A Local Search Interface for Interactive Evolutionary Architectural Design

Jonathan Byrne, Erik Hemberg, Anthony Brabazon, and Michael O'Neill

Natural Computing Research & Applications Group,
University College Dublin, Ireland
{jonathanbyrn,erikhemberg}@gmail.com,
{anthony.brabazon,m.oneill}@ucd.ie

Abstract. A designer should be able to express their intentions with a design tool. This paper describes an evolutionary design tool that enables the architect to directly interact with the encoding of designs they find aesthetically pleasing. Broadening interaction beyond simple evaluation increases the amount of feedback and bias a user can apply to the search. Increased feedback will have the effect of directing the algorithm to more fruitful areas of the search space. We conduct user trials on an interface for making localised changes to an individual and evaluate if it is capable of directing search. Examination of the locality of changes made by the users provides an insight into how they explore the search space.

1 Introduction

Interaction was introduced to Evolutionary Algorithms (EA) for problems where no objective fitness function could be found. This allowed EAs to tackle problems that were aesthetic in nature. Traditional interactive evolutionary computation (IEC) limited the users input to that of a fitness function, evaluation. Takagi [23] defined this type of IEC as *"narrowly defined"* IEC (NIEC). Limiting users to evaluation creates a bottleneck for the evolutionary algorithm. There is an additional burden on the algorithm to intuit what the user actually desired from their selections. We address the problem in this paper by introducing a local search interface that enables users to focus on a particular area of the search space.

Grammatical Evolution (GE) is an EA that uses an integer string, a *genotype*, to pick rules from a grammar and generate an output, a *phenotype*. The grammar based approach is capable of generating complex output and so is suitable for generating architectural designs. The shape grammar used in this paper focuses on the architectural domain of foot bridge designs. Our approach allows for *active user intervention* [11] by providing an interface for directly manipulating the genotypic encoding. By mutating the genotype the designer can change the phenotype, combining the generative output of the algorithm with the intention of the designer. Instead of cosmetic changes being made to the output, the changes made by the designer are also reflected in the individual's genotypic encoding.

P. Machado, J. Romero, and A. Carballal (Eds.): EvoMUSART 2012, LNCS 7247, pp. 23–34, 2012.

As mutation is applied to the genotype and not the phenotype, single mutation events combined with the mapping process can result in different magnitudes of change to the phenotype, i.e., mutation events have different *locality*. Locality is a measure of how a small change in the input corresponds to the change in the output. Previous work [4] examined the locality of mutation events in GE when using a shape generating grammar. We shall examine the locality of user selections to explore how they use mutations of different locality to navigate the search space.

Our work introduces a novel interface that allows the user to perform a large number of mutation evaluations and apply mutation operations. Enabling the user to perform single mutation events allows them to perform a local search on a particular individual. A local search iteratively makes small changes to move between neighbouring solutions. When a user finds an aesthetically pleasing design they generally want to explore that area of the search space. Our experiments test if the user is capable of directing search using our local search interface.

This paper is organised as follows. Section 2 discusses related research in computer generated design and active user intervention and gives a description of GE and our previous interface. Our experimental setup, grammar choice and experimental design are described in Section 3. The results of our experiments are presented and explained in Sections 4. Finally, we discuss our conclusions and future work in Section 5.

2 Background

2.1 Computer Generated Architectural Design

Computers are ubiquitous in architectural design but they are normally used for analysis rather than design generation. In recent years software has been developed that allows the user to explore the search space of possible designs. A direct approach that allows the designer to explore the design search space is to implement a parametric system. The user inputs their design and then modifies individual components of that design. EIFForm [19] was a successful approach to implementing parametric design and the results have been used to design a structure in the inner courtyard of Schindler house. Parametric design tools have now been introduced into more mainstream design software. There is the Grasshopper plug-in for the Rhino modelling system [20] and Bentley Systems have implemented a program called Generative Components [21].

An evolutionary approach to conceptual design exploration is implemented in GENR8 [17]. This system uses GE and Hemberg Extended Map L-Systems (HEMLS) to generate forms. The user can influence the growth of the L-System through the use of tropism and fitness weighting. Objects can be placed in the environment that either attract or repel the design. Each design is evaluated by a series of metrics; symmetry, undulation, size, smoothness, etc. The user is able to weight these metrics according to their preference. Our approach extends evolutionary design exploration by allowing the user to iterate through the parameter values of an evolved design, thus incorporating aspects of parametric design systems.

2.2 Active User Intervention

Approaches that increase user participation in the evolutionary process are categorised as active user intervention (AUI) [23]. Several successful methodologies have been used to increase user participation. Online knowledge embedding (OLKE) [22] provides a mechanism for accepting hints, ideas or intentions. The user highlights components of a design that they think have high fitness. The genes relating to these components are then fixed, which reduces the search space. OLKE is only possible if each component of the output maps directly to a particular gene.

Visualised IEC (VIEC) collapses a multi-dimensional search space into a 2D representation. The individuals are then mapped to the 2D space and presented to the user. The user is able to observe the distribution and fitness of the population and direct the search to particular parts of the search space, thus combining both evolutionary and human search techniques. VIEC has been shown to dramatically improve convergence [11] [10] but a meaningful mapping from n-D to 2-D space must be performed and the topological relationships must remain intact.

Human based genetic algorithms (HBGA) enable the user to apply low level genetic operators such as mutation, initialisation, selection and crossover to the population [14]. Using humans is useful in problems such as evolving natural language statements, where it is hard to design efficient computational operators. HBGA requires that an individual in the population can be understood by the user and that the operators perform in a manner intuitive to the user. Hyperinteractive evolutionary computation (HIEC) extends HBGA by giving the users access all the genetic operators. HIEC treats the operators as a tool set for the user. Additional operators such as duplicate, delete are available to the users [1]. Our approach is similar to HBGA in that the users choose when and where to apply mutation operators. The difference is that we present the users with the consequence of applying an operator and they select the change they want.

2.3 Grammatical Evolution

To evolve architecture we required a technique to generate evolvable shapes. We used a shape grammar in conjunction with Grammatical Evolution to accomplish this. Grammatical Evolution is an evolutionary algorithm that is a grammar based form of GP [7][16]. It differs from standard GP by replacing the parse-tree based structure of GP with a linear genome. It generates programs by using a list of integers (also called a chromosome) to select rules from the grammar. The rule selections build a derivation tree that represents a program. Any mutation or crossover operators are applied to the linear genome instead of the tree itself.

Standard GE mutation can be divided into two types of events, those that are structural in nature and those that are nodal. *Nodal mutation* changes a single leaf of the derivation tree. The structure of the derivation tree remains

unchanged. A nodal codon encodes for a rule that only has terminal productions. *Structural mutation* changes one or more internal nodes of the derivation tree (and zero or more leaves) and can result in a change to the structure of the derivation tree. A structural codon encodes for a rule with non-terminal productions and zero or more terminal productions.

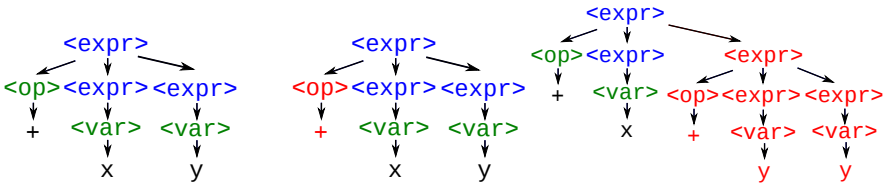


Fig. 1. Nodal (green) and structural (blue) nodes of a derivation tree

Locality refers to how well neighboring genotypes correspond to neighboring phenotypes, i.e., a small change to the genotype results in a small change to the phenotype and vice versa. A mapping has *locality* if the neighbourhood is preserved under that mapping. In EC this generally refers to the mapping from genotype to phenotype [18]. Previous work investigating locality in GE mutation [4, 5, 2] showed that structural and nodal mutation had different effects on locality during the mapping process and on the resulting output. As the user is exclusively applying mutation, we shall investigate how the different locality of events are used to navigate the search space.

2.4 Original Interface Design

The interface plays a vital role in how the user navigates the search space. Our previous work [3] presented the user with a panel of individuals to choose from. The interface enabled them to apply different mutation operators to their preferred individual. The mutation events were categorised based on their locality. "Big" changes and "small" changes equated to structural and nodal events respectively. The user chose the individual and their preferred mutation operator. When an individual was selected, 8 mutated variations of hamming distance one were generated for the user to select from.

The interface had a number of drawbacks. User trials showed that the average number of selections made over a five minute period was 17. As each selection presented the user with 8 more images the user is only presented with 136 designs during the course of a run. To assume that a significant improvement could be made in this short distance was optimistic. The user also had no expectation of the consequences of applying an operator, which meant that the users intention of"big" and "small" changes may not evident in the generated individuals. The smaller changes also presented a unique problem to the user. Some nodal changes fell below the threshold of a Just Noticeable Difference (JND). JND is a concept

from cognitive psychology that was first described by Ernst Heinrich Weber [24]. JND is the smallest difference between two stimuli that is still capable of being perceived. The lack of what the user perceived as new variations also hindered them in completing the task.

3 Experiment

3.1 Interface Implementation

The interface used in this experiment addresses the problems described in Section 2.4. A comparison of the new and old user interfaces are available online [13]. Instead of the user choosing an operator, all possible mutations of hamming distance one were applied to an individual. Each codon was mutated in turn, the result was rendered and then the codon was restored to its original value. The productions for each codon generated a collection of nodal and structural mutation events to choose from. This process is shown in Figure 2. A Euclidean comparison (described in Section 3.3) was performed so that individuals identical to the original were removed from the population, thus reducing the search space presented to the user. By making no assumption about operator choice and instead presenting the user with every possibility, we can now examine how they navigate the search space based on their selections.

Presenting an entire population of mutation events to the user simultaneously is not feasible. Our interface instead uses a single window for exploring the population. The interface is shown in online at [13]. The current user selection is on the left and the target is the image on the right. The leftmost panel states the instructions, user controls, time remaining and the distance from the target. The user scans through the mutation events using the left and right arrow keys and the selects the mutation change they want and that now becomes the basis for generating the next population. The refresh rate for the window was ten frames per second. As the frame rate is below that of persistence of vision the user is capable of perceiving the bridges distinctly. Codon changes were made sequentially so a codon's productions are grouped in their presentation to the user. Overlaying groups of changes in the same window allowed the user to pick up smaller JND changes by viewing them in rapid succession.

3.2 Design Grammar

The grammar was originally conceived based on a brief provided to third year students in the UCD architecture and structural engineering course of 2010. The brief specified that the bridge was to be composed of timber, had an optional arch, a width of 2 metres and bridge a span of 10 metres. The size of the grammar meant that it could not be appended to the paper. The grammar creates graphs using networkx [9], a Python class for studying complex graphs and networks. Three desirable characteristics for a design generator are modularity, regularity and hierarchy [12]. We implement these characteristics using the novel method of higher order functions. See McDermott et al. [15] for a more detailed discussion on grammars based on higher order functions .

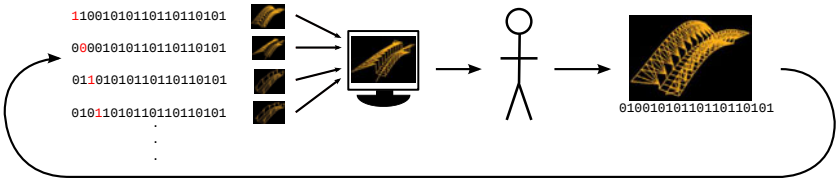


Fig. 2. Generating hamming distance 1 individuals for user selection

3.3 Euclidean Graph Distance

To analyse the participants behaviour we used a Euclidean graph comparison to check for isomorphism between the target with the user's selections. The Euclidean distance is calculated using the same information presented to the user, i.e., a bridge output. Euclidean distance is defined as the straight-line distance between two points on the same plane. An approach was developed for calculating the Euclidean distance between bridge designs. The bridge output is essentially a graph where each node has a cartesian coordinate attribute. When performing a comparison, the graph with the most nodes is selected for iteration. This step ensures the symmetry condition for a metric, $d(x, y) = d(y, x)$. Each node in the larger graph is then iterated through and the nearest node in the smaller graph is found. Exhaustively exploring the smaller graph fulfills the triangle inequality, $d(x, z) \leq d(x, y) + d(y, z)$, as the global minimum is returned. The distance between these nodes is then added to the total distance between designs, thus ensuring non-negativity, $d(x, y) \geq 0$.

Our bridge designs consist of points and the edges between them but the Euclidean distance formula only compares points in space. The edges must be taken into consideration if we are to fulfill the "identity of indiscernibles" condition, $d(x, y) = 0$ if and only if $x = y$. Exhaustively checking for subgraph isomorphism between the graphs is an NP Complete problem [6] and so it is not a feasible approach. Instead, the number of edges connected to the nodes is compared and the difference is added to the distance. This simplified approach could theoretically allow for the distance to be 0 when $x \neq y$, therefore Euclidean graph distance does not satisfy all the conditions of a metric. The use of higher order functions to connect the nodes however, means that if two graphs consist of identical points and that each point has the same amount of node edges than invariably the graphs would be isomorphic.

3.4 Experiment Design

The subjective nature of aesthetics makes evolutionary design search a difficult area to quantify. To generate measurable results we specify a target design that the user must attempt to match, instead of the user searching for a design that they like. The experiment asked the user to match ten different targets. The random seed was fixed so that the interface always started from the same

individual. The targets were mutated variants of the starting individual and they increased in difficulty as the Hamming distance was greater for each successive target. The Hamming distance, the number of nodal and structural mutations and the Euclidean graph distance from the starting graph for each target is shown in Table 1. Each participant had two minutes in which to try to match the target. They were free to make as many selections within this time frame and they could undo selections if they wished. At the end of target exercise the user was asked a short survey. 25 volunteers participated in this experiment and the experiment itself was approved by the Ethics Committee in UCD.

A sample of random trials were generated to examine if the users were capable of using the interface to match the target. The same setup was used except individuals were chosen randomly. 25 random samples were generated for each of the targets. The distribution of the random selections for each target matched the user's click average and standard deviation, as shown in Table 1. Given a sample of randomly selected individuals, μ_0 and a sample of user selected of individuals, μ_1 the following hypothesis is stated. $[H_0]$ There will be no statistically significant difference between the samples i.e., $\mu_0 = \mu_1$ $[H_1]$ There is a statistically significant decrease in distance from the target, i.e., $\mu_0 \neq \mu_1$. The significance (α) level of the Wilcoxon rank-sum is 0.01.

Table 1. The distance and change types for each target

Target	Hamming Distance	Nodal: Structural	Euclidean Distance	User Clicks	User Evaluations	User Matched	Random Matched
Target 1	1	1:0	10	1.0 ± 0.0	22.4 ± 34.9	100%	4.0 %
Target 2	3	2:1	92	3.8 ± 2.1	190.3 ± 80.6	64.6%	11.8%
Target 3	3	2:2	184	2.7 ± 1.3	202.4 ± 61.6	64.9%	12.1%
Target 4	5	4:1	158	4.1 ± 2.2	245.2 ± 126.4	74.1%	20.3%
Target 5	6	5:1	214	3.3 ± 1.1	325.6 ± 130.2	36.6%	31.8%
Target 6	7	7:0	188	3.5 ± 1.3	196.0 ± 61.4	58.9%	31.3%
Target 7	8	6:2	465	3.5 ± 1.2	262.3 ± 83.2	57.6%	15.9%
Target 8	8	7:1	457	4.0 ± 1.9	364.7 ± 131.0	37.5%	27.0%
Target 9	9	7:2	273	4.8 ± 2.0	306.4 ± 86.1	28.2%	36.4%
Target 10	11	7:4	117	4.9 ± 2.0	282.4 ± 103.7	29.7%	42.5%

4 Results

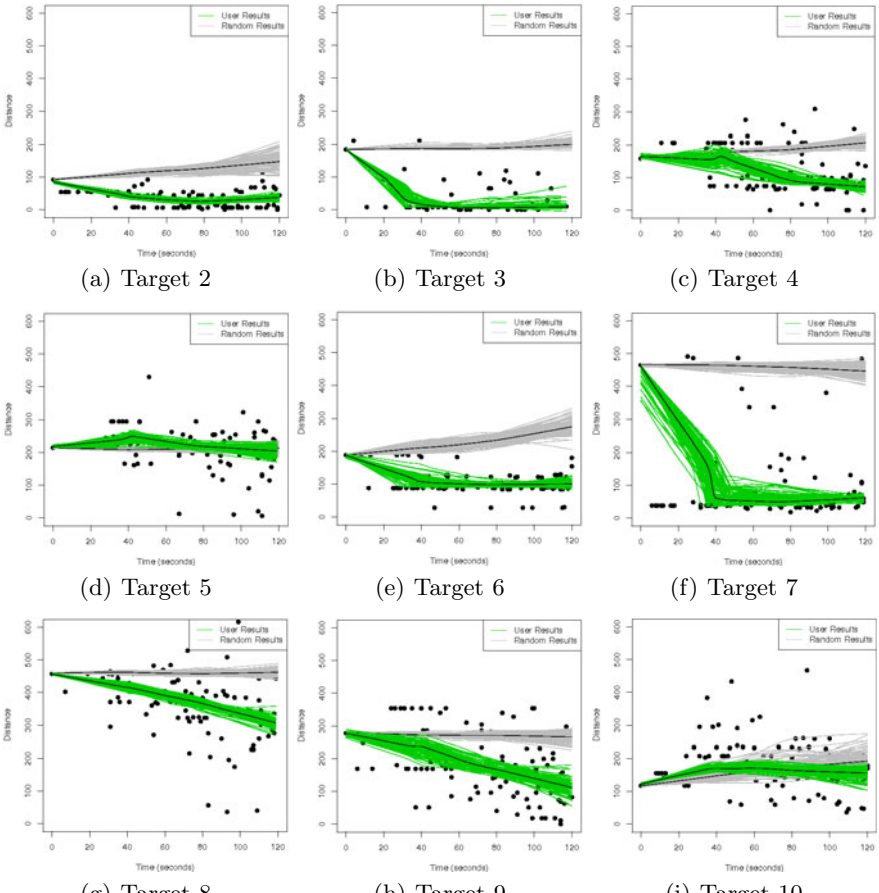
All plots are available online in a larger format at [13]. The number of user selections (user clicks) and the number of images presented to the user (user evaluations) are shown in Table 1. In our previous experiment [3], the users selected 17 individuals and evaluated 136 designs on average in a five minute time period. The user click and user evaluation results show that while the user made fewer selections with the new interface than in the previous experiment they

evaluated many more designs within a two minute time period. The percentage of user mutations (User Matched) and random sampling mutations (Random Matched) that matched the target codon changes are also shown in Figure 1. With some exceptions, the percentage of matched codon mutations decreases as the Hamming distance increases. The opposite is true for random sampling: as more codons are changed there is an increased likelihood that a random mutation would match that of the target.

To examine if this correlates the Euclidean graph distance from the target we generated scatter plots for the data. Each user selection generated a data point that recorded the time, distance from the target and the mutation type. A locally weighted scatterplot smoothing (LOESS) was performed on the results to plot a smooth curve of the average values. The set of data points was then bootstrapped [8]. Bootstrapping is a resampling technique that generates an estimation of the distribution during the course of a run. The LOESS curves for each of the samples were plotted. Figure 3 compares the user average (black) and distribution (green) with the random sampling (grey). As every participant successfully matched the first target within one selection, the results for target 1 are not shown.

A Wilcoxon rank-sum test compared the final selections for the user and the random sampling. The results show that user could successfully use the interface to direct search as there was a statistically significant difference from random with exceptions of target 5 and target 10. The result disproved the null hypothesis for 8 of the 10 targets. Target 10 had a Hamming distance of 11, meaning that a third of the used codons had been changed. As the user only made 4 selections on average it is unlikely the user would be able to match the target. Although the Hamming distance was less for target 5, the participants only matched the target mutated codons 36% of the time resulting in a poor score. A surprising result is that there was a definite improvement in Euclidean graph distance for targets 8 and 9 while the overall percentage of codons matched were low (37.5% and 28.2% respectively). The result implies that it is possible to get close to matching a target without following the exact same path.

Figure 4 shows what type of mutational change was made for each selection. The x-axis is the number of selections made while the y-axis shows frequency for a particular type of mutation. The histograms show that users started by predominantly applying a structural mutation and then moving to nodal mutation. The result means that the participants commenced their search by making large phenotypic changes and then fine tuning that solution with high locality mutation events. The instances that ran contrary to this were target 2 and target 6. Target 2 consisted of 2 nodal changes and 1 structural change. As the Euclidean distance from the starting point was only 92, it could be that both nodal and structural changes had comparable locality. Target 6 consisted of only nodal mutations and so it would follow that only nodal changes were required to match the target.

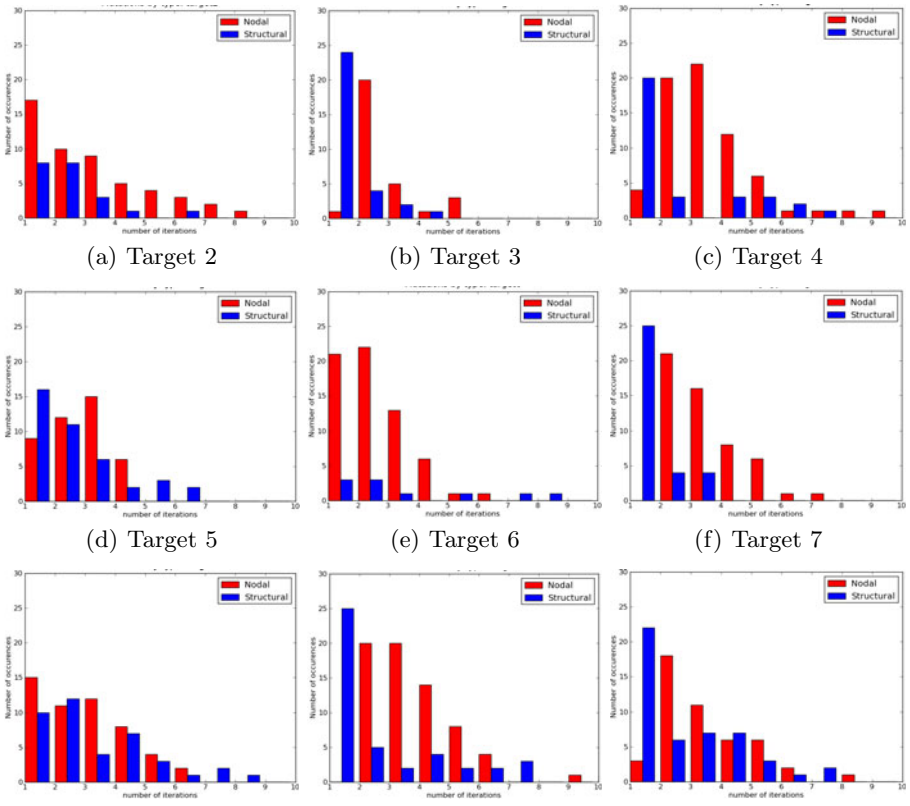


(a) Target 2 (b) Target 3 (c) Target 4

(d) Target 5 (e) Target 6 (f) Target 7

(g) Target 8 (h) Target 9 (i) Target 10

Fig. 3. Loess and bootstrapping results for user trials. The user results are shown in green and the random sampling shown in grey.



(a) Target 2 (b) Target 3 (c) Target 4

(d) Target 5 (e) Target 6 (f) Target 7

(g) Target 8 (h) Target 9 (i) Target 10

Fig. 4. This graph shows the types of mutation users select during the course of a run. The x-axis is the order of the selection made by the user and the y-axis is the cumulative frequency.

5 Conclusions and Future Work

In this paper we presented a novel interface that enabled the user to perform a local search on an individual. Our experiments showed that the user was able to use the interface to match a target individual by directly manipulating the genotypic representation. This result supports that our interface can perform user directed local search toward a desired individual. Examining the user generated results showed that they moved from low locality operators to high locality operators to both explore and exploit the search space. As the user changes are made to the genotypic representation, the new individual can be reintroduced into the population and evolutionary algorithm can continue. Our future work intends to explore the additional benefits of combining AUI with evaluation for evolutionary architectural design.

Acknowledgments. We would like to thank Andrea McMahon for her unceasing support. This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/IN.1/I1868 and 08/RFP/CMS1115.

References

1. Bush, B., Sayama, H.: Hyperinteractive evolutionary computation. IEEE Transactions on Evolutionary Computation 15(3), 1–10 (2011)
2. Byrne, J., O'Neill, M., Brabazon, A.: Structural and nodal mutation in grammatical evolution. In: Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation, pp. 1881–1882. ACM (2009)
3. Byrne, J., Hemberg, E., O'Neill, M.: Interactive operators for evolutionary architectural design. In: GECCO 2011: Proceedings of the 13th Annual Conference Companion on Genetic and Evolutionary Computation, July 12-16, pp. 43–44. ACM, Dublin (2011)
4. Byrne, J., McDermott, J., López, E.G., O'Neill, M.: Implementing an intuitive mutation operator for interactive evolutionary 3d design. In: IEEE Congress on Evolutionary Computation, pp. 1–7. IEEE (2010)
5. Byrne, J., O'Neill, M., McDermott, J., Brabazon, A.: An Analysis of the Behaviour of Mutation in Grammatical Evolution. In: Esparcia-Alcázar, A.I., Ekárt, A., Silva, S., Dignum, S., Uyar, A.Ş. (eds.) EuroGP 2010. LNCS, vol. 6021, pp. 14–25. Springer, Heidelberg (2010)
6. Cook, S.A.: The complexity of theorem-proving procedures. In: Proceedings of the Third Annual ACM Symposium on Theory of Computing, STOC 1971, pp. 151–158. ACM, New York (1971), http://doi.acm.org/10.1145/800157.805047
7. Dempsey, I., O'Neill, M., Brabazon, A.: Foundations in Grammatical Evolution for Dynamic Environments. Springer (2009)
8. Efron, B., Tibshirani, R.: An introduction to the bootstrap. In: Monographs on Statistics and Applied Probability. Chapman & Hall (1993), http://books.google.ie/books?id=gLlpIUxRntoC
9. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using networkx. In: Proceedings of the 7th Python in Science Conference, Pasadena, CA USA, pp. 11–15 (2008)
10. Hayashida, N., Takagi, H.: Visualized IEC: Interactive evolutionary computation with multidimensional data visualization. IECON-PROCEEDINGS 4, 2738–2743 (2000)
11. Hayashida, N., Takagi, H.: Acceleration of EC convergence with landscape visualization and human intervention. Applied Soft Computing 1, 245–256 (2002)
12. Hornby, G.: Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, pp. 1729–1736. ACM (2005)
13. iecgallery: Online image gallery (2011), http://imgur.com/a/24fP9
14. Kosorukoff, A.: Human based genetic algorithm. In: 2001 IEEE International Conference on Systems, Man, and Cybernetics, vol. 5, pp. 3464–3469. IEEE (2001)
15. McDermott, J., Byrne, J., Swafford, J.M., O'Neill, M., Brabazon, A.: Higher-order functions in aesthetic EC encodings. In: 2010 IEEE World Congress on Computational Intelligence, pp. 2816–2823. IEEE Press, Barcelona (2010)
16. O'Neill, M., Ryan, C.: Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language. Kluwer Academic Publishers (2003)

17. O'Reilly, U.M., Hemberg, M.: Integrating generative growth and evolutionary computation for form exploration. Genetic Programming and Evolvable Machines 8(2), 163–186 (2007); special issue on developmental systems
18. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Physica-Verlag (2006)
19. Shea, K., Aish, R., Gourtovaia, M.: Towards integrated performance-driven generative design tools. Automation in Construction 14(2), 253–264 (2005)
20. Software, R.: Grasshopper, generative modeling (2010), http://www.grasshopper3d.com/
21. Sytems, B.: Generative components, v8i (2011), http://www.bentley.com/getgc/
22. Takagi, H., Kishi, K.: On-line knowledge embedding for an interactive ec-based montage system. In: Third International Conference on Knowledge-Based Intelligent Information Engineering Systems, pp. 280–283. IEEE (1999)
23. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proc. of the IEEE 89(9), 1275–1296 (2001)
24. Weber, E.: De Pulsu, resorptione, auditu et tactu: Annotationes anatomicae et physiologicae. CF Koehler (1834)