Generative Design in an Evolutionary Procedure

An approach of genetic programming

Hung-Ming Cheng
China University of Technology

Keywords: Artificial intelligent, Genetic algorithm, Generative design tools, Procedural design studio, Design exploration

Abstract: This study describes a procedural design studio using Genetic Programming as the evolutionary mechanism and formal generation. This procedural design is integrated with a visualisation interface, which allows designers to interact and select from instances for design evolution. Evolutionary design facilitates designers in three areas: 1) diversify instances of design options; 2) inspect specific goals; 3) and enhance the possibility of discovering various potential solutions.

1. INTRODUCTION

In essence, design and designing involve different disciplines that are influenced by participants, knowledge, and information from various domains. For such design frameworks, design problems require a procedure to reconcile multiple viewpoints that are distinguished by particular interests and emphases. For example, a design team consists of architects, engineers, and constructors, with each of them concerning issues/aspects of design from different angles. An architect would be interested in aesthetic and figural aspects of design, a structural engineer concerned about structural members and the underlying reasonable, while a constructor focusing on building costs and other construction issues. They derive their views based on different disciplines and professions that require an integrated method for searching potential solutions. The design task with problem frameworks provides an exploratory nature of design procedure, which also offers...
profound approaches and solutions. This is why we employ such mechanism to explore the design space from various domains of design knowledge.

Genetic programming provides a way to genetically breed a computer program to solve a wide variety of problems. The recently developed genetic programming search the space of possible computer programs for a highly fit individual computer program (Koza, 1992). Search is a framework for problem solving where alternative solutions are evaluated in a trial-and-error iterative circulation. The meaning of design search is to find out alternatives that lead to good solutions. Searching in enumerative space of design solutions is unlikely, the ideal heuristics such as genetic programming use rule-based method as a guidance for searching alternative solutions. Genetic programming is derived from the hint of Natural Selection (Darwin, 1859) to decide the ordering of alternative solutions, which has been applied to a wide range of problems in combinatorial optimization, automatic programming and model induction.

The evolutionary procedure applies genetic programming as algorithmic method that evaluates and refines the design during conceptual formation. The design procedure integrates synthesized solutions of design teams. The evolutionary procedure reflects different disciplines of designers that collect the fitness from each designer’s selection. To explore the design space, we need appropriate representations and procedures to generate the representation of the design task. The generative design tool uses an iterative approach that refines design by evaluating candidates in the process of genetic programming. An algorithmic method is implemented as a structure with regard to the evolution of genetic characters. All told, we implement genetic programming as a platform to search for consensus of design.

### 2. OVERVIEW

Genetic programming is a heuristic search technique; it is also a common application of artificial intelligence, where search is the framework for problem solving with computers. A design solution based on such concept potentially explores the enumerative space of generative design. Efficient search depends on the evolutionary procedure on the dynamic procedure of designers and algorithmic implementation. The following section will clarify the relationship and interaction between genetic programming and generative design.
2.1 Genetic Programming

Genetic programming inspires problem solving, but this also implies the limitation of its applicability. The strengthening in computing power, enhancement in applications, and the ability to collect tremendous amount of scientific data all require a computational framework, such as genetic programming. This is why in this paper we intend to apply computer to derive massive generative solutions.

There are two key issues in the genetic programming. 1) selection of a population for alternative solutions; 2) how to generate and evaluate individuals of fitness.

A stochastic selection method chooses better solutions from the population that fetch stochastic variations to produce new alternatives. In this way, the design procedure is responsible for guiding a parallel search throughout the design space. However, the evaluation of each population member becomes increasingly expensive and ineffective. The whole processes of evaluation depend on the direct encoding of criteria in the computational environment. Beside, the amount of data produced by a population based search over an enormous design space makes the prediction and analysis difficult.

With the ability to generate and evaluate a possible solution, a search strategy must be defined. Search methods repeatedly generate solutions, evaluate them and generate more by computation mechanism. Genetic programming uses representation of previously generated solutions when a solution meets a particular criterion, which depends on the design quality threshold or designers’ consensus. In addition, alternative solution is evaluated by means of a fitness function that allows it to be compared with previously evaluated solutions. The programming procedures with evolutionary steps are the design objective should achieve that is also the goal of evolutionary design.

2.2 Generative Design

Generative systems offer a methodology that produces design space via dynamics and their outcomes. As such, generative systems offer an information processing theory to problem-solving design. Based on the information processing theory, some scholars define design process as a cyclical process from specification, generation and evaluation. (Mitchell, 1992) For designers, generative design would involve reconsidering the permutation of potential types. Conceptualisation that shifts from the primacy of objects to interacting components, and from systems to processes, generates new artefacts with special characteristics.
Encapsulated in a navigating structure of paths and landmarks, design space offers an exposition for actions and intentions associated with design (Chien and Flemming, 1996). To explore the design space we need appropriate representations and procedures to fulfil design task. Algorithmic method is thus implemented as the structure for design generation. A formal representation presents design computing in term of type, symbol, colour and status. In all, generative design provides a possibility that walks through the generation of solutions.

The generative methodology offers an unconventional way for conceptualising and operating in design process. Research in generative design is closely related to the general concept of synthesis, mostly presented in the form of nature and/or discreet. Natural selection develops a specific mechanism for generalised synthesis, using the physical apparatus of DNA, protein synthesis, and biochemistry. The discreet of generative design diversifies into numbers and scalable shape, which demonstrates the capability of generative design to overcome design problems, and to construct diverse forms from relatively simple units.

### 2.3 Designing

Designing is a reflective conversation that involves the recursive processes of seeing, moving and seeing (Schön and Wiggins, 1992). Choices, alternatives and versions emerge from the interaction between designing (acting) and discovering (reflecting). Exploration encompasses the formulation of requirements and the generation of solutions based on these requirements. Exploration in the design space is an integral part of the process of solution reformulation, and solution reformulation requires efficient searching and selection.

Exploration rationale (Smithers, 2002) and design selections are critical supporters for exploration. Designers must have the capability to exploit from numerous options through navigating and recombining the paths of exploration. Design representation must provide a unified model for representing potential solutions, which designers take into consideration and reformulate. Designers should make decision to select alternative problems and solutions. In addition, representation captures the characteristics of problems (solutions) as well as selections that designers made during exploration. The above issues require designers’ reconsideration in order to address representations in design process.

- Characteristics of problems – Problems must correspond to designers’ view to problem formulation. As such, the characteristics of problems must correspond to the initial, intermediate and final stages of designs. In essence, an evolutionary procedure must capture
the frame of problems and connect the problems with potential solutions.

- Selections – During the exploration process, problems and requirements of design create a large design space that requires a criterion to decide whether solutions fit or not. During a designing process, a problem formulation may have no solutions, a finite number of solutions, or enumerative solutions. The intentional selections that designers made in the reformulation of problems and commitments to solutions during exploration must be recorded in an evolutionary design procedure.

3. METHODOLOGY

Genetic programming is an evolutionary algorithm that applies either a procedural or functional representation. This section describes design representation and the specific algorithm components used in the canonical version of algorithm. The fundamental of genetic programming are initially presented, followed by a discussion of algorithm and description of two evolutionary procedures. Issues with regard to design research and metaphors of genetic programming applications will also be discussed.

3.1 Evolutionary Algorithms

Darwinian evolution applies the principles of competition, inheritance, and variation within a population. These concepts are often used to define iterative improvement in computer programming. These methods, evolutionary algorithms, use a population of solutions and genetic operators to carry out selection and evaluation. The evolutionary algorithm employs the following items: (Gustafson, 2004)

- A population of candidate solutions called individuals,
- A fitness function that evaluates and assigns each individual a score, or fitness value,
- Transformation operators that produce offspring individuals from parent individuals, implementing the concept of inheritance through stochastic variation, and
- A stochastic selection method for selecting individuals with better fitness to produce offspring.

With evolutionary procedure, we adopt a similar search strategy as a genetic algorithm, uses a program representation and special operators. The
representation of evolutionary design process makes genetic programming unique. The basic algorithm is refined by design process and shows as follows:

- Initialise a population of solutions
- Assign fitness value to each population member
- While the convergence is not met
- Produce new individuals using operators and the existing population
- Place new individuals into the population
- Assign new fitness value to each population member, and test for the convergence satisfied
- Return the best fitness found

### 3.2 Design Model with Genetic Programming

Genetic programming has become a popular search technique since early 1990s thanks to the work by Koza (Koza, 1992). Nowadays, genetic programming is applied mostly related to adaptive system and optimization, where representation of programs is used in conjunction with hybrid crossover to evolve a multiplication function. This research employed partial weighting assignment as functional activity and convergence of selections. In addition, the design with genetic programming is not traditionally considered in canonical genetic programming.

Theories of evolutionary algorithms use abstract representations of the solution space, called schemata, to describe various components and behaviours of algorithm. Holland’s (Holland, 1975) notion of schema for genetic algorithms was extended by Koza (Koza, 1992) to include syntax trees. Syntax trees are the most popular representation in genetic programming, and these schemas were intended to represent trees that refine data structure as computer programming language. In this case, the code of computer programming language allows the geometry representation to become schema in order to represent the design figures and characters.

The schema of design model is developed into two evolutionary processes of design operations which include natural selections and the evolutionary mechanism. Natural selections provide the tournament for the distribution of designers’ weighting that calculates fitness of each population. The parallel process is evolutionary mechanism that interact fitness (selection) and individuals to evolve the population. These processes present actual execution of the algorithmic method with all its characteristics and degrees of freedom (figure 1). Thus, the process will end once it meets the design convergence. This evolutionary process of design is also coherent
with the rational design model that is an iterative cycle of design analysis, synthesis, and evaluation (Asimow, 1962).

4. EXPERIMENTAL DESIGN

4.1 Experiment Installation

We start our experiment with a studio assignment “windmill design” to seek for formal solutions. The windmill evolves its possible forms in an evolutionary design process. We implement genetic programming and derive 15 generations for observation. The gene types of the windmill are defined as follows (see figure 2):

- Gene type 1: The legs of windmill could range from 2 to 8.
- Gene type 2: The leaf shapes of windmill could be either square, rectangle, circle or triangle.

*Figure 1. Generative design model.*
Gene type 3: The relative of each leg could be connected by a circle.
Gene type 4: The foundation of windmill may change the width of the windmill.
Colour: The colours in all segments of the windmill are changeable.

The design team comprises two characters to test the tournament selection. They perceive thinking of architects and structural engineers. They also employ the knowledge of domain as rules of selection. The entire cognitive process is the interaction between designers’ selection and fitness individuals in the evolutionary procedure. The design team adopts designers’ view and knowledge in each tournament with weightings. Selected individuals under an evolutionary mechanism rely on tournament selections for survival decision – the weak die, while the strong survive and reproduce. These procedures are implemented via natural selection associated with their fitness, crossover, and mutation (Goldberg, 1989). In the end of generation, potential solutions arrive that correspond to the design team and genetic programming.

4.2 Experimental Procedures

The experimental procedures of design are employed in order to examine the efficiency of design selections and that of searching between designers and computer supporting system. We implemented the schema of genetic programming based on previous study on genetic programming. With computational operators and structure, genetic programming includes mutation, reproduction, selection/fitness, and other representations in evolutionary procedure (Holland, 1975). We express in Figure 3 (Procedure of Genetic Programming) for implementing programming (Michalewicz, 1992) in evolutionary mechanism (Figure 4).
Generative Design in an Evolutionary Procedure

Procedure of Genetic Programming

Begin

\[ T = 0 \]

Initialize \( p(t) \)

Evaluate \( p(t) \) //\( p(t) = w1*p1(t)+w2*pa(t)+... \)

While (not termination-condition) do

Begin

\[ T = t+1 \]

Select-parents from \( p(t-1) \)

Form \( p(t) \): reproduce the parents //mutation

Evaluate \( p(t) \) //\( p(t) = w1*p1(t) + w2*pa(t)+... \)

End

End

End

Figure 3. Procedure of genetic programming.

\[ T = 0 \]
\[ \text{Evaluate } p(t) \text{ //} p(t) = w1*p1(t)+w2*pa(t)+... \]
\[ T = t+1 \]
\[ \text{Evaluate } p(t) \text{ //} p(t) = w1*p1(t) + w2*pa(t)+... \]

Figure 4. Implementation of the evolutionary mechanism.

To understand the experimental procedure, we developed two procedural settings – tournament procedure (Figure 5) and independent procedure (Figure 6). Both intermediary activities and final queries were recorded in these two settings in order to analyse the evolutionary procedure. Observation and discussion of the two evolutionary designs are presented in the following section.
Figure 5. Evolutionary procedure (Tournament procedure).

Figure 6. Evolutionary procedure (Independent procedure).
4.3 Results and Discussions

The two experiments above employ the evolutionary procedure to seek for possible outcomes, which however demonstrate different characters and individuals. The tournament procedure truly reflects the fitness/selection of designers as well as correspondence of the evolutionary mechanism. On the other hand, the independent procedure intensifies potential results whereas falls short of integration in the same process of evolution.

Fitness and weighting of selection decide the survivability and continuity of population. Designers thus are required to exchange their intuitions and/or concepts – to some extent this looks like a cooperative design process. For example (Figure 7), the initial selection suggests architects and engineers adopt fairly different strategies – architects intuitively select five or more legs and circle-like wing. On the contrary, engineers chose a three-leg windmill while dislike the one with more than five legs. The counteraction from the above tournament impacts individuals and gene pool. However, counteracting selections cannot produce high rates of survival, although the selection is normally processed. On the other hand, an independent procedure would lead selected individuals to become more homogenous (Figure 8).
The above design processes generated numerous conceptual options, but resulted in distinguishing outcomes at the subsequent design stage. Although inspired by the evolutionary mechanism as exhibited in Figure 9, students eventually produced totally different artefacts.

Figure 9. Samples of students’ work.

5. CONCLUSION

Designing can be displayed as a dynamic and formal operation of an evolutionary procedure. This study employs the computer as an interface for genetic programming to generate a canonical population for selection. As for generative design with genetic programming, concepts of evolutionary selection are developed that explain different knowledge behaviours. These behaviours are categorised into two evolutionary procedures. First, the evolution of populations towards a stable state corresponding to the designers’ consensus. Second, once such a stable state is reached, the fitness solutions emerge and terminate the programming procedure.

An ideal design process is to reflect designers’ consensus while evolving with principles and concepts of design. Still, this could leave the design space incorrectly defined. Generative design or hierarchic organisation may help solve this dilemma.

The genetic programming as an evolutionary design process can be transformed into the analysis of modified genetic algorithms. As such, genetic programming in generative design that reconsiders certain changes to the selection operator may produce the fittest population in the evolutionary cycle. Mutation-control selection schemes, including the selection with a divergent election operator, ensure that at least the first breadth individual of a population will become a member of the next generation’s population. On the other hand, strait-forward selection schemes reveal not enough breadth samples to become the fitness selections. This also explains that some populations have no chance to be transitory populations. In this case, we
Generative Design in an Evolutionary Procedure

need evolutionary strategies to dynamically adjust the mutation rate in order to reach asymptotic stability in every single evolutionary procedure.

We propose in this study that a design research in the dynamics of tournament selection will develop solutions for evolutionary procedure design. With these instruments, we observe interesting design processes that could lead to numerous potential works.

6. ACKNOWLEDGMENTS

We are grateful to insightful discussions with Prof. Sheng-Fen Chien and the research group CODE, Department of Architecture, National Taiwan University of Science and Technology.

7. REFERENCES