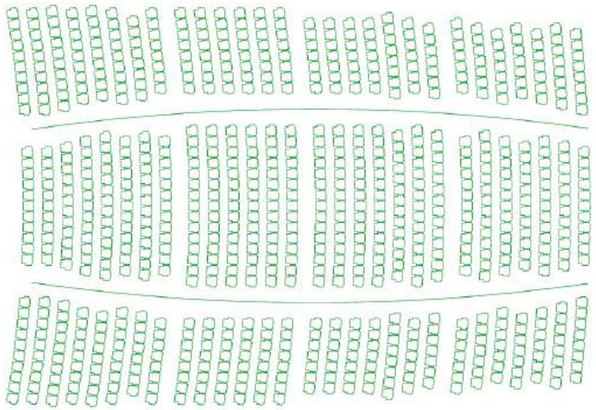


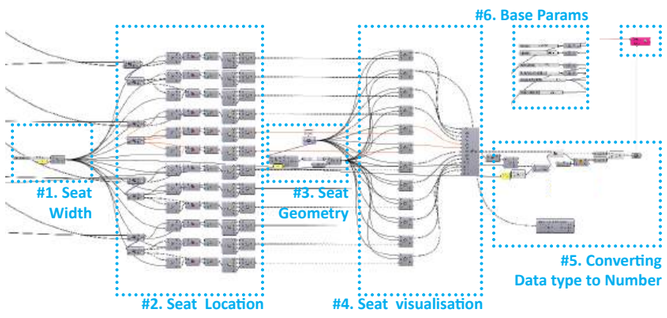




Trimming Radial Layout

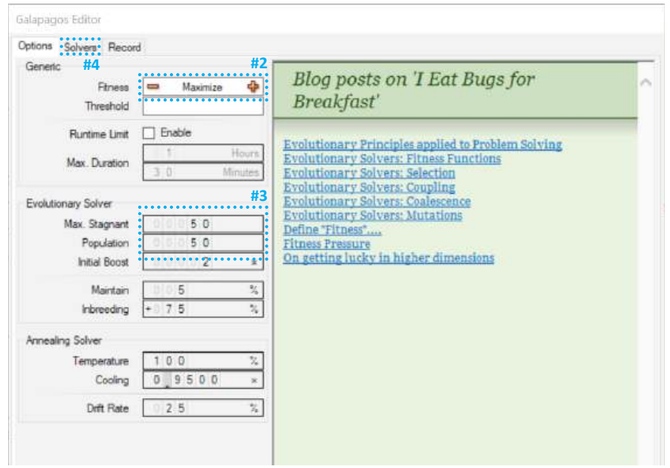
1. Create Outer Boundary of the Arc seat layout according to the Hall Volume
2. Join the Hall boundary curves as a Region to trom
3. Trim the Arcs with a Region via ‘Trim with Region’ component

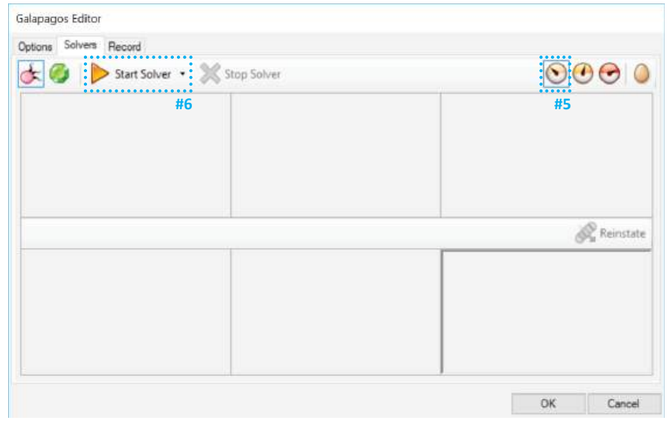




Cull Seat ovelaps with Aisle width

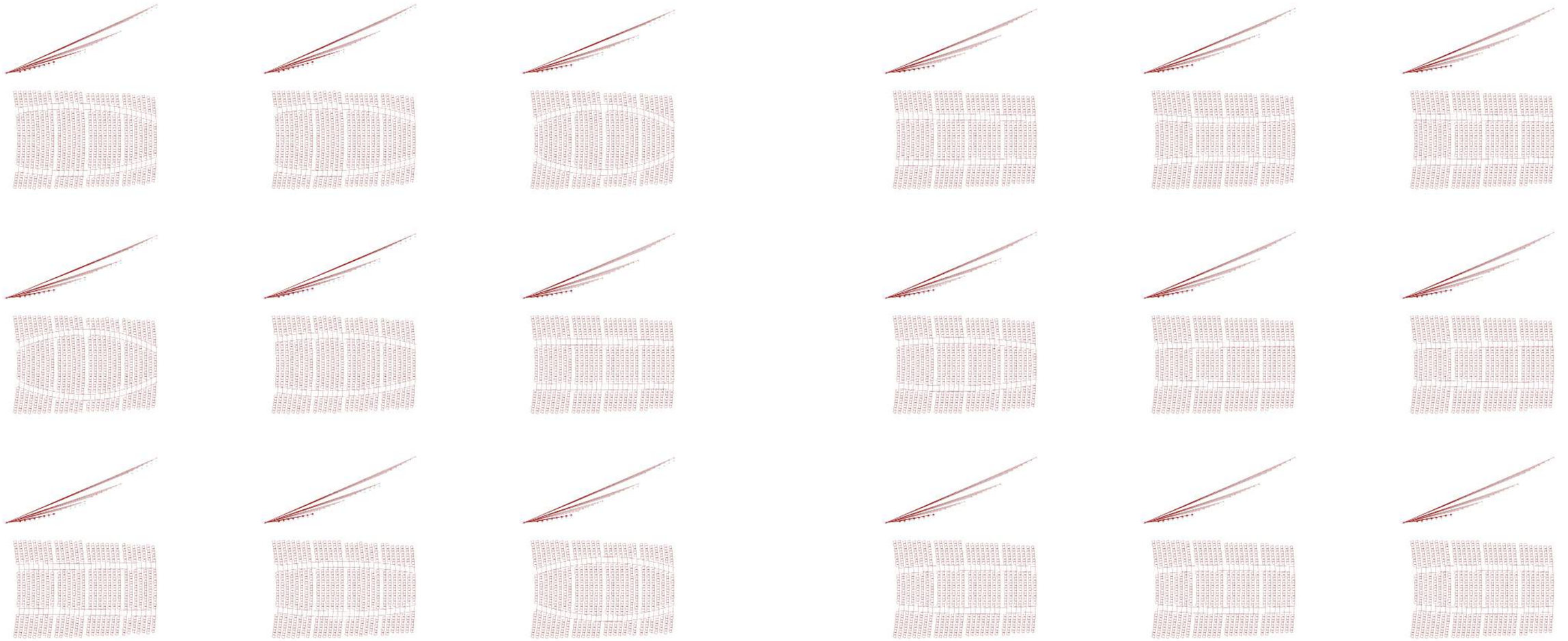
1. Set up the Width of each seats
2. Create the Locational points on the Arcs to create Seat Location
3. Set up Seat geometry by importing the seat shaped curve in Rhino to Grasshopper
4. Place the geometry on each point for the data visualization
5. Converting the length of the item list of the seat to Number and Connect it to Fitness
6. Connect the Base params (Height of the rise / number of rows) to Genome

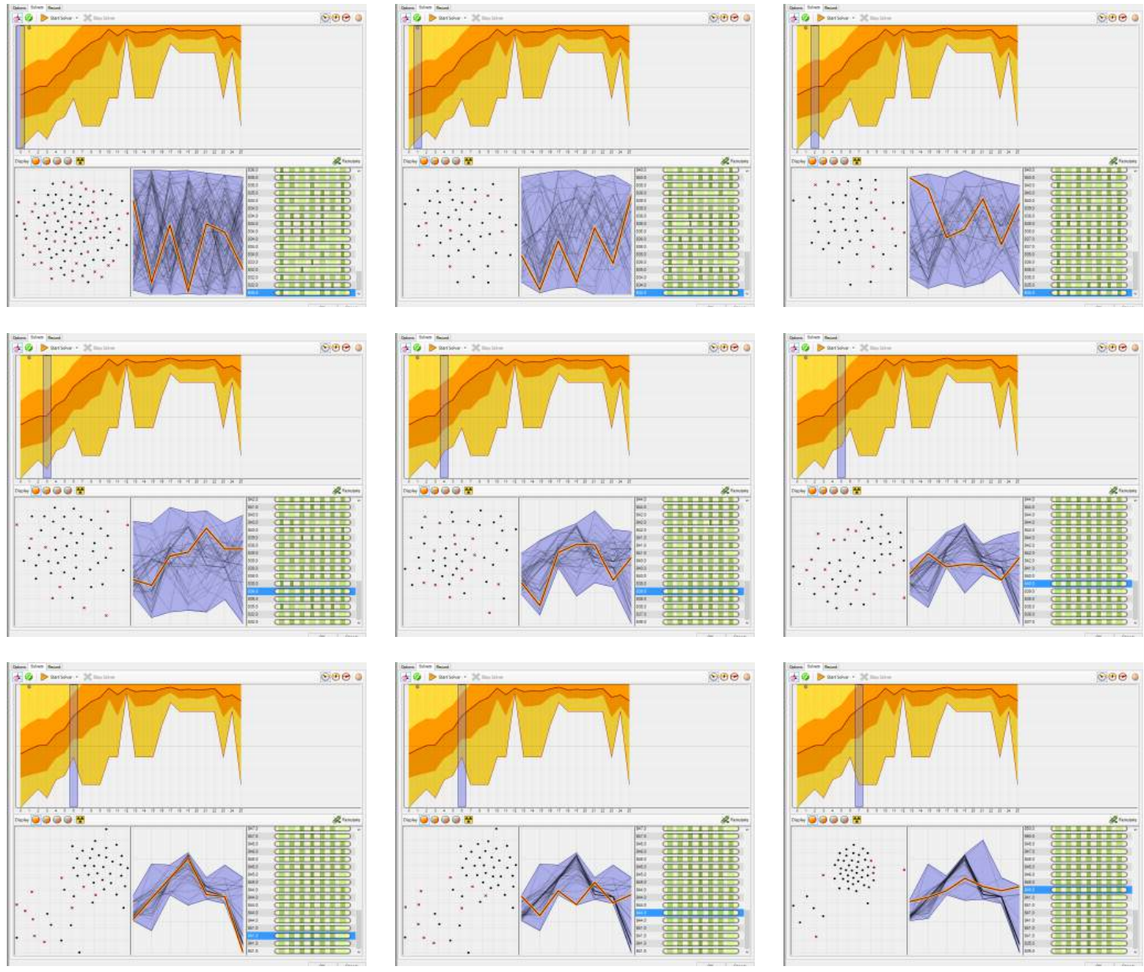


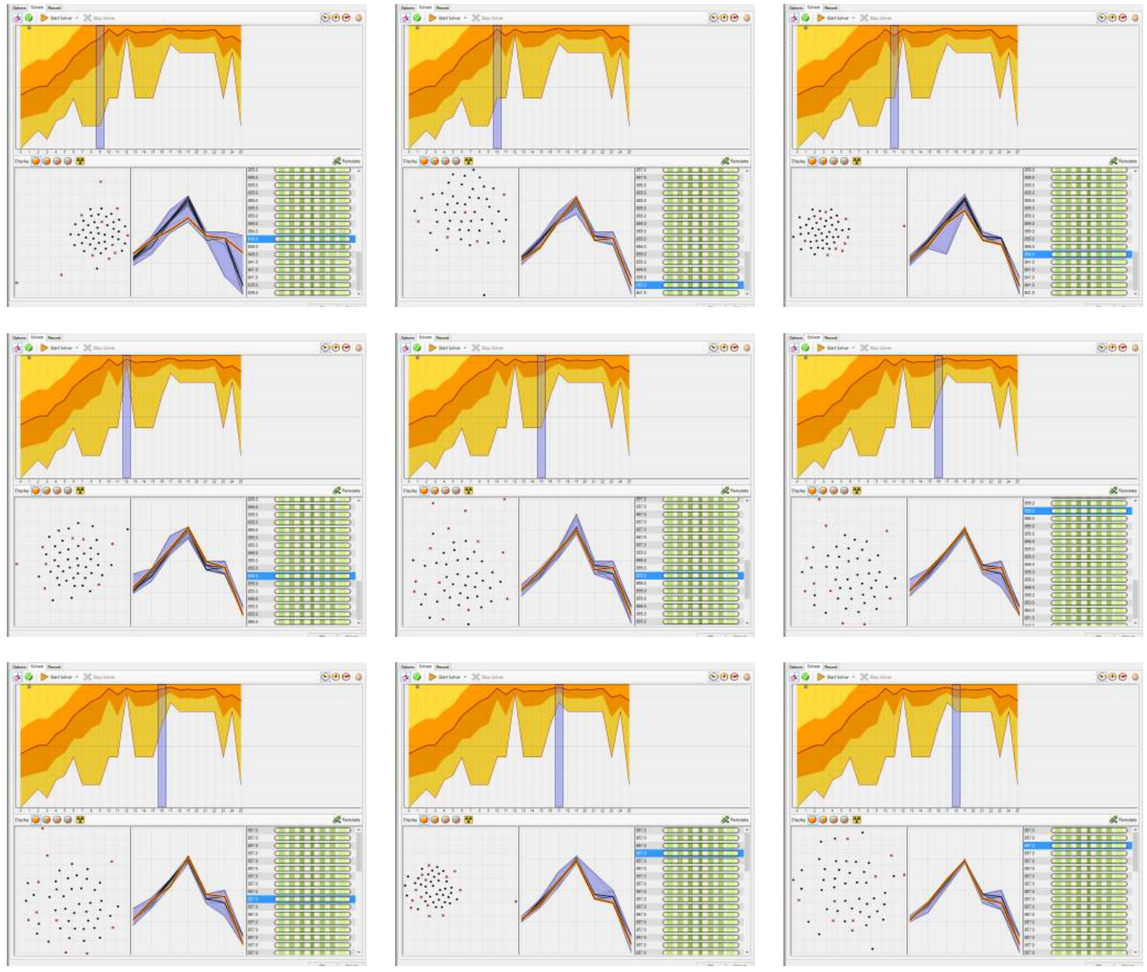


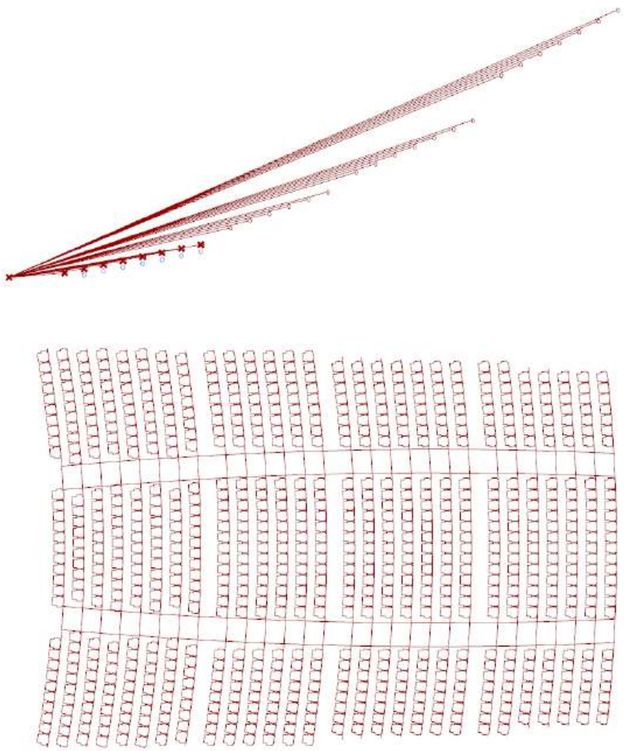
Galapagos Setting & Run

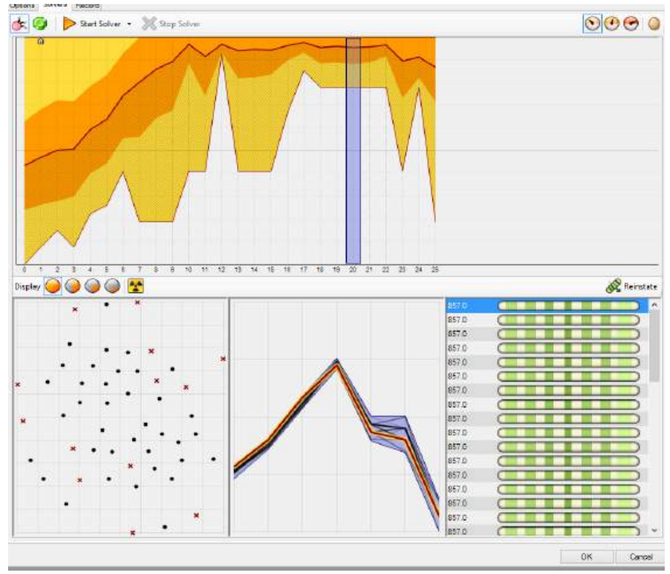
1. Double Click the Galapagos Solver to open up the pop-up window
2. Set 'Fitness' as 'Maximize' to create more venues in the space
3. Set 'Max. Stagnant' and 'Population' to 50 for Precise simulation
4. Click 'Solver' tab for the Simulation
5. Click the left clock icon to visualize overall simluation in Rhino viewport
6. Start Solver







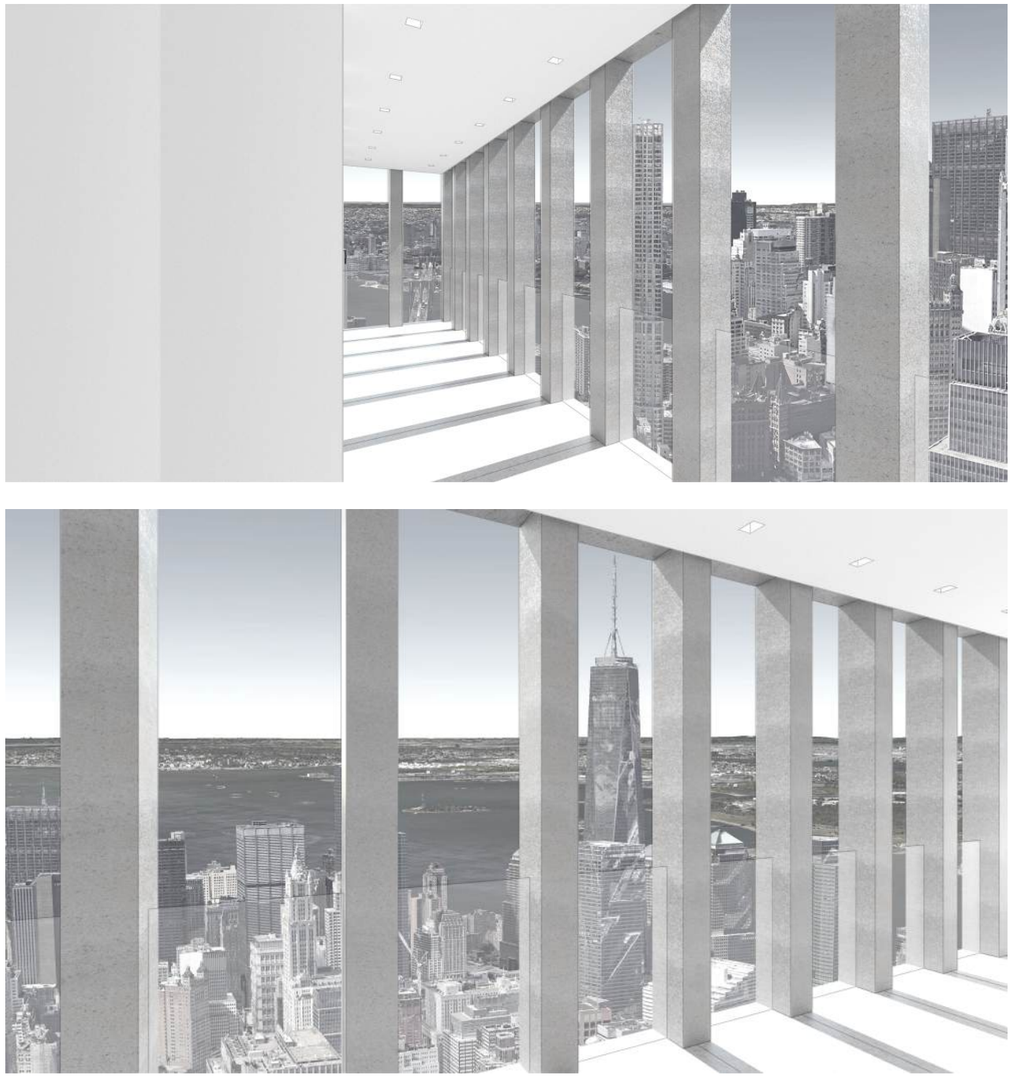




The major issue of creating the seat layout was related to the building codes about the egress and visual interruption to the stage from the seats in front. The major input parameter was the shape of the inner corridor and by adjusting it, the number of the seats was increased greatly, which meant the maximized income. Another input parameter was the adjustment of the seats according to the visual interruptions. The audiences seating in front blocks the view of the stage to the audience seating behind. Thus, the seat layout of the following row was coded to shift to provide avoid the obstruction. The coding was designed to simulate and evaluate these parameters, which affected the final result of the seat layout.

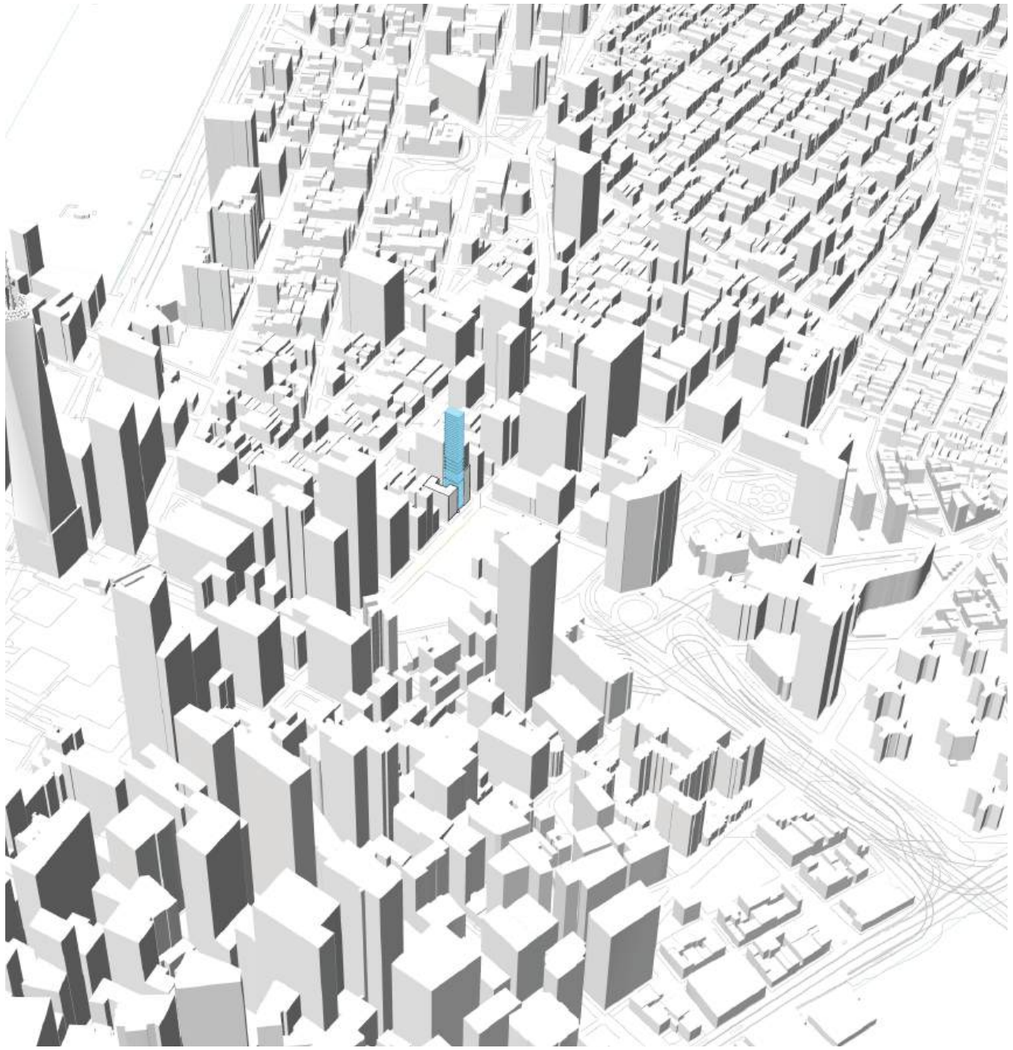
Post Design Stage

Building elements such as the sun shading devices -louver, fin, etc- are usually considered during the later design process due to the fact that the entirety of the building is subject to stricter parameters such as the Building code, Economical benefits or Client's request. When the overall building form, floor plans, and sections are fixed, the building elements are the only way to control the radiation, shading, irradiance, daylighting or glaring effects. The post design elements can be designed through the environmental simulation and the resulting information. However, the fine-tuning of the elements such as the dimension of the depth, width, or shape cannot be perfected from just the simulation alone. The evolutionary solver can be the fine tuning method for these elements by constantly calculating and evaluating the design results.

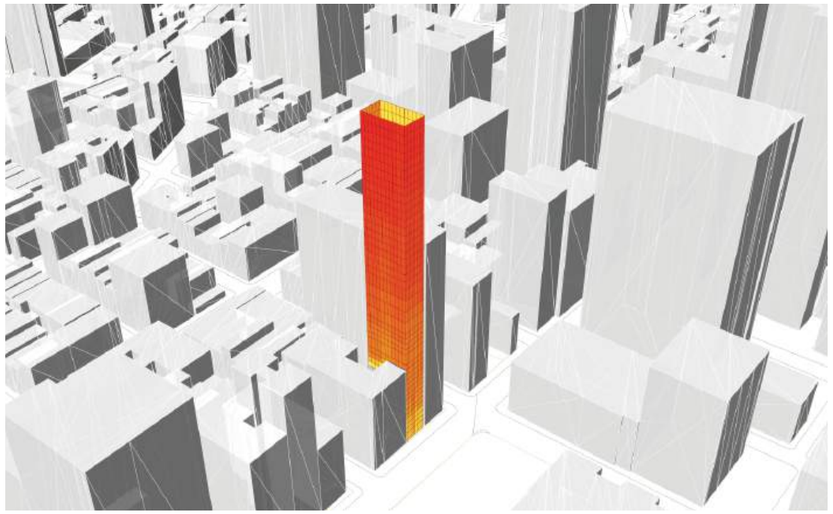


267 Broadway is a skyscraper that H-architecture proposed in 2018. The shape of the tower is decided by strict building code and other social considerations. The elevation of the tower was designed as a grid-based louver and fin combination which is the classic Manhattan skyscraper style.

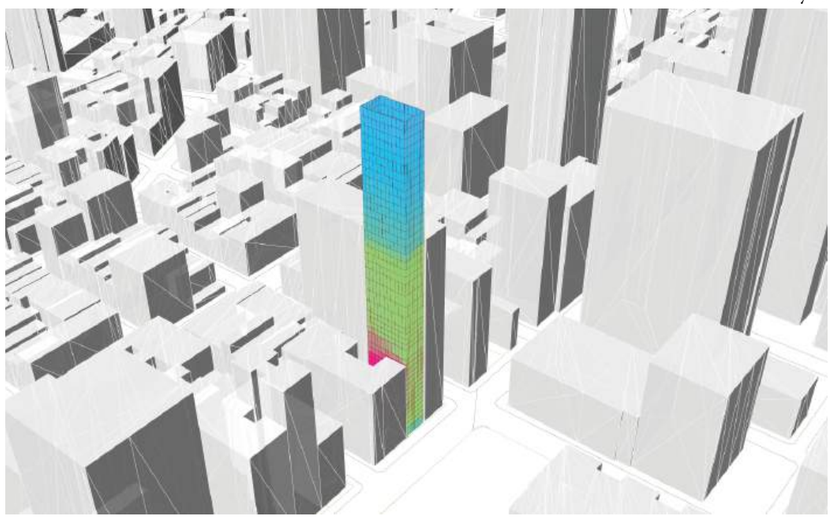
The Environmental analysis of this elevation focused on two controversial aspects of the fin and louver which is the obstruction of the view and the reduction of the total radiation. To achieve the two conflicting objectives simultaneously, the multi-objective optimization was applied to the sun shading devices.



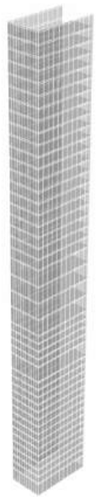
Context

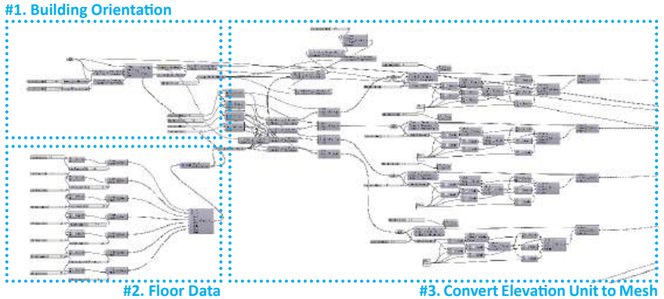


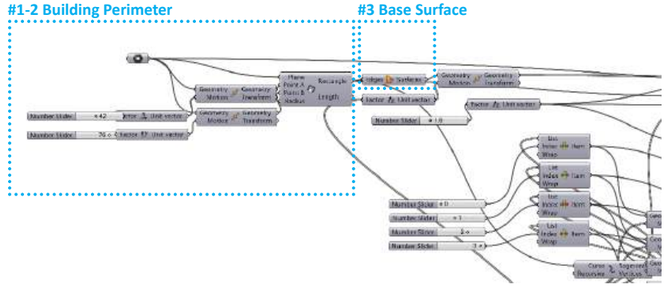
Radiation Analysis



View Analysis





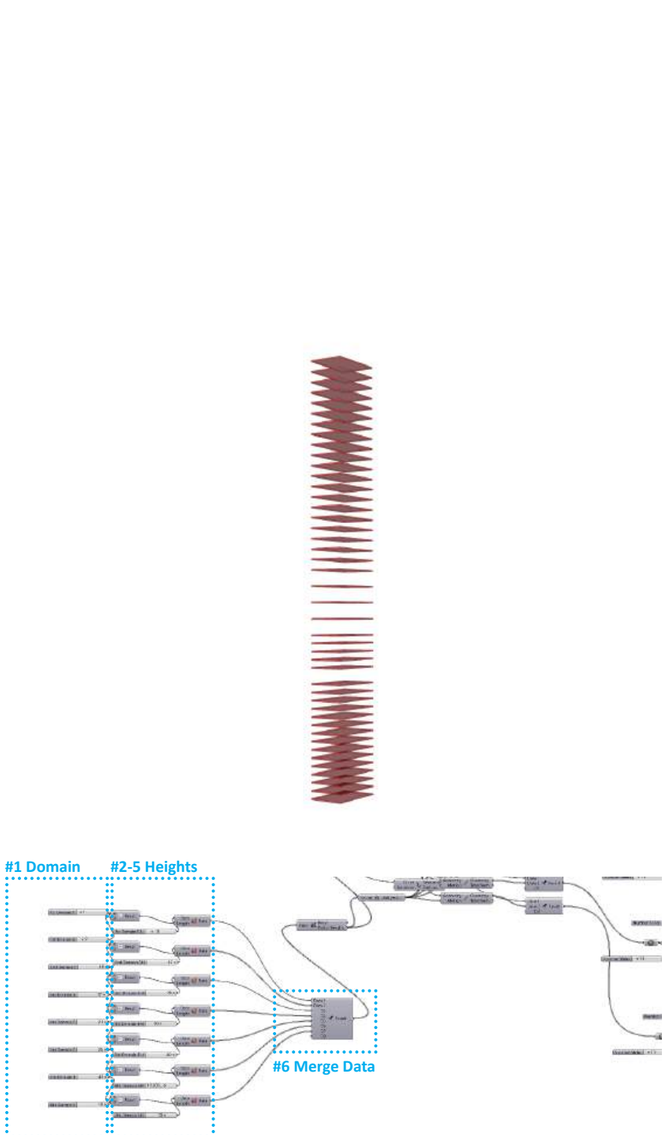


Create Base Modeling in Grasshopper

1. Set the orientation of the buliding (XY plane to EWSN plane)
2. Create building modeling with Floors, Heights and Elevation modules
3. Extract the Building skin with proper Mesh conversion

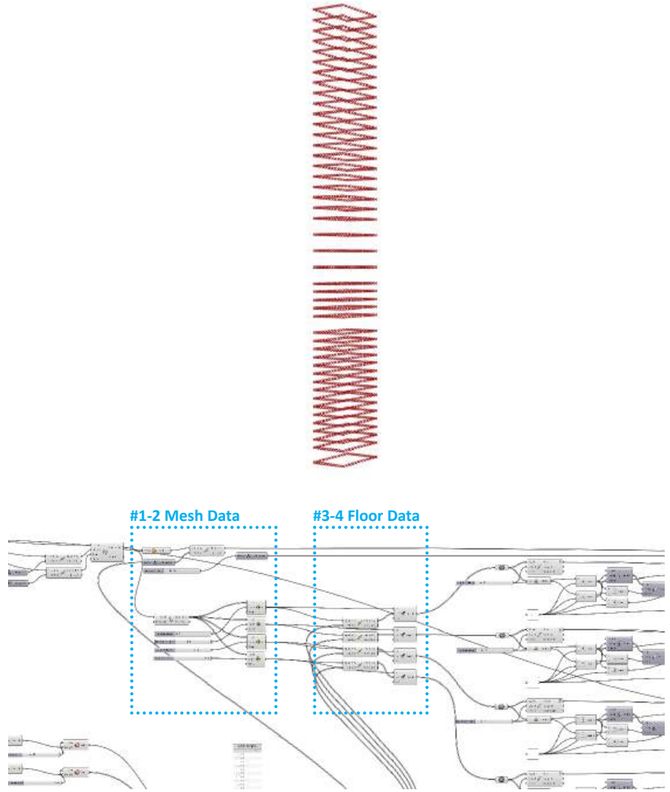
Building Orientation

1. Create a Rectangle using “Rectangle 2Pt” Component.
2. Connect diagonal points to Point A and Point B of “Rectangle 2Pt.”
2. Connect the Rectangle output to Edges input of “Boundary Surfaces.”



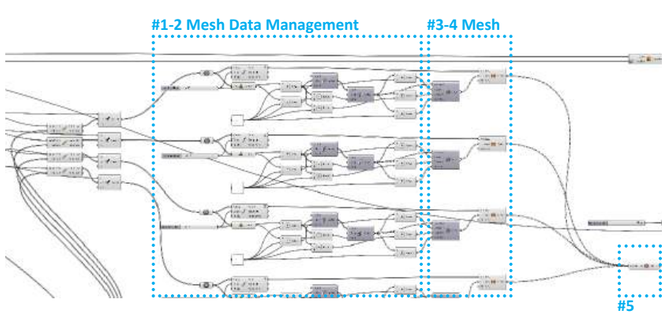
Floor Data

1. Create “Number sliders” for each height domain.
2. Subtract the lower domain from the higher domain.
3. Connect the result into the Length input of “Repeat Data” components.
4. Create “Number sliders” with the floor-to-floor heights at each domain.
5. Connect the step 4 results to Data inputs of “Repeat Data” Components.
6. Connect the “Repeat Data” outputs into “Merge” Component.



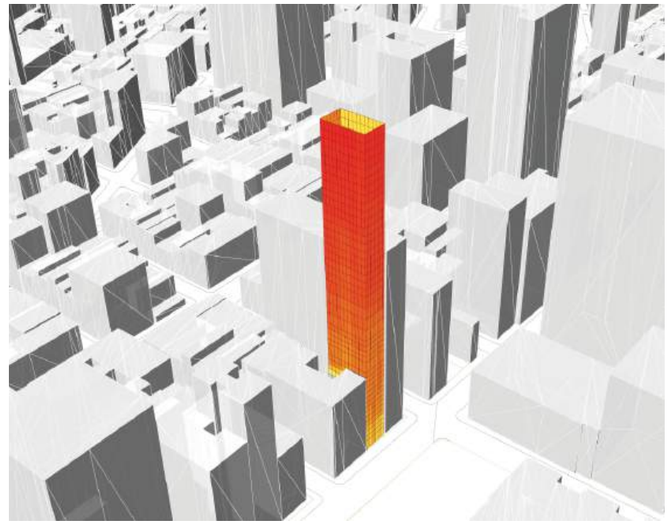
Convert Elevation Unit to Mesh-1

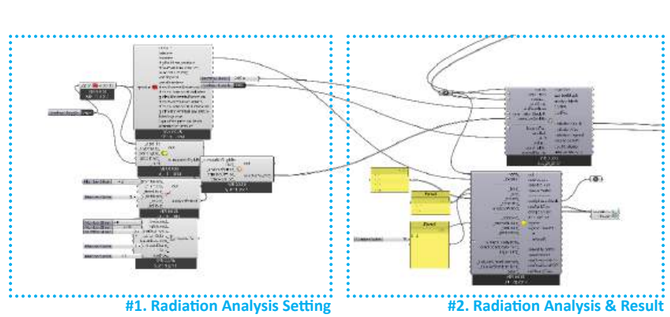
1. Explode the Rectangle in Building Orientation
2. List the explosion results using “List item.”
3. Move the lists using the results from Floor Data
4. Merge the moved lists and the original lists.



Convert Elevation Unit to Mesh-2

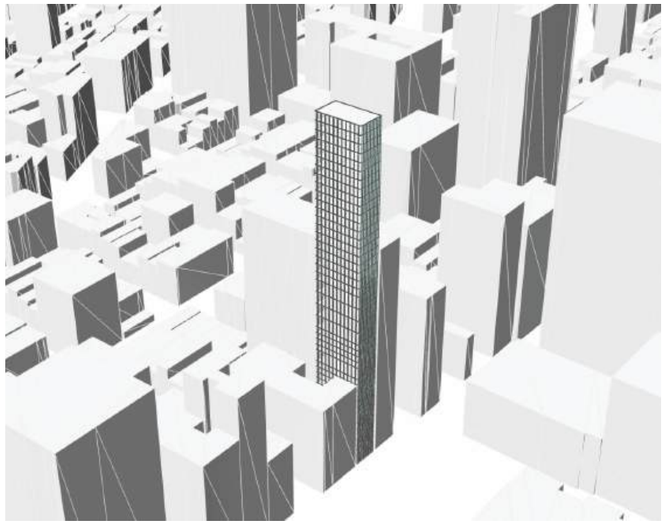
1. Divide the curves created.
2. Create four appropriate series (X, X+1, X+Y, X+Y+1).
3. Create Mesh faces using “Mesh Quad” and the series generated.
4. Construct Mesh.
5. Join the Meshes created.

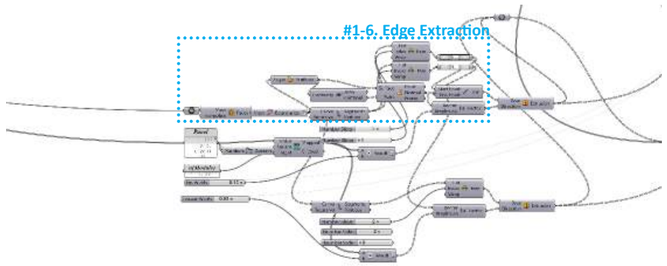




The 1st Radiation Analysis

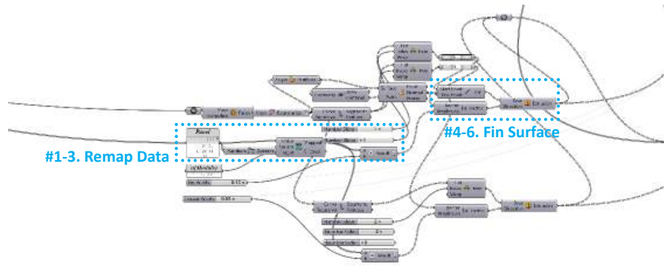
1. Create Radiation Analysis setting
2. Run the analysis and extract the Result





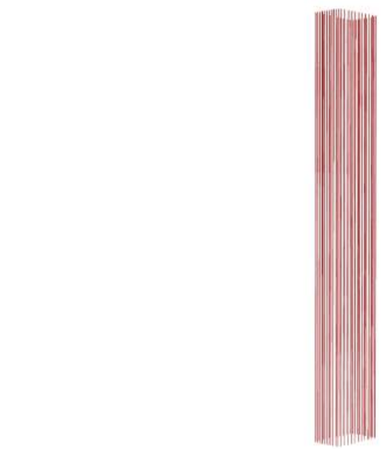
Create Shading Devices - Fin

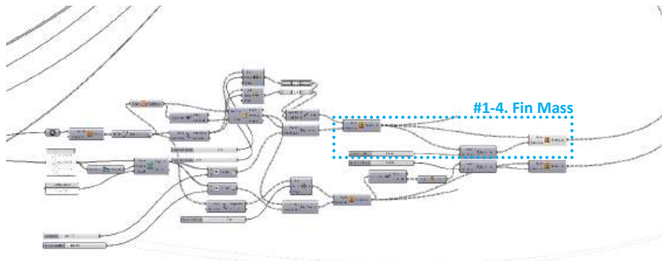
1. Extract the mesh edges using the “Mesh Explode” and “Face Boundaries.”
2. Extract the bottom and top edges of each mesh face using “List Item.”
3. Select the center points of the edges by setting “Point on Curve” to “0.50.”
4. Construct a Line between the center points.
5. Extract the normal vectors of mesh faces using “Evaluate Surface.”
6. Connect normal output to vector input of “Amplitude.”



Create Shading Devices - Fin

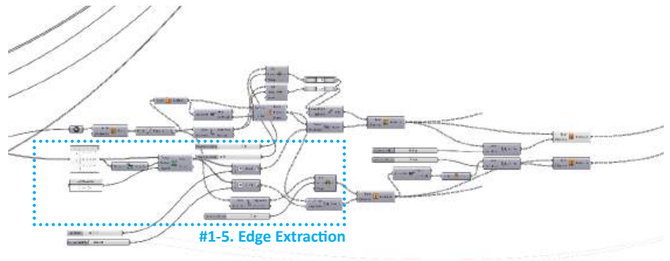
1. Remap the radiation simulation data to numbers between 1 and 20.
2. Connect the remapped data to A input of “Multiplication.”
3. Create a number slider between 0.02 and 0.10, connect to B input of “Multiplication.”
4. “Graft” the multiplication results and connect to “Amplitude” created earlier.
5. Connect the resulting vector to “Extrusion.”
6. Connect the lines created earlier to “Extrusion.”





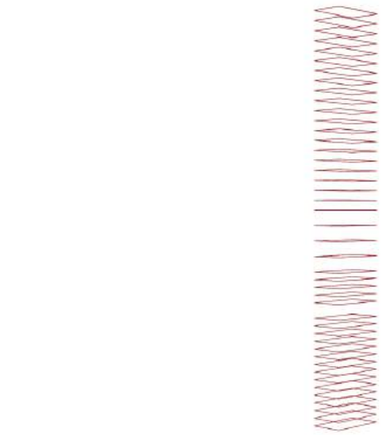
Create Shading Devices - Fin

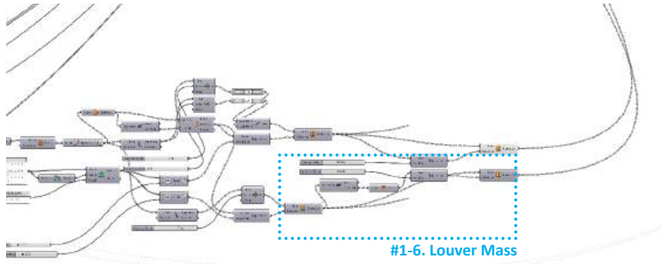
1. Connect the extrusion created earlier to “Amplitude.”
2. Connect a number slider to “Amplitude.”
3. Connect the extrusion created earlier to another “Extrusion.”
4. Connect amplitude to the new “Extrusion.”



Create Shading Devices - Louver

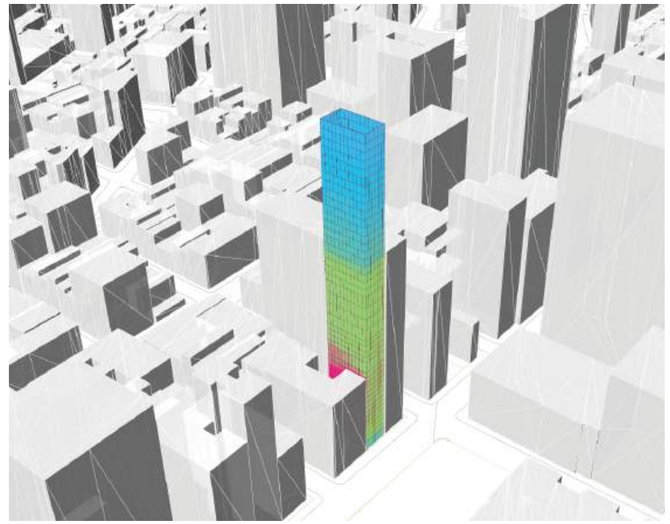
1. Connect the remapped radiation data to A input of “Multiplication.”
2. Create a number slider between 0.02 and 0.10, connect to B input of “Multiplication.”
3. “Graft” the multiplication results and connect to “Amplitude” created earlier.
4. Connect the normal vectors of mesh faces to “Amplitude.”
5. Extract the top edges of each mesh face using “List Item.”

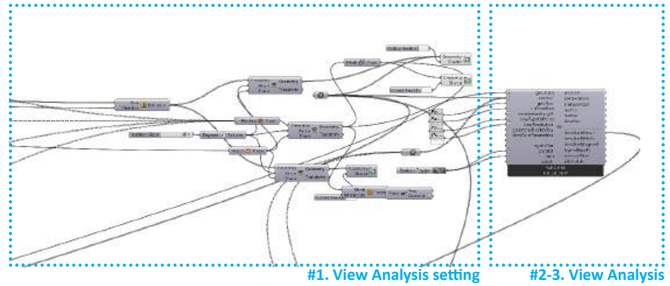




Create Shading Devices - Louver

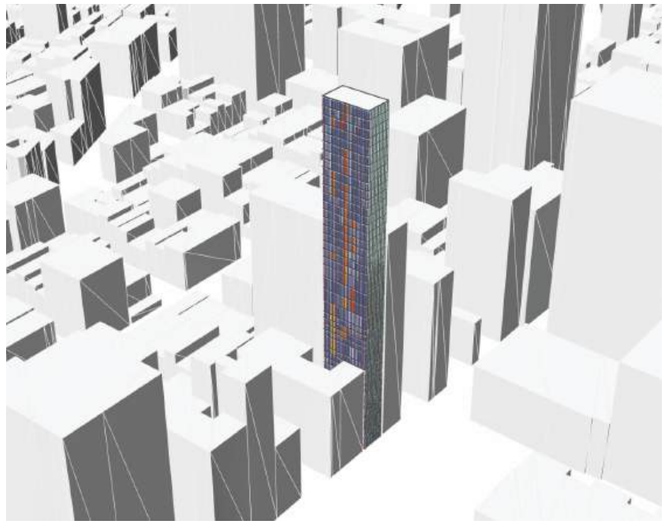
1. Connect the extracted edges and vectors to “Extrude.”
2. Extract the centroids of the extrusion using “Area.”
3. Connect centroids to “XY Plane.”
4. Connect XY Plane and a number slider to “Amplitude.”
5. Connect the extrusion to another “Extrusion.”
6. Connect the amplitude to the new “Extrusion.”

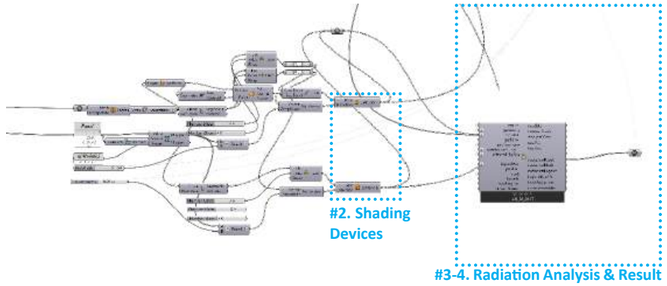




View Analysis with Fin & Louver (Maximizing Value)

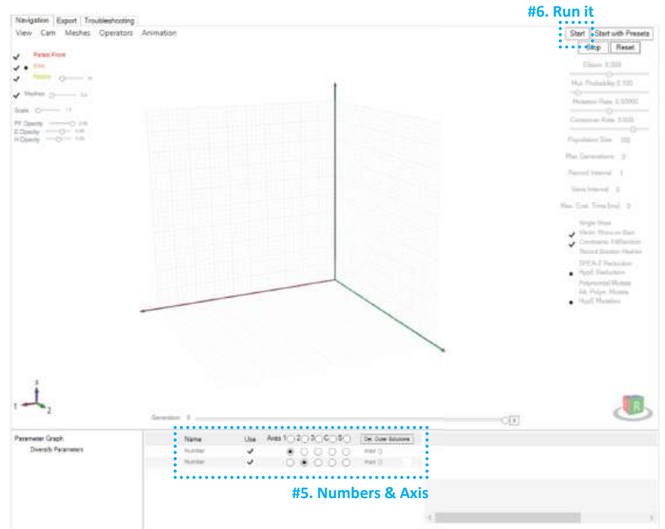
1. Create View Analysis setting with Shading devices as context
2. Run the Analysis
3. Extract overall view percentage as a numeric data from ‘viewStudyResult’ of simulation

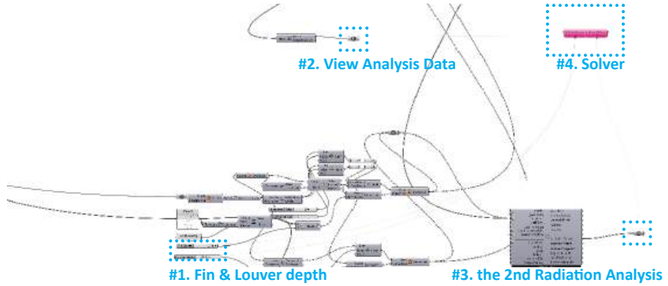




The 2nd Radiation Analysis (Minimizing Value)

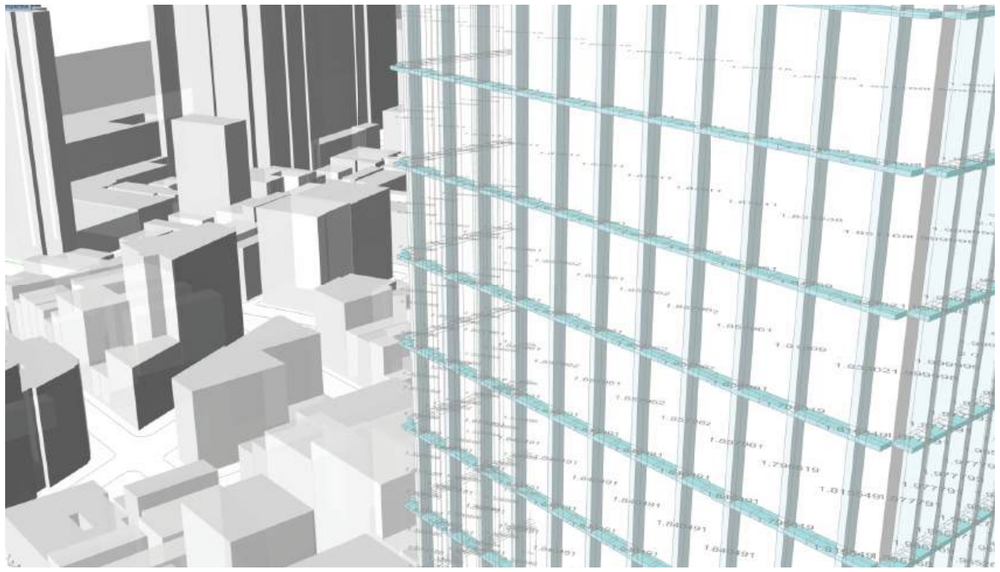
1. Share the 1st Radiation Analysis setting for the 2nd simulation
2. Plug the Fin&Louver as additional contexts of the radiation simulation
3. Run the analysis
3. Convert the overall radiation value to numeric data via ‘Number’ component



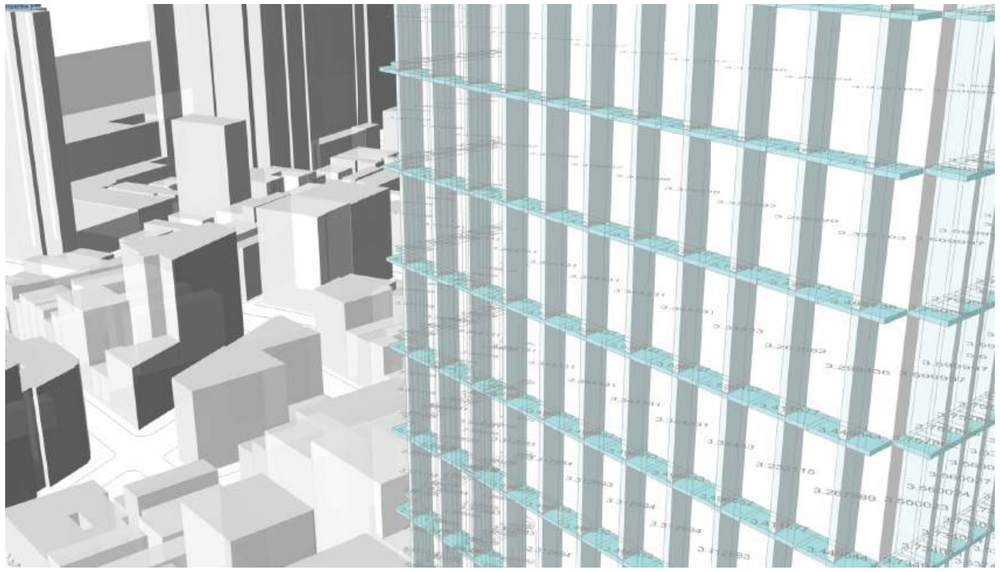


View Analysis with Fin & Louver (Maximizing Value)

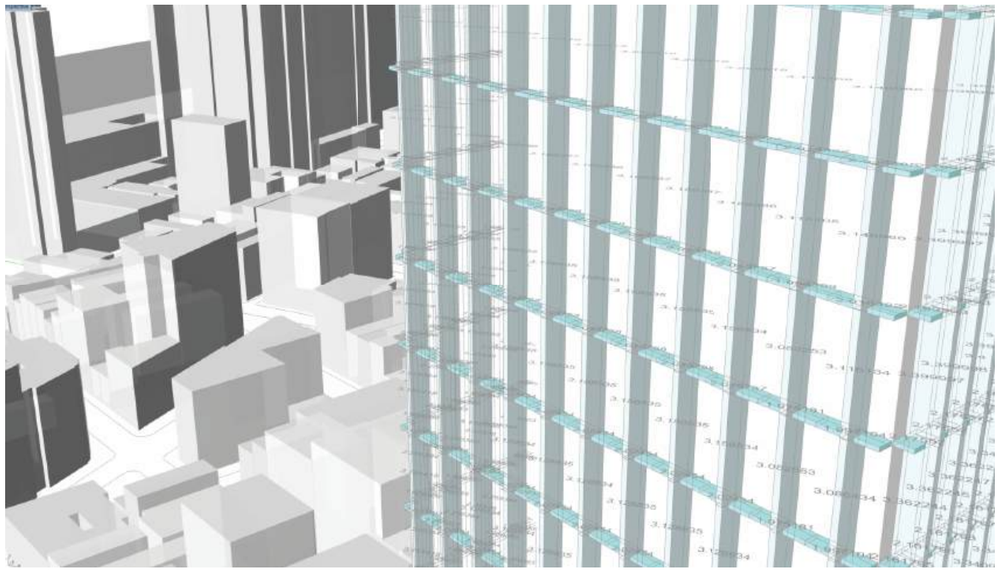
1. Connect the isolated Fin & Louver params to the ‘Genome’ of the solver
2. Connect converted numeric data of View analysis to ‘Octopus’ of the solver
3. Connect converted numeric data of Radiation analysis to ‘Octopus’ of the solver
4. Double-click the solver to open up the pop-up window
5. Check the numbers to have individual Axis
6. Run the solver by clicking ‘Start’ button



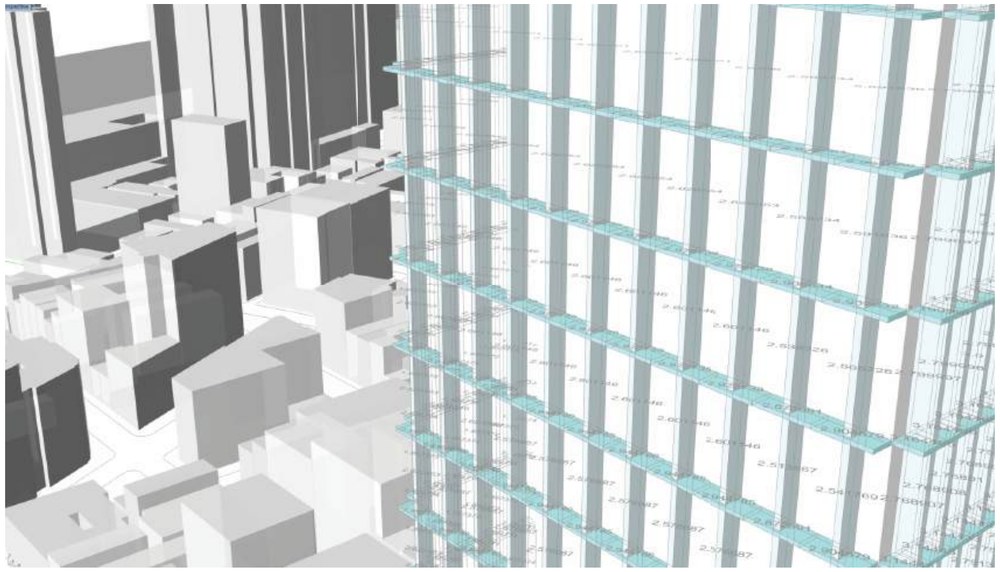
Simulation Option #1



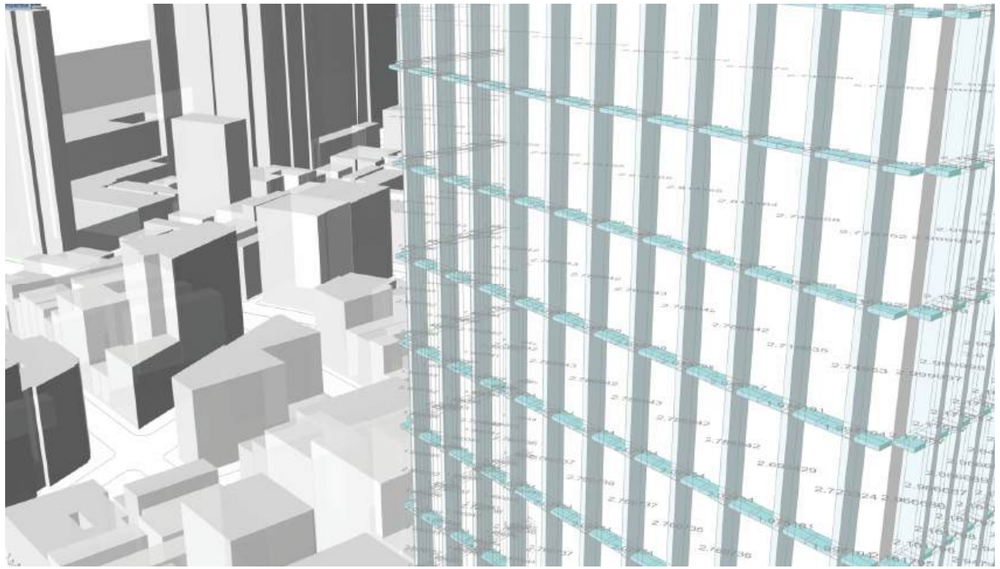
Simulation Option #2



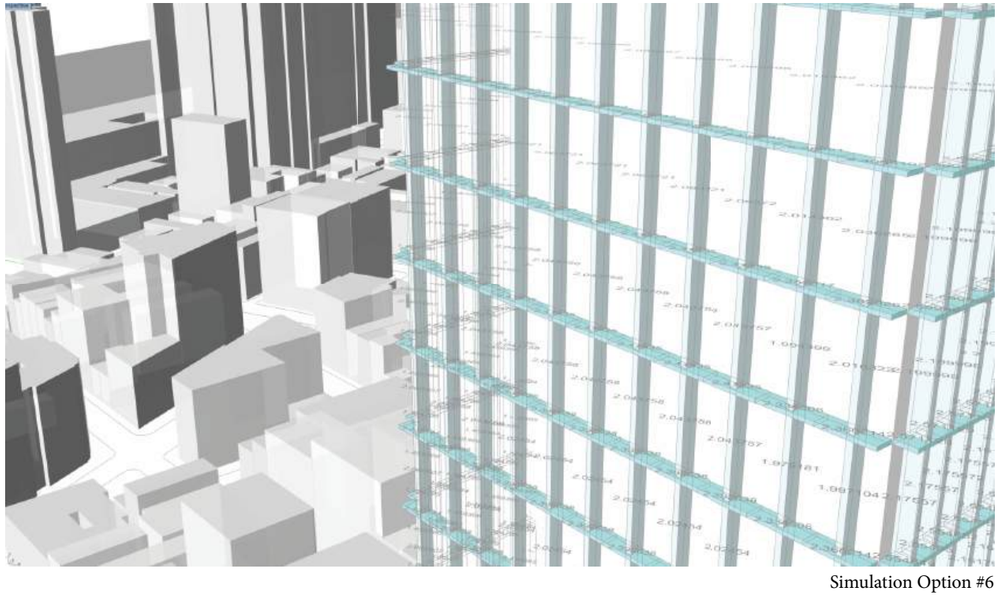
Simulation Option #3



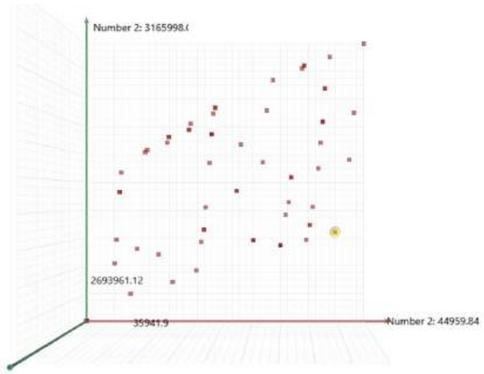
Simulation Option #4



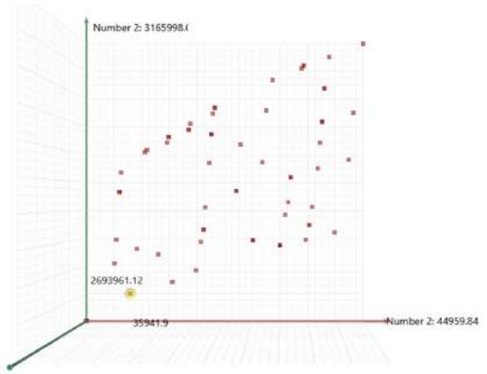
Simulation Option #5



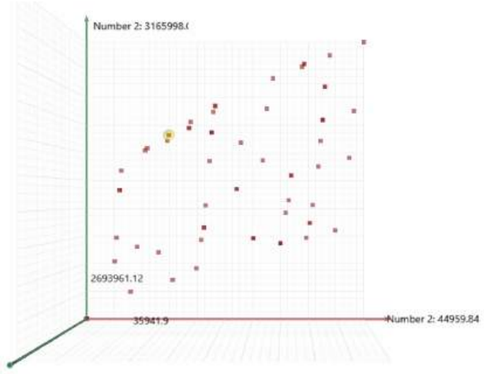
Simulation Option #6



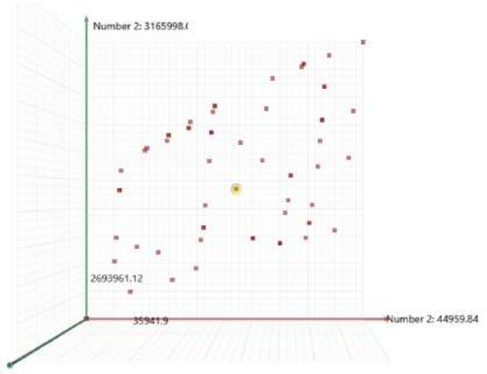
Option #1



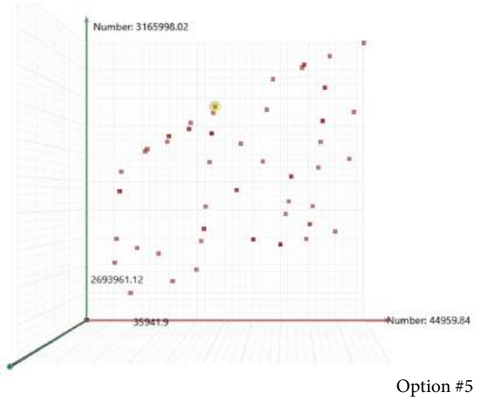
Option #2



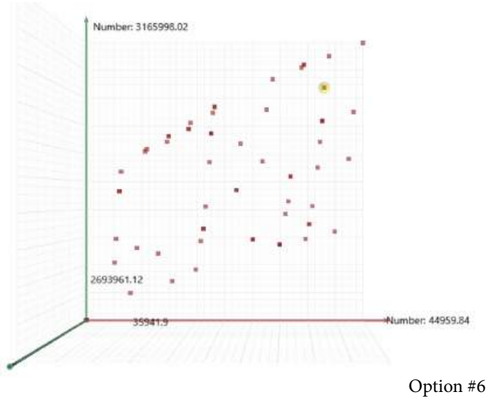
Option #3



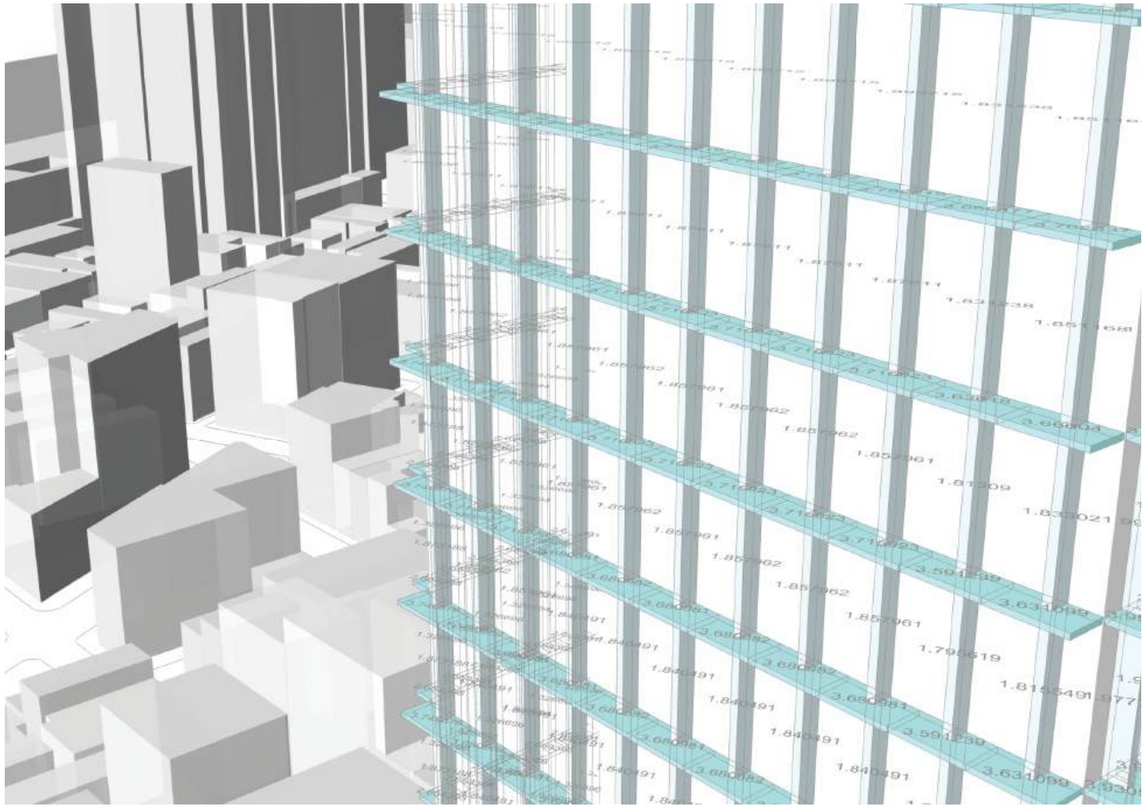
Option #4



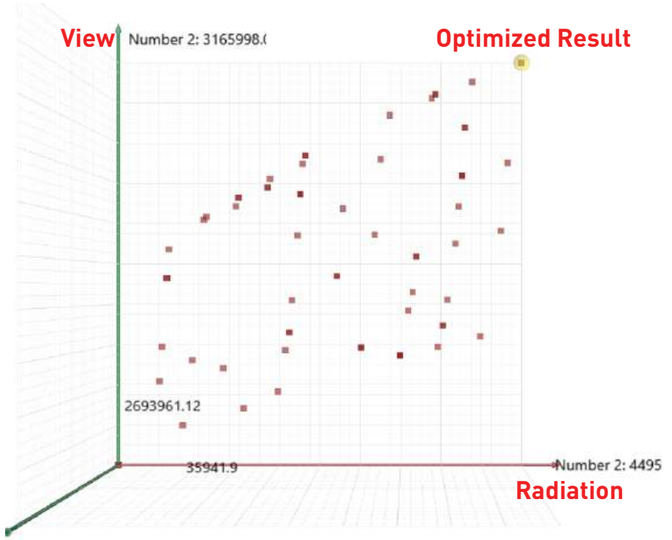
Option #5



Option #6

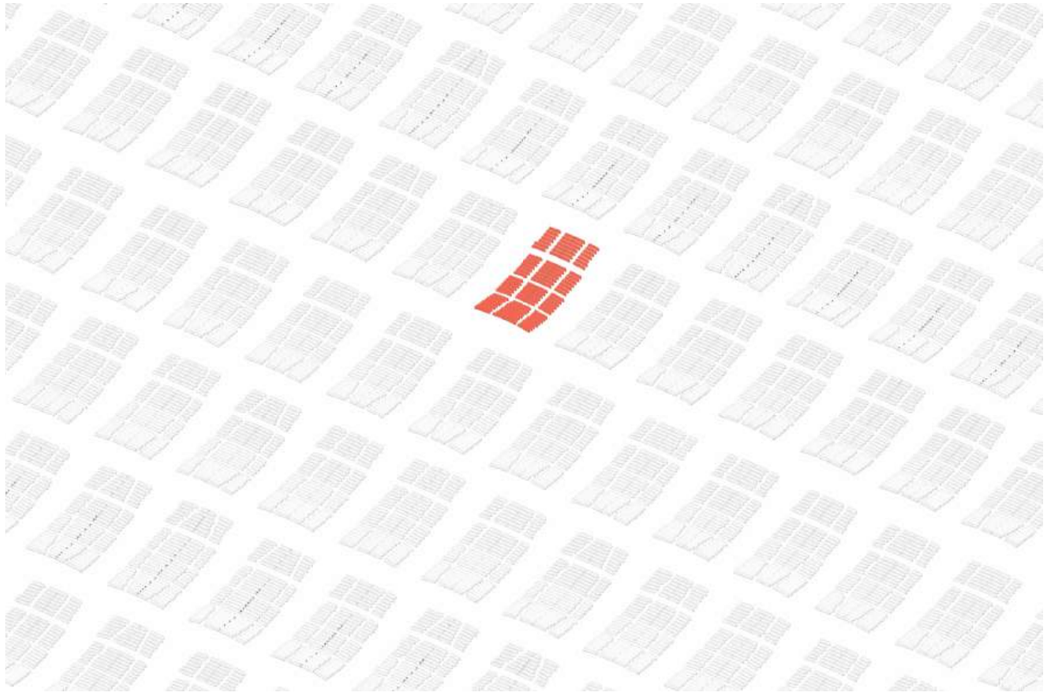


Optimized Solution



The optimization process of the fin and louver forms was based on the view and the radiation. Ideally, the building would have the maximum view with minimum radiation. However, due to other set conditions of the project, the view and the radiation became conflicting parameters. Thus, when the view was maximized the radiation was also maximized instead of the ideal condition of being minimized. The graph above reflects the conflicting situation and the design has to be a compromise between the view and radiation. The marked genomes above is the superior geometry.

Result & Limitation



Bucheon Concert Hall Optimized Center Venue

Result & Limitation

Evolutionary computing programs are extremely helpful for making design decisions or, at least, useful for narrowing down the design options when it is used with proper data flow algorithms and logical foundation of the practical considerations. The understanding of the design process is crucial for a designer to create a specific code response related to design parameters that are usually rooted in building code or general design criteria. The list below is the fundamental understanding of the Evolutionary solver that needs to be followed for future applications.

1. Not a stand-alone program

The Evolutionary Solver requires additional coding to generate diverse design options. Thus, the designer who wants to utilize the Evolutionary Solver must learn the skills to create additional data flow structure in the Grasshopper environment.

2. Requires only numerical value as objectives

The Evolutionary Computing only accepts numerical data as the simulation objectives. Thus, all of the objectives must be converted into quantitative data with proper units. For example, the Lx is used for simulating the daylighting, number of venues, or hours of shadow in range.

3. Application of the program in overall design process is limited

The Evolutionary Solver doesn't need to be involved in the entire design process. As mentioned above, the Evolutionary Solver can only accept measurable data, therefore, understanding when to use the Evolutionary Solver during the design process is crucial for the effective use of the Evolutionary Solver.

4. Simulation time vary depends on base algorithms

The duration of the simulation varies greatly on the complexity of the base algorithms. Some complex algorithms may take unexpectedly long hours. Thus, the designer must be able to compromise between the accuracy of the result and the duration it takes for the effective use of the Evolutionary Solver.

Conclusion

Design Beyond Human Cognition

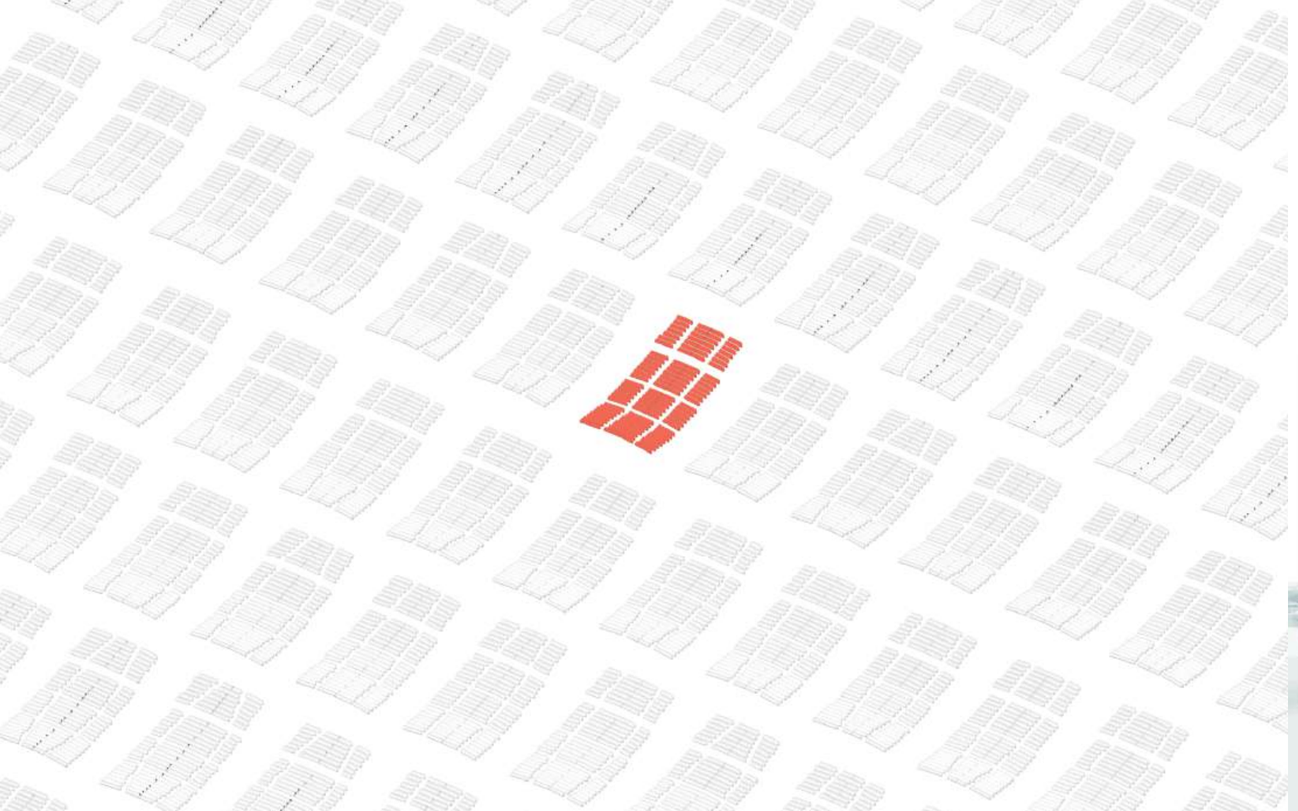
Design Beyond Human Cognition

Conclusion

The contemporary designers have a wider range of computer-based design tools more than ever. As these modeling, drawing, and simulating software develop, the algorithms behind the visible results become ever more sophisticated. While some of the computer aided design software are intuitive for the designers and thus widely used in the professional practices, other software that require deeper understanding in the computation have higher entry barrier-such as the Evolutionary Solver and therefore are less commonly used.

Although learning the Evolutionary Solver may require time and effort, learning the apparatus can be an extremely useful skill for architectural designers. Through the knowledge of the Evolutionary Solver, a designer may design beyond the human cognition by cooperating with the computer. While humans have knowledge and intuition, we are also subject to prejudices. On the other hand, computation algorithms are created with no preconception. Thus, the computers can generate design options, simulate and compare them fast and unbiased. Therefore, the computational processed designs are less subject to prejudice and personal preferences then the traditionally processed design. Thus, the designers can surpass the human cognition by cooperating with the Evolutionary Solver Program and other simulation algorithms and create innovative and objective design proposals.

In the last two research projects, “Performative façade” & “Data Management in Architectural Design”, data-driven design methods and data management techniques were covered. The aim of this research is to increase the precision of the former researches through the Evolutionary Solvers and grant the designers a skill to extract refined data through simulations. Through the knowledge gained in the series of research and practical applications data-driven design, the teams of H Architecture & Haeahn Architecture will be able to deliver innovative high-performance designs.



Computational and Algorithmic Design for Design Innovation in Architecture

Optimization Process in Architectural Design

H Architecture

This research paper was written and edited by Dongil Kim, Jake Jaekyung Han and John Hanghyun Cho at H Architecture. 2018.

Computational and Algorithmic Design for Design Innovation in Architecture

Optimization Process in Architectural Design

H Architecture