MORPHO
DESIGN EXPLORER



First Edition - October 2026

TABLE OF CONTENTS

3 Introduction
4 Installation Guide
5 Component Overview
6 Order(s) of Use

8 GENE GENERATOR
9 Algorithm Parameters
9 Initiate
10 Fitness Filters
10 Iteration Limit
11 Intervals
12 Directory
12 Genes
13 Genes With Names
14 Parents
15 PARAMETERS
16 FITNESS FILTERS
17 Filter Logic
18 DIRECTORY
19 DATA AGGREGATOR
20 Outputs
21 Images
21 Files
22 Aggregated Data
23 IMAGE CAPTURE
24 SAVE TO POPULATION
25 Directory
26 Enabled
27 Outputs
28 VISUALIZER
28 Launching the Visualizer
28 Exploring Design Solutions
30 Filtering Design Solutions
30 Visualizing the Design Space
30 2D Scatter Plots and Pareto Fronts
32 3D Scatter Plots
33 Parallel Coordinates Plot
34 Exporting Visualization Results
34 Using the Visualizer to Support Design Decisions
35 Copyright, Referncing, Aknowledgement, Contact

INTRODUCTION

Morpho is a multi-objective design exploration tool aimed at aiding designers in making informed decisions during the early design stages. It employs the Non-Destructive Dynamic Population Genetic Algorithm (NDDP-GA) to search for suitable generative design solutions. Morpho is designed to assist designers in considering both performance-based quantitative design objectives and qualitative design goals, enabling a comprehensive exploration of the design space and facilitating well-rounded decision-making.

The name "Morpho" is inspired by the Morpho butterfly species, which is renowned for its iridescent blue wings. These wings are not only visually striking but also structurally complex, featuring microscopic scales that reflect light in a unique and dynamic way. This remarkable characteristic of the Morpho butterfly mirrors the essence of the Morpho design tool. Just as the butterfly's wings display an intricate blend of beauty and functionality, the Morpho tool aims to generate design solutions that are both aesthetically pleasing and technically sound.

Morpho continues to evolve into an open-access, comprehensive design-aid tool. It offers assistance in the configuration processing of spatial forms, supports the exploration of generative design solutions, and facilitates continuous interaction between the search engine and designers. Additionally, it aims to include as diverse a range of design objectives as possible. This holistic approach ensures that designers can explore a wide array of possibilities, much like the Morpho butterfly, which adapts and thrives in varied environments. By embodying the complexity and beauty of the Morpho butterfly, the Morpho design tool aspires to inspire and empower designers to create innovative and effective design solutions.

INSTALLATION GUIDE

1. Download the Morpho plugin .zip file at https://www.food4rhino.com/en/app/morpho
2. Locate the Grasshopper components folder in C:\Users\USERNAME\AppData\Roaming\Grasshopper\Libraries by opening Grasshopper, clicking "File" in the top-left corner of the screen, clicking "Special Folders", and then clicking "Components Folder".

On a Windows machine, you can also locate your Components Folder by typing "%appdata%" into the windows search bar and pressing ENTER to open the "Roaming" folder. From there, you can open the "Grasshopper" folder, and then the "Libraries" folder.

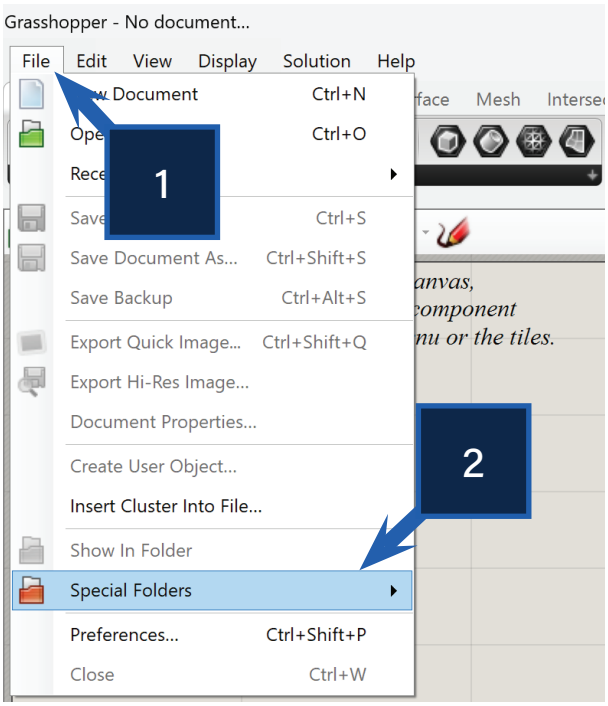


Figure 1

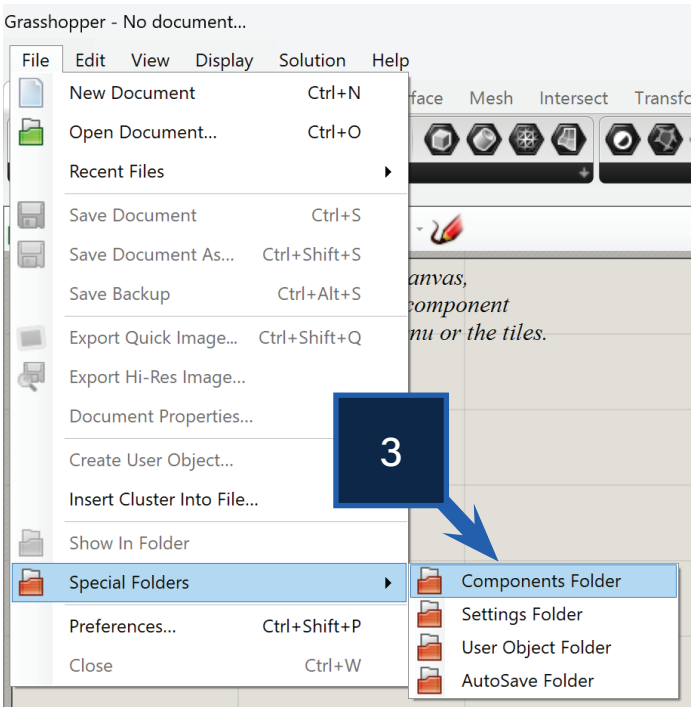


Figure 2

3. Right Click on the downloaded Morpho .zip file, choose Properties, and "Unblock" the folder.
4. Extract the content of the .zip file.
5. Copy the extracted folder into the Grasshopper Components Folder.
6. Close Grasshopper and Rhino if they are open. The next time Grasshopper is opened, the Morpho plugin will be installed. Enjoy!

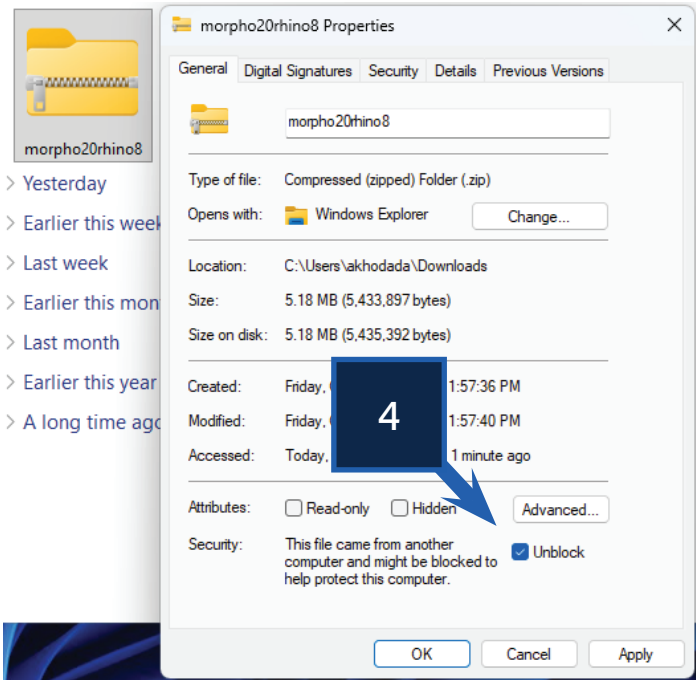
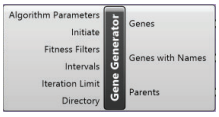
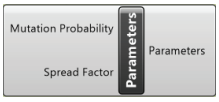
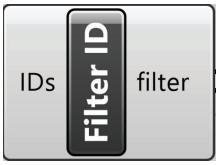
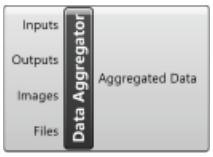
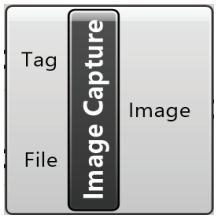
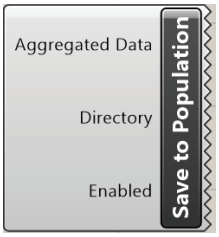
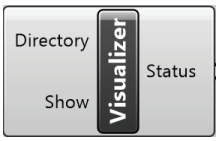


Figure 3

COMPONENT OVERVIEW

ICON	NAME	DISPLAY	FUNCTION
	GENE GENERATOR		This is the core component where the NDDP-GA algorithm is implemented. It allows designers to either generate an initial population of random solutions or proceed with a fitness-based generation strategy. The output is a list of gene values corresponding to the variable design parameters. The component also accepts algorithm parameters, such as mutation probability and spread factor.
	ALGORITHM PARAMETERS		This component allows designers to define the mutation rate and spread factor, which control solution diversity and convergence behavior.
	FILTER PALETTE	 	Filter palette includes a set of logical filter components — FilterAND, FilterOR, Less than, Greater than, and etc.—that allow the designer to define one or multiple fitness criteria. It is also possible to filter solutions directly by their ID. These criteria are passed to the Gene Generator to guide systematic solution generation.
	DIRECTORY		This component creates the database at the given address on the local machine.
	DATA AGGREGATOR		This component collects all relevant input values, performance outputs, images, and other associated files for each generated design solution.
	IMAGE CAPTURE		This component captures views from multiple Rhino viewports, enabling the designer to store a variety of visual representations for each design alternative.
	SAVE TO POPULATION		This component writes all collected data to a structured database and saves associated media (images, analysis files, etc.) in a local directory. Each design solution is tagged with a unique numeric ID to ensure clear association between data records and visuals.
	VISUALIZER		This component allows studying the population of generated solutions dynamically by filtering, sorting, providing an array of solutions linking their visuals and numeric values, and offering 2D or 3D distribution graphs and parallel coordinate plots.

ORDER(S) OF USE

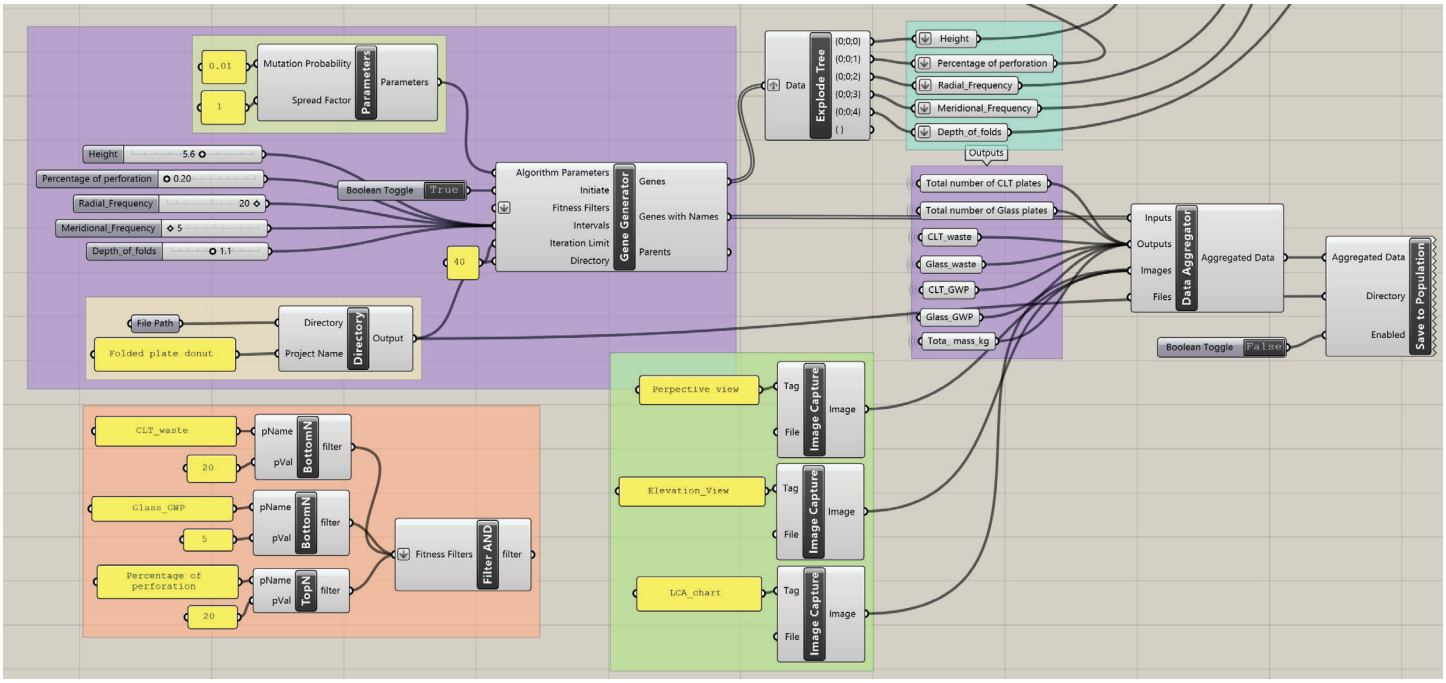


Figure 4: Overall order of using "Gene Generator", "Data Aggregator", and "Save to Population" to carry out design exploration.

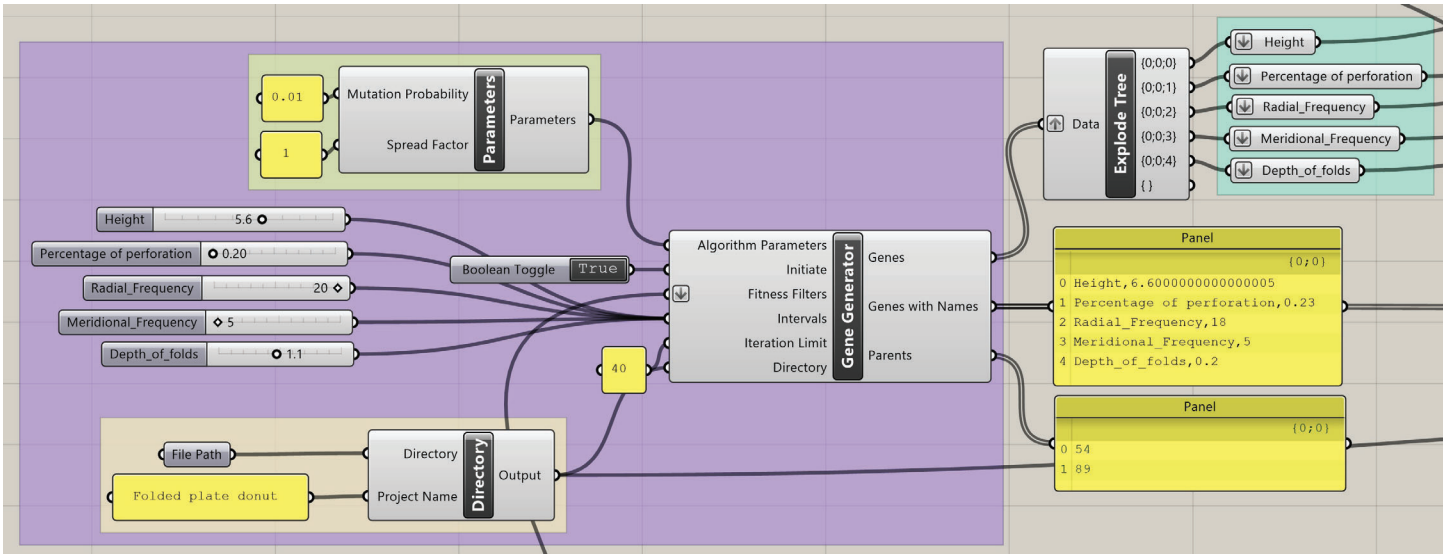


Figure 5: "Gene Generator" component order of use.

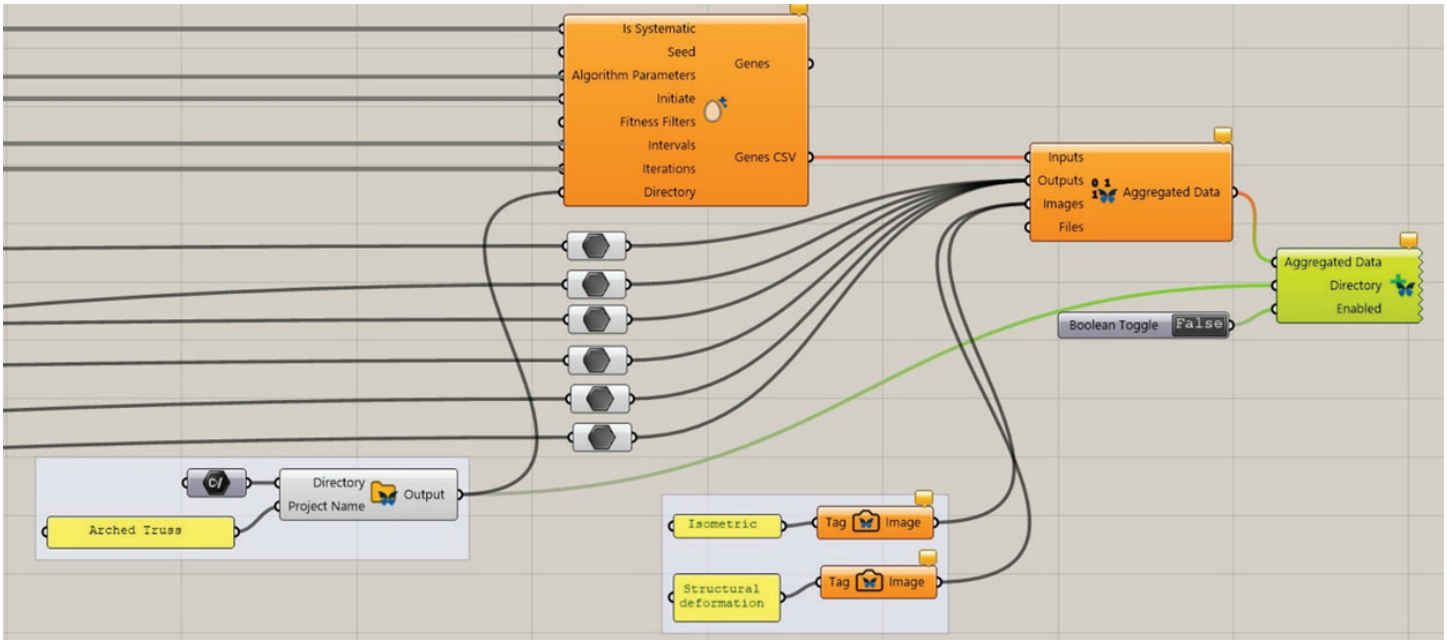


Figure 6: "Data Aggregator" and "Save to Population" components and their order(s) of use.

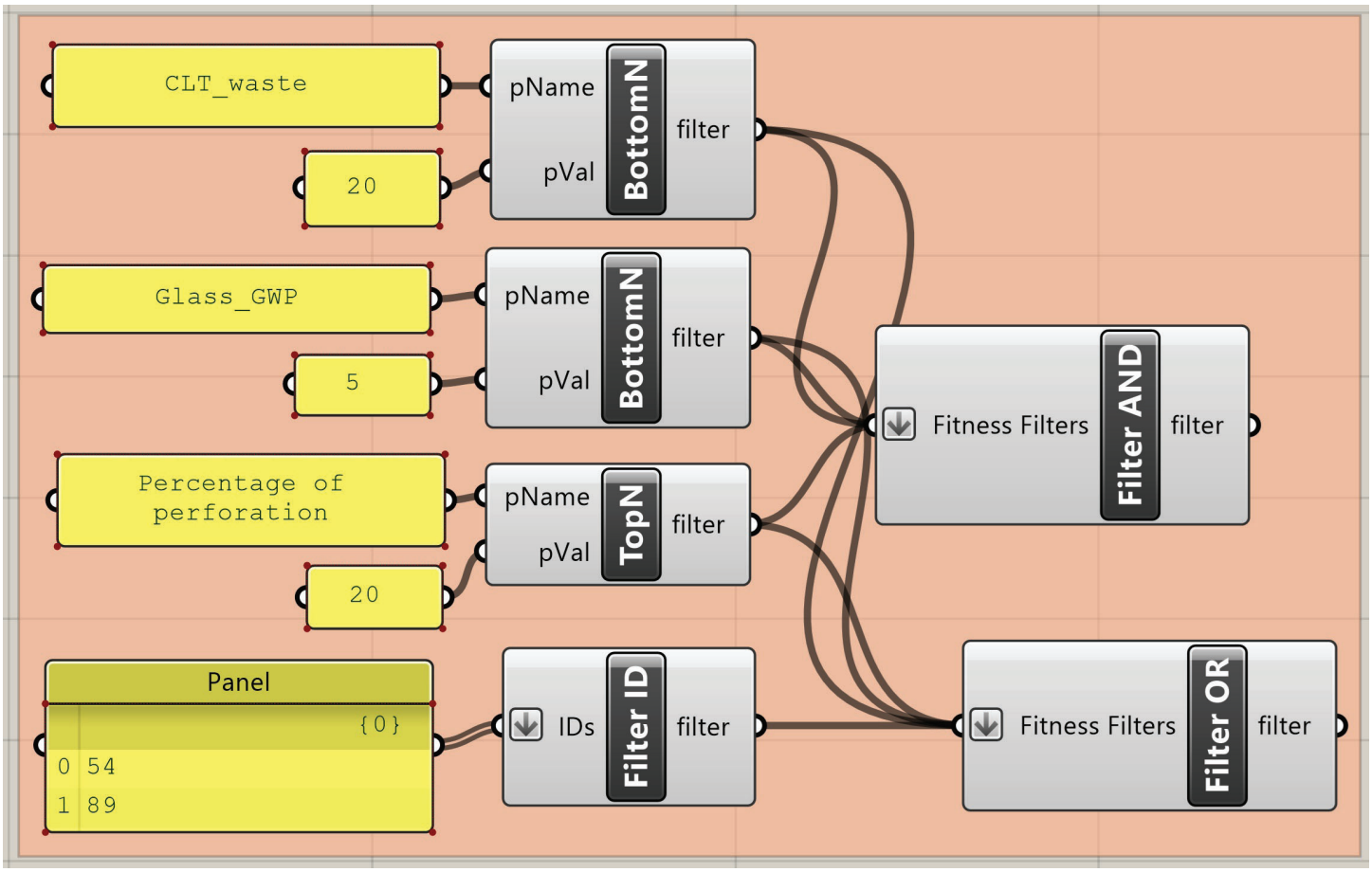
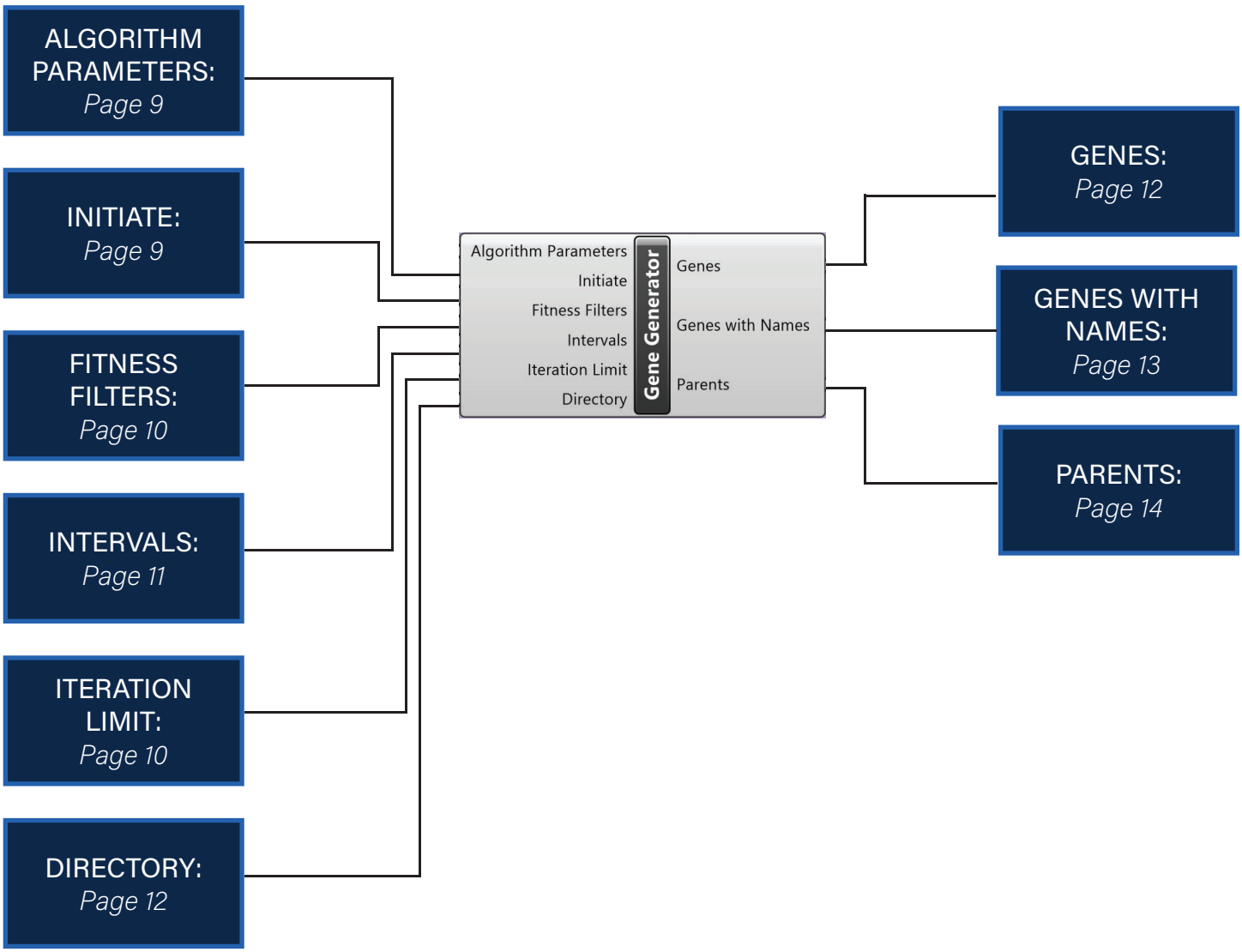


Figure 7: Different components of the "Filter Palette" and their order of use, allowing designers to define the exploration's fitness function.

GENE GENERATOR



ALGORITHM PARAMETERS

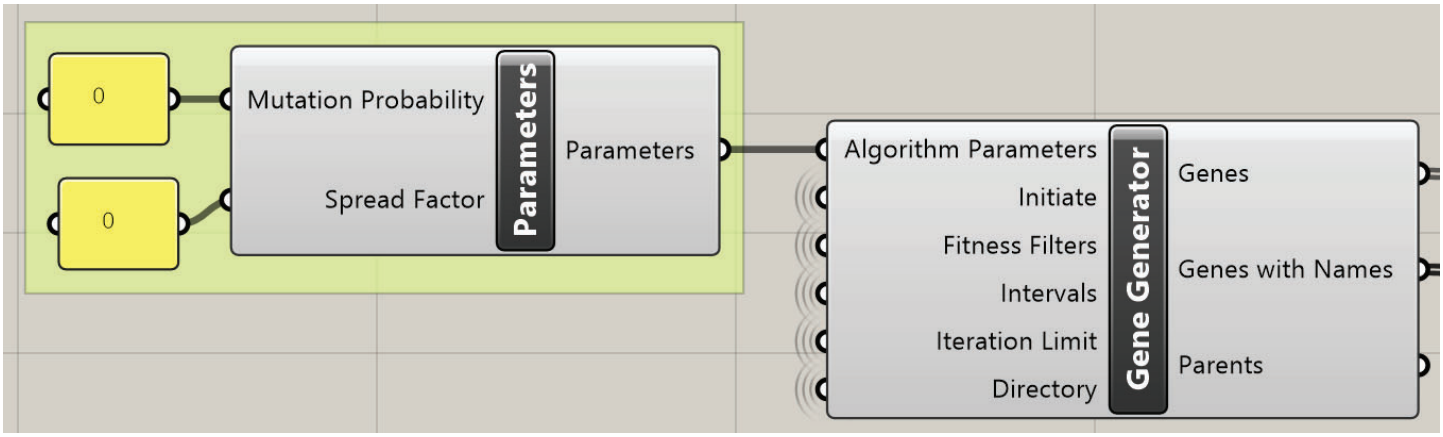


Figure 8

The "Algorithm Parameters" input requires a "Parameters" component, also from the Morpho plugin, which is discussed further on Page 15.
The "Algorithm Parameters" input determines the degree of randomness or variability from one generation of individuals to the next.

INITIATE

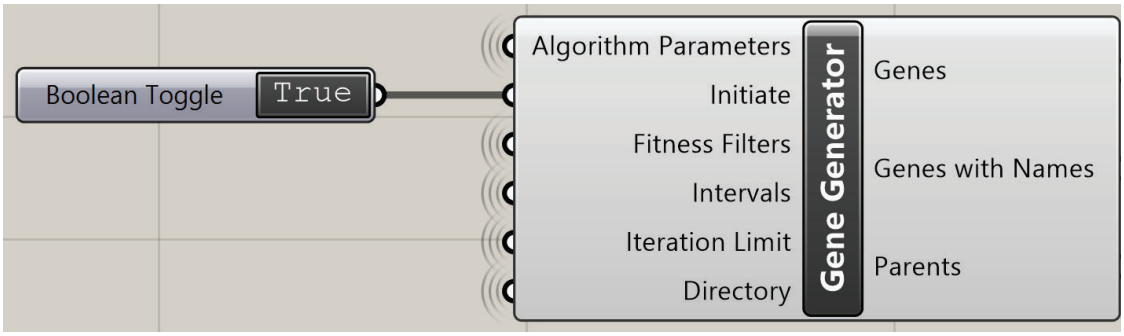


Figure 9

The "Initiate" input requires a "Boolean Toggle" component to function properly.
When set to "True", the gene generator will run and return values for each interval provided.
When set to "False", the gene generator will not run, and is effectively "paused".
When changing inputs for the gene generator, such as new algorithm parameters or fitness filters, it is best to set this toggle to "False" such that the generation does not need to re-run for each change made.
Toggling to "False" and then back to "True" will reset the gene generator, resulting in a new set of values.

FITNESS FILTERS

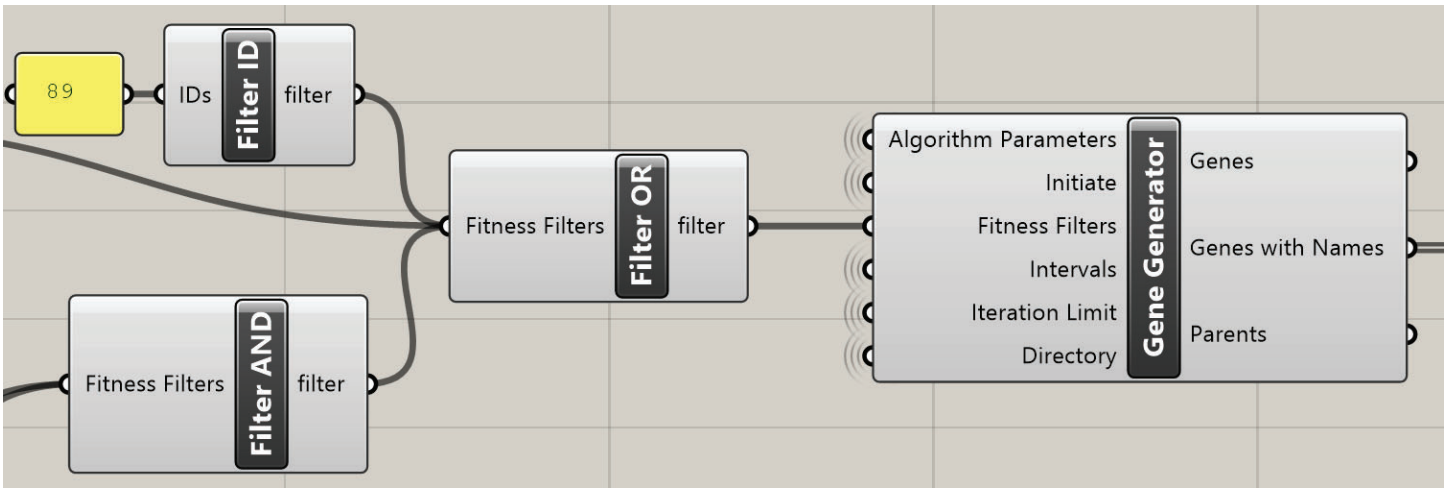


Figure 10

The "Fitness Filters" input requires a filter logic component, either "FilterAND" or "FilterOR", both from the Morpho plugin, in order to function properly. Fitness filters, and the logic gates that combine them, are discussed further on Page 16.
Fitness filters are used to determine which genes are desirable, and which are not. They are used to filter out individuals that do not meet criteria set by the user, and are critical for determining which individuals will be used to parent the next generation.

ITERATION LIMIT

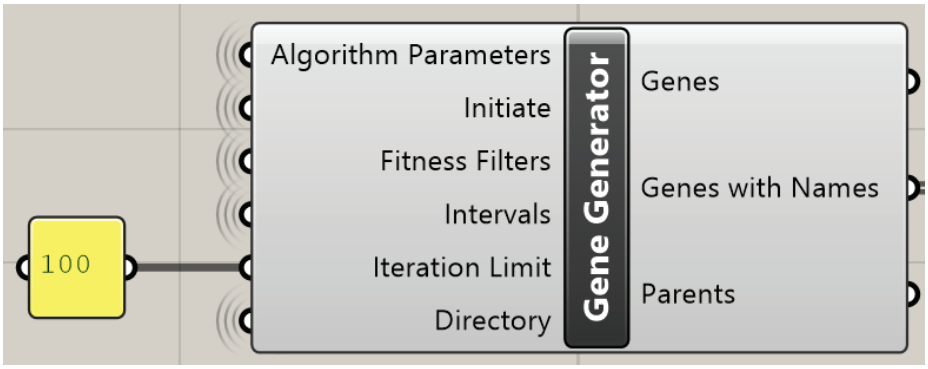


Figure 11

The "Iteration Limit" input requires a numerical input, in this case from a "Panel" component, to function properly.
The iteration limit determines the number of cycles that the engine runs to generate solutions. In the case above in Figure 11, 1the engine runs for 100 cycles.

INTERVALS

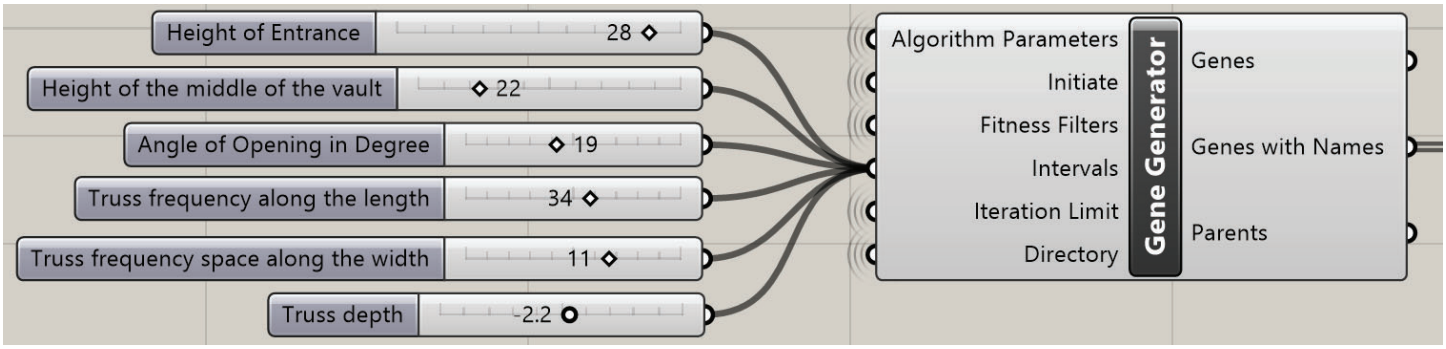


Figure 12

The "Intervals" input requires one or more "Number Slider" components to function properly. The minimum and maximum values of each slider represent the boundaries of the interval within which the gene generator will create a value.
Additionally, the number slider(s) being used as inputs must be properly named to align with the name of parameters provided to the "Fitness Filter" component(s) (Page 16). This name is also what will be listed in the final database. Thus, it is crucial that it is clear and relevant. The bounds of each slider must also be changed such that each output will yield a relevant number. See Figure 13 below for an example of a well-formatted number slider.

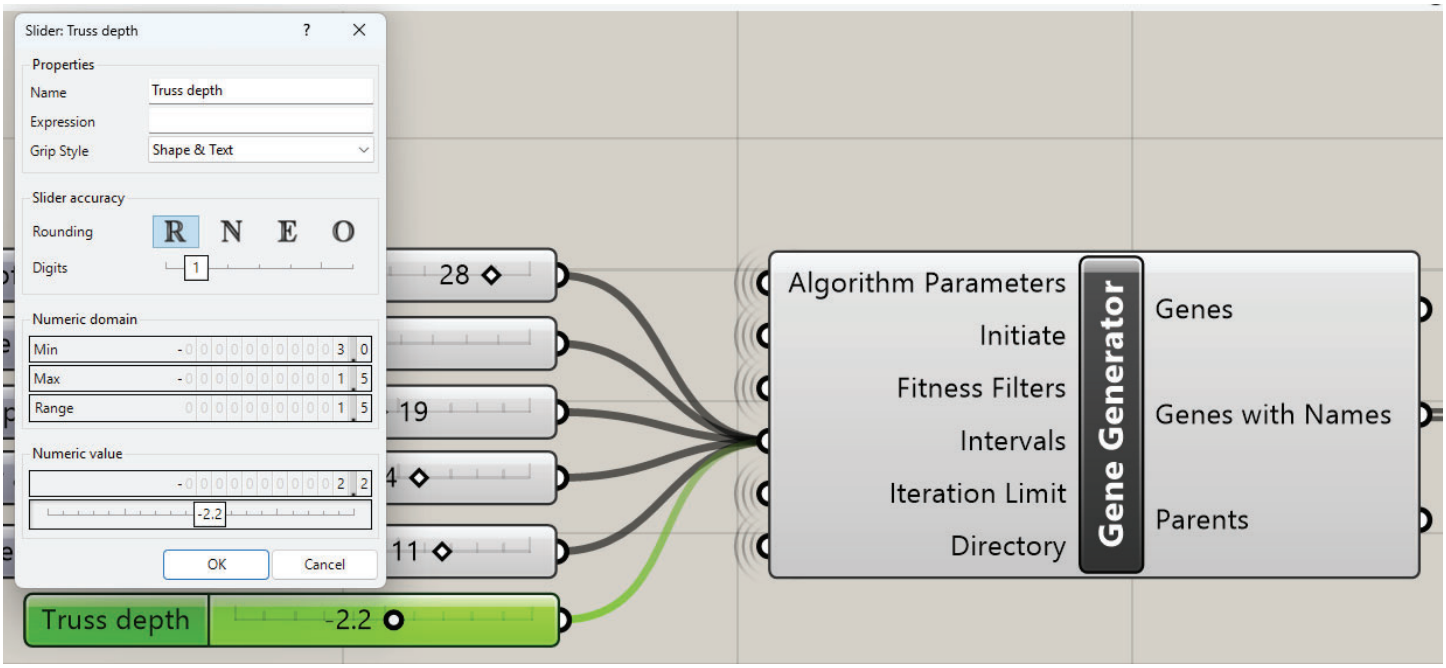


Figure 13

DIRECTORY

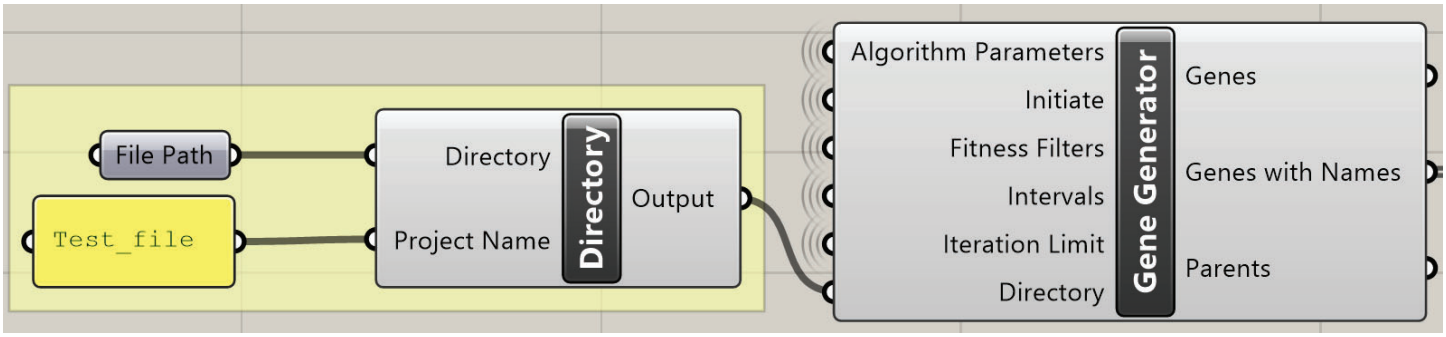


Figure 12

The "Directory" input requires a "Directory" component, also from the Morpho plugin, which is discussed further on Page 28.
This component determines the location on the user's machine to which the gene generator's database will be saved. Typically, it will also be the location to which the "Save to Population" component (Page 24) stores the database.

GENES

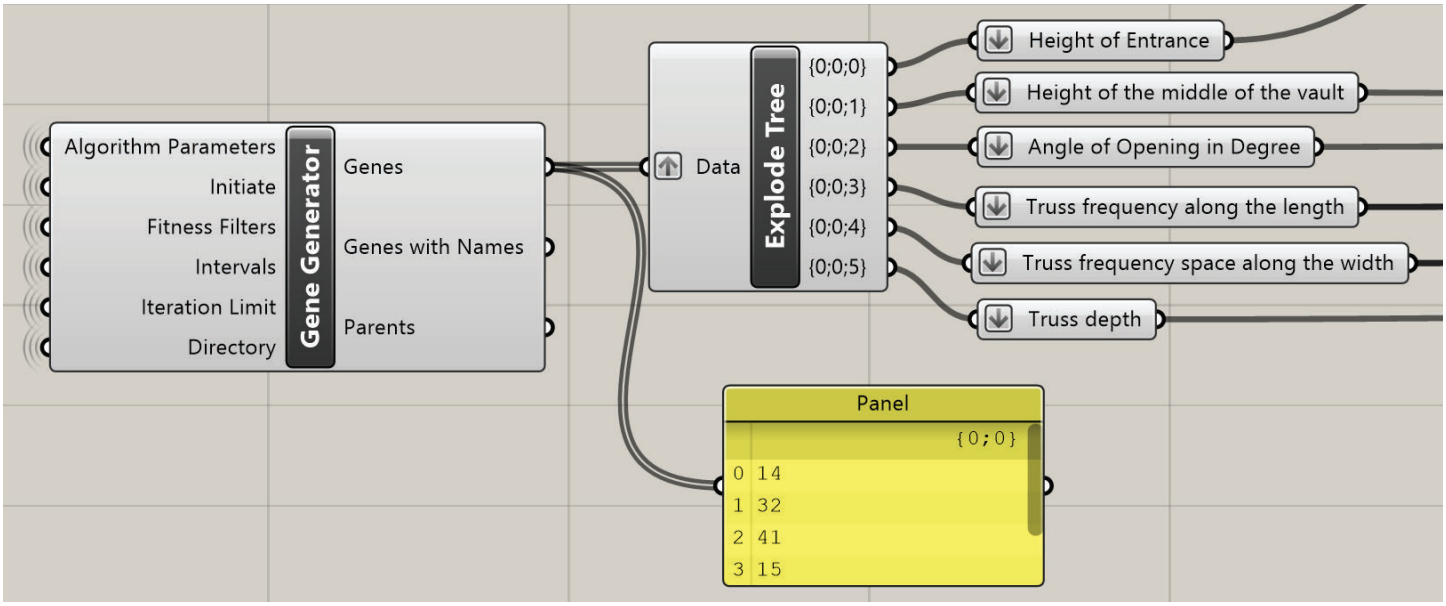


Figure 15

The "Genes" output of the gene generator represents the values created within the intervals provided by the "Intervals" component. They are the outputs that will be used to create the geometry of individuals. The "Genes" output is by default a flat list of generated values that must be "Grafted" when connecting to an "Explode Tree" component.

The order of values is determined by the order in which the number sliders were attached to the “Intervals” input. Otherwise, a panel component can be used to compare the “Genes” output with the “Genes with Names” output (Page 13) to determine which value is attached to which name.
For example, in Figure 15, “Height of Entrance” was the first number slider interval attached, “Height of the middle of the vault” the second, so on and so forth. In this case, the user has used “Data” components and renamed each of them in order to make it clear which output is used for what. Additionally, after exploding, data can be re-flattened to simplify organization.

GENES WITH NAMES

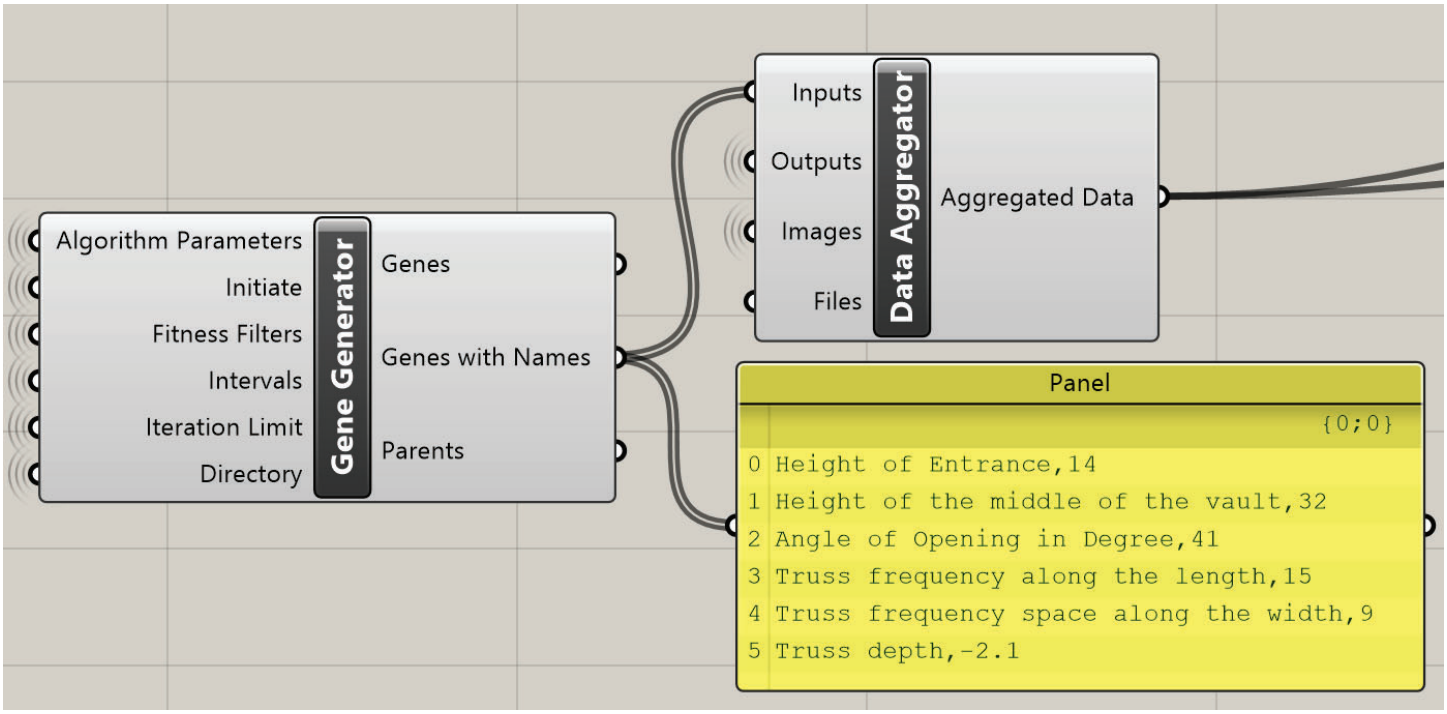


Figure 16

The “Genes with Names” output is used as the “Inputs” of the “Data Aggregator” component. “Data Aggregator”, also from the Morpho plugin, is discussed further on Pages 19.
The “Genes with Names” output is a list of names provided by the names of number sliders attached to the “Intervals” input, alongside the values attached to each. It is used to generate the final database of individuals and their associated values.

PARENTS

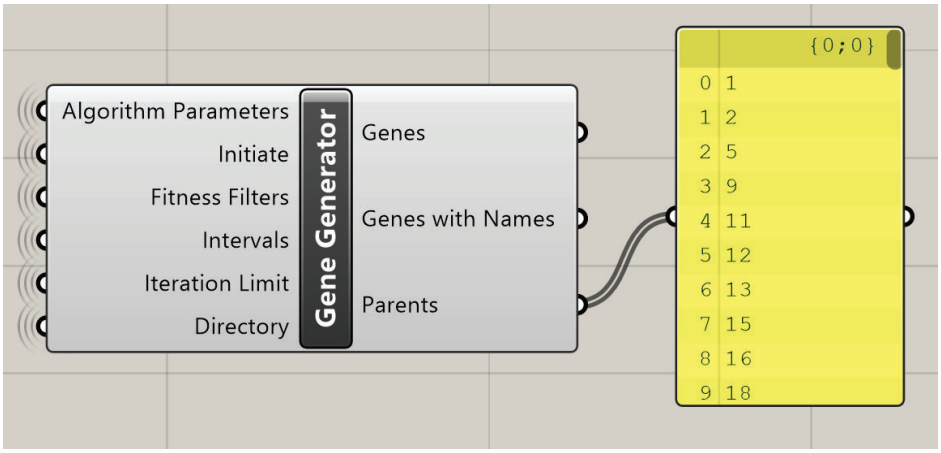


Figure 17

The “Parents” output gives a list of qualified solutions (well-performing parents) that exist in the pool of good solutions. Two solutions from this pool will be randomly paired to breed the new offspring.

PARAMETERS

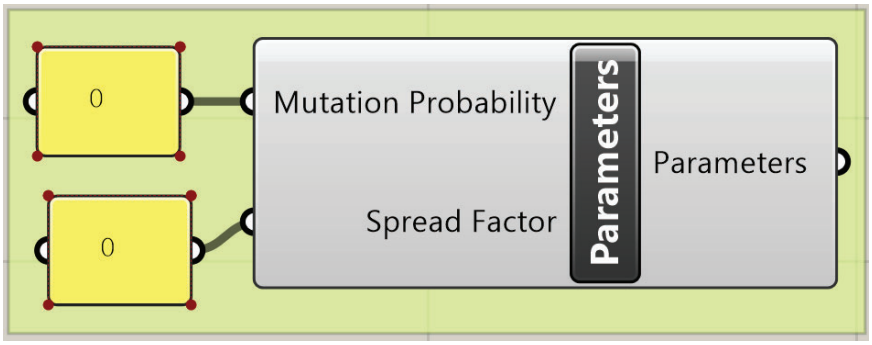


Figure 18

As mentioned on Page 9, the “Algorithm Parameters” input of the gene generator requires a “Parameters” component, pictured above in Figure 18.

The “Parameters” component requires two basic inputs. The first, “Mutation Probability”, requires a percentage input expressed as a number between 0 (0%) and 1 (100%). This determines the percent chance that a genetic mutation will occur from one generation to the next, resulting in increased variability in outputs.
The second input, “Spread Factor”, also requires a numerical input, typically between 0 and 1. The Spread Factor is defined as the ratio of the absolute difference between the offspring values to the absolute difference between the parent values.
When the spread factor is 0, it represents a precise contracting midpoint crossover where the two generated offspring are structurally identical and lie exactly at the average of the two parent solutions. A spread factor between 0 and 1 indicates an increasingly nonuniform distribution. A spread value of 1 does not necessarily mean all solutions are clustered at one point. It indicates poor distribution according to the metric.

FITNESS FILTERS

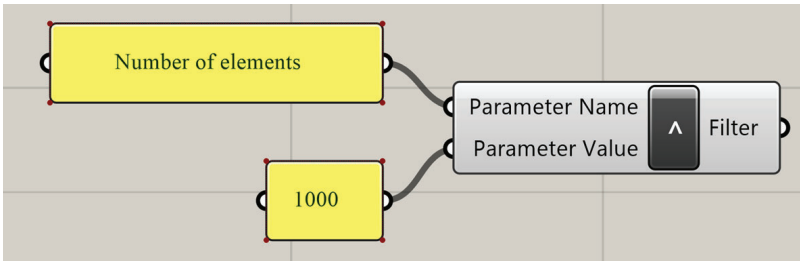


Figure 19

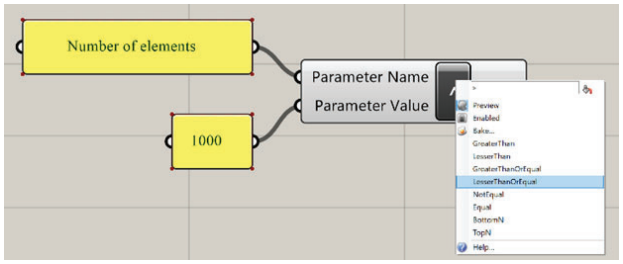


Figure 20

As mentioned on Page 10, fitness filters are used to determine which genetic traits are desirable, and which should be excluded from the pool of cood solutions. The "Filter" component, shown above, requires a few basic inputs. The first, "Parameter Name", requires a text input from a "Panel" component containing the exact name of the output value that is being tested for. In the case of above, that's the total number of elements in the geometry being created. The names of outputs is determined by the "Outputs" input of the "Data Aggregator" component, discussed on Page 19.
The second input, "Parameter Value", requires a numerical input that the named parameter will be tested against. The example below shows how:

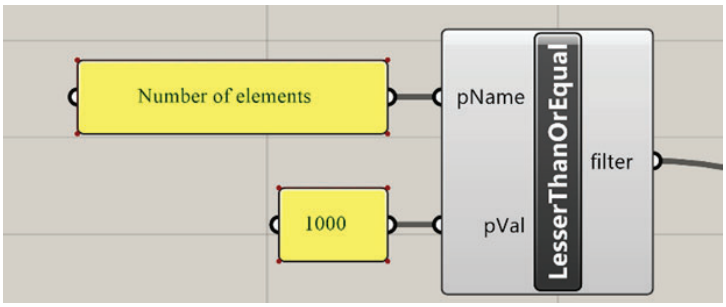


Figure 21

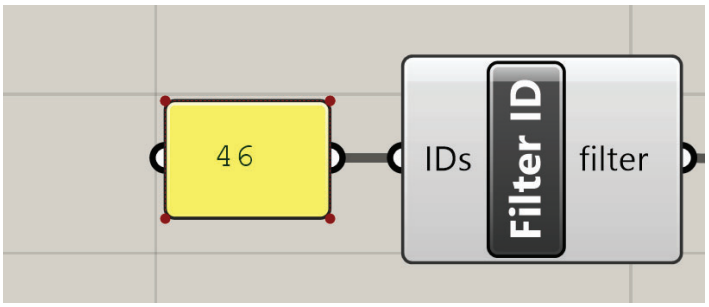


Figure 22

By right-clicking on the "Filter" component (Figure 20), the user can set the logic by which the parameter is being tested. When doing so, the component name will change from ">" to the selected logic (Figure 21). In this case, this filter will ensure the gene generator only outputs those individuals with a "Number of elements" "Lesser Than Or Equal" to "1000".

Filter ID (Gifure 22) allows selecting a solutions or a list of desirable solutions by their ID number and include them in the list of well-performing parents. This feature lets the designer to select the solutions based on visual attribbutes or subjective goals that are not quantifiable.

FILTER LOGIC

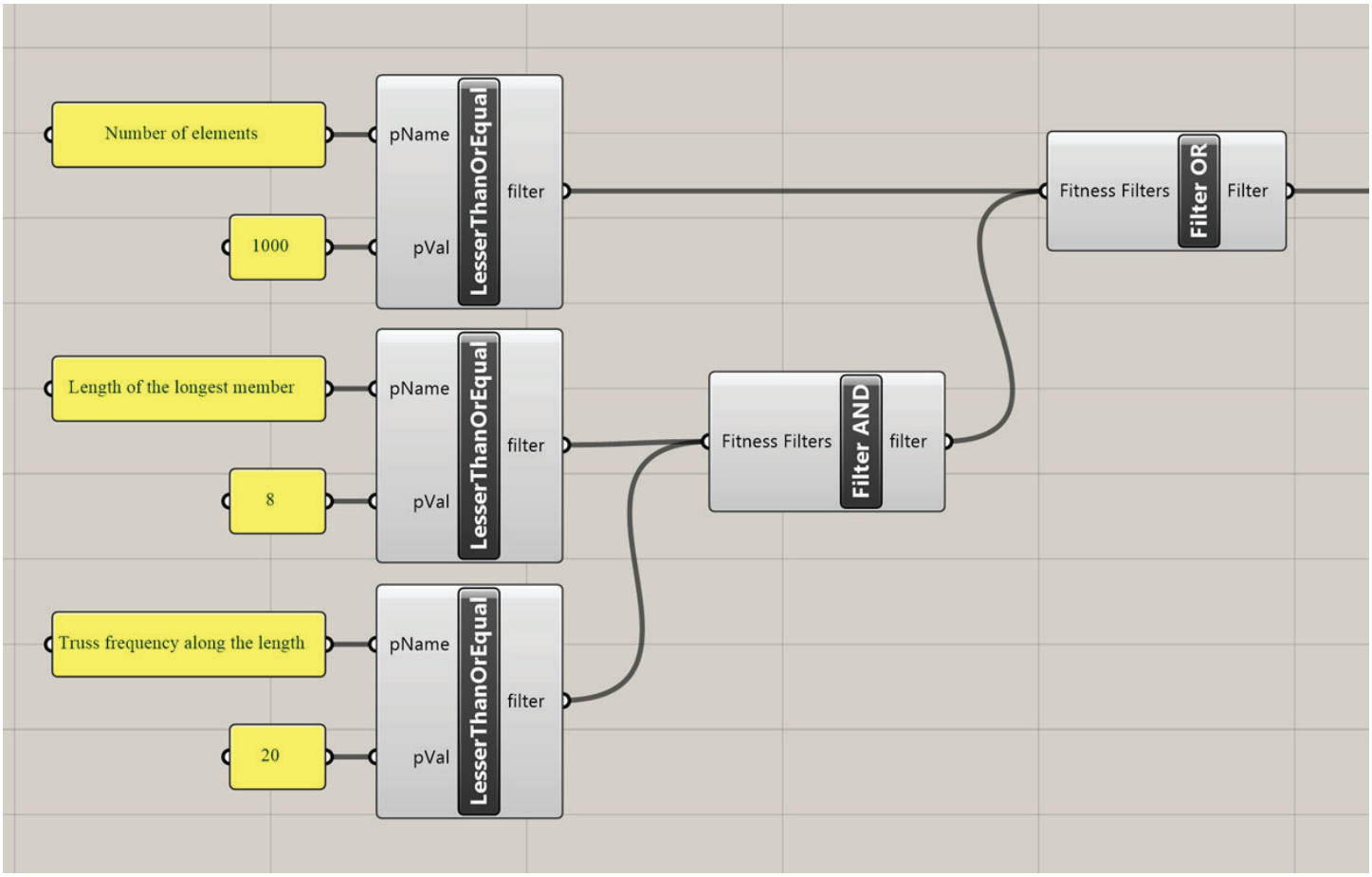


Figure 23

"Filter" components can be combined via logic gates using the "FilterAND" and "FilterOR" components. Multiple "Filter" components cannot be attached directly to the "Gene Generator" component's "Fitness Filters" input; it is best they first flow through a series of one or more logic gates.

In the example above (Figure 23), when given this series of filters, the gene generator will only output those individuals who have:

(Number of elements lesser than or equal to 1000)

OR

(Length of the longest member lesser than or equal to 8 AND Truss frequency along the length lesser than or equal to 20).

DIRECTORY

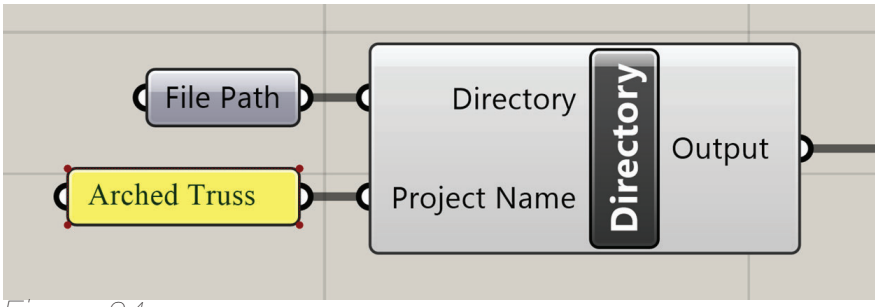


Figure 24

As mentioned on Page 12, the "Directory" component determines the location on the user's machine to which the gene generator and "Save to Population" components' databases will be saved.

The Directory component requires two inputs. The first, "Directory", requires a "File Path" component, which is native to Grasshopper. By right-clicking on a "File Path" component, users can select a directory that file path will represent. See right (Figure 25):

The second input, "Project Name", requires a text input, typically from a "Panel" component. This will simply determine what name project files will be given. In the case of above (Figure 24), the project will be named "Arched Truss" when saved to the directory set by the user.

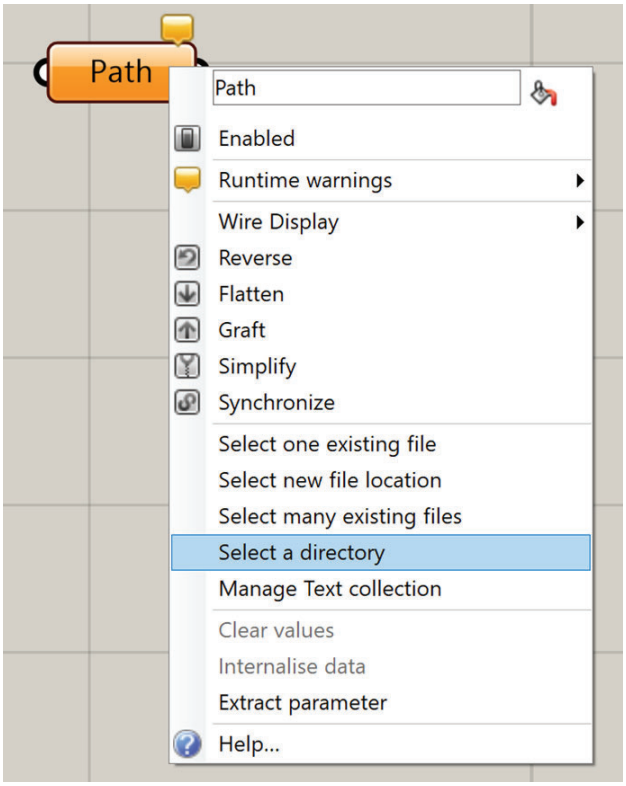
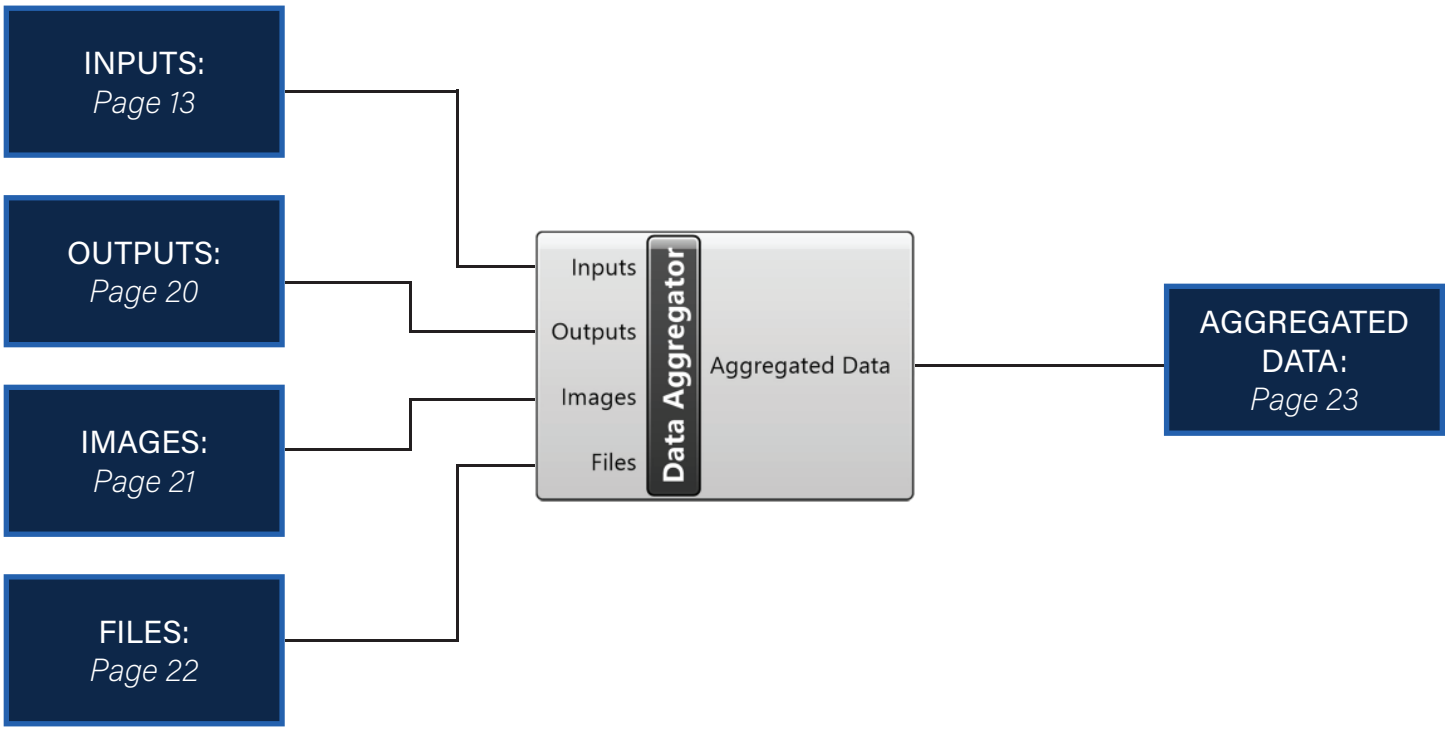


Figure 25

DATA AGGREGATOR



OUTPUTS

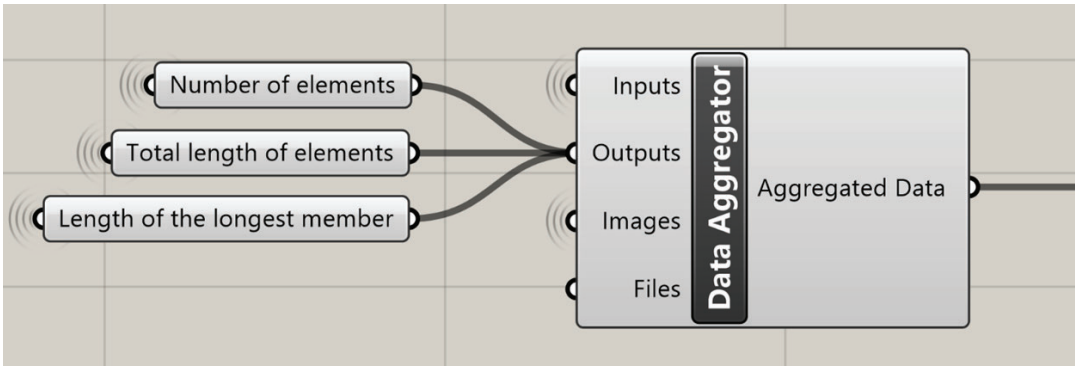


Figure 26

The "Outputs" input of the "Data Aggregator" component represents the geometry that is being tested. After the gene generator's outputs are used to create a series of individuals, data from the modelling script is pulled out and used here.

In the case of above (Figure 26), the user is testing for the number of elements in their geometry, the total length of the elements, and the length of the longest member. How that data is extracted is dependent on the script that is used for modelling. In any case, it represents the data that is being tested for, and on Pages 16 and 30, it is shown that these data points can be used to filter out desirable or undesirable individuals using "Filter" components and logic gates.
These inputs are always numerical values, and most importantly, as shown above, they should be first sent through "Data" components, which are native to Grasshopper, and renamed to their associated value. This is important, as it is via the name of these "Data" components that the "Filter" components read and filter data, and it is also this name that will appear in the outputted database.
Data components can be renamed via right-clicking, as shown to the right (Figure 27).

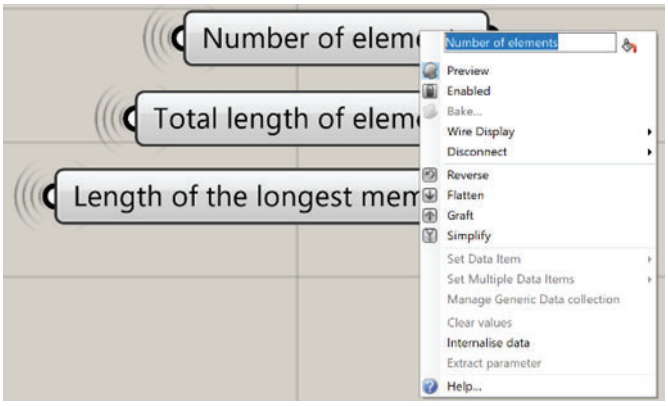


Figure 27

IMAGES

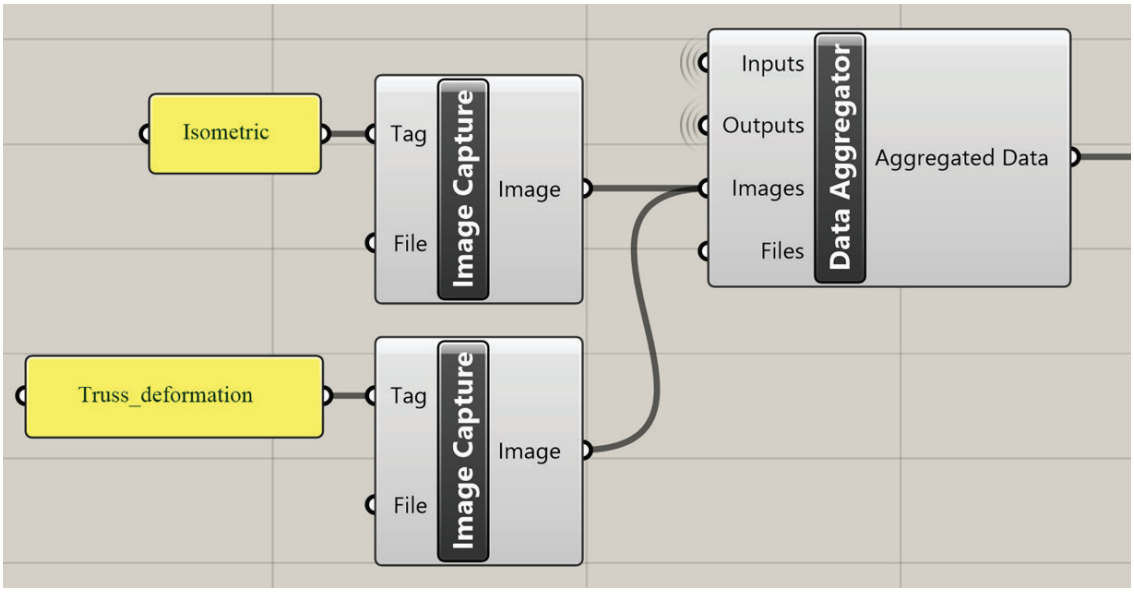


Figure 28

The "Images" input of the "Data Aggregator" requires one or more "Image Capture" components, also from the Morpho plugin, which is discussed further on Page 23. The "Image Capture" component will capture screenshots of any Rhino viewport that the user desires for each individual that is generated by the gene generator component.
If the user does not desire or require images of the individuals generated, the "Images" input can be left empty, though this is not usual, as without these the user will only have numerical data points to represent each individual.

FILES

The "Files" input of the data aggregator requires one or more "File Path" components, which are native to Grasshopper, or it can be left empty.
Here, the "Data Aggregator" component will collect any existing files on the user's machine and assign each to its corresponding individual. For example, if the user had used another software to perform lighting analysis or analysis of thermal performance, those files can be selected here, attached to the generated individuals, and slotted into the same database created by the Morpho components.
To select multiple files, the user must right-click on the "File Path" component and "Select Many Existing Files", as shown to the right (Figure 30):

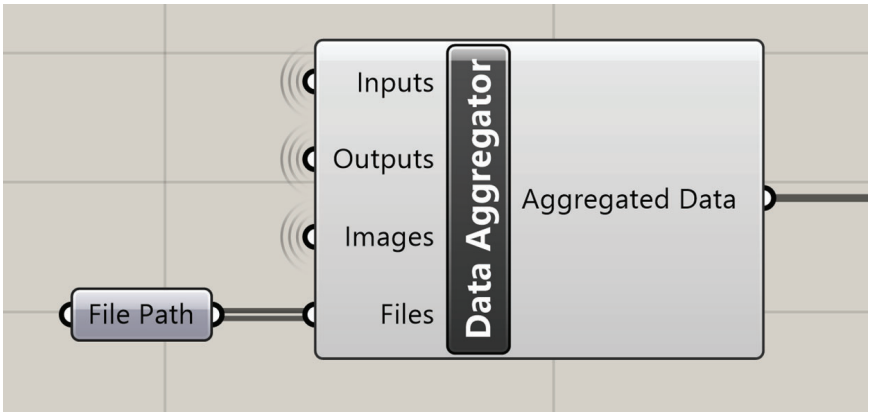


Figure 29

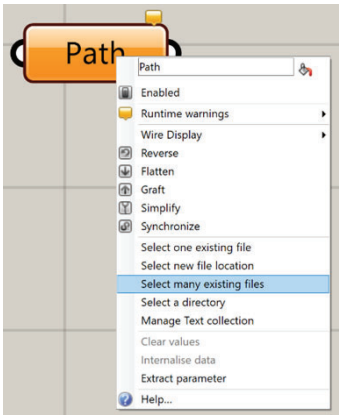


Figure 30

AGGREGATED DATA

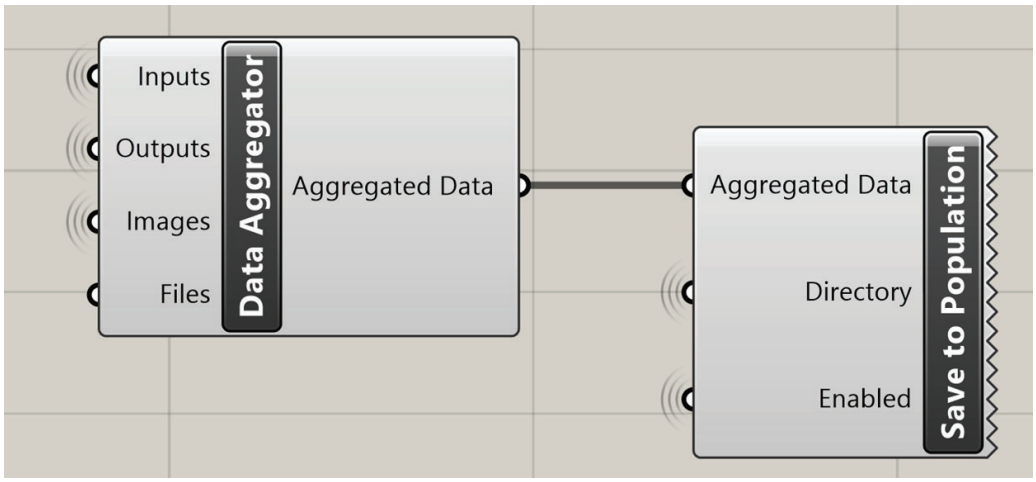


Figure 31

The "Aggregated Data" output of the "Data Aggregator" component is a string of text, including all concatenated data points that will be recorded for each individual. This includes every data interval set by the gene generator's "Intervals" input, every output set by the data aggregator's "Outputs" input, every image to be taken by the "Image Capture" component(s), and every existing file set by the data aggregator's "Files" input.

The "Aggregated Data" output is used exclusively as the "Aggregated Data" input of the "Save to Population" component, also from the Morpho plugin, which is discussed further on Pages 24-27.

IMAGE CAPTURE

The “Image Capture” component, mentioned on page 21, is used to capture images of generated individuals from any open Rhino viewport.

It requires two inputs. The first, “Tag”, is a text input, typically a “Panel” component, that is used to title each captured image. In the example to the right (Figure 32), captured images will be titled 1, 2, 3, etc. and placed within a folder on the user’s machine titled “Isometric”.

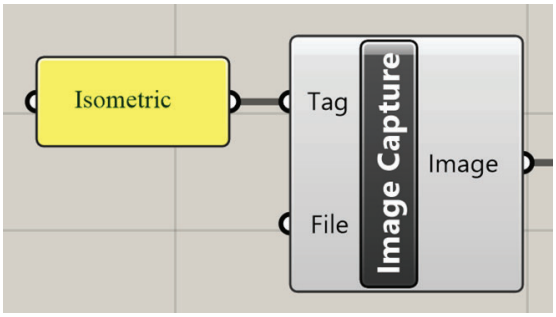


Figure 32

The second input, “File”, is used to instead pull from existing image files, using a method identical to that of the Data Aggregator’s “Files” input, discussed on Page 21. The “File” input can also be left empty – It is typically used only when the user already has images saved on their machine of each individual.

The user can select from which Rhino viewport an image will be taken by right-clicking on the Image Capture component and selecting it from the “Select Viewport” list as seen to the right in Figure 33. Currently, only four different viewports can be captured at a time.

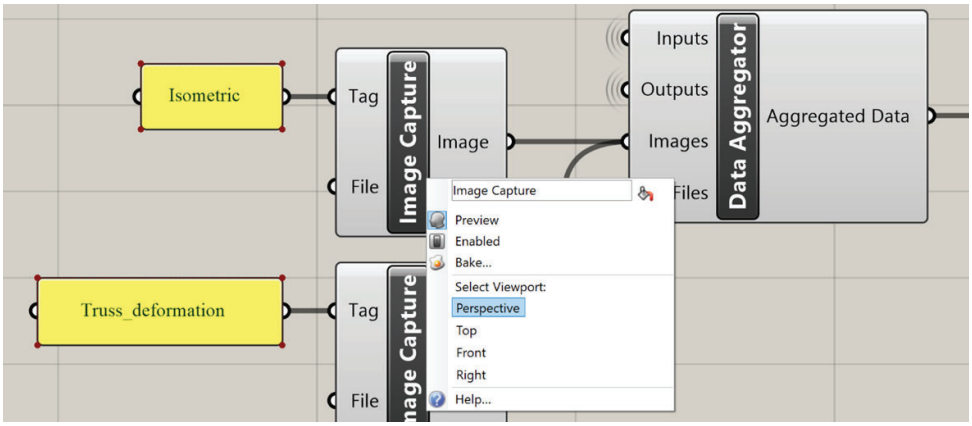
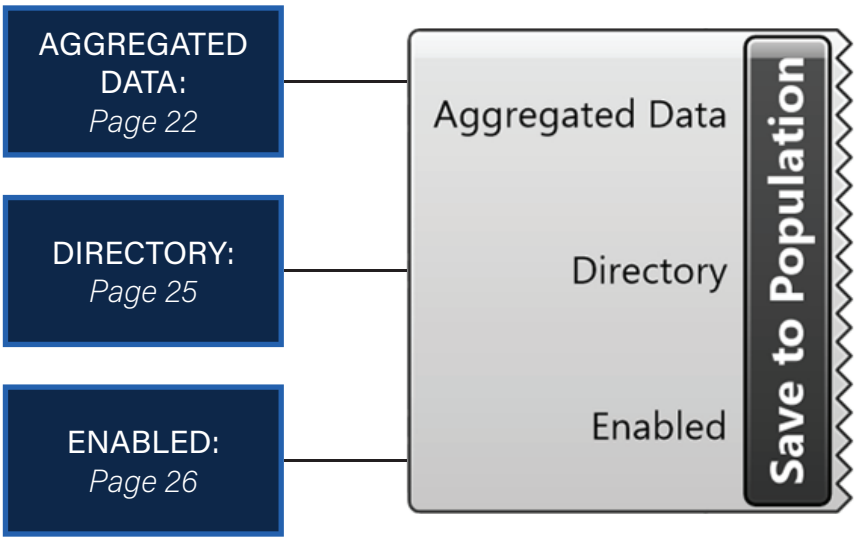


Figure 33

SAVE TO POPULATION



DIRECTORY

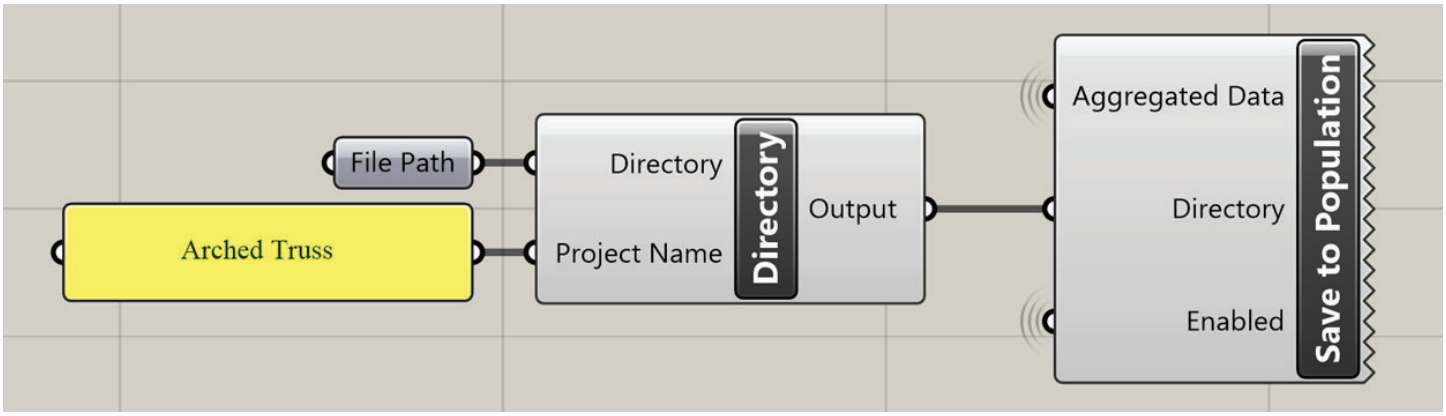


Figure 34

The "Directory" input of the "Save to Population" component requires a "Directory" component, also from the Morpho plugin, which is discussed further on Page 18.
Here, the "Directory" component determines where on the user's machine the structured database will be saved. This will include a .csv file containing all numerical data collected for each individual, as well as all associated images and other media.
As mentioned on page 12, the same directory component that was used for the "Gene Generator" component can also be used here, ensuring all databases are saved to the same location.

ENABLED

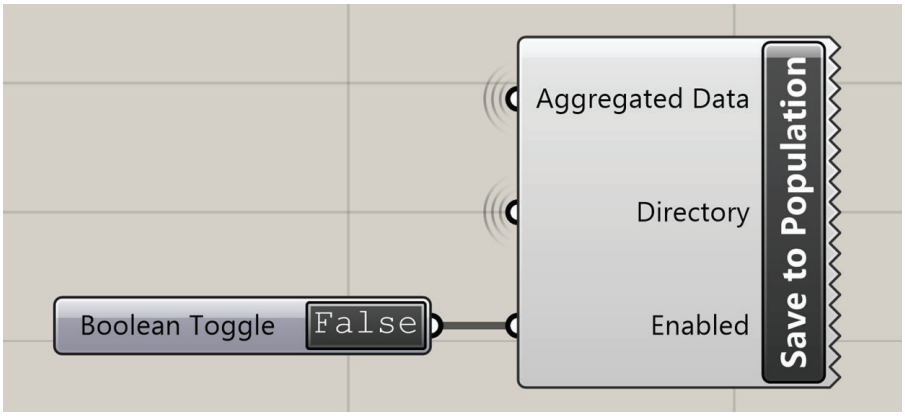


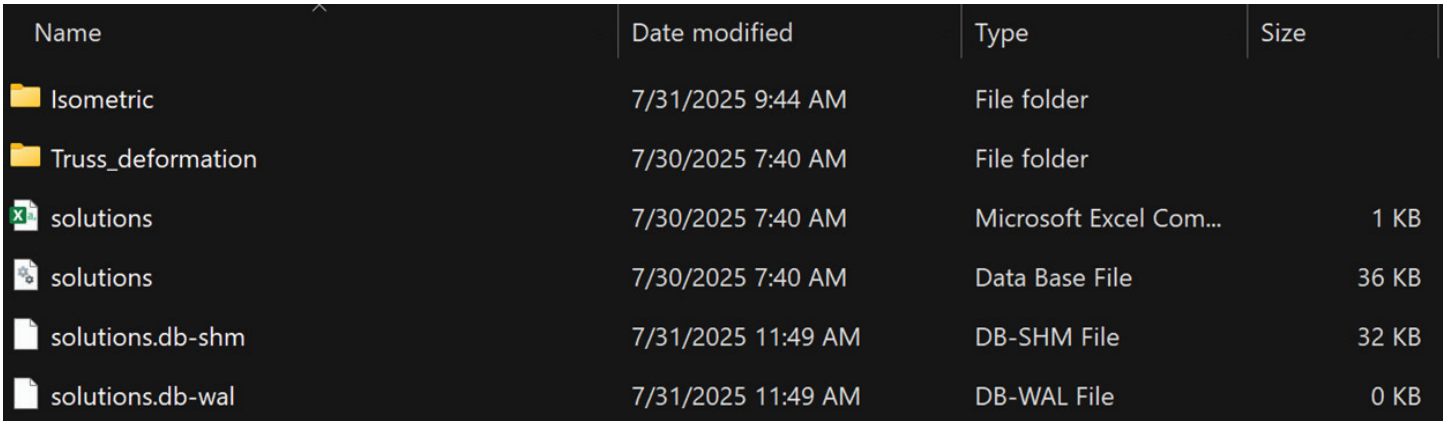
Figure 39

The “Enabled” input of the “Save to Population” component requires a Boolean toggle to function properly.

When set to “False”, the “Save to Population” component will not run, and is effectively “paused”. This is the default position for the script.
When set to “True”, a few things happen. First, the “Save to Population” component will begin creating values for an entire generation of individuals based on the gene generator. Previously, the gene generator only output one set of parameters at a time. When this toggle is set to “True”, it will move on to the second after outputting and recording the first, then the third, so on and so forth until the iteration limit (Page 10) is reached. It will then save all aggregated data to a .csv file in the set directory, alongside all associated media.

It is crucial that this toggle be set to “False” while any changes are being made to the script in any component that precedes this point. If changes are made while the toggle is set to true, the script will run by default, saving over itself and any data previously generated. This can get very slow and very messy, very quickly.
Additionally, the “Save to Population” component will not run if any file within its directory is open. All files within the designated directory, especially the .csv database, should be closed prior to setting this toggle to “True” and running this portion of the script.

OUTPUTS



Name	Date modified	Type	Size
Isometric	7/31/2025 9:44 AM	File folder	
Truss_deformation	7/30/2025 7:40 AM	File folder	
solutions	7/30/2025 7:40 AM	Microsoft Excel Com...	1 KB
solutions	7/30/2025 7:40 AM	Data Base File	36 KB
solutions.db-shm	7/31/2025 11:49 AM	DB-SHM File	32 KB
solutions.db-wal	7/31/2025 11:49 AM	DB-WAL File	0 KB

Figure 36

Upon running the "Save to Population", the following will be output to the designated directory:

1. One folder per "Image Capture" component, each containing one image per design solution, tagged with its respective numeric ID. In the above example (Figure 36), this included "Isometric" and "Truss_deformation".
2. One folder per "Files" input of the "Data Aggregator" component, each containing one file per design solution, tagged with its respective numeric ID.
3. One .csv file named "solutions" which can be easily read by any spreadsheet software.
4. Three database files, used by both the gene generator and "Save to Population" components to organize data.

The media folders and .csv file will typically be the most relevant outputs to the user. Below is an example of a .csv file from the example script, using Microsoft Excel to be easily read as a spreadsheet. Each row represents an individual design solution. Subtracting 1 from the row number, because row 1 is used for column titles, will yield that individual's unique numeric ID, which corresponds to the images and other media in the media folders.

	A	B	C	D	E	F	G	H	I
1	Height of Entrance	Height of the middle of the vault	Angle of Opening in Degree	Truss frequency along the length	Truss frequency space along the width	Truss depth	Number of elements	Total length of elements	Length of the longest member
2	16	38	19	47	10	-2.7	3760	15311.42	7.93
3	29	34	27	38	4	-2.5	1216	10108.65	20.97
4	17	24	39	39	3	-1.8	936	7422.25	17.16
5	15	30	44	26	11	-2.3	2288	9630.58	5.79
6	20	20	31	17	4	-2.3	544	4271.24	15.11
7	15	21	14	23	13	-2.7	2392	9501.01	4.66
8	17	43	34	16	6	-2.3	768	5856.00	14.08
9	20	15	37	41	3	-2.7	984	7448.39	19.05
10	22	18	16	26	6	-2.8	1248	6888.86	11.71
11	21	20	38	24	10	-2	1920	8300.78	6.65

Figure 37

VISUALIZER

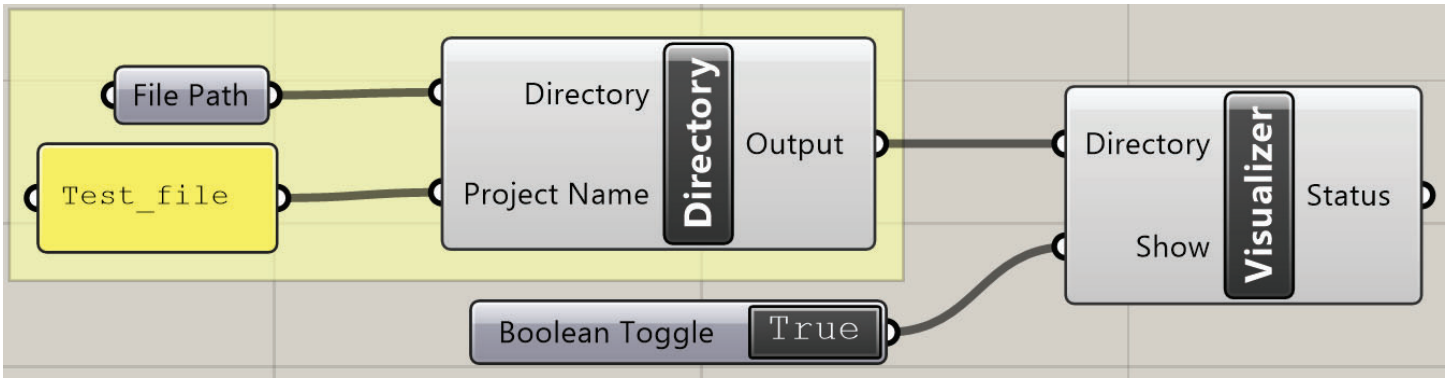


Figure 37

The Morpho Visualizer component provides an interactive environment for exploring, comparing, and evaluating populations of design solutions generated through Morpho's evolutionary design exploration process. By integrating visual representations with quantitative performance data, the Visualizer enables designers to investigate the design space, identify patterns and trade-offs, and select promising solutions based on multiple criteria.

LAUNCHING THE VISUALIZER

To launch the Morpho Visualizer, connect the directory path of the generated design database to the Visualizer component in Grasshopper and set the Boolean toggle to True. This opens a separate visualization window, where the stored design solutions and their associated performance data and visuals can be explored (Figure 37).

EXPLORING DESIGN SOLUTIONS

The main window displays a gallery of generated design solutions (Figure 38), allowing users to visually examine and compare alternatives. Users can select their preferred image view to explore the solutions from different perspectives, depending on the images stored during the design exploration process.

In addition to the visual representations, users can select specific design parameters or performance metrics to display as captions alongside each solution (Figure 39). These may include geometric properties, structural performance indicators, environmental metrics, or other quantitative values recorded in the database. Displaying these values alongside the images facilitates comparisons between the visual characteristics and measured performance of different alternatives.

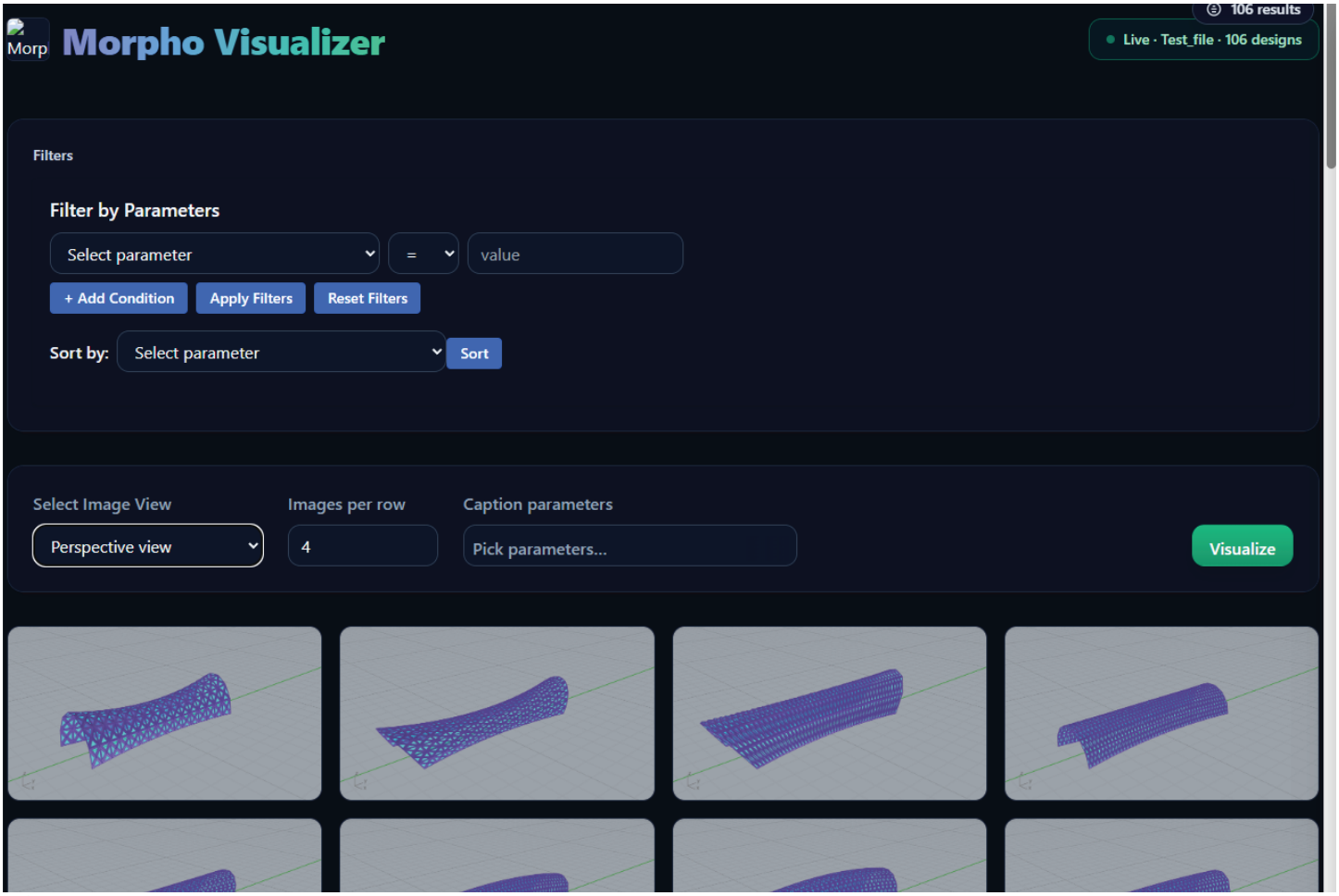


Figure 38

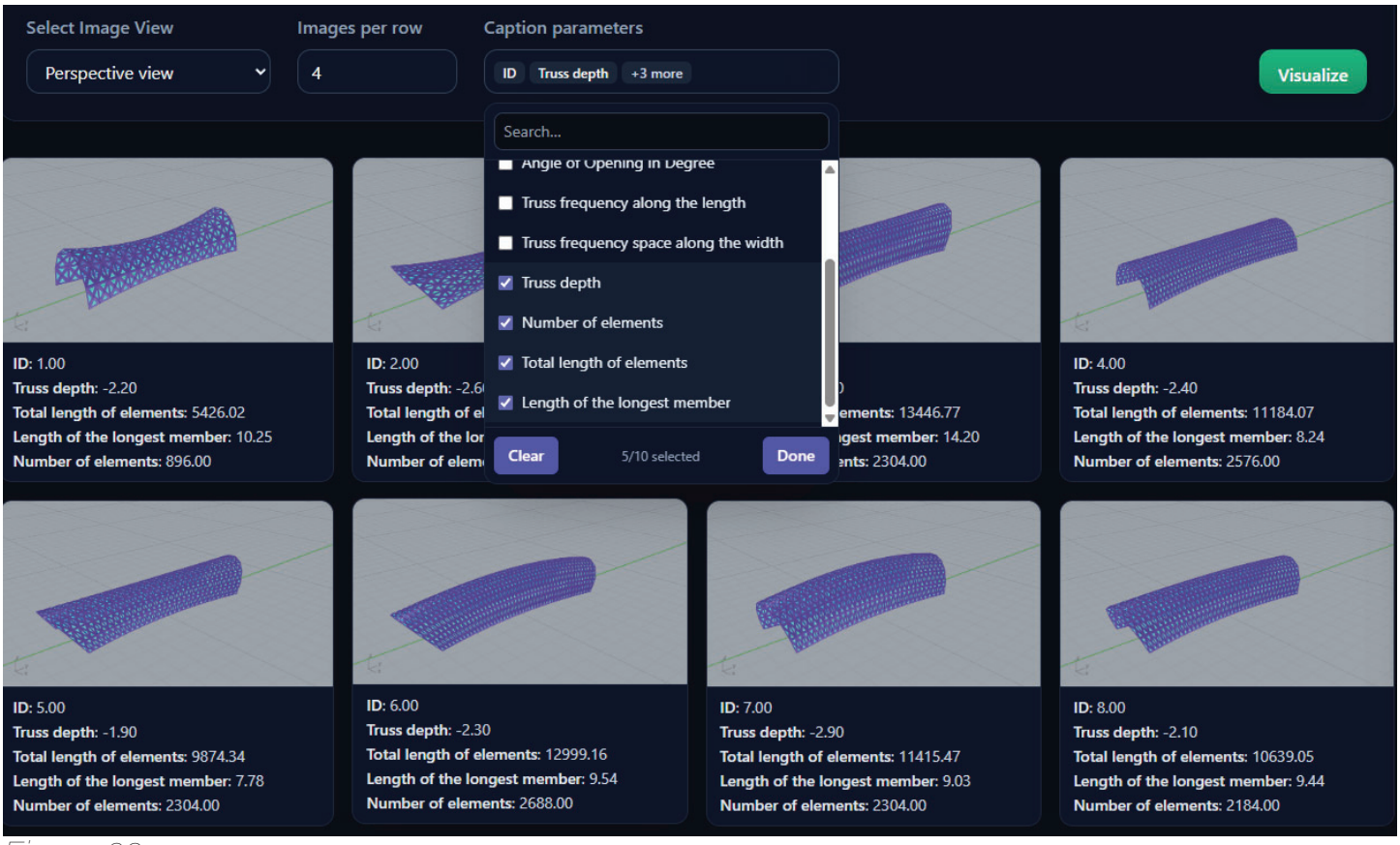


Figure 39

FILTERING DESIGN SOLUTIONS

The Visualizer provides a set of filtering options that allow users to narrow their search based on selected criteria (Figure 40). By defining ranges or conditions for individual parameters, users can focus on particular regions of the design space and exclude solutions that do not meet their requirements.

For example, when exploring a structural system, users may filter solutions based on maximum displacement, structural mass, or daylight availability. Applying multiple criteria enables designers to identify alternatives that satisfy specific performance requirements while retaining desirable architectural characteristics.

This feature is particularly useful for progressively narrowing a large population of alternatives into a smaller set of promising candidates for further evaluation.



Figure 40

VISUALIZING THE DESIGN SPACE

Clicking the Visualize tab opens a secondary window containing graphical tools for examining the distribution of generated solutions and the relationships among their parameters.

2D SCATTER PLOTS AND PARETO FRONTS

The 2D scatter plot allows users to select two parameters and examine their relationship across the population of generated solutions (Figure 41). Each point represents an individual design alternative, positioned according to its corresponding parameter values. These plots help users identify correlations, trends, clusters, and trade-offs between competing design objectives.

The Visualizer also enables users to identify Pareto-optimal (non-dominated) solutions. A solution is considered Pareto-optimal when no other solution performs at least as well across all selected objectives and strictly better in at least one. The resulting Pareto front highlights alternatives that represent different compromises among competing objectives.

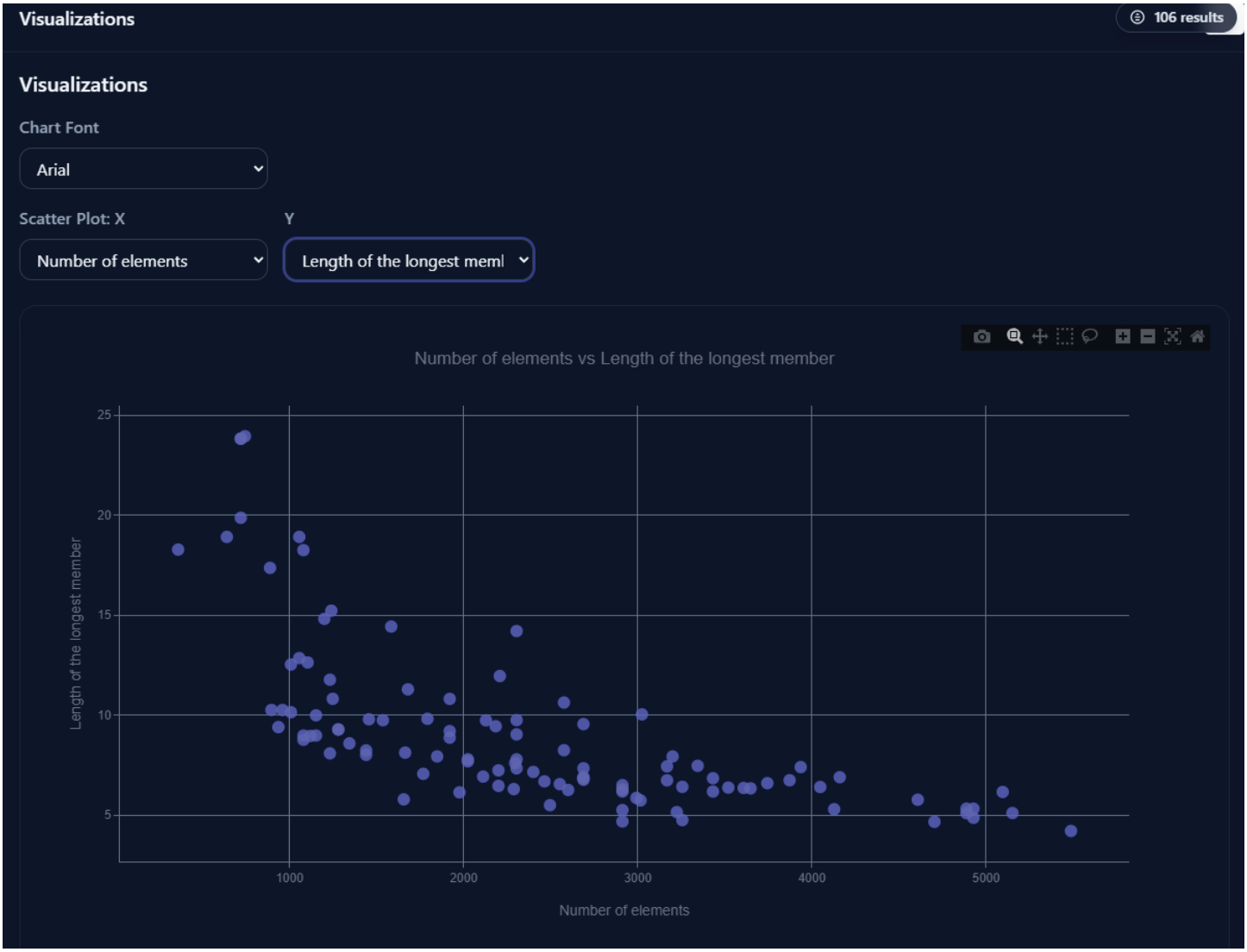


Figure 41

For example, when seeking to minimize the fabrication complexity of a truss system, the Pareto front may reveal a range of efficient solutions, allowing designers to evaluate trade-offs between the maximum member length and the total number of elements. Designers can assess whether reducing member lengths justifies an increase in the number of elements, considering the implications for fabrication, assembly, and construction.

It is important to note that Pareto optimality depends on the selected objectives and their optimization directions (minimization or maximization). A solution identified as non-dominated in a two-objective comparison may not remain non-dominated when additional objectives are considered.

By clicking on any data point in the plot, users can open a separate window displaying detailed information about the corresponding design solution, including its parameter values and all available images. This interactive feature allows designers to examine individual alternatives in greater detail, linking quantitative performance metrics with visual design characteristics and supporting more informed comparisons and design decisions.

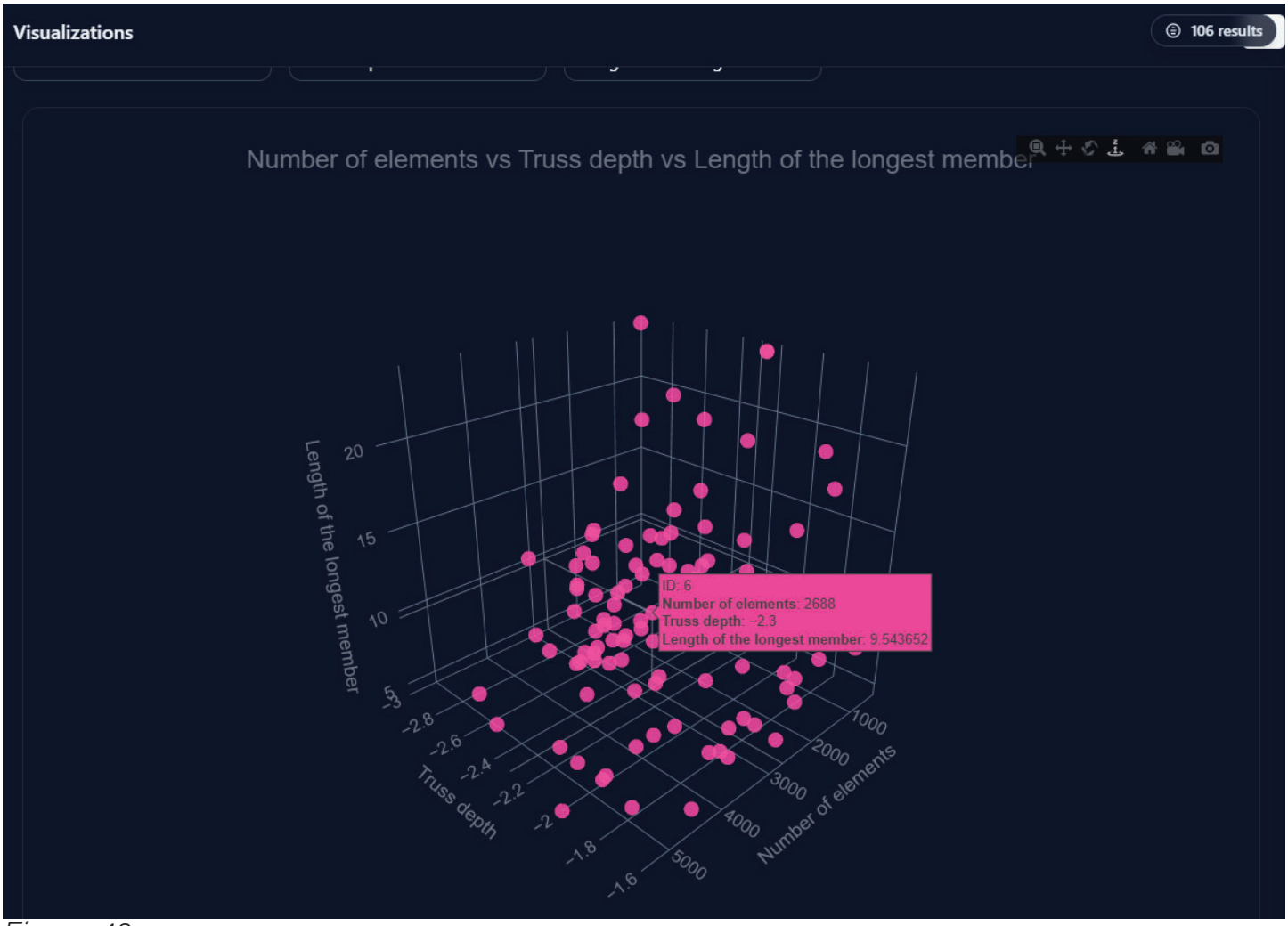


Figure 42

3D SCATTER PLOTS

The 3D scatter plot (Figure 42) extends this analysis by allowing users to examine the distribution of solutions across three selected parameters simultaneously.

This visualization is particularly useful for investigating complex relationships that may not be apparent in two-dimensional comparisons. For example, designers may examine the relationship among truss depth, total number of elements, and maximum element length to better understand the fabrication complexity of different design alternatives and the trade-offs among these parameters.

By clicking on any data point in the plot, users can open a separate window displaying detailed information about the corresponding design solution, including its associated parameter values and all available images. This feature facilitates a closer examination of individual alternatives and connects quantitative performance data with visual design characteristics.

Together, these capabilities enable users to investigate the distribution of solutions, compare individual alternatives, and identify promising regions of the design space for further exploration.

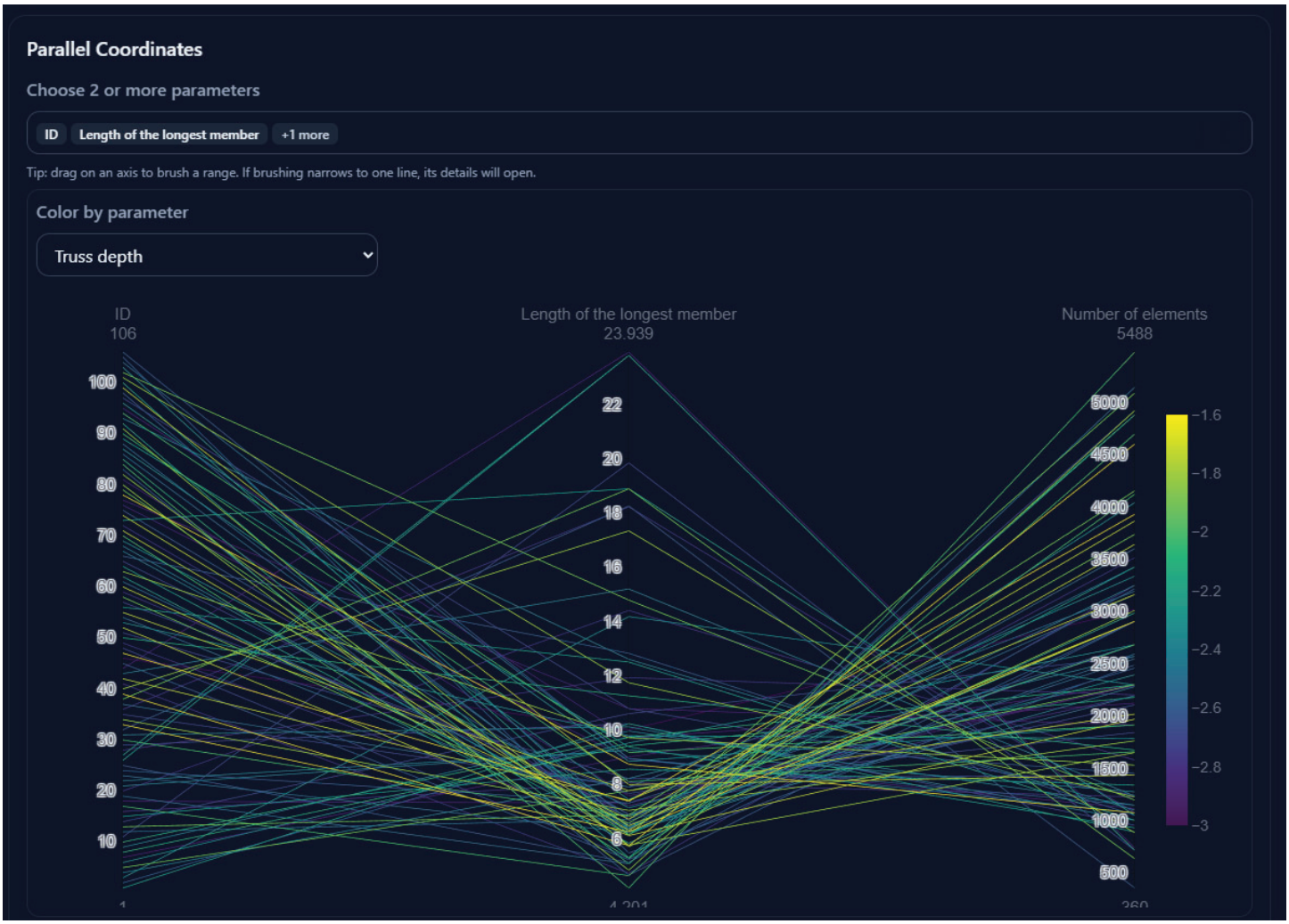


Figure 43

PARALLEL COORDINATE PLOT

The parallel coordinates plot (Figure 43) provides an alternative method for examining relationships among multiple design parameters simultaneously.

In this visualization, each parameter is represented by a parallel axis, and each design solution is represented by a polyline connecting its values across the axes. This allows users to compare several performance metrics within a single graphical representation.

Parallel coordinates are particularly valuable for multi-objective design exploration, as they help reveal patterns, relationships, and potential conflicts among multiple parameters.

For example, designers can investigate whether solutions with lower structural mass consistently exhibit greater displacement or whether certain geometric configurations achieve favorable performance across several objectives.

Unlike scatter plots, which typically display two or three parameters at a time, parallel coordinates support the simultaneous examination of a larger number of design variables and performance indicators. Users can use these plots to investigate the distribution of alternatives and identify promising regions of the design space for further exploration.

Exporting Visualization Results

The graphs generated through the Visualizer can be downloaded for further analysis, documentation, and presentation. Exporting these visualizations enables users to communicate their findings, compare design alternatives, and document the decision-making process.

Using the Visualizer to Support Design Decisions

Morpho Visualizer is intended not only to display optimization results but also to support designer-centered exploration and decision-making.

Rather than automatically selecting a single optimal solution, the Visualizer enables designers to investigate the diversity of generated alternatives, recognize performance trade-offs, and combine quantitative assessment with qualitative design judgment.

A typical exploration workflow involves examining the overall population, filtering solutions according to performance requirements, investigating relationships among objectives using graphical representations, and identifying a shortlist of promising alternatives.

Through this iterative process, designers can move between broad exploration of the design space and focused evaluation of selected solutions, making informed decisions that account for both measurable performance and architectural design intentions.

COPYRIGHT

Morpho copyrighted by Anahita Khodadadi is licensed under Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0). To view a copy of this license, visit https://creativecommons.org/licenses/by-nc/4.0/

REFERENCING

To reference this document, please use the following citation:
Khodadadi, Anahita, (2026) Morpho 2.0 User Manual. Zenodo. https://doi.org/10.5281/zenodo.23477542

ACKNOWLEDGMENT

The development of Morpho Design Explorer was supported by the University at Buffalo Research Foundation. The author acknowledges the contributions of the Morpho Lab Team, including Krishna Sivakumar, Jagruthi Kelavalli Jagadish, and Babak Askari, who assisted with computational development; Ryan Mellen, who assisted with the initial documentation and formatting of this manual; and Kimia Ghaderian, who designed the Morpho logo. The author led the project and was responsible for the verification, refinement, and further development of Morpho and its documentation.

CONTACT

For any questions, feedback and valuable suggestions, please reach out to:
akhodada@buffalo.edu or morphodesignexplorer@gmail.com