EVOLUTIONARY DESIGN EXPLORATION SYSTEMS

PATRICK H. T. JANSSEN, JOHN H. FRAZER
School of Design, Hong Kong Polytechnic University

AND

MING-XI TANG

Abstract. This paper proposes a software and hardware architecture for an evolutionary design exploration system for building design. First, the typical architecture for such systems is described. Certain limitations of this typical architecture are identified. The proposed architecture is then described and the advantages of this architecture are highlighted. Finally, the development of systems implementing this architecture is discussed.

1 Introduction

Research into generative programs and evolutionary systems for design has been ongoing for over three decades.

- Generative design programs are used to generate large numbers of design alternatives that differ significantly from one another. These programs define a complex growth process that transforms an encoded seed into a design. By making small modifications to either the transformation process or the seed, alternative designs can be generated.
- Evolutionary design systems are used to evolve alternative design possibilities. These systems are loosely based on the neo-Darwinian model of evolution through natural selection. Such systems consist of a cyclical process whereby whole populations of designs are continuously being manipulated in order to ensure that the population as a whole gradually evolves and adapts.

From the 1950s onwards, a variety of evolutionary algorithms for a number of different purposes were developed. The four main types of evolutionary algorithm are evolution strategies (Rechenberg 1965, 1973, Bäck 1996), evolutionary programming (Fogel 1963, Fogel 1995), genetic algorithms (Holland 1975), and genetic programming (Koza 1990). Of these four algorithms, genetic algorithms became most popular and were primarily

used as a search system for finding optimal solutions to well-defined problems.

Within evolutionary design systems, designs exist in two forms: as an encoded *genotype* and as a fully developed *phenotype*. Each generation, phenotypes are created from the encoded genotypes, and these phenotypes are then evaluated by simulating and analysing their performance. Designs with the highest evaluation are then used to create a new generation of genotypes. This ensures that the traits of the most suitable designs are inherited by later generations.

1.1 WHY EVOLUTION?

The evolutionary process in nature is an extraordinarily impressive design process, producing vast numbers of complex biological designs highly adapted to their environment.

The traditional human design process, on the other hand, is far more limited. The ability of human designers to foresee the future consequences of the numerous and interrelated design decisions made during the design process is limited, particularly when considering design decisions made early on in the design process. The complexity of the design task is typically so great that the designer will need to rely on the 'rule of thumb' and their 'gut feeling' when it comes to making early design decisions (Purcell and Gero 1996, Bentley 1999). The consequences of these decisions will only become clear much later on in the design process when it may already be too late to explore alternative avenues. Even if further time and resources are available, only a small number of alternatives could ever be feasibly explored.

Evolutionary design systems aim to harness some of the awesome power of natural evolution in order to overcome these limitations of the traditional design process. In particular, evolutionary software is inherently parallel in its mode of operation and is therefore able to explore very large numbers of possible alternative design approaches. Furthermore, the evolutionary process will ensure that the population of designs will gradually adapt to the environment within which they are being generated and assessed. The resulting design may therefore become adapted to the environment in a multitude of complex and interrelated ways, as is common with biological designs in the natural world.

1.2 DIVERSITY AND DISPARITY

Any evolutionary design system must be capable of evolving a *variety* of phenotypes. However, two types of variation can be identified: *diversity* and *disparity* (Jaanusson 1981, Runnegar 1987, Gould 2000). Diversity refers designs that differ in the proportions and dimensions of their parts, but that share the same overall organisation and configuration of parts. Disparity refers to designs that have a fundamentally different organisation and configuration of parts.

If only diversity is required, then the genotype can encode a set of parameters that map directly onto the parametric model of the phenotype. Such systems may be described as *evolutionary design optimisation systems*. The vast majority of evolutionary design systems are optimisation systems. Most of these systems use some form of genetic algorithm. One of the key advantages of these algorithms is that the rules and data structures used are highly generic, and as a result optimisation systems can be applied to a wide variety of conditions.

On the other hand, if disparity is required, then the system must incorporate some kind of generative program that is capable of generating phenotypes from genotypes. In this case, the evolutionary system will evolve the seed modifications and/or process modifications for the generative program. The modifications are encoded as genotypes, and the generative program is then used to generate the different phenotypes. Such systems may be described as *evolutionary design exploration systems*. Exploration systems are much more recent, and also far less common that optimisation systems. Frazer (1995) has developed a number of such exploration systems that have generative programs embedded within them. These systems also follow a more general trend in evolutionary computing of becoming less generic and more bespoke. Developing rules and data structures that are bespoke allows evolutionary systems to tackle much more complex tasks (Michalewicz 1996). As a result of these differences, the algorithms used by optimisation systems are not directly applicable in exploration systems.

2 A Computational Architecture

This paper presents a computational architecture for an evolutionary design exploration system capable of evolving disparate design possibilities. The aim is to evolve designs that, as well as fulfilling various quantifiable criteria, are also interesting and challenging for the design team.

The architecture encompasses both an evolutionary algorithm and an implementation scheme. The algorithm defines an abstract procedure that must be implemented. The algorithm defines two interrelated procedures that might be described as life cycles:

- The life cycle of an individual in the population.
- The life cycle of the population as a whole.

The implementation scheme gives a broad outline of how this algorithm will be implemented, in terms of both software and hardware.

- The software configuration used to implement the algorithm.
- The hardware configuration used to host the algorithm.

The proposed architecture is based on the architecture of existing systems, but also introduces key modifications not found in other systems. The four points above are used to describe both the typical architecture of an exploration system and the proposed architecture. First, the typical is discussed and then the proposed architecture is described in relation to this typical architecture.

2.1 TYPICAL ARCHITECTURE FOR EVOLUTIONARY DESIGN EXPLORATION SYSTEMS

The typical architecture for exploration systems, as developed by Frazer (Frazer and Graham 1992, Frazer 1995, Frazer and Frazer 1996), Bentley (1996) and others (Bentley 1999, Bentley and Corne 2002) will be described by considering the four points identified above:

2.1.1 The life cycle of an individual

The life cycle of an individual tends to be broken down into a sequence of evolutionary steps. For evolutionary design systems, the life cycle is usually broken down into four steps: *(a)* the *reproduction step* creates new genotypes from existing genotypes; *(b)* the *generation step* generates new phenotypes from the genotypes; *(c)* the *evaluation step* calculates a fitness score for each design by simulating or analysing the performance of the phenotype; and, *(d)* the *selection step* selects parents whose genotypes will be used to create new genotypes for the next generation.

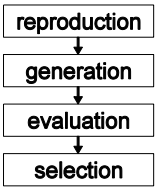

Figure 1: The four life cycle stages of each individual in the population.

2.1.2 The life cycle of the population

The life cycle of the population as a whole tends to follow a *synchronous* pattern. Each generation, individuals progress through their four evolutionary steps in a synchronous manner. For example, that the evaluation steps must be performed after all of the generation steps and before any of the selection steps.

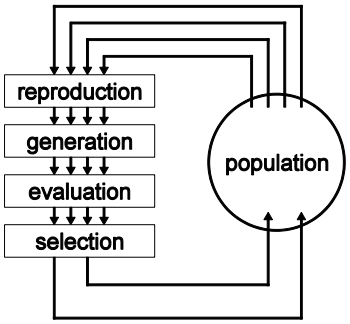

Figure 2: The synchronous life cycle of the population as a whole.

2.1.3 The software configuration

The software configuration varies form system to system. However, two important traits can be identified. First, bespoke rules and data structures that define the evolutionary steps of an individual tend to be embedded within the system. Second, the simulation and analysis programs tend to be custom-written routines that are highly simplified.

2.1.4 The hardware configuration

The hardware configuration tends to be a single stand-alone computer. Other alternatives include networked or distributed configurations that use multiple computers and/or processors. For exploration systems, both these types of configurations are rare.

2.1.5 Limitations of the typical architecture

Buildings are complex entities; as a result, any exploration system aiming to evolve disparate designs for buildings will require bespoke rules and data structures that are highly specific to a type of design. Furthermore, in order to evaluate a building, a wide variety of sophisticated simulations and analyses are required. These requirements result in the typical architecture having certain limitations. Systems implemented following this architecture tend to have three key limitations:

- Such systems tend to be bespoke. The bespoke rules and data structures that embedded within the system will impose biases and constraints on the designs that can be evolved.
- Such systems tend to be relatively slow. The synchronous population life cycle and the stand-alone configuration result in the designs in the population having to be processed in a serial way, one after another.
- Such systems tend to use coarse evaluation methods. Due to the complexity of developing simulation and analyses routines, these tend to become highly simplified.

2.2 PROPOSED ARCHITECTURE FOR AN EVOLUTIONARY DESIGN EXPLORATION SYSTEM

The proposed architecture attempts to overcome these limitations by introducing certain modifications and additions with important consequences for the potential success of such systems. Recent advances in networking and other technologies have also played an important role in developing this new architecture. The proposed architecture will be described by considering the same four points identified earlier.

2.2.1 The life cycle of an individual

The life cycle of an individual is broken down into six steps: reproduction, generation, *validation*, *prediction*, evaluation, and selection. The two additional steps that have been inserted are a validation step and a prediction step. The validation step verifies that the phenotype created by the generation step conforms to certain requirements. The prediction step gathers together the simulations and analyses in one step. The evaluation step can consequently focus on combining the results of the predictions to create a single fitness score.

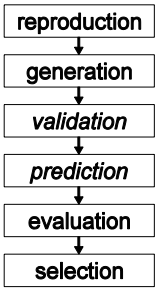

Figure 3: The six life cycle stages of an individual in the population.

2.2.2 The life cycle of the population

The life cycle of the population follows an *asynchronous* rather than a synchronous pattern. Individuals are therefore able to progress through their life-cycle steps irrespective what other individuals in the population are doing. As a result, the population will contain designs at various stages of development: genotypes, phenotypes, validated phenotypes, predicted phenotypes, and evaluated phenotypes. New designs (i.e. genotypes) are continuously being added to the population and existing designs (i.e. evaluated phenotypes) with low fitness scores are continuously being removed from the population.

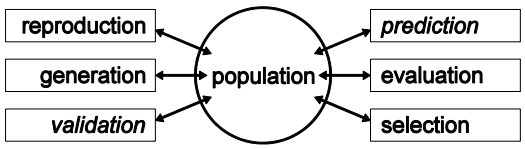

Figure 4: The asynchronous life cycle of the population as a whole.

2.2.3 The software configuration

The software configuration separates out two types of entities that were previously embedded within the system. First, the bespoke rules and data structures are brought together in a set of files that the exploration system can access. Second, the simulation and analysis programs are specified as independent programs that are invoked by the evolutionary system.

2.2.4 The hardware configuration

The hardware configuration uses multiple computers in a networked configuration. The exploration system is decomposed into a server program that maintains the population of designs within a database and a set of client

programs that communicate with the server. For example, the six evolutionary steps that constitute the life cycle of an individual are each performed by a separate client.

2.2.5 Advantages of the proposed architecture

The proposed architecture aims to allow the three limitations identified above to be overcome. Systems implemented following this architecture may overcome these three limitations as follows:

- Such systems will tend to be significantly more generic. Separating out the rules and data structures for the six evolutionary steps allows the schema to be easily changed or replaced without requiring any modifications to the underlying system. In addition, the simulation and analysis programs are also separated out as independent programs, thereby further increasing the generality of the exploration system.
- Such systems will tend to be significantly faster. The asynchronous population life cycle in combination with the networked configuration allows the client programs to be duplicated numerous times. The introduction of the prediction step allows the simulation and analysis programs to be duplicated, thereby allowing these programs to run in parallel on different computers using different operating systems.
- Such systems will tend to use evaluation methods that are significantly less coarse. The independent simulation and analysis programs together with the networked configuration allow existing third-party simulation and analysis programs to be integrated with the exploration system. The insertion of a validation step allows the correctness of the models to be checked prior to simulation and analysis.

3 Conclusions

A computational architecture for an evolutionary exploration system has been proposed. In particular, this architecture focuses on exploration systems capable of evolving disparate building designs.

The architecture uses a networked configuration that decomposes the exploration system into a server program and a set of client programs. The server program manages the asynchronous evolution of the population of designs. The client programs download designs from the server, perform the transformations specified by the evolutionary steps, and then upload the

transformed design back to the server. In total, six evolutionary steps are define – reproduction, generation, validation, prediction, evaluation, and selection – each one performed by a separate type of client. Each of these steps performs a transformation specified by a set of rules and data structures.

Three key advantages of this architecture have been highlighted. First, the architecture allows the bespoke aspects of the system to be separated out from the core generic system. Second, the architecture allows multiple simulation and analysis programs to run in parallel on separate computers. Third, the architecture allows for the integration of existing third part simulation and analysis programs to be integrated.

3.1 FURTHER WORK

The first phase of this research has developed the overall architecture for an evolutionary design system. In the second phase, a software demonstration of the evolutionary design system for building design is being developed that implements the architecture developed in the first phase. The software demonstration is being developed using Java and XML technologies. The clients run Java programs that communicate with Java Servlets on the server which manage the asynchronous exchange of data with the population database.

One key area in the development of the software demonstration is the interaction with prediction and analysis programs. These programs require three types of data: component-based data (e.g. cost estimation), space-based data (e.g. thermal simulation), and networked based data (e.g. structural analysis) (Mahdavi 1998). The data structure for the design model must therefore contain semantic data as well as well as purely geometric data. Recent advances in interoperability between software applications in the building industry will have a very important impact on the feasibility of such an approach. Interoperability between software is based on the idea of a Building Information Model or BIM (sometimes described as virtual building model or building product model). A BIM is a term used to describe a type of data structure developed specifically for describing objects and relationships specific to buildings. The Industry Foundation Classes (IFC) by the International Alliance on Interoperability (IAI) represents the latest effort, jointly by research organizations and commercial vendors, to develop a BIM (Eastman 1999). The analysis and simulation applications now capable of importing the IFC BIM include thermal comfort applications, energy simulation applications, airflow simulation, structural analysis applications, and so forth.

Acknowledgments

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

References

Bäch, T.: 1996, *Evolutionary Algorithms in Theory and Practice*, Oxford University Press, New York.

Bentley, P. and Corne, D. (eds.): 2002. *Creative Evolutionary Systems*, Academic Press.

Bentley, P. J.: 1996, *Generic Evolutionary Design of Solid Objects using a Genetic Algorithm*, Ph.D. Thesis, Division of Computing and Control Systems, Department of Engineering, University of Huddersfield.

Bentley, P. (ed.): 1999, *Evolutionary Design by Computers*, Morgan. Kaufmann Publishers Inc, San Fransisco.

Eastman, C. M.: 1999, *Building Product Models: Computer Environments Supporting Design and Construction*, CRC Press, Boca Raton.

Fogel, D.B.: 1995, *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence*, IEEE Press, New York.

Fogel, L.J.: 1963, *Biotechnology:Concepts and Applications*, Prentice Hall, New Jersey.

Frazer, J.H.: 1995, *An Evolutionary Architecture*, Architectural Association Publications, London.

Frazer, J.H. and Frazer, J.M.:1996, The Evolutionary Model of Design, *in* A. Asanowicz and A. Jakimowicz (eds.) *Approaches to Computer Aided Architectural Composition*, Technical University of Bialystok, pp. 105-117.

Frazer, J.H., Graham P.C.: 1992, Genetic Algorithms and the Evolution of Form, *Proceedings of the Third International Symposium on Electronic Art*, Sydney.

Gould, S. J.: 2000, *Wonderful Life: The burgess shale and the nature of history*, Vintage, London.

Holland, J.: 1975, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor.

Jaanusson, V.: 1981, Functional thresholds in evolutionary progress, *Lethaia* 14:251-260.

Koza, J.R.: 1990, *Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems*, Report No. STAN-CS-90-1314, Stanford University, 1990.

Mahdavi, A.: 1998, A Middle Way to Integration, *Proceedings of the 4th Design and Decision Support Systems in Architecture and Urban Planning Conference*, Maastricht, The Netherlands, July 29.

Michalewicz, Z: 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, Sringer-Verlag.

Purcell, A. T., Gero, J.S.:1996, Design and Other Types of Fixation, *Design Studies*, 17:363 – 383.

Rechenberg, I.: 1965, Cybernetic Solution Path of an Experimental Problem, Ministry of Aviation, Royal Aircraft Establishment (U.K.), Library Translation 1122. Reprinted in *Evolutionary Computation -- The fossil record*, D. B. Fogel, Ed., chap. 8, pp. 297-309, IEEE Press 1998.

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

Runnegar, B.: 1987, Rates and modes of evolution in the Mollusca, *in* K.S.W. Campbell and M.F. Day (eds.), *Rates of evolution*, pp. 39-60, Allen and Unwin, London.