# Biomorpher

*Interactive Evolution for Grasshopper*

User Manual for v0.7.0

Written by John Harding

Summer 2020

# Contents

# 0. What’s new in Version 0.7.0

- Geometry input takes Curve and Surface geometry as well as meshes.
- Creation of the ‘Biomorpher Solution’ parameter for internalising data.
- Complete overhaul of the Reader component now using Biomorpher solutions.
- Lists of performance values can now be used with repeated names automatically indexed.
- Main component in ‘galapagos pink!’
- Incorporated elitism into selection to negate losing ‘best’ designs between generations.
- Optional elitism mutation.
- Restart button for a new evolutionary run negating the need to restart Biomorpher.
- New scatter graph plot for comparing two performance criteria.
- Performance plot now includes current population data.
- GUI improvements:
    - Grey colour to relate to standard Rhino background.
    - Main control area is now on the left hand side.
    - K-means display angles are distributed evenly.
    - Fixed min-max bug on performance graph.
    - Minimise and maximise criteria clearly labelled.
    - Performance display above design now based on radius rather than opacity.
    - Evolutionary history is justified correctly.
    - Scrollbars now visible on history canvas.
    - Donate button in case you wish to say hello and buy me a coffee :)

# 1. Introduction

Biomorpher is a Rhino Grasshopper component with a focus on the *interactive evolution* of designs. By 'designs' we of course mean, a particular set of parameter states that lead to a grasshopper solution. With this in mind, the effectiveness of any evolutionary system is dependent on how the model is set up (i.e. the Grasshopper definition itself). In essence, Grasshopper takes a series of numbers (the genotype, or 'DNA') and converts these (using a series of functions/components) into geometry (the phenotype).

As any relatively experienced Grasshopper user knows, when the number of parameters increases, it can become hard to explore these huge design spaces easily just by manually 'tweaking sliders'. This is where Biomorpher comes in. Large design spaces can be explored by allowing human users to interact with the evolutionary process, by guiding the search process. In classic optimisation, this guidance is automatic (i.e. fitter individuals are more likely to survive through the generations), whereas Biomorpher allows mix-mode optimisation, manual selection is just as important as 'performance' against objective functions during evolution.

## 1.1. Motivation

The story begins whilst working on another Grasshopper component, Embryo in 2016. This project aimed to generate grasshopper graphs (definitions) automatically, however the huge combinatorial design space this opened meant design spaces were vast and difficult to control with any intuition. It became clear that guiding the evolution of these graphs was playful and fun, and not necessarily determined through any standard optimisation metaheuristic (i.e. using objective functions).

Through an interactive search, design spaces could be explored without commiting to explicit goals, allowing intuition and tacit knowledge to enter the search process. As Christian Derix asked (2010, p.63)[1], "how can a designer better interfere with computational heuristics and understand the search struggle, offering the opportunity for identification between designers' analogue and computational heuristics?"

Although there were some tools out there back in 2016, they were not *devoted* to an interactive search. I started to develop a simple tool called 'design explorer' that could display more than one gh preview simultaneously, which later inspired Biomorpher.  The idea was not new at all, but a Grasshopper based environment felt like the correct place for such a tool to flourish, hence Biomorpher was born. The name is inspired by Richard Dawkins' *Biomorphs* from 1986[2]. (fig.1).

[1] Derix, C. (2010, October). Mediating spatial phenomena through computational heuristics. In Proceedings of the 30th Annual Conference of the Association for Computer Aided Design in Architecture (pp. 61-66).
[2] Dawkins, R. (1986). The blind watchmaker: Why the evidence of evolution reveals a universe without design. WW Norton & Company.

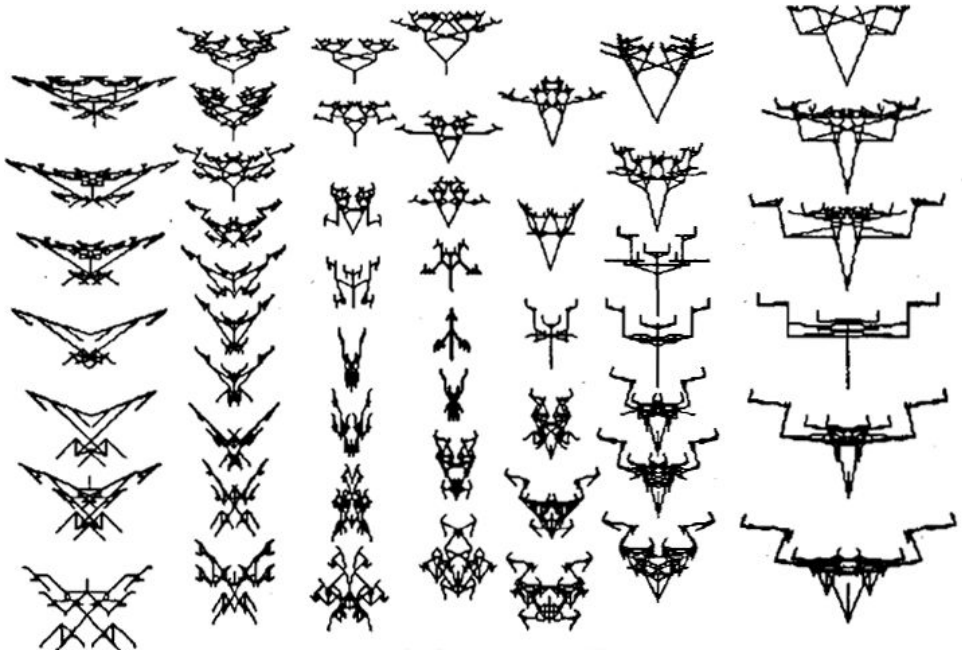

Fig.1. Some original Biomorphs evolved using Dawkins' program (1986)

One of the main principles of Biomorpher has always been to keep the interface clean and easy to use, sometimes at the expense of having millions of features. *Just because you can, doesn't mean you should.*

## 1.2. Acknowledgement

The initial development  was sponsored by a UWE Early Career Researcher Development Award which helped facilitate some much needed help from Cecilie Brandt-Olsen. I am completely indebted to Cecilie's help at the start of the project, setting up the main interface using Mahapps as well as writing the k-means algorithm from scratch! Thank you Cecilie.

## 1.3. Dependencies

Much of Biomorpher is written from scratch such as the genetic algorithm and k-means clustering, however it does rely on the following libraries to support the GUI:

- Helix toolkit version 2.11.0 (MIT Licence).
  *Used to display the 3d representations used in the Biomoprher window.*

- MahApps.Metro (MIT Licence).
  *A toolkit for creating metro-style WPF applications.*

## 1.4. Source Code & Licence

Biomorpher is fully open-source and released for free under the MIT licence. Latest version is available on Food4Rhino. Source code and latest release also available on the github page. See the links here:

- https://github.com/johnharding/Biomorpher
- https://www.food4rhino.com/app/biomorpher

## 1.5. Installation

Due to Biomorpher being reliant on some libraries, the component file and some dll files must be copied into the Grasshopper libraries folder and make sure they are *unblocked* (see file properties) before running Rhino. Take the following steps:

1. Launch Grasshopper.
2. Navigate to: Menu > File > Special Folders > Components Folder.
3. Move Biomorpher.gha and the following dynamic link libraries there:
    a. HelixToolkit.dll
    b. HelixToolkit.Wpf.dll
    c. MahAppsMetro.dll
    d. System.Windows.Interactivity.dll
4. Restart Rhino and Grasshopper.
5. Is Biomorpher shown alongside Galapagos (tab: Params > Util)?
    a. Yes: All good. Time to open some of the examples.
    b. No: Right click on gha and dll files and tick the 'Unblock' checkbox. Go to step 4.

The components are located alongside Galapagos, inside Params > Util (fig.2).

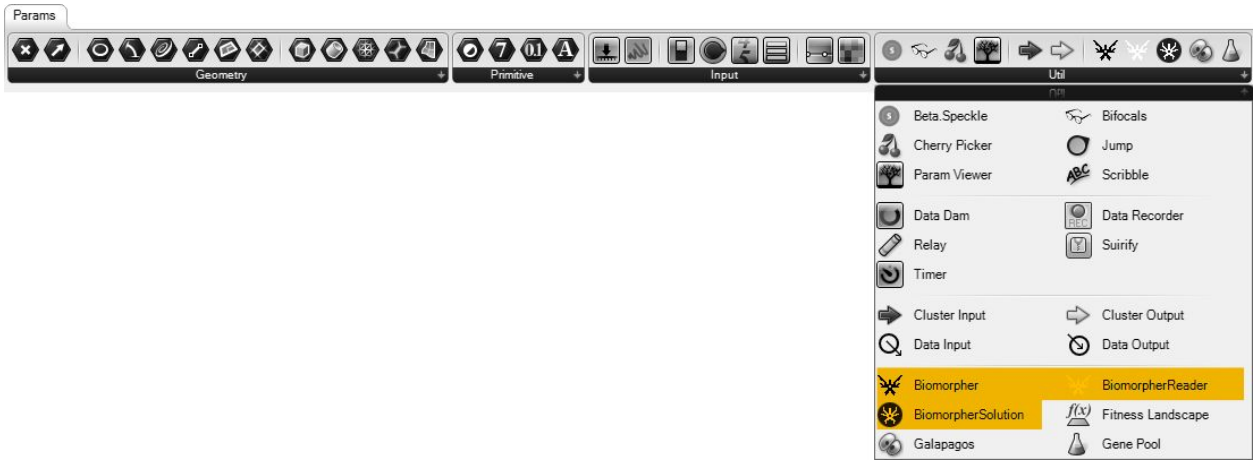


Fig.2. Location of the Biomorpher components in Grasshopper menus after successful installation

# 2. Process overview

Biomorpher uses a subset of Metaheuristic algorithms: Evolutionary Algorithms (specifically a Genetic Algorithm or 'GA') in order to evolve designs. The algorithm used is that described by Goldberg (1989)[3] in his now classic text on GAs. As opposed to a binary encoding, floating point genomes are used in order to deal with the nature of 'sliders' in grasshopper. In short the following processes are used in the following order, looping until the user is content (fig.3):

1. Set 'meta' parameters (population size, mutation rate, crossover rate, etc.)
2. Create a new population of genotypes (default = 48 designs) that control parameters (sliders and genepools).
3. Generate phenotypes from population genotypes (using Grasshopper).
4. Perform [optional] performance analysis of these phenotypes to assign fitness.
5. Display a subset (12) of these to the user using K-means clustering of parameter space.
6. Allow the user to reinstate any historic population if desired.
7. Allow the user to review and manually select designs (in addition to optional performance based fitness). Assign high fitness scores to all designs within a selected cluster.
8. Using the cumulative fitness scores from both performance based and manual selection, Create a new population of genotypes using a Roulette Wheel approach with Elitism.
9. Mutate genotypes according to a specified probability.
10. Crossover genotypes according to a specified probability. This is now the new population of genotypes.
11. Return to item 3.

[3] Goldberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, Reading, Ma.

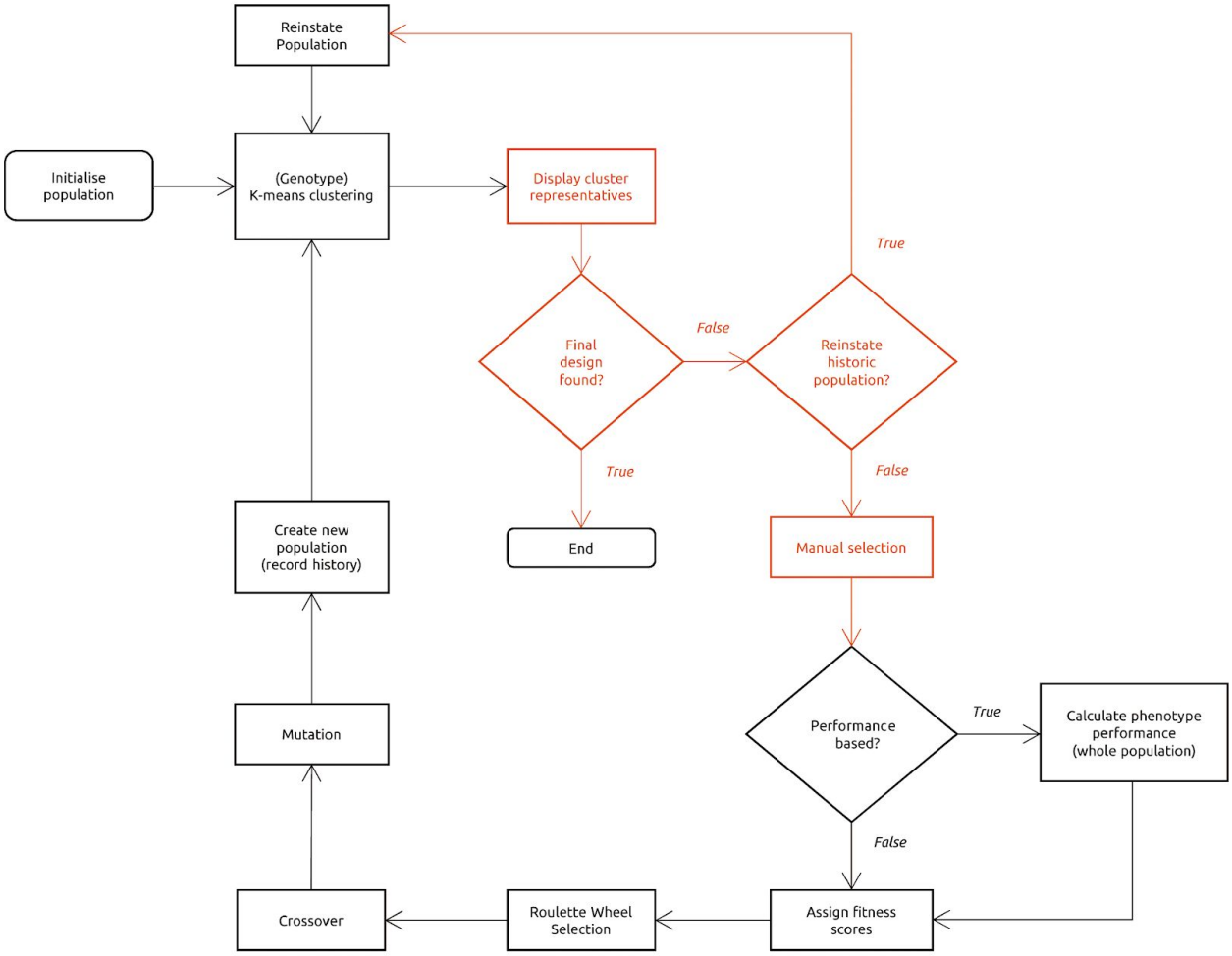


Fig.3. Schematic diagram showing process behind Biomorpher

More detail on the inner workings of Biomorpher - particularly for researchers - can be found in Harding and Brandt-Olsen (2018)[4].

[4] Harding, J., & Brandt-Olsen, C. (2018). Biomorpher: Interactive evolution for parametric design. International Journal of Architectural Computing, 16(2), 144-163.

# 3. Grasshopper Components

There are two Grasshopper components installed, the main Biomorpher component and the Biomorpher Reader component. A single parameter is also installed, Biomorpher Solution which is capable of storing historic evolutionary runs (fig.4). The main component is dressed in a tasteful Galapagos pink.

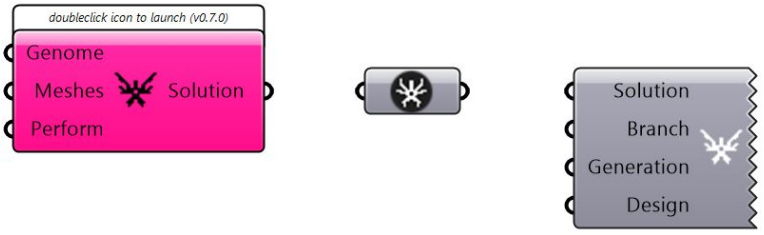


Fig.4. From left to right: The main Biomorpher component, Solution parameter and Reader component.

## 3.1. Biomorpher Component

The main Biomorpher component has three inputs (the last of which is optional) and one output, with the custom parameter data type ‘Biomorpher Solution’.

### 3.1.1. Genome Input

Sliders and Gene pools can be added just like with Galapagos or Octopus (fig.5). These are the parameters that will be adjusted by Biomorpher as it explores the design space. To speed up this process, It is possible to right click on the biomorpher component to add all selected sliders. The wire is faint to differentiate these from standard connections within Grasshopper (note: this may be updated in future to alter this to the Galapagos ‘pointer’ wire).

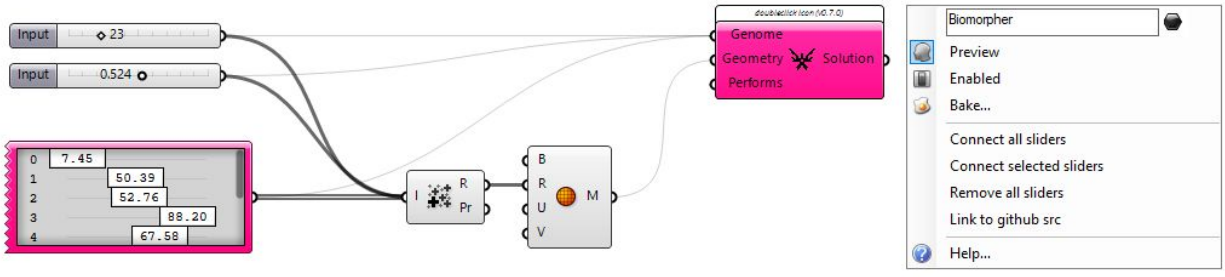


Fig.5. Sliders and gene pools can be used with Biomorpher.

## 3.1.2. Geometry Input

Geometry to be displayed in the Biomorpher viewports can be added here. It is important to note that everything that the definition creates need be added, just enough to guide variation. Surface geometry will be converted to a mesh automatically, however using a meshing component within grasshopper offers more control (such as Mesh Surface for example). As of version 0.7, curves can also be used and displayed during evolution.

Biomorpher will include mesh colours (fig.6), however due to the limitations of wpf 3d graphics, individual colours per mesh face are not possible. Instead, Biomorpher will average the colours over the entire mesh. However, for a list of meshes each colour will be respected and shown in the 12 viewport displays within the main Biomorpher design window.

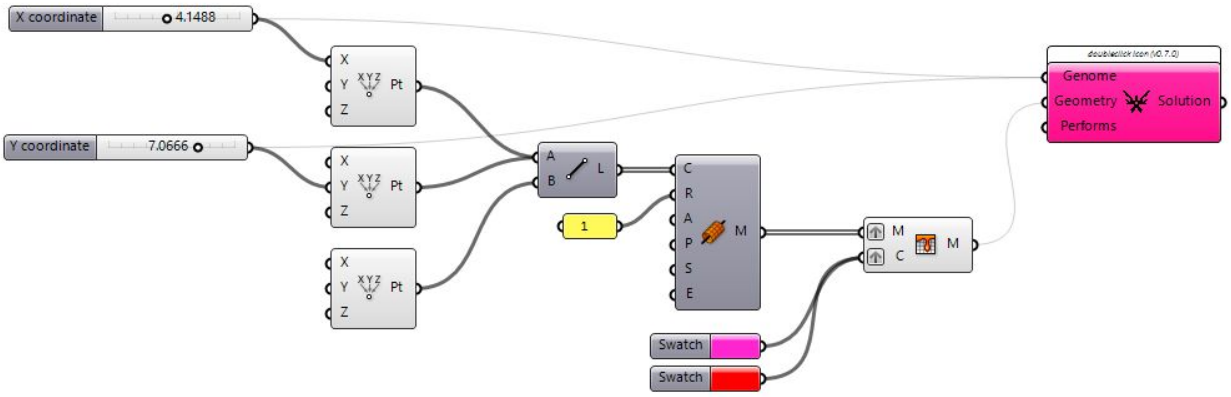


Fig.6. Using mesh colours with Biomorpher.

## 3.1.3. Performance Input

This is an *optional* input - multiple performance values may be entered here, much like the 'fitness' pointer in Galapagos. Biomorpher weights these equally during any evolutionary run that involves performance metrics. Even if these values are not used to drive evolution, they will be displayed in each design window to *guide* how designs are selected manually, in a way nudging designs to a certain performance score without dominating the search.

A cap of 8 performance values is applied, with each having a unique name. The names of the performance values are read from the source parameter. If a list of values is included, due the parameter name being shared, a numeric suffix is added to each. Alternatively, it is possible to explode, merge, etc. in order to assign parameter names to each performance value, and indeed this is good practice rather than relying on the automatic numeric suffix (fig.7).

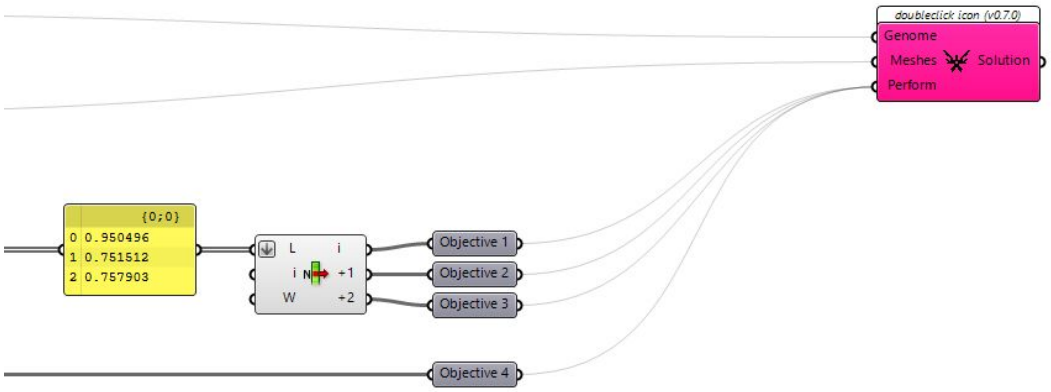


Fig.7. An example of good practice when naming performance criteria before sending to the Perform input (note that 'list item' outputs cannot be renamed, hence the extra parameters).

Changing the *number* of performance values used during an evolutionary run will cause Biomorpher to flag an error, informing the user that the performance count has changed.

### 3.1.4. Solution Output

Any evolutionary run will have a history of parameter values (genotype). These values are stored alongside a reference to the Sliders/GenePools within a new parameter type called 'Biomorpher Solution'.

## 3.2. Biomorpher Solution [Param]

Previous runs may be stored within the parameter and internalise this data and then save within a Grasshopper file. The idea is that when the Biomorpher window is closed, the results can therefore be saved (serialized) within a Grasshopper file itself without having additional files.

## 3.3. Biomorpher Reader

The Reader component is an optional extra that will take any biomorpher solution and automatically adjust sliders and genepools for a particular genotype/design. Any branch (see Section 4.3.), generation or design can be selected from a biormopher session. Figure 8 shows how multiple biomorpher solutions previously explored can update the parameters with a definition. Note that the exact same sliders and gene pools must of course be used (as per the original evolution), as the reader component will try to find these on the canvas before adjusting the values automatically.

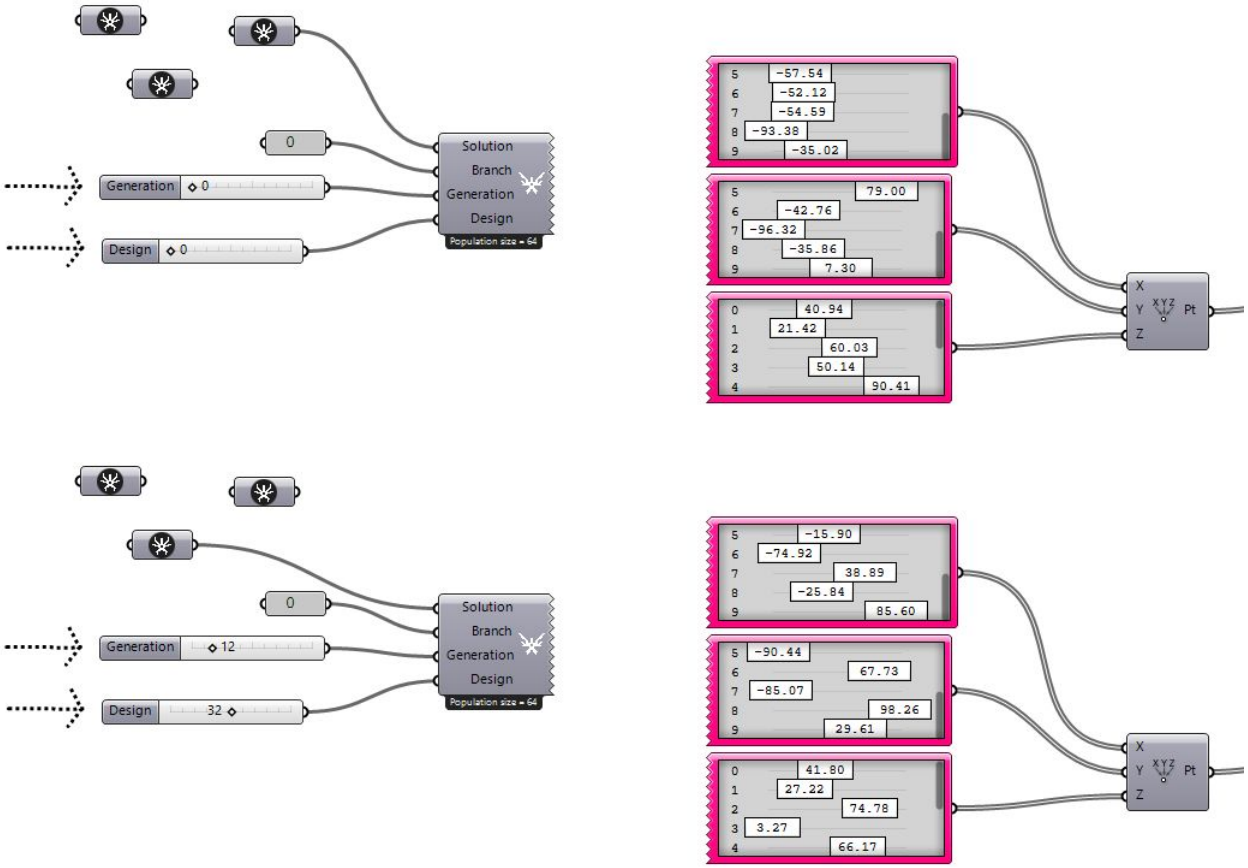


Fig.8. Biomorpher Reader component used with several different internalised solutions.
Selecting the branch, generation and design adjusts the sliders and gene pools accordingly..

# 4. Biomorpher Window

Once the component inputs are set up on the main biomorpher component, double click on the centre of the Biomorpher component to open the main window that will control evolution (Fig 9.). There are 6 tabs at the top, most of which become active after a population is intialised.

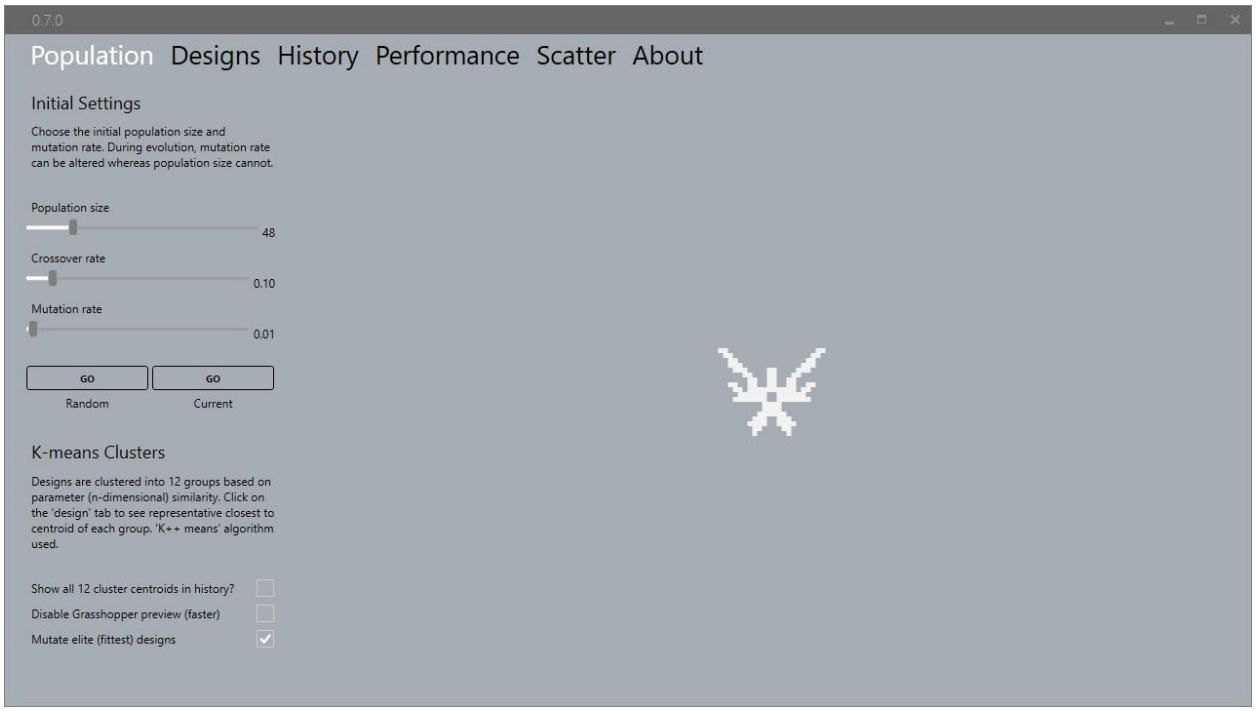


Fig.9. The Biomorpher GUI upon launch. Tabs at the top, controls to the left, display to the right. *Here I am… etc.*

Before the population is initialised, the size, cross-over rate and mutation rate must be specified. Effective evolution is strongly dependent on these, although common values are set as default. Further reading on effective use of genetic algorithms can be found by reading Mitchell (1998)[5]. In general, increasing population size can help to explore the solution space more effectively, however there is a computational/time cost that may not be necessary for a given design problem. Mutation and crossover rates may be changed during evolution, but population size cannot.

Some additional options can be specified here, such as recording all 12 cluster representatives in the history window (see next section), disabling the view preview (similar to the same setting in galapagos) and whether or not to mutate elite designs. Setting this latter option to true is default, as often an evolutionary run can get stuck too early by homing in on a solution (local optima), and with elitism this would result in all the designs in the population being the same.

[5] Mitchell, M. (1998). An introduction to genetic algorithms. MIT press.

Once the settings have been established, there are two 'GO' buttons used to initialise a population, either creating a random population, or by using the current parameter state in grasshopper. Note that if the latter is used, the mutation rate may benefit from being higher in order to kickstart evolution (note, this can be reduced if required during evolution).

## 4.1. Population Tab

Biomorpher is geared towards manual (artificial) selection, however there is a problem: Genetic algorithms work best with large populations, however for a human user to view and compare a large number of options is difficult. With this in mind, Biomorpher uses k-means clustering to group similar designs (in terms of genotype) together, reducing the designs displayed to 12 only. This retains the large population size, beneficial for performance optimisation, whilst allowing manual intervention during evolution. Bonham and Parmee (2004)[6] were the first to use clustering of a parameter space in a GA to enable such a mixed-mode evolution.

The display therefore shows each cluster with a central representative[7] design and associated neighbours (essentially, closest points in a high dimensional space that includes all sliders and genepools in a gh definition) set out radially. The closer the point, the closer the design to the cluster representative (fig.10).

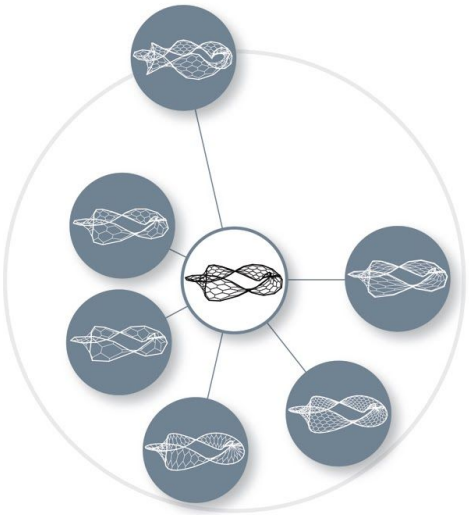

Fig.10. Groups of similar designs (in parameter space) are represented by a 'cluster representative'; the design closest to the k-means centroid. *Image by Cecilie.Brandt-Olsen.*

[6] Bonham, C. R., & Parmee, I. C. (2004). Developments of the cluster oriented genetic algorithm (COGA). Engineering Optimization, 36(2), 249-279.
[7] The cluster representative the design in a population that is closest to the k-means centroid.

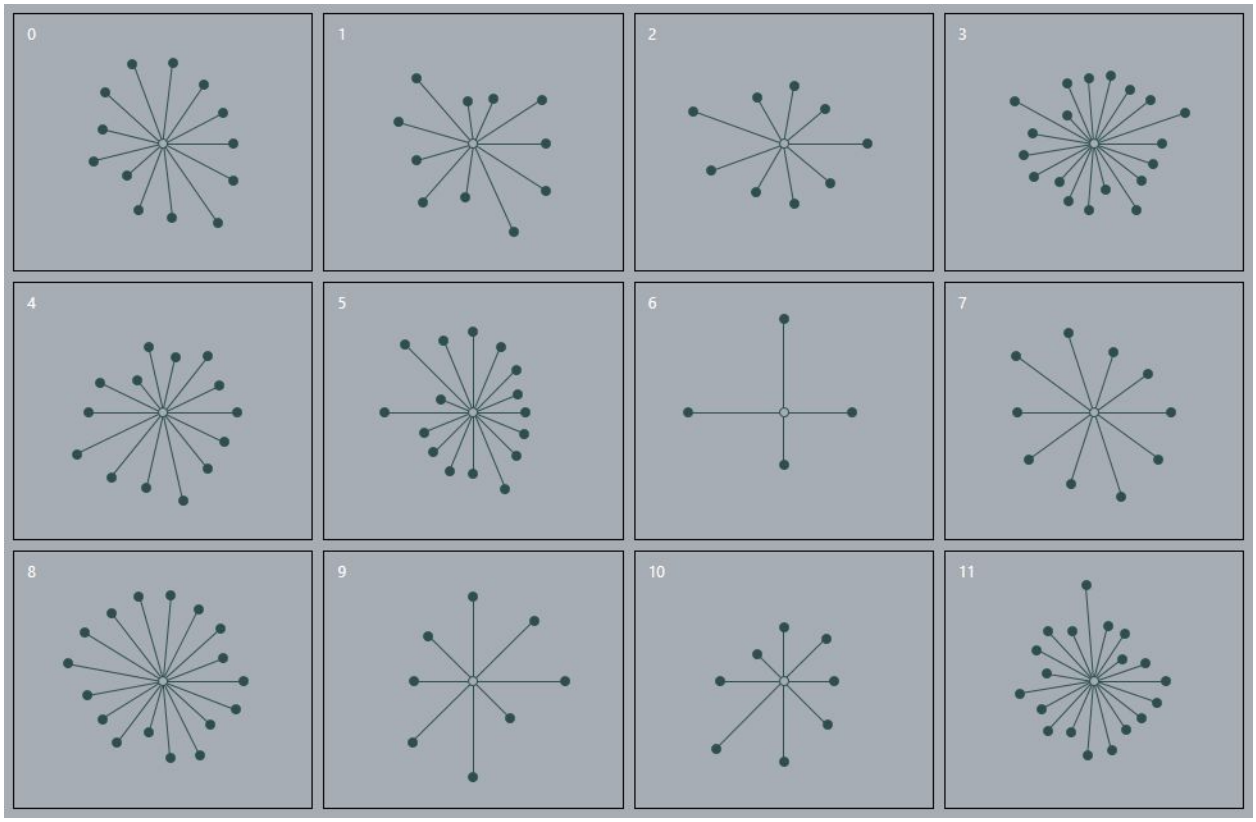


Fig.11. Display of clusters for a population of 160 designs.

Figure 11 shows how the clusters for each of the 12 designs are displayed. The method of clustering using parameter values does assume there is quite a *direct* mapping between genotype and phenotype. In practice this is common with grasshopper definitions, although not guaranteed. In essence, (in future it would also be beneficial to cluster in terms of performance as well as parameter space).

## 4.2. Design Tab

The representative design for each of the 12 clusters can be viewed by hitting the 'design' tab. Fig 12. shows the 'vase' example included in release 0.7. Each design can be viewed/ orbited/ etc... in each viewport, however the camera will update for all concurrently which in practice, makes comparison between designs much easier.

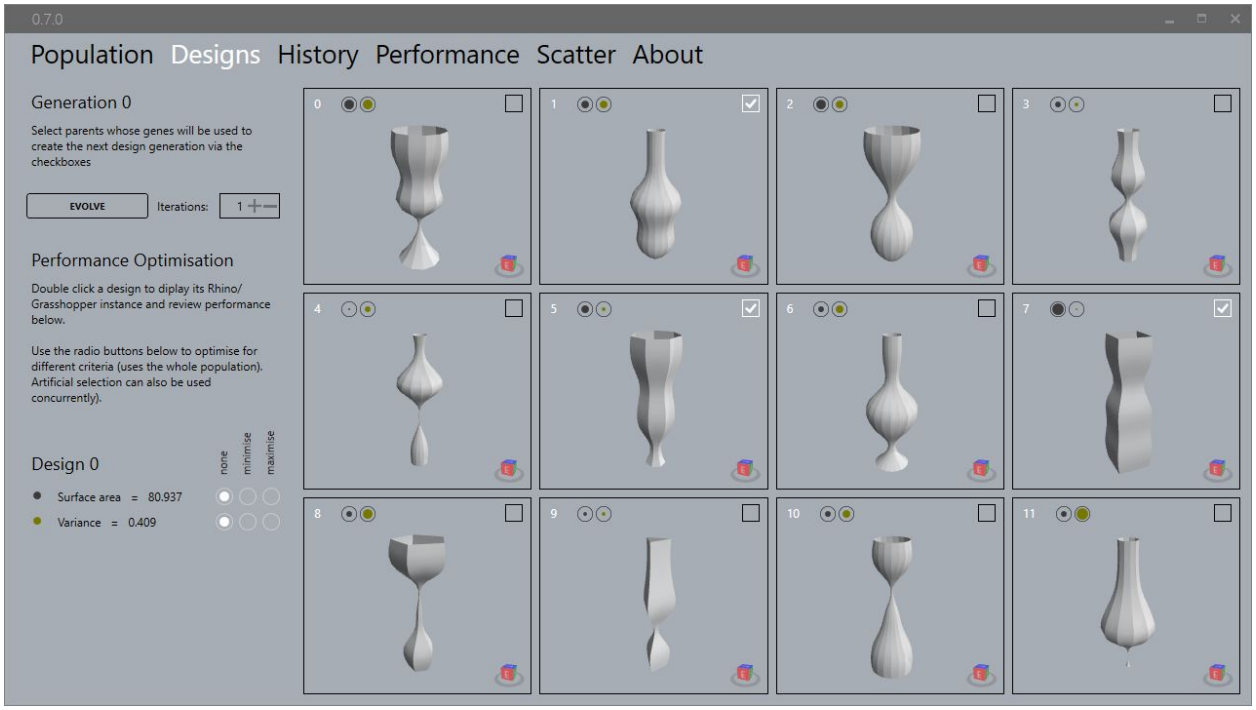


Fig.12. Display 12 custer representatives. Three are selected in this example.

## 4.2.1. Manual Selection

In any genetic algorithm, a particular Phenotype will be given a fitness that influences whether or not it survives to the next generation. This is usually given automatically against a performance measure (objective function), for example in a survival simulation, a creature that collects more food is given a higher fitness score, or a structure that uses material in the most efficient way. In Biomorpher, fitness scores are given from 0.0 to 1.0 per design.

Manual selection takes place by ticking the checkbox at the top right of the cluster. This sets the fitness to maximum (1.0) *for all designs within that cluster*, increasing the likelihood they will be selected for the next generation. Manual selection always overrides any fitness based on a quantitative performance measure (Section 4.2.3).

## 4.2.2. Performance Display

Having a performance value is optional in Biomorpher (see Section 3.1.3), you can evolve designs using manual selection alone. However, if the performance input is used, the values are shown in two places within the design tab. Firstly, for each of the 12 cluster representatives, relative performance is shown in a coloured circle in the top bar of each viewport. The number of circles equals the number of performance measures, with the radius of each representing the value compared to that of the population. This display gives a quick overview of comparative performance, enabling the user to select designs based on this information.

The actual values are displayed in the bottom left hand corner. Double clicking on a viewport highlights this design, as well as adjusting the parameters in the grasshopper model itself. Indeed, double clicking on any viewport will set the parameters to that state, including the history viewports (see Section 4.3).

### 4.2.3. Performance Optimisation

Adjacent to the performance values, three radio buttons per value are displayed (fig.13). Here can be specified for each performance metric whether to minimise or maximise this value, alongside manual selection or independently. Indeed, Biomorpher can be used as per a traditional GA by optimising criteria here without manual selection at all. Each performance metric is weighted equally if set to minimise or maximise (unequal weightings are considered for a future release).

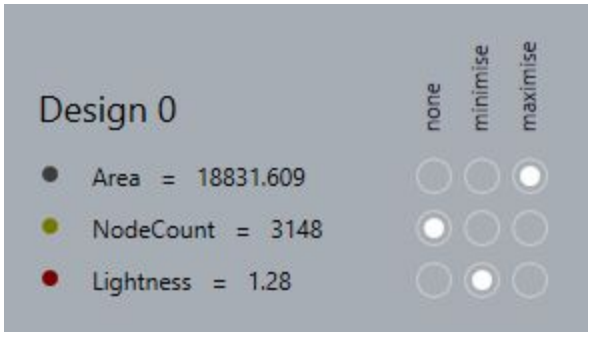


Fig.13. Performance optimisation options for three metrics from the Möbius Band example.

### 4.2.4. Evolution

Once designs and, if applicable, performance measures have been selected, it is time to evolve the population. If any performance measures are included, multiple evolutionary runs may be specified by adjusting the iteration count, then press the ‘Evolve’ button and a new generation of designs will be spawned.

The new population is selected using the fitness values and roulette wheel selection (fig. 14). It can be thought of a spinning wheel used to select designs, with areas proportional to the relative fitness. As well as roulette wheel, at least one design with the highest fitness of the population is guaranteed to survive regardless, known as elitism.

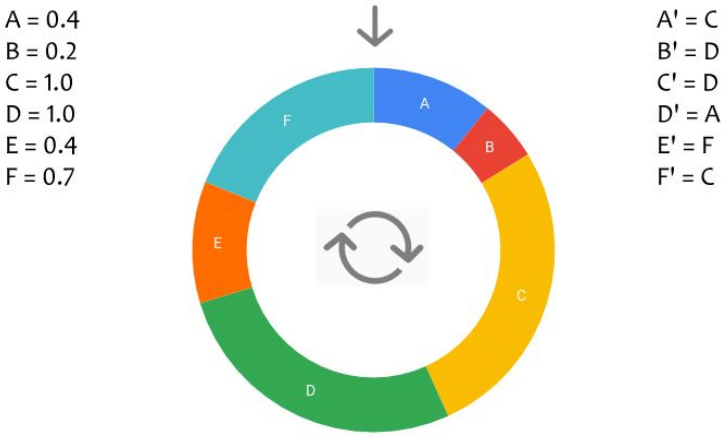


Fig.14. Roulette wheel selection for a small population of 6 designs.

The new population's genotypes are then subject to crossover and mutation as per the user settings (see Section 4.1), and the new generation is displayed.

## 4.3. History Tab

Selecting the 'history' tab shows a map of evolution that records all previous populations. By default only the selected designs and/or optimal designs (according to performance metrics) are shown, although all 12 representatives can be recorded (see Section 4.1). A small viewport shows each design. These cannot be orbited as per the 'designs' tab, however it is still possible to double click each to set the parameter state within grasshopper.

Historic populations can be reinstated by clicking the **'reinstate'** button. This creates a new **'branch'**, named in accordance with the tree data structure (fig.15). You can return to any previous population and generate a map of design exploration of your own which can be exported to a png image file.

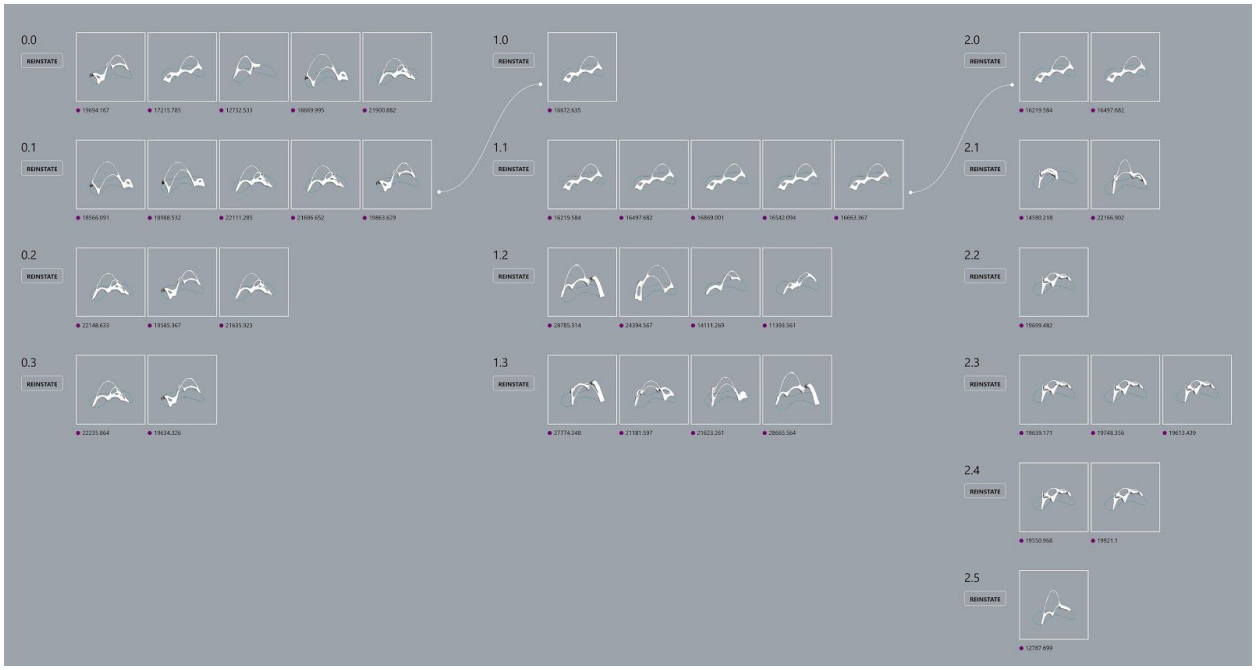


Fig.15. Two new ‘branches’ spawned from an earlier population.Taken from the Henry Moore K2 example.

## 4.4. Performance Tab

A simple line plot for each performance measure (fig.16). Ticking each checkbox allows a relative comparison between different objectives (fig.17). Due to graphic performance and to avoid clutter particularly for large populations, only the 12 cluster representatives and any additional optimal designs are shown for each generation, as well as  an average for the population. Hover over this circle to show the numeric average. The line colours are related to the performance colours shown in the ‘design’ tab.

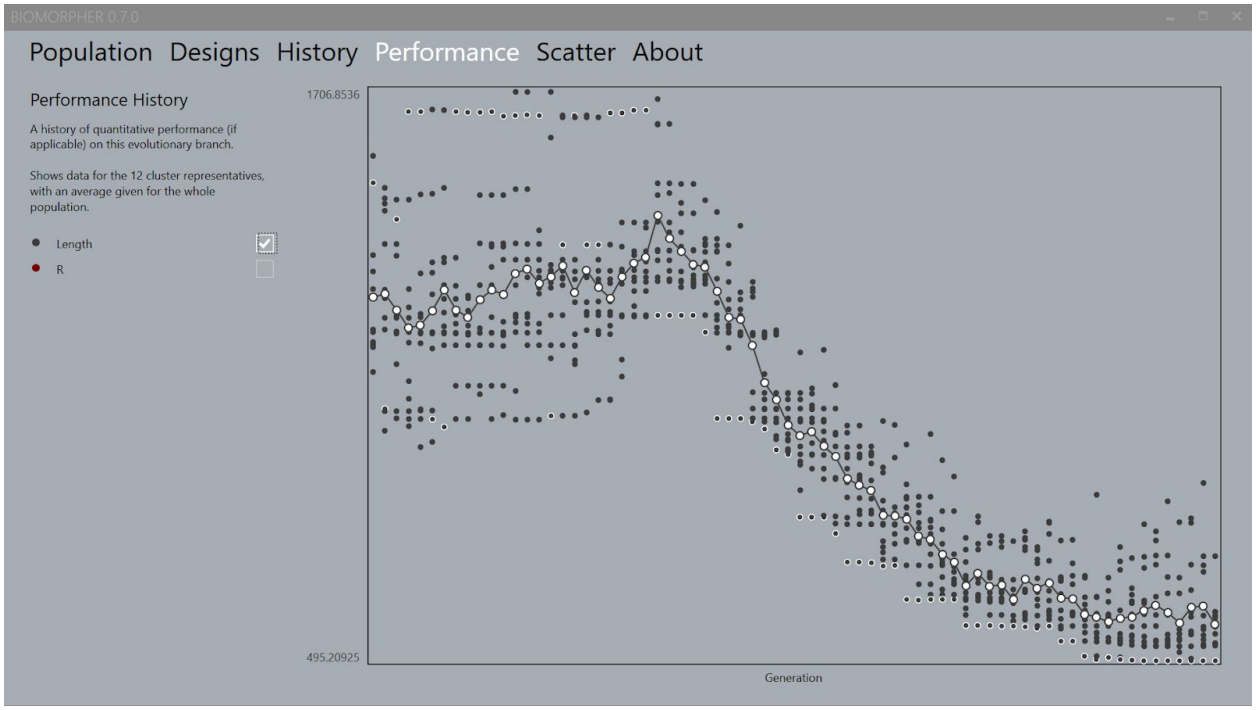


Fig.16. Line plot for one objective.

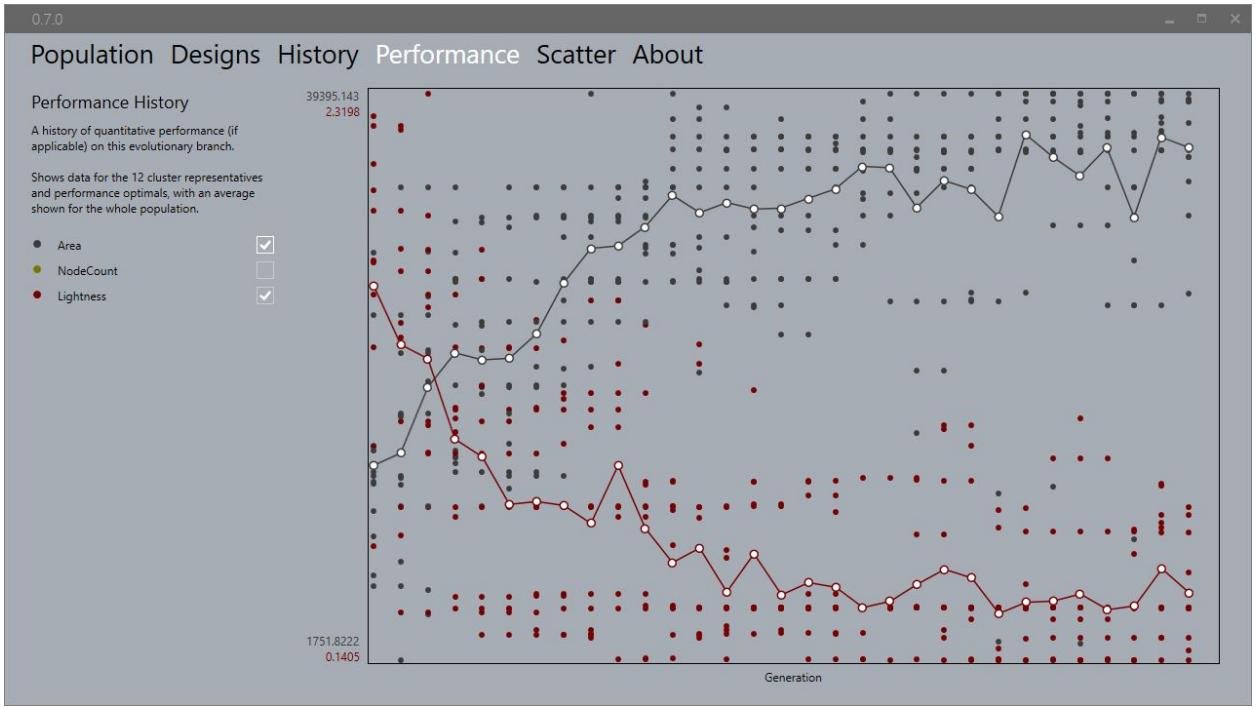


Fig.17. Comparing two objectives on the same plot.

## 4.5. Scatter Tab

Whilst more tools to show the relationship between multiple objectives are possible in future (i.e. Pareto fronts, dimensionality reduction, etc.) as of v0.7 a simple scatter plot is possible for two objectives (fig.18). Simply select the two metrics to compare from the menus on the left hand side. Previous generations are shown increasingly faded compared to the latest population, which is shown highlighted with a white surround.

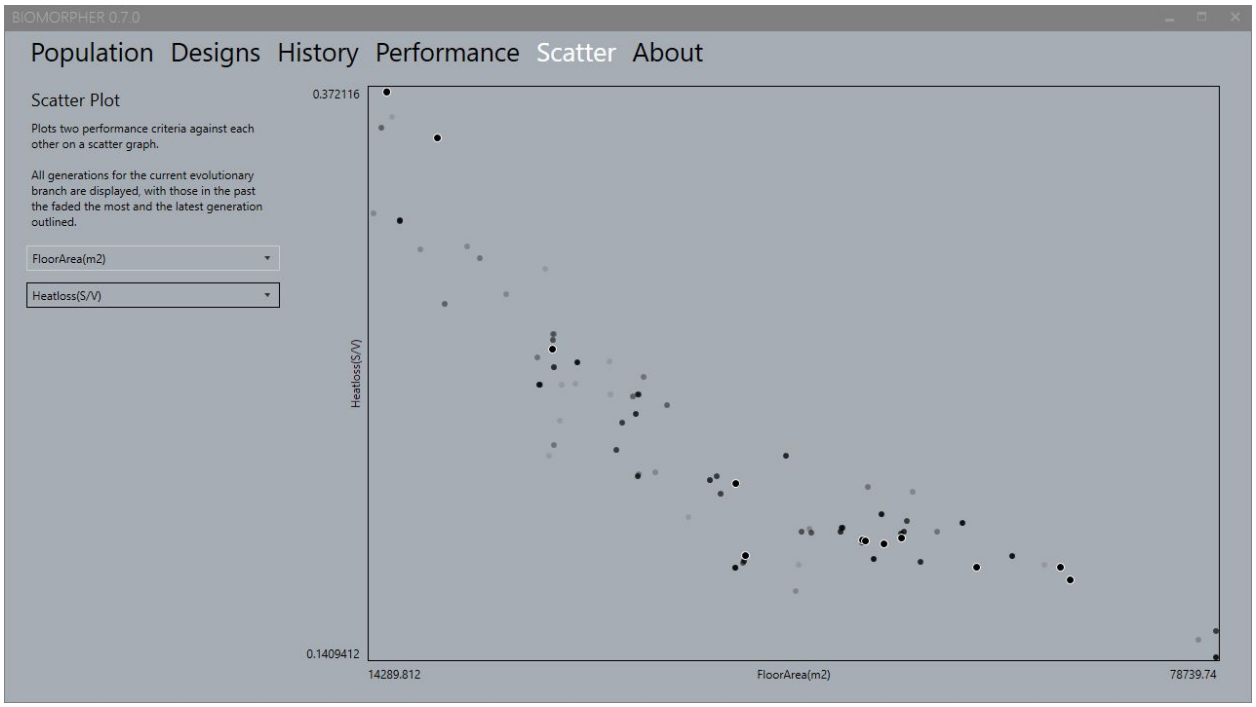


Fig.18. Scatter plot for comparing two performance metrics.

## 4.6. About Tab

Displays some information about the tool and contact details, as well as offering a chance to leave me a message and a coffee :) I hope you enjoy using Biomorpher.

DONATE