Science Arena Publications
Specialty journal of Engineering and Applied Science
Available online at www.sciarena.com
2016, Vol, 1 (1): 1-19

# Applying Genetic Algorithms in Architecture

**Hami dreza saremi[1], Sahar khodabakhshi[2]*, Matin khalaghdoost[3]**

1- Assistance Professor Department of Art and architecture, Tarbiat Modares University،Tehran, Iran
saremi@modares.ac.ir

2-Young Researchers Club, khomein Branch, Islamic Azad University, khomein, Iransahar.khodabakhshi@hotmail.com

3-Department of Art and Architecture,Yadegar -e- Imam Khomeini (RAH) Shahre Rey Branch,Islamic Azad University, Tehran,Iran

m.khallaghdoost@gmail.com

**Abstract:** This paper is aimed to apply Genetic Algorithms in creating architecture forms. Firstly, the introduction of genetic algorithms is presented .Secondly, experiments on designing forms for an architecture project by genetic algorithms are brought forward. This paper investigates whether the utilization of Genetic Algorithms is a necessity or a trend; whether Genetic Algorithms are used to accommodate specific needs of architecture or merely to bear innovative and complex forms; and consequently whether Genetic Algorithms serve reality or utopia. In order to answer these questions, the paper examines the operation of Genetic Algorithms in other disciplines and in architecture as well as the implications of its applications in architecture. Finally, the paper demonstrates the premises for a successful operation of Genetic Algorithms in architecture.
**Keywords**: Artificial intelligence; Genetic algorithms; Building design; Low–energy design; Generative systems; Optimization in architecture; Architectural design; Genetic algorithms in architecture; Artificial intelligence in architecture

## Introduction: An Overview of Genetic Algorithms

Evolutionary biology studies the origin, the change and the multiplication of species overtime. In evolutionary biology, the enormous set of possibilities of prospective genetic sequences, and the desired "solutions" are the results of highly fit organisms that are able to survive and reproduce within their environments. Due to that fact, evolutionary biology is an appealing source of inspiration for addressing complex computational problems that require searching through a huge number of possible solutions. Moreover, evolution can be seen as a massively parallel search method; rather than work on one species at a time, evolution tests and changes whole populations of species simultaneously. The natural procedures of life evolution and the techniques that are used in evolutionary biology have influenced many other disciplines that use evolutionary algorithms to solve complicated problems. A particular class of these computational algorithms is Genetic Algorithms. [1]

A genetic algorithm is a type of searching algorithm. It searches a solution space for an optimal solution to a problem. The key characteristic of the genetic algorithm is how the searching is done. The algorithm creates a “population” of possible solutions to the problem and lets them “evolve” over multiple generations to find better and better solutions. The generic form of the genetic algorithm is found in Figure 1. The items in bold in the algorithm are defined here.

The population is the collection of candidate solutions that we are considering during the course

of the algorithm. Over the generations of the algorithm, new members are "born" into the population, while others "die" out of the population. A single solution in the population is referred to as an individual. The fitness of an individual is a measure of how "good" the solution represented by the individual is. The better the solution, the higher the fitness – obviously, this is dependent on the problem to be solved.

The selection process is analogous to the survival of the fittest in the natural world. Individuals are selected for "breeding" (or cross-over) based upon their fitness values – the fitter the individual, the more likely that individual will be able to reproduce. The cross-over occurs by mingling the two solutions together to produce two new individuals.

During each generation, there is a small chance for each individual to mutate, which will change the individual in some small way

To use a genetic algorithm, there are several questions that need to be answered:

1. Create a **population** of random candidate solutions named *pop*.
2. Until the algorithm termination conditions are met, do the following (each iteration is called a generation):
   (a) Create an empty population named *new-pop*.
   (b) While *new-pop* is not full, do the following:
      i. **Select** two **individuals** at random from *pop* so that individuals which are more **fit** are more likely to be selected.
      ii. **Cross-over** the two individuals to produce two new individuals.
   (c) Let each individual in *new-pop* have a random chance to **mutate**.
   (d) Replace *pop* with *new-pop*.
3. Select the individual from *pop* with the highest **fitness** as the solution to the problem.

*Figure 1: The Genetic Algorithm*

To use a genetic algorithm, there are several questions that need to be answered:

- How is an individual represented?
- How is an individual's fitness calculated?
- How are individuals selected for breeding?
- How are individuals crossed-over?
- How are individuals mutated?
- How big should the population be?
- What are the "termination conditions"?

Most of these questions have problem-specific answers. The last two, however, can be discussed in a more general way.

The size of the population is highly variable. The larger the population, the more possible solutions there are, which means that there is more variation in the population. Variation means that it is more likely that good solutions will be created. Therefore, the population should be as

large as possible. The limiting factor is, of course, the running time of the algorithm. The larger the population, the longer the algorithm takes to run.

The algorithm in Figure 1 has a very vague end point – the meaning of "until the termination conditions are met" is not immediately obvious. The reason for this is that there is no one way to end the algorithm. The simplest approach is to run the search for a set number of generations – the longer you can run it, the better. Another approach is to end the algorithm after a certain number of generations pass with no improvement in the fitness of the best individual in the population There are other possibilities as well.

Since most of the other questions are dependent upon the search problem, we will look at an example problem that can be solved using genetic algorithms: finding a mathematical function's maximum problem.

**Function Maximization**

One application for a genetic algorithm is to find values for a collection of variables that wil maximize a particular function of those variables. While this type of problem could be solved in other ways, it is useful as an example of the operation of genetic algorithms as the application of the algorithm to the problem is fairly straightforward.

For this example, let's assume that we are trying to determine the variables that produce the maximum value for this function:

f( w, x, y, z ) = w3 + x2 – y2 – z2 + 2yz – 3wx + wz – xy + 2

This could probably be solved using multi-variable calculus, but it is a good simple example of the use of genetic algorithms. To use the genetic algorithm, we need to answer the questions listed in the previous section.

**How is an individual represented?**

What information is needed to have a "solution" to the maximization problem we are working on? Hopefully it is clear that all we need is to have values for w, x, y, and z. Assuming that we have values (any values) for these four variables, we have a candidate solution for our problem. The question is how to represent these four values. A simple way to do this is to simply have an array of four values (integers or floating point numbers, either way). However, for genetic algorithms it is usually best to have a larger individual – this way, variations can be done in a more subtle way. The research shows [Beasley, Bull, and Martin 1993; Holland 1975] that using individuals represented by bit strings offers the best performance. So, we can simply choose a size

in bits for each variable, and then concatenate the four values together into a single bit string. For example, we will choose to represent each variable as a four-bit integer, making our entire individual a 16-bit string. This is obviously very simplistic, and in reality we would probably want to have the variables represented as floating point values with more precision, but for an example it should work. Thus, an individual such as

1101 0110 0111 1100

represents a solution where w = 13, x = 6, y = 7, and z = 12.

**How is an individual's fitness calculated?**

Next we consider how to determine the fitness of each individual. There is generally a differentiation between the **fitness** and **evaluation** functions. The evaluation function is a function that returns an absolute measure of the individual. The fitness function is a function that measures the value of the individual relative to the rest of the population.

In our example, an obvious evaluation function would be to simply calculate the value of f for the given variables. For example, assume we have a population of 4 1individuals:

1010 1110 1000 0011

0110 1001 1111 0110

0111 0110 1110 1011

0001 0110 1000 0000

The first individual represents w = 10, x = 14, y = 8, and z = 3, for an f value of 671. The values for the entire population can be seen in the following table:

| *Individual* | *w* | *x* | *y* | *z* | *f* |
|---|---|---|---|---|---|
| **1010111010000011** | **10** | **14** | **8** | **3** | **671** |
| **0110100111110110** | **6** | **9** | **15** | **6** | **-43** |
| **0111011011101011** | **7** | **6** | **14** | **11** | **239** |
| **0001011010000000** | **1** | **6** | **8** | **0** | **-91** |

The fitness function can be chosen from many options. For example, the individuals could be listed in order from lowest to highest evaluation function values, and an ordinal ranking applied. Or, the fitness function could be the individual's evaluation value divided by the average evaluation

value2. Looking at both of these approaches would give us something like this:

| *Individual* | *evaluation* | *ordinal* | *averaging*[3] |
|---|---|---|---|
| **1010111010000011** | **671** | **4** | **2.62** |
| **0110100111110110** | **-43** | **2** | **0.19** |
| **0111011011101011** | **239** | **3** | **0.81** |
| **0001011010000000** | **-91** | **1** | **0.03** |

The key is that the fitness of an individual should represent the value of the individual relative to the rest of the population, so that the best individual has the highest fitness.

**How are individuals selected for breeding?**

The key to the selection process is that it should be probabilistically weighted so that higher fitness individuals have a higher probability of being selected. Other than these specifications, the method of selection is open to interpretation.

One possibility is to use the ordinal method for the fitness function, then calculate a probability of selection that is equal to the individual's fitness value divided by the total fitness of all the individuals. In the example above, that would give the first individual a 40% chance of being selected, the second a 20% chance, the third a 30% chance, and the fourth a 10% chance. Clearly this is giving the better individuals more chances to be selected.

A similar approach could be used with the average fitness calculations. This would give the first individual a 72% chance, the second a 5% chance, the third a 22% chance, and the fourth a 1% chance. This method makes the probability more dependent on the relative evaluation functions of each individual.

**How are individuals crossed-over?**

Once we have selected a pair of individuals, they are "bred" – or in genetic algorithm language, they are crossed-over. Typically two children are created from each set of parents. One method for performing the cross-over is described here, but there are other approaches. Two locations are randomly chosen within the individual. These define corresponding substrings in each individual. The substrings are swapped between the two parent individuals, creating two new children. For example, let's look at our four individuals again:

1010 1110 1000 0011

0110 1001 1111 0110

0111 0110 1110 1011

0001 0110 1000 0000

Let's assume that the first and third individuals are chosen for cross-over (making sense, as these are the two top individuals). Keep in mind that the selection process is random, however. The fourth and fourteenth bits are randomly selected to define the substring to be swapped, so the cross over looks like this:

1010111010000011 1010111010000011 1011011011101011

→ →

0111011011101011 0111011011101011 0110111010000011

Thus, two new individuals are created. We should keep creating new individuals until we have created enough to replace the entire population – in our example, we need one more cross-over. Assume that the first and fourth individuals are selected this time.

Note that an individual may be selected multiple times for breeding, while other individuals might never be selected. Further assume that the eleventh and sixteenth bits are randomly selected for the cross-over point. We would see a second cross-over like this:

1010111010000011 1010111010000011 1010111010000000

→ →

0001011010000000 0001011010000000 0001011010000011

So, our second generation population is:

1011 0110 1110 1011

0110 1110 1000 0011

1010 1110 1000 0000

0001 0110 1000 0011

How are individuals mutated?

Finally, we need to allow individuals to mutate. When using bit strings, the easiest way to implement the mutation is to allow every single bit in every individual a chance to mutate. This chance should be very small, since we don't want to have individuals changing dramatically due to mutation. Setting the percentage so that roughly one bit per individual will change on average is probably a reasonably good number.

The mutation will consist of having the bit "flip" – 1 changes to a 0 and a 0 changes to a 1. In our example, assume that the bold and italicized bits have been chosen for mutation:

1011011011101011 → 1011011011101011

0110111010000011 → 0110101010000011

1010111010000000 → 1010111010010000

0001011010000011 → 0101011010000001

**Wrapping Up**

Finally, let's look at the new population:

| *Individual* | *W* | *x* | *y* | *z* | *f* |
|---|---|---|---|---|---|
| **1010111010000011** | **11** | **6** | **14** | **11** | **1,045** |
| **0110100111110110** | **6** | **10** | **8** | **3** | **51** |
| **0111011011101011** | **10** | **14** | **9** | **0** | **571** |
| **0001011010000000** | **5** | **6** | **8** | **1** | **-19** |

The average evaluation value here is 412, versus an average of 194 for the previous generation. Clearly, this is a constructed example, but the exciting thing about genetic algorithms is that this sort of improvement actually does occur in practice, although the amount of improvement tends to level off as the generations continue.

Applications of Genetic Algorithms

Genetic Algorithms address quite a large number of problems including image processing, face recognition, protein structure prediction, time series analysis, computer software automatic evolution, cellular automaton rule evolution, robotics, control, aeronautics, and many more. Fields in which Genetic Algorithms have been extensively used include Optimization, Automatic Programming, Machine Learning, Economics, Immune Systems, Ecology Population Genetics, Evolution and Learning, and Social Systems. These lists are by no means complete, but illustrate the variety of applications that Genetic Algorithms offer in various fields.

Genetic Algorithms in Architecture

Since the 1990's a shift has been noticed in the way avant-garde architects have used new technologies of evolutionary biology to address or depict the increased complexity that is noticed in today's architecture. Indeed, the layer of complexity that is introduced cannot be resolved by conventional design methods. Likewise, the quantity of information and the level of complexity involved in most building projects surpass designers' abilities to thoroughly comprehend and predict them. Genetic Algorithms, among many other evolutionary techniques, have been used in architecture as optimization tools or as form-generation tools.

In the former, Genetic Algorithms address well-defined building problems, such as structural and mechanical. Genetic Algorithms are used as stochastic methods for solving optimization and search problems, operating on a population of possible solutions. Hereafter, this utilization of Genetic Algorithms is addressed as "necessity". In the later utilization, Genetic Algorithms are used under the scope of the concept of emergence. Genetic Algorithms are used to produce innovative representations and descriptions of processes by which emergent structures, often with tremendous complexity, are derived. Hereafter, this utilization of Genetic Algorithms is addressed as "trend".

While other disciplines have adopted computational tools based on the principles of evolutionary biology, in architectural design evolutionary processes have not been broadly applied. Only recently has there been a noticeable shift in the way architects explore such techniques to address complex problems. Indeed, one of the main problems in architecture today is the quantity of information and the level of complexity involved in most building projects.

Genetic Algorithms offer an effective solution to this problem by solving optimization and search problems, operating on a population of possible solutions. In architecture Genetic Algorithms operate in two ways: as optimization tools and as form-generation tools. In the first way Genetic Algorithms address well-defined building problems, such as structural, mechanical, and thermal and lighting performance. In the second way Genetic Algorithms are used under the scope of the concept of Emergence.

The dual operation of Genetic Algorithms in architecture will be analyzed hereafter.

Genetic Algorithms and Design Optimization

Design optimization has been introduced to building industry as a tool to achieve the best possible building performance, the highest reliability and / or the lowest cost. Building

performance includes among others the structural, acoustic, lighting, energy and spatial attributes/properties of a building. For example one of the basic aims of structural optimization is to minimize the overall weight so as to minimize the material cost. With the increased demands of the global market for more effective and complex buildings, the utilization of Genetic Algorithms, as one of numerous optimization techniques, is a necessity. Especially for large-scale structures with thousands of elements or structures with very complicated geometry manual calculations cannot satisfy the increased demand so the use of optimization techniques is inevitable. For example, in the project for the Aquatics Centre for the 2008 Olympic Games in Beijing, the new automated approach for selecting section sizes and checking them to design codes for all 25 000 steel sections was crucial for the feasibility of the project's roof


Figure 2: Aquatics Centre of 2008 Olympic Games, Beijing

Most optimization problems are made up of three basic components. The first is the objective function which we want to minimize or maximize, the second component is the designation of a set of design variables that affect the value of the objective function, and the third component is the determination of a set of constraints that allow the design variables to have certain values. For instance, in terms of the structural performance of a panel, we determine what we want to minimize that might be the stress in a particular region, and then we determine the variables, that could be the geometry and material of the panel, and then we set the constraints that could be the minimization of the weight of the panel.

For a thorough investigation of the operation of Genetic Algorithms as an optimization technique in design the Genetic Algorithm Tool for Design Optimization implemented by Luisa Caldas and

Leslie Norford in 1999 will be examined. The Genetic Algorithm Tool [2] explores the use of Genetic Algorithms in the context of generative and goal-oriented design in order to develop and evaluate criteria related to the environmental performance of a building. The tool searches for the optimal window size in a building in order to optimize three characteristics of the adjacent room: lighting, heating and cooling performance. Environmental conditions including climate conditions, window orientation, and glazing, are parameterized since they influence the achievement of low energy consumption solutions.

Two office buildings are examined, both with controlled internal climate and artificial lighting. The buildings are located in two different cities: Phoenix, Arizona and Chicago, Illinois. After the Genetic Algorithms have generated possible design solutions, the designs are then evaluated in terms of lighting and thermal performance through a detailed thermal analysis program (DOE2.1 E).

Then the Genetic Algorithms use the results from these simulations to further investigate towards finding low energy solutions to the problem under study. Solutions are visualized using an AutoLisp routine, since AutoLisp procedures allow the results to be visually inspected as AutoCAD drawings.

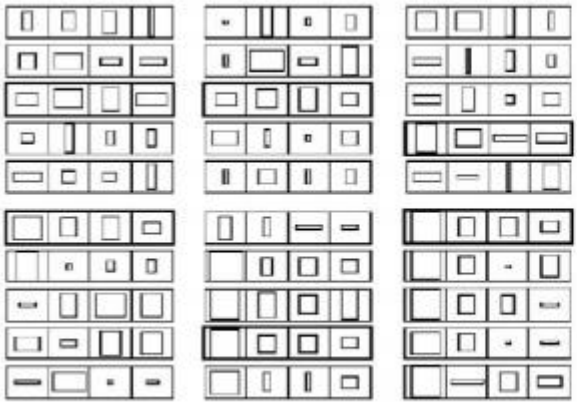

Figure 3: Generations for Phoenix project

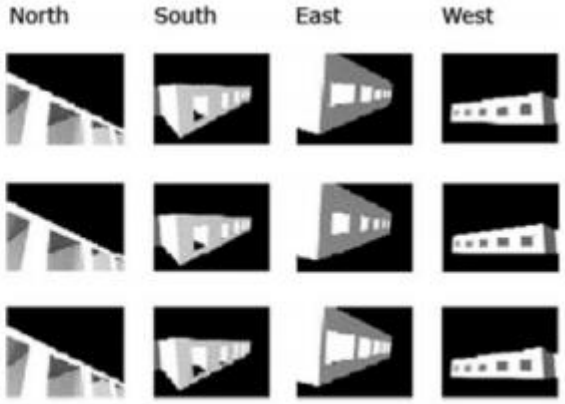


Figure 4: Final solutions for Phoenix project

**Genetic Algorithms and Emergence**

Emergence was introduced as a subset of complexity theory in the 1980s and it is linked back to

the development of systems theory in the 1920s. Emergence refers to the universal way in which small parts of systems in nature driven by very simple behaviors are tended toward coherent organizations with their own distinctly different behaviors. Vivid examples from the natural world are the hive, swarming, and flocking where independent parts are formed into one system with a complex or / and random behavior.

While such models are used to generate novelty design or evolving forms, only recently have architects started to explore the ideas of Emergence and examined the new technologies in the natural science to enrich the scopes of architecture, redefining new formal paradigms.

Avant-garde architects have referred to theoreticians such as Rene Thom and D'arcy Thomson to investigate the ways in which biological ideas can operate and be generative in architecture. The term of Emergence equates architecture with nature assuming that design is dominated by the same principles as the natural world. Architects attempt to create architecture as nature; architecture that is nature. That is why the notion of emergence is linked tightly with the notion of growth, evolution, continuity and behavior. Indeed, behavior is a dynamic process of feedback between states of forming no axis or center. However, if architects want to use this concept, they have to equate buildings with the living organisms, parts with the whole and function with behavior. In this case, each unit is a part of an environment defined by the building and its neighbors. To produce such environments, architecture must be expressed as rules of generation, in order to allow evolution to be defined.

In this context Genetic Algorithms are introduced, offering a bottom-up approach in multiple cycles of evolution. For a thorough investigation of the operation of Genetic Algorithms as a formgeneration technique the Generative Form Modeling and Manufacturing (Genr8) [3] developed by the Emergent Design Group in 2001 will be examined. The Generative Form Modeling and Manufacturing is a tool that combines a generative design algorithm -- Map L-systems -- with evolutionary search algorithms – Genetic Algorithms. Through the combined use of these two algorithms, Genr8 designs the geometry of surfaces and allows the user to interfere with growth, by stopping evolution, altering fitness and run-time parameters, including the environment in which growth takes place. Genr8 was used to conduct an experiment to explore the potential of achieving a coherent coexistence of the logic behind manufacturing, constraints implied by construction materials and complex geometrical forms.

The development of the surfaces is an outcome of two processes: definition and re-writing rules.

The application of these rules to the initial definition and any subsequent development alters all parts of the surface, and the results are continuously translated into three dimensional drawings. The conditions that influence the growth of the surface also influence the grammar of the Genetic Algorithms and along with innate factors define the form of the surface. Growth is achieved with the use of the Genetic Algorithms. A critical issue in the use of the Genetic Algorithms is the definition of the fitness criteria. These are defined as mathematical functions representing properties of the form of the surface including size, smoothness, soft boundaries, subdivisions, and symmetry.

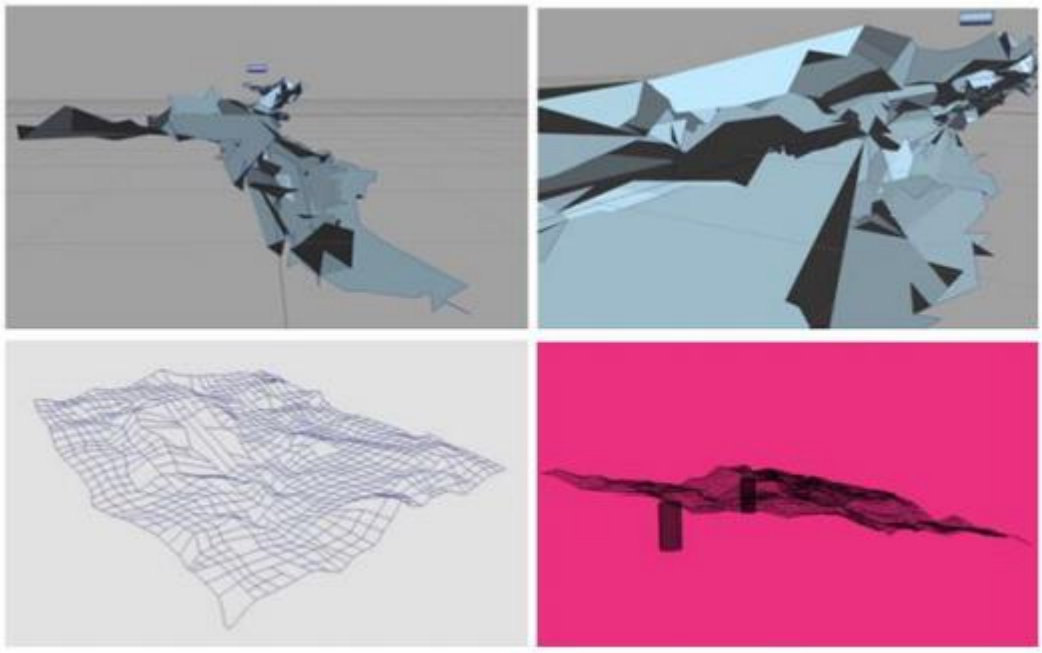

**Figure 5: Images generated by Genr8, http://mit.edu/edgsrc/www/genr8/media.html**

**Genetic Algorithms in architecture and other disciplines**

Based on the former analysis, we can argue that there is a basic difference between architecture and other disciplines in the utilization of Genetic Algorithms. Unlike other fields that address problems whose targets are well-defined, many of the problems that architecture deals with are illdefined. Liddament, in his article "The Computationalist Paradigm in Design Research," [4] shows that although computational tools, such as Genetic Algorithms are powerful in scientific domains to solve many problems, they do not adequately fit the actual design activity. He acknowledges the fact that the computationalist paradigm presents itself as a "scientific approach with a correspondingly rigorous methodology." Design intention, for example, is usually an ill-defined problem. Even if designers finally manage to code design intention, this process will not be enough

to guarantee a successful design solution. This is because an architectural project is a composition of design performances including spatial, structural, lighting, acoustic, and thermal. These elements continuously interact during the design process. Consequently, the optimum solution of one of those elements does not lead to a successful combination of all elements, hence, to a global optimum and successful design solution.

Another noticeable difference between architecture and other fields is that architecture is not just a rule-driven science. While architecture primarily serves the function of the buildings, it also deals with cultural, social and aesthetical notions whose codification and fitness is subjective, but still could be encoded into rules.

### Genetic Algorithms: Necessity or Trend?

Despite the difficulties in coding many architectural problems, algorithmic processing of Genetic Algorithms-- as a way that builds human's thought to solve problems -- can be used in the design process. Taking this into account, the next question that is addressed is whether the use of Genetic Algorithms in architecture today is driven by necessity or trend.

### Necessity

The increased human needs and the today's lifestyle call for more complicated functional requirements and the quest for more innovative forms augments the complexity of the formal manifestation. The level of complexity that is introduced and the quantity of information that it entails constitute one of the basic problems that architecture must deal with today. This problem cannot be resolved by conventional design methods. Likewise, the constraints of this problem surpass designers' abilities to thoroughly comprehend them and predict their solution. Seeing Genetic Algorithms as tools that can answer to the specific needs of practical architecture and not merely of experimental design, the notion of necessity of Genetic Algorithms arises.

Genetic Algorithms, as stochastic processes for solving optimization and search problems, go through thousands of iterations in a second and find the solution sets, extending designers' thoughts into a once unknown and unimagined world of complexity. Under the scope of necessity architectural projects utilize Genetic Algorithms as optimization tools to address well-defined problems, such as the structural, mechanical, thermal and lighting performance of a building.

In this case, Genetic Algorithms serve design for architecture, design for the real world. This is the reason why this utilization of Genetic Algorithms is called a necessity.

However, even in the case of necessity there is a possibility for architects to select Genetic

Algorithms --among numerous other optimization techniques -- just because Genetic Algorithms are a cutting edge method. Indeed, there are many other optimization methods, including hill climbing, simulated annealing, tabu search, stochastic tunneling, and harmony search. The selection of a particular optimization method should be based on the specific needs and nature of a problem and how efficiently the method can respond to these demands and not how trendy a method is. For example Prof. Kristina Shea used the method Structural Topology and Shape Annealing (STSA) to develop EifForm, a stochastic optimization software demonstrator for generative structural design. [5]

**Trend**

On the other hand, we have noticed a more generalized utilization of Genetic Algorithms as form generation tools. Influenced by the concept of emergence and evolutionary architecture, architects use Genetic Algorithms as means of exploring innovative forms. The number of architects who adopt these techniques is rapidly increasing. Likewise, their designs are getting more and more complex.

Reading the description and observing the final output of those experiments, one could argue that formal complexity is the primary consideration for those architects. The fact that these forms often do not follow or serve any functional or structural requirement boosts this argument. At this point, a sequence of questions is raised:

- Why should the complexity of buildings be increased today if it is not to justify functional or structural requirements?
- Do those architects believe that only through complexity they will achieve formal innovation?
- Do they believe that the formal complexity of buildings can reflect the complexity of the building environment and everyday life?
- Why is formal complexity their primary task anyway?

The fact that those architects utilize Genetic Algorithms merely as form-generation tools, omitting spatial and structural performance, in conjunction with the fact that most of those projects do not produce architecture but rather abstract shapes which are difficult to be translated into architecture could lead to the conclusion that in this case, Genetic Algorithms serve design for abstract shapes, design for a conceptual world. This is the reason why this utilization of Genetic Algorithms is called a trend.

Also most of these architects do really respect the notion of Emergence or they have not truly

understood the real meaning of this notion. Emergence is not interested in parts; it is the science of wholes. And since design process is comprised of many parts they should not omit them and focus only in formal representation. However, even if architects are able to combine many building requirements and performances with formal generation, a fallacy still exists in this attempt. This fallacy is based on the equalization of architecture to biology. Indeed there are some similarities between architecture and biology, both are materially and organizationally based, both are concerned with morphology and structuring. Nevertheless, those similarities do not lead to the perception of building as artificial life form, which is dominated by the same principles that the natural world is dominated. This speculation equates cultural evolution with organic evolution, which is the same as to equate the Darwinian with Lamarckian theories of evolution. Since form in architecture is a cultural artifact, imaging numerous abstract meanings as previously mentioned, form cannot be subjected to the Darwinian definition of evolution. Architects must clarify where architecture is literally considered as part of nature, where there are analogies or metaphors, and where nature is a source of inspiration.

There is no doubt that abstract shapes of a conceptual world may be implemented in the future, opening the path for new formal expressions. However, the real challenge would be to use Genetic Algorithms in real-world architectural contexts. Indeed, it is very difficult for a designer to combine complexity and constraints imposed by the design problem with evolutionary formal generation. Possibly this is the reason why architects use Genetic Algorithms either as a tool to solve structural and mechanical problems or as a tool to generate forms.

**Design process and designer**

The changes in the design process and the role of design are one of the most important implications of the utilization of Genetic Algorithms. In the case where Genetic Algorithms are used as an optimization method those changes are not that important. Even if at first glance it may appear that those methods lead to the replacement of the designer from the design loop, this is not really the case. Indeed, the designer or the engineer is the one who decides what will be optimized setting the variables and the functions of the problem while the optimization techniques are just another tool available to the designers shorting the design cycle times and performing the calculations.

On the other hand, when Genetic Algorithms are used as a form generation tool these implications are more important, since Genetic Algorithms are used during the conceptual design

phase. Evolutionary simulations replace the traditional design processes and the designer in a sense is neutralized and marginalized. This is because most of the designers use Genetic Algorithms to breed new forms rather than merely design them. As fascinating as the idea of breeding buildings inside a computer may be for some designers, it is clear that merely using digital technology without functional, structural and topological thinking will never be adequate for real architecture. Consequently, the main role of the designer is to be the judge of aesthetic fitness. In the words of Steadman:

"Just as Darwin inverted the argument from design, and 'stole away' God as designer, to replace Him with natural selection, so the Darwinian analogy in technical evolution removes the human designer and replaces him with the 'selective forces' in the 'functional environment' of the designed object." [6]

**Selection**

The selection of the final output is another issue that is derived by the use of Genetic Algorithms. In the case of necessity the target is well-defined and consequently the selection of the result is based on the optimal solution. For example, in the project of Caldas and Norford that was previously examined the target was the optimal window size, in terms of lighting, heating and cooling performance. Based on this, the final output the sizes of the windows were selected based on the lowest-energy consumption. However, in the case of trend the target is ill-defined and the criteria for the selection are only aesthetic. At this point a sequence of question is raised:

- Why should designers follow such a strict, logical, and subjective process, if they eventually base their selection on abstract and objective aesthetic criteria?
- Why do they not design something similar to the final output at the very beginning?
- Is it because some architects merely use Genetic Algorithms as a tool to surpass their limitations of creativity?
- Is it because they believe that the utilization of Genetic Algorithms document their design process, providing it with a "theoretical" and conceptual substance?

**Genetic Algorithms: Necessity and Trend**

Undoubtedly there are many difficulties in the application of Genetic Algorithms in architecture. However, Genetic Algorithms have the potential to play a more effective role in the future of architecture. On one side it is the architectural problem that has a "reason" and a fundamental "factor." The "reason" is the increasing quantity of information and the increasing level of

complexity involved in most building projects today. The “factor” is the various building performances including spatial, functional, aesthetic, structural, energy, and lighting. All these performances interact and interlink and cannot be considered separately. On the other side there is the potential of Genetic Algorithms. Genetic Algorithms are search methods for addressing complex problems, finding optimum design solutions from indeterminate search spaces constrained by multiple input factors. If architects want to use computational design to address architectural problems Generative

Algorithms can be one of possible tools to do so; they can solve the “reason”, taking into account the primary “factor”.

Yet, in order for Genetic Algorithms to be applied in architecture, some things must be done. Today the usage of Genetic Algorithms is local; that means that Genetic Algorithms are used merely for form generation or for optimization of one building performance. The problem in today’s utilization is that the local function of Genetic Algorithms is influenced during the design process and, in the end, the local “optimal” is lost. For example, if a designer utilizes Genetic Algorithms so as the form of a building to be emerged, in the next step of the design process he/she will need to change the form partly or radically so as to fit the function. The same will happen if a designer uses Genetic Algorithms to find the optimal form minimizing the cost and maximizing daylight. Then, if he/she tries to calculate the thermal performance with the conventional tools the design will risk losing the local optimal which has been calculated in the first step, since in the next step of the process another performance must be calculated and new functions will be introduced.

A possible solution to this problem is the coordination of generative tools with optimization tools; the combination of dual utilization of Genetic Algorithms: trend and necessity. The form will emerge by the simultaneous calculations of most of the performance-driven functions. This process can be seen as a potential effort to achieve the design goal finding the ‘fit’ between form and context, defining the context as “anything in the world that makes demands of the form” [7] -- including meanings, aesthetics, environment, and function -- as Christopher Alexander stated. Indeed, for Alexander “the form is the solution to the problem; the context defines the problem.” [8] This process will help architects to find global “optimal” or satisfactory solutions, aid multidisciplinary negotiations, shorten the project delivery time and cost, and find feasible design

alternatives. However, some potential negative consequences will be derived by the application

of this process. Architects must make sacrifices related to the generalizations and reductionisms of some of the design problems. Those must be done in order to transform most of the ill-defined problems -- that are related to the subconscious and subjectivity of the architect, such as the design intention -- into well-defined ones, by coding them. Undoubtedly, this transformation will be very difficult to achieve. Many authors have referred to the difficulties of coding the very complex and unexpected way that the human mind functions. On the other hand these transformations seem inevitable since the constraints of recent architectural problems surpass designers' abilities to thoroughly comprehend them and predict their solution. Another aspect that architects must take into account applying this method is the fact that many problems emerge or diversify during the design procedure.

Consequently, the interactivity between designer and computational tool must be accommodated allowing the designer to add/reduce the variables or changing the fitness functions.

## Conclusion

One of the problems that contemporary architecture has to deal with is the quantity of information and the increasing complexity of most of the architectural projects. Only recently have architects started to utilize Genetic Algorithms to address this problem. This paper demonstrated the dual operation of Genetic Algorithms in architecture: as optimization tools, and as formgeneration tools, addressing these as necessity and as trend respectively. This paper also indicated the implications of Genetic Algorithms' applications. Some of these implications include the replacement of the traditional design process by the evolutionary simulations, the neutralization and rescission of designers, the abstract criteria of the final selection, and the local utilization of Genetic Algorithms only in some of the design phases. However, Genetic Algorithms have the potential to play a more effective role in the future of architecture. Indeed, Genetic Algorithms can answer the architectural problem taking into account the processes of the design, if they are properly utilized. A possible solution to this problem is the coordination of generative tools with optimization tools so as to achieve a simultaneous calculation of the performances and a global evaluation of the design which will be emerged by the performance-driven functions.

The methodology of incorporating a number of search spaces has been applied in other engineering fields, such as aeronautics and astronautics, leading for example to lighter, stronger, stiffer, and often cheaper automotive bodies, airplane wings, and ship keels. This method is called Multidisciplinary Design Optimization (MDO). Through the utilization of Genetic Algorithms and

other evolutionary techniques, MDO solves complex coupled systems, exploring the interacting disciplines or phenomena at every stage of the design process. Further research in the application of Genetic Algorithms will involve investigating the operation and possible application of MDO in architecture.

**References:**

[1] Scott M. Thede., "An Introduction to Genetic Algorithms." (DePauw University, Greencastle, IN 46135.), 2004

[2] Luisa Caldas, Leslie Norford, A genetic Algorithm Tool for Design Optimization, Media and Design Process [ACADIA '99] Salt Lake City, (29-31 October 1999): 260-271, http://cumincad.scix.net/cgibin/works/Show?b4d2, retrieved November 01, 2007.

[3] Peter Testa and Devyn Weiser, "Emergent Structural Morphology", Architectural Design 72, no1, (January2002), 13-16.

[4] T. Liddament., "The Computationalist Paradigm in Design Research." Design Studies, Vol 20, No 1, (1999)44.

[5] Kristina Shea, Robert Aish, Marina Gourtovaia, "Towards integrated performance-driven generative design tools" [eCAADe '03] Graz University of Technology (Austria): 553-560, http://iam.tugraz.at/~e/cd/ecaade_ files/pdfs/Friday/149_shea.pdf, retrieved November 01, 2007.

[6] Philip Steadman, the Evolution of Designs: Biological Analogy in Architecture and the Applied Arts. (New York, Cambridge University Press: 1979), 189.

[7] Christopher Alexander Notes on the Synthesis of Form. (Cambridge, Harvard University Press: 1964), 15.