P. Janssen, P. Loh, A. Raonic, M. A. Schnabel (eds.), *Protocols, Flows and Glitches, Proceedings of the 22nd International Conference of the Association for Computer-Aided Architectural Design Research in Asia (CAADRIA) 2017*, 303-313.

EVOLUTIONARY URBANISM

Exploring Form-based Codes Using Neuroevolution Algorithms

PATRICK JANSSEN
[1]*National University of Singapore, Singapore*
[1]*patrick@janssen.name*

Abstract. Form-Based Codes are legal regulations adopted by local government that allow specific urban forms to be achieved. Such codes have a significant impact on the performative potential of the urban environment. This paper explores the possibility of using a neuroevolution algorithm to elucidate the complex relationship between Form-based Codes and their performative potential. More specifically, Compositional Pattern Producing Networks (CPPN) are used to generate parameter fields, which then drive the generation of varied urban models. For evolving the CPPN networks, a neuroevolution algorithm is used, called Neuroevolution of Augmenting Topologies (NEAT). In order to test the feasibility of the proposed approach, an abstract experiment is described in which a population of urban models are evolved, optimising a set of performance criteria related to the vista and location of the residential units.

Keywords. Form-based codes; evolutionary design; neural networks; neuroevolution; urban planning.

1. Introduction

This research explores the use of evolutionary algorithms to help understand the impact that planning codes may have on the performative potential of the urban environment. The research focuses specifically on the development of Form-based Codes (Parolek et al. 2008), which are legal regulations adopted by local government that allow specific urban forms to be achieved. Form-based Codes differ from the more common Use-based Codes, which focus on the division of areas of land into zones within which various uses are permitted. They also differ from various other type of planning and design guidelines, which tend to be advisory rather than regulatory.

Form-based Codes have a significant impact on the performative potential of the urban environment. To date, Form-based Codes have been applied in relatively

low density types of scenarios. In high density mixed-use districts, the interactions and possible conflicts between urban blocks are accentuated. As such, it is argued that Form-based Codes could be used to control and regulate these interactions and conflicts. However, the relationship between the urban forms and their performative potential is highly complex and cannot be easily understood by the planners developing such codes. This research therefore explores the idea of using evolutionary algorithms to help elucidate this relationship.

Evolutionary algorithms 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 evolutionary procedures that create, transform, and delete individuals in the population. In order to evolve urban models, two types of procedures need to be defined: a developmental procedure that generates different variants of urban models and one or more evaluative procedures for evaluating the performative potential of each urban model. These procedures need to be defined by the planner in order to incorporate variables and objectives that are seen as being relevant and worthy of investigation.

In general, the developmental procedure starts with a set of parameters encoded as genes in the genotype, and then generates urban blocks for all the parcels in the site area. The aim is to generate urban blocks that can be used as a basis for developing Form-based Codes. However, such codes should still allow significant flexibility, so that architects can develop designs that are diverse and that vary in their specific configurations and layouts. As a result, the urban blocks that are generated must remain relatively abstract, consisting mainly of a massing model.

One of the key challenges when creating such a developmental procedure is encoding variability over the site area. For example, depending on the local context, it may be preferable that the urban blocks at one side of the site have a different typology or different height from the urban blocks at the other side. The question is, how can this variability be efficiently encoded in the genotype? A number of researchers have used parametric modelling techniques to generate or model Form-based Codes. For example, see Kim et al. (2011), Janssen at al. (2016), and Schnabel and Ayadin (2017). However, in these previous examples, the parametric modelling process was not part of an evolutionary system and as a result many aspects were still manually controlled. Since this research aims to evolve urban models, a fully automated model generation process is required.

One approach is to define a genotype that directly encodes all the parameters for all the parcels on the site. However, this would result in a very large number of genes with a high-dimensional search space that would frustrate the evolutionary process. Furthermore, since all the genes would be totally independent from one another, it would also likely result it urban models incorporating many sudden changes from one parcel to the next.

In this paper, an alternative approach is proposed using a neuroevolution algorithm to generate and evolve urban models. Section 2 gives a brief overview of the proposed approach and section 3 presents a demonstration. Finally, section 4 draws conclusions and discusses future research directions.

2. Proposed Approach

The proposed approach uses a type of neural network called a Compositional Pattern Producing Network (CPPN) for generating variability over the site and a neuroevolution algorithm called Neuroevolution of Augmenting Topologies (NEAT) for evolving CPPNs. Together, this pair of algorithms is known as CPPN-NEAT. For more details, see Stanley and Miikkulainen (2002) and Stanley (2007).

2.1. GENERATING VARIABILITY

The developmental procedure needs to generate varied urban blocks for the parcels on the site. To achieve this, the CPPN is used for generating a series of patterns that cover the entire site area, referred to as parameter fields. These parameter fields allow parameter values to be assigned to each parcel in a way that is highly generic and also very compact.

The CPPN neural network has two input nodes, zero or more hidden nodes, and one or more output nodes. The inputs and outputs are all normalized, in the range [0.0, 1.0]. The two CPPN inputs are the position on the site, defined as (u,v) coordinates on a normalised two-dimensional parametric surface that covers the whole site area. The CPPN outputs are the normalised parameter values for a specific (u,v) position in the parameter field.

Each parcel on the site is assigned a set of parameter values through a three step process of sampling the parameter fields. First, the (x,y) coordinates for each parcel centre point are normalised by converted them to (u,v) coordinates. Second, the (u,v) coordinates are fed into the CPPN, thereby generating a set of normalised parameter values. Third, the normalised parameter values are mapped to non-normalised values that can be applied within the user-define parametric models. These parametric models are then used to generate the parcel blocks,

An urban model will consist of many parcel blocks. The generation of the urban model needs to ensure that certain global constrains are met. Three constraints are considered: required area constraints, plot ratio constraints, and block typology constraints. First, the required area constraints define the required floor areas for different programmatic functions. Second, the plot ratio constraints define the maximum and minimum plot ratios for the different programmatic functions. These are necessary in order to avoid evolving urban models with unrealistically high or low plot ratios. Third, the block typology constraints define a selection of user-define parametric typologies that can be used for generating blocks. These typologies should ensure that the spaces that are generated are appropriate for the programmes in question. This includes accessibility to the circulation system and accessibility to daylight and fresh air.

2.2. EVOLVING URBAN MODELS

The NEAT evolutionary algorithm evolves these CPPNs by encoding the neural network in the genotype, including all the hidden nodes, the weights, and network typology. Such neuroevolution algorithms need to overcome certain key challenges relating to the diversity of networks in the population. The NEAT algorithm uses three key techniques: tracking genes with history markers to allow

crossover between networks with different topologies, applying speciation to preserve innovations, and developing topologies incrementally from simple initial structures. On benchmark problems, the NEAT algorithm performs better than many other contemporary neuro-evolutionary techniques. For more details on the CPPN-NEAT approach, see Stanley and Miikkulainen (2002).

In order to test the feasibility of the proposed approach, an experiment is described in which a series of urban models are evolved.

3. Demonstration

The demonstration scenario considers a theoretical urban site, 1km x 1km square, situated on the waterfront facing a promenade (figure 2). On either side, a set of existing high density urban areas are assumed to exist, overshadowing some of the site. At the back, the site faces onto a noisy expressway.

For this site, a fixed street pattern is assumed, consisting of an orthogonal grid of primary and secondary streets. The primary streets (26m wide, including pavements) divide the 1km square are into a 3 x 3 grid, and the secondary streets (14m wide, including pavements) further subdivide each grid a smaller 3 x 3 grid. This results in a 9 x 9 grid, with 81 relatively small urban parcels measuring 96m x 96m.

The task for the planner is to create a set of Form-based Codes for a mixed-use urban district that would cater for 100,000 people, including both residential and commercial facilities. The research aim is to evolve urban models for the site that would maximise desirable qualities and minimise undesirable qualities.

3.1. DEVELOPMENTAL PROCEDURE

The developmental procedure consists of three main stages of generation: normalised parcel parameters, mapped parcel parameters, and parcel blocks.

3.1.1. Stage 1: Generation of normalised parcel parameters

For this demonstration, a CPPN is set up to create four parameter fields. The first field is used to define the location of three open parks within the site. The second and third fields are used to define the residential and commercial plot ratios. The fourth field is used to define a rotation of the block on the site.

Normalised parameter values are generated for each parcel on the site by sampling the parameter fields.

3.1.2. Stage 2: Generation of mapped parcel parameters

For each parcel, the normalised parcel parameters are mapped to values that directly relate to the blocks generated in the next stage. The first and fourth parameters have very simple mappings. The first is used to define a rotation parameter for the block massing on the site. The normalised parameters in the range [0.0, 1.0] are proportionally mapped to {0, 90, 180, 270}. The fourth parameter is used to define the park parameter, which can have one of two values: {0, 1}, with 1 indicating that it is a park. The parameter values for all parcels are sorted, and the

parcels with the lowest parameter values are assigned a value of 1.

The second and third parameters have more complex mappings. These are used to define the the residential and commercial plot ratios. The complexity is due to the global constraints that need to be taken into account, namely the total floor area needs to be correct, and the plot ratios need to be within the allowable range.

For this demonstration, the residential area is set to 2.4 million m2 and the commercial area is set to 1 million m2. The minimum and maximum plot ratios for both residential and commercial programmes are set to 0.4 to 6.0. This results in a total plot ratio in the range [0.8, 12.0].

Given a set of n parcels, the plot ratio, PR , of each parcel is:

$$PR = \frac{\left((A - (\sum_{i=1}^{n} a_i \cdot PR_{\text{min}})) \cdot \left(p / \left(\sum_{j=1}^{n} p_j\right)\right)\right) + (a \cdot PR_{\text{min}})}{a} \quad (1)$$

where A is the required total floor area for all parcels, PR_{min} is the minimum allowable plot ratio, a is the area of the parcel, and p is the parameter value for the parcel. The same formula is used to calculate both residential and commercial plot ratios.

This calculation is performed for all parcels, and if the resulting plot ratio of any of the parcels is greater than the maximum allowable plot ratio, then all parameter values are incremented by a small amount, and the resulting plot ratios are recalculated. This process is repeated until no parcels exceed the maximum allowable plot ratio. (Incrementing all parameter values by a small amount results in a redistribution of plot ratios, so that those with low values increase a little and those with high values decrease a little.)

At the end of this stage, each parcel will have four mapped parameters: the park toggle, the residential plot ratio, the commercial plot ratio, and the block rotation.

3.1.3. Stage 3: Generation of parcel blocks

For each urban parcel, the mapped parcel parameters are used to generate a block. Three typologies are defined, labelled as A, B, or C. Type A is a perimeter block typology, with perimeter units (maximum three stories high) arranged around a courtyard, with the possibility of an additional tower block in the centre. Type B is a slab block typology, with four slabs blocks arranged around a courtyard. Type C is a podium block typology, with variable sized podiums and either one or two towers. For typologies A and B, the lowers floors are assigned as commercial and upper floors as residential. For Type C, the podium is commercial and the towers above are residential.

The typologies are selected based on the sum of the two plot ratios, as follows:

- Type A: for parcels where the plot ratio sum is in the range [0.8, 3.5]
- Type B: for parcels where the plot ratio sum is in the range [3.5, 6.0]
- Type C: for parcels where the plot ratio sum is in the range [6.0, 12.0]

Once the typology has been selected, a block is then generated using a parametric modelling procedure. The number of floors in the model are calculated in order to

achieve the required plot ratios for both residential and commercial programmes. The process of generating these blocks is not described in detail as it is relatively straight forward using typical parametric modelling tools.

3.2. EVALUATIVE PROCEDURE

The developmental procedure creates variants of urban models that all abide by the specified constraints, including overall minimum and maximum plot ratios and a fixed total floor area for residential and commercial programmes. However, each model will also differ with respect to various other performance criteria. In order to be able to evolve urban models, an evaluative procedure is required for calculating a relative fitness measure.

For this demonstration, the evaluative procedure focuses on the residential units. (Although the commercial floors are not directly evaluated, they may still have an impact on the quality of the residential units.) Each urban model has approximately 18 thousand units, each between 108 to 324 m2. (Note that when detail plans are developed, these larger unit may be subdivided into smaller apartments.)

For the evolutionary process, the evaluative procedure must assign a single overall score to the whole urban model. The strategy taken is to first calculate a 'quality' score for each residential unit and to then calculate the fitness of the overall urban model as the average score for the worst 25% of the units. This fitness score ensures that the evolutionary pressure is always focusing on reducing the number of badly performing units.

The quality score for each unit includes two aspects: the quality of the vista and the quality of the location. For each of these two aspects, a score is assigned in the range [-1.0, 1.0], indicating a range from bad to good, with a zero score indicating an acceptable quality. The score for a particular unit is taken to be the minimum (i.e. the worst) of these two scores.

3.2.1. Evaluation procedure for vista quality

The vista procedure aims to quantify the potential quality of the vista of a residential unit. Depending on the block configuration, a unit can have between one and three exterior facades, facing in different directions. (For example, a corner unit would have two facades.) For each unit, the overall vista quality is calculated as the maximum vista score of all the facades. The reasoning behind this is that it is assume that when individual units are designed in detail, then designer will select the best vista as the primary direction for the main view and windows.

For each facade, the procedure calculates scores for three performance objectives, all in the range [-1.0, 1.0]. The overall vista score for the facade is defined as the minimum of these three performance scores.

- Unobstructed view: The horizontal unobstructed view calculated from the centre of the facade polygon for a 60 deg cone of vision, with a maximum depth of 100m. A score of -1.0 indicates that the area of the cone of vision is 0, while a score of +1.0 indicates that area of the cone of vision is equal to the maximum area.

- Scenic view: The visibility of a set of desirable and undesirable points calculated from the centre of the facade polygon. 12 undesirable points are placed along the expressway and 12 desirable points are placed along the seafront promenade. A score of -1.0 indicates that all undesirable points are visible, while a score of +1.0 indicates that all desirable points are visible.
- Insolated view: The visibility of the sun during the two solstice days (21st June and 21st December) calculated from the centre of the facade polygon. For each solstice day, 7 points are defined, starting and ending with sunrise and sunset. A score of -1.0 indicates that the sun is always visible, while a score of +1.0 indicates that the sun is never visible. (Note that this demonstration is assumed to be located in the tropics, where the sun is never desirable.)

Typically, each urban model will have approximately 26 thousand individual residential facades. Since each evolutionary run may result in approximately 30 thousand urban models being generated, this results in approximately 780 million facades being evaluated. It is therefore essential that the procedure is fast to execute. The scores for the performance objectives are all calculated based purely on the geometry of the model, using a raycasting algorithm.

3.2.2. Evaluation procedure for location quality

The location procedure aims to quantify the potential quality of the location of a residential unit. For this procedure, the proximity to desirable and undesirable locations are calculated. Desirable locations are given a score in [0.0, 1.0] while undesirable locations are given a score [-1.0, 0.0]. For each unit, the overall location quality is calculated as the sum of the maximum desirable score and the minimum undesirable score. This results in an overall location quality score in the range [-1.0, 1.0].

Two desirable locations and one undesirable location are defined, as follows:

- Proximity to parks: The minimum straight-line distance from the unit to one of the three parks is calculated. A score of 0.0 indicates that the unit is more than 300 meters from the parks, while a score of 1.0 indicates that the unit is right next to a park.
- Proximity to train station: The minimum straight-line distance from the unit to the train station (located adjacent to the expressway) is calculated. A score of 0.0 indicates that the unit is more than 500 meters from the station, while a score of 1.0 indicates that the unit is right next to the station.
- Proximity to main roads: The minimum straight-line distance from the unit to any of the main roads is calculated. A score of -1.0 indicates that the unit is right next to a main road, while a score of 0.0 indicates that the unit is more than 30 meters from the main roads.

3.3. RESULTS

The CPPN-NEAT algorithm was used to evolve a population of 200 individuals for 300 generations, using the NEAT Python library (NEAT-Python 2016). For a description of the various configuration settings, see NEAT-Python Docs (2016). For the initial population, a set of fully connected neural networks was randomly

generated with two input nodes (for the u and v values), 5 hidden nodes, and 4 output nodes (for generating the 4 parameter fields). The minimum and maximum weights were set to -30 and 30 respectively, with a weight standard deviation of 1. The activation function was set to only use the sigmoid function. For reproduction, elitism was set to 10 and the survival threshold was set to 0.2. For speciation, the compatibility threshold was set to 2.3 and the maximum stagnation threshold was set to 10. For all other settings, default setting were used. In total, approximately 27 thousands different urban models were evolved, taking approximately 15 hours on a laptop with an Intel i7 2.3GHz processor with 4 cores.

The improvement in fitness over time is shown in figure 1, ending at a value of -0.03. Thus, in the final evolved urban model, the average overall quality score of the worst 25% of residential units was -0.03. This is just below the acceptable level (which is defined as being 0).

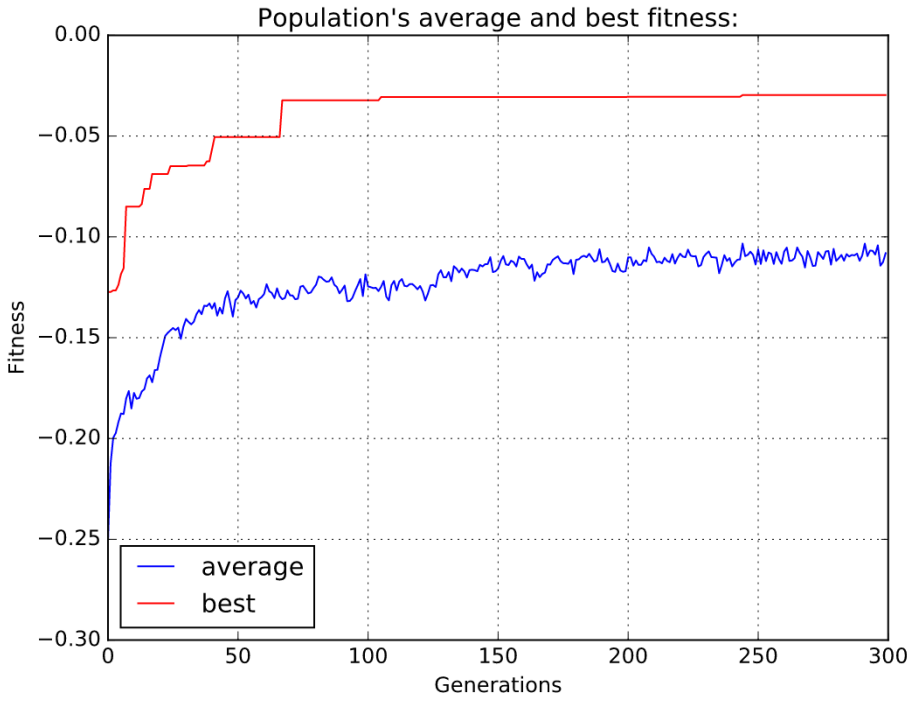

Figure 1. Fitness during the evolutionary run, showing best and average fitness in the population.

The urban model with the highest score is shown in figure 2. The residential programme is shown in yellow, the commercial in blue, and cores in dark grey. An analysis of the different species revealed certain key common traits. First, the evolved urban models all have the parks close to the waterfront and have two clearly defined zones - a mainly residential zone on the left side facing the water-front and a mainly commercial zone on the right side facing the expressway. In

addition, the type B blocks have been mostly eliminated due to the fact that those typologies tend to have residential units with low vista scores. Only when there are sandwiched between A and C typologies do the B typologies perform well.

The plot ratios for residential and commercial programmes have also been adjusted so as to minimise the number of residential units close to the main roads. By pushing up the residential plot ratio in the commercial zone, the type A perimeter block typologies have become almost completely commercial, with only a few residential units remaining in the residential tower

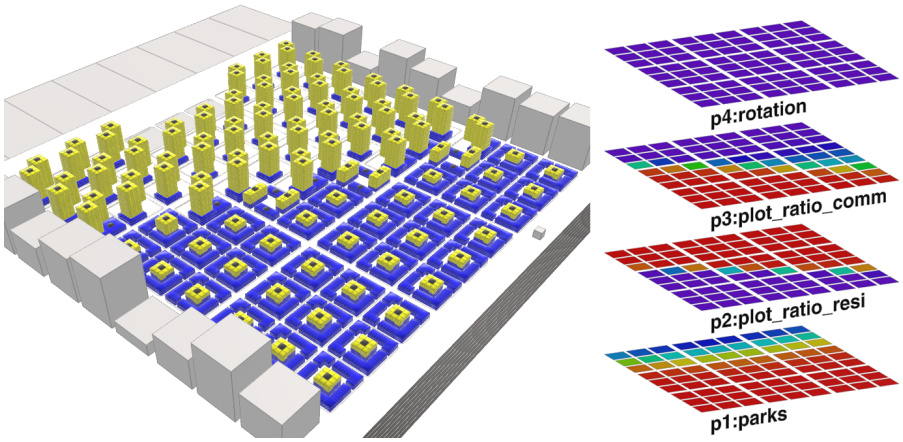

Figure 2. The best performing urban model in the population (ID= 26900). Left: the 3D model. Right: the four parameter files used to generate the model.

4. Conclusions

This paper explores the possibility of using CPPN-NEAT algorithm to elucidate the complex relationship between Form-based Codes and their preformative potential. The demonstration has shown how CPPNs can be used for generating variability in large scale urban models while still abiding by certain constraints. Furthermore, the paper has also explored how the performative potential of these urban models can then be evaluated using simplified metrics that are fast to execute.

In future research, the developmental and evaluative procedures can be further elaborated in various ways. For the developmental procedure, the site as currently defined is highly simplified. More realistic scenarios could include the evolution of variable street networks with different patterns, the evolution of urban fabrics that include both empty and occupied parcels, or the evolution of urban transformations that unfold over time. For the evaluative procedure, many other aspects could be analysed beyond vista and location quality. This may also lead to multi-objective approaches being considered.

Initial experiments in evolving designs using the NEAT algorithm are showing some promising results. However, the species in the current population are quite

similar, with the overall diversity of urban models being poor. It is believed that higher performing urban models could be evolved by either adjusting configuration settings and/or by creating bigger populations. In general, further experiments need to be conducted in order to investigate the impact that various CPPN-NEAT configuration settings have on the effectiveness of the evolutionary search process.

In order to be able to conduct such additional experiments, one of the key challenges is execution speed. The execution time is currently already more than half a day, and with larger populations, this can easily increase to a number of days. In the experiments conducted so far, the developmental and evaluative procedures are being executed in parallel, on multiple cores within a single machine. In order to achieve further significant increases in execution speed, it is possible to also distribute the execution of these procedures over multiple machines. In the next stage of the research, a distributed implementation of the CPPN-NEAT algorithm will be developed, thereby allowing a large number of additional experiments to be conducted.

References

"NEAT-Python" : 2016. Available from <https://github.com/CodeReclaimers/neat-python> (accessed Dec 2016).

"NEAT-Python Docs" : 2016. Available from <http://neat-python.readthedocs.io/en/latest/> (accessed Dec 2016).

Janssen, P., Stouffs, R., Mohanty, A., Tan, E. and Li, R.: 2016, Parametric Modelling with GIS, *Proceedings of eCAADe 2016*, Oulu, Finland, 59-68.

Kim, J.B., Clayton, M. and Yan, W.: 2011, Parametric Form-Based Codes: Incorporation of land-use regulations into Building Information Models, *Proceedings of ACADIA Regional 2011*, 217-223.

Parolek, D.G., Parolek, K. and Crawford, P.C.: 2008, *Form Based Codes: A Guide for Planners, Urban Designers, Municipalities, and Developers*, Wiley.

Schnabel, M.A.r.c., Zhang, Y.i.n.g.y.i. and Ayin, S.e.r.d.a.r.: 2016, Using Parametric Modelling in Form-Based Code Design for High- dense Cities, *Proceedings of SBE 2016*.

Stanley, K. O.: 2007, Compositional Pattern Producing Networks: A Novel Abstraction of Development, *Genetic Programming and Evolvable Machines*, **8**(2), 131–162.

Stanley, K.O. and Miikkulainen, R.: 2002, Evolving neural networks through augmenting topologies, *Evolutionary Computation*, **10**, 99-127 .