A COMPUTATIONAL SYSTEM FOR GENERATING AND EVOLVING BUILDING DESIGNS

PATRICK H. T. JANSSEN, JOHN H. FRAZER
School of Design, Hong Kong Polytechnic University, Hong Kong.
patrick@janssen.name
Digital Practice Ecosystem, Gehry Technologies
john.frazer@gehrytechnologies.com

AND

MING-XI TANG
School of Design, Hong Kong Polytechnic University, Hong Kong.
sdtang@polyu.edu.hk

Abstract. This paper describes a generative evolutionary design system that aims to fulfil two key requirements: *customisability* and *scalability*. Customisability is required in order to allow the design team to incorporate personalised and idiosyncratic rules and representations. Scalability is required in order to allow large complex designs to be generated and evolved without performance being adversely affected. In order to fulfil these requirements, a computational architecture has been developed that differs significantly from existing evolutionary systems. In order to verify the feasibility of the this architecture, the generative process capable of creating three-dimensional building models has been implemented and demonstrated.

1. 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.

Two types of evolutionary design may be broadly identified: *parametric* evolutionary design and *generative* evolutionary design. Parametric evolutionary design is the more common approach. A design is predefined and parts that require improvement are parameterised. The evolutionary system is then used to evolve these parameters.

The generative approach, although more complex, can also be much more powerful. This approach 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; Rosenman (1996); Coates et al. (1999); Funes and Pollack (1999); Sun (2001).

1.1. GENERATIVE EVOLUTIONARY DESIGN FRAMEWORK

The generative evolutionary design approach has had limited success in evolving the overall configuration and organisation of complex designs. The primary problem identified is the generation of designs that incorporate an appropriate level of *variability*. If design variability is low, then the performance of the evolutionary system will be high but the types of designs will tend to be very predictable. Conversely, if design variability is high, then the types of designs may be unpredictable, but the performance by the evolutionary system will tend to be low.

In order to overcome this problem, a framework has been developed that allows designers to restrict design variability by specifying the character of designs to be evolved. This approach is based on the notion of a design entity that captures the essential and identifiable character of a family of design. This design entity is called a *design schema*. The design team encodes the design schema as a set of rules and representations that can be used by the evolutionary system. The system can then be used to evolve designs that embody the encoded character. This approach is based on the work of Frazer and Connor (1979); Frazer (1995a); and Sun (2001).

A design schema 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.

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 design team. The evolutionary system is a software system used by the design team for generating and evolving alternative designs. This paper will focus on the architecture of this evolutionary system. Janssen et al. (2005) provide an overview of the overall framework. For a detailed description and analysis of the framework, see Janssen (2004).

2. The evolutionary system

Within the schema framework, the evolutionary system must fulfil two key requirements:

- The system should be *customisable*, allowing for the modification and replacement of the rules and representations used by the system. This will allow each design team to incorporate their encoded schema and to input data relating to the design constraints and design context. In addition, the design team should also be able to integrate existing applications that perform useful functions such as modelling, analysis, and simulation.
- The system should be *scalable*, allowing for the evolution of large complex designs without performance being adversely affected. In most cases, the developmental and evaluation steps are likely to be the most computationally demanding.

A computational architecture for an evolutionary system has been developed that fulfils these two requirements. This architecture is shown in Figure 1. A single

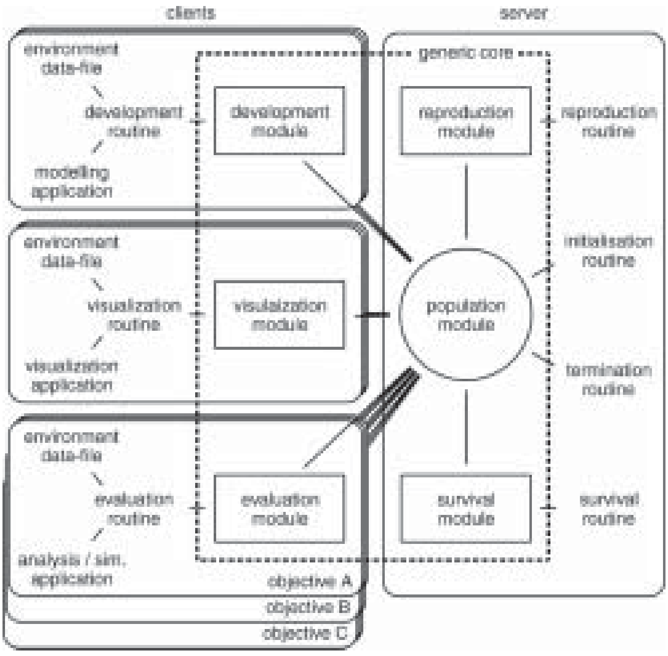

Figure 1: Schema-based generative evolutionary design system.

population is processed by seven steps: an initialisation and a termination step, four evolution steps and a visualization step. The four evolution steps consist of a reproduction step, a development, an evaluation step and a survival step. The visualization step allows design models in the population to be visualized.

A parallel implementation is employed using a standard client-server model in a networked computing environment. The server manages the population of designs and performs the reproduction and survival steps, while multiple client computers perform the developmental and evaluation steps.

2.1. CUSTOMISABILITY

In order to support customisability, the evolutionary system is 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 programme 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 evolution steps. The design team must create a set of such routines that together constitute the encoded schema.
- *Data-files* encapsulate information about the design environment, 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.
- *Applications* are existing software applications whose functionality the design team may require, in particular for modelling, visualising and evaluating design models. For example, CAD, rendering, analysis and simulation applications may be used.

2.2. SCALABILITY

In order to support scalability, the evolutionary system employs an evolutionary process that can be described as using a *decentralised control structure* and an *asynchronous evolution mode*.

- A decentralised control structure is used, whereby the four evolution steps 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.

- An asynchronous evolution mode is used, whereby the evolution steps process individuals or small groups in the population as soon as they become available. Individuals in the population will therefore be in various different states, depending on whether they have been developed or evaluated.

This type of evolutionary process differs markedly from the process employed by most existing evolutionary systems. A wide variety of evolutionary algorithms exist, with the four main types being genetic algorithms (Holland, 1975), evolution strategies (Rechenberg, 1973), evolutionary programming (Fogel, 1995), and genetic programming (Koza, 1992). Such algorithms differ in the rules and representations that they use in order to implement the evolution steps, but the overall evolutionary process is similar for all of them. This process uses a *centralised control structure* and a *synchronised evolution mode*. The centralised control structure consists of a main loop that repeatedly invokes and executes the evolution steps. The synchronous evolution mode consists of a procedure that stops and waits for the processing of all individuals by one evolution step to be completed before proceeding on to the next evolution step.

For scalability, the decentralised asynchronous approach has certain advantages over the centralised synchronous approach. The decentralised control structure 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. The asynchronous evolution mode reduces the execution time and is highly effective in situations where the development and evaluation steps are costly (Rasheed and Davidson, 1999). The evolutionary process does not need to wait for the whole population to be developed and evaluated. As soon as an individual has been evaluated, the evolutionary process can start to either discard or incorporate its genetic information.

2.3. INDIVIDUALS IN THE POPULATION

Individuals in the population are processed by both the generic core and the specialised components. The generic core consists of a set of modules that manipulate individuals, while the specialised components include a set of routines that transform individuals. The manipulation of individuals by the generic modules is fixed and cannot be modified by the design team. The individual can therefore incorporate predefined representational structures required by these modules.

The transformation of individuals by routines, on the other hand, depends on how the design team implements these routines. As a result, the representation of the parts of the individual (such as the genotype, phenotype and evaluation scores) cannot be predefined in advance. The way that an individual is represented therefore needs to incorporate an inherent flexibility. This is achieved by breaking this representation into a generic part and a specialized part. Figure 2 shows an example of a partially evaluated individual. The generic part includes two representations: a set of possible flags and a unique ID.

- Flags are boolean values used to store information relating to whether the individual has been developed and evaluated.
- The ID is an integer value that is unique in the history of the evolutionary process.

The specialised part of an individual consists of three placeholders that can contain variable representations: a genotype, a phenotype and a set of evaluation scores.

- The genotype may have any type of representation, such as a binary or real-valued string, or some other more complex type of data-structure.
- The phenotype may also have any type of representation, including standard three-dimensional file formats.
- The evaluations consist of a list of one or more representations, which will usually consist of a set of performance scores.

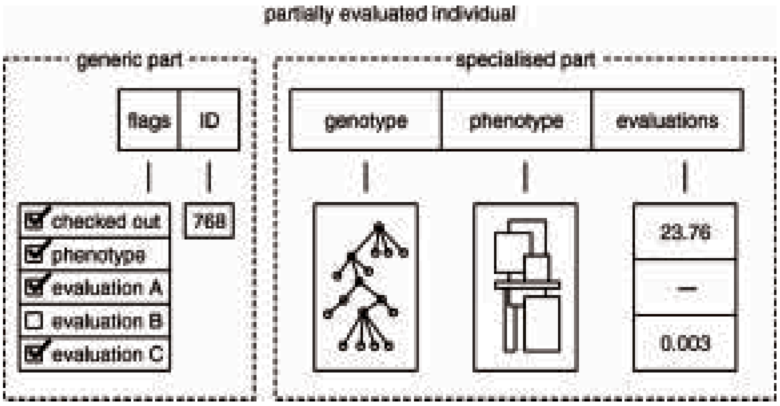

Figure 2: The representation of an individual in the population.

2.4. EVOLUTION STEPS

The population module manages the population. This module is executed on the central server, but does not control the evolution steps. Instead, it passively waits to be contacted by the evolution steps, thereby allowing the evolution steps to act independently from the population module and from one another.

Each of the evolution steps is conceptualised as a modular software components that performs a transformation: each step requires a number of individuals, processes these individuals, and produces some result. The result is usually a new or updated set of individuals, or —in the case of the survival step—one or more individuals to be deleted. The evolutionary process consists of evolution modules contacting the population module to request a set of individuals, processing these individuals, and then contacting the population module once more to send the results back again.

This decentralised control structure results in a population where individuals are in various different *states*: some only have a genotype, others have been processed by the developmental step and have a phenotype, while others also have evaluation scores. It is the responsibility of the population module to ensure that each step is sent to individuals in an appropriate state. The individuals in an appropriate state are referred to as *candidates*. When the population module receives a request from one of the evolution steps, it will first identify all possible candidates in the population and will randomly select the required number of individuals from these candidates.

The four evolution steps are described as follows:

- The reproduction module requests a pool of fully evaluated parents. This module then produces one or more new individuals by executing the reproduction routine.
- Each developmental module requests a single undeveloped individual. This module then creates a phenotype for the individual by executing the developmental routine.
- Each evaluation module requests one individual that has not been evaluated for the objective in question. This module then creates an evaluation score for the individual by executing the evaluation routine.
- The survival module requests a pool of fully evaluated individuals. This module then identifies one or more individuals to be deleted from the population by executing the survival routine.

The pools in the reproduction and survival steps vary dynamically in size, and will be equal to the number of candidate individuals in the population at the time. This ensures that the largest possible pool of individuals will always be used without requiring the evolutionary process to stall.

If multiple objectives are being evaluated, at least one evaluation module per objective must be defined. Each evaluation module will be associated with a different evaluation routine, and each evaluation routine may make use of different analysis or simulation applications. Once an individual has been fully evaluated, it will contain a separate performance score for each objective. When such an individual is processed by the survival routine, some form of scalarization will need to take place in order to make a decision as to whether the individual should be deleted or not. Various scalarization techniques exist, including calculating the weighted average of the performance scores, or used Pareto-optimal ranking methods.

3. Demonstration

The evolutionary system is currently under development. One of the critical aspects of this system is the ability of the design team to develop a generative process that is capable of generating designs that vary in a controlled manner, referred to as *controlled variability*. The process of encoding a design schema has therefore 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 organization 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.

3.1. 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.

3.2. 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, and then translates the phenotype representation to the model representation used by Ecotect.

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.

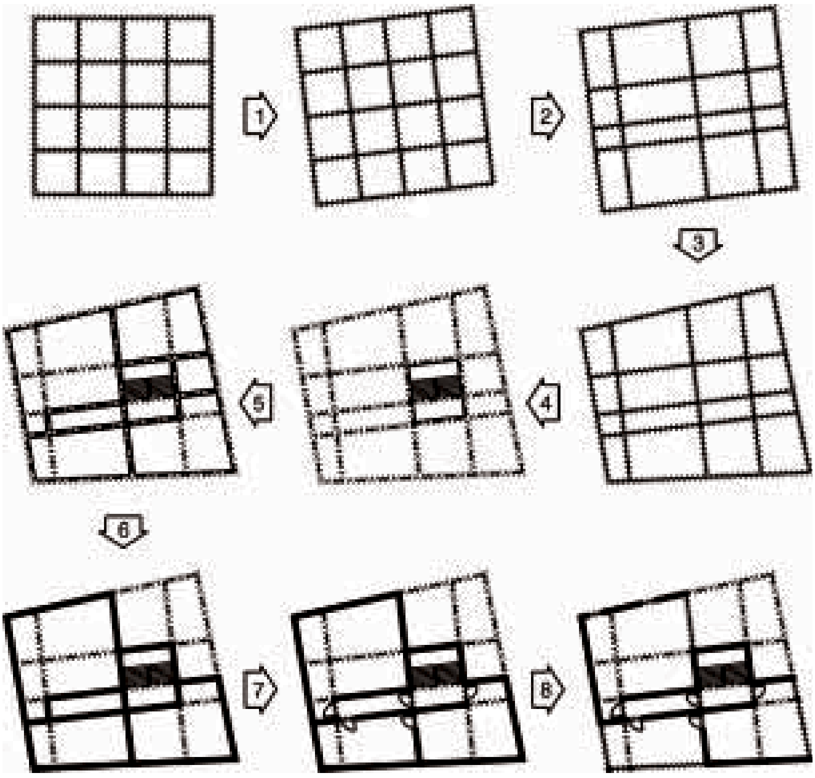

Figure 3: The generative process.

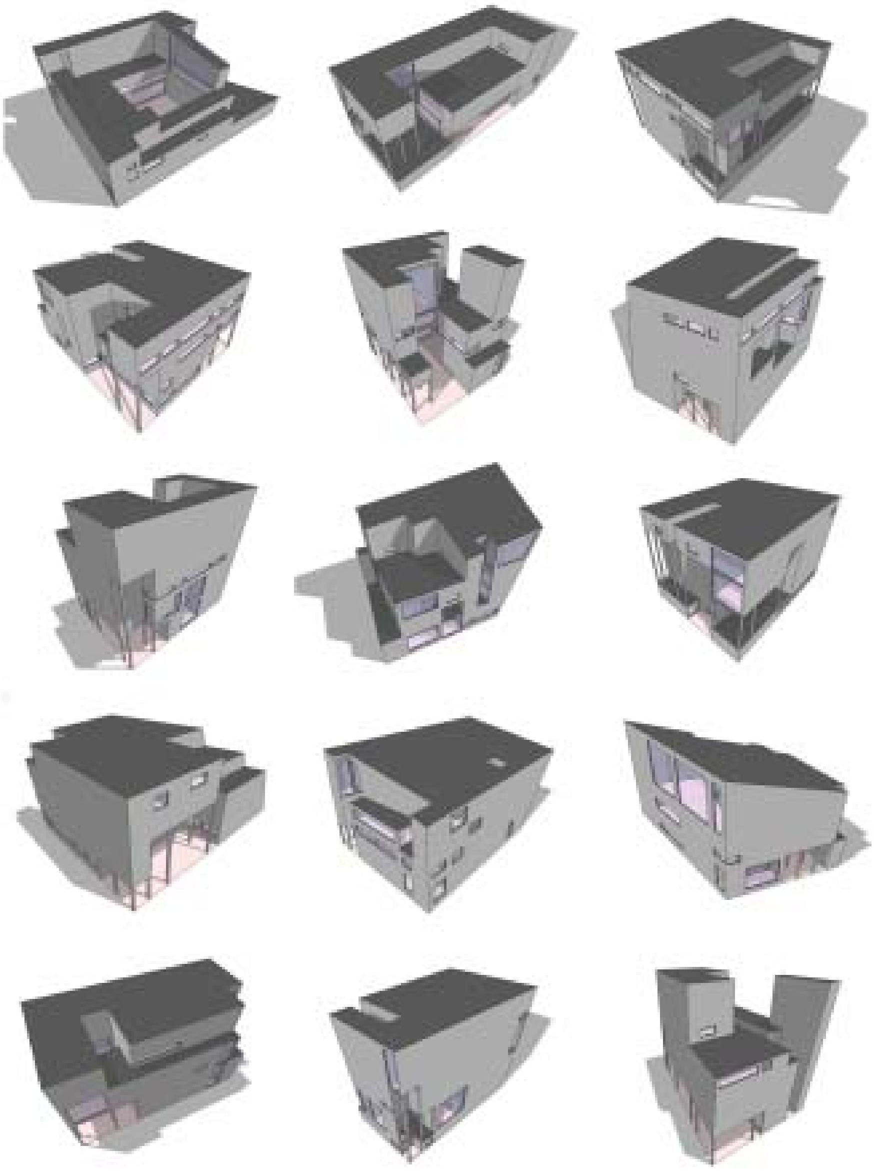

Figure 4: A set of generated (but not evolved) designs.

4. Conclusion

An evolutionary system has been described that fulfils two key requirements for generative evolutionary design: *customisability* and *scalability*. The feasibility of this system is supported by the demonstration, which has shown that it is possible to create a generative process that generates complex three-dimensional models that vary in a controlled manner.

Since the designs have not yet been evolved, they have yet to adapt to the objectives and the environment. 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

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, pp. 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, pp. 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), pp. 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., pp. 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, pp. 61–75.

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

Janssen, P. H. T., Frazer, J. H., and Tang, M-X. 2005, “Generative Evolutionary Design: A Framework For Generating And Evolving Three-Dimensional Building Models.” International Conference on Innovation In Architecture, Engineering And Construction (AEC) (to appear).

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.

Rasheed, K. M. and Davidson, B.D. 1999, “Effect of global parallelism on the behaviour of a steady state genetic algorithm for design optimization.” In proceedings of the Congress on Evolutionary Computation (CEC’99), volume 1, pp. 534-541. IEE Press.

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, pp. 33–38.

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