Breeding Topology
Special Considerations For Generative Topology Exploration Using Evolutionary Computation

Peter von Buelow
Dr.-Ing., University of Michigan, TCAUP

TOPOLOGY OPTIMIZATION OF ENGINEERING STRUCTURES HAS LONG BEEN A TOPIC OF RESEARCH SCRUTINY. Many methods have been successfully developed for the determination of continuum structures. Some of these techniques, for example the homogenous method, have also been adapted for use with discrete structural frames or trusses. Most commonly the topology optimization of truss structures is carried out with the aid of a ground structure, a simple raster that describes potential joint locations. Although this simplifies the computation, it greatly limits the range of potential solutions that fit the gridded raster. Additionally, when using Evolutionary Computation (EC) methods, the level of computational intensity increases exponentially with the size of the ground structure making anything above a very modest level of complexity impractical to process.

This paper demonstrates several practical techniques that can be used with EC, and more specifically Genetic Algorithms, when applied to topology exploration of discrete structures. First a method of chromosome coding that avoids the use of ground structures is shown. Then specific genetic recombination techniques are illustrated that are well suited for breeding different topologies. The combined techniques are demonstrated in a topology design problem. The paper concludes with a discussion of advantages of EC over traditional optimization methods in the area of overall form design.
1 Introduction

Techniques of Evolutionary Computation (EC) have been successfully applied in recent years to aid in the exploration of architectural (in particular structural form). These techniques offer several advantages to designers which point to the distinction between design exploration versus traditional engineering form optimization (von Buelow, 2008). This paper offers some EC techniques that would allow architectural designers to explore more complex forms without imposing restrictions on the potential solutions due to the computational method.

In the traditional coding of Genetic Algorithms (a subset of EC), operations of selection, recombination and mutation act on string based ‘chromosomes’ to adaptively alter forms toward better or fitter individuals. This takes place in a series of cycles operating always on populations of ‘parents’ which produce ‘children’ in successive generations. By selecting the next generation of parents based on an evaluated fitness to some criteria, the population evolves toward more fit individuals (Holland, 1975; Goldberg, 1989). Figure 1 shows a schematic of this process based on a particularly efficient GA developed by Larry Eshelman (1991).

In order to use a Genetic Algorithm (GA) to explore or generate structural form, it is necessary to distinguish between geometry and topology. Figure 2 illustrates this distinction with an example using simple trusses. The exploration of visual form involves both different topologies and different geometries with the same topology. For designers, the exploration of both different geometries and different topologies is an important part of finding a good solution. There is an inherent advantage in GA based methods because the process itself involves the generation of a range (population) of solutions.

2 What are Ground Structures?

One method used to code topology and geometry for the GA chromosomes is to use binary strings based on a ground structure. In Figure 1 these chromosome strings are illustrated with black and white banding which represents the genetic coding. Ground structures have been used since Dorn et al. (1964) proposed their use for topology optimization of plane trusses. Dorn describes a ground structure as the set of all admissible bars which can be considered to connect a set of admissible joints. Ground structures were first applied in direct computational methods like Linear Programming. Figure 3 shows an example by Klarbring et al. (1995) of a ground structure used in a multi-objective variation of Linear Programming to design a simple bridge truss. On the left in the figure, all possible
In using ground structures with GA optimization methods, both geometry and topology are usually treated together. For reasons discussed in the next section the string length used as a chromosome is usually quite a bit shorter than Klarbring’s 21945 (Klarbring did not use EC for the analysis). The shorter length however has drawbacks which limit the complexity of the problem as well as the quality of the result. Figure 4 shows how a
The ground structure used in GA optimization affects results. The example is based on work published by Dep & Gulati (1999). The center truss in Figure 4 was found by using a GA to determine the lightest structure by selecting members from a set of 39 admissible bars which comprise the ground structure shown to the left (some bars overlap). The geometry of the truss is locked into the coordinates of the ground structure. This means there are only a limited number of points (in this case 12) where the joints can be placed. The truss on the right in Figure 4 has the same topology as the center truss, and was optimized for the same conditions except that the joints were allowed to move free of the ground structure. Comparing the weight efficiency of the two shows that the truss based on the ground structure is 36% heavier than the one without the ground structure. Comparing the visual appearance makes the difference even more pronounced.

But one begins to realize the real difference that using the ground structure makes when one compares the range of different possible topologies found for the same problem but without a ground structure. The trusses show in Figure 5 are based on similar loading and span conditions as described above, but in this case, the solutions were found without the use of a ground structure (von Buelow, 2007). Here it is immediately obvious how restrictive the ground structure in Figure 4 is. The ground structure topology found as the ‘best’ in Figure 4, does not even come into consideration in the set of ‘pretty good’ solutions found in Figure 5. For designers, using a pallet of solutions like the one shown in Figure 5, it is possible to attain a solution that Herbert Simon would call “satisficing” (Simon, 1969).

By inspecting the range of solutions beyond the restrictive ground structure, the designer may use criteria other than the coded objectives in choosing a solution. Seeing a pallet of solutions, as shown in Figure 5, can actually make the designer aware of non-coded criteria, such as aesthetics, that can then be added to the selection criteria.

2.2 EFFECTS OF GROUND STRUCTURES ON COMPUTATION

Of course, the solution found by Deb in Figure 4 could be much improved if the raster of the ground structure was increased. Deb used 12 points, carefully eliminating the corners and center from a 3x5 grid of 15. This was done to decrease the computational load of the problem. In running a GA, the computational intensity correlates with the length of the chromosome string used to describe the individual structure. So the fewer bars the faster the problem will run. This is because the population size used in the GA must be increased as the string length gets longer. So to go from a chromosome string length of 39 to a length of 21945 as was used in Klarbring’s ground structure in Figure 2, is considerable. And the string length for the higher resolution ground structure would still be this high, even though the members actually used might only number 17 as in Deb’s solution. Just moving Deb’s truss from his 39 bar ground structure to the 21945 bar ground structure of Klarbring, would cause the chromosome string length to increase to 21945. The difference in the
two strings would be the ratio of 1's to 0's, i.e. 17:22 with the small ground structure and 17:21928 with the larger ground structure. But this ratio makes little difference to the computational burden. The total string length is the critical factor—39 for the one and 21928 for the other.

3 Doing it without ground structures

Thus, Sections 2.1 and 2.2 lead to a dilemma. To get good solutions the resolution of the ground structure needs to be high, but to have reasonable computational requirements; the resolution needs to be low. The better solution is to avoid ground structures altogether.

3.1 GENETIC CODING

As mentioned in section 2.1, the solutions in Figure 5 were not coded with ground structures. The chromosome string used for the geometry GA was based on the real number coordinates of the nodes. The topology was coded separately based on the upper triangular of the incidence matrix. This is basically how finite element codes describe geometry and topology.

Figure 6 illustrates how a chromosome string can be coded for topology based on the incidence matrix. By sorting joints with fixed locations to the beginning (in this case 1, 2 and 3) they can remain constant during breeding or mutation. Figure 7 shows a crossover breeding mechanism for two different topologies having different length chromosome...
strings. Care is taken to choose the cut point past the fixed joints and before the end of the shorter chromosome.

3.2 BREEDING GEOMETRY AND TOPOLOGY

With topology generated as shown in Figure 7, a geometry must be determined before the structure can be assessed. This is performed by a second GA running as it were within the outer topology GA. Of course other methods can be used to optimize the geometry alone, but the GA approach has the advantage of allowing better exploration by potentially allowing the designer to find different geometries which share a single topology (as in the top row of Figure 2). Figure 8 shows the breeding of geometry. The chromosome string is simply the coordinates of the nodes. Using the techniques shown in Figures 6 through 8 the GA string lengths, and thus the population sizes, are kept very low, while the resolutions of the solutions are very high (based on real number nodal coordinates). Population sizes for truss geometries of this size and up to about 20 members can be around 50. Topology populations seem to work at about half the size of the geometry population (about 20). These numbers are much lower than many researchers using ground structures. For example in the cited work by Deb & Gulati, the population size was 630.

4 A simple example

As a simple example, architecture students were given the task to re-design a small two lane bridge near campus. Using the actual 119 ft. [36 m] span and AASHTO lane and moving truck loads, the students used the authors GA based program to explore design alternatives. Figure 9 shows a pallet of ‘pretty good’ solutions found by the program.

4.1 USING THE TOOL

The goal for the students in using the GA tool was to search for an efficient bridge form. To do this, they could either base their design on suggestions generated by the program alone (like those shown in Figure 9), or use the program interactively to search for new form possibilities. Run in the interactive mode, the following sequence is used:

- the GA suggests a pallet of solutions on the screen
- the designer selects or adds new solutions to form a new set of parents
- the GA breeds these parents and displays a new selection pallet

This sequence continues as long as it profits the designer. In this way the program can be steered to search for solutions based on the designer’s visual criteria. In the end, a solution is chosen for further development. Figure 10 shows students using the interactive program and Figure 11 shows a final result.

4.2 SEARCH EFFICIENCY

The rate of convergence is most rapid in the early generations. Using the example problem, the rate of convergence during the first cycle of geometry optimization is shown in Figure 12. The graph shows that the solutions do converge to a better fitness (in this case less weight). Figure 13 shows actual geometries which relate to generations from the graph. The four geometries show the range of evolution from the random start geometry, to convergence of at the end of the cycle.

Further refinement of the geometry toward the end of the cycle can stagnate as the GA...
finds local optima (hill tops) in the search space. To overcome the problem of getting stuck on local optima, the adaptive plan of the CHC-GA proceeds in cycles. When a generation reaches a certain level of convergence, the cycle is terminated, and a new cycle is initiated by taking the last best solution and mutating it repeatedly to form a full population for generation 0 of the new cycle. Cycles end when no improvement is shown between successive cycles. A typical geometry cycle will have between 100 and 300 generations with 3 to 5 cycles in a run. For the geometry show in Figure 13, the middle truss in the top row of Figure 9 would represent the result after several cycles. In the interactive mode, new topologies are usually the goal. In that case, each topology shown on the screen is the end result of a geometry optimization.

The topology GA is similar to the geometry GA except that the quantities are generally smaller. For the above example, a topology population of 20 was used. There are usually about 30 to 50 generations in a cycle with 3 to 5 cycles in a run. A more detailed description of the method used by the author along with other examples can be found in the references (von Buelow, 2007).

5 Conclusion
This paper has given a brief overview of the use of Evolutionary Computation for exploration of architectural structures. It has specifically cited problems that can be encountered using ground structures in the coding of chromosome strings used in Genetic Algorithms. An alternative approach is offered to code both geometry and topology.

An example illustrates the use of the chromosome coding and breeding mechanisms described. This approach has advantages particularly for designers interested in form exploration. The GA was shown to converge on a solution within a few hundred generations. The program currently has capabilities to explore 2D and 3D truss structures. Further development continues to expand the program to include free form discrete structures.

6 References


