M. A. Schnabel (ed.), *Cutting Edge: 47th International Conference of the Architectural Science Association*, pp. 197–206.

EVOLUTIONARY DESIGN OF HOUSING:

A template for development and evaluation procedures

PATRICK JANSSEN and VIGNESH KAUSHIK
National University of Singapore, Singapore
patrick@janssen.name, vigneshkaushik@gmail.com

Abstract. Evolutionary design is an approach that evolves populations of design variants through the iterative application of a set of computational procedures. This paper proposes a template and set of techniques for creating the development and evaluation procedures. The template defines a clear structure for the procedures, while the techniques provide specific strategies for generating models and handling constraints. A demonstration is presented where the template is used to create development and evaluation procedures for a large complex residential housing project.

Keywords. Evolutionary design; generative modelling; constraint handling; decision chain encoding; point block housing.

1. Introduction

Evolutionary design (Frazer 1995, Bentley 1999, Caldas 2001, Bentley and Corne 2002, Janssen 2004) is an approach that evolves populations of design variants through the iterative application of a set of computational procedures. The development procedure generates design variants, one or more evaluation procedures assess the performance of design variants, and the feedback procedure drives the evolutionary process by applying selective pressure to the population. The feedback procedure applies selective pressure by ensuring that design variants with low performance scores are more likely to be killed, while design variants with high performance scores are more likely to survive, and to be selected for reproduction.

This paper will focus mainly on the development and evaluation procedures. Section 2 describes a template for creating such development and evaluation procedures, section 3 presents the demonstration of the application of the template, and section 4 briefly draws conclusions and indicates avenues of further research.

2. The Development and Evaluation Procedures

The development procedure generates a phenotype, which is a design variant. One or more evaluation procedures generate a set of evaluation scores, which are measures of performance for a design variant. Janssen and Kaushik (2013a) proposed a template for development procedures. This paper builds on this previous template, by expanding the scope to include both development procedures and evaluation procedures. Figure 1 shows the procedures and sub-procedures of the proposed template.

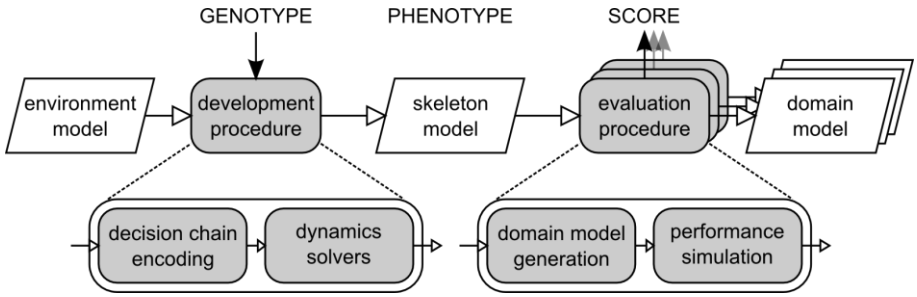

Figure 1: The template for development and evaluation procedures

2.1 DEVELOPMENT PROCEDURE

The development procedure starts with a model of the environment (e.g. the site and surroundings) and generates a skeleton model of the design variant, under the influence of a set of genes. The aim is to create development procedures that result in phenomes with sufficient design variability. Phenomes for architectural and urban designs are typically both highly variable and highly constrained. They are highly variable in the sense that there is no fixed organisational plan, but instead entities can be organised in space in a wide variety of ways. At the same time, these organisations are highly constrained by various rules delineating the validity of possible designs. This type of phenome that is both highly variable and at the same time highly constrained is described as having *bounded variability*.

In order to achieve bounded variability, a key challenge is effectively handling constraints. In particular, two type of constraints need to be handled: combinatorial constraints and geometric constraints. For combinatorial constraints, a technique called decision chain encoding can be used (Janssen, 2004; Janssen and Kaushik, 2013b), while for geometric constraints, dynamic solvers can be used. Together, these two sets of techniques form a simple yet powerful toolkit for handling a wide variety of constraints.

The decision chain encoding technique structures the skeleton generation process as a sequential chain of decision points. Each decision point involves

choosing one option from a list of options. The list of options is created by a set of rules that generate options and then and then filter out options that violate constraints. Note that for each decision, the total number of valid options may not be known and may depend on the previous decisions.

The decision chain encoding technique can therefore be used to generate configurations that adhere to a range of combinatorial constraints. However, the resulting configurations may still violate other geometric constraints. For resolving these geometric constraint violations, dynamics solvers can be used. These dynamics solvers will, over a series of time steps, try to modify the configuration in order to resolve any constraint violations. Depending on the types of constraints, a variety of dynamic solvers can be used, such as particle solvers, rigid body solvers, inverse kinematic solvers, and cloth solvers.

2.2 EVALUATION PROCEDURE

The evaluation procedure starts with a skeleton model and generates an evaluation score for the design variant. As a side effect, the evaluation procedures also generate domain models required for analysis or simulation. Since the skeleton model is a sparse model, data compensation techniques need to be used in order to add the missing data.

3. Demonstration

In this section, the implementation of a development procedure and a set of evaluation procedures for an example design schema are described. The schema is for a residential housing development consisting of a set of point blocks.

3.1 DESIGN SCHEMA

In the design scenario, it is envisaged that a developer plans to build a set of residential buildings with flats arranged around central cores containing circulation and services, a typical typology referred to as a 'point-block'. Typical layouts of the individual flats are defined in advance but the positions and heights of the point blocks and the number of flat types for each point block can be varied.

The site is located in Singapore, with an area of 8.4 hectares and a plot ratio of 2.0. In total, 1400 flats are required. Figure 2 shows the 7 flat types (together with the required quota for each flat type), and the 4 block types. Flats are always arranged around the core in pairs, sharing a common wall and forming vertical stacks of flats of variable height (each between 6 to 12 floors high). At the ground level, each block type can accommodate a differ-

ent number of flats around the core (4, 6, and 8 flats). However, due to the variable stack heights, the number of flats and the core may reduce as it goes up.

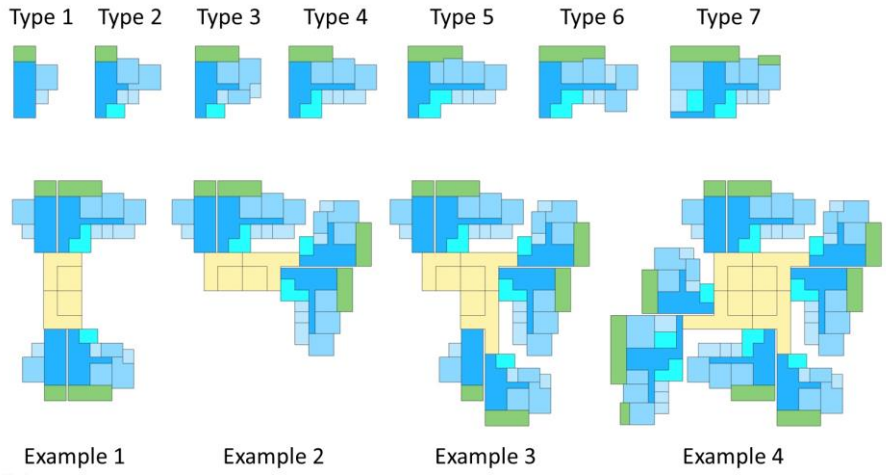

Figure 2: Flat types and block types

Figure 3 shows an example of a single block, together with a conceptual section showing a level of car parking at the bottom covered by a landscaped level on top. The blocks can be freely positioned on the site, with all blocks being accessible either by car via the lower car park and by foot via the upper landscaped level. The upper level has greenery as well as swimming pools and playgrounds. The aim is to optimise the configuration of point blocks and flats so as maximise saleable value and at the same time maximise a number of window performance criteria (described in more detail below).

Both the development and evaluation procedures were created in the procedural modelling software SideFX Houdini. This software allows these procedures to be defined visually using Visual Dataflow Modelling (VDM) (Janssen and Chen 2011). In addition, the software also includes a wide range of procedural modelling tools and dynamic solvers.

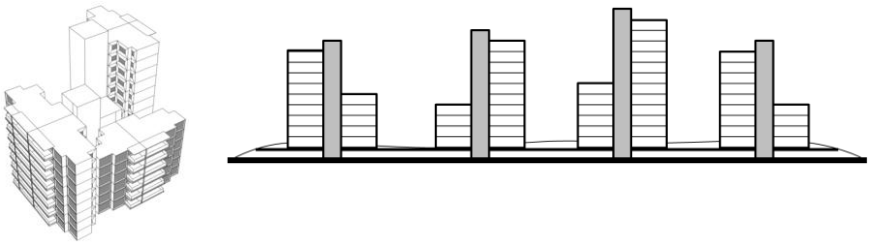

Figure 3: Conceptual section across the site

3.2 DEVELOPMENT PROCEDURE

The development procedure uses a decision chain encoding technique in order to handle the combinatorial constraints on the flat types and a dynamics particle solver technique in order to handle the geometric constraints related to point block positioning.

The point block configurations are generated using a combination of simple parametric modelling techniques and decision chain encoding techniques. The genotype is structured so that there are a repeating set of 16 genes defined for each point block.

The process of positioning and orientating the block on site is performed using some simple parametric rules. For positioning, the site area is modelled as a UV surface, and two genes are used to define a UV position on that surface. For orientation, the third gene is used to rotate the block between 0 and 360 degrees. This process of positioning and orientation ignores possible collisions between blocks, as those will be resolved later using the dynamics particle solver. The remaining 13 genes are used to form a single block of flats through the decision chain encoding process. This process takes into account various combinatorial constraints in choosing the appropriate flat types and stack heights so as to not overshoot the quota.

Since the blocks can be of variable height, the total number of blocks required to achieve the desired number of flats may also vary, with the maximum number of blocks set at 32 blocks. In order to handle this variability, genotypes with redundant genes are used. Since there are a maximum of 32 blocks with 16 genes per block, the total number of genes for generating the blocks is 480. However, due to the redundancy, some of these genes may not be used. For example, if a block is of type 1 (4 flats per floor), then it will only require 2 stack genes and 4 flat type genes, meaning that the other stack and flat type genes are not used. Similarly, if the required number of flats has been achieved with 30 blocks, then the last 32 (16x2) genes will not be used.

In addition the blocks, swimming pools and playgrounds are also added to the site layout. These are positioned in the same way as the blocks, using UV positioning genes. These additional programmatic functions are not directly evaluated, but they play an important role since they create open spaces between the blocks.

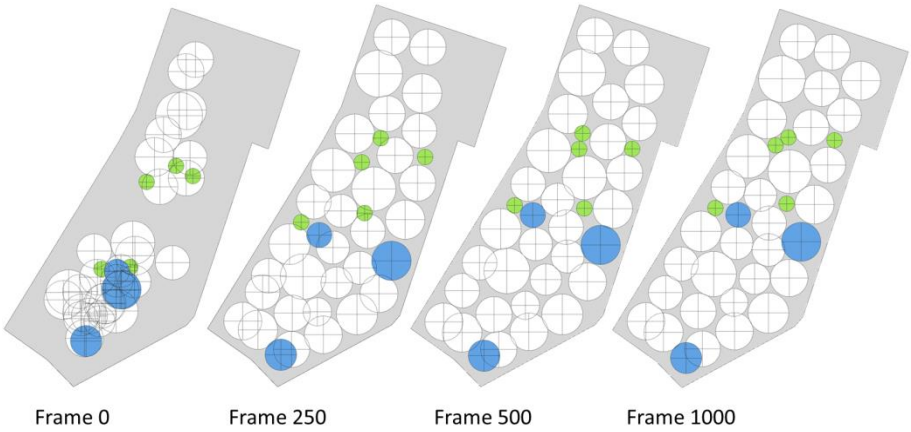

Figure 4: Dynamics particle solver repositioning particles on the site.

Once all the blocks, swimming pools, and playgrounds have been generated, they may be intersecting and overlapping. In order to resolve these issues, a dynamics particle solver is used. For this solver, each block, swimming pool, or playground is represented as a circular particle. For the blocks, the radius of the particle is adjusted to fit the size of the block. The site boundary is defined as a particle boundary and the particles are then positively charged so that they repel one another. These particles are then animated for 1000 frames, allowing the particle to reposition themselves, thereby automatically resolving the overlaps between the blocks. Figure 4 shows a set of frames from the animation.

3.3 EVALUATION PROCEDURES

The first step in the evaluation process is the generation of the domain specific models. The skeleton model resulting from the development procedure is a sparse 2D skeletal model. For each block, the model contains a set of polygons that represent the plans of the flat types tagged with attributes defining the number of floors. A number of different domain specific models are then generated from this 2D skeleton as inputs for analysis and simulation. In addition to the domain models, visualisation models can also be generated in order to help designers evaluate other aspects of the design. Figure

5 shows the 2D skeleton model on the left, along with three other models generated from this skeleton.

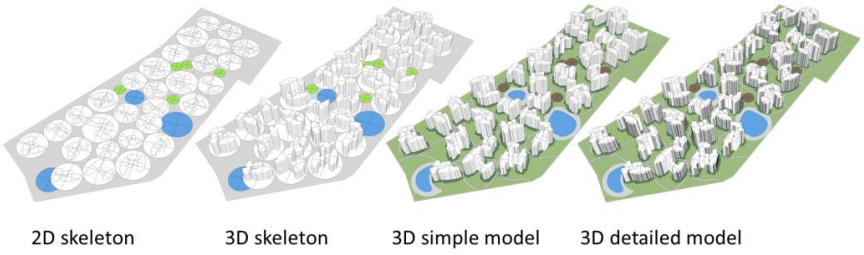

Figure 5: Different types of models generated from the 2D skeleton.

For evaluating saleable value, a domain model is generated that includes only the floor plates of the individual flats. The calculation uses a simple formula that adjusts the price per square meter according to the floor level.

For evaluating window performance, a domain model is generated that only includes the outer building surface together with the living room and bedroom windows of the flats. The window performance takes into account certain site conditions, including a canal along one side of the site that is treated as a desirable view, and a number of busy roads that are treated as sources of noise pollution. For each window, three different criteria are considered:

- Maximisation of unobstructed view in front of the window, where 100% indicates a completely unobstructed view of 50 meter radius.
- Maximisation of views of the canal, where 100% indicates that the whole stretch of the canal in front of the site is visible.
- Minimisation of exposure to road noise, where 100% indicates that there is a road directly in front of the window.

These three criteria could be treated as separate performance criteria. However, this would result in four evaluation scores, which makes the evolutionary search process more difficult. For this example, it was therefore decided to combine these criteria into a single window performance score.

The saleable value and window performance evaluation criteria are in conflict with one another. For a high saleable value, larger number of low height blocks is preferred since it maximises the number of garden flats in the ground floor. However, this results in closely packed blocks that obstruct one another. For a high window performance, smaller number of tall blocks is therefore preferred. The evolutionary process allows designers to explore the trade-offs between such conflicting performance criteria.

3.1. RESULTS

The evolutionary process was executed on the Amazon EC2 cloud computing platform using Dexen, a distributed execution environment for population based optimisation algorithms (Janssen et. al. 2011). Compute instances were started with a total of 200 CPUs.

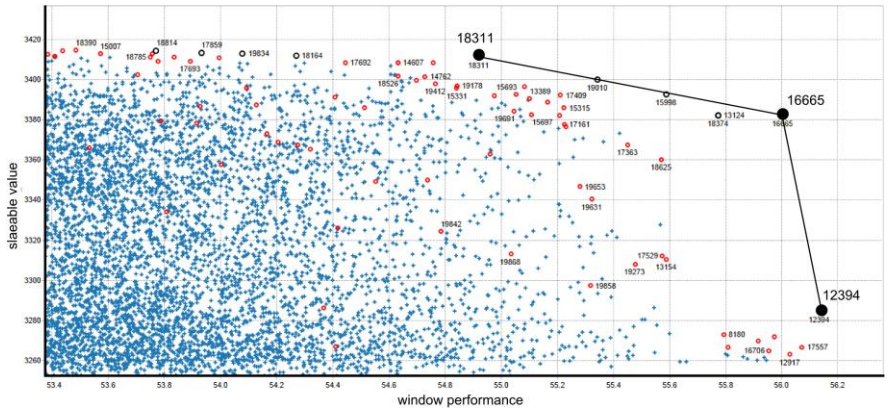

Figure 6: Final non-dominated Pareto set.

The population size was set to 200 and a simple asynchronous steady-state evolutionary algorithm was used. Each generation, 50 individuals were randomly selected from the population and ranked using multi-objective Pareto ranking. The 2 individuals with the lowest rank were killed, and the 2 individuals with the highest rank (rank 1) were used as parents for reproduction. Standard crossover and mutation operators for real-valued genotypes were used, with a mutation probability of 0.02 and crossover probability of 0.9. Reproduction between pairs of parents resulted in 2 new children, thereby ensuring that the population size remained constant.

The final non-dominated Pareto set for the whole population contains a range of design variants with differing trade-offs between saleable value and window performance. The Pareto graph is shown in Figure 6. Three of the design variants from this non-dominated set are shown in figure 7.

ID: 18311
Score windows: 54.9%
Score value: $3411 million
No. of blocks: 29
No. of units: 1458
Plot area ratio: 2.10

ID: 16665
Score windows: 56.0%
Score value: $3382 million
No. of blocks: 29
No. of units: 1448
Plot area ratio: 2.08

ID: 12394
Score windows: 56.1%
Score value: $3284 million
No. of blocks: 27
No. of units: 1410
Plot area ratio: 2.02

Figure 7: Design variants.

4. Conclusions

This paper has proposed a template and a set of techniques for the creation of development and evaluation procedures for evolutionary design. The development procedure generates a sparse skeletal model adhering to a variety of constraints. For combinatorial constraints, decision chain encoding techniques are used, and for geometric constraints, dynamics solver techniques are used. Each evaluation procedure calculates an evaluation score for a specific performance criterion. The skeleton model is used in order to generate a more detailed domain specific model, which is then used for analysis and simulation. The resulting performance data is then condensed into a single evaluation score.

The techniques used in the development and evaluation procedures can be created by designers with limited programming skills using VDM software. A demonstration has been presented where the template is used to create development and evaluation procedures for a large and complex residential housing project. In the demonstration, the development and evaluation procedures are defined using a VDM software called Sidefx Houdini, leveraging the procedural modelling tools and dynamic solvers that exist within the software.

References

Bentley, P.J., editor: 1999, *Evolutionary Design by Computer*s, Morgan Kaufmann Publishers, San Francisco, CA.

Bentley, P.J. and Corne, D.W., editors: 2002, *Creative Evolutionary Systems*, Academic Press, London, UK.

Caldas, L.: 2001, An evolution-based generative design system: using adaptation to shape architectural form, *Doctoral dissertation*, Massachusetts Institute of Technology.

Draghi, J., and Wagner, G.: 2008, *Evolution of evolvability in a development model*, Theoretical Population Biology, **62,** 301–315.w

Eiben, A.E., and Smith J.E.: 2003, Introduction to evolutionary computing. *In: Springer, Natural Computing Series, 1st edition.*

Frazer, J.H.: 1995, *An Evolutionary Architecture*. AA Publications, London, UK

Janssen, P.H.T.: 2004, A design method and computational architecture for generating and evolving building designs. *Doctoral Dissertation*, School of Design, Hong Kong Polytechnic University.

Janssen, P.H.T.; Chen, K.W., and Basol, C.: 2011, Evolutionary development design for non-programmers, *In Proceedings of 29th eCAADe Conference*, Ljubljana (Slovenia), 245-252

Janssen, P.H.T. and Kaushik, V.: 2012, Iterative Design Simulation: Exploring trade-offs between speed and accuracy. In *Proceedings of 30th eCAADe Conference*, Czech Technical University, Czech Republic.

Janssen, P. H. T. and Kaushik, V.: 2013a, Skeletal modelling – a developmental template for evolutionary design. *In Proceedings of the 18th International Conference on Computer-Aided Architectural Design Research in Asia (CAADRIA)*, Singapore, 15-18 May 2013, 705–714.

Janssen, P. H. T. and Kaushik, V.: 2013b, Decision chain encoding: Evolutionary design optimization with complex constraints. *In Proceedings of the 2nd EvoMUSART Conference*, Vienna, Austria, 3-5 April 2013, 157–167.

Kumar, S., and Bentley, P.J..: 1999, Three ways to grow designs: a comparison of embryogenies for an evolutionary design problem, *In Proceedings of the Genetic and Evolutionary Computation Conference '99*, 35–43.

Mahner, M. and Kary, M.: 1997, What exactly are genomes, genotypes and phenotypes? And what about phenomes? *J. Theor. Biol*, **186**, 55–63.

Michalewicz, Z.: 1996, *Genetic Algorithms + Data Structures = Evolution Programs*, Berlin Heidelberg: Springer-Verlag, 3rd edition (First edition 1992).