Controlled Unpredictability: Constraining Stochastic Search as a Form-Finding Method for Architectural Design

Daniel Cardoso
Department of Architecture,
Computation Group
Massachusetts Institute of Technology

Provided with a strict set of rules a computer program can perform the role of a designer. Taking advantage of a computer's processing power, it can provide an unlimited number of variations in the form while following the same set of constraints. This paper delineates a model for interrelating a rule-based system based on purely architectural considerations with non-deterministic computational procedures in order to provide controlled variations and constrained unpredictability.
The experimental model consists of a verisimilar architectural problem, the design of a residential tower with a strict program of 200 units of different types in a given site. Following the interpretation of the program, a set of rules is defined by considering architectural concerns such as lighting, dimensions, circulations, etc. These rules are then encoded in a program that generates form in an unsupervised manner by means of a stochastic search algorithm. Once the program generates a design it is evaluated, and the parameters on the constraints are adjusted in order to produce a new design.

This paper presents a description of the architectural problem and of the rule building process, images and descriptions of three different towers produced, and an evaluation of the algorithmic process used for generating the form.

The succesful evolution of the experiments show how in a computation-oriented design process the interpretation of the problem and the rule setting process play a major role in the production of architectural form, outlining the shifting role of human designers from form-makers to rule-builders in a computation-oriented design endeavour.

1. Introduction

How can we devise a dialogue between designers and computers that takes advantage of the inherent capabilities of both? Recent architectural thinking has been concerned by this and other questions surrounding the changes that current technology claims from the architectural practice and from the form-generation process in particular. Concerned with the creative limits imposed by software tools architects and designers have started to value a deeper understanding of the computer beyond the graphic user interfaces and the standardized tools provided by CAD packages (Manovich 2001). End user programming languages have allowed designers to take advantage of the potential of the computer's processor to perform a very large number of calculations manipulating complex geometry through scripting; this new kind of interaction with the computer is still new for most practising designers, yet has proved to provide an extended field for design exploration (Loukissas 2003). The autonomy implicit in the process of generating architectural form through code suggests a shift in the role of the designer from form-maker to rule-builder or breeder of architectural forms (De Landa 2000). Following this line of tought it can be inferred that constraining a design space is a heavily charged design act. Providing a concrete example of this assertion is the main goal of the experiments presented in this document.

In this paper I put forward a concrete example of an active dialogue between computer and human designer that provides different solutions to a specific architectural problem. Taking the program and site for a residential tower as a theme, I document the rule-building process and the implementation of these rules in a Stochastic (Random) Search program written in MELScript. This document clearly shows how adjustments made in the set of rules after evaluating the

output of the program change in a substantial way the new designs produced by the program.

In the following section of this paper I will describe the architectural program for the tower, and how it led to the definition of a set of architectural rules or constraints; in the third section I will explain how these rules are encoded in a MELScript program that produces correct towers in a non-deterministic manner using Stochastic Search, and I will show images and descriptions of three different towers produced by the program, explaining how changing the parameters of the program affected the search space and thus the resulting form.

The final section examines the generality of the results, and suggests some discussion points on the implications of the experiment regarding authorship in design.

Relevant precedents of non-deterministic form-generation strategies

The use of non-deterministic methods as creative force is present in XXth century art in the works by Ianni Xenakis like Phitopracta, Diamorphoses, and N'Shima; interestingly Xenakis often relied on computers for the interpretation of this pieces. The setting of rules for musical interpretation in some works by composer Pierre Boulez is also a precedent of constrained unpredictability; in his Third Piano Sonata (1955) the interpreter could choose form a prefixed range of alternatives provided by the composer thus altering the structure of the piece; Boulez called this, conveniently, Controlled Chance. Cage's music of changes by John Cage is one early example of a musical piece composed entirely through random procedures. The active role of the interpreter on an open, cross-referenced text was developed from the semiotic perspective by Umberto Eco in his Open Work (Eco 1962). In the domain of architecture the underground station Iidabashi in Tokyo by Sei Watanabe relied on computer generated proposals and human-computer evaluation for developing the design (Watanabe 2002).

2. Methodology

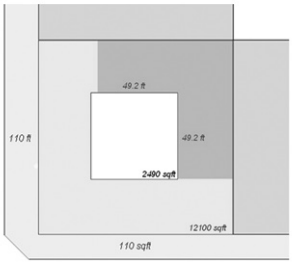
2.1 Interpreting the program

In this section I describe the architectural program and the considerations taken into account by the human designer in order to develop the set of rules.

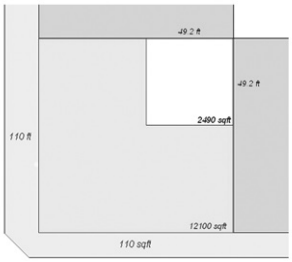
The program requires the building to place 200 residential units in a flat urban site with dimensions 110' x 110' on a corner of two streets. The design should leave at least 20% of the area to circulations. The units to be placed are divided into three groups: fifty efficiency units, 400 sqft each, one hundred one-bedroom units, 900 sqft each, and fifty two-bedroom units, 1600 sqft each. It is a desirable condition for the units to face the street.

The corner condition of the site and the proximity of the neighbours makes the north-east corner of the site inconvenient for residential units because of poor lighting condition and views. Looking for an arrangement that provides a maximum of street façade to the apartments, the core is located in the north-east corner of the site (Figure 1B).

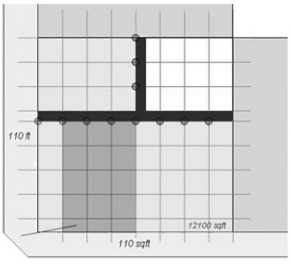
The three different types of units are set to have a common width in order to allow for a regular distribution along a modular three-dimensional grid; the modularity of the grid has a logical



A



B



C

and a constructive reason that relates to the continuity of the structural frame. The depth is variable depending on the required area of each unit. Unit C is split into two floors in order to meet the area requirement; the small width of the units will guarantee that a larger number of apartments have façade on the street.

The width of the unit is related to the total size of the site, and it defines a grid that will serve as the skeleton for the whole structure (Figure 1C). The regularity of the grid is altered to allow all of the units to have optimal lighting conditions, providing more circulation area in front of the units located in the south-west corner of the building. This increase in the circulation area generates a new constraint, because the slots in the south-east quadrant of the tower are too small to allocate type B Units. The circles on the grid indicate the insertion spots for the units. For providing common spaces, a number of empty spaces will be allowed in each tower, however this number should never be bigger than 20.

The overall orientation of the tower is guaranteeing that most apartments have optimal lighting conditions from the south.

2.2 Implementation in a script

In this section I show how the considerations made in the previous one define a set of rules that is encoded in a compuer script.

An individual (a tower) is an arrangement of randomly placed units along the three-dimensional grid of insertion points defined in the previous section. The maximum number of floors of the tower is a parameter that can be adjusted. As a result of the human-driven problem evaluation and constraint setting process, each individual or tower must fullfill the following rules:

- The units are to be placed on the three-dimensional grid of insertion points. No more than one unit can be allocated in a insertion point
- A Type B unit cannot be placed in any of the south-west corner insertion points
- A Type C Unit has two stories, thus it occupies two insertion points
- A succesful tower must allocate all of the 200 units
- There cannot be more than ten unused insertion points in one individual

At each iteration of the program, an insertion point is selected randomly, and once it has checked if the point is not already occupied by a previously placed unit, a unit of type random is inserted. If it's being used, go to another random point and follow the same procedure. The data structure of a tower is as follows:

((1,1,3,3,0,3,2,2,2,3,4) , (2,3,3,2,1,1,1,0,3,1,1) , … and so on for evey floor until the required number of units is reached)

Where 0 stands for an unused insertion point, 1 stands for an insertion point with an efficiency, 2 stands for an insertion point with a 900 sqft unit; 3 stands for an insertion point with a 1600 sqft unit. Conceptually, this data structure can be described as an array of floor elements. Each one of these elements holds the information about the type of it's respective units; notice that the number of elements in a floor equals the number of insertion slots in the floorplate (Figure 1C). There are two different cases for the termination of the program: the first one occurs when the conditions are met and a 'correct' individual is created. The second one occurs when the maximum number of iterations is reached without having a solution: the script asks the user to increase the search field.

2.3 Experiments

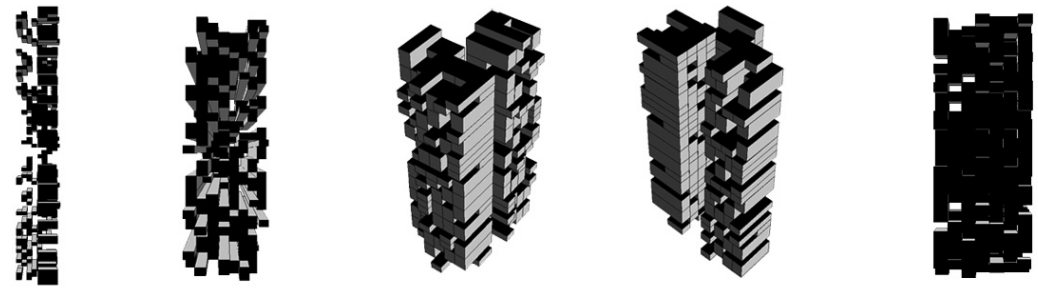
2.3.1 First Tower

Search field (stories) = 80
Minimum Empty Spaces allowed = 20
Units Placed:
Type 1 : 50
Type 2: 100
Type 3: 50
Type 0: 20
Floors: 78
Iterations required to complete this tower: 820
Success: Yes

The search field defines the maximum number of floors. The tower is interesting as a form, but totally unfeasible as a building because of the discontinuity of the volumes. (Figure 2)

2.3.2 Second Tower

Search field = 40
Minimum Empty Spaces allowed = 20
Units Placed:
Type 1 : 50
Type 2: 100
Type 3: 50
Type 0: 20
Floors: 39
Number of iterations required by the algorithm to find this solution: 2311.



First Tower Second Tower Third Tower

For this tower the search field is reduced to 40 in order to produce a denser tower. The second tower is twice as dense as the first one, and equally succesful in terms of full placement of units. (Figure 2)

2.3.3 Third Tower

Search field = 29
Minimum Empty Spaces allowed = 0
Units Placed:
Type 1 : 50
Type 2: 88
Type 3: 50
Type 0: 20
Floors: 29

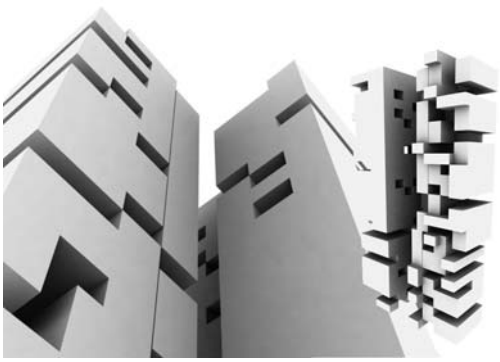
For this tower, the search field will be reduced to 29 in order to generate more density. The Stochastic Search doesn't output a correct tower. This tower has still 12 Type B Units left top place. The number of iterations required to get to this solution: 250 000. The brute force of the Stochastic Search Algorithm is not sufficient to find a correct solution within this search space (Figure 2).

3. Discussion

The methodology presented in this paper provides a scheme of interaction between the designer and the tool in the process of defining a search space through constraint setting and the synergy between the algorithm and a simplified but verosimile architectural problem through human rule-building and evaluation. This is a particular methodology for design exploration based on the definition of a set of rules and the implementation of a computer program that provides solutions of a given design problem. It shows how a clear division between the rule-building process and the actual production of form fosters an active dialogue between the computer and the human designer. The Stochastic Search Algorithm plays the role of the form-maker, whereas the human designer plays the role of the rule-builder and evaluator. The implemented Stochastic Process is a search algorithm that iterates over a given spatial structure assigning alleatory architectural elements to specific spots until a set of conditions defined by the rules is met. The alleatory nature of the algorithm is intended to make evident the distinction between the rule-building and a non-deterministic form-making process, therefore showing to what extent in computer-oriented design problem interpretation, rule-building and evaluation are the most important design acts.

The tension between random methods and design constraints is a viable alternative to computational design. This paper does not try to assess that the proposed methodology is the only way to stablish this dialogue, but it does highlight the fact that computational design is constraint based in nature.

It is often said that digital technologies will make the human designer disappear; in this



paper I show how computational design systems are an open field for new design methodologies and strategies that rather than making human designers disappear, will change their role enabling them to create new kinds of dialogues with the technology, and therefore new scenarios for design exploration.

References

DeLanda, Manuel.(2001). Philosophies of Design: the Case of Modelling Software. Verb: Architecture Bookazine. Madrid: Actar Press, 2001.

Eco, Umberto, The Open Work [Opera aperta](1962). Cambridge: Harvard UP, 1989.

Loukissas Yanni, Rulebuilding: explorign design worlds through end user programming Thesis (S.M.)--Massachusetts Institute of Technology, Dept. of Architecture, 2003.

Makoto Sei Watanabe, Induction Cities (2002). Birkhauser Verlag AG.

Manovich Lev, The Language of the New Media(2001). The MIT Press.