Improving Generative Design through Selective Breeding

Peter von Buelow

1Associate Professor, University of Michigan, Ann Arbor, USA, pvbuelow@umich.edu

Summary: Generative Design coupled with associative parametric modelling is proving to be a successful tool for many designers particularly in the early phases of design. The wealth of design solutions which can be rapidly generated certainly help in expanding ideas and exploring possibilities. But simply having the capability to generate different solutions is only half of the process. The designer must make decisions as to which solutions are better. ParaGen, as described in this paper, offers a way to direct the generative process toward more desirable results, and to explore better solutions.

Keywords: generative design, genetic algorithms, design exploration, design optimization

1. INTRODUCTION

ParaGen is an explorative tool which is intended to aid the designer particularly in the early phases of design. It combines parametric form generation with analysis tools which are used to evaluate the generated solutions. A genetic algorithm (GA) is used to steer the process to follow the intention of the designer. The GA can operate either with defined objectives as a fitness function or by allowing the user to visually choose better solutions. In a recent IASS Journal article [1] the overall operation of the method was explained with examples illustrating the single and multi-objective capabilities using defined fitness functions. This paper concentrates on methods which allow user interaction and selection of populations based on visual criteria.

2. OVERVIEW OF PARAGEN METHOD

The ParaGen method has been outlined in other papers in more detail [1,2,3] so only a brief description is given here. As originally described by John Holland at the University of Michigan, a GA follows some reproductive plan by employing certain genetic operators [4]. The plan is carried out through individuals selected from a population and genetic operators which include crossover (breeding of parents through chromosome like strings) and mutation. In the ParaGen method, the cycle is split between selection and breeding on a server and development and evaluation of the children on a cluster of client PCs attached to the server through an internet browser connection. Fig. 1 shows the ParaGen cycle split between server and client machines.

Fig. 1. The ParaGen cycle

The five step ParaGen process begins with the generation of a new child through the breeding process. The child arrives in the form of a coded string (a chromosome) which is decoded by the parametric software into a geometry or solution. The coded string is passed to the parametric software as an Excel or CSV file, and is comprised of a list of parametric variables. After the form is generated, it is next passed, usually as a DXF file, to some simulation software for evaluation. In the example show below an FEA simulation was made using STAAD.Pro by Bentley Systems. Any one or multiple programs can be used to perform the desired evaluations. All numeric performance data is collected and stored in a CSV file ready to be input into the SQL database. Also, it is helpful for the later evaluation by the designer, to gather images both of the geometry itself as well as images of the performance. The images are displayed in the web interface as shown in Figs. 2-4.

Fig. 2. Sample from the initial 100 random solutions of Phase 1

The selection of parents is normally accomplished through one or more preset fitness functions. These are defined in terms of SQL queries and
can include relatively complex filters which dynamically determine a population set for each parent. In the interactive mode parents can be chosen directly by the designer (one or two at a time) or in a “batch” mode by selecting a breeding population.

With the breeding of a new child the cycle is complete. ParaGen is a non-destructive, dynamic population GA (NDDP-GA). This aspect is explained below and in more detail in another recent paper [5].

3. SELECTION OF PARENTS

This paper looks in more detail at the selection methods available in ParaGen. This is one aspect where the program differs from traditional GAs. There is a choice of two types of selection: 1. Fitness selection, and 2. Human selection.

3.1. Fitness selection

Examples of selection using a fitness function can be found in earlier work cited. In this paper some of the more atypical features and the multi-objective capability will be highlighted.

Fig. 3. An example using SQL filters and sorts to define a dynamic population. Shown is a partial set where NumberOfMembers < 100 AND SystemHeight > 6 ft AND ModalFrequency > 3.0 Hz sorted by least Deflection_R

Populations are a feature common to all GAs. Parent solutions are selected from a population to breed children solutions using some crossover operator. Generally, in each generation the better solutions remain in the population and the poorer performing solutions die out. In this way successive populations evolve toward the fittest individuals. One drawback with this approach is that occasionally a good solution may be killed off and never be seen by the designer. Usually, because of the large number of solutions generated, only the final population or the end results are available for review by the user. This means that interim solutions, which do not survive, are never seen. These terminated solutions presumably did not perform well against some predetermined fitness criteria, and therefore did not survive. But the trouble is that in many design problems the criteria by which we judge performance may not be so well defined or may change as more about the problem is learned. In fact we often determine some of these criteria through an initial discovery phase. This is just the phase where ParaGen, as an exploration tool, is most useful. So it is worthwhile not to destroy these early trials even though they might not initially be seen as fruitful.

ParaGen is based on a non-destructive dynamic population GA (NDDP-GA) [5]. Rather than killing off less fit solutions, they are entered in a SQL database where they can always be revisited or inspected if criteria change or new combinations of criteria are seen as significant. The SQL database contains both all of the initial parametric variables (the original chromosome of the solution) as well as a list of performance values that were determined when the solution was evaluated. In addition other forms of information that can be useful to a design decision are retained as well. These naturally include images of the solution, including images that can describe performance, e.g. deflection diagrams, stress plots, lighting levels, or even animations or acoustic files. All such files are keyed to the solution id in the database, and can be viewed and compared in a post processing phase. Sets of image types can be selected (switched out) in order to study specific features. Fig. 4 shows an alternate image view of Fig 3. The images used in Fig. 4 show axial force with compression as blue and tension as red. Other analysis images can also be alternately displayed for comparison.

Fig. 4. The same selection as shown in Fig.3 but with an alternate image using axial force plot from STAAD analysis

Because the ParaGen method is non-destructive, all past solutions are always potentially available for breeding. This eliminates any reason to generate duplicate solutions, since the chance of breeding any solution can be determined by performance values rather than its number of occurrences in a population. ParaGen creates breeding populations dynamically from the solution database as a whole using SQL queries. This allows complex, multi-objective searches that can even be fine tuned while a run is in progress. For example in the project shown below a breeding population might be created by using the query: NumberOfMembers < 100 AND SystemHeight > 1.8 m AND ModalFrequency > 3.0 Hz sorted by least Deflection_R LIMIT 40. This will yield a certain population from which a parent is randomly selected. SQL queries are very flexible and powerful ways to filter large sets of data and return a resulting subset. ParaGen incorporates a full set of SQL sorts and filters on all parametric parameters as well as performance values. These are used to dynamically build the breeding population for each parent as needed. By controlling the LIMIT
statement, the search can start off general (large LIMIT) and gradually choke down to a smaller population of more fit solutions toward the end of the run (smaller LIMIT).

### 3.2. Human selection

Human selection, where the designer selects parents based on visual evaluation, is also possible, and is the primary approach used in the example below. Normally, human interaction would not be the first choice for selection since designer preferences can usually be expressed in terms of the performance values. Presumably if there are particular aspects of the performance that influence the form of the design, those performance values would be obtained through some simulation and analysis so that the forms could be more accurately evaluated. Also using the SQL queries as described above allows the designer to sift through 100’s or 1000’s of solutions with far more reliability and less tedium. Nonetheless, there are some situations where selecting solutions visually can be expedient, and yield at least an initial survey of possible solutions.

There are two ways in which human interaction can be used to make selections for breeding. The first is by direct selection of a pair of parents, and the second is by selecting a whole breeding population. The first approach is very straightforward and requires the designer simply to click on a pair of images of solutions in the main display. Fig. 5 shows the result of such a selection. Two parents have been chosen and are ready to breed. By next clicking on the breed button below the pair a single child is produced. If only one parent is selected, then that solution is mutated to create a new child solution. If no parent is selected and the breed button is clicked, then a random mutant is created. Although the breeding is simple and quick, the generation of the child geometry in the parametric modeler may take some time. Likewise, depending on what is desired, the evaluation step may also be time consuming so that the delay between breeding and seeing the finished child with performance data included may be upward of 30 min. This is rather too long to be considered interactive. Besides, there is no guarantee that the resulting child will be anything desirable. As in all GAs good results tend to evolve through the breeding of many solutions.

![Fig. 5. A selected pair of solutions ready to breed](image)

For these reasons the second human selection method was developed. This approach can be seen as “batch” breeding. After making the selections the designer allows the run to continue unattended for some period until a number of solutions is generated. Any size population can be selected as a breeding population, and the process can be repeated with new populations as often as seems productive. In the following example the process was repeated twice using selections shown in Figs. 6 and 8.

### 4. EXAMPLE RUN

To illustrate the selection methods an example is now shown. The problem came from a student project by Mark Wright and Matthew Jensen at the University of Michigan that included the design and construction of a fabric form inflated with a polyurethane foam. Through physical experimentation, the students had a good sense of what could be constructed and also had an idea of the type of form they were looking for (visually). The designed object was to be an example of the construction method, and had no function beyond a sculptural object.

#### 4.1. Problem setup

The students initially wrote a parametric Python script to run inside of Rhino. It read the parametric variables from a CSV file which was the result of breeding two parent solutions. The CSV file was downloaded from the web server to a client PC and read by the Python script.

![Fig. 6. The first visually selected breeding population for Phase 2](image)

The script then generated the form as well as created a perspective image and a short animation (rotating view in 3D). A first run was made actually without the finite element analysis – simply images of the form. Populations in this case were chosen visually by the students. This was done partly for expedience, but also because the member sizes were limited by the construction method of sewn fabric tubes.

The run was executed initially in four phases based on human selection and one last phase using set fitness functions.

<table>
<thead>
<tr>
<th>Phase</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Random generation (Fig. 2)</td>
</tr>
<tr>
<td>2</td>
<td>Breeding of first selected population (Fig. 6 - 7)</td>
</tr>
<tr>
<td>3</td>
<td>Breeding of second selected population (Fig. 8 – 9)</td>
</tr>
<tr>
<td>4</td>
<td>Mutation of single selection</td>
</tr>
<tr>
<td>5</td>
<td>Breeding based on given fitness function</td>
</tr>
</tbody>
</table>

3
4.2. Procedure

With the Python script in place the variable list was coded into the breeding program on the server and a SQL database was put in place.

Fig. 7. Sample solutions bred in Phase 2 from the population in Fig. 6

4.2.1. Phase 1 - Random generation

The breeding program was set to initially breed about 100 random children which were run through the Python script and uploaded with images back to the server.

Fig. 8. The second visually selected breeding population for Phase 3

4.2.2. Phase 2 - Breeding of first selected population

After some solutions had been recorded, a selection of 15 individuals was made that had the desirable characteristics. This selection was made visually based on the experience of the designers. Fig. 6 shows this selection.

The run then continued using this selected set as the population pool from which parents were chosen. Fig. 7 shows a sample of children breed from this first selected population. The difference between these and the initial random children shown in Fig. 2 can be readily seen. The designers were looking for solutions with fewer members and some height (not flat). The bred solutions definitely tended more in this direction.

4.2.3. Phase 3 - Breeding of second selected population

After about 60 more children had been bred from the first selection, a second selection was made. This selection was a little smaller; 9 solutions, and mostly from the newer solutions that had more of the desired characteristics. This second selection can be seen in Fig. 8.

Fig. 9. Sample solutions bred in Phase 3 from the population in Fig. 8

Using the second selected population (Fig. 8) as parents, the run continued, producing another 260 solutions. Fig. 9 shows a sample of these. Here one can see a further tendency in the direction of the designers’ selections. The members have generally fewer members and less often flat. The plots show in Section 5. Conclusions, also show the shift in the solution space being developed.

4.2.4. Phase 4 - Mutation of single selection

Finally, a single solution was selected as closest to the desired result. This was solution number 239 shown in Fig. 10. This solution was set to mutate further children to get solutions close to this but a bit different. In this way the solution space closest to a given solution can be explored in detail. In the mutation mode, each variable is allowed a 50% chance to mutate. If the variable is selected for mutation, then its value is changed by some small random amount. Fig. 11. shows some of the mutated children from solution 239.
4.2.5. **Phase 5 - Breeding based on given fitness function**

In order to provide a comparison with the human interactive selection, a fitness function was established and used to generate a further set of solutions. The fitness function based on a SQL query was: `SUM(MemberLength) < 50 AND SystemHeight > 5 ORDER BY ModalFrequency DESC LIMIT 40`. This focused on solutions in the same range of members and height as selected by the designers, plus it added the performance parameter of stiffness using ModalFrequency from the FEA. A sample of results is shown in Fig. 12.

4.3. **Post run exploration**

In the end of course the designer could simply pick a solution to use from the full display, but if solutions range into the 1000’s in number this may not be so effective. As described in Section 3.1 above and shown in Fig 3., SQL queries can be used to filter and sort the full solution set to show only a particular range of results. This is a very good way to explore the end results and compare similar solutions on the screen visually. Because the information is all contained in the SQL database at this point, results appear on the screen almost instantaneously. This makes the human interaction very effective in exploring solutions.

![Fig. 10. Solution 239 used to spawn similar solutions through mutations in the final generation Phase 4](image1)

![Fig. 11. Samples of Phase 4 based on mutations from solution 239 shown in Fig. 10](image2)

![Fig. 12. A sample of results from Phase 5 using a fitness function](image3)

![Fig. 13. The plotting function used to find desired solutions](image4)
5. CONCLUSIONS
This paper shows how ParaGen can be used effectively as a design exploration tool. In particular the use of designer selection based on visual qualities of the solutions has been demonstrated. There are certainly cases where qualities of the design are visually recognizable, but difficult to define through computational means. In such cases the human selection mode available in ParaGen makes it an effective means to find good solutions.

In this problem the designers had particular desires for the solution: a height just a little over head level, buildable in a limited time – therefore fewer members, and an “interesting” form.

These qualities were relatively easy for the designers to recognize in the images, but could not entirely be described computationally. In order to investigate how the selection of certain solutions affected further generation of forms, a series of plots were made after the run, which show where the different phases of the run fell in the solution space. Fig. 14 shows a comparison of Phase 1 the initial random generation, to Phase 2 the first breeding phase, to Phase 3 the second breeding phase. The concentration of solutions definitely shifts, and is more clustered in the second and even more in the third phase in response to breeding based on the selected populations.

Fig. 15 shows a comparison between the solutions in Phase 3 with solutions in Phase 4. The solutions here have a lower number of members, so that the plot scale is changed from Fig. 14. Again a further shift is seen in the area of the solution space being sampled. Looking at the results (Figs. 9 and 11) gives a visual depiction of the different concentrations.

Finally, Fig. 16 shows a comparison between solutions in Phases 4 and 5. Here it is seen that approximately the same solution space is being sampled. This shows that the visual based human selection gave very similar results to the algorithmic, fitness based selection. Again, observing the visual results in Figs. 11 and 12 bears out this conclusion.

6. REFERENCES