GENERATIVE EVOLUTIONARY DESIGN: A FRAMEWORK FOR GENERATING AND EVOLVING THREE-DIMENSIONAL BUILDING MODELS

Patrick Janssen
patrick@janssen.name
School of Design, Hong Kong Polytechnic University

John Frazer
Digital Practice Ecosystem, Gehry Technologies

Ming-Xi Tang
School of Design, Hong Kong Polytechnic University

ABSTRACT

This paper describes a comprehensive framework for generative evolutionary design. The key problem that is identified is generating alternative designs that vary in a controlled manner. Within the proposed framework, the design process is split into two phases: in the first phase, the design team develops and encodes the essential and identifiable character of the designs to be generated and evolved; in the second phase, the design team uses an evolutionary system to generate and evolve designs that incorporate this character. This approach allows design variability to be carefully controlled. In order to verify the feasibility of the proposed framework, a generative process capable of generating controlled variability is implemented and demonstrated.

Keywords: evolutionary design, generative design, generative system, genetic algorithm, evolutionary algorithm

INTRODUCTION

Evolutionary design systems are loosely based on the neo-Darwinian model of evolution through natural selection. A population of individuals is maintained and an iterative process applies a number of *evolution steps* that create, transform, and delete individuals in the population. Each individual has a genotype representation and a phenotype representation. The genotype representation encodes information that can be used to create a model of the design, while the phenotype representation is the actual design model. The individuals in the population are rated for their effectiveness, and on the basis of these evaluations, new individuals are created using 'genetic operators' such as crossover and mutation. The process is continued through a number of generations so as to ensure that the population as a whole evolves and adapts.

A wide variety of evolutionary algorithms exist, with the four main types being genetic algorithms (Holland1975), evolution strategies (Rechenberg1973), evolutionary programming (Fogel1963), and genetic programming (Koza1992).

The evolutionary process typically has a centralised control structure consisting of a main loop that repeatedly invokes the evolution steps. Individuals are processed in a synchronous manner, whereby the evolutionary process stops and waits for the processing of all individuals by one evolution step to be completed before proceeding onto the next evolution step.

The evolution steps include a *reproduction step* that creates new genotypes by combining genetic material from randomly selected parents; a *developmental step* that creates design models (phenotypes) from encoded genotypes; and an *evaluation step* that evaluates the performance of the design with respect to one or more objectives. In addition, *selection* and *survival* steps allow the genetic material of the best individuals to pass from one generation to the next.

Types of evolutionary design

Two types of evolutionary design may be broadly identified: *parametric* evolutionary design and *generative* evolutionary design.

Parametric evolutionary design is usually used late in the design process and focuses on the optimisation of design solutions to well-defined design problems. An existing design is defined and parts that require improvement are parameterised. The evolutionary system evolves the parameter values. Such systems are generally described as convergent search systems that search the parameter space for an optimal or satisficing set of parameter values. Two recent examples of parametric evolutionary design systems are Rasheed (1998); Caldas (2001).

Generative evolutionary design, on the other hand, may be used early on in the design process and focuses on the discovery of inspiring or challenging design alternatives for ill-defined design tasks. A generative process is created that uses information in the genotype to generate alternative design models. The evolutionary system will tend to evolve a divergent set of alternative designs, with convergence on a single design often being undesirable or even impossible. Such systems are sometimes described as divergent systems or exploration systems. Examples of generative evolutionary design systems include Frazer and Connor (1979), Graham et al. (1993); Frazer (1995b); Bentley (1996); Rosenman (1996a); Coates et al. (1999); Funes and Pollack (1999); Sun (2001).

With regard to the types of designs produced, the main difference between these two approaches relates to the variability of designs. With the parametric approach, design variability is low. Since the designs are all based on the same parametric model, the designs will all have the same overall organisation and configuration. With the generative approach, the variability in designs can potentially be much greater.

Of these two approaches, the parametric approach is the more common and well developed. The generative approach, although more complex, can be much more powerful.

ANALYSIS OF PROBLEM

For generative evolutionary design, a distinction can be made between systems whose main purpose is to *inspire,* versus systems whose main purpose is to *challenge*. Many of the existing systems evolve forms that may be inspiring to designers. These types of systems may produce highly abstract forms that trigger new possibilities in the minds of the designers. However, if the forms do not incorporate a relatively explicit and comprehensive description of the design, then the designers will be required to interpret and read meaning into the forms. The designers thereby perceive a design within the form.

In order to challenge, *designs* rather than *forms* must be evolved. Such designs must fulfil four key criteria. The designs must be:

- *Complex*: The level of complexity within the designs must be commensurate with the complexity of the entities being designed. Designs must therefore consist of three-dimensional models consisting of a realistic number and variety of related elements.
- *Intelligible*: The forms must be directly intelligible as designs by both people and by other software systems. Important characteristics of the forms must therefore be explicitly represented, thereby allowing for unambiguous understanding.
- *Unpredictable*: The forms must vary from one another in significant ways. This must include variation in the organised and configured of the elements that comprise the form.
- *Desirable*: The forms must embody certain qualities that are seen to be desirable by the designers using the system. These qualities may be either qualitative or quantitative.

The first three criteria – for designs to be complex, intelligible and unpredictable – depend primarily on how the generative process is implemented. Such a process must use a set of generative rules to generate individual designs.

The fourth criterion – for designs to be desirable – typically depends on the implementation of the evaluation process, which may include both an automated and manual component. The automated component will invoke simulate and analysis software in order to evaluate quantitative qualities. The manual component will rely on human judgement to evaluate qualitative qualities. As well as the evaluation process, it may also be possible to hard-code such qualities within the generative process.

The variability problem

In order for a generative evolutionary system to challenge the designers, the generative process must generate designs that are complex, intelligible and unpredictable. This brings to light a fundamental problem: *given a certain level of complexity, it is very difficult to create a generative process that generates designs that are both intelligible and unpredictable*. If unpredictability is required, then the generative process tends to become under-restricted, resulting in forms that are chaotic and unintelligible. If intelligibility is required, then the generative process tends to become over-restricted, resulting in designs that are all very similar and predictable. This conflict is referred to as the *variability problem*.

The generative process consists of a set of generative rules that are applied to some starting condition. The same set of rules will be used to generate each design. The genotypes encode variations in the starting condition and in how the rules are applied.

In order to overcome the variability problem, a generative process is required where the variability of designs is carefully controlled in order to ensure that designs are both intelligible and unpredictable. This is referred to as *controlled variability*.

Developing such rules requires the identification of a set of characteristics common to all the designs to be generated. Rules can then be created based on these shared characteristics. Such characteristics may relate to issues of aesthetics, space, structure, materials, construction, and so forth.

In the domain of architecture, it is difficult to identify any significant characteristics that are shared by all possible designs (including future designs as well as existing designs). Buildings simply vary too much for this to be possible. Even for existing designs, attempting to pin-down shared characteristics is problematic. As a result, some sub-set of designs needs to be considered that includes designs that are similar in some way, but excludes others that are not similar. Since the included designs will share certain characteristics, they may be described as a *family* of designs. A decision must be made as to how to define this family.

One approach to defining the family of designs is to focus on conventional designs. This involves defining a sub-set of designs based on typical characteristics found in existing designs. However, such an approach is problematic since it fundamentally limits the creativity of the designer. The aim of developing a generative evolutionary design system is to enhance the creative process by allowing designers to explore populations of alternative designs. If the designs being generated are required to be conventional, then the creative process will be hindered rather than enhanced.

An alternative approach is to focus on the oeuvre or body of work of one design team. This involves defining a sub-set of designs based on a set of characteristics developed by a particular design team. This is the approach pursued in this research.

PROPOSED FRAMEWORK

A framework has been developed that allows designers to incorporate and express their own design ideas. The core concept within this framework is the notion of a design entity that captures the essential and identifiable character of a varied family of designs by one designer or design team. This conceptualisation is defined as a *design schema*. It encompasses those characteristics common to all members of the family, possibly including issues of aesthetics, space, structure, materials and construction. Although members of the family of designs share these characteristics, they may differ considerably from one another in overall organisation and configuration. Design schemas are seen as formative design generators; their intention is synthetic rather than analytic.

When a design schema is codified in a form that can be used by a generative evolutionary system, it provides a way of overcoming the variability problem. The encoded schema allows complex designs to be generated that are both intelligible and predictable. This approach is based on the work of Frazer and Connor (1979); Frazer (1995a); Sun (2001).

The schema framework consists of two parts: a *design method* and an *evolutionary system*. The design method broadly defines a set of tasks to be carried out by the designer team. The evolutionary system is a software system used by the designer team for generating and evolving alternative designs. Both the design method and the evolutionary system are introduced below. For a more detailed discussion, see Janssen (2004).

Design method

The design method breaks the design process down into two sequential phases. Figure 1 shows the overall structure of the design method. In the first phase, a design schema is developed that may be used to evolve designs for a range of different projects. In the second phase, the schema developed in the first phase is applied to a specific project and a detailed design proposal is developed.

Each project defines a specific environment for a design, encompassing both design constraints and design context. Examples of design constraints may include the budget, the number of spaces, floor areas, performance targets and so forth. The design context may include site dimensions, site orientation, neighbouring structure, seasonal weather variations, and so forth. The schema developed in the first phase is not specific to one design environment. Instead, the design team develops it with a certain *type* of environment in mind, referred to as the *niche environment*. This niche environment encompasses a range of possible constraints and a range of possible contexts. The schema can be used in any project whose design environment falls in the niche environment for which the schema was designed. The first phase may therefore be viewed as a *generalization* process, and the second phase as a *specialization* process.

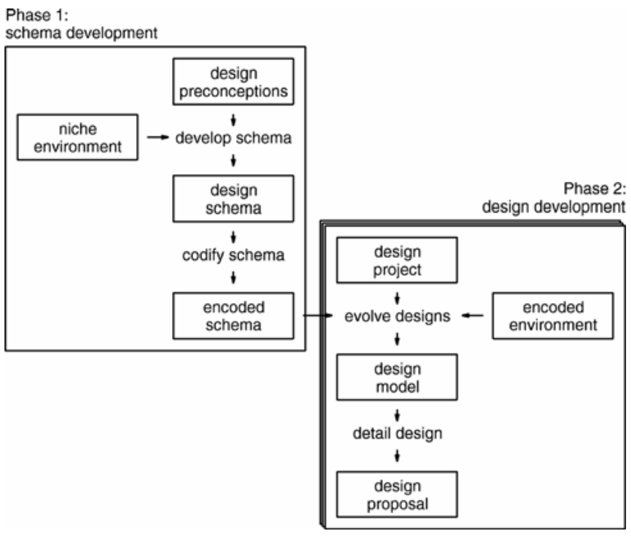

Figure 1: The schema-based generative evolutionary design method.

The encoded schema will consist of a set of small programs, or *routines*, that encapsulate the rules and representations for all of the evolution steps within the evolutionary system. In order to generate and evolve alternative designs, the evolutionary system requires the encoded schema and the encoded environment. Both need to be encoded in a format that is compatible with the system and with one another.

Evolutionary system

The evolutionary system maintains a single population that is manipulated by four evolution steps: *reproduction*, *development*, *evaluation* and *survival*. In addition there is also a *visualization* step that allows users to select and visualise designs in the population, and *initialisation* and *termination* steps for starting and stopping the evolutionary process.

One of the key requirements for the evolutionary system is *customisability*. The design team must be able to encode and input both the design schema and the design environment within the system. In addition, the design team is also likely to want to make use of existing software applications for modelling, visualising and evaluating design models.

The evolutionary system is therefore broken down into two parts: a *generic core* and a set of *specialised components*. The generic core defines the main structure of the evolutionary system and can be used unmodified by any design team, on any project. This core consists of underlying program modules that communicate and interact with one another. However, in order to be functional, these modules must be linked to the specialised components. The specialised components are completely customisable and must be defined by the design team. Three types of specialised components exist: *routines, data-files,* and *applications.*

- Routines encapsulate the rules and representations used by the evolutionary system. The design team must create a set of such routines that together constitute the encoded schema.
- Data-files encapsulate information about the design environment. The design team must create these data-files that together constitute the encoded environment.
- Applications are existing software applications whose functionality the design team may require, in particular for modelling, visualising and evaluating design models.

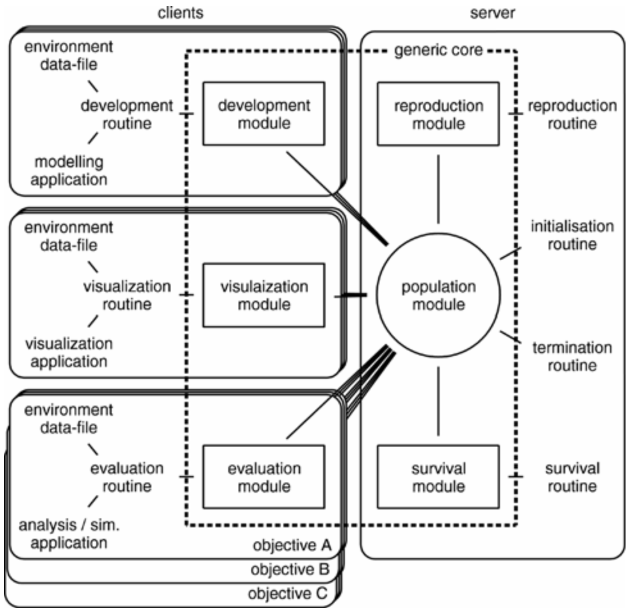

Figure 2: Schema-based generative evolutionary design system.

As well as customisability, a second key requirement is *scalability*. The system should allow for the evolution of large complex designs without performance being adversely affected. This is especially pertinent for the evaluation step, since the simulation or analysis of the performance of a building design can be time consuming.

In order to support scalability, a parallel architecture has been developed using a standard client-server model in a networked computing environment. The architecture of the system is shown in figure 2. The server manages the population of designs and performs the reproduction and survival steps, while multiple client computers perform the most time consuming developmental and evaluation steps.

The architecture uses an *asynchronous* evolutionary process in combination with a *decentralised* control structure. The four evolution steps are therefore not centrally controlled and act independently from one another. Each step extracts a small number of individuals from the population, processes these individuals, and either inserts the resulting individuals back into the population or – in the case of the survival step – deletes a number of individuals in the population.

The asynchronous evolutionary process reduces the execution time and is highly effective in situations where the development and evaluation steps are costly. The decentralised control allows client computers to be easily be added and removed from the evolutionary process and allows the system to cope gracefully with failure of one or more client systems.

CONTROLLED VARIABILITY

The evolutionary system is currently under development. In order to verify the feasibility of the schema-based framework, the process of encoding a design schema has been demonstrated. The demonstration consists of three parts:

- An example design schema has been created for a family of multi-story buildings. The overall building form, the organisation of spaces, and the treatments of facades may all vary significantly. Some additional complications such as sloping walls have been included, but not curved walls.
- A generative process has been created for generating design models in the example schema. This process consists of a series of transformations that gradually transform a three-dimensional orthogonal grid structure into a design for a building.
- A developmental routine, an initialisation routine and a visualization routine have been implemented for the example schema. These routines have been used to generate and visualise a variety of design models. The designs that are generated are complex, intelligible, and unpredictable. Controlled variability has therefore been achieved.

Generative process

The generative process consists of a sequence of eight generative transformations that gradually change an orthogonal grid into a 3-dimensional building model. Figure 3 shows (diagrammatically in two-dimensions) the eight generative transformations: positioning of the grid in the site, translation of the grid-faces, inclination of outer grid-faces, insertion of the staircase, creation of spaces, selection of outside spaces, insertion of doors, and insertion of windows.

Most transformations require a set of parameters encoded within the genotype. The genotype consists of a fixed length string of parameters, with each parameter being encoded as real values in the range 0.0 to 1.0. The encoded value may be mapped to a value within a different continuous or discrete range as required. Some transformations may also require certain parameters or data encoded in the environment data-file.

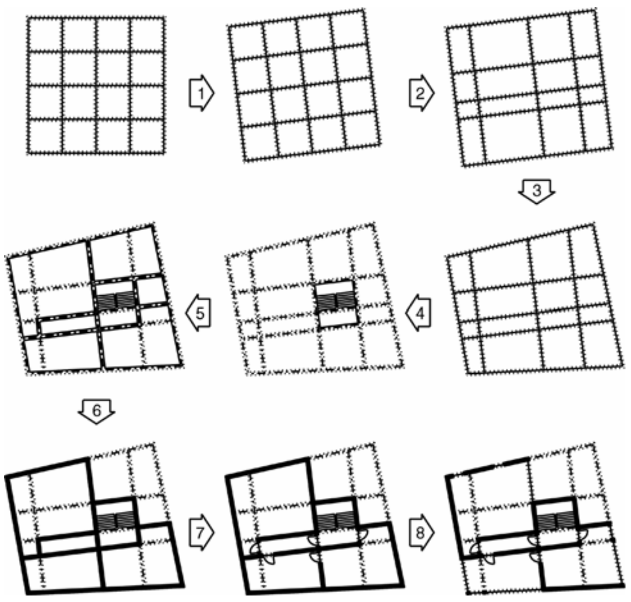

Figure 3: The generative process

Implementation

In order to verify the character and variability of the designs that would be produced by the generative process described above, the initialisation, developmental and visualization routines were implemented:

- The initialisation routine was used to generate a population of individuals with randomly generated genotypes, but with no phenotypes or evaluation scores. This routine calculates the length of the required genotype, and creates a random value for each parameter.
- The developmental routine was used to create phenotypes for each individual. The generative process used by this routine has already been described above.
- A visualization routine has been created that uses Ecotect by developed by Square One Research to visualise the design models that are generated. This routine extracts the phenotype from each individual, translates the phenotype representation to the model representation used by Ecotect, and then allows the design to be visualised using the Ecotect interface.

The initialisation routine was used to generate a population of genotypes, the developmental routine was then used to generate a population of design models, and finally the visualization routine was use to view these models. Figure 4 shows a selection of models generated.

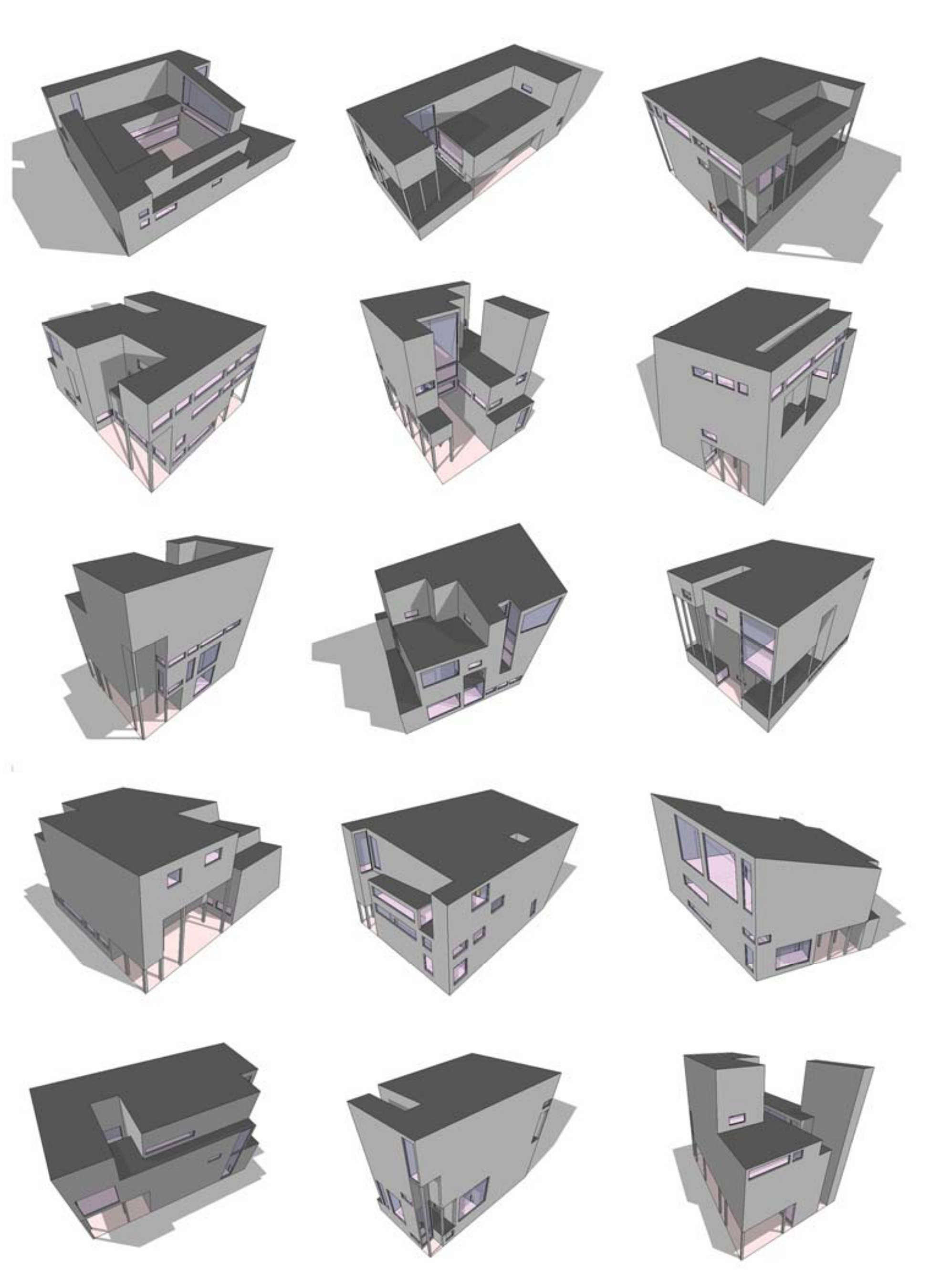

Figure 4: A set of generated (but not evolved) designs.

CONCLUSIONS

The demonstration has shown that it is possible to create a generative process that generates complex three-dimensional models of building designs that are both intelligible and unpredictable. Controlled variability has therefore been achieved.

Since the designs have not yet been evolved, they do not yet incorporate desirable qualities. The next stage of the research will focus on developing the complete evolutionary system. This will allow designs to evolve and adapt in response to the environment and the evaluation criteria, thereby resulting in qualities that are seen to be desirable.

ACKNOWLEDGEMENTS

Our research project is supported by a UGC PhD project grant from the Hong Kong Polytechnic University.

REFERENCES

Bentley, P. J. (1996). “Generic Evolutionary Design of Solid Objects using a Genetic Algorithm.” Doctoral dissertation, Division of Computing and Control Systems, Department of Engineering, University of Huddersfield.

Caldas, L. (2001). “An Evolution-Based Generative Design System: Using Adaptation to Shape Architectural Form.” Doctoral dissertation, Massachusetts Institute of Technology.

Coates, P., Broughton, T., and Jackson, H. (1999). “Exploring three-dimensional design worlds using Lindenmayer Systems and Genetic Programming.” In Bentley, pages 323–341.

Fogel, D. B. (1995). “Evolutionary computation: Towards a new philosophy of machine intelligence.” IEEE Press.

Frazer, J. H. (1995a). “An Evolutionary Architecture.” AA Publications, London, UK.

Frazer, J. H. (1995b). “The interactivator.” AA Files, 72–73.

Frazer, J. H. and Connor, J. (1979). “A conceptual seeding technique for architectural design.” In Proceedings of International Conference on the Application of Computers in Architectural Design and Urban Planning (PArC79), pages 425–434, Berlin. AMK.

Funes, P. and Pollack, J. (1999). “Computer evolution of buildable objects.” In Bentley, P. J., editor, Evolutionary Design by Computers. Morgan Kaufmann Publishers, San Francisco, CA., pages 387–403.

Graham, P. C., Frazer, J. H., and Hull, M. C. (1993). “The application of genetic algorithms to design problems with ill-defined or conflicting criteria.” In Glanville, R. and de Zeeuw, G., editors, Proceedings of Conference on Values and, (In) Variants, pages 61–75.

Holland, J. H. (1975). “Adaptation in Natural and Artificial Systems.” University of Michigan Press, Ann Arbor.

Janssen, P. H. T. (2004). “A design method and a computational architecture for generating and evolving building designs.” Doctoral dissertation, School of Design Hong Kong Polytechnic University (submitted October 2004).

Koza, J. R. (1992). “Genetic Programming: On the Programming of Computers by Means of Natural Selection.” MIT Press, Cambridge, MA.

Rasheed, K. M. (1998). “GADO: A Genetic Algorithm for Continuous Design Optimization.” Doctoral dissertation, Department of Computer Science, Rutgers University, New Brunswick, NJ. Technical Report DCS-TR-352.

Rechenberg, I. (1973). “Evolutionstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution.” Frommann-Holzboog Verlag, Stuttgart, Germany.

Rosenman, M. A. (1996). “An exploration into evolutionary models for non-routine design.” In AID’96 Workshop on Evolutionary Systems in Design, pages 33–38.

Sun, J. (2001). “Application of Genetic Algorithms to Generative Product Design Support Systems.” Doctoral dissertation, School of Design, Hong Kong Polytechnic University.