Design Architecture By Genetic Algorithm
Short Paper for GA2009

Yibo Xu
Master Of Landscape Architecture
Department of Architecture and Urban Planning, Politecnico Di Milano
e-mail: xybvsxyb@hotmail.com

Abstract

This paper is aimed to apply genetic algorithms in creating architecture forms and establishing a prototype system of such designs. It is divided into four parts. Firstly, the introduction of genetic algorithms and the aims are presented. Secondly, an experiment on designing forms for an architecture project by genetic algorithms is brought forward. The methods and processes would be listed below. It proves that genetic algorithms could not only find the optimized structure for the engineers but also search the 'best' forms for the architects. Thirdly, according to the experiment, a system of how to imply genetic algorithms in designing architectural form is established and explained in detail. This system could be expanded by anyone who could translate the real limitations into genetic representation and fitness functions (modules). Finally, some discussions and conclusions are brought forward.

1. The Introductions:

1.1. The Definition of Genetic Algorithms

The genetic algorithm is perhaps the most well-known of all evolution-based search algorithms. Genetic algorithms were developed by John Holland over twenty-five years ago in an attempt to explain the adaptive processes of natural systems and to design artificial systems based upon these natural systems [1]. In short, the genetic algorithm resembles natural evolution more closely than most other methods.

1.2. The Reason to Use Genetic Algorithm

Evolution is the best designer in the world. For millions of years, designs have been evolved in nature. Biological designs that far exceed any human designs in terms of complexity, performance, and efficiency are rich throughout the living world. From the near-perfection of the streamlined shape of a shark, to the extraordinary molecular structure of a virus, every living thing is a marvel of evolved design.

Genetic algorithm, as the method most closely to natural evolution, gives human design a chance to imitate the great creative process in order to gain the flexibility and efficiency of evolution when optimizing solutions. When analyzing the process of human design, it constructs new solutions to problems using the best features of existing solutions and evolves over time as genetic algorithm and natural evolution do. Hence, genetic algorithms could be used as the bridge linking the human design and natural evolution, which promote human design into more advanced stage.

The relationship of genetic algorithm, natural evolution and human design could be summarized in *Figure 1.2.*

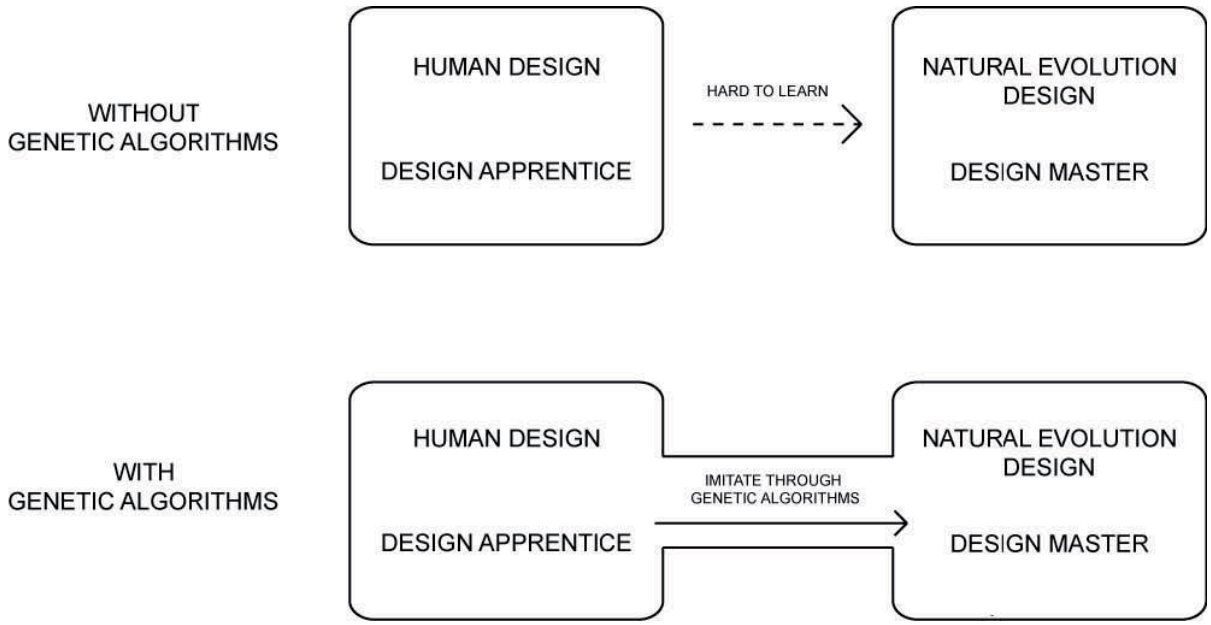


Figure 1.2 Genetic Algorithm is the bridge of Human Design and Natural Evolution

1.3. The Mechanism of Genetic Algorithm

Natural selection ensures that more successful creatures are produced each generation than less successful ones. In the same way, within a genetic algorithm a population of solutions to the problem is maintained, with the 'fittest' solutions being favored for 'reproduction' every generation, during an otherwise random selection process. 'Offspring' are then generated from these fit parents using random crossover and mutation operators, resulting in a new population of fitter solutions. [2]

Coded parameters are normally referred to as genes. A collection of genes in one individual of the population is held internally as a string, and is often referred to as a chromosome. The entire coded parameter set of an individual is known as the genotype, while the solution that the coded parameters define is known as the phenotype.

See Figure 1.3.a

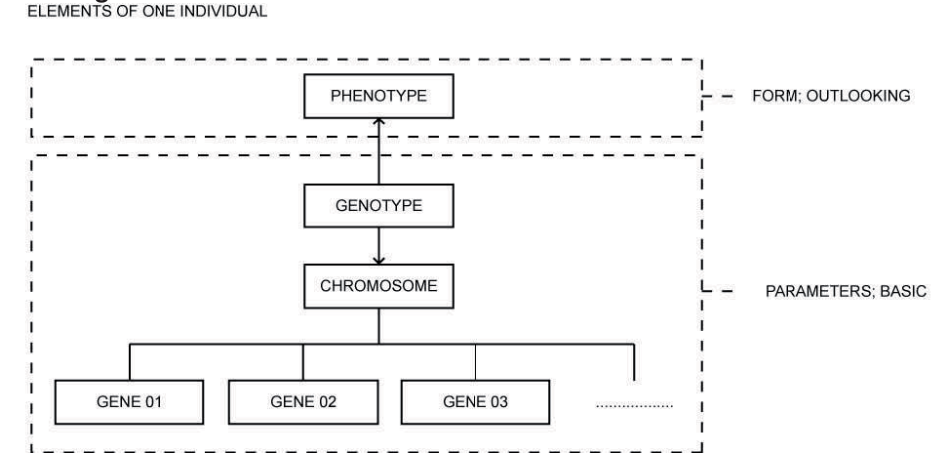


Figure 1.3.a

Typically, populations are initialized with random values by genetic representation, which will translate genotype into phenotype. The main loop of the algorithm then begins, with every member of the population being evaluated and given a fitness value according to how well it fulfills the objective or fitness functions. These scores are then used to determine whether this individual could be placed into a temporary area often termed the 'mating pool' .Two parents are then randomly picked from this area. Offspring are generated by the use of the crossover operator which randomly allocates genes from each parent to each offspring. Mutation is then occassionally applied (with a low probability) to offspring, which is used to keep the variety of the population. When it is used to mutate an individual, typically a single gene is changed randomly. This entire process of evaluation and reproduction then continues until either a satisfactory solution emerges or the genetic algorithm has for run a specified number of generations. [3][4][5].

The simple genetic algorithm is summarized in *Figure 1.3.b*

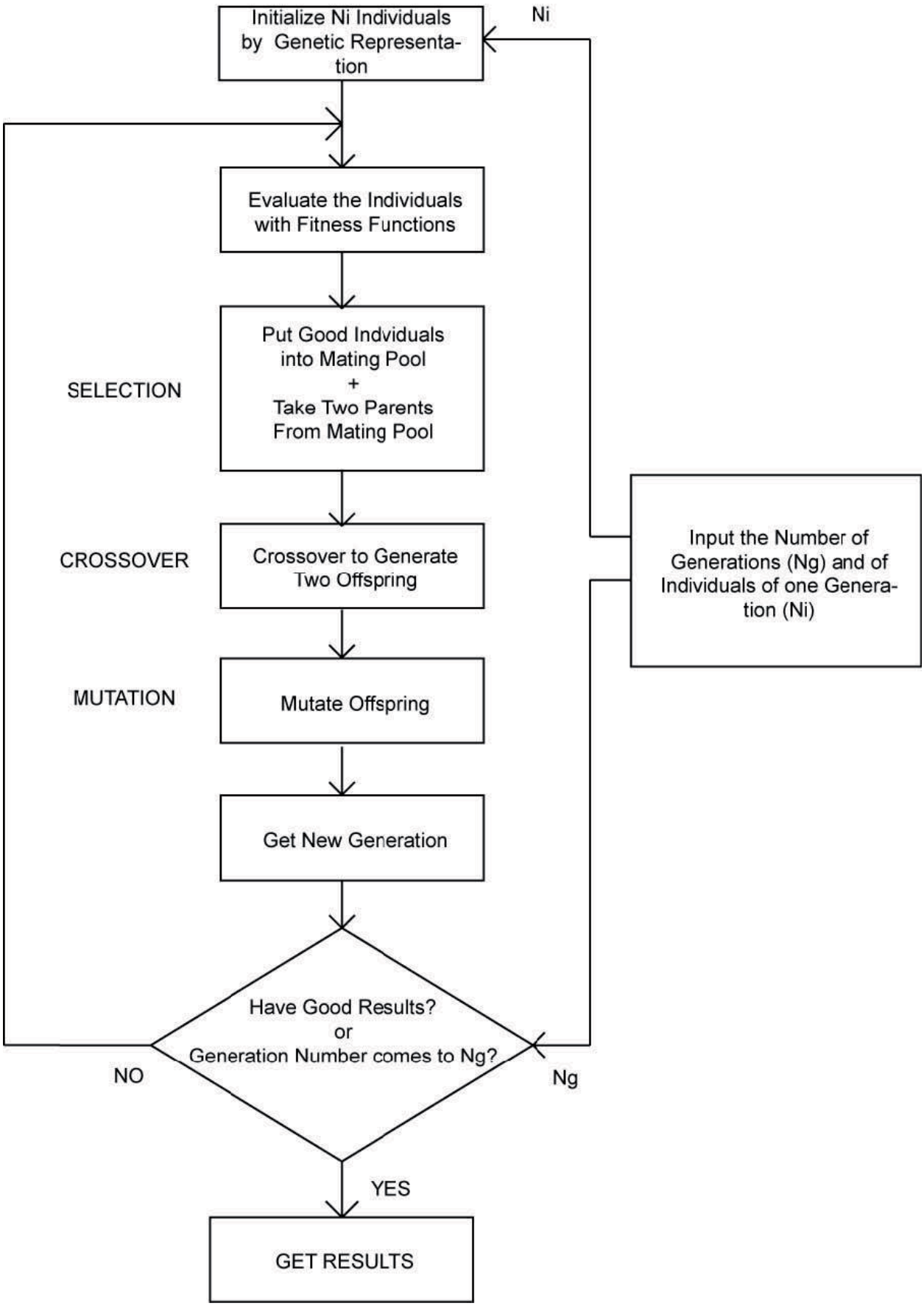


Figure 1.3.b The Structure of A Simple Genetic Algorithm

1.4. The Aim: Apply genetic algorithms in Generative Architecture Design

The last 20 years genetic algorithm is used mostly in the optimization problems. This strong searching method has helped the engineers to optimize the existed technical designs according to the objective requirements. In this paper, we would try to use genetic algorithm to build an automated computer design system to create the design instead of optimizing existed design. Most significantly, the system would not be limited by the 'conventional wisdom' of humans, and could potentially create designs radically different from any produced by a human designer. It could also speed up the design process by automatically providing different designs. [6]

The contents are as following: firstly, with an experiment (designing an architecture project by genetic algorithm), this paper attempts to prove that it is possible to evolve new designs in an architecture project by using the genetic algorithm. Secondly, this paper attempts to develop a prototype, expandable computer system capable of automatically creating new designs by genetic algorithms. Such a system could be used to create designs on its own, or to inspire human designers to try alternative or unconventional designs suggested by the computer.

2. The Experiment:

In order to explain how genetic algorithms work in the process of architecture forms design, an experimental project, a tea house in an old town, would be designed.

2.1. Site Introduction and Limitations:

The site is in Zhu Jia Jiao, a traditional Chinese water town. Rows of wooden buildings were built along the river, which later form the feature 'river street views'. Our building could be any plot along the riverside since this design is face to a type of house instead of a specific house. Siteplan *see Figure 2.1a* [7], Site Situation *Figure 2.1b [8].*

Figure 2.1b [7]
The current situation, the wooden buildings along the river as the river street facade.

Figure 2.1a [7]
The yellow blocks represent the typical wooden houses which could be used as an experiment. The white stripe is the river.

Since the project is in the heart of 'Historical Center', many strict limitations are set by the governors. The most important ones are as following:

1. The maximum of height, width and length of the new building are set as showed in *Figure 2.1c.*
2. The forms of the new building should have relationship with those of the traditional typical wooden houses. *Figure2.1d [9]*
3. The design should meet the required functions.
   a) Entrance 20 sqrmeters
   b) Bar Area 50 sqrmeters
   c) Facility 30 sqrmeters
   d) Tea Area 150 sqrmeters

In fact, more limitations exist besides the ones above. However in order to simplify and pay more attention to the feasibility of the genetic algorithms system, only the crucial points are presented.

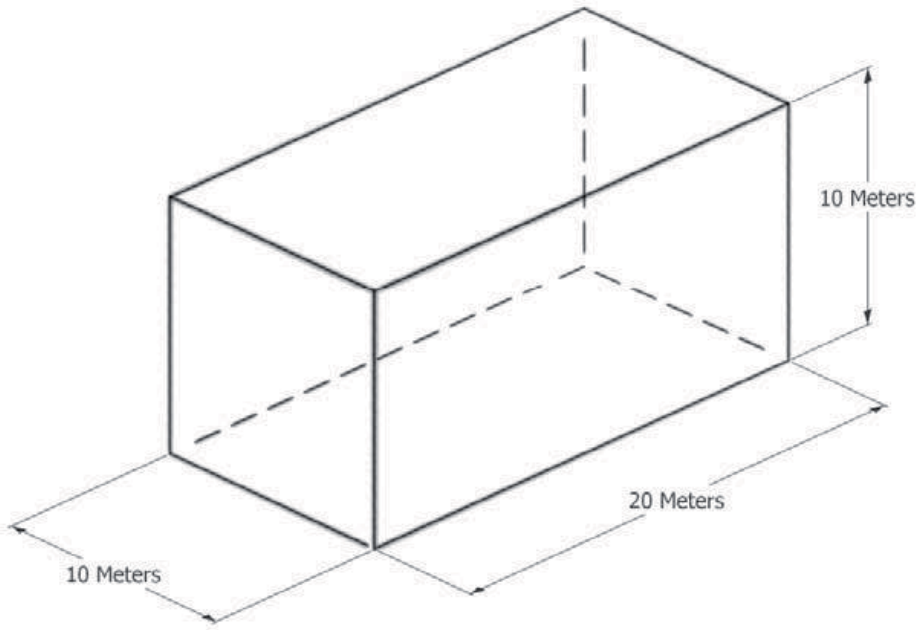


Figure 2.1c.
The limitation of the dimensions
Figure2.1d
Typical Wooden Houses

2.2. Translating Project Limitations to Genetic Algorithms

The most difficult point of using genetic algorithms in architecture design is how to translate the limitations in real project into computer language. Since the translated scripts would be used as the input parameters of genetic algorithms, firstly, we need to understand what kind of input functions genetic algorithms need. Secondly, several steps are presented in order to guide the process of translating limitations.

2.2.1. The Input Functions of Genetic Algorithm: Genetic Representation and Fitness Functions

Typically, a simple genetic algorithm contains two key elements: genetic representation to create the random solutions, fitness functions to evaluate the solutions. The former is the original point that the evolution starts from. The latter is essential for selecting, eliminating and reproducing the 'good' solutions. [4]

2.2.2. Steps of Translation:

Firstly, change 'string' into 'number'. Computer can only understand 0 and 1. That means all limitations, whether is originally of quality or of quantity, should be changed into a description that contains only numbers so that it could be translated into computer language easily in the following step.

Secondly, use the basic rules or forms to translate limitations into computer methods. In another word, it is to find the easiest and simplest way to describe the problems in the way computer could understand. For example, if we want to make a '1x1x1' box, several ways could be used. After analyzing, there are at least three basic elements and three basic rules to make the solid. One is to find an array of 8 points and use them to make solid. Or one could find 12 lines and let them to make solid. Or one could find 6 surfaces and then intersect them into a solid. Here we have points, lines and surfaces and their ways to make a box. Which are the basic ones to create a box? Different designers would give diverse answers and the selection could be based on the goals one wants to achieve. However, one could create this box only after when one understands the basic elements and basic forming process behind the 'appearance'. After knowing the basic rules, it would be easy to create a function that could attain the requested results with the simplest methods. *See Figure 2.2.2*

Thirdly, categorize the functions into the genetic representation or fitness functions. It is hard to differentiate whether the function is the former or the latter. In short, genetic representation will cause probability and diversity while fitness functions will lead to determination and similarity. If all of the limitations are put into fitness functions, it would calculate the solutions since all of them are coming from total randomness (since there is no limitation for the genetic representation, it would output absolute random solutions, which would create laborious works for fitness function). Moreover, the genetic algorithm could be degraded into random search method because of no limitation at the beginning of the calculation. In the other hand, if all the limitations are translated into genetic representation, the system would be too much human-involved since most of the 'random' solutions are preset by the human designer. Hence, in order to keep the merits of natural evolution, a

balance between these two key elements should be considered.

In conclusion, three steps are set to control the translation:

1. Change all the limitations into number based descriptions so that it could be translated into computer language.
2. Use basic rules, elements or forming process to translate the descriptions into a function of computer methods according to the designers' understanding of the limitations.
3. Put the function into either genetic representation or fitness functions. Moreover a balance between the number of genetic representation and fitness functions should be made.

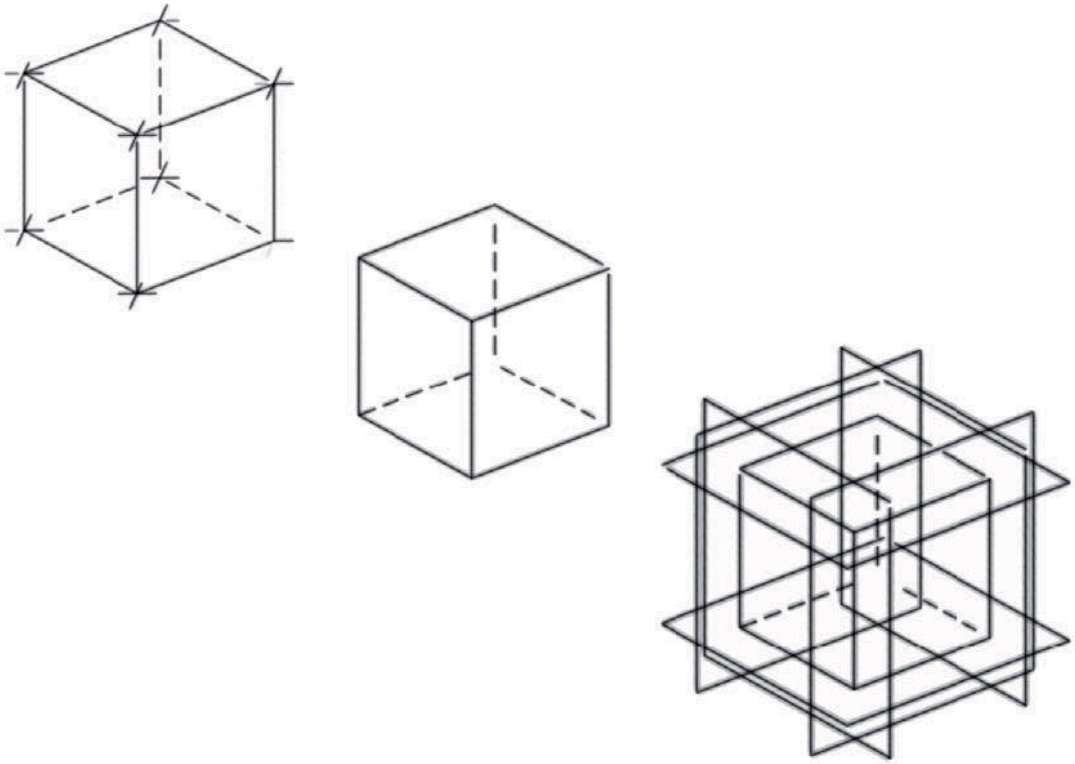


Figure 2.2.2
Three ways of making box. Points, Lines and Surfaces.

2.2.3. Start theTranslation

There are three limitations as mentioned above. The maximum of dimensions of the new building; the new forms should show respects to the typical wooden houses; the design should meet the required functions. Now we would analyze them one by one and try to translate them into genetic representation or fitness functions according to the steps listed above.

2.2.3.1. Limitation1: The Maximum of the Dimensions:

Firstly, since they are numbers which define the upper limits of the function and the limit functions, there is no need to do the first translation.

Secondly, we need to find the simplest rule to describe the 'maximum'. Sometimes, design process is a game of adding and subtracting. If we think of this problem as 'subtract volumes from the existing box', the problem would be much easier than adding volumes. The action of subtracting assures that the final dimensions of the building must be inside the 'maximum'. Otherwise, it could be complicated to calculate every random generated points or curves whether they are under the 'maximum', which could cause enormous calculations.

Thirdly, the limitations could be put into either genetic representation or fitness functions. If the former, the computer would generate the initial solutions with the dimensions that under the 'maximum'. If the latter, the computer would first generate forms with random dimensions and then fitness functions come and use the 'dimension limits' function to eliminate and select the solutions.

In this problem, since the limits of the dimension are set by the urban design laws of old historical district. No solutions could escape the limitations. So, it would be more efficient and easier to translate this limit into genetic representation. Hence, at the beginning of the genetic representation, a box of the set dimensions is created as the 'Drawing Paper'.

As a result, the final translated function would be like this: The genetic representation makes a parameter-set box. With this function, it is not necessary to find another fitness function to judge the dimensions of the solutions. This optimizes the whole system.

We call it Character 1 (C1), and the center point of the box is assigned as original point (0,0,0). *See Figure 2.2.3.1*

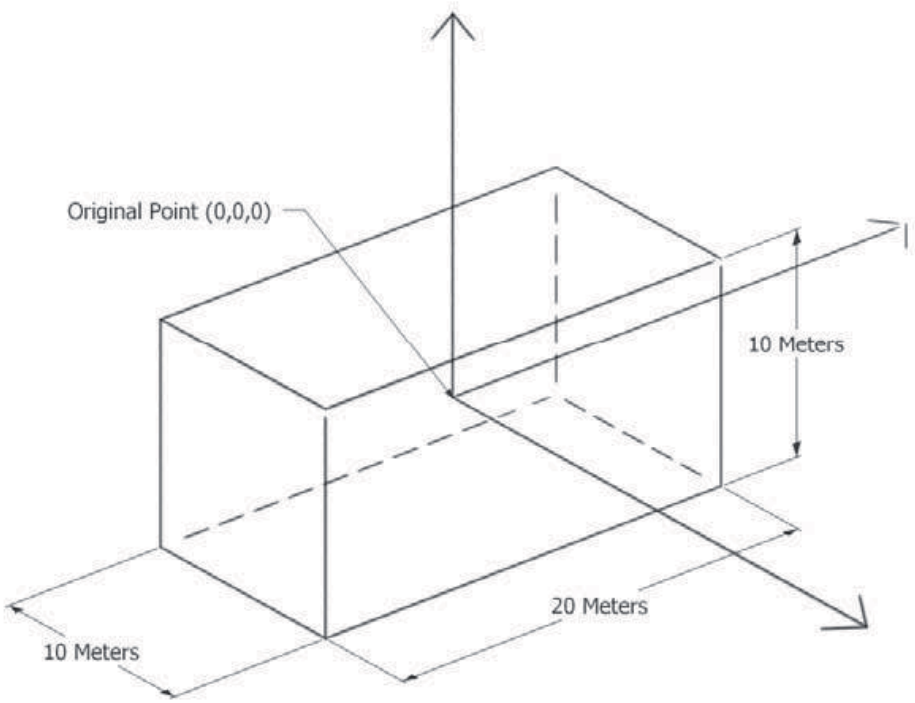


Figure 2.2.3.1
Character 1, Original Box

2.2.3.2. Limitation2: Show Respects to Traditional Buildings

This part is based on the designers' understanding of 'what are the characters of traditional Chinese wooden houses'. In this paper we could not discuss this huge problem deeply. The basic and simplest characters presented below are from the writer's personal opinion. Meanwhile, the feasibility to be translated into quantities is another evaluation that needs to be considered in the process of finding the characters.

Firstly, since there is no number based characters existing (hope someone could write a book about this, which would surly promote the generative architecture design in the domain of merging the tradition and modernity), we need to find some rules behind the traditional shape.

When analyzing a row of the traditional wooden houses along a street, one character is important that all the houses consist of 4 faces: front, back, front roof and back roof. The right and left side are not taken into consideration because they are the common walls shared with the neighbor (built of bricks in order to prevent the spread of the fire), which could not be changed in this project. *See Figure 2.2.3.2a*

Another character is a common rule that Chinese traditional buildings always have roofs. In this problem, the building needs to have at least 2 roofs. [10]
Hence, 2 number based characters are translated.

Character2 (C2), the house needs to have 4 faces (at least).
Character3 (C3), the house needs to have 2 roofs (at least).

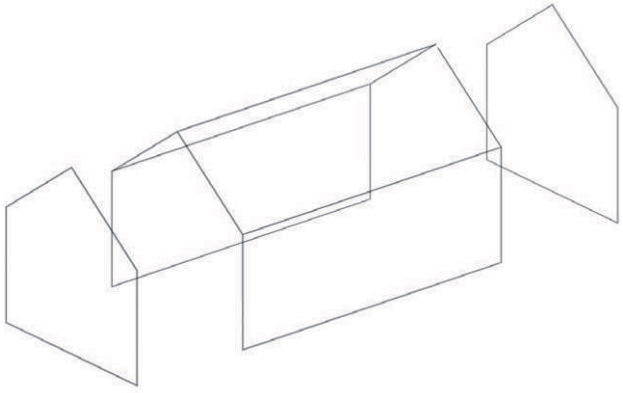


Figure 2.2.3.2a
The basic elements, four faces.With two unmovable walls

Secondly, establish the simplest ways and forms to realize the characters (C2 and C3 together since they could be formed in one action).There are many ways to create the faces. However the best way is that needs the least parameters to control the action. Since the action is subtracting and cutting, it could be considered as defining the position of the cutters. Here are several ways to control the position of the plane. First, get three points and generate the plan through them. Second, get the angles of the plan between XOY and XOZ and get the perpendicular distance from the original point. Third, get one point and two lines which cross on that point.

The second choice is the best since the elements it uses could not be subdivided and easy to control. Moreover, only two angles and one length are needed to make a plan. For the first choice, it's easy to get three points but every point is an array of (x, y, z). That is to say nine parameters are needed to form a plan. The third choice could be subdivided because the lines need to be formed by 2 points first. Actually it needs 4 points (12 parameters) to make a plan. As a result, C2 and C3 use the same function to realize themselves. *Figure 2.2.3.2b*

Finally, we would decide which side they need to be put in. For C2, if it is taken as the fitness functions, the computer counts the number of the faces. This could take a lot of time and low the efficiency of the system. Here we put it into the genetic representation. In another word, when the solutions are randomly created, they would already have at least 4 faces. And for C3, it could also be translated as the genetic representation. When the solutions come out, they could have two faces acting as the roofs.

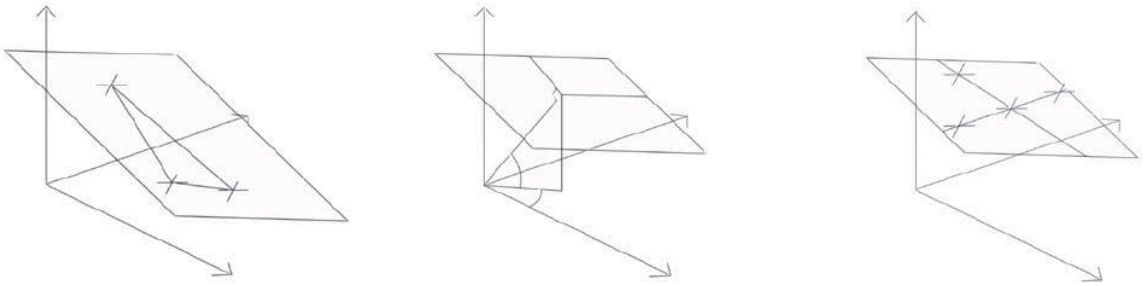


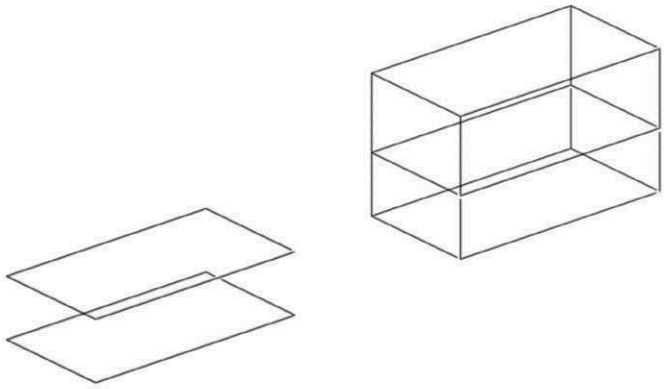
Figure 2.2.3.2b Three ways to make a cutter plan

2.2.3.3. Limitation3: Meet the Functional Requirements

These kinds of limitation are technique and objective so that they are originally number based. The more they are, the more logical the final result would be.
First, the space requirements are listed above. The total sqrmeter needed is 150. It is not necessary to translate it.

Second, the simplest way in this experiment is not to calculate the plan surface and get the square meters since the result we get is the solid instead of surface. The valid space could be taken into consideration. If we calculate the volume instead of the plan surface the problem could be much simplified. So, we translate the square-meters requirement into volumes requirement. This requested volume numbers would be labeled as character4 (C4). *Figure 2.2.3.3*

Third, this requirement is objective and hard to get from the genetic representation (it is much easier to calculate the result to get the parameters) so that it is suitable to be a fitness function. First genetic representation would make the forms. Then this fitness function would check the results. And then we can compare the volume of the results that genetic representation create with the volume requirement to judge whether these results are good or not.



Consider the slope roof, the height of ground floor. The target volume is 1400 m3
200 m2
Figure 2.2.3.3
From sqmeters to volume.
Simplify the process of the computation

Now three limitations are translated into C1, C2, C3, C4 and they categorized into genetic representation or fitness functions. There is no dead rule for categorizing. The only way to differentiate them is to make a balance between human control (determination, preset, genetic representation) and computer control (probability, random, fitness functions).

2.2.4. The Process of the Calculation:

After the translation, now the essential input elements we need to execute the genetic algorithm are ready: C1, C2, C3 are used as genetic representation and C4 is used as fitness function. In the following texts the translation would be put into the process of genetic algorithms. Other basic methods of genetic algorithm, like generation numbers, individual numbers, selecting method, crossover method, mutation method and reproduction method will be discussed.

According to the process mentioned in the introduction part, the steps could be listed as:

1. Input the number of Individuals in one generation and the number of the generation the computer would calculate.
2. Use the genetic representation to initialize the individuals with randomness to get the first generation.
3. Evaluate all individuals by fitness functions to determine their fitness numbers.
4. Select good individuals into ‘Mating Pool’ according to their fitness numbers.
5. Select two parents from ‘Mating Pool’ according to their ranks in the fitness list
6. Use crossover function to generate two new offspring.
7. Randomly mutate the offspring.
8. Get a new generation with new individuals (offspring).
9. Go back to step 3 to start creating a new generation until the number of the generation comes to the set number or the results are satisfying.

2.2.4.1. Step1: Input the Number of Individuals and Generations

The number of individuals decides the scale of the generation. The bigger the scale, the more possibilities the computer creates and the more chance the designer gets unexpected results from the randomness. However, the big scale would also cause enormous calculation. The balance of the speed and quality could be achieved through object-based experiments.

2.2.4.2. Step2: Get the First Generation with Genetic Representation

Before making a huge number of different individuals, we need to know how to create the first individual. The process of making an individual by the genetic representation (which consists of C1, C2, C3) is listed below.

First, take the (0, 0, 0) as the original point. This point would be also the centre point of the original box. *Figure 2.2.4.2a*

Second, randomly get two angles (Angle1, Angle2, from 0 to 360 degrees) and one vector of which the absolute distance is 1. Then first rotate the vector on XOY with Angle1 and then rotate the vector on YOZ with Angle2. Here, according to the C_, the vector should be more likely to form a roof than an elevation.. In order to create roof, we just need to limit the angles below 180 degrees. (the ratio of the number of roofs to that of elevations is 2 to 1) *Figure 2.2.4.2b*

Third, randomly get the distance and multiply the vector with this distance to get the final vector which would control the plan that is perpendicular to it. Since there is possibility that the plan could be generated out of the box (the distance is too big), the error trap is needed. If the plan is out of the box, the distance would be shorted by 50% until the plan touch and cut the box. *Figure 2.2.4.2c Figure 2.2.4.2d*

Finally, the above three steps would be made four times according to C2. After that, the final shape is generated. *Figure 2.2.4.2e, Figure 2.2.4.2f*

Then, computer generates another ‘N-1’ (N is the number of individuals) individuals using the process above. Hence, a whole generation of N different random individuals is created.

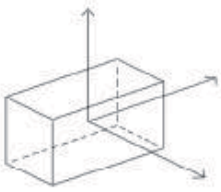
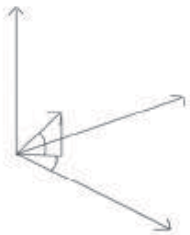


Figure 2.2.4.2a Original Box

Figure 2.2.4.2b



Get Vector by angle1 & angle2

Figure 2.2.4.2c Get the Distance

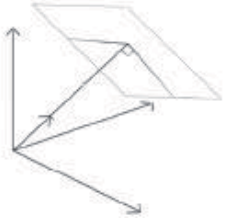
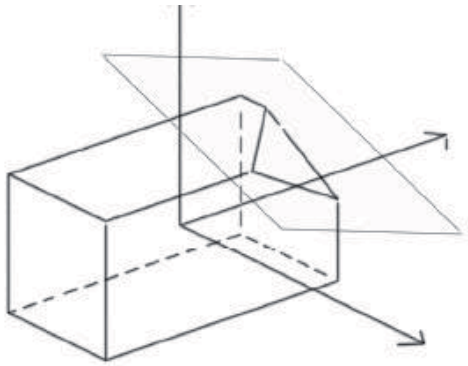


Figure 2.2.4.2d Get the Perpendicular Cutter Surface



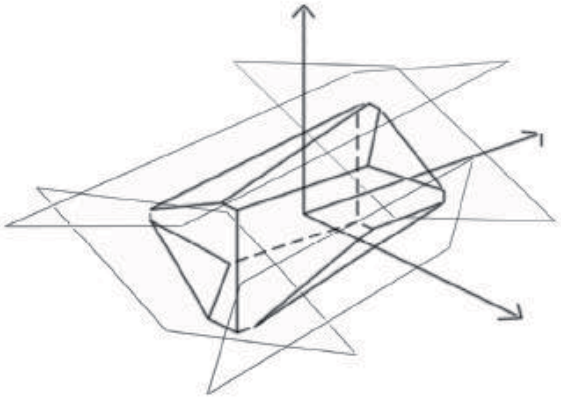


Figure 2.2.4.2e
First Cuting

Figure 2.2.4.2f
Final Form After 4 Cuttings

2.2.4.3. Step3: Evaluate all individuals by fitness functions

The most suitable volume (C4) is translated from the limitation. The closer the volume of individual is to the suitable volume the better this individual is. So, we use the absolute number of 'individuals' volume minus best volume' as the fitness value. The less the fitness value, the more chance this individual would be selected to evolve. After calculating all the individuals, they would be sorted ascending by fitness value.
Fitness Value= absolute Value (Individual Volume – C4 Volume)

2.2.4.4. Step4: Select good individuals into 'Mating Pool'

Since evolution would eliminate the bad and keep the good, a mating pool is needed to temporarily divide the good and the bad. After sorting ascending the individuals, we put best 50% individuals into the mating pool and delete the rest bad ones. *Figure 2.2.4.4*

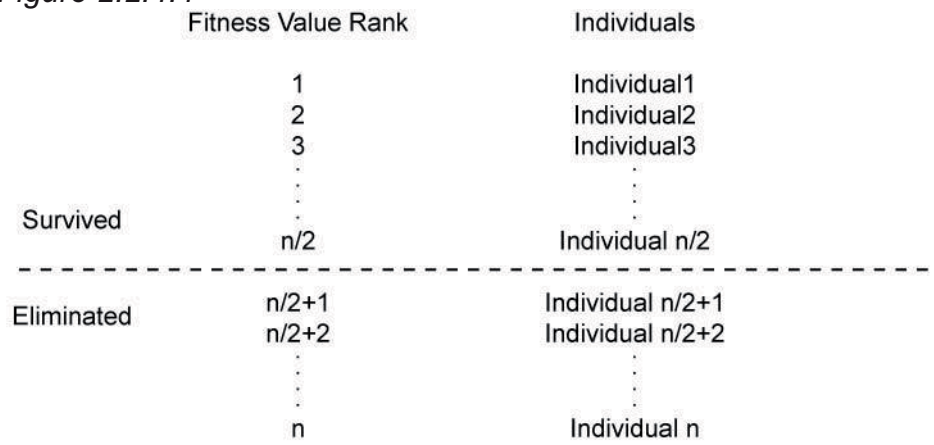


Figure 2.2.4.4

2.2.4.5. Step5: Select two parents from 'Mating Pool'

One of merits of evolution is that the offspring would keep the good features of the parents so that the generations could be better and better. The process of selection of genetic algorithm is analogous to the maintenance of the good features in natural evolution. Before doing the crossover and mutation method, two parents need to be selected from the mating pools. Up to now, there are many methods that could be used in selection. Here we would use roulette wheel selection, which is one of the most popular and well-studied selection methods. [1]

If fi is the fitness of individual i in population, its probability of being selected is

where N is the number of individuals in the population. While solutions with a higher fitness would be less likely to be eliminated, there is still a chance that they may be. *Figure 2.2.4.5*

$$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$$

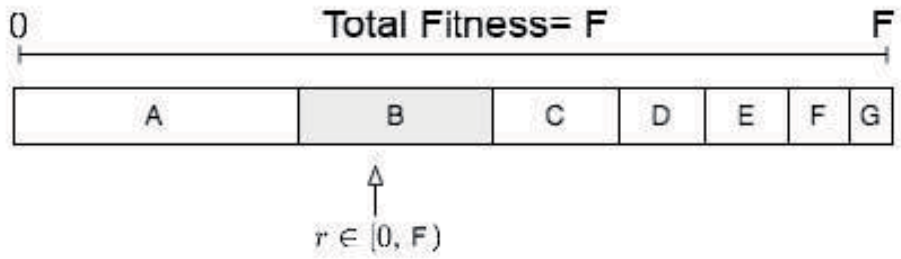


Figure 2.2.4.5 How roulette wheel selection works

2.2.4.6. Step6: Crossover function to generate new offspring

In genetic algorithms, crossover is a genetic operator used to vary the programming of a chromosome or chromosomes from one generation to the next. It is analogous to reproduction and biological crossover, upon which genetic algorithms are based. Here the one-point crossover is used: a single crossover point on both parents' organism strings is selected. All data beyond that point in either organism string is swapped between the two parent organisms. The resulting organisms are the children. See Figure 2.2.4.6a. In this problem, the chromosomes of the individual should be clarified. The tree structure of one individual is listed in Figure 2.2.4.6b. The genes of the two parents are Parent1 (G11, G12, G13, G14) and Parent2 (G21, G22, G23, G24). There are 3 possibilities of the offspring by the one-point crossover method. Figure 2.2.4.6c [3]

The generating of offspring would be ended until the number of the new generation comes to the set generation number.

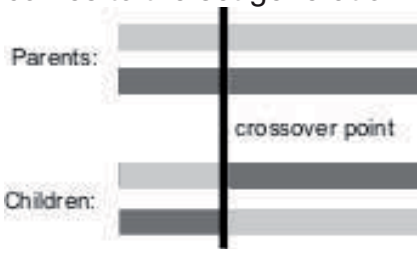


Figure 2.2.4.6a One Point Crossover Method

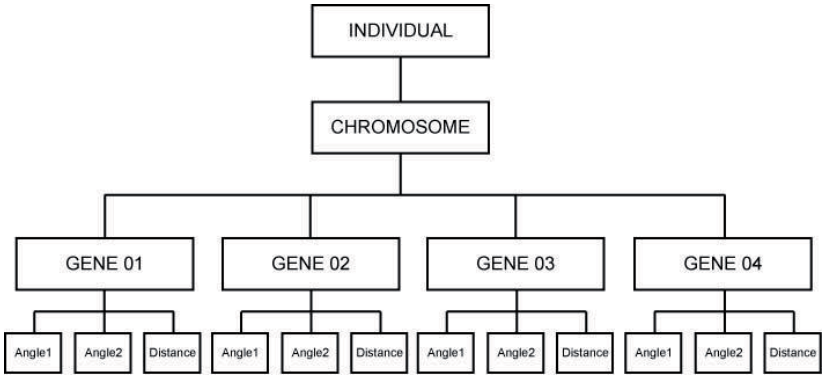


Figure 2.2.4.6b Chromosome Structure of One individual

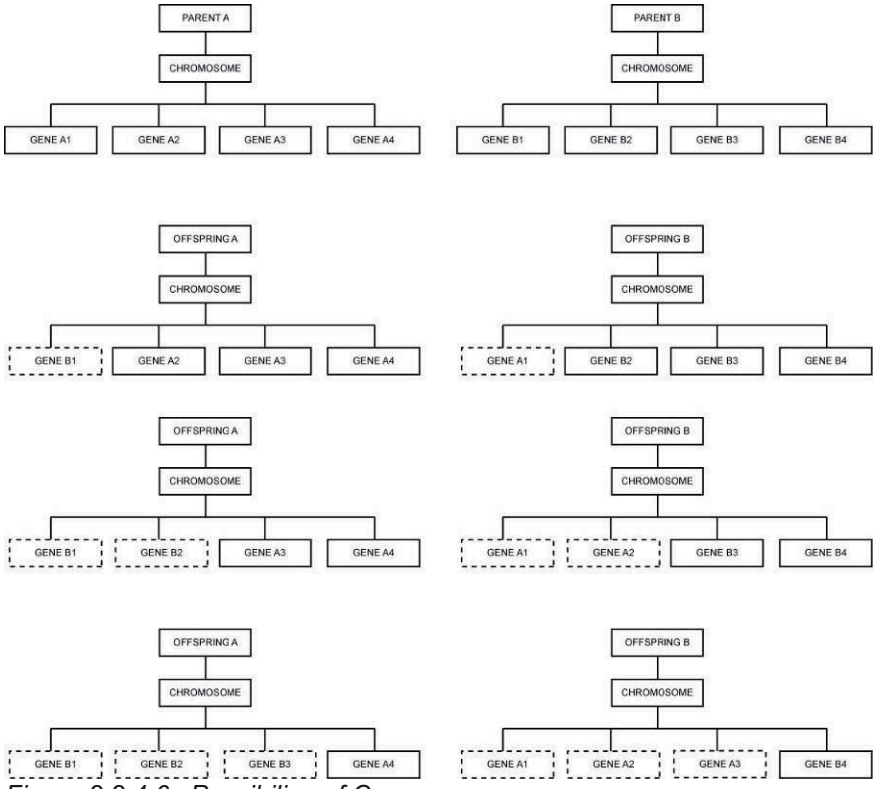


Figure 2.2.4.6c Possibilies of Crossover

2.2.4.7. Step7: Randomly mutate the offspring

The selection and the crossover function would make sure that the good features passed to the next generation. However, the mutation would change the genes of the individuals harshly in order to keep the diversity of the generation. It is analogous to biological mutation. The classic example of a mutation operator involves a probability that an arbitrary bit in a genetic sequence would be changed from its original state. In this problem, the gene of the individual selected to mutation would have the chance to generate a new random parameter. And then the fitness number of this individual would be recalculated. It could be better or worse. However, the purpose of mutation in genetic algorithms is to allow the algorithm to avoid local minima by preventing the population of chromosomes from becoming too similar to each other, thus slowing or even stopping evolution. The action could be seen clearly in *Figure 2.2.4.7* [3]

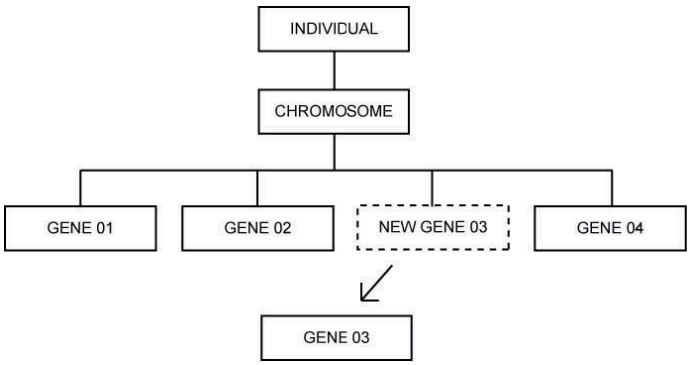


Figure 2.2.4.7
How Mutation Works

2.2.4.8. Step8: Get a new generation with new individuals

After selection, crossover and mutation, the offspring would form a new generation with the set number of individuals.

2.2.4.9. Step9: Start the loop to generate new generations

The step 3 to step 8 is a complete loop for creating a generation. The genetic algorithm could be stopped until the number of the generation comes to the set number or the results are satisfying. The number of the generation is set by the experiment. Moreover, if the results become more similar, that means the genetic algorithm comes to convergence and there is no need to continue the calculation. In these two situations, we can end the genetic algorithm.

These steps discussed above use the simple genetic algorithms. According to the objective project, the algorithms could be changed. For some complicated situations, hierarchical genetic algorithms and parallel genetic algorithms could be used.
Except the input number of the individuals and generations, the detailed chart of the 9 steps is the same in *Figure 1.3b.*

2.2.5. Tools to Execute the Calculation

In this experiment, Rhino, Monkey Editor and Microsoft Access would be used to execute the evolution. The advantage and reasons to use them are discussed below.

2.2.5.1. Rhino and Monkey

The most attractive point of Rhino is that it features a scripting language, Rhinoscript, based on the Visual Basic language, which is one of the most popular and basic computer languages in the world. Moreover, Rhino supports methods of free-form NURBS modeling which stimulates the imagination of the designers. [11]

Monkey is a new script editor in Rhino4 which can be used to edit, run, debug and compile scripts. It contains all the standard programmer editor features such as Find/Replace (with regular expressions), multi-document interface, code trees and integrated help files. It was after the publication of MoneyEditor that people could really combine Visual Basic and Rhinoscript to the best. [12]

In short, Rhino and Rhinoscript provides a platform for designer to create generative architecture and Monkey is the bridge linking the Rhinoscript and VBscript. With Monkey, it is much easier to make the advanced algorithms like genetic algorithms.

2.2.5.2. Microsoft Access and SQL

Since the calculation would generate enormous data that store the information of the individuals, a database is needed. In this experiment we use SQL language to link the Rhinoscript and Microsoft Access. Every individual would have _2 parameters: Generation Number, Individual Number, Fitness Number, String of the Final Form,

Vector1 (the vector direction that decides the perpendicular plan), Distance1 (the distance of the vector1), Vector2, Distance2, Vector3, Distance3, Vector4, Distance4. This information would be useful in the later scientific analysis and reuse the records. *See Figure 2.2.5.2*

GenerationRecords

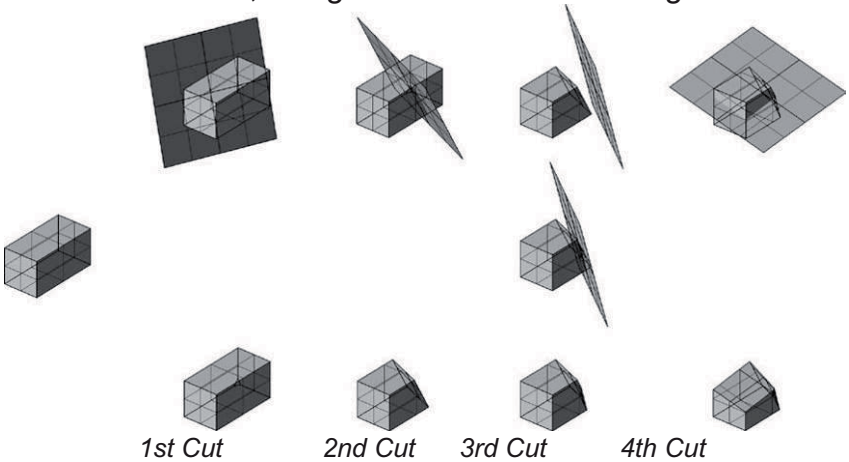
Gene	Individual_	FinalForm	Fitness_Nu	Gene1_Dist	Gene2_Dist	Gene3_Dist	Gene4_Distance
1	1	3f0c4daa-c7a0-489b	277.083346058	4.737537	7.401632	6.460533	2.235407
1	2	018407ea-bfdb-484e	116.227162546	7.004696	4.589689	9.677887	8.68152
1	3	528749b7-16e1-4a11	123.644657513	9.550541	6.33927	9.670876	4.425241
1	4	64537d16-2710-4f91	42.1354529482	8.071362	7.282454	3.350143	7.623552
1	5	a84456b5-d14a-4a28	439.214371544	3.507221	2.718313	6.678947	2.323328
1	6	1a2e902e-ef1e-494a	453.874964749	4.473843	3.055906	2.044495	4.156527
1	7	ecdc9d1e-6489-4d0c	342.525857180	9.83353	4.758612	9.753059	7.615824
1	8	d15e5b31-6121-4df4	215.111895377	2.171124	2.877821	8.203568	5.022387
1	9	6dd1d5a4-d649-4f29	420.791779088	3.422169	2.633261	4.464867	4.489008
1	10	a4ded5e7-b25b-45d2	156.066792188	7.044308	6.847685	5.076752	9.443911
1	11	5bb47ba5-039d-49ff	37.4037290934	6.823185	8.158911	7.977333	8.737558
1	12	413b0953-217b-40fb	822.254260647	9.261974	2.5977	2.178574	6.451982
1	13	3a954279-d9d4-4903	47.2982217456	3.131916	4.561392	8.558769	7.656153
1	14	7e27ff62-06b9-4fe8	28.2869342322	4.038357	7.000181	8.742432	9.358858
1	15	2c21c373-ca07-4aae	611.593664434	2.395115	5.949224	7.19294	2.041056
1	16	90fd35f6-4ec6-449b	345.267098234	8.845623	8.886549	9.002701	7.103014
1	17	cfa7f7ca-4b88-431f	141.297296102	3.782948	5.212423	3.530846	7.804255
1	18	6d27479d-4a38-488a	28.0703035856	8.733456	2.162931	3.905182	3.929323
1	19	96a73233-7456-4fea	66.9199952073	6.578373	4.364172	8.361549	9.71406
1	20	450efe41-fa8c-4df2	17.5653268631	3.248852	4.808821	4.295944	4.320086
1	21	7d3cae66-5c9c-49c0	194.415746072	3.218403	6.678762	2.639397	8.538904
1	22	141e4728-337f-43c1	223.201291913	2.167446	7.752439	5.837708	9.612581
1	23	568eaa08-966b-4367	40.8508186653	8.617954	4.185369	6.02137	6.045511
1	24	0f358797-7eda-4001	468.283011804	2.190166	9.401258	4.471878	6.49016

Figure 2.2.5.2 Microsoft Access Database linked with Rhinoscript with SQL

2.2.6. Visualizing the Calculation in Rhino

With the help of Rhino, the results could be visualized parallel to the running of the script. *Figure 2.2.6.*a show how one individual is made according to the script. *Figure 2.2.6.b* shows how N different individuals are generated for the first generation. *Figure 2.2.6.c* displays the final generation.

After M (the number of the generations) generations or the satisfying results are made, the final generation is presented. Since those best features (genes) of the individual would be passed on generation by generation, it is common that some of the features of different individuals are the same in the final generation. That means in this calculation, this gene dominates all along the evolution. *Figure 2.2.6.d*



1st Cut *2nd Cut* *3rd Cut* *4th Cut*

Figure 2.2.6a The Process of Making a Solution Error Trap Original Box

Figure 2.2.6b First Generation With 40 Individuals

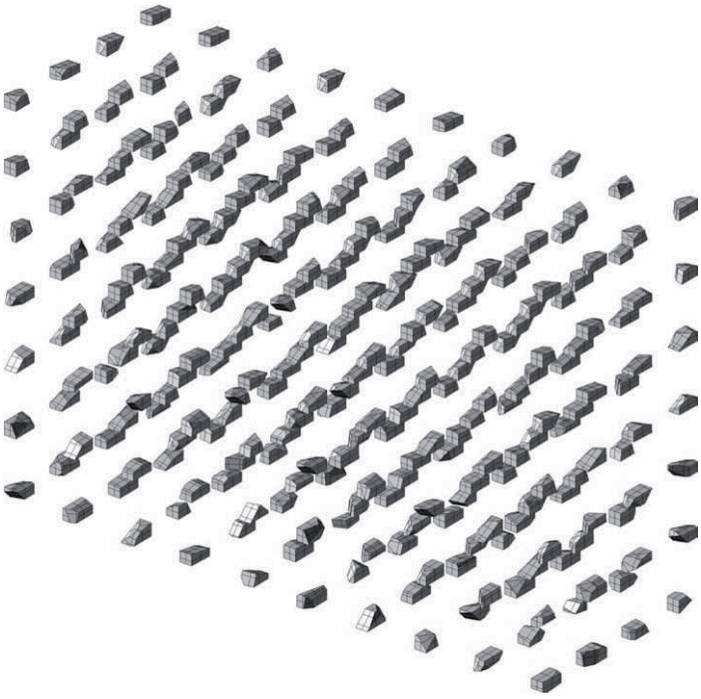


Figure 2.2.6c
After 8 Generations

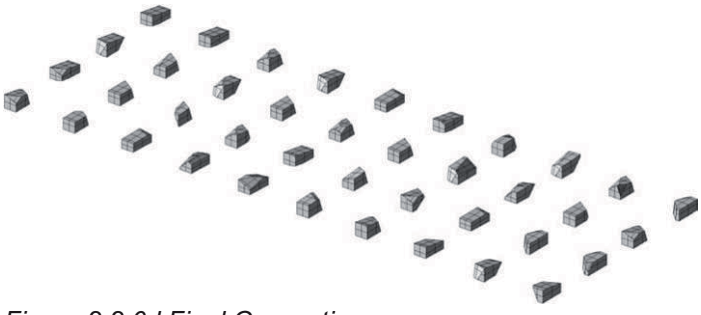


Figure 2.2.6d Final Generation

2.2.7. Final Results:

After running the scripts, the Rhinoscript together with genetic algorithm create hundreds of forms that accord with the limitations we set before. These results are decided by the commands and the limitations so that they are not limited by the empirical ideas of humans. The computer itself creates designs different from any produced by a human designer. Now human designers have hundreds of alternative and unconventional wooden houses suggested by the system.

In this experiment, the results together form a group of project. They should be taken as a whole. Here we will present 9 of the results that the writer prefers. *Figure 2.2.7a*

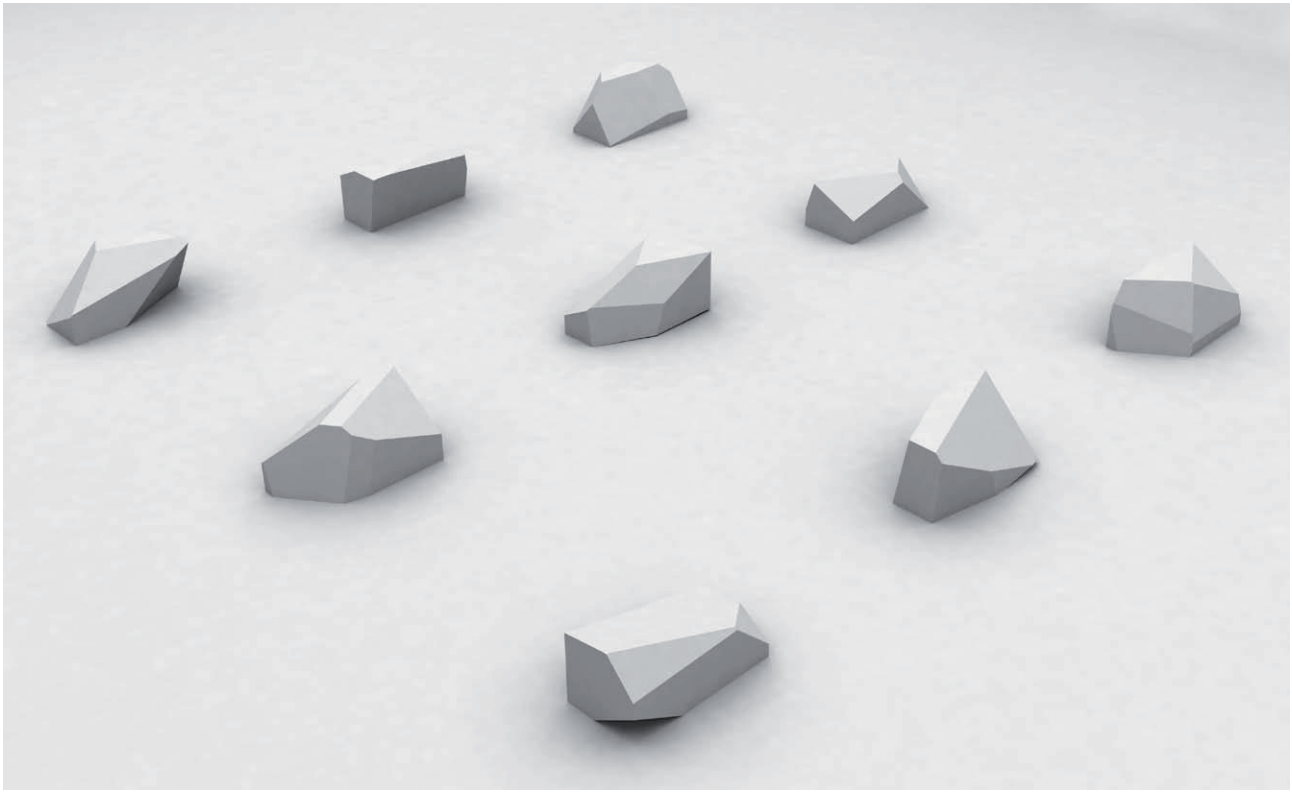


Figure 2.2.7a
Rendered by Vray 1.5, 3DMAX8

It could be imagined that if there are more fitness functions involved into the calculation, the results would be more specific to the certain problems. Meanwhile, the diversity might be still kept because of the mechanism of genetic algorithms so that the user could have lots of alternative solutions.

View From the Back Street Individual 06 is used

View From the Water1 All the Individuals are used

Interior View Of Individual 01 Individual 07 Individual 08

View From the Water Street 2
Individual 07 is used

View From the Water Street 3
Individual 09 is used

3. The System:

With the experiment, the feasibility of genetic algorithm in designing is proved. However, this design is based on a specific situation and the genetic representation and fitness function could not be used directly on other project. In order to apply the system in the other projects, a prototype, expandable computer system capable of automatically creating new designs by genetic algorithms will be developed. The aim of this system is to create designs automatically, or to suggest alternative or unconventional designs to inspire human designers. Since the logics of the system are the same, the software of running the system could be changed to Maya (Mayascript), 3dmax (3dmaxscript), Sketchup (Ruby), etc.

3.1. The Structure of the System

From the experiment, we first analyze the limitations of the project and then translate them into readable parameters for genetic algorithm (the input of genetic algorithm). Second we use computer to evolve the forms by itself by genetic algorithm and get a group of best possible solutions. Finally, we check the solutions and select the ones that are 'beautiful' according to personal judgment. Thus, the structure of the system could be divided into three parts:

1. *Analysis and Translations of Current Project*
2. *Run Genetic Algorithms*
3. *Select Solutions from the Suggested Results*

3.1.1. Analysis and Translations

This part is totally human controlled since how one translates the requirements is based on personal judgment. However, the output of this step is the input of the genetic algorithm so that the results of the translation should be categorized to two elements: genetic representation and fitness functions. In order to control the translation, three steps are brought forward. First, change all the limitations into number based descriptions. Second, find the simplest way to explain the descriptions and numbers into computer readable methods. Third, put this function into either genetic representation or fitness functions. To decide the category of a function is to find a balance between computer control and human control. After these steps, the genetic representation and fitness functions are ready.

3.1.2. Run Genetic Algorithms

The two input elements are ready. Now the genetic algorithm will run automatically and design the shapes by itself. First, the number of Individuals in one generation and the number of the generation should be set which decide the scale and the time. Secondly, the genetic representation initialize the individuals to get the first generation. The individuals of the generation then are evaluated by fitness functions so that good individuals could be selected into 'Mating Pool'. After that, two parents from 'Mating Pool' will be selected out to crossover to generate two new offspring. Finally the offspring will mutate randomly to get the next generation. The loop of evolution starts with the evaluation of the new generation.

In this process, there are many methods that could be discussed deeply. After _0 years development, some key methods of genetic algorithm have evolved into different kinds branches. For example, selection method includes roulette wheel selection, stochastic universal sampling, local selection, truncation selection, tournament selection, etc. [13]

In this system, these methods could be changed according to the objective problems.
After calculation, the final generation and the database will be the outputs of the genetic algorithms. These results would be selected and analyzed again by human being.

3.1.3. Select Solutions from the Suggested Results

The computer provides the designers with hundreds of unexpected and unconventional solutions. The final step of the system will give the control back to human being. However, the aesthetic perspectives are different from person to person. The results could be totally ugly for Tom but quite nice for Jack. In this experiment, the computer is a tool to accelerate the design process, to provide the probability within the limitations and to stimulate the designers with strange but suitable solutions. The final decision should be made by human being.

3.2. The Potentiality of the System

The genetic representation and fitness function could be filled with any different characters translated by designers from real requirements to computer based functions (a module). If libraries for these modules of genetic representation and fitness function are established, new designs could be easy for a user to specify, with the additional modules being selected from the libraries. Moreover, anyone who is interested in doing generative design could translate his or her ‘problems’ into functions and put them to this library. Hence, the library is enriched by the public and could be shared through all the researchers. Ideally most design problems should be specified by the user simply selecting a combination of existing function modules. Also, in the process of GA, the selection methods, crossover methods, mutation methods even the whole structure of GA could be changed according to the library of GA methods which is supported by computer technology scientist.

In conclusion, with the help of the GA system and the potentiality of additional modules, a system for generating the forms automatically is possible, a chart that illustrate how this system works is summarized in Figure 3.

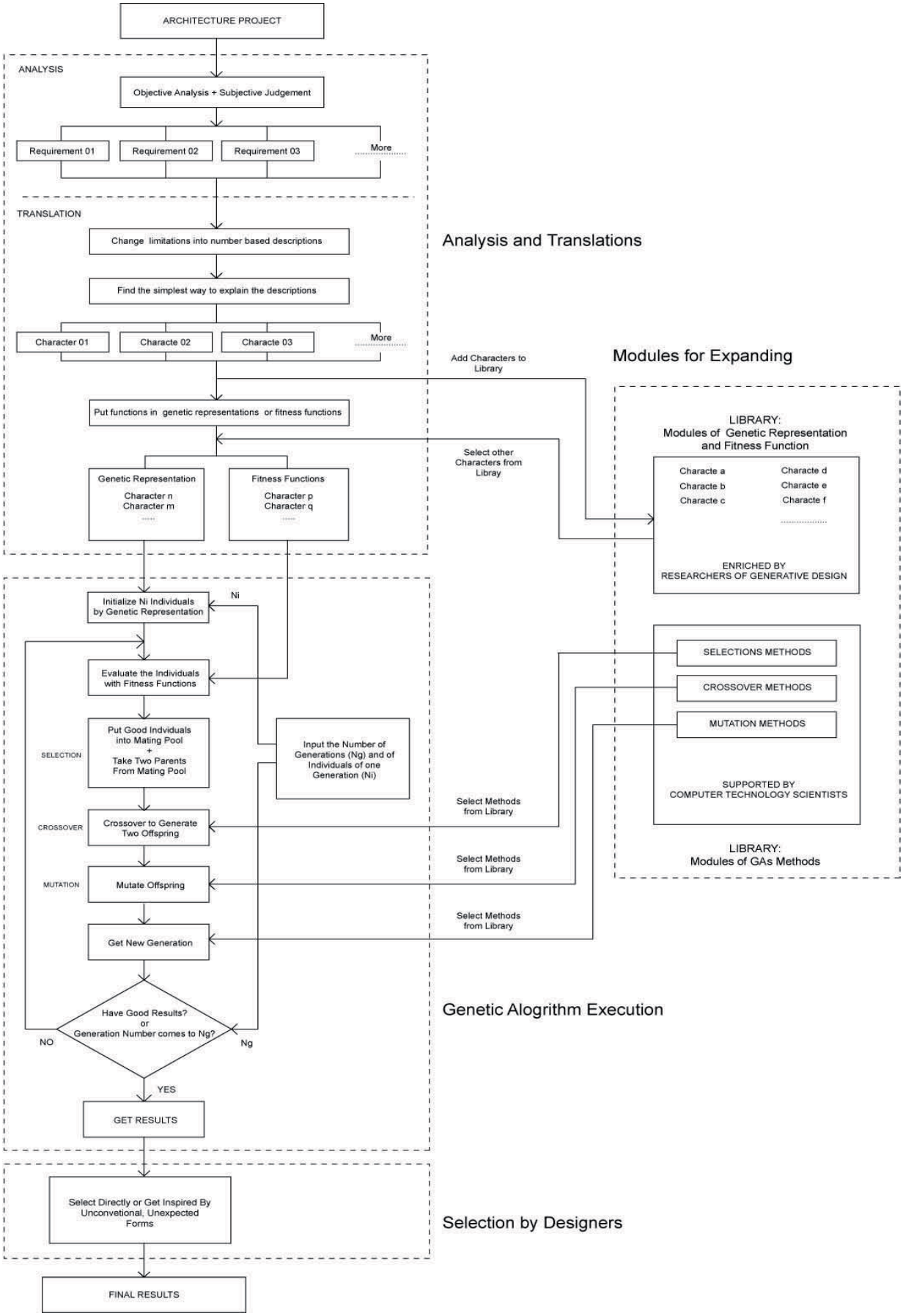


Figure 3 The System

4. Further Discussions and Conclusions

4.1. Categories of Methods in creating generative architecture

Nowadays many methods are used in generative architecture design. There is a common feature in all the generative design methods: the extent that computer is involved into the process of the design. There are two extreme situations: First, totally human controlled. Second, totally computer controlled. The former is like clicking the mouse to draw a line in Autocad, ArchiCAD.etc. The later is like letting the computer generate random parameters without meanings. Now we are in the situation as Terzidis described, 'the shift from computerization to computation'. It's hard to classify every method easily since during the 'shift' methods are mixedly used. [14]

In order to have a general understanding of these methods, _ categories of designs in generative architecture design are presented as following and the key rule to differentiate them is how much extent computer involved.

4.1.1. GUI/CUI Based Design

GUI and CUI means graphic user interference and command user interference. The designer uses the button or command to direct computer to make certain actions. Actually, "entities or processes that are already conceptualized in the designer's mind are entered, manipulated, or stored on a computer system." Terzidis(2008). To make an analogy, the computer is the paper and the mouse is the pen. And people use pens to draw the designs on the paper. [14]

4.1.2. Traditional Algorithms Based Design

Traditional algorithms in design could be described as following:

1. Processes of creating the entities are translated into language (VBA, VB, C++). Computer will generate forms according to the scripts instead of 'mouse clicks'.
2. The computer follows 'a finite sequence of instructions an explicit,procedure for solving a problem'. It could not change or choose the solutions or results by itself.

Most of the generative designs nowadays are made by traditional algorithms like the organic arts made by fractal algorithms, rhinoscript with L-shape method, curve-on-shape method etc. To make another analogy, here we have an automatic machine arm. People design what to draw and then input the instructions into the machine, which will draw the design step-by-step according to the commands.

4.1.3. Evolutionary Algorithms Based Design

'An evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based metaheuristic optimization algorithm'. The procedure of evolutionary algorithms is almost the same with that of genetic algorithms, which is

discussed above. Several EA methods are listed as following: The Shape Grammar, Genetic Algorithms, Stochastic Algorithms, Cellular Automata (Conway's game of life), etc. People only need to tell computer the limitations and the rules of a goal and it will generate the forms by itself.

The relationship of these three categories is summarized in Figure 4.1

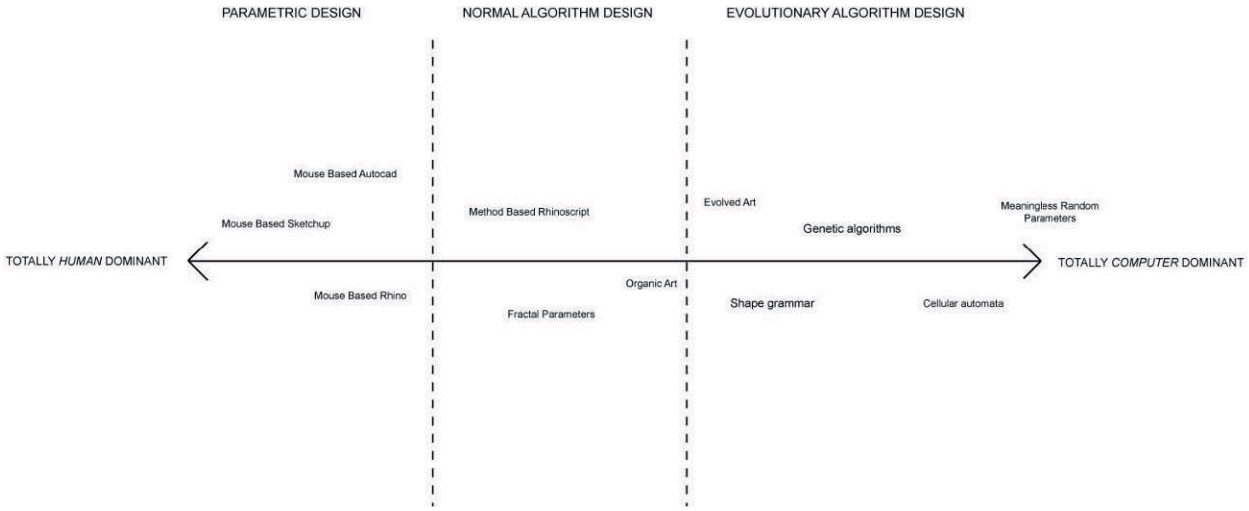


Figure 4.1 The Category

4.2. Intended Capabilities of the System

In the end, the capability of the proposed evolutionary design system is listed below

4.2.1. Evolve designs from a combination of random and user-specified initial values

By initializing the first population of the genetic algorithm with random designs according the genetic representation, the proposed system will be given freedom to evolve any shape that will fulfil the fitness functions. This should allow the system to create novel and potentially unconventional solutions to a design problem.

4.2.2. Solve different design tasks by changing the modules

Designs should be easy for a user to specify, with the modules for genetic representation and fitness functions being required. Ideally most design problems should be specified by the user simply selecting a combination of existing modules from the library.

4.2.3. Evolve designs guided by computer during the process

The system should be able to evolve new designs guided by genetic representation and fitness functions alone. This would not only relieve designers of the tiring task of continuously monitoring the system during evolution, it could also help evolution of more unusual and unconventional ideas. If a human designer guides the evolution of designs by the system, empirical designs will usually be evolved. Preventing human interaction during evolution removes the potential limitation of 'conventional wisdom' from the system.

4.2.4. Evolve useful and innovative designs

The goal of this research is to produce a design system capable of evolving truly useful and innovative solutions to real-world architectural design problems. These designs could either be used directly, or could be used by human designers for inspiration. For this experiment, it is intended that the system should have the ability to successfully evolve acceptable and potentially innovative solutions to architecture form design tasks.

4.3. Conclusions: Design by Evolution

This research has created a conceptual system of generative architecture design by genetic algorithms. This was demonstrated by evolving acceptable designs for a real architectural project.

The system could 'discover' and independently. The shapes of designs were searched by randomness in order to ensure that they are unconventional and by limitations so that these forms perform the desired function accurately. The less constrained the design problems are, the wider the variety of alternative and sometimes highly unusual design solutions the system evolves.

In conclusion, evolutionary design has been performed in nature for millennia. This research has made the first steps towards using power of natural evolutionary design, by demonstrating that it is possible to use a genetic algorithm to evolve architecture design from scratch, such that they are created to perform desired functions, without any human intervention.

References

[1]. Holland, J. H., (1973). Genetic Algorithms and the optimal allocations of trials. SIAM Journalof Computing 2:2, 88-105.
[2]. Holland, J. H., (1975). Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor.
[3].Goldberg, D. E., (1989). Genetic Algorithms in Search, Optimization & Machine Learning. Addison-Wesley.
[4]. Davis, L. (1991). The Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York.
[5]. Fogel, D. B., (1995). Evolutionary Computation: Towards a New Philosophy of Machine Intelligence. IEEE Press.
[6]. Peter John Bentley, (1996). Generic Evolutionary Design of Solid Objects using a Genetic Algorithm. Doctoral Thesis for University of Huddersfield.
[7]. Photo: Satellite of the Site (2009). Position in 31° 6'33.93"N, 121° 2'58.97"E. Image of GeoEye Copyright 2009. Google Earth.
[8]. Photo: Shanghai, Zhu Jia Jiao 2007.05. By blackhumor@126.com. http://www.panoramio.com.
[9]. Photo: The Forgiving Bridge. http://xcw.shqp.gov.cn/gb/content/2005-03/09/content_49801.htm.
[10]. Li Yun He(2005). Design Theory of Chinese Classical Architecture. Tian Jin University Press. ISBN: 7-5618-1902-1.
[11].Description of Rhinoscript and VBA. http://en.wiki.mcneel.com/default.aspx/McNeel/RhinoScript.html.
[12].Description of Monkey. http://en.wiki.mcneel.com/default.aspx/McNeel/MonkeyForRhino4.html.
[13].Wang Xiaoping, Cao Liming(2002). Genetic Algorithm-The Theory, Implication and Software Realization. Xi'an Jiaotong University Press. ISBN: 7-5605-1448-0.
[14]. Kostas Terzidis (2006), Algorithmic Architecture. Elsevier Science & Technology Books. ISBN-13: 9780750667258.