Brock University

Department of Computer Science

Generative Representations for Artificial Architecture and Passive Solar Performance

Adrian Harrington and Brian J. Ross

Technical Report # CS-13-02
March 2013

Brock University
Department of Computer Science
St. Catharines, Ontario
Canada L2S 3A1
www.cosc.brocku.ca

Generative Representations for Artificial Architecture and Passive Solar Performance

Adrian Harrington
Dept. of Computer Science
Brock University
St. Catharines, Ontario
Canada L2S 3A1
Email: ajpharrington@gmail.com

Brian J. Ross
Dept. of Computer Science
Brock University
St. Catharines, Ontario
Canada L2S 3A1
Email: bross@brocku.ca

***Abstract*—This paper explores how the use of generative representations improves the quality of solutions in evolutionary design problems. A genetic programming system is developed with individuals encoded as generative representations. Two research goals motivate this work. One goal is to examine Hornby's features and measures of modularity, reuse and hierarchy in new and more complex evolutionary design problems. In particular, we consider a more difficult problem domain where the generated 3D models are no longer constrained by voxels. Experiments are carried out to generate 3D models which grow towards a set of target points. The results show that the generative representations with the three features of modularity, regularity and hierarchy performed best overall. Although the measures of these features were largely consistent to those of Hornby, a few differences were found.**

Our second research goal is to use the best performing encoding on some 3D modeling problems that involve passive solar performance criteria. Here, the system is challenged with generating forms that optimize exposure to the Sun. This is complicated by the fact that a model's structure can interfere with solar exposure to itself; for example, protrusions can block Sun exposure to other model elements. Furthermore, external environmental factors (geographic location, time of the day, time of the year, other buildings in the proximity) may also be relevant. Experimental results were successful, and the system was shown to scale well to the architectural problems studied.

I. Introduction

Artificial architecture is an area of research which combines the fields of architectural design and computer science [1]. Challenged with the difficult task of designing entire building structures, architects are using computer technology to aid in the design task. A call for systems which aid in the creative design process for generating structural form has been made [2], to alleviate problems such as design fixation by providing a range of inspirational models.

There are many directions of research in artificial architecture, which is understandable given the many goals when developing architectural forms. An example of such a goal is to maximize the amount of natural light that hits a model. Watanabe tackled this in his Sun God City project, which built form through rule-based logic [3]. Work has also investigated the procedural synthesis of structures using generative grammars, which are amenable to computer implementation and control. Shape grammars [4] have been used in creating plans for whole cities [5], [6], procedurally model and generate buildings [7], including structurally-sound masonry buildings [8]. Systems have also been created which allow for interactive editing of grammars for the generation of architecture [9].

In pursuit of the goal of creating higher-level, more autonomous software tools for artificial architecture design, ideas from artificial intelligence are being investigated. For example, evolutionary design explores the power of evolutionary computation for (semi-)automatic generation of design solutions to complex, difficult problems [10], [2]. Selected examples of such systems are the following. GENR8 is a design tool, developed using grammatical evolution for the generation of organic surface [11], [12]. Frazer evolves building envelopes and towers using mathematical functions to generate form [13], [14]. Shape grammars have been used with genetic programming for the creation of 3D models [15], [16], [17]. Kicinger *et al.* present a survey of research using evolutionary algorithms for stuctural design [18].

One branch of evolutionary design research in artificial architecture is the use of generative representations for model synthesis. Many researchers are examining the evolution of grammar-based model generators, which are used to create solutions or families of solutions [19], [15]. Using this technique, the creation of robust phenotypes is made possible through the evolution of simple genotypes. For example, the use of L-systems [20] has proven to be successful in this regard [21], [22]. It is of interest how these grammar-based systems compare, in both achieving high fitness scores and enabling better solutions for problems of increased complexity.

The generation of higher quality results in architectural applications is an important research goal in evolutionary design. Research has examined this issue, with the goal of improving upon the complexity of problems solved by evolutionary systems [23], [24]. Hornby identifies three basic characteristics of designs to complex problems – modularity, reuse and hierarchy (see Table I). These factors are present in solutions in both engineering and software development [24]. Furthermore, the attributes which enable these three characteristics are identified to be combination, control-flow and abstraction [23]. Hornby develops an evolutionary system which enables these three attributes and creates metrics which attempt to monitor how each of them is present in a candidate solution. His findings suggest that the best solutions are found when all these characteristics are enabled.

The research in this paper uses genetic programming (GP) and L-systems to automatically synthesize 3D models for artificial architecture. There are two main research objectives in this investigation. The first objective is to re-examine the applicability of Hornby's generative representations for the automatic generation of 3D models in a scalable design problem. We will consider a more difficult problem domain, one where the 3D models are not constrained to a voxelized space (unlike in [23], [24]). Here, the target model is represented by a set of randomly created points. We will establish the complexity and performance issues that Hornby identified, as they arise in this new problem setting.

Our second research objective is to examine how the system performs by adding a new architectural evaluation criteria, that being passive solar performance. Passive solar design is based on the idea that the Sun can be used as a source of heating and illumination for a building [25]. A building's passive solar performance is a measurement of its success in accomplishing such tasks in a given problem setting. Such factors are important in designing buildings that are energy efficient, especially in today's environment of growing climate change.

The paper is organized as follows. Section II contains details on enabling and measuring scalability with generative representations. Section III presents the design language and evaluation methods used. Section IV shares results from experiments performed using generative representations. Section V contains experiments on architectural synthesis. Section VI provides conclusions. See [26] for further details of this research.

II. Generative Representations

Tree-based genetic programming (GP) represents programs as a syntax trees [27]. These trees are directly interpreted in order to solve a problem. In the evolutionary design of architectural models, the language denoted in such a tree may represent aspects of a model to be built, for example, each of the four legs and top surface of a table [24]. There is a concern with this "direct encoding" method, however, that it does not scale well to problems of growing complexity. This is because complex problems will require correspondingly complex trees, which may be difficult for GP to discover in the search space.

Generative representations are a different approach to the encoding of individuals in GP. In this technique, syntax trees (genotypes) create some form of grammar which is then translated in order to generate a solution (phenotype). In Hornby's work, the generative representations are encoded as L-systems, which allow for abstraction, control flow and combination [19].

We have developed a GP system which extends ECJ [28], allowing for individuals to be encoded as tree-based generative representations. For more information on how a grammar is represented in tree form, see [26]. An individual is transformed from tree-form into a parametric L-system with conditionals, loops (represented by {body}(n)), and push\pop operators (represented by [and]). The grammar is composed of a starting string, value for rewrite depth and a set of production rules. Each production rule can maintain one to many condition-pairs, where a condition must be met before a rewrite operation can be performed. The starting string in the developed system always calls the first production rule *A()*, with a number of arguments. An example of a grammar with one production rule and two condition-pairs is as follows:

Starting string: *A(1,3)*
Rewrite depth: *3*

First, the arguments and rewrite depth are determined by ERCs. Next, the production rules are determined by parsing the GP-tree:

A $\rightarrow$ b $>$ 4: rt(1) [ ] A(3,a)
A $\rightarrow$ b $>$ 1: {ex(a)}(2) [ ] A(b,4+a)

Once the generative system has been created, it is rewritten as follows:

Starting string: *A(1,3)*
Rewrite 1: {*ex(1)*}*(2) [ ] A(3,5)*
Rewrite 2: {*ex(1)*}*(2) [ ] rt(1) A(3,3)*
Rewrite 3: {*ex(2)*}*(2) [ ] rt(1)* {*ex(3)*}*(2) [ ] A(b,4+a)*
Design encoding: *ex(2)ex(2) rt(1) ex(3)ex(3)*

The design encoding is then taken and executed to generate a three-dimensional model. In this context, *ex()* and *rt()* are two design operators that alter a 3D model in some way. In this example, all of the features of modularity, reuse and hierarchy are enabled.

Five encodings are implemented, that incorporate various combinations of the characteristics of modularity, reuse and hierarchy. These representations are:

1) *Simple*: No features are enabled in this representation. It contains only one procedure, whose condition always evaluates to true. This representation is similar to the classical GP encoding.
2) *Modular*: In this encoding, procedures and procedure calling are allowed. Only the first procedure is allowed to execute any other procedure. The number of condition-pairs are limited to one.
3) *MR*: This representation enables repetitions as well as procedures. It enforces all of the constraints of the Modular encoding, though the condition-pairs can be set to more than one.
4) *MH*: Hierarchy is enabled by allowing procedures to call one another, creating a stack of procedure calls. Each procedure is limited to a maximum number of times it may be called.
5) *MRH*: This encoding enables all of the features. It allows

TABLE I
MEASURES OF COMPLEXITY.

Measure	**Description**
Modularity	The number of times a structural module is contained in a design.
Reuse	The average number of times elements are used to create a design.
Hierarchy	The number of layered modules in a design.

for multiple procedures, procedures to call any other procedures and the use of repetitions.

A new set of parameters are accepted by the system, to setup the generative representations. *Procedure amount* determines the number of procedures. *Procedure arity* determines the number of arguments each procedure accepts. *Symbol arity* determines the number of arguments a build operator has. *Seed object* is the geometry the system begins with. *Condition pairs* is the amount of conditional pairs per procedure.

A. Measuring complexity

It is important to understand how these features correlate with fitness. To gain some perspective on how the features of generative representations play a role in allowing evolutionary design systems to perform better in more complex problem domains, Hornby's metrics were implemented that examine individuals in both genotype and phenotypic form (see Table I). Hornby measures the characteristics of modularity, reuse and hierarchy in his representations to contrast how they relate to fitness scores [24]. Here, modularity refers to the grouping of elements so that they can be manipulated as a unit. Reuse is a repetition or reuse of elements in a design. Hierarchy refers to the amount of layers of modules and elements in a structured design.

III. DESIGN LANGUAGE AND EVALUATION

A. Design Models

The goal of this research is to apply an evolutionary system for the automatic generation of 3D models. We wish for the creation of diverse architectural forms, which are compelling, novel and inspirational. The creation of these models will be directed by both the design operators that are available to the system and the types of fitness functions that will guide the evolution of those models.

The architectural designs generated by the system will be both fully connected and watertight. A fully connected model is one that does not have any disjoint components. Watertight is a term used to describe if a model is suitable for 3D printing. This means that there are no holes or missing faces which expose the interior of the model. These two properties ensure that the model has some structural integrity and will allow for the models to be rapidly prototyped with a 3D printer – an important feature when attempting to create form for inspirational purpose. Typically, the generation of 3D form in evolutionary design is done using either only surface-based techniques or in a voxelized space. In this research, we wish to generate polygonal models with minimal constraints.

TABLE II
DESIGN OPERATORS

Operator	**Description**
Extrude	Extrudes the currently active face along its normal.
Left	Changes the currently active face to its neighbour on the left.
Right	Changes the currently active face to its neighbour on the right.
Up	Changes the currently active face to the neighbour above it.
Down	Changes the currently active face to the neighbour below it.
Rotate	Rotates the currently active face.
Scale	Scales the currently active face.
[,]	Push and Pop the currently active face.

The benefits of using polygonal form include being able to evolve more novel models with many unique surface-normals, removing the need for post-processing steps which can bias the emerging design process.

Examining a non-voxelized space presents extra difficulties. A lack of strict boundaries means that a system can evolve a model which is large and time consuming to evaluate. Also, the evolved models can generate embedded structures. This can potentially create non-visible geometry within the design.

B. Design Language

The design languages are composed of a seed object and a set of design operators. The seed object is the starting geometry with which the system can begin to alter. The design operators are applied to the structure to refactor its form.

A parametric language is used in this research, allowing for parameters to be passed to design operators. It is based on a LOGO-style turtle drawing system, with extensions for rotation and scaling. The seed object is a plane, and the operators are made up of extrude, rotate and scale operators which can be applied to the currently focused polygonal face. It is also possible to change the currently focused polygon with a set of change-face operators. Table II provides a list of the operators used and [26] contains further details on design languages.

C. Fitness objective

Multiple kinds of fitness objectives are used. Some of them evaluate basic geometric characteristics of a model. For example, *surface area* attempts to minimize or maximize the overall polygonal surface area of a model, or match it's surface are to a target value. Similarly, *polygonal count* considers the number of polygons composing a model.

The **form filling** fitness objective is designed to guide the generation of design models towards some rough form. Form filling works by inputting a set of 3D coordinate *targets* into the system and having the generated geometry attempt to fill towards those target positions. The use of form filling is important, as it is likely the system would only generate very simple models - single cubes for example - to achieve the goals of other architecturally related evaluation functions. The technique works by calculating distances between each polygon within the model and the targets. The algorithm matches each polygon to its closest target, adding the distance between the target and the polygon's furthest vertex to that targets score. As evolution progresses, each polygon is drawn in towards it closest target. This results in a set of faces which roughly approximate a set of targets.

The **Sun exposure** fitness function calculates the approximate amount of Sun exposure the model is receiving. Each polygon in the model is divided into a grid of N by N points. A ray casting algorithm, as used in raytracing systems, is implemented [29]. A ray is cast from the sun to a grid point, and the relative intensity of exposure to the Sun (angle of incidence) is measured. Furthermore, solar exposure obstructions with other components of the model are determined with ray collisions. The overall exposure score can be calculated using one-to-many sun vectors, which allows the solar exposure to be averaged over a day. It can also be minimized or maximized, in the case of summer or winter. Finally, external objects can be added to cast shadows onto the design model and ray casting will determine (via ray collisions) if portions of the model are exposed to the Sun.

D. Multi-objective evaluation

Our modeling problems are instances of multi-objective problems (MOP), in which multiple objective must be simultaneously optimized [30]. We use the normalized summed rank (or average rank) MOP scoring scheme [31], [32]. It is effective for low- and high-dimension MOP problems. It benefits over the more common Pareto ranking in that outlier solutions are unlikely to be produced.

Given a search problem in which each individual i has a feature vector $\overrightarrow{V}^i = (f_1^i, ..., f_k^i)$, where each $f_j^i (j = 1, ..., k)$ is one of the different fitness scores out of the k used in total. For each feature j, all the individuals in the population are assigned an integer rank for each of its features. For example, the individual with the best score for a particular objective gets assigned a rank 1 for that objective, the next best a rank 2, and so on. After all individuals are ranked over all objectives, each individual i is assigned a rank vector $\overrightarrow{R}^i = (r_1^i, ..., r_k^i)$, where $1 \leq r_j \leq N$, for a population of size N. The maximum rank can be less than N, due to tied scores. $\overrightarrow{R}$ is then normalized, by dividing each r_i by the maximum rank found for that objective. The sums of the normalized ranks are used as fitness values (low values are preferred).

E. Diversity

The creation of diverse population sets is extremely important in the domain of evolutionary design. To enforce this, a diversity strategy created by Flack was implemented to penalize potential duplicate individuals [33]. The strategy works by examining the fitness vector of each individual per generation. If two individuals are evaluated to the same fitness scores, they are deemed to be clones of one another and one individual is given a penalty score. It is important to note that it is possible that two individuals could potentially differ as individuals but achieve the same fitness scores. This fact is disregarded when using this strategy, as it is an uncommon event in this problem domain.

TABLE III
GP And Encoding parameters for experiments with generative representations.

GP Parameter	Value
Generations	3000
Population Size	100
Crossover Rate	90%
Tournament Size	4
Elite Count	1
Maximum Crossover Depth	10
Mutation Rate	10%
Maximum Mutation Depth	17
Prob. of Terminals in Cross/Mut	10%
Initialization Method	Half-and-half
Tree Grow Max / Min	5 / 5
Tree Full Max / Min	12 / 5
Experimental Trials	30
GR Parameter	**Value**
Procedure Amount	7
Procedure Arity	2
Condition Pairs	2
Design Language	Growth
Seed Object	Cube (1x1x1 units)

IV. Experiments with Generative Representations

A. Problem Description

The goal of these experiments is to investigate and compare the use of generative representations in a new problem area, with focus on examining their performance in comparison to one another. It will also determine if the space-filling evaluation strategy is valid for generating forms. We also wish to identify the features of complexity that are important for evolutionary 3D structure synthesis. Two sets of experiments are carried out in this section.

The goal of the first experiment is to have the evolutionary system generate an arch whose geometry approximates a set of target points. These points all lie within the same plane, along the $z = 0$ axis, making it a simplified version of the space filling problem. It is a minimization problem, where the system must attempt to reduce the error between the evolved shape and the target point set. Three sizes of the arch problem are considered: small (28x24 units), medium (56x48 units) and large (112x96 units). This will allow for comparison on the scalability of the generative representations if needed.

The random targets problem is designed as a difficult evolutionary design problem. The goal is to generate a form whose geometry approximates a set of N random points. Four sets of experiments are undertaken using 5, 10, 20 and 50 target points. Random points are used instead of a preconceived construct to gain a better understanding of the style of design objects that are created using these techniques. Another reason for this is to eliminate any potential bias that could be present by selecting an object that would be better suited for a particular representation. Parameters are shown in Table III.

TABLE IV

AVERAGE AND BEST FITNESS FROM THE FIVE REPRESENTATIONS ON THE SMALL, MEDIUM & LARGE ARCH PROBLEM OVER 30 TRIALS. (BEST SCORES IN BOLDFACE.)

Rep	**Small**		**Medium**		**Large**	
	Avg	Best	Avg	Best	Avg	Best
Simple	45.0	18.8	111.5	72.5	292.6	231.3
M	30.2	9.2	60.4	21.4	150.6	58.3
MR	28.8	12.1	72.5	31.5	145.4	61.8
MH	**23.8**	**8.5**	**53.4**	**19.1**	**112.8**	66.5
MRH	47.0	11.4	91.2	26.0	188.4	**52.1**

TABLE V

AVERAGE AND BEST FITNESS FROM THE 5 REPRESENTATIONS ON THE 5, 10, 20 AND 50 RANDOM POINT PROBLEM OVER 30 TRIALS. (BEST SCORES IN BOLDFACE.)

Rep	**5 Point**		**10 Point**		**20 Point**		**50 Point**	
	Avg	Best	Avg	Best	Avg	Best	Avg	Best
Simple	161	147	267	236	483	429	1165	1026
M	103	77.2	169	130	290	215	592	431
MR	112	92.9	186	150	299	242	595	473
MH	**94**	**70**	**164**	**128**	**279**	216	640	497
MRH	109	73	182	133	292	**210**	**586**	**409**

B. Arch Experiment Results

Table IV displays the average of mean and average of best scores of the final population from the fifteen experiments, taken over thirty trials. Here we see that all five representations perform relatively close in the small arch problem. In the medium arch problem, the simple representation begins to diverge from the four generative representations. In the large arch problem, the simple representation garners approximately double the score of the other four representations. MH performs the best overall in all categories except one, where MRH outperformed it in best scores on the large arch problem.

Equipped with a very simple design language, the evolutionary system was successful in automatically generating form that resembled the target point set. The designs that were created using the system were generated as single, water-tight objects. Generated arches came in a wide variety of styles, as is shown in Figure 1, validating its usefulness in design problems as an inspiration tool. The four generative representations did reasonably well in discovering an arch for all sizes of the arch problem, the simple representation was not able to do as well in larger versions of the problem. By adding more objectives to the form-filling problem, we were able to generate more specific designs, showing great promise in what is possible with this technology.

C. Random Target Experiment Results

Table V contains the final population results from the four experiments. It can be seen that the simple representation performs the worst in each field and is not able to scale effectively to larger problems. MH performs best overall in the five and ten point experiments, with the M and MRH representations performing very closely. In the twenty point experiment, MRH takes over as the leader, with both M and MH doing nearly as well. In the fifty point experiment MRH clearly outperforms all others, showing its ability to scale well to larger and more complex problems. A one-tailed t-test was performed on the results from the fifty point experiment and it was seen that the MRH outperformed all other representations, and each representation outperformed the Simple representation with a 95% confidence.

Samples of the best individuals from the ten point experiment are shown in Figure 2. From this image, we can see that the simple representation is not able to handle this problem, generating stick-like design objects rather than voluminous ones. While MH appears to achieve high fitness scores, it does so by creating large container structures to envelope all target points, failing to provide geometrical detail. The M and MR representations are able to generate interesting results, though they do not contain the geometric fidelity of the MRH representation. The models created by the MRH encoding are very compelling, creating sub-structures and detailed geometry that are appealing to the eye.

Figure 3 displays measures of modularity of both the program and design encoding, reuse, and hierarchy from the fifty point experiment. Each scatter diagram contains nodes from each individual of the final population from the 30 MRH trials, totalling 3000 individuals.

Modularity scores are calculated for the design encoding. There is a cluster of strong performing individuals with a high $modularity$ score. Measures of *reuse* show a pocket of very strong individuals sitting at around a reuse of 40 similar to that of the $modularity$ metric. *Hierarchy* of all strong individuals is in the range of 5 to 7.

It is important to note that these measurements are taken on the final population of the MRH experiment. Typically, the final population will contain a set of relatively good solutions. It appears in these experiments that the measures of *reuse* and *modularity* were the most useful. They clearly show that the best individuals in the population were enabled by these features of complexity. *Hierarchy* displays interesting results. In this cases, seven was the highest score that could be achieved for this experiment, and the populations tended towards higher values of *Hierarchy*. The measurements explored here, help to verify that modularity, reuse, and hierarchy are important features in encoding individuals for evolutionary design.

V. EXPERIMENTS WITH PASSIVE SOLAR PERFORMANCE

A. Problem Description

In the previous section, we have shown that the MRH representation performs well in this problem domain, and so we will only be using that representation in this section. Here, we will examine the generation of forms that are optimized for passive solar performance.

In passive solar building design, the building is made to collect solar energy in the form of heat during the winter and reject it during the summer [25]. A set of problems which examine the Sun in various ways were designed. The generated designs are influenced by multiple objectives, these being the Sun exposure and form filling fitness functions. A series of Sun vectors were created to simulate the change of location

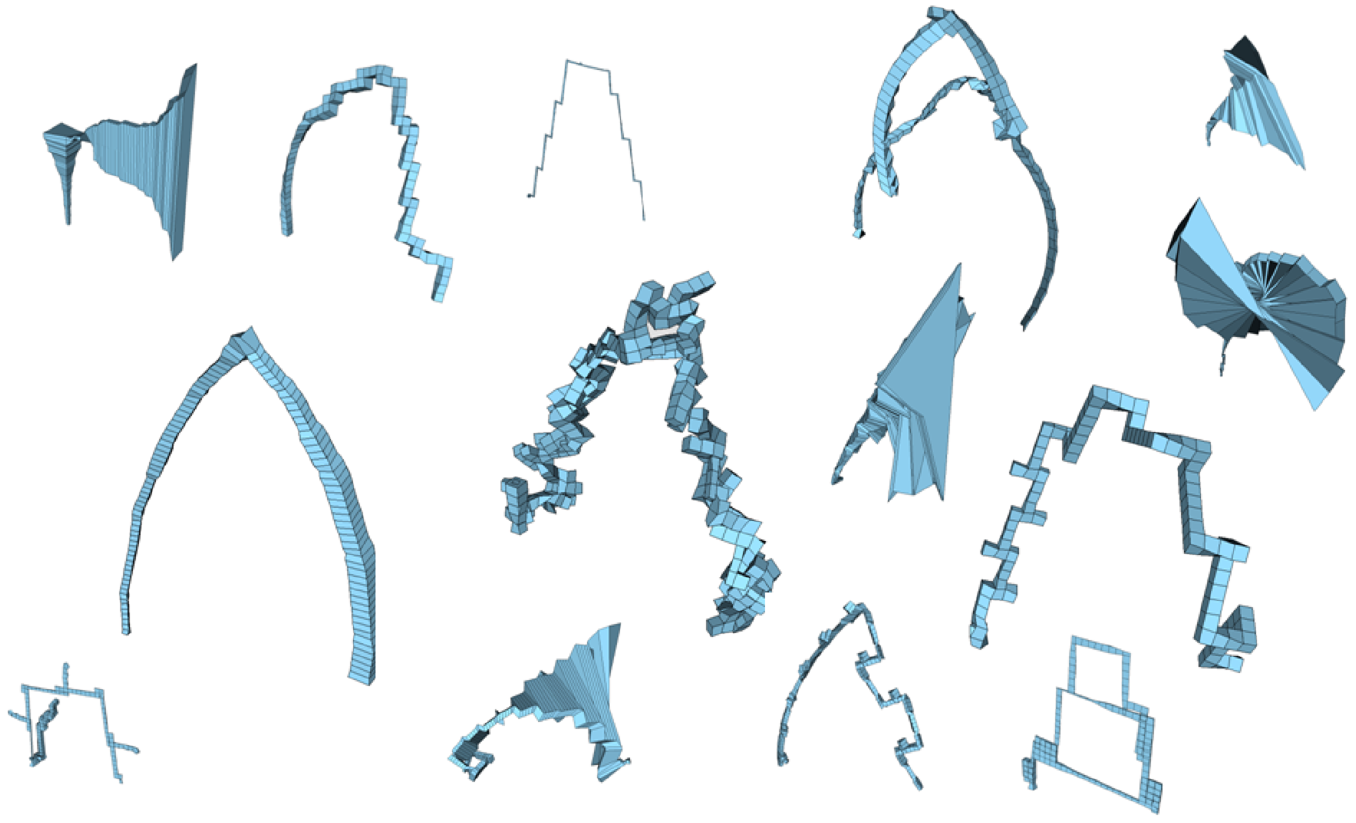

Fig. 1. Variety of arch designs.

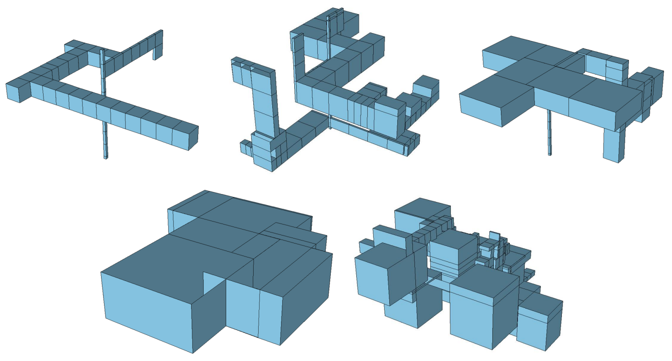

Fig. 2. Images of best results from the S, M, MR (top: left to right), MH and MRH (bottom: left and right) representations in the *Ten* random target problem.

in the sky over the course of a day in Toronto, Canada, during the summer and winter solstices.

In this section, we performed three experiments related so Solar performance. The first experiment will generate models that maximize exposure to the Sun, averaging exposure based on multiple Sun positions. This experiment is based on two objectives, the ten-point random target form filling objective and the winter Sun exposure objective, in which scores are to be maximized.

The second experiment extends upon the first. In this experiment we add three extra structures externally, blocking sunlight to the design models. The goal is that the evolved structures grow around the blocking models, to maximize exposure to sunlight.

The final experiment examines passive solar performance. In this experiment we create a set of target points which represent a "skyscraper". Three objectives are evaluated, the first is a form filling objective using the target points mentioned previously. The second is the winter Sun objective, in which the Sun exposure is to be maximized. The third objective is the summer Sun objective, in which the Sun exposure is to be minimized.

B. Results

1) Winter Sun: Experiments were performed taking into account the movement of the Sun through the sky during the course of a day. Images of a strong and poor result are seen in Figure 4. The Sun is shining from the bottom-left direction in the figure. The images display the average intensity of light from all Sun positions using a greyscale, where white indicates high exposure and black indicates no exposure. The strong

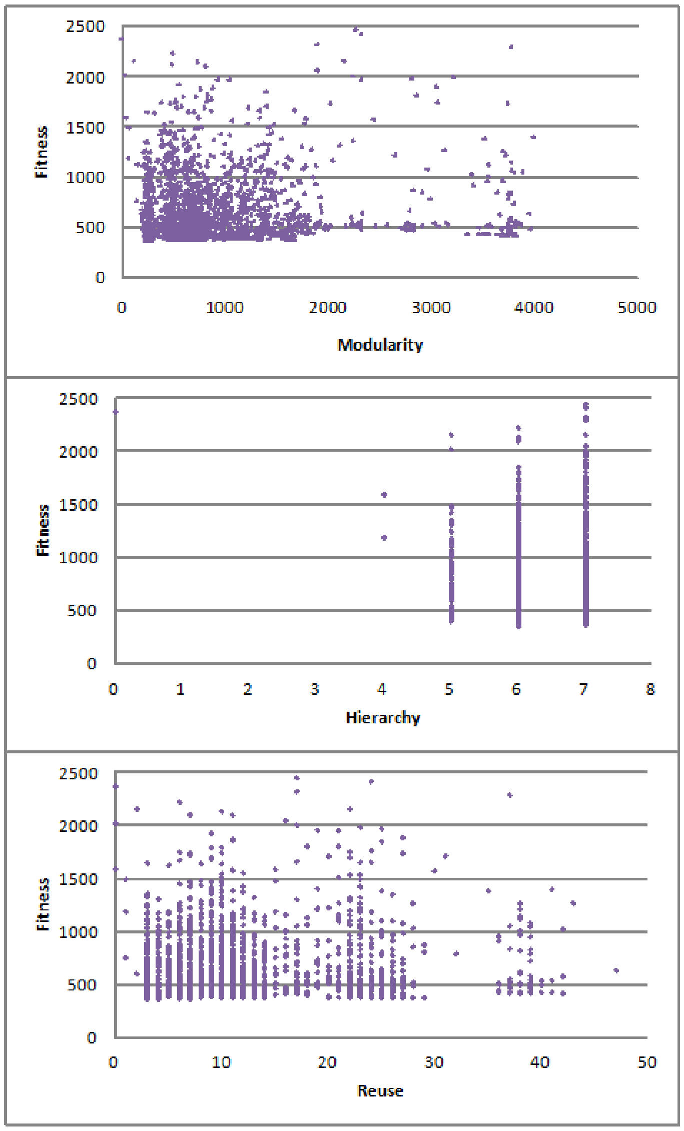

Fig. 3. Measured values of modularity, reuse, and hierarchy for MRH individuals, from the fifty target point experiment.

individual is quite interesting, in that it contains a number of gaps which allow more light to get into the interior of the model. These results resemble that of Watanabe's Sun God city, where form was generated using rule-based logic [3]. The individual that performed poorly is composed of larger block structures and its front facing surfaces are taking much of the sunlight away from the rest of the structure.

2) External obstructions: Experiments were executed, using a set of external models to block Sun exposure to the evolved models, to represent other buildings in a city. A surface area minimization objective was added to these experiments, as per observations from the winter Sun experiments. Figure 5 contains images of a top performing result found in these experiments. The left image shows the three volumes (lower-left in the image) which are blocking sun from the

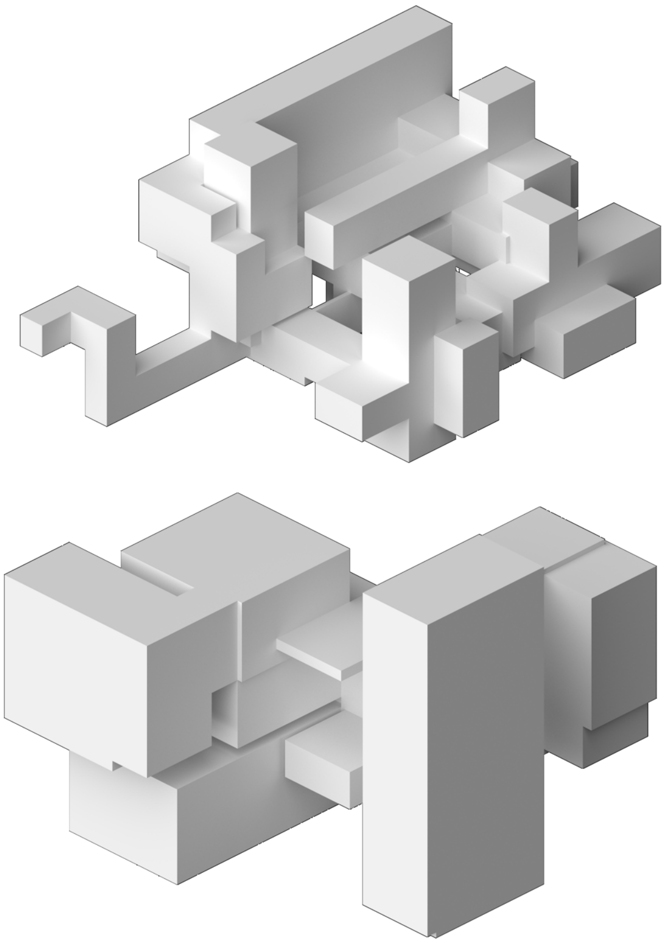

Fig. 4. Example results of a strong (top) and poor (bottom) performing individuals from the winter Sun experiment. The bright/dark shades denote relative solar intensity on surfaces.

evolved model. We can see that the design structure has grown out sideways to capture more light. It has also grown at a distance from the blocking objects and maintains features that resemble a staircase, as seen in the good result from the winter experiment in Figure 4.

3) Passive-solar tower: Experiments were performed to generate form using target points representing skyscraper structures. We found that the form fitting scores were all within the same range as those from the experiments in the previous section, proving that the system can scale to more objectives. Examples of individuals from the passive solar experiment can be seen in Figure 6. In this figure, cyan is shown where surfaces are more receptive to the winter Sun. Some interesting building structures begin to appear which do resemble skyscrapers. In the passive solar trials, we see overhanging and sloping surfaces, exposing more surface area directly towards the winter Sun.

C. 3D Printing

The system is capable of generating novel design structures, which are able to be created with a 3D printer as seen in

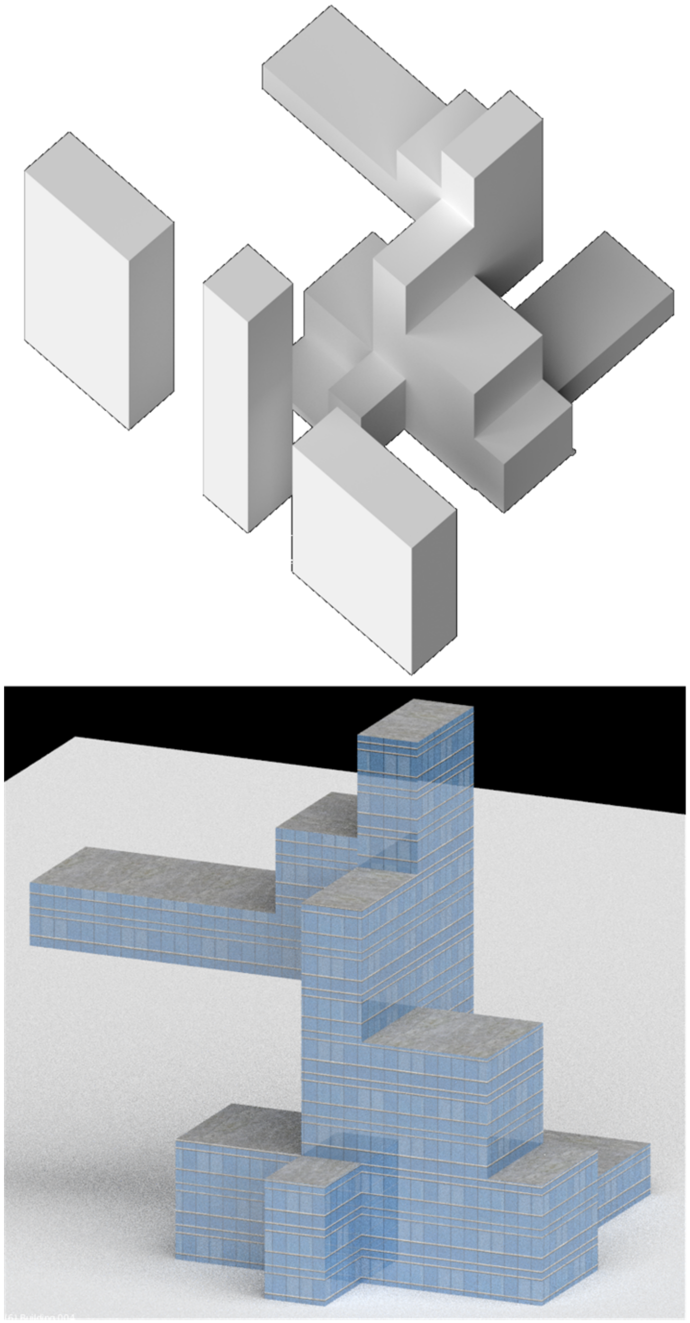
Fig. 5. Generated model from external obstruction experiment with three blocking forms shown (top). Texture and shading applied as a post-processing effect (bottom).

Figure 7. With the growing popularity and affordability of 3D printing, it is possible to create a pipeline in which an evolutionary system is able to automatically generate models and have them rapidly prototyped for inspection as real-world objects. This is a new and exciting feature in the development of inspirational tools and future evolutionary tools should consider this.

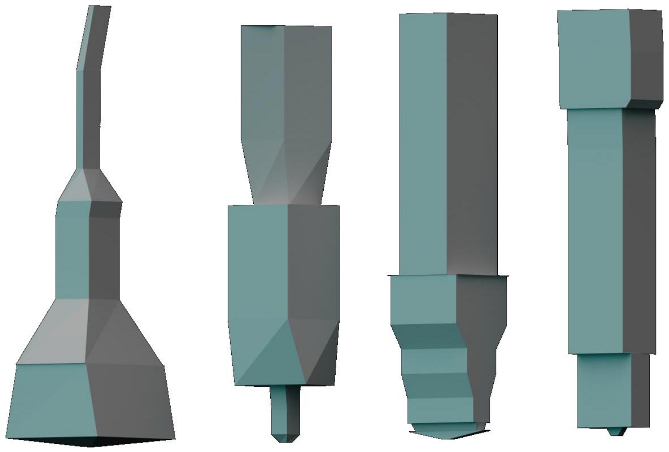
Fig. 6. Example individuals from the passive solar trials. Cyan is shown where winter Sun exposure is stronger than the summer Sun exposure.

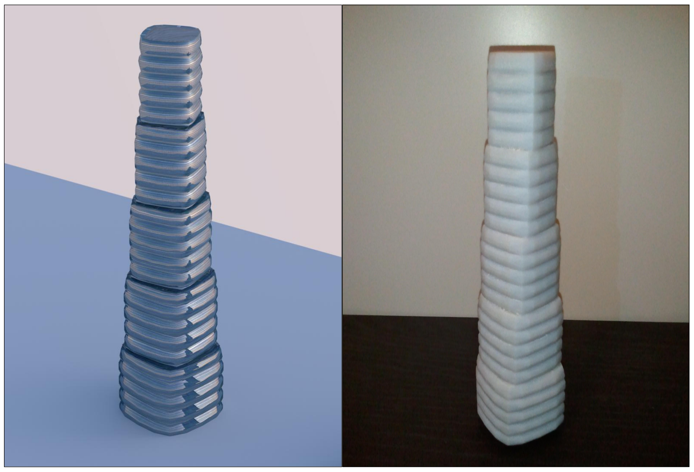
Fig. 7. Generated model from passive solar skyscraper experiment. Smoothed, textured and shaded (left). 3D print (right).

VI. Conclusion

This paper shows that the classic GP encoding is not sufficiently scalable in complex design problems, validating the use of generative representations, and the features of modularity, reuse and hierarchy. These results are consistent with those of Hornby [19], [24], although some minor differences arose in the measures of complexity. We found that high values in the complexity metrics did not equate to better fitness, but rather, individuals with good fitness scores tended to have higher scores in the features of complexity. The measurements of modularity, reuse and hierarchy were very useful in highlighting how the features of complexity enabled higher quality solutions.

We also showed that the form filling fitness function is an effective method for guiding an ED system to create geometry. Using the parametric growth language, a variety of models were evolved. These rough forms guided by random target points could be refined using architecturally related fitness evaluations. The passive solar performance criteria showed that the proposed system generates building models that are applicable to an assortment of architectural problems. It was also shown that passive solar performance did not negatively impact the form filling scores, proving that the system can scale to handle more objectives.

The system is also capable of generating models that can be processed by 3D printers. There are a number of ways in which the methods proposed here could be enhanced to generate even better results. To enhance this research, one could improve on the generative representations, design languages, evaluation functions, or perform continued experimentation. With more refined design languages and evaluation measures, we will be able to provide human-competitive results in artificial architecture.

Acknowledgements: This research was supported by funding from Brock University, OGS and NSERC Discovery Grant 138467. Thanks to Steven Bergen, Beatrice Ombuki-Berman, and Sheridan Houghten for their advice and assistance. Also thanks to Javelin Technologies for access to their 3D printer.

References

[1] K. Terzidis, *Algorithmic Architecture*. Architectural Press, 2006, vol. 1.

[2] P. von Buelow, *Genetically Engineered Architecture - Design Exploration with Evolutionary Computation*. Saarbrücken, Germany: VDM Verlag, 2007.

[3] M. Watanabe, *Induction Design: A Method for Evolutionary Design*. Birkhäuser, 2002.

[4] G. Stiny, "Introduction to shape and shape grammars," *Environment and Planning B: Planning and Design*, vol. 7, no. 3, pp. 343–351, May 1980.

[5] J. Halatsch, A. Kunze, and G. Schmitt, "Using shape grammars for master planning," *Proc. Design Computing and Cognition 08*, pp. 655–673, 2008.

[6] Y. I. H. Parish and P. Müller, "Procedural modeling of cities," in *ProceedingsSIGGRAPH '01*. New York, NY: ACM, 2001, pp. 301–308.

[7] P. Müller, P. Wonka, S. Haegler, A. Ulmer, and L. Van Gool, "Procedural modeling of buildings," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 614–623, Jul. 2006.

[8] E. Whiting, J. Ochsendorf, and F. Durand, "Procedural modeling of structurally-sound masonry buildings," *ACM Trans. Graph.*, vol. 28, no. 5, pp. 112:1–112:9, Dec. 2009.

[9] M. Lipp, P. Wonka, and M. Wimmer, "Interactive visual editing of grammars for procedural architecture," in *ACM SIGGRAPH 2008 papers*, 2008, pp. 102:1–102:10.

[10] D. W. Corne and P. J. Bentley, *Creative Evolutionary Systems (The Morgan Kaufmann Series in Artificial Intelligence)*, 1st ed. Morgan Kaufmann, Jul. 2001.

[11] U.-M. O'Reilly and M. Hemberg, "Integrating generative growth and evolutionary computation for form exploration," *Genetic Programming and Evolvable Machines*, vol. 8, no. 2, pp. 163–186, June 2007, special issue on developmental systems.

[12] M. Hemberg, U.-M. O'Reilly, A. Menges, K. Jonas, M. Gonçalves, and S. Fuchs, "Genr8: Architects' experience with an emergent design tool," in *The Art of Artificial Evolution*, 2008, pp. 167–188.

[13] J. Frazer, *An evolutionary architecture*. Architectural Association, 1995.

[14] J. Frazer, J. Frazer, L. Xiyu, T. Mingxi, and P. Janssen, "Generative and Evolutionary Techniques for Building Envelope Design," 2002.

[15] M. ONeill, J. McDermott, J. M. Swafford, J. Byrne, E. Hemberg, A. Brabazon, E. Shotton, C. McNally, and M. Hemberg, "Evolutionary design using grammatical evolution and shape grammars: Designing a shelter," *International Journal of Design Engineering*, vol. 3, no. 1, pp. 4–24, 2010.

[16] J. McDermott, J. M. Swafford, M. Hemberg, J. Byrne, E. Hemberg, M. Fenton, C. McNally, E. Shotton, and M. ONeill, "String-rewriting grammars for evolutionary architectural design," *Environment and Planning B: Planning and Design*, vol. 39, no. 4, pp. 713–731, 2012. [Online]. Available: http://EconPapers.repec.org/RePEc:pio:envirb:v:39:y:2012:i:4:p:713-731

[17] C. Coia and B. J. Ross, "Automatic evolution of conceptual building architectures," in *IEEE Congress on Evolutionary Computation*, New Orleans, LA, USA, 2011, pp. 1140–1147.

[18] R. Kicinger, T. Arciszewski, and K. D. Jong, "Evolutionary computation and structural design: A survey of the state-of-the-art," *Comput. Struct.*, vol. 83, no. 23-24, pp. 1943–1978, Sep. 2005.

[19] G. S. Hornby, "Generative representations for evolutionary design automation," Ph.D. dissertation, Brandeis University, USA, 2003.

[20] A. Lindenmayer, "Mathematical models for cellular interaction in development: Parts i and ii." *Journal of Theoretical Biology*, vol. 18, 1968.

[21] S. Bergen, "Automatic structure generation using genetic programming and fractal geometry," Master's thesis, Brock University, 2011.

[22] C. Jacob, *Illustrating Evolutionary Computation with Mathematica*. Morgan Kaufmann, 2001.

[23] G. S. Hornby, "Functional Scalability through Generative Representations: the Evolution of Table Designs," *Environment and Planning B: Planning and Design*, vol. 31, no. 4, pp. 569–587, Jul. 2004.

[24] ——, "Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design," in *GECCO '05*. New York, NY, USA: ACM, 2005, pp. 1729–1736.

[25] B. Anderson and M. Wells, *Passive solar energy: the homeowner's guide to natural heating and cooling*. Brick House Pub. Co., 1981.

[26] A. Harrington, "Enabling and measuring complexity in evolving designs using generative representations for artificial architecture," Master's thesis, Brock University, 2012.

[27] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA, USA: MIT Press, 1992.

[28] S. Luke, "ECJ - a java-based evolutionary computation research system," in *http://cs.gmu.edu/ eclab/projects/ecj/*.

[29] A. Glassner, *An Introduction to Ray Tracing*. Academic Press, 1989.

[30] C. A. C. Coello, G. B. Lamont, and D. A. V. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[31] P. J. Bentley and J. P. Wakefield, "Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms," in *Soft Computing in Engineering Design and Manufacturing*, P. K. Chawdhry, R. Roy, and R. K. Pant, Eds. Springer-Verlag, Jan. 1998, pp. 231–240.

[32] D. Corne and J. Knowles, "Techniques for highly multiobjective optimisation: Some nondominated points are better than others," in *Proceedings GECCO 2007*. ACM Press, 2007, pp. 773–780.

[33] R. W. J. Flack and B. J. Ross, "Evolution of architectural floor plans," in *EvoApplications (2)*, 2011, pp. 313–322.