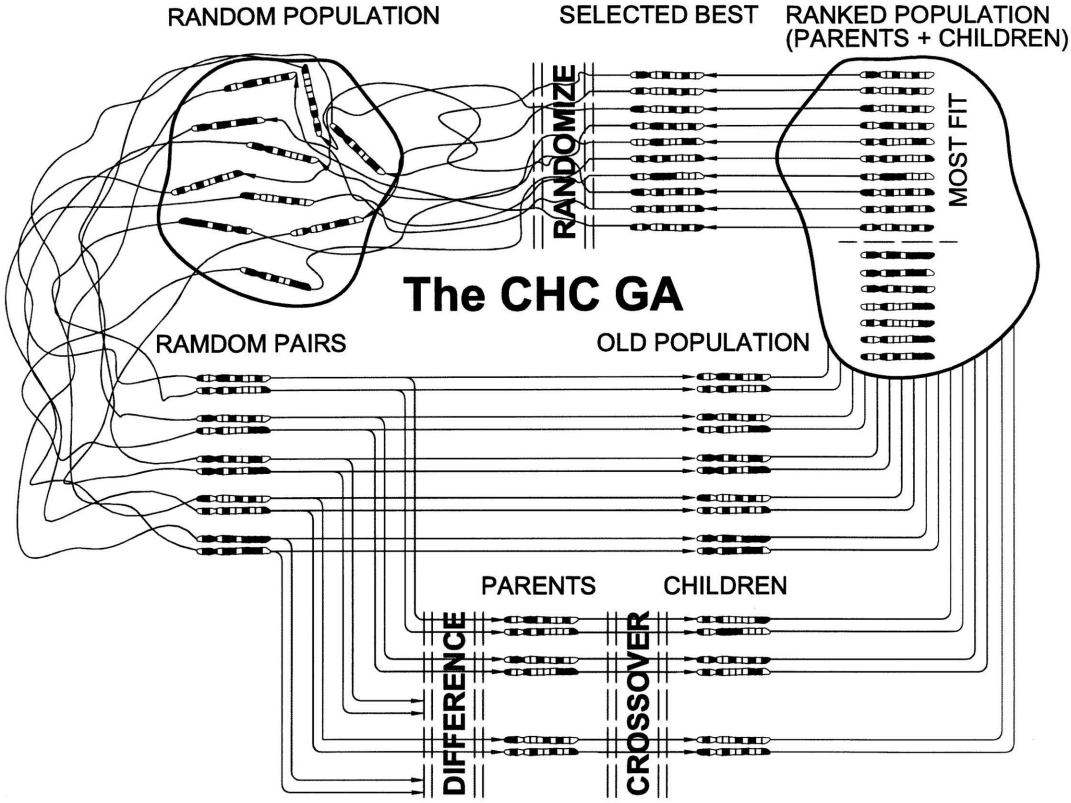
TECHNIQUES FOR MORE PRODUCTIVE GENETIC DESIGN:

EXPLORATION WITH GAS USING NON-DESTRUCTIVE DYNAMIC POPULATIONS

Peter von Buelow
University of Michigan
Taubman College



1 Schematic of the cyclic method used by the CHC GA

ABSTRACT

The products of generative design are ever more commonly explored and refined through evolutionary search techniques. Genetic algorithms (GAs) belong to this class of stochastic procedures, and are particularly well-suited to the way designers investigate a problem. GAs search by mixing and matching different parts of a solution, represented as parametric variables, to find new solutions that outperform their predecessors. Generally the method proceeds through generations of populations in which the better solutions out-survive their less desirable siblings. Inherent to this approach, however, is the fact that all but the select solutions perish. This paper discusses a non-destructive GA that uses dynamic populations drawn from a bottomless pool of solutions to find the most productive breeding pairs. In a typical GA the survival or destruction of a solution depends on a well-defined fitness function. By not enforcing the destruction of less fit individuals, the possibility is held open to modify the fitness function at any time, and allow different parts of the solution space to be explored. This ability is ideal for more complex multi-objective problems that are not easily described by a single fitness function. Generally, design presents just such a problem.

1 INTRODUCTION

A genetic algorithm, as originally described by John Holland, is a stochastic search method that progresses through repeated cycles toward solutions which have certain desired characteristics, namely a fit solution (John Holland 1975). A cycle in a GA is represented by a generation, and the solutions in a generation are all members of a population. The next generation is bred from the current generation by selecting the more fit solutions to be parents. The solutions are described in terms of chromosomes, where parametric variables are genes. These chromosomes are bred to form children solutions that inherit characteristics through the genes of their parents.

There are many different variations and methods for selection and breeding (Mitchell 1996). John Holland's original book, *Adaptation in Natural and Artificial Systems* (1975), followed the analogy to biological systems fairly closely. David Goldberg, who studied under Holland and Wylie at the University Michigan, did much to introduce the engineering community to the potential of GAs with his book *Genetic Algorithms in Search, Optimization, and Machine Learning* (1989), and more recently with *The Design of Innovation: Lessons from and for Competent Genetic Algorithms* (2002).

GAs have been applied to a wide range of search and optimization problems in fields of engineering and architecture, as well as areas such as drug design, scheduling and routing, financial prediction, data mining, system control, and even the composition of poetry, music and art (Goldberg 1998; Lund 1995; Beasley 1997). This broad scope of application is part of the attraction of GAs.

To solve a problem using a GA only a very few criteria must be satisfied. Firstly, the parameters that describe a solution must be coded as a genetic string or chromosome. Secondly, there needs to be some type of evaluation to determine the relative fitness of the solutions. Otherwise, very little needs to be known about the problem or the solutions. The process continues by cyclically applying certain genetic operators to generations of populations of solutions until acceptably good solutions evolve. The steps can be summarized as follows:

1. Coding the problem into a chromosome
2. Generating an initial population
3. Breeding through recombination
4. Determining fitness by evaluating the solutions
5. Selecting parents
6. Filling the population of the next generation

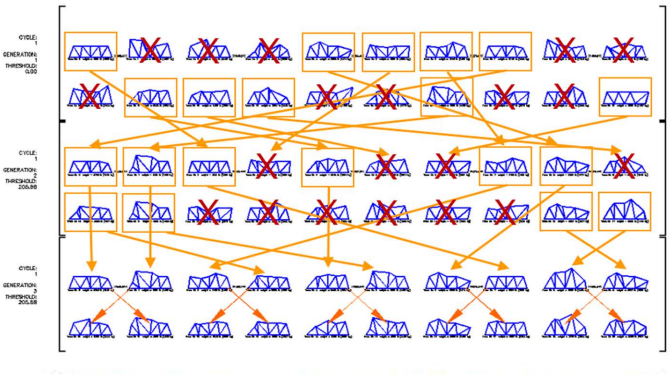
The procedure then cycles through steps three to six until satisfactory solutions are reached or the rate of improvement is diminished below a significant level. Figure 1 shows the cycle of a GA of this type. Unlike classic deterministic optimization methods which result in a single solution, the stochastic GA approach being population based, inherently results in a set of solutions. Depending on how the fitness function is defined, this solution set may converge to a single solution or represent different well-performing solutions. Multi-Objective Evolutionary Algorithms (MOEAs) are particularly interesting for designers, since design generally involves trade-offs of conflicting objectives. ParaGen as described below, makes use of multiple objectives both in generating solutions and, in an even more flexible way, in exploring solutions through post-processing, Pareto plots and limiting filters.

2. COMPARING THE DESIGN PROCESS TO THE GA CYCLE

2.1 PROPERTIES OF EARLY DESIGN

Part of the reason that GAs are particularly well-suited to design problems is the similarity of the GA method to the design cycle. In general, design can be described as a process that is purposeful, goal-oriented and creative. Many activities are purposeful and goal-oriented without being particularly creative. For example, standard engineering analysis or even optimization, where the goal is to simply find the single most compliant solution, is generally not thought of as creative design. In such cases the problem is completely defined from the outset so that there is only one possible best solution. A design problem is usually not completely defined at the beginning of the process, since part of the process is to gain knowledge about the problem. Design is not constrained by attempts to describe "what is", the single reality, but left totally open to describe "what ought to be." Herbert Simon, made this distinction regarding design: "The natural sciences are concerned with how things are. ... Design, on the other hand, is concerned with how things ought to be, with devising artifacts to attain goals" (Simon 1969).

It is also worth considering another aspect of design that Herbert Simon points out in a comparison he draws between "ill-structured problems" as compared to "well-structured problems". The distinction is that well-structured problems can be solved by general problem solving procedures in a straight forward manner, that is analysis, while ill-structured problems cannot be solved so directly. Simon builds the argument that all design problems are ill-structured because at the outset of the process the problem space is not fully specified. In fact, some parameters of the problem may only occur to the designer after considerable work and search. Simon outlines a procedure for solving these ill-structured problems (Simon 1973), and also recognizes that the approach is cyclic, and alternates between solving component parts of the problem and attaining and assimilating new information about the problem. Unlike analysis, it is not necessary for the process to be completely defined by constraints. In fact, Simon observes, "The more distinguished the architect, the less expectation that the client should provide the constraints" (Simon 1973).



2 Three generations of parents and children. Boxed solutions continue while the X'ed solutions die off

2.2 PROPERTIES OF GA

GAs can be seen to contain many of the same properties described above for early design. Firstly, although they contain stochastic procedures, they are purposeful and goal-oriented. They are oriented toward a specific goal as a fitness function, and not simply searching at random. But the operation of the GA is not dependent on the complete and explicit definition of the problem. Because of this they are well suited to solving Simon's ill-structured problems. In fact by allowing human interaction, the fitness function can be based on visual criteria (visually picking parents from a set of solutions), and not coded at all. Insofar as the designer is consistent in making the choice of parents, the children will gravitate in the direction of the designer's preference.

GAs operate in a cyclic manner also similar to design procedures. Since a GA gains knowledge about the solution space through direct sampling, the starting information is limited to that provided by the initial population. Through successive generations the knowledge of the problem increases and the scope of the search becomes more constrained. The directions explored particularly at the beginning of the cycle can be compared to creative aspects of design since they are minimally constrained and tend to be more wide-ranging and unpredictable.

GAs also promote creativity by avoiding fixation on a single solution. GAs inherently operate with populations of solutions. This is very different from deterministic techniques that only find a single solution. Because GAs constantly explore sets of solutions it is easier for the designer to make comparisons or see positive aspects in different solutions. It is hard to make a comparison using a tool that only returns one solution. A GA, on the other hand, by always supplying a palette of solutions, requires the designer to continually view different possibilities. In the process of considering arrays of solutions, creative thinking is stimulated, and the likelihood of finding a new or unexpected solution increases. Optimization tools that only reveal a single solution run the danger of design fixation. By always displaying an array of possible solutions, the possibility of design fixation is lessened in a GA.

3 LIFE AND DEATH CYCLE

3.1 SURVIVAL OF THE FITTEST

GAs generally advance from generation to generation through a "survival of the fittest" procedure. Figure 2 shows the first three generations of a GA that searches for lighter truss forms. Each generation is composed of two rows with parents directly above their children. Each boxed solution survives and each X'ed solution dies out. The arrows show how the surviving solutions in one generation become parents in the next generation. The third generation simply shows the resulting parents and children before a selection is made.

There are many different ways to make selections. Examples include selection by tournament or by roulette wheel, and through rank-based or elitist processes (Deb 1997). It is important to keep in mind, of course, that selection is only half of the game. To produce a new generation requires breeding or recombination of the parents to form children. Oftentimes the goal of the search is simply to find the best-performing individual in as few generations as possible. But this is seldom the only goal in architecture, particularly in early phases of design. Design problems also require an exploration of possible solutions — a comparison of different solutions that perform well. The goal is generally more complex, multi-objective and ill-structured. In these cases there is actually a danger in killing off the poor performing solutions. There are a few reasons why any solution could at some point be useful.

For one thing, the performance of solutions reflects a decision of what is desired, what is thought to be good. However, at the outset of the problem there may not be much information on which to base that decision. The definition (the fitness function) of what is good may be flawed. In that case a truly good solution could be lost, and not be able to provide the designer with a clue to better understand the problem. Also, as the problem progresses it may be useful to shift or change some of the fitness criteria. Here again, if an early, poor performing solution was eliminated from the solution set, it would be unavailable for reconsideration under the changed criteria. Then too, even bad solutions sometimes increase knowledge about what makes a solution good. So being able to see the full range of performance response can be helpful. Finally, by not killing off bad solutions, it is possible to gain a better picture of the overall range of performance possible in the problem. If you only see twenty solutions at a time it is very different from an overview of 2000 when judging the range of performance and the sensitivity of geometry to different performance characteristics.

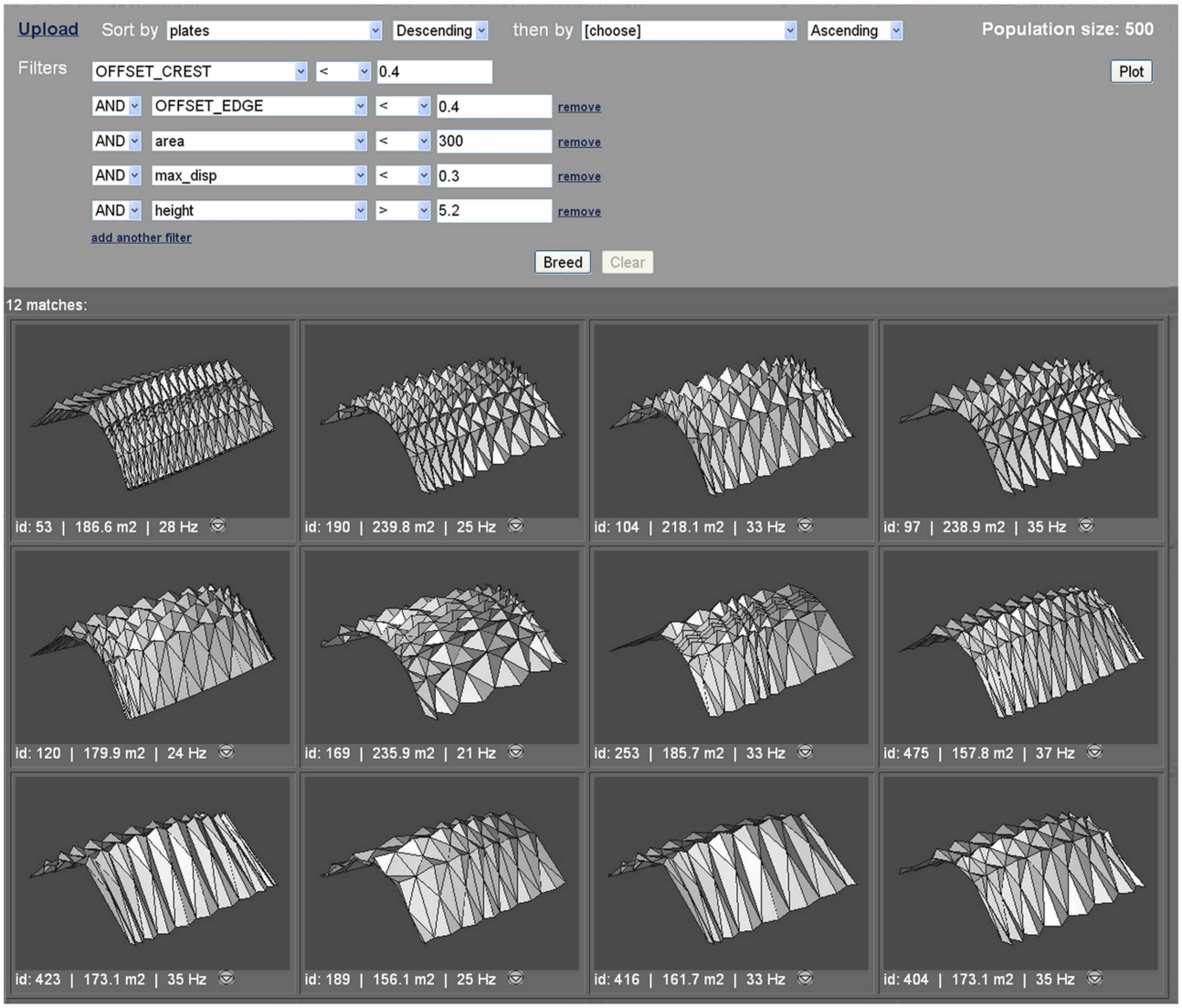
3.2 BREEDING OF THE FITTEST

My proposal is to kill no solutions. Let them all survive in perpetuity. I call this a non-destructive GA. Then, in order to drive the species (species being the level above generations of populations) toward better solutions, selective breeding is used. This means selecting the better performing parents from the entire gene pool of the species. Where, normally the defining data for a population is held in an array, the data for a whole species is best contained in a database. The database can also contain more than just a single fitness value; it can hold almost any amount of performance data that can be used to describe the solution. In addition a database can easily reference other types of data such as images, sounds and video.

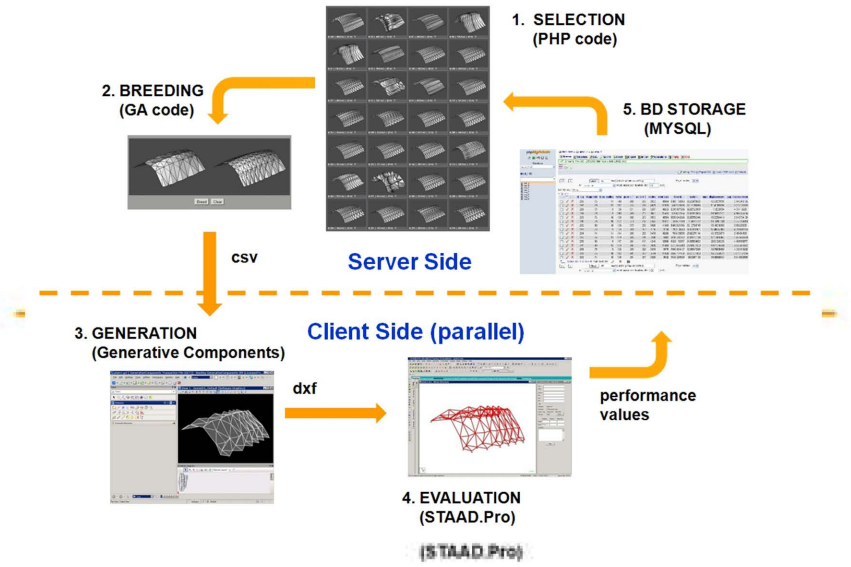
Since a database can access any of the contained solutions, there is no reason to have duplicates —they can be eliminated. There is no value for the designer in viewing duplicate solutions. As each solution is bred it can be checked for uniqueness, and if a solution with the same genetic code is already in the database, the parents can be re-bred to find a new solution. Normally, GAs process many duplicate solutions because there is no record of what solutions have already been tried and eliminated. This is one aspect that causes a GA to be relatively slow and inefficient. By only breeding different solutions the method as a whole becomes more efficient.

3.3 STEADY STATE SELECTION AND DYNAMIC POPULATIONS

Since the breeding pairs are taken as they are needed from the entire gene pool (the database of all solutions), a generation of parents no longer exists in the same way as in a traditional GA. Selection of parents proceeds one after another, at a steady rate, rather than cyclically in generations. Selection of parents is still made from a limited population of solutions, but the population is



3 ParaGen solutions with several filtering parameters and a sort by the number of plates



4 A schematic of the ParaGen cycle showing examples of incorporated programs

created dynamically for each selection through filtering queries made to the database. In this way two parents can actually come from different populations. For example, one parent can be taken from a population of least weight and the other taken from a population of least deflection. Using SQL queries, relatively complex populations can be created which allow multi-objective breeding. In fact, as the problem is running (usually several hours) the population definitions (SQL queries) can be modified, which shifts the area in the solutions space that is being searched. This can be used to refine, expand or focus the exploration of the problem.

3.4 POST PROCESSING ANALYSIS

In a traditional GA, when the search converges there is only the last population to inspect (the rest have not survived). In the process described above, after a run (or actually even during a run) is when the real exploration begins. Since all solutions are at that point stored in a database, it is possible to filter, sort and display images of the solutions in a variety of ways. The solutions can be sorted by any of the parametric variables or any of the performance results.

Figure 3 shows a sorted set of solutions which meet certain performance criteria. The post-processing interface includes all of the standard SQL queries. Database queries provide very powerful and fast ways to explore large data sets using multiple parameters. The solutions can be filtered by any number of parametric parameters or performance values, and then displayed after being sorted by some parameter. In Figure 3 the set of twelve solutions shown was extracted from the larger database by constraining two of the parametric variables (OFFSET_CREST 0.4 and OFFSET_EDGE 0.4) and placing limits on three performance values (area of plates 300 m^2 and maximum displacement 30 cm and height 5.2 m). Then the set was sorted by the number of plates (facets). Using sorts and filters in this way allows the designer to instantly create a viewable selection of solutions defined by any set of parameters or performance values. Because the solutions already exist in the database, the interface is very rapid resulting in an interactive tool to explore the solution space.

Performance data can also be plotted to study distributions and relationships of different parameters such as Pareto front formation (Figure 10). These concepts have all been included in a system developed at the University of Michigan called ParaGen. The following section gives an overview of the system and an example run.

4 PARAGEN – THE PROGRAM

4.1 OVERVIEW OF THE METHOD

The ParaGen system, developed at the University of Michigan over the past three years, is based on a non-destructive dynamic population genetic algorithm (NDDP GA) as described above. It is designed to run on any cluster of Windows based machines linked to a server through the Internet. The interface is a web page that is publicly accessible, and can be used to view and explore the database developed by the system (von Buelow 2012).

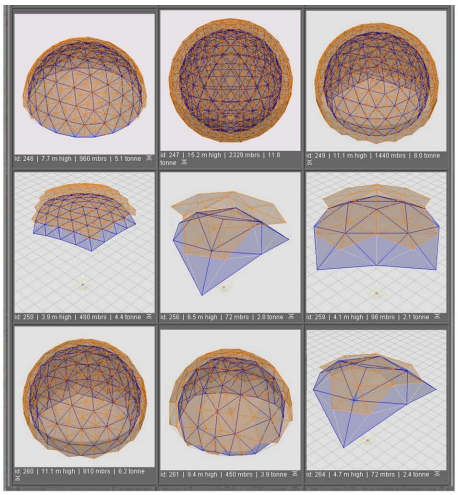
Figure 4 shows a schematic of the cycle. The overall scheme of the system follows a five step procedure similar to a GA:

1. Select - either with human interaction or by the GA
2. Breed - GA program on a web server
3. Generate - the geometry using parametric software
4. Evaluate - with simulation software, for example FEA
5. Store - store the results in a database for selection – on a web server

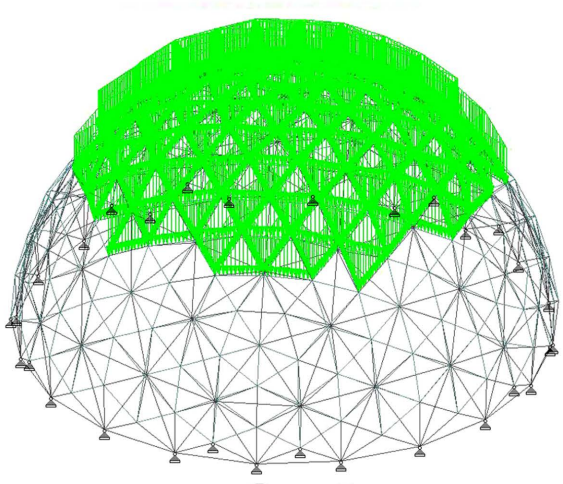
ParaGen combines both custom written code and scripts with commercial and public domain software to execute the different steps in the cycle. The software modules used in steps three and four can be interchanged to suit either the problem or user's preference. For example, in step three, the user could use any parametric program such as Generative Components by Bentley Systems, Digital Project by Dassault Systemes or Grasshopper by Robert McNeel & Associates. This could be combined in step four with any desired analysis tools for evaluation such as finite element analysis (FEA), computational fluid dynamics (CFD) or any appropriate lighting, acoustic, thermal or behavioral simulation software. In the fifth step, the results of the analyses, along with a description of the geometry, are then uploaded for storage in a SQL database over the Internet. Any number of client machines can be clustered to work on a problem in parallel. All that is required is an Internet connection and whatever commercial software is being used. Further details of each step are described in other articles by the author (von Buelow 2012; Turrin, et al. 2011).

4.2 AN EXAMPLE RUN

Basically, any parametric geometry can be explored using the ParaGen system. To date, a variety of types including towers, bridges, domes, vaults, branching structures and pneumatics have been investigated. All that is really required is for the form to be described by some parametric software. The analysis is also free to take any form including simple visual selection by the designers. For this example I chose a run we made recently to explore a variety of geodesic dome forms. The parametric script was written in Generative Components, and the structural analysis was made using STAAD.Pro, both by Bentley Systems.



5 A random selection of solutions showing the variation in the set

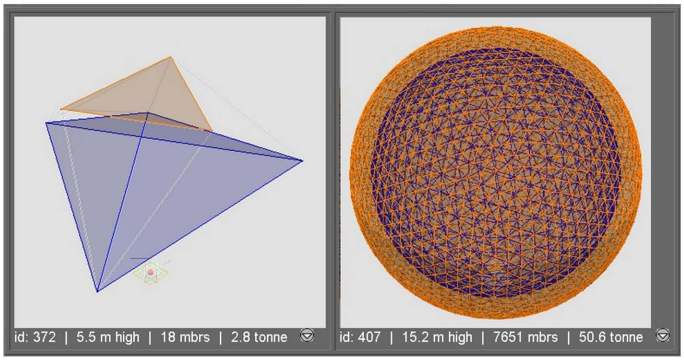


6 Example of the uniform snow load distribution on a dome based on the panel slope

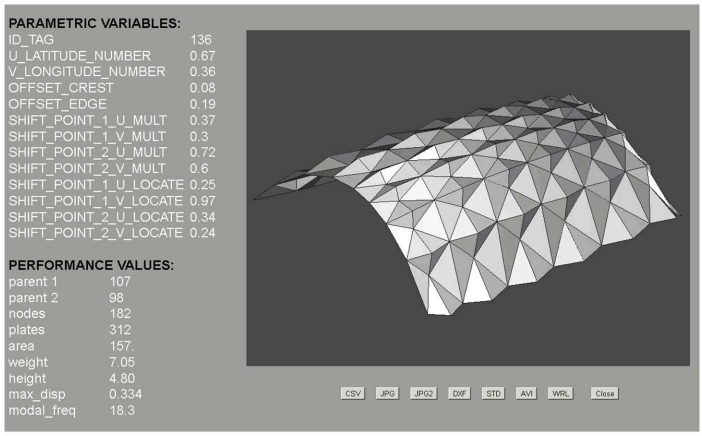
Geodesic domes have been part of the architectural vocabulary since they were popularized by Buckminster Fuller in the 1950's and 1960's. The type investigated here consists of a double layer space truss. The outer shell is based on one of three types of polyhedra. The inner shell is the dual of the outer. Altogether the GC script uses five variable parameters to control the dome topologies:

1. Primitive Type - outer shell (inner is dual)-tetrahedron octahedron icosahedron
2. Power - number of subdivision generations
3. Frequency - subdivisions of primitive face
4. Trim - cut section from a whole sphere
5. Radius Multiplier - distance between layers

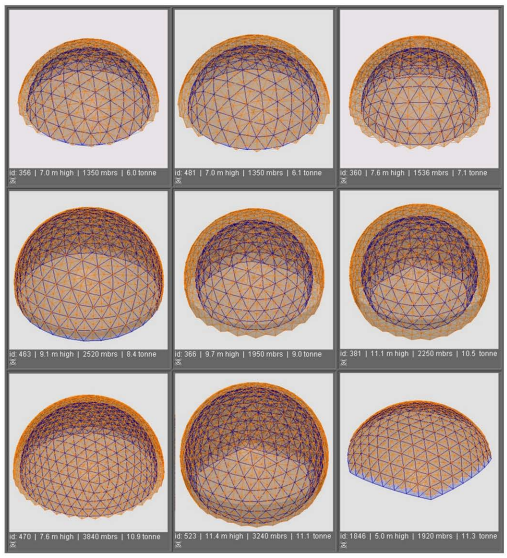
The number of sub-triangles per original primitive triangle is given by the formula: frequency^(2*power). With the ranges and constraints assigned to each parametric variable there are 418,500 possible combinations. Figure 5 shows a small selection from the range of possibilities.



7 Two "domes" showing the extremes in terms of member count included in the solution set.



8 A detailed view of one solution



9 A selection of solutions derived using filters and sorts

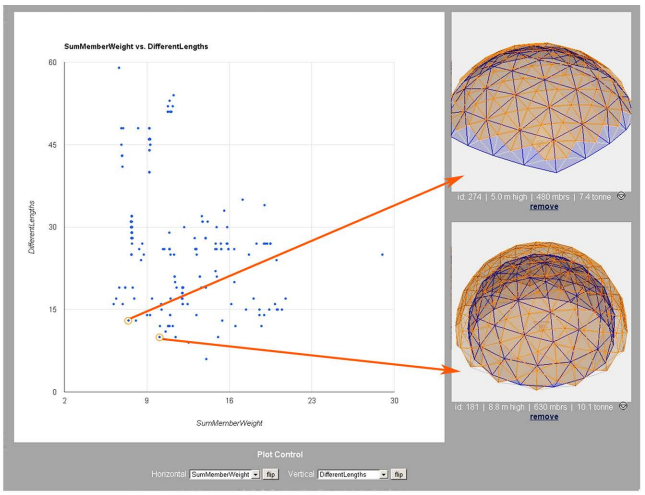
Each dome was scaled so that the maximum diameter was equal to 15.2 meters. After the individual solution was generated in GC, the geometry was transferred to STAAD for the structural analysis.

In STAAD.Pro the forces and member sizes were determined based on a snow load of 42 psf (2.0 kPa) with the appropriate reduction for panel slope as per ASCE-7 (Figure 6). Members were sized based on the US steel code (AISC ASD), and the total weight of the structure was determined. The natural (modal) frequency based on the final member sizes was also determined. The data recorded for each solution included: numbers of members, nodes and plates, number of different member sizes and lengths, number of supports, surface area, weight, modal frequency, total member length, maximum member length, and average member length. These data were used to both direct the search (multiple fitness functions) and to explore the database after the run.

The SQL filters were used during exploration to limit the range of dome geometries viewed after the run, rather than to limit the range of the parametric variables during the run. In this way, judgment of what is good or bad is deferred until after the solutions exist. Under different combinations of variables some of the resulting dome geometries would not be practical in terms of an architectural solution. For example, Figure 7 shows the two extremes in terms of number of members. The solution on the left has too few members to function as a dome, and the solution on the right nearly forms a complete sphere with minimal floor area for the volume. These extremes can be visually investigated after the run to estimate workable limits based on the images of the solutions. The ParaGen interface also includes a small arrow icon below each image. By clicking this icon the full list of parametric variables and performance parameters is revealed (Figure 8). Also, a complete list of additional downloadable files is provided (for example all analysis files, CAD files and jpg images). The thumbnail image used in the view of the solution set can also be changed to any image collected during analysis such as stress plots, deflected shape, or solar shadows.

Other parameters can also be used as filters to focus the solution set. For example the actual member length impacts both the structural performance (slenderness of members under buckling) as well as constructability. It also correlates with the number of members since the diameter of each solution is scaled to be the same. Figure 9 looks at a selection of solutions which were filtered based on member lengths less than two meters and then sorted by weight. The members were all sized in STAAD for the same 42 psf snow load. Looking at the set visually as well as having other performance data quickly accessible below each image or through the arrow icon, a designer will quickly be able to give further interpretation to the various solutions, and modify the search filters to explore more in one direction or another.

Like in any design problem, tradeoffs in criteria are necessary to find a good overall solution. To study the effects of multi-objectives, plots can be made of the solutions to find either a Pareto front or see how the criteria relate. Figure 10 shows a plot that compares the number of different member lengths (vertical scale) with the total weight of the structure (horizontal scale). Hovering over the points reveals the data, while clicking the points brings up an image of the solution. The lower left boundary can be seen as a Pareto front. The two solutions selected are at different points on this front and both represent good solutions, but with different tradeoffs. Again the arrow icon below each image gives access to further data and files.



10 A plot of two different objectives resulting in a Pareto front

5 CONCLUSIONS

In this paper I have described a different approach to constructing a genetic algorithm called non-destructive dynamic population (NDDP GA). The method has advantages in solving problems centered on design and exploration of structural geometry. An application of the method was shown in the ParaGen system. The method is successful in finding good solutions and allowing a more detailed exploration of the design space based on integrated performance parameters.

ACKNOWLEDGEMENTS

I would like to thank Taubman College at the University of Michigan for support of the ParaGen project. Theodore Hall, at the Michigan 3D Lab, developed the parametric dome model in Generative Components. And Bentley Systems most generously provided free educational licensing to both GC and STAAD.Pro.

WORKS CITED

Beasley, David (1997) Possible applications of evolutionary computation, in: Bäck, Thomas; Fogel, David and Michalewicz, Zbigniew. (eds.). Handbook of Evolutionary Computation. Oxford and New York: Oxford University Press.

Deb, Kalyanmoy (1997) Selection - Introduction, in: Bäck, Thomas; Fogel, David and Michalewicz, Zbigniew. (eds.). Handbook of Evolutionary Computation. Oxford and New York: Oxford University Press.

Eshelman, Larry (1991) The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination, in: Foundations of Genetic Algorithms. San Mateo, California: Morgan Kaufmann Publ.

Goldberg, David (2002) The Design of Innovation : Lessons from and for Competent Genetic Algorithms. Boston: Kluwer Academic Publishers.

Goldberg, David (1989) Genetic algorithms in search, optimization, and machine learning. Reading, Mass: Addison-Wesley Publishing Company, Inc.

Holland, John (1975) Adaptation in Natural and Artificial Systems. Ann Arbor, Mich.: The University of Michigan Press.

Lund, Henrik; Pagliarini, Luigi, Miglino, Orazio (1995) Artistic Design with Genetic Algorithms and Neural Networks. in: Proceedings of 1NWGA, Alexander, J. T. (ed.). Vaasa, Finland: University of Vaasa.

Mitchell, Melanie (1996) An Introduction to Genetic Algorithms, MIT Press.

Simon, Herbert (1973) The Structure of Ill Structured Problems. Artificial Intelligence. Vol 4, Issue 3-4. North-Holland Publ. Co.

Simon, Herbert (1996) The Sciences of the Artificial. Cambridge, Mass.: The MIT Press.

Turrin, Michela, von Buelow, Peter, Stouffs, Rudi (2011) Design explorations of performance driven geometry in architectural design using parametric modeling and genetic algorithms. In Advanced Engineering Informatics. Vol. 25, Is. 4, Oct., pp. 656-675. Elsevier.

von Buelow, Peter (2012) ParaGen: Performative Exploration of Generative Systems. In Journal of the International Association for Shell and Spatial Structures. Vol.53 No. 4 December n. 174.

PETER VON BUELOW studied under Werner Sobek at ILEK at the University of Stuttgart, and holds degrees in both Architecture and Engineering. His research involves evolutionary computation for exploration and optimization of structural systems. Professor von Buelow has worked professionally in both architecture and engineering offices in Germany and the US including: RFR-Stuttgart, Greiner Engineering and SL-Rasch, and spent a year at IL (under Frei Otto) as a Fulbright Scholar. Research projects include thin shell, light weight, personnel blast shelters (US Air Force) and a modular series of frame supported tensile structures for aircraft maintenance (US Marine Corps).