AS322747

Optioneering and Optimization Using Generative Design and Evolutionary Solvers

Maciej Wypych
BVN

Matt Wash
BVN

Learning Objectives

- Understanding of the principles of evolutionary solvers
- Pros and Cons of current evolutionary solvers
- Identify how and when these tools could be used within your business

Description

With the introduction of tools such as Autodesk's Refinery (previously Fractal) and Galapagos, Octopus and Wallecei plug-ins through Grasshopper and Rhino, the designer can explore an almost infinite number of design options.

It is however not as simple as it may appear. This talk will give an overview of the principles of optioneering and optimisation. It will discuss some of the limitations that exist and how you can get the most out of the tools that are currently on the market.

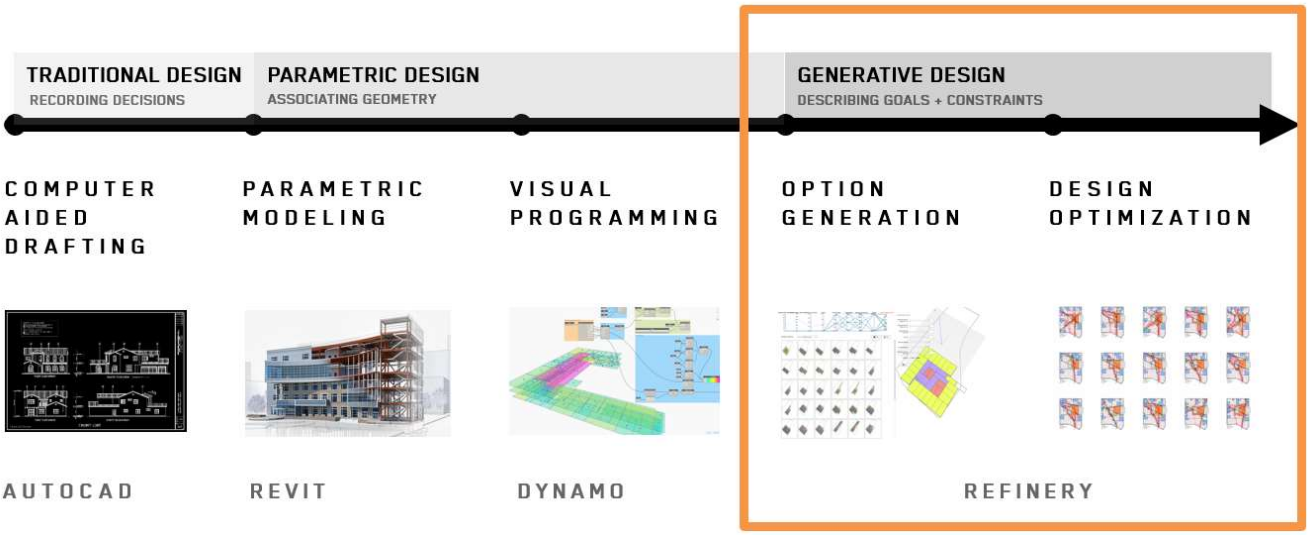
By the end of the talk attendees should have a sound understanding of how these tools could be applied to their field of work.

Speaker

Maciej Wypych is a Design Technology Coordinator at BVN. Prior to joining BVN Maciej was a Studio BIM Manager for Warren and Mahoney in Sydney. Maciej is also a sessional Tutor at University of New South Wales. He is a committee member and frequent speaker at Dynamo User Group Sydney as well as BUILT ANZ, Wellington Digital Design User Group and other conferences. He has over 15 years' experience in the architecture and building industry in Australia and UK.

Credits to Matt Wash – Design Technology Coordinator at BVN for creating this presentation

The Evolution of Generative Design



The introduction of AutoCAD mimicked the traditional manual drafting process. With the introduction of BIM tools such as Revit, parametric design was possible, enabling smart components that would react when an input was changed. These changes were updated in the BIM in all views that the element was visible in. This was a fundamental change in the evolution of CAD into a world where “information” was king. Visual programming tools such as Dynamo enabled manual tasks to be automated by carrying out repetitive tasks in a much faster way. The user interface was simple to pick up for non-coders and many old school drafter and technicians began to see the benefits of being able to manipulate their models through a scripting interface. Until Autodesk introduced “Fractal” (2016) using Dynamo for optioneering was a manual process of tweaking the variable inputs and seeing what effect this has on the output. Whilst exploring each option took a matter of seconds, it would take days if not weeks to explore a full spectrum of options if there were many input variables. “Fractal” allowed the user to automate the optioneering task, to suggest a range of options, many of which the designer may not have visualised manually, to filter down the inputs to select the most appropriate solutions. In November 2018 Autodesk released a beta of “Refinery” which took “Fractal” a step further offering the possibilities of both optioneering and optimisation.

“The goal of generative design is not to automate the design process, or to replace human designers with artificial ones”

– Danil Nagy, The Living

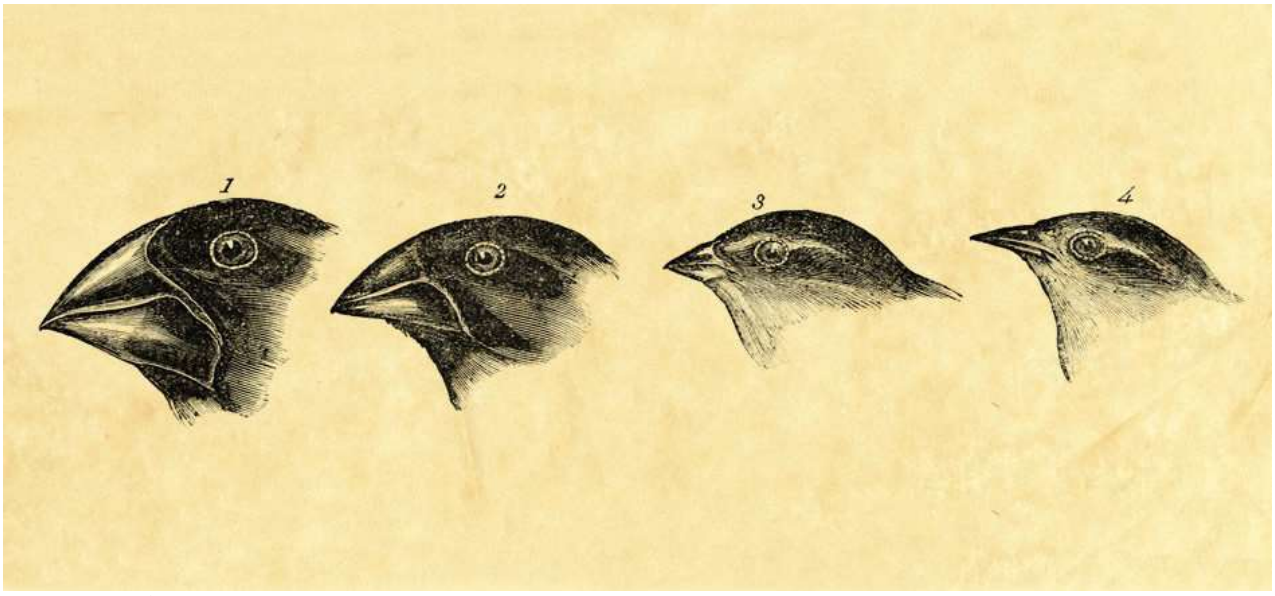
Looking beyond the evolution of CAD, BIM and Generative design at Autodesk, McNeel had offered Rhino and Grasshopper, from which developers have been building optimisation plug-ins for nearly 10 years.

An example of the first use of Galapagos, a single objective optimiser in 2010 by David Rutten.
https://www.youtube.com/watch?v=4wNcUwyyTPk

Octopus, a multi objective optimiser was released a few years later by the Unversity of Applied Arts, Vienna in collaboration with Bollinger+Grohmann Engineers.

In 2018 a new player entered the market in the shape of https://www.wallacei.com/

How do Evolutionary Solvers Work?



Evolution of a

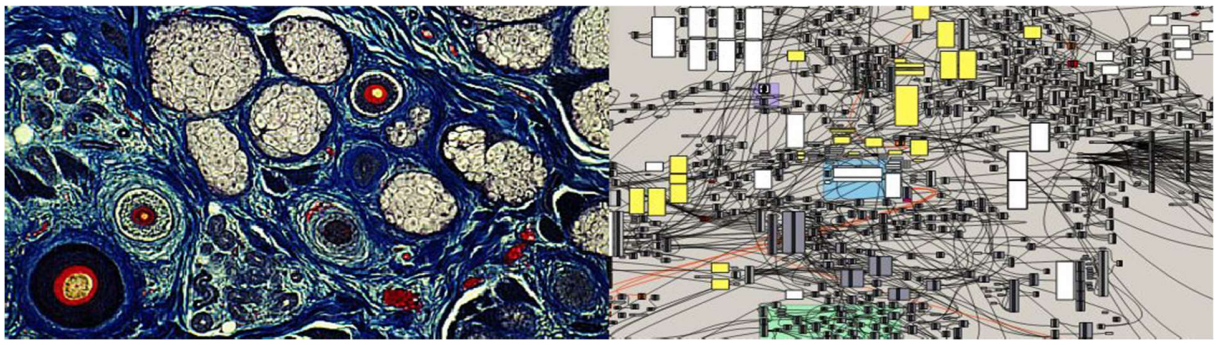
Charles Darwin (1809 - 1882)
I have called this principle, by which each slight variation, if useful, is preserved, by the term of Natural Selection.

EVOLUTIONARY ALGORITHMS SELECT THE BEST OPTIONS TO GENERATE MORE OPTIONS

PRINCIPLES OF BIOLOGIC AND ALGORITHMIC EVOLUTION

VARIATION
Without it, wouldn't be possible to pick something that is better than something else
INHERITANCE
Qualities of the individuals are transmitted to the next generations
SELECTION
The fittest individuals are likely to survive and have offspring

What is the difference between biological and algorithmic evolution?



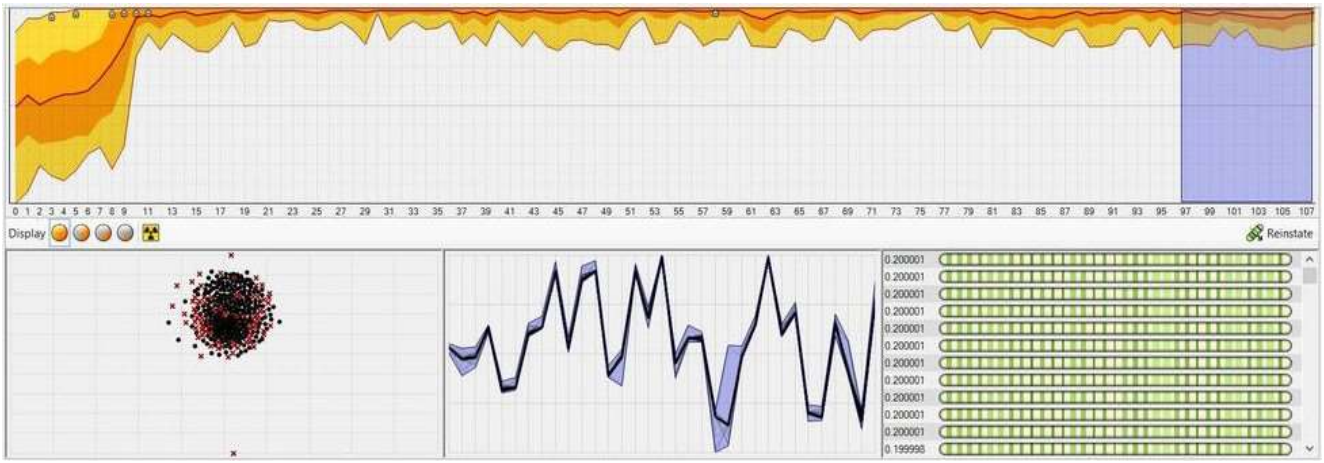
Biological evolution

- **Changing environment**
- **Entities interact with each other**
- **Gender and complex reproduction processes**
- **Arbitrariness**
- **Millions of inputs and variables**

Algorithmic evolution

- Stable environment
- Limited or inexistent interactions
- No sex or gender
- Simulated arbitrariness
- Limited amount of inputs

Pros and Cons of Evolutionary Solvers



Pros
- Flexible
- Progressive
- Forgiving
- Interactive

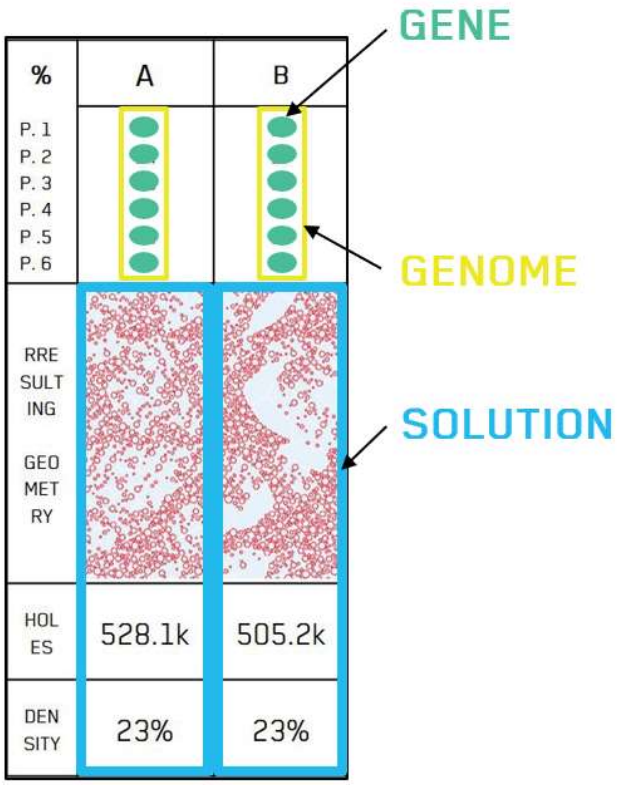
Cons
- Slow
- No Guaranteed Solution

What is a Gene and a Genome?
A combination of **GENES** is a **GENOME**, which outputs a unique **SOLUTION**

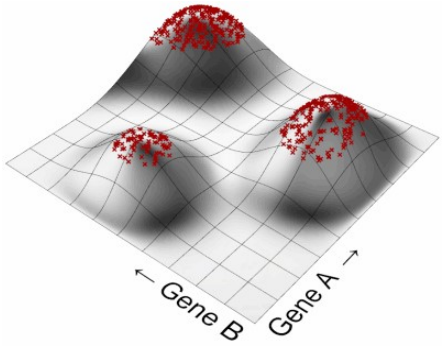
In the case of the ceiling pattern, there were 6 patterns to choose from, each pattern being a **GENE**. The combination all those patterns formed the **GENOME.** Changing the percentages of the patterns output various **SOLUTIONS.**

However, the percentage of each **GENE** were manually manipulated by the designer and there were no output values, the number of holes or percentage density that a minimum or maximum value was trying to be reached.

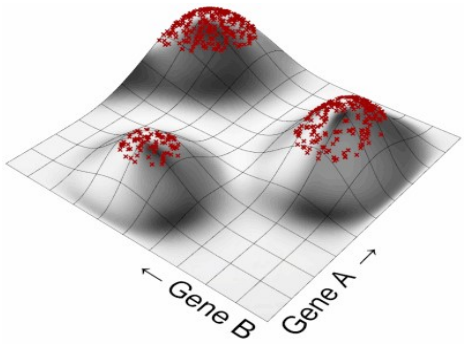
Therefore, in this example Rhino and Grasshopper were used for optioneering only.



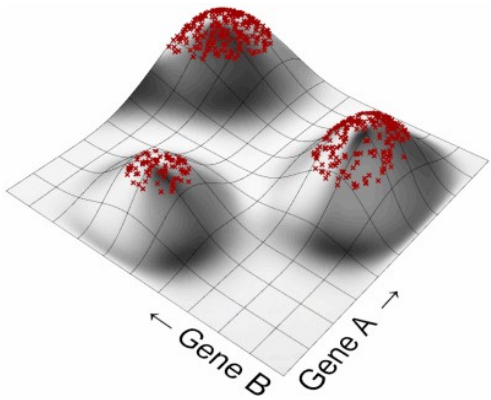
What is a fit genome and a fitness landscape?
FITNESS is whatever we want it to be. We are trying to solve a specific problem, and therefore we know what it means to be fit. All the possible solutions conform a **FITNESS LANDSCAPE**, which represents the nature of the problem that we are trying to solve.



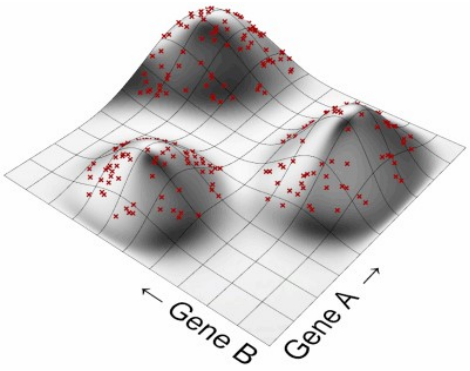
Fitness Landscape containing 2 genes or variables (A and B)
Unknown at the beginning of the process

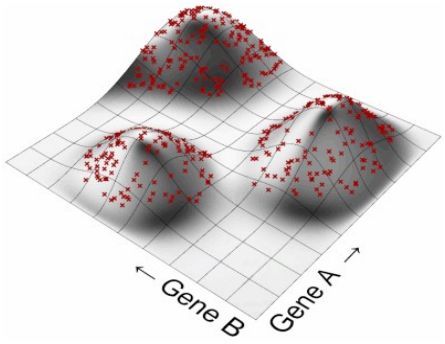


Basin of Attraction
Determine in which direction genomes should travel

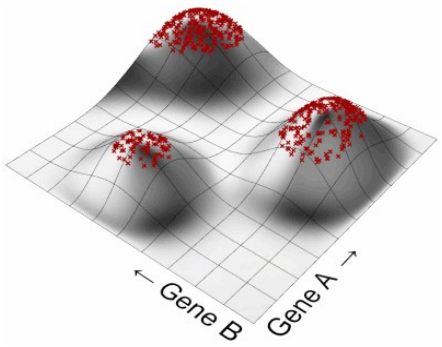


Start
Random population to approximate the nature of the Fitness Landscape



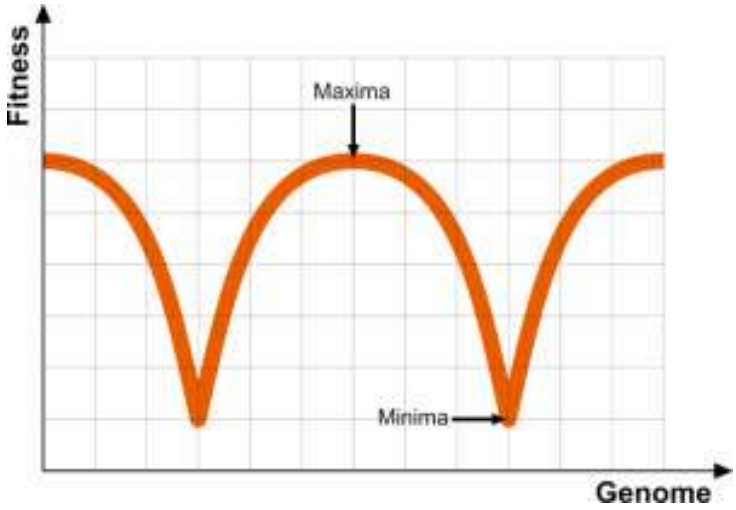


New Generation
Defined by fittest genomes and basin of attraction

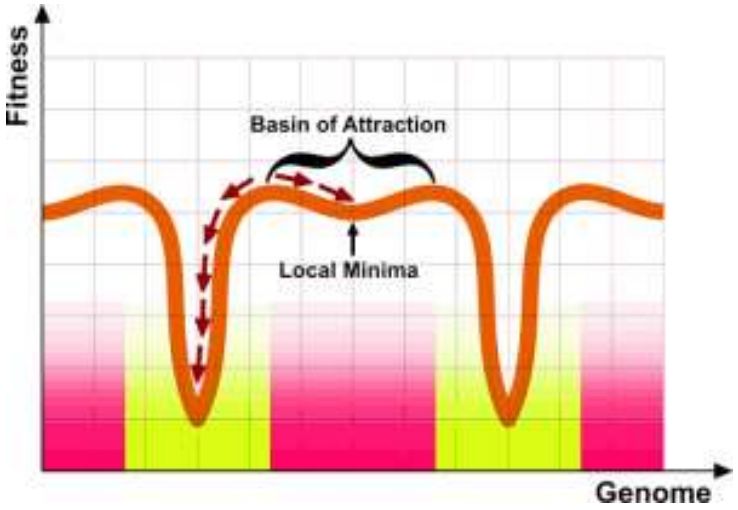


Subsequent Generations
The process repeats until satisfactory solutions are found (or not)

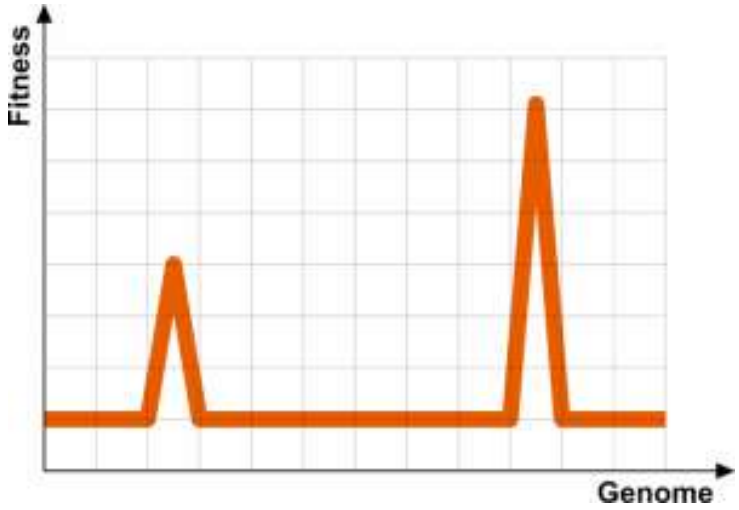
Fitness Landscapes



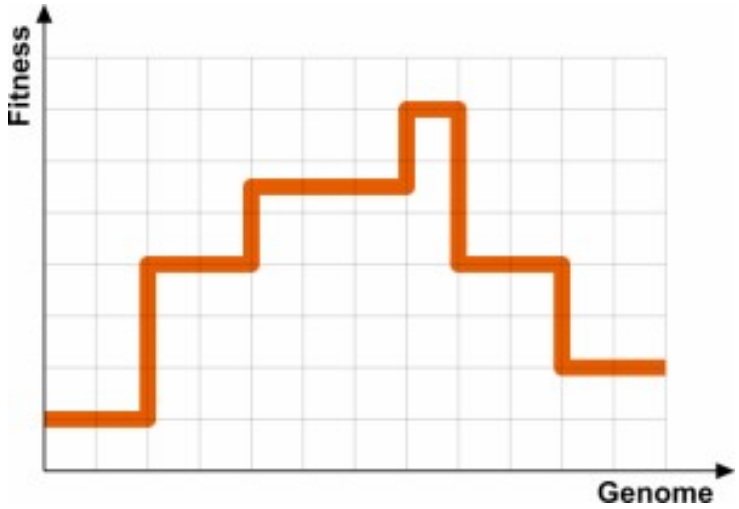
Simple
Basins of attraction will always take you to an optimal result



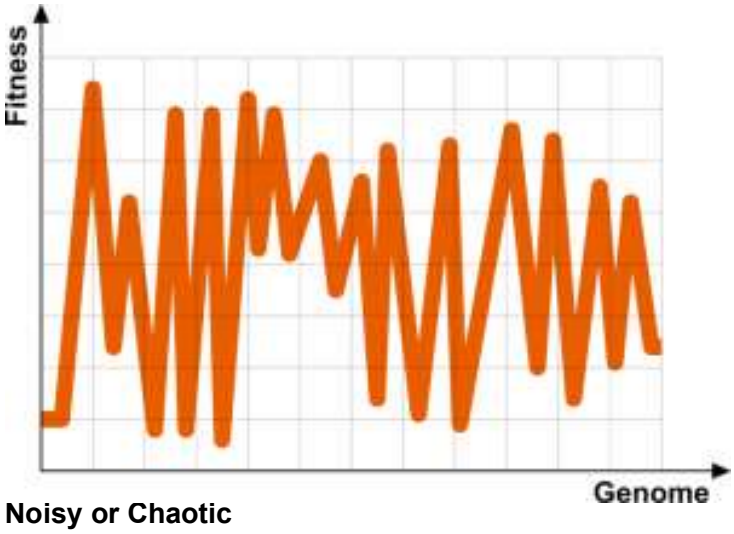
Complex
Half of the landscape dominated by a basin that attracts to a poor solution



Small Basins
Low chances of finding a (good) peak



Discontinuous
Plateaus without 'improvement'. No basin to an optimal solution



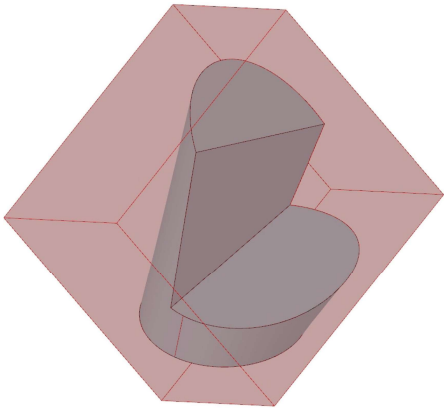
Noisy or Chaotic

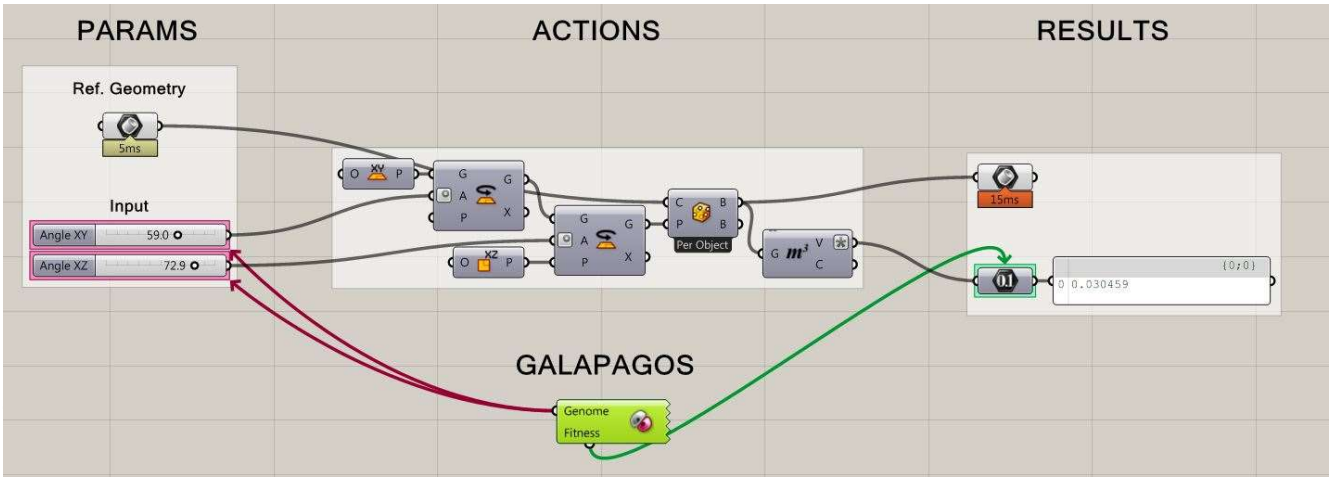
Impossible to make any intelligible pronunciations regarding the fitness of a local patch

Single and Multiple Objective Optimisation
Single objective optimization - Galapagos for Grasshopper

When a problem has only a single fitness function, i.e. only one output is required to be minimised or maximised, a single objective evolutionary solver will find an optimised solution. The number of input variables, and the range of these inputs will determine how quickly the optimised solution will be reached.

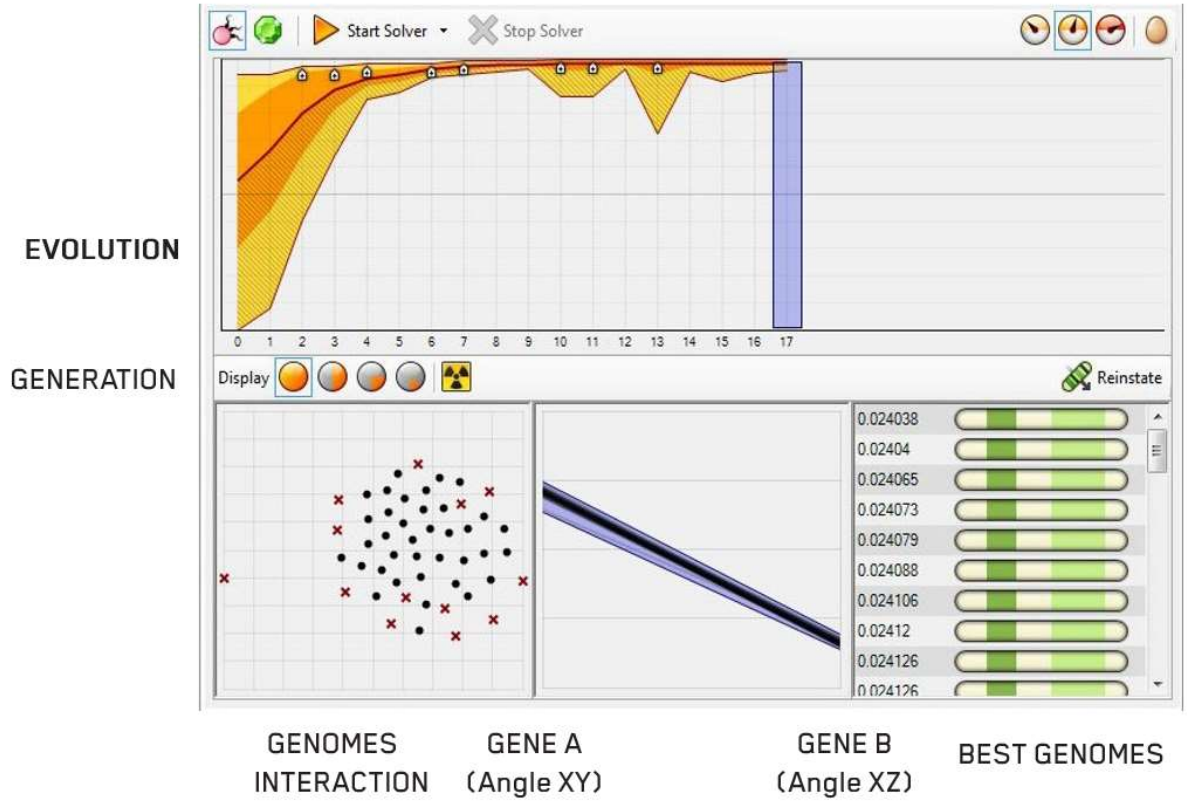
In the example of the bounding box, there are only 2 variable inputs, namely the rotation angles. The fitness function is the minimum volume.



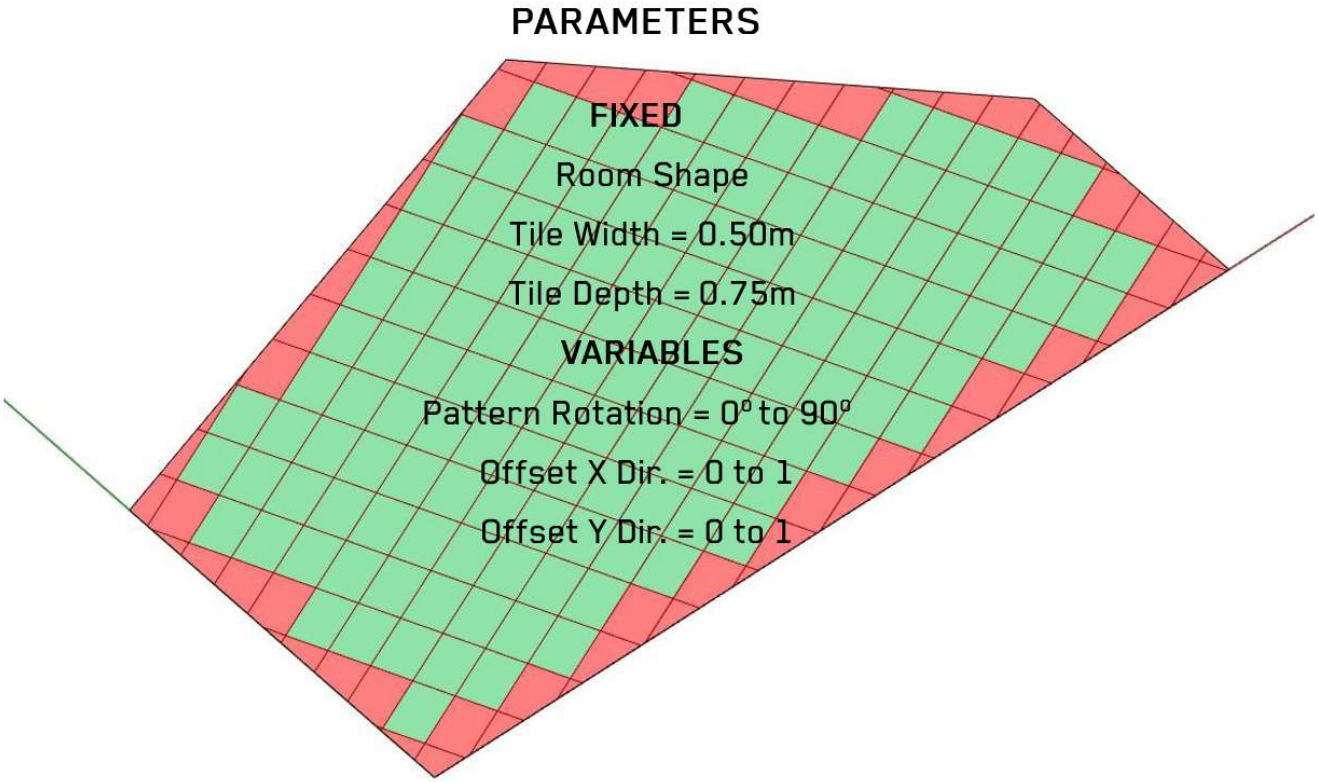


Interrogating the results in Realtime
One of the advantages Galapagos is the interaction the user can have once the solver has started. At any point the any of the better genomes can be reinstated and used as the new starting point from which to find the optimal solution.

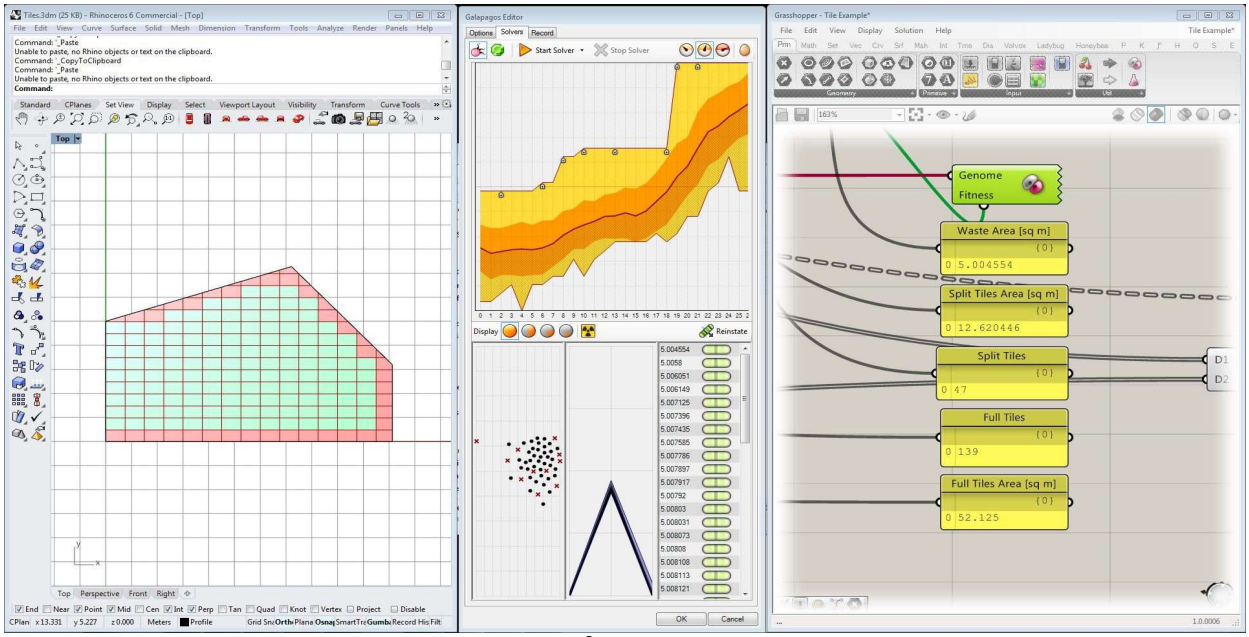
From the image below, it is evident that a genome close to the optimal solution was found after only 4 generations. This could indicate that because the optimal solution was only marginally improved upon in the subsequent generations, that the fitness landscape was similar to the simple example given earlier.



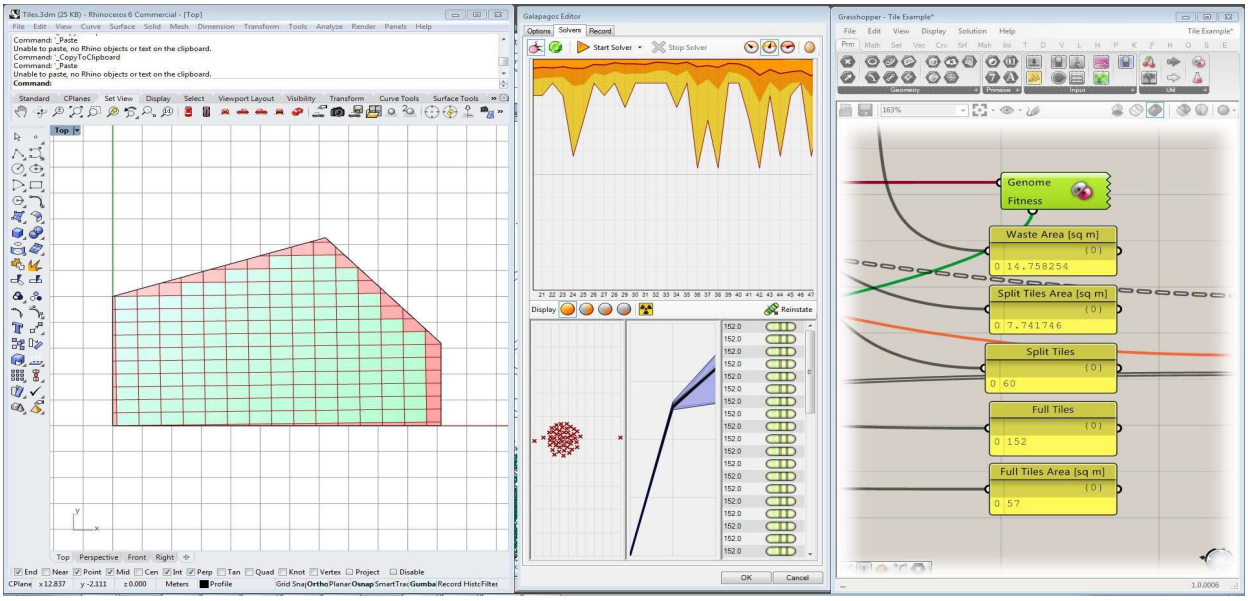
Using a single objective optimiser to determine two fitness objectives
Trying to determine the optimal tiling layout considering maximising the number of full tiles and minimising the waste area can be achieved independently of one another using Galapagos. Combing the two fitness functions would require an equation that would combine the maximum tiles and minimised waste area. This could be achieved with a single fitness function of say, time taken to lay and cut tiles, including the cost of the wastage, but without combining the two fitness functions, a single objective optimiser will only be able to generate a number of "good" solutions.



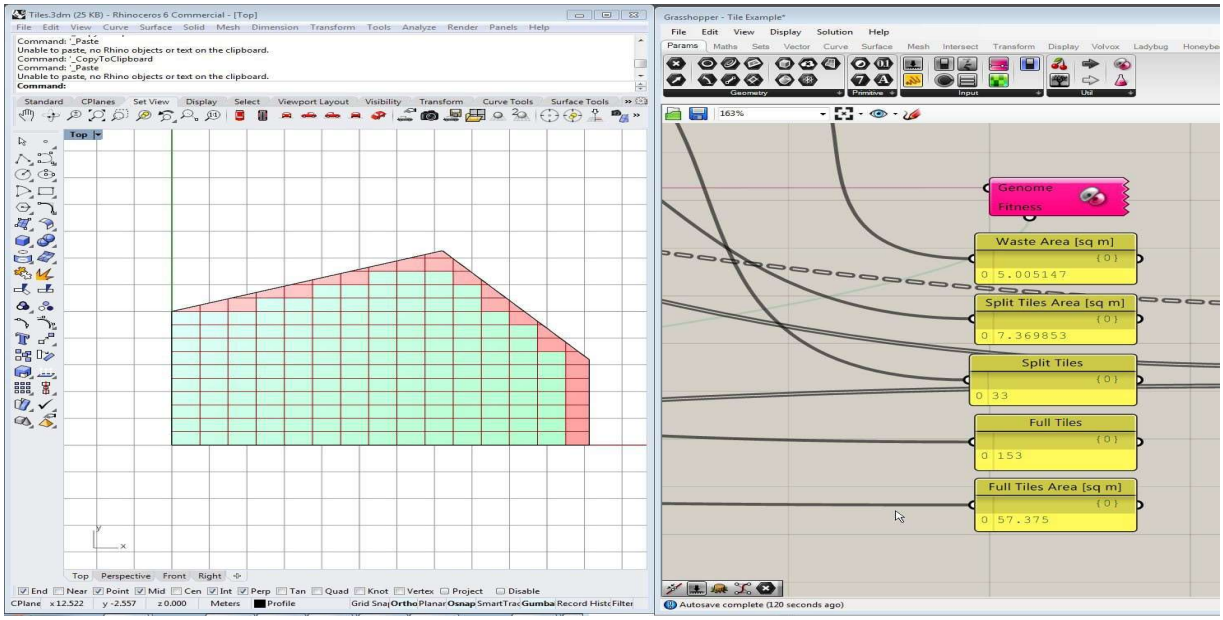
Combing the computational power of the computer with the common sense, intuition and experience of the human
In the tiling example the three variables included the rotation of the tile. Whilst the evolutionary solver eventually determined the tiles would be laid parallel to the tile depth along the longest boundary line, intuitively the human could have predicted this far quicker and ran a few tests along each boundary line rotation using only the x and y offsets as the variables. A tiler would also have probably said he would always start from one corner, so the optimal solution to this problem could have been solved by the human alone in a similar method to the ceiling example earlier.



OPTIMIZATION A - Minimum Waste Area [m²]

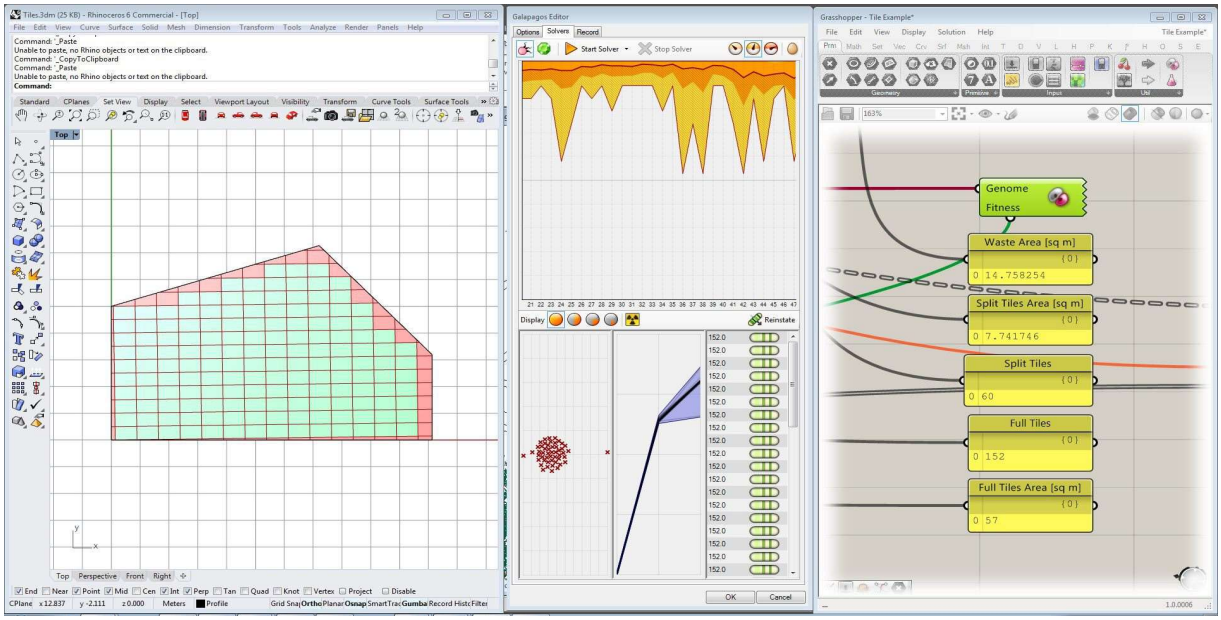


OPTIMIZATION - Maximum Full Tiles [n]



OPTIMAL SOLUTION

After studying both cases, we can manually set the optimal parameters

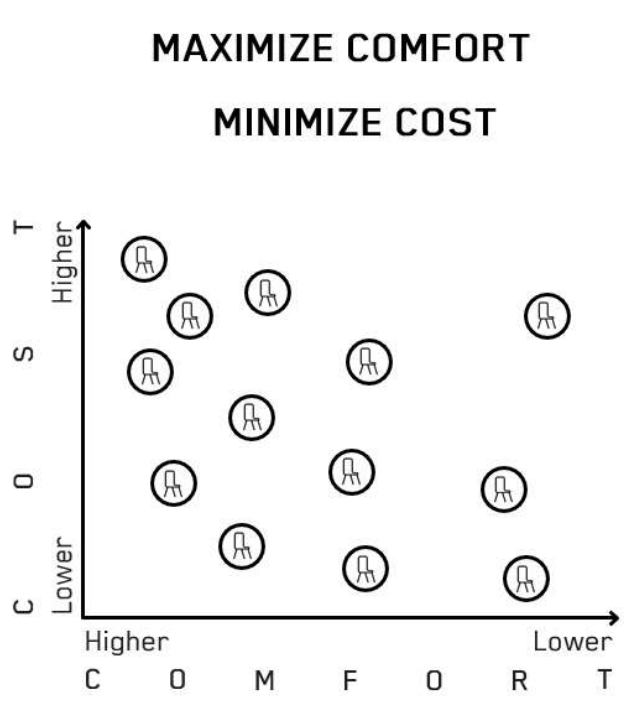


Multi objective optimization
An alternative to single objective optimisation is the possibility of combining more than one fitness function to arrive a few “best” solutions. When optimising for more than one outcome, a compromise will often need to be made.

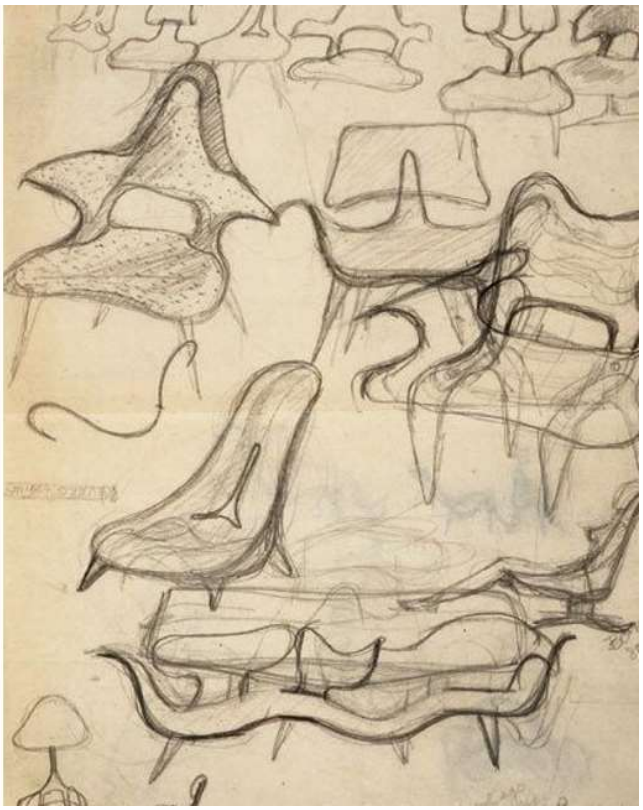
If we take the example of optimising a chair for comfort ‘v’ cost, it is highly unlikely that we will arrive at a single solution that is both the most comfortable and the cheapest. In addition, the chair problem has the added complexity of defining “comfort”. What is the most comfortable chair for one person may not be the most comfortable for someone else. Cost, however, is an ideal input to optimise for the minimum price.

In Architecture there will often be several outputs that cannot be quantified by a number. What is aesthetically pleasing to one person may not be as pleasing to another.

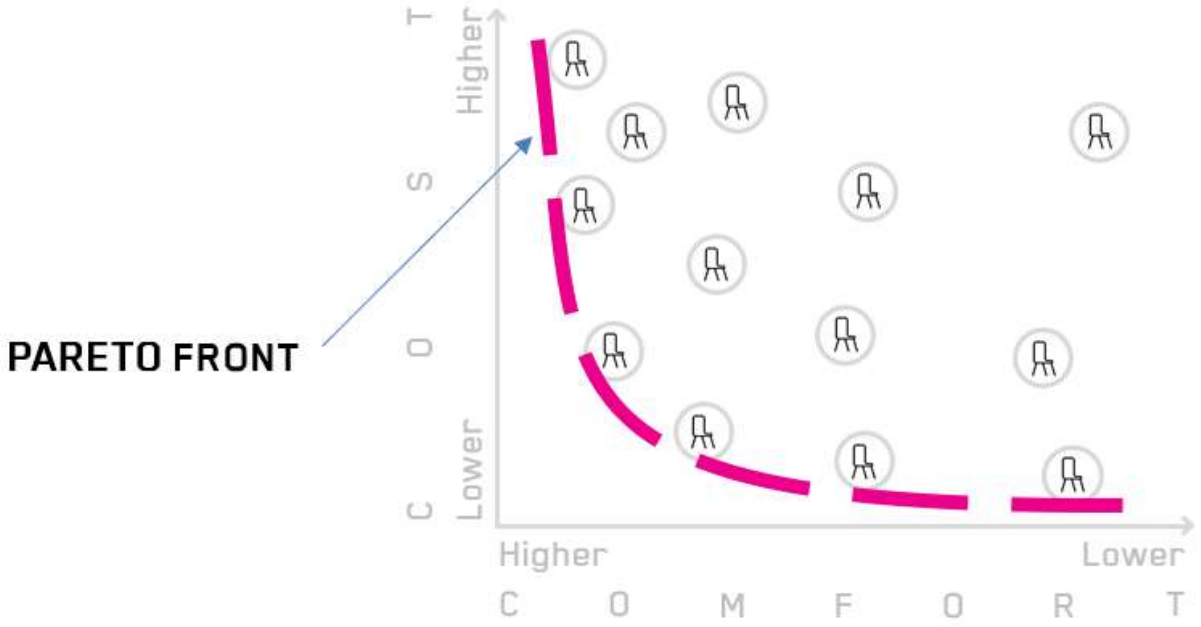
Typically, our goal in any problem is to minimise the cost and time and maximise the quality. The first two outputs can be calculated and are not ambiguous. Quality, however can be represented by many metrics, many of which are personal preferences which cannot be quantified through an evolutionary solver.



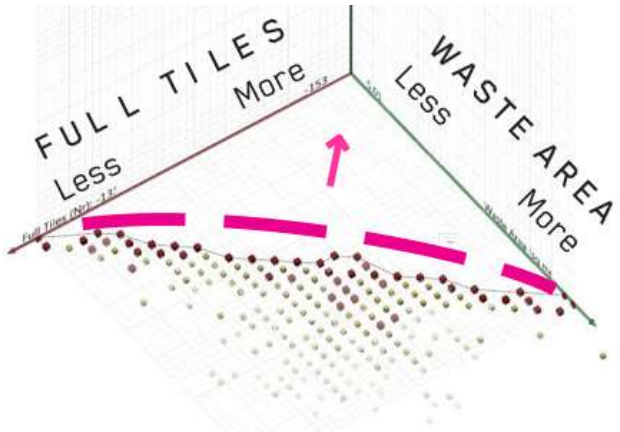
Each genome (variation) has a representation in the Multi Objective Optimization Graph

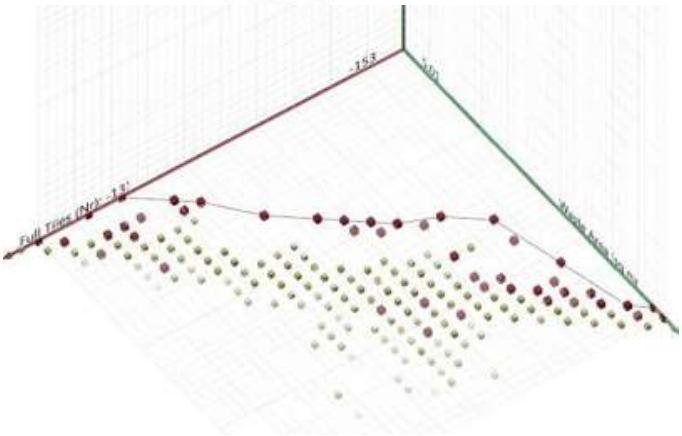


The Pareto Front
A collection of options that represent “optimal” solution will for a curve that moves closer and closer to the axis origin. This is called the “Pareto Front”.

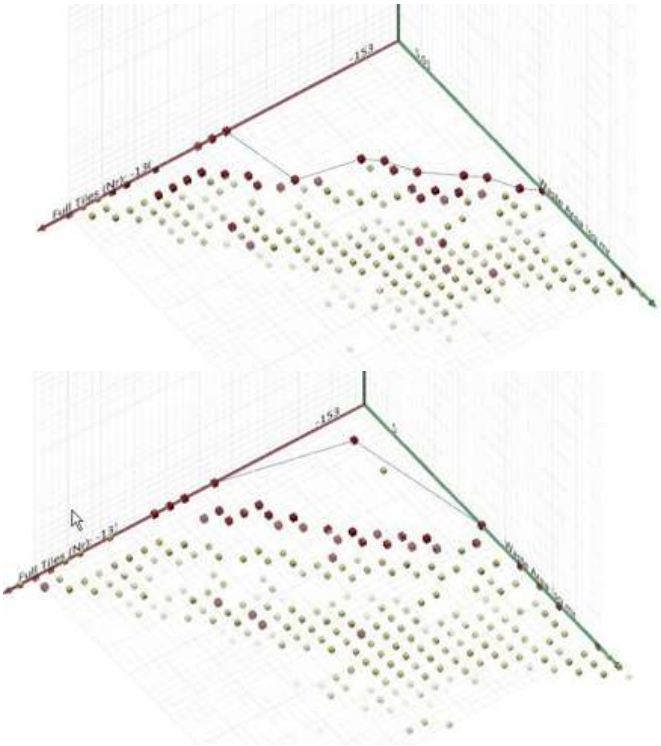


In the example of the floor tiles, the Pareto front can be seen to move closer to the axis origin as the number of generations increases.





Generation 18 **Generation 28**

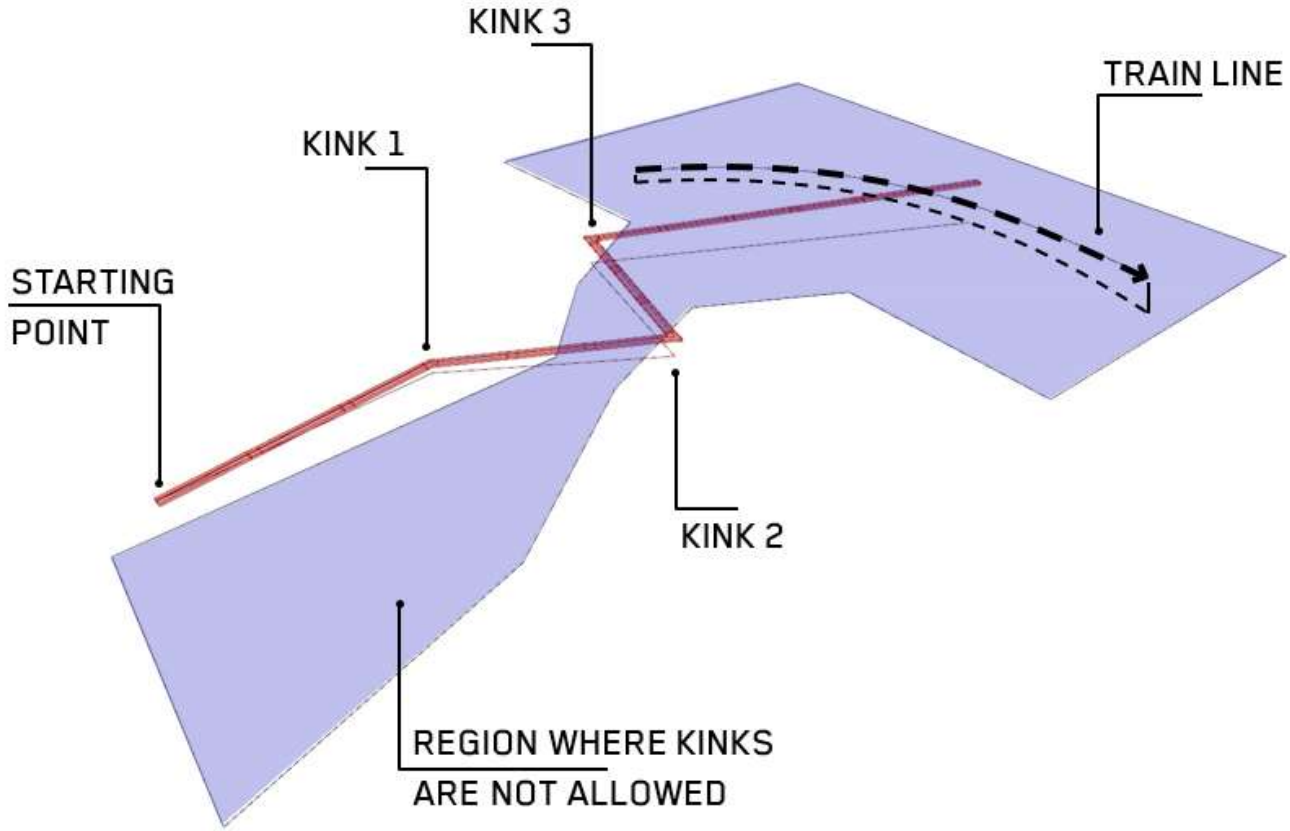


Generation 46 **Generation 52**

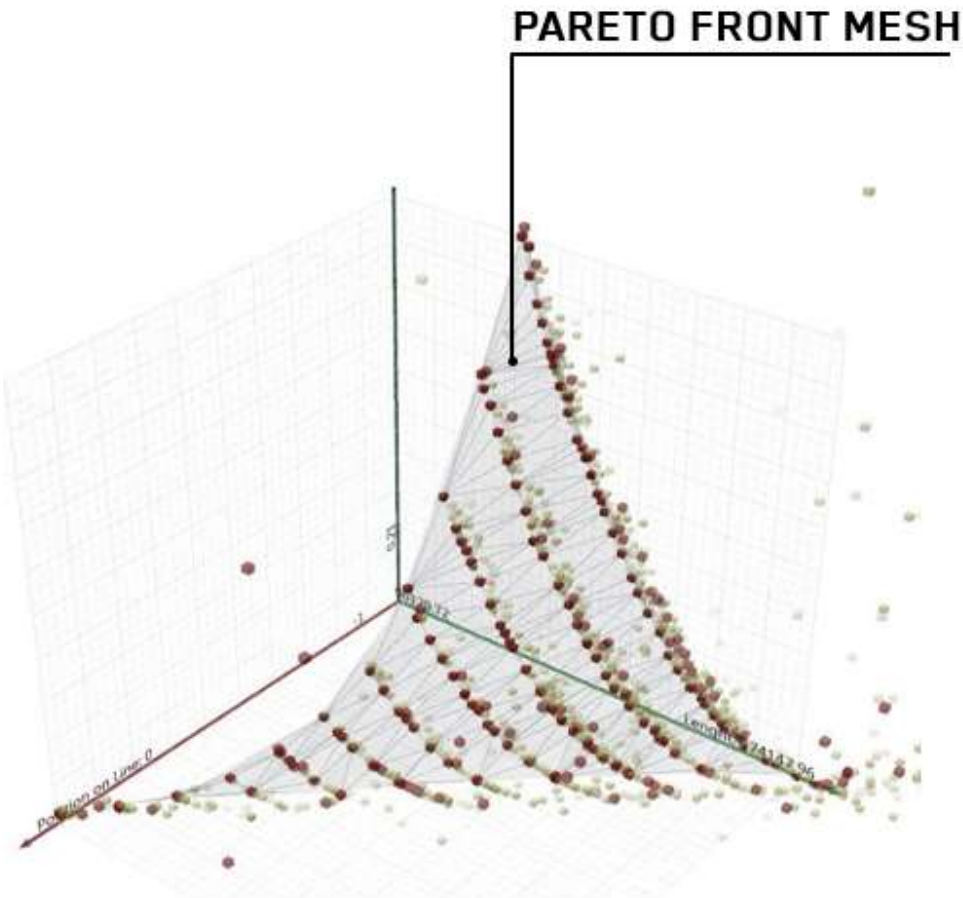
Pareto Front for 2 Outputs in Octopus for Grasshopper

Visualising the Pareto front is relatively simple when optimising for only 2 outputs, as shown above. The more outputs that are being optimised, the harder it becomes to visualise the Pareto front.

Optimising for 3 outputs
In the example of the bridge, the objectives were to minimise the length of the bridge, minimise the deviation from the train clearance and maximise the position of the end of bridge to be as close to the train station as possible.



Fixed parameters	Variables	Optimised outputs
Segments Length	Kink Position	Minimise Length
Landings Length	End of the bridge	Minimise clearance deviation
Ramp Gradient		Minimise distance from
station		
Landing Gradient		
Ramp Width		
Train Clearance		



Pareto Front Mesh for 3 Outputs in Octopus for Grasshopper

Comparisons of multi objective optimisers

For this study we compared three multi-objective optimisers, namely Octopus and Wallecei, which are free plug-ins for Rhino/Grasshopper and Refinery for Dynamo.

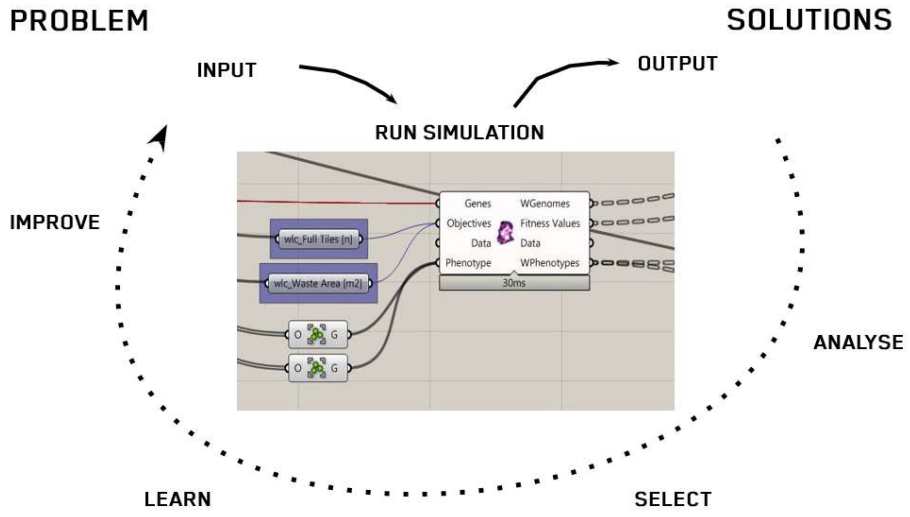
Octopus for Grasshopper/Rhino

Need to add some things in here about the ability to interrogate the model

Wallecei for Grasshopper/Rhino

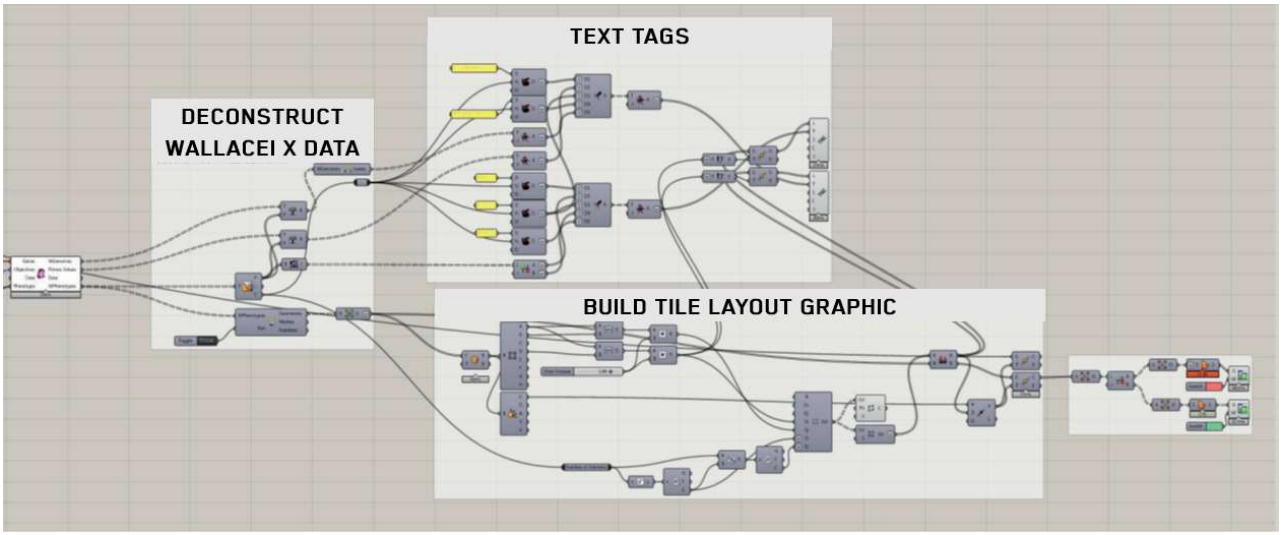
Wallecei is the most recent addition to the market of multi-objective optimisers having been released towards the end of 2018. The free plug-in from Grasshopper enables users to have far more interaction with the evolutionary process and the data that is being generated.

The grasshopper script required to run the optimisation runs in a similar way to Galapagos and Octopus regarding its inputs, the difference is the ability to take the data for further analysis as the user sees fit.

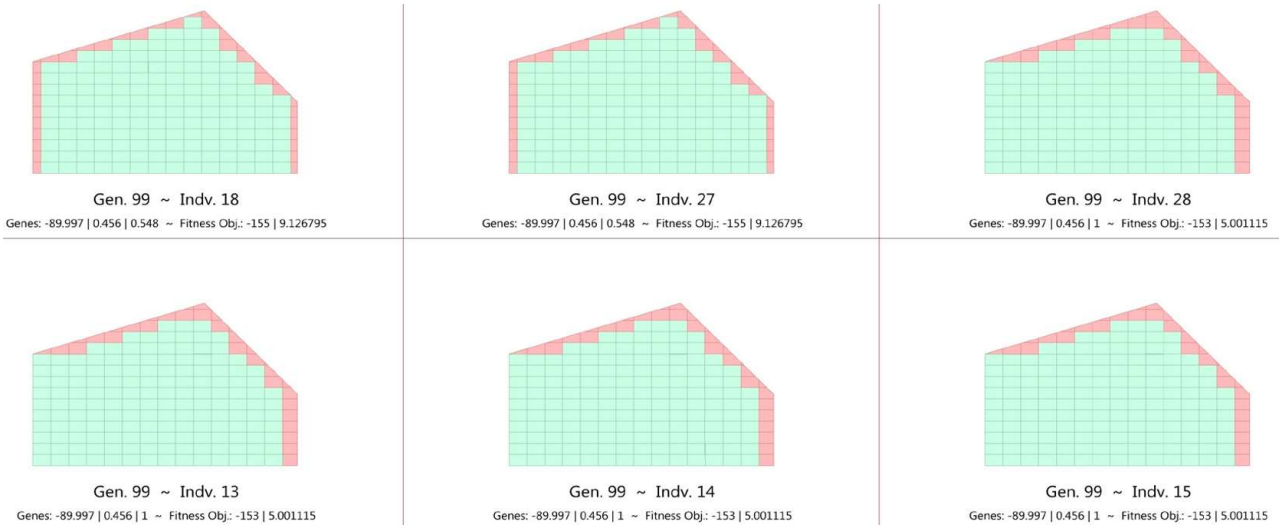


In the example of the tile optimisation routine, Wallecei focuses on maximising computation by focusing on the data and minimising the visual output of the tiling pattern. The benefit of this in terms of time taken to find the optimised solutions is evident.

The pattern was generated post analyse, once the results had been interpreted to minimise the number of unwanted patterns.



Deconstructing the Wallecei data to generate the tiling patterns - Script



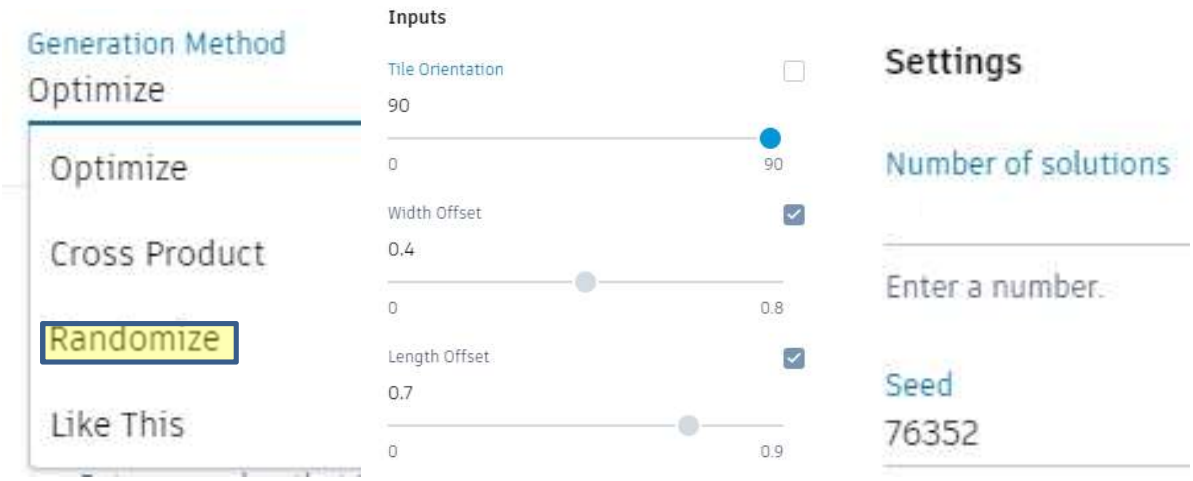
Deconstructing the Wallecei data to generate the tiling patterns – Output Patterns

Refinery for Dynamo

"An Autodesk generative design **beta** for the architecture, engineering and construction industry that gives users the power to **quickly** explore and optimize their Dynamo designs."

Refinery was release around the same time as Wallecei and is an extension of Autodesk's Project Fractal.

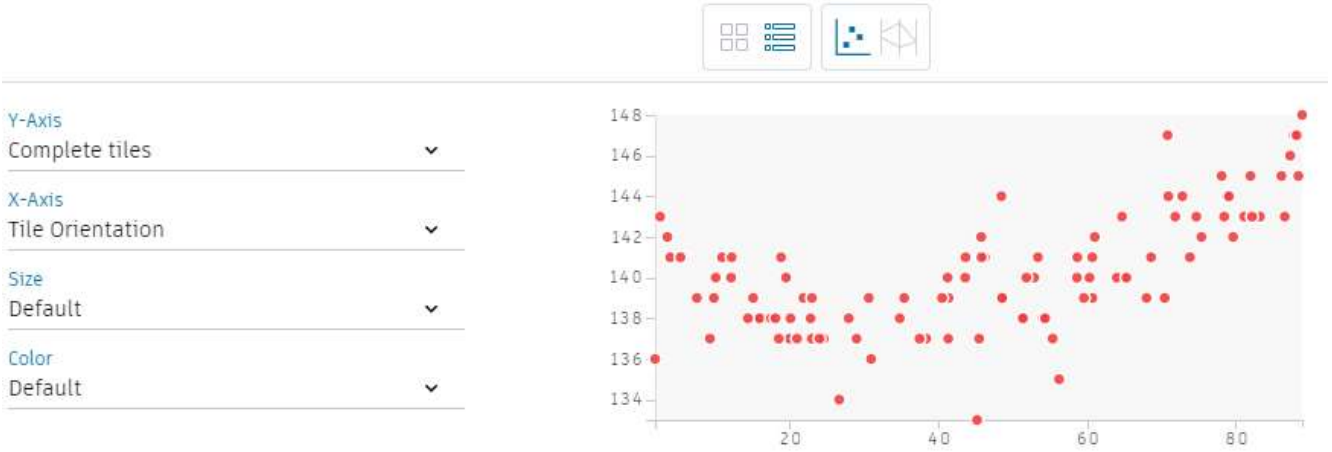
Refinery Generation Methods
Within the Generation Method inside refinery the user has 4 No. options to choose from.



Generation Method - Randomise Solutions – (Optioneering)
Selecting “Randomise” allows the user to carry out some “optioneering” by selecting the variable inputs that are to be included in the randomisation and the “number of solutions” to be explored. The “Seed” is just a number to control where the randomisation starts. Any number can be entered here. The choice of number cannot be consciously added to positively or negatively affect the outcome. Selecting a different “Seed” each time the solver is ran which just generate a different starting point for the possible solutions.

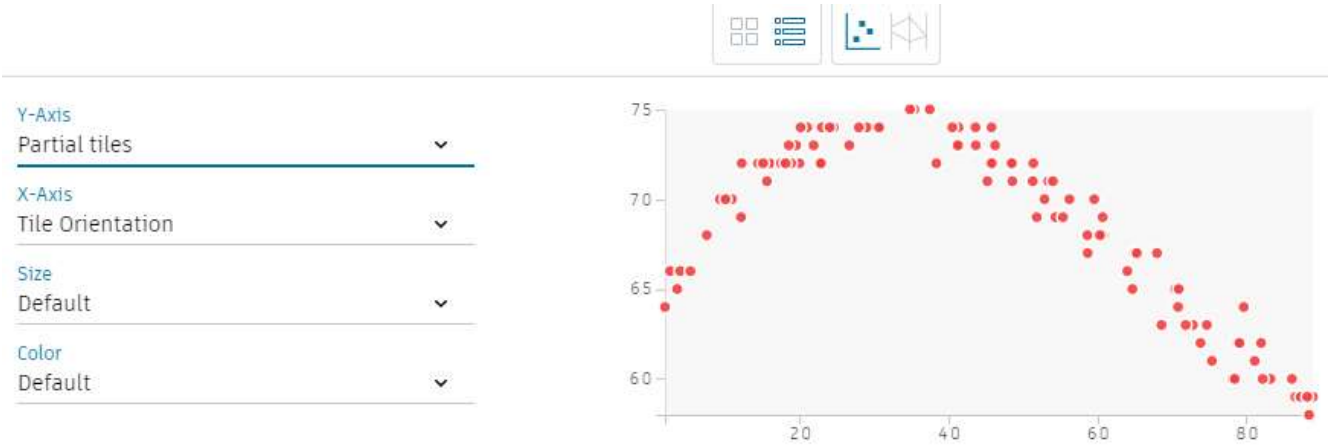
In the tiling example, after we had learned that going straight to “Optimisation” before “Optioneering” a study of 100 possible random solutions was used as an initial study. This took approximately 5 minutes to run. The results for which are noted below.

The refinery user interface allows for interpretation of the results in real-time and once the solver has finished. If a trend is observed during the solving stage, the process can be stopped, settings modified and re-ran.



Visualising Random Options – Completed Tiles (Y-Axis) against Tile Rotation (X-Axis)

It could be interpreted that the maximum number appears to be maximised when the tile orientation trends towards 90 degrees.

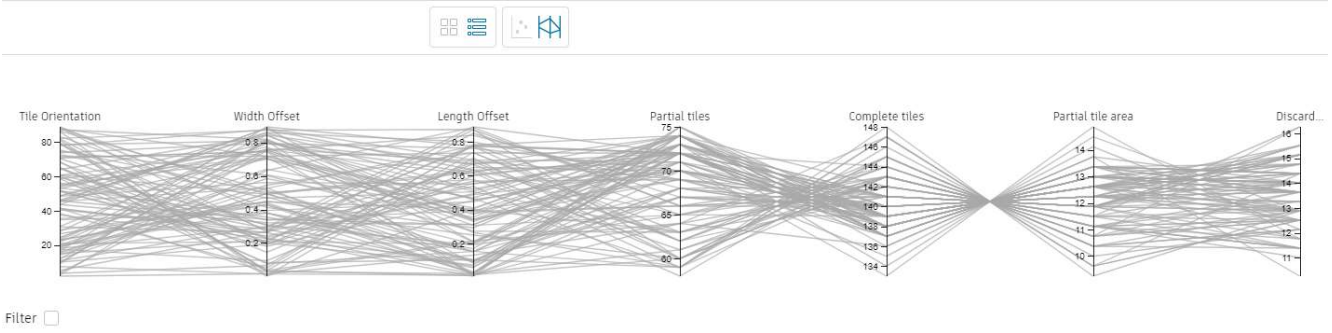


Visualising Random Options – Partial Tiles (Y-Axis) against Tile Rotation (X-Axis)

This theory is backed up when looking at the minimum number of partial tiles, where the optimised rotation appears to be 0 or 90 degrees.

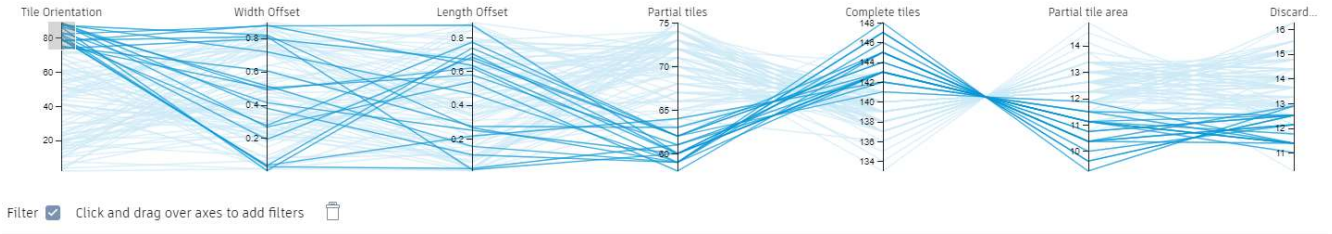
Without even running the simulation, logic would suggest that aligning the tile to the longest boundary edge on either the short or long edge of the tile would be a good starting point. These graphs support that logic.

The results can be visualised by viewing all the combinations of each genome, in graphic and numeric form. The results can be sorted (smallest/largest) via the numeric chart or filtered by dragging over a single input.



Partial tiles	Complete tiles↓	Partial tile area	Discarded area	Tile Orientation	Width Offset	Length Offset
59.0	148.0	9.242	12.883	88.934	0.379	0.104
64.0	147.0	9.617	14.383	70.831	0.745	0.011
59.0	147.0	9.617	12.508	88.196	0.489	0.619

Visualising Random Options – Genome combinations with numeric values ordered by maximum completed tile

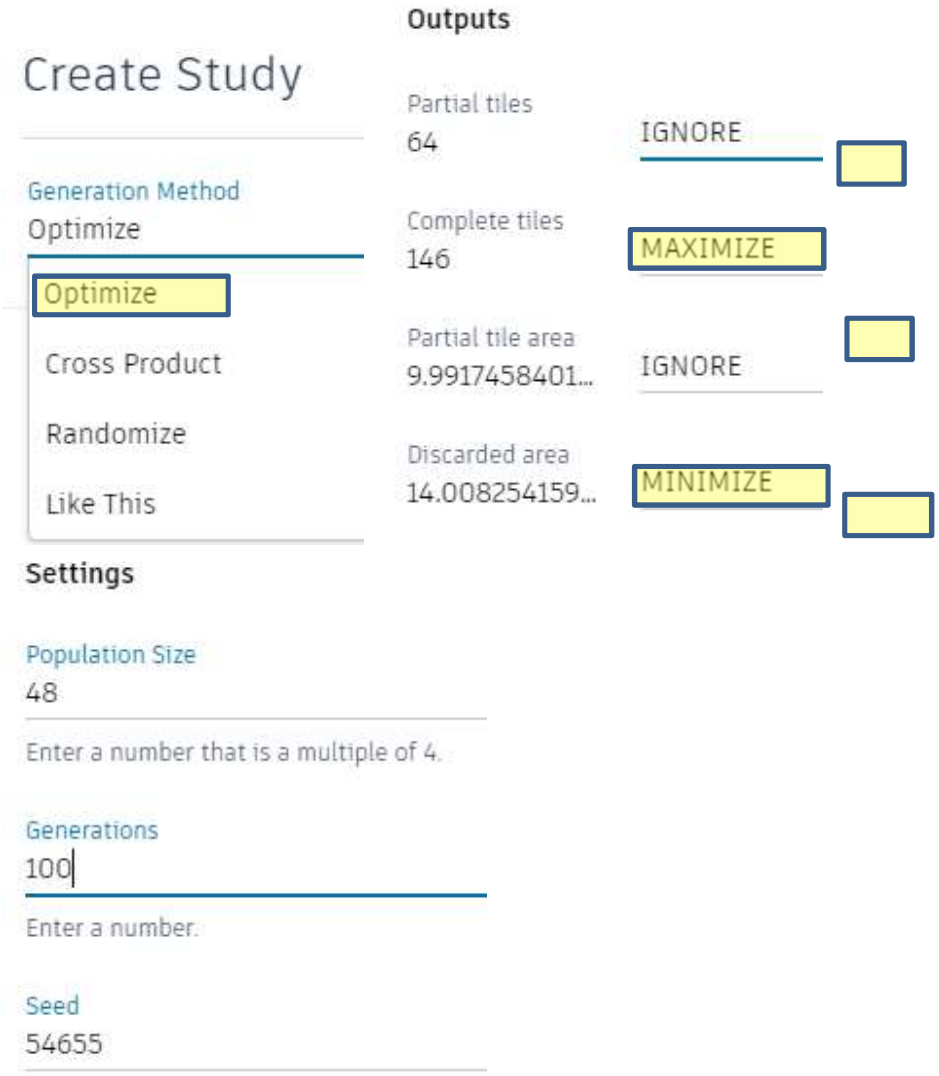


Partial tiles	Complete tiles↓	Partial tile area	Discarded area	Tile Orientation	Width Offset	Length Offset
59.0	148.0	9.242	12.883	88.934	0.379	0.104
59.0	147.0	9.617	12.508	88.196	0.489	0.619

Visualising Random Options – Genome combinations with numeric values ordered by filtering the tile orientation to approximately 75-90 degrees only

In this study, these graphs are much harder to interpret, but filtering by tile orientation close to 90 degrees does highlight that these solutions are generally in the least partial tiles and most completed tile range.

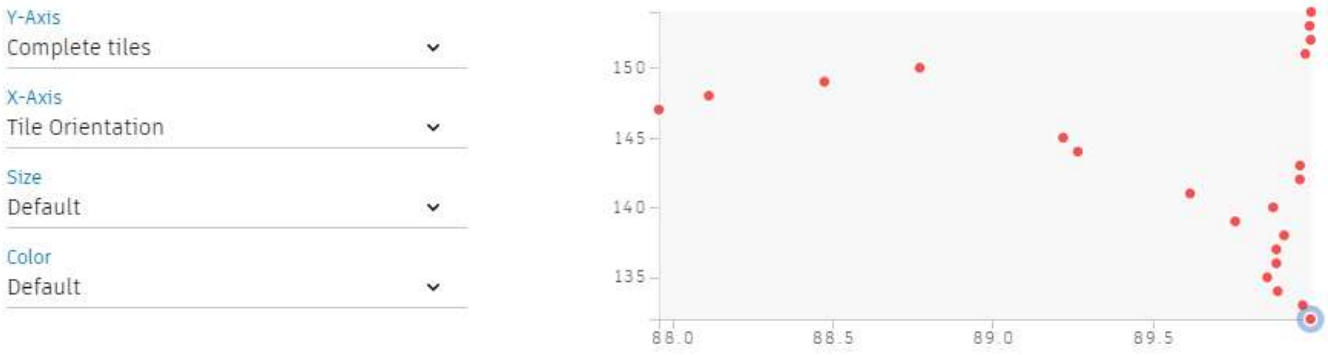
Generation Method Optimize (Evolutionary Solver – Population/Generation)



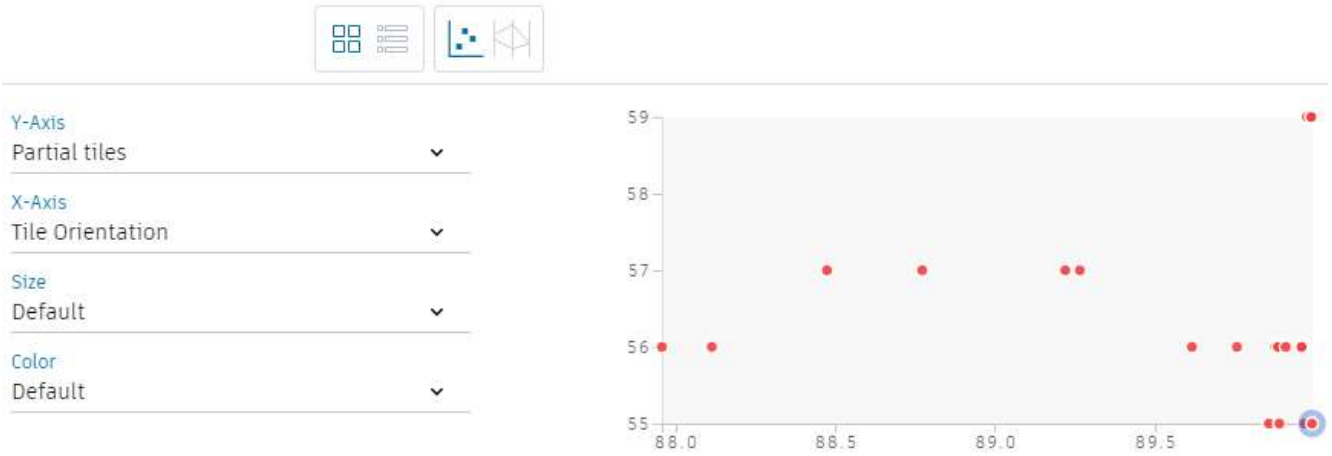
Optimised Settings

Generating an optimisation study for a population size of 48, with 100 generations took approximately 45 minutes to run.

Although 4800 possible options were analysed, only one single solution was suggested from each generation.

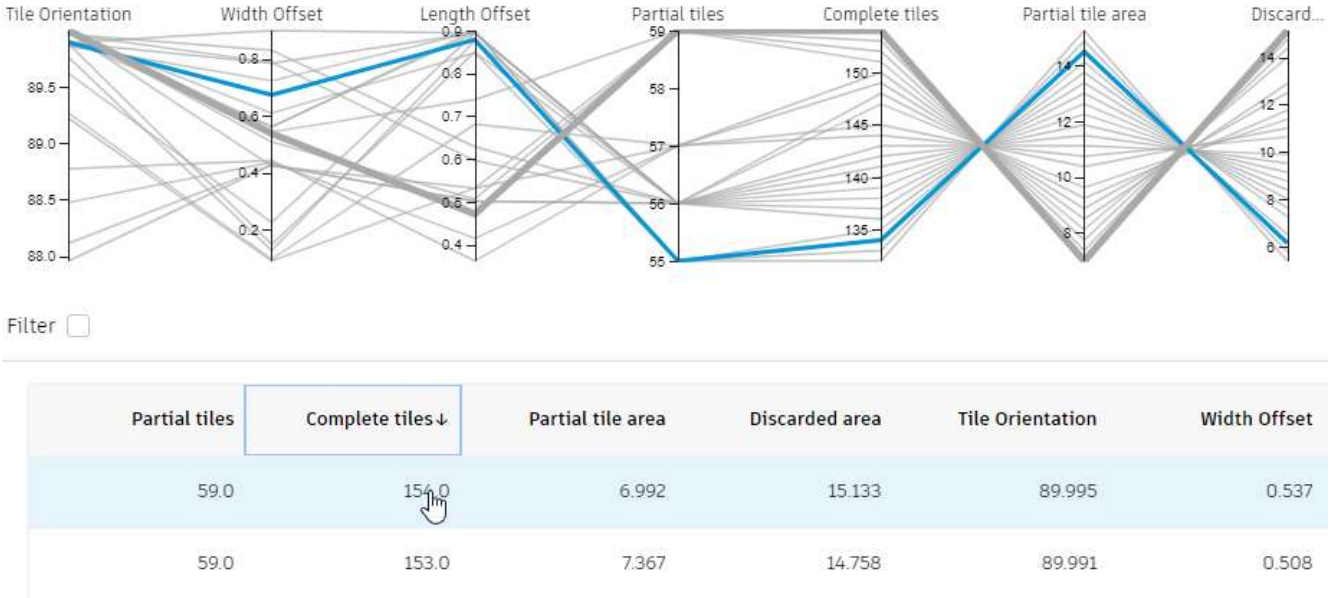


Visualising Optimised Solutions – Completed Tiles (Y-Axis) against Tile Rotation (X-Axis)
The interpretation of the randomisation generation that the maximum number of complete tiles occurs with the tile orientation of 90 degrees has been confirmed by the optimisation method above. All of the optimised solutions fall between 88-90 degrees.



Visualising Optimised Solutions – Partial Tiles (Y-Axis) against Tile Rotation (X-Axis)

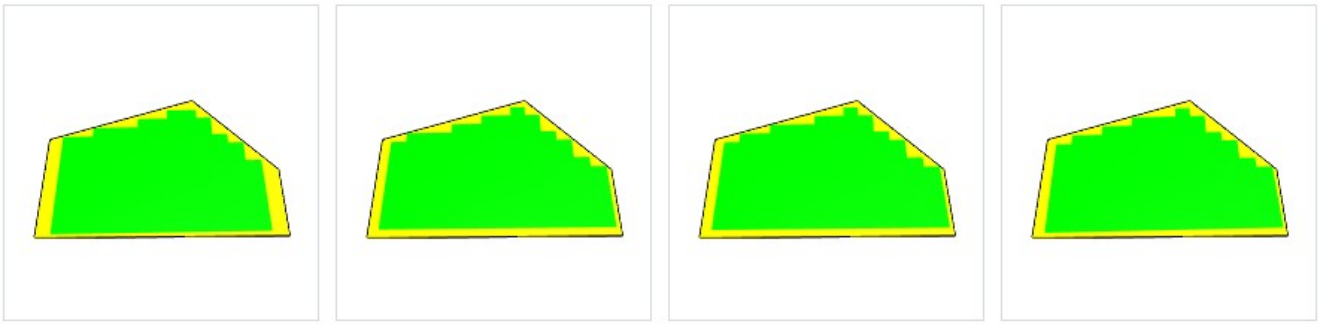
Again, the tile orientation required to minimise the number of partial tiles for the optimised solution only has a range of 88-90 degrees.



Partial tiles	Complete tiles↓	Partial tile area	Discarded area	Tile Orientation	Width Offset
59.0	154.0	6.992	15.133	89.995	0.537
59.0	153.0	7.367	14.758	89.991	0.508

Visualising Optimised Solutions – Genome combinations with numeric values ordered by maximum completed tile

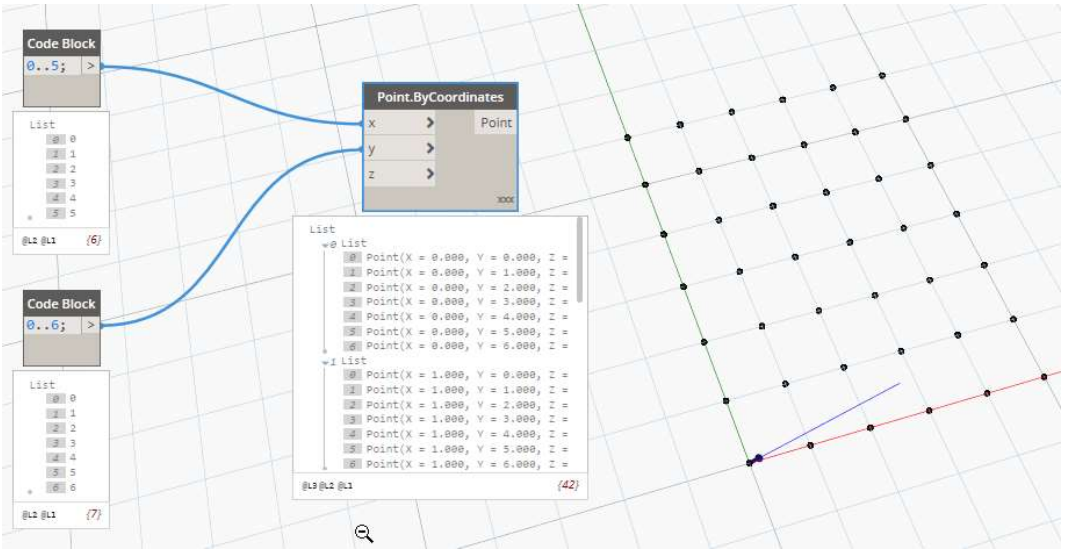
A criticism of the graphical tiling layout is that it is not clear where the tiles are laid length ways along the longest edge, for 0 degrees and short ways for for 90 degrees.
This is much clearer in the earlier Wallecei example.



Visualising Optimised Solutions – Graphical Tile Layout

Generation Method - Cross Product

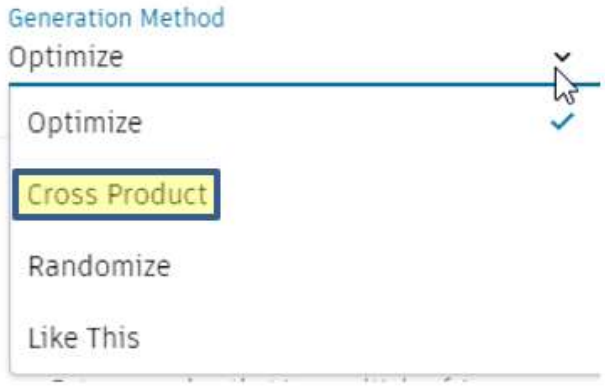
For those familiar with the cross-product lacing option within Dynamo, input variables will be tested with every combination possible. Therefore, a total of 42 solutions will be computed with an input with 6 items combined with an input of 7 items.

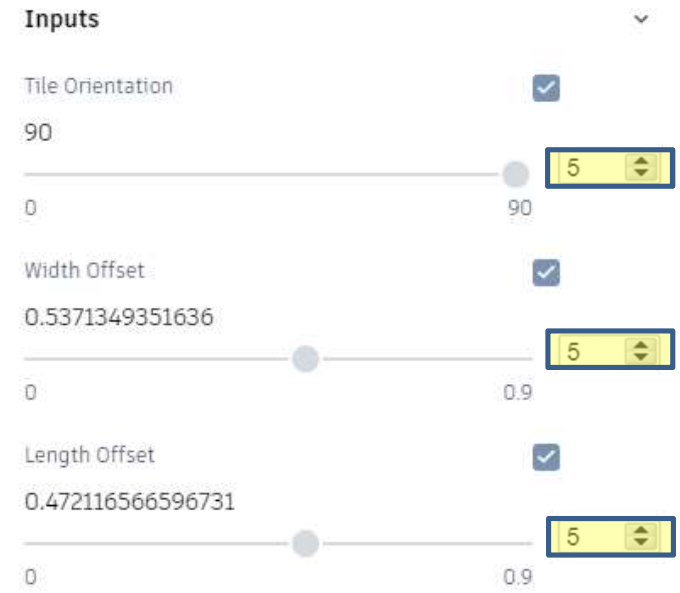


Cross Product Lacing within Dynamo

As it would often lead to an almost infinite number of solutions when using this approach in the tiling example, when using the cross-product function within Refinery the user can specify the number of variations of each input from all variables.

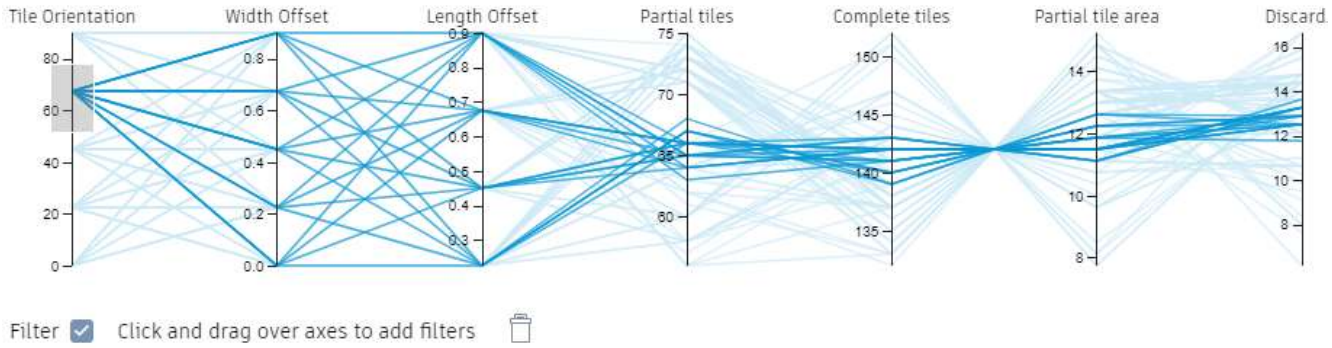
In the example below, the solver ran 125 designs in around 5 mins based on the cross-product options with 5 variations for each input (5x5x5) dividing the input range equally.





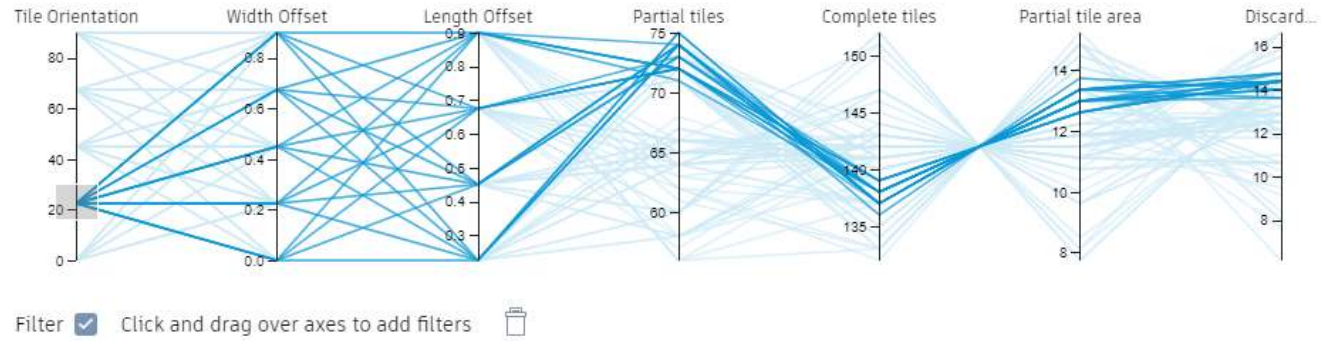
Cross Product Settings

Visualising the parallel coordinate view, it is evident that the optimised solution for any of the output values is not coming from the tiling orientations of 67.5 degrees and that the offset width or length make little difference to the output values. All possible solutions return neither good or bad solutions.



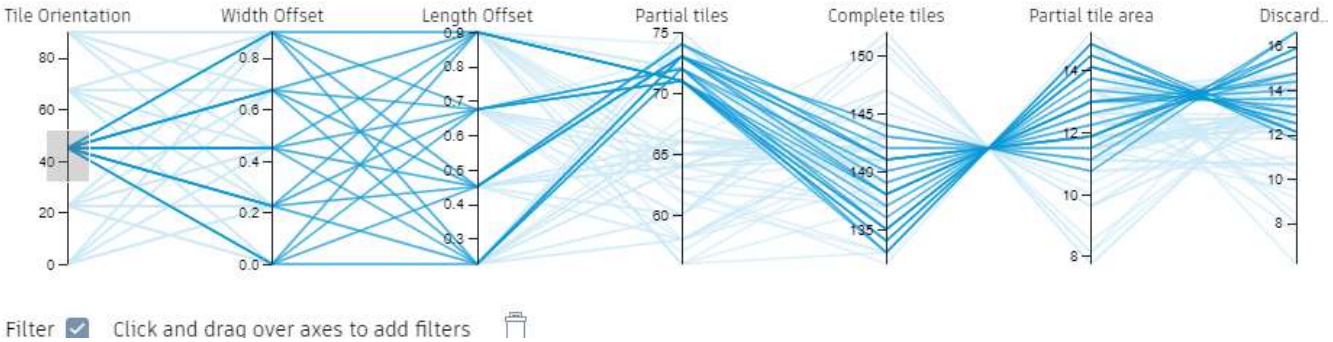
Visualising Cross Product Solutions Filtering for tile orientation of 67.5 degrees

Filtering for 22.5 degrees indicates most of the solutions are undesirable



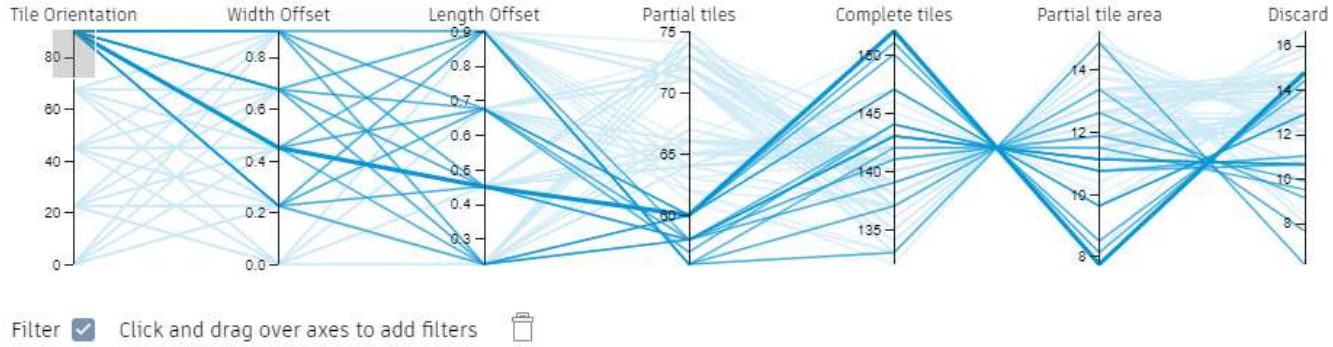
Visualising Cross Product Solutions Filtering for tile orientation of 22.5 degrees

Filtering for 22.5 degrees indicates most of the solutions are undesirable with a larger spread of results for completed tiles, partial and discarded tile area

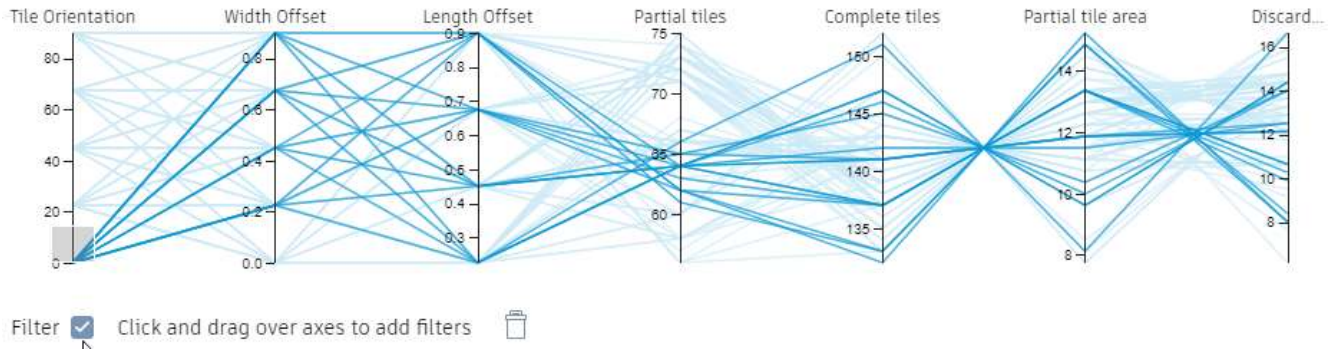


Visualising Cross Product Solutions Filtering for tile orientation of 45 degrees

Filtering for 90 degrees indicates most solutions are optimal solutions but it is clear that a solution to achieve maximum complete tiles, minimum partial tiles, partial tile area and discarded area will not be possible, and a compromise will need to be made.



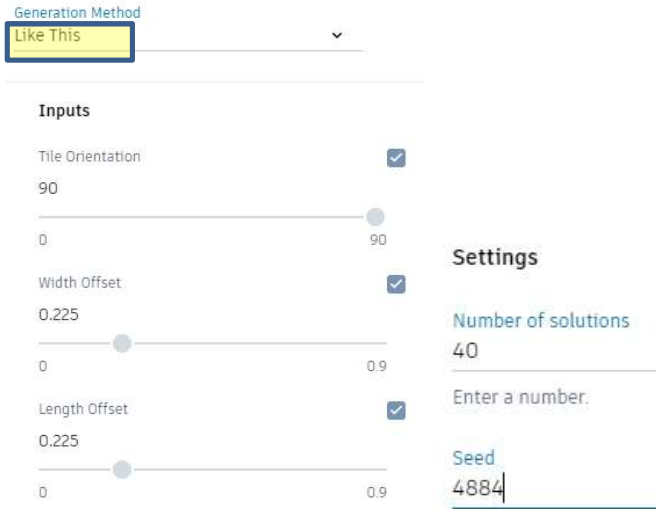
Visualising Cross Product Solutions Filtering for tile orientation of 90 degrees
Possibly the most interesting results are when the tile orientation is filtered to 0 degrees. The number of partial tiles. Whilst it has been noted that a tiler would set out from the longest edge at one of the ends, would it have been possible to predict whether lying the tiles along the longest or shortest length would have achieved an optimal outcome? For the four outputs that could be optimised, none of the solutions in this study come from a 0-degree tile angle.

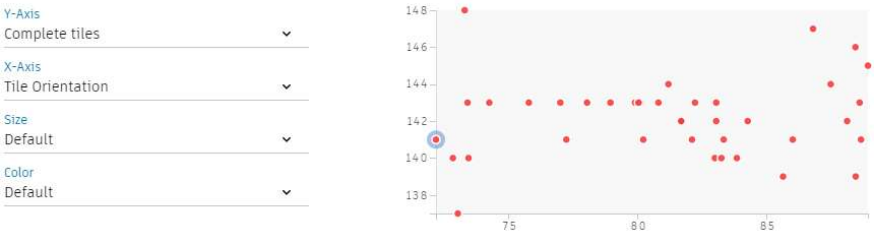


Visualising Cross Product Solutions Filtering for tile orientation of 0 degrees

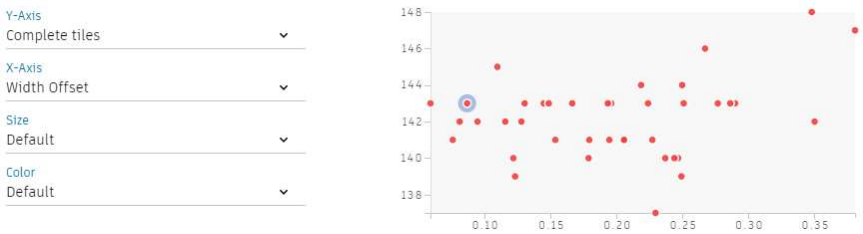
Generation Method - Like This

The “Like This” method carries out a minor variation to the noted inputs.

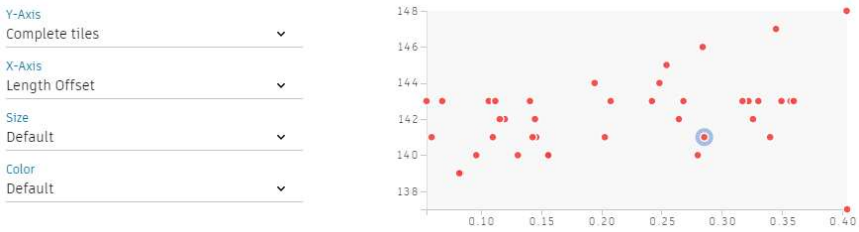




Tile Orientation is limited to 73-90 degrees



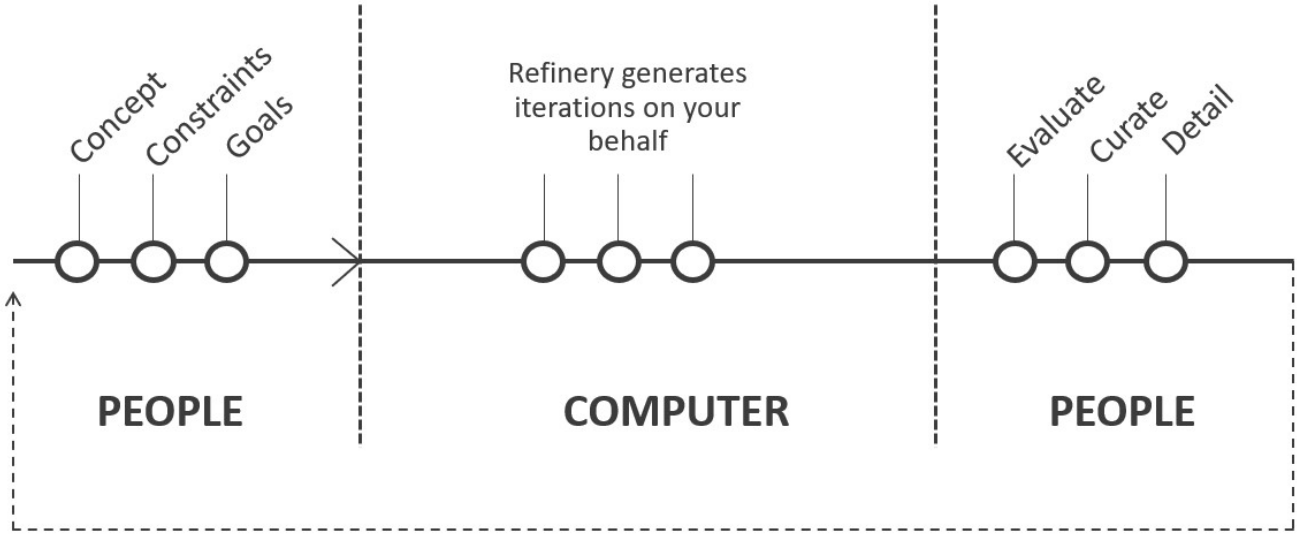
Width offset is limited to 0.05 – 0.40 degrees



Length offset is limited to 0.05 – 0.40 degrees

From the results above, it is not clear how far the “like this” option deviates from the selected inputs

How can we learn from the Generative Design Process?
In all the examples we tested it was clear that generative design is not a process which is solely achieved by the computer alone. As a simplified process we can look at the workflow below. This could be extended to separate the process to carry out “optioneering” first, to better inform the constraints and goals to speed up the “optimisation process”



Evaluating the data from the evolutionary solver engine
The ability to understand the algorithmic evolution

Generative Design Workflow Tips

1. Understand what you want to achieve
2. Define your problem – what decisions are best suited to the computer and what can the human resolve (in the tiling example, tile orientation could have been limited to 0 and 90 with possibly a few options between to confirm the gut feel to align to the longest edge)
3. Decide on ways to measure success
4. Run a series of random “optioneering” solutions
5. Evaluate the options to see if you can identify trends
6. Minimise the range of input variables
7. Run a small number of optimized solutions
8. Repeat steps 5-6
9. Run a larger number of optimized solutions
10. Repeat steps 5-6 and increase number of optimized solutions if required
11. Think about how to review the results
12. Select a range of “best” solutions
13. Try to assign a weighting to the optimized outputs
14. Refine best solutions with all stakeholders

Conclusions

- Often the unmeasurable will take precedence over optimised solutions
- Application for using optimisation - use where appropriate. Use to think outside the box but remember the steps to always evaluate and refine - tiling example was a good rest for human logic and computing power to proof assumptions
- Try to combine into a single fitness function of possible or add weighting to optimised outputs
- Evolutionary Solvers are powerful tools to be used on specific or partial problems.
- To formulate the right fitness function and the set the key variables is crucial.
- The process helps to understand the nature of the problem.
- They are slow because of the amount of options to be tested and depending on the complexity of the problem, the efficiency of the script, the platform we are using and the hardware capabilities.
- Galapagos is a robust built-in GH tool which is ideal to solve single objective optimization problems.
- Octopus is a multi-objective optimization plugin for GH which tackles more complex problems and enables user interaction and solutions exploration.
- Refinery is a multi-objective optimization beta product by Autodesk that computes Dynamo scripts faster than within the Revit / Dynamo environment (but still slower than Rhino / GH).
- Unlike its GH competitors, access to the evolutionary data is harder to access, behind which is a valuable information to identify good solutions and improve fitness functions.

Lessons Learnt / Possible Issues
Testing the Beta for Refinery didn't come without it's challenges. Here are some of the lessons learnt and issues.

- Step Value in Dynamo appears not to be respected in Refinery – We set a step value of 0.1 for the tile offsets. When the results came back in Refinery the Dynamo input updated to 15 decimal places
- Always remember to save your Dynamo Script when you want to run a new study
- When using Custom Nodes, ensure these are saved into the c:\users\<yourname>\AppData\Roaming\Dynamo\DynamoRevit\2.0\definitions
- Ensure you have updated all packages to those that run in Dynamo 2+
- How a script is built can have a significant effect on the speed to analyse within Refinery

Useful resources
https://github.com/DynamoDS/RefineryPrimer

https://dynamobim.org/london-hackathon-stealthy-roofscapes/

https://www.keanw.com/2019/04/revisiting-mars-for-au-london-2019.html

https://www.grasshopper3d.com/group/octopus

https://www.grasshopper3d.com/group/galapagos

https://discourse.mcneel.com/t/wallacei-evolutionary-and-multi-objective-optimization-engine-integration-to-revit-via-rhino-inside/80942