Multiple-constraint Genetic Algorithm in Housing Design

Taro Narahara
Massachusetts Institute of Technology
Kostas Terzidis, Ph.D.
Harvard University

Abstract

As architectural projects are becoming increasingly more complex in their formal manifestation as well as in their functional requirements, methods are sought to address these complexities. Genetic algorithms offer an effective solution to the problem allowing multiple constraints to compete as the system evolves towards an optimum configuration that fulfills those constraints. A case study is presented that involves a housing project with multiple environmental, functional, and economic constraints.

Introduction

A genetic algorithm (GA) is a search technique for optimizing or solving a problem. Its mechanism is based on evolutionary biology, using terms and processes such as genomes, chromosomes, cross-over, mutation, or selection. The evolution starts from a population of completely random individuals and happens in generations. In each generation, the fitness of the whole population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), modified (mutated or recombined) to form a new population, which becomes current in the next iteration of the algorithm.

While some genetic algorithms are aimed to solve a pre-defined problem (e.g., guess the specific string 001010) they can also be modified to address a problem whose solution is not known in advance, but can be described through attributes or other indirect characteristics. In that sense, the operation of matching can be applied toward a desirable principle instead of a fixed pattern. For instance, if symmetry were the sought out principle instead of matching a fixed pattern (i.e. 001010), then a series of other patterns as potential
solutions may have emerged, such as 100001, or 011110, or 101101, or 010010, etc.

Genetic algorithms originated from the studies of cellular automata, conducted by John Holland and his colleagues at the University of Michigan (Holland 1992). Research in GAs remained largely theoretical until the mid 1980s, when a dramatic increase in desktop computational power allowed for practical application of the new technique.

Unlike a shape grammar, a genetic algorithm uses stochastic selections based on the fitness measures from multiple individuals. A shape grammar consists of a vocabulary of shapes including an initial seed shape and a set of shape rules, and is effectively used to describe and generate styles and languages of design. The critical difference lies in the generative process: optimum versus deterministic. The majority of production rules in a shape grammar are context sensitive, deterministic in their manners, and do not implement a fitness measure for their outcome. Choice of the methods for the generative manner can be conditional on their use and purpose, although a combination of the two—such as using the sequence of rule applications in a shape grammar as genotypes for the genetic algorithm—can be an interesting future exploration as it takes advantage of both methods.

In architecture, GAs are of special interest mainly because of their ability to address a problem offering a multiplicity of possible solutions. Contrary to other algorithms where the objective is to accommodate a manually conceived part or diagram, GAs are emergent procedures that evolve over time through multiple attempt cycles (i.e. generations) and therefore offer a bottom-up approach to design. In addition, by using the computational power of computers, they can resolve complex interactions between multiple factors and under multiple constraints offering solutions that occasionally surprise the designer.

The Problem

One of the main problems in architecture today is the quantity of the information and the level of complexity involved in most building projects. As globalization and economic development have started to rise at unprecedented levels, the need for large urban developments has become commonplace. Housing projects for thousands of people have started to emerge over large urban areas. In such cases, the old paradigm for housing design was the development of high rises that served as stacking devices for multiple family housing units. Such a direction was unfortunately the only way to address excessive complexity using manual design skills mainly because it was simple to conceive but also simple to construct. The unfortunate nature of this approach lies in the uniformity, similarity, and invariability that these projects express, in contrast to the individuality, discreteness, and identity that human beings and families manifest.

Genetic algorithms solve local neighborhood behavior and then move further to resolve global to the system issues. In that way there is an emergent behavior embedded in the process of deriving possible solutions to a design problem. This behavior is based on the premise that individual units under certain
constraints may emerge into globally functional configurations by resolving their local neighboring conditions in a repetitive manner. Contrary to common belief, such seemingly chaotic local behavior does not necessarily result into chaotic overall behavior, but rather into an evolved form that solved the local constraints.

One of the main areas of complexity that could benefit architecture is in housing projects. In these projects there is a typology of residential units that need to be combined in various schemes that will fulfill multiple functional, environmental, and economic constraints. While small apartment buildings may be solvable within one architect’s design capabilities, the design and planning of large projects with several thousand inhabitants is a challenge. The problem here is to fulfill all complex requirements without using conventional repetitive high-rise patterns.

Methodology

The design of a 200-unit residential complex on a corner of two streets in an urban context was investigated. Three types of units—50 efficiencies, 100 one-bedrooms, and 50 two-bedrooms—are given as a required number of units to be arranged, and each unit’s value is evaluated based on the solar exposure, view, and construction economic factors. Creating the structure to satisfy the above requirements for the lowest cost became the main agenda for the project. Beyond the certain quantitative factors and the complexity of the problem, a search process was initiated that challenged the unpredictable domain of the human perception. Thus, a Genetic Algorithm was selected as a design/optimization method for this project.

First, arrangements of the units are mapped (encoded) into an artificial chromosome of a certain fixed size of strings. In this case, it was 200 characters composed of three different ASCII codes denoting three different types of units, (“1” for efficiency, “2” for one-bedroom, and “3” for two-bedroom) and they had to maintain the total numbers of units (50, 100, 50 respectively) regardless of the arrangements orders. Initial populations of 100 strings were randomly created for the manipulation using the genetic algorithm, and the fitness measure (function) was defined based on the above constraints factors. More precisely, each unit facing the street gained higher scores, and providing more floor levels affected the construction costs and unit’s price. This fitness function assigns a fitness value to each chromosome in population. In this case, function provides an evaluation score to a unit arrangement pattern of each building scheme. The higher a floor level, the more the value for the units are increased by gaining better views; yet adding another floor levels costs more in construction fees, which will be deducted from the total fitness value. These contesting constraints are commonly seen in real architectural programmatic issues, and are increasing the complexities to the design problems. Under such multiple constraints, search by the conventional deterministic methods may not be always effective. Over the course of history in design, decisions tend to be determined in a singularly deterministic manner, although a dramatic increase in desktop computational power opened new potential for designers to solve the problems by optimizations. The genetic algorithm manipulates a population...
by using the operations of reproduction, crossover, and mutation.

To accelerate and enhance the optimization process, self-crossover operation within the chromosome (as a string) was executed. Single string is cut into three parts at the randomly selected positions within it (A-B-C) and rearranged into all six possible orders (A-B-C, A-C-B, B-A-C, B-C-A, C-A-B, and C-B-A). Out of these six newly reproduced chromosomes, only the one that has the best fitness value was selected as a new population member so that the new population set can constantly contain the selected chromosomes to evaluate. This method also does not violate one of our constraint factors to maintain the total number of units for each type, 50, 100, and 50.

Finally, optimized solutions were sampled back into the search space of the project and decoded into the building structure schemes by using visualization functions.

The best chromosomes from the population of every one hundred generations and up to one thousand generations were sampled as building schemes. Up to the first one hundred schemes, there were rapid changes and improvement in the units’ arrangement patterns. After one thousand generation,
Figure 3. Possible design solutions.

Figure 4. Design possibilities in the form of a pedigree (left) and one chosen design solution (right).
reproduction slows down to show the convergence into a certain characteristics of the arrangement schemes. Efficiency units with smaller square feet area shows tendencies to be collected at the lower area facing the streets sides to gain more units with higher solar exposure, and vice versa, two-bedroom units with larger square feet area occupy higher floor levels of the building.

**Discussion and Critique**

Architectural design has a long history of addressing complex programmatic requirements without a specific design target. Unlike other design fields where the target is to solve a particular problem in the best possible way, architectural design is open-ended, flux, and uncertain. Codified information, such as standards, codes, specifications, or types, simply serve the purpose of conforming to functional requirements, yet are not guarantees for a successful design solution.

In large housing projects, the typical approach is to design a plan resolving the complexities within a single floor. Then copies of each floor are stacked one on top of another to form a high-rise. Then copies of high-rises are made until the number of required units is met. While this approach may be considered complex enough for manual design, it is certainly too simple if not naïve for computational design. Complexity in design today can be addressed through computational means that can resolve a problem without the designer knowing in advanced its formal solution. Instead, a series of constraints alone can be enough to produce algorithmically formal solutions that resolve complexities that surpass a designer’s capabilities.

While complexity may be a characteristic of many natural systems or processes, within the field of design the study of complexity is associated with artificial, synthetic, and human-made systems. Such systems, despite being human creations, consist of parts and relationships arranged in such complicated ways that often surpass a single designer’s ability to thoroughly comprehend them even if that person is their own creator. Paradoxical as it may appear, humans today have become capable of exceeding their own intellect. Through the use of intricate algorithms, complex computations, and advanced computer systems, designers are able to extend their thoughts into a once unknown and unimaginable world of complexity. Yet the inability of the human mind to single-handedly grasp, explain, or predict artificial complexity is caused mainly by quantitative constraints, that is, by the amount of information or the time it takes to compute it and not necessarily to the intellectual ability of humans to learn, infer, or reason about such complexities.

Nonetheless, in genetic algorithms a paradoxical phenomenon occurs: while a programmer has conceived an algorithm that will address a specific problem, the same algorithm might be used to address another completely different problem that was never predicted by the original author. Further, it is possible that using the same algorithm but utilizing different parameters than the ones that were originally designed may result in a behavior that is completely unexpected. Consequently, when a designer uses an algorithm to design, the designer may not be aware, knowledgeable, or conscious of the mechanisms, specifications, or repercussions of
the programmer’s algorithm. The gap of discrepancy that separates the programmer from the designer is indeed problematic mainly because of the nature of algorithms. While the quantity and composition of external data may appear to be infinite, random, or incoherent logical filtering will lead progressively to an ordered formation. Unlike blind randomness, genetic algorithms are capable of selectively controlling the shaping of information. Such algorithmic events result from factors that are neither arbitrary nor predictable yet seem to be guided by some sort of intelligence. While these events are made possible by simulating natural processes without involving human intelligence, it is inevitable to assume that some human intelligence is involved in the selection of the natural process that best fits the problem of randomness.

Traditionally, designers maintain full intellectual property over their designs or manifestations thereof, based on the assumption that they own and control their ideas. This is not always the case with algorithmic forms. While the hints, clues, or suggestions for an algorithm may be intellectual property of the designer-programmer, the resulting tangible or virtual representations of those ideas is not necessarily under the control of their author. Algorithms employ randomness, probability, or complexity, the outcome of which is unknown, unpredictable, and unimaginable.

**Conclusions**

Recent advancement in tectonics and structural engineering enables the realization of buildings in mega scales and starts to introduce another layer of complexity into the building programs. Conventional design methods relying on the preconceived knowledge based approaches are no longer reliable. Beyond the certain quantitative factors and the complexity of the problems, search occasionally enters into the unpredictable domain of the human perception. Computational approaches to design allows us to go through thousands of iterations in a second and find the solution sets beyond the reach of designers’ intuitive search spaces. Genetic Algorithm can be a potential derivative for finding optimum design solution from indeterminate search spaces constrained by multi dimensional factors.

**References**


Elezkurtaj, Tomor and Georg Franck.