Interactive Operators for Evolutionary Architectural Design

Technical Report UCD-CSI-2011-05

Jonathan Byrne, Erik Hemberg and Michael O'Neill

Natural Computing Research & Applications Group
University College Dublin
jonathanbyrn@gmail.com, erik.hemberg@ucd.ie, m.oneill@ucd.ie

April 12, 2011

Abstract

In this paper we explore different techniques that allow the user to direct interactive evolutionary search. 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. This paper examines whether additional feedback from the user can be a benefit to the problem of evolutionary design. We find that the interface between the user and the search space plays a vital role in this process.

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 limited the users input to that of a fitness function, evaluation. The limited number of evaluations a user is capable of means that it forms a bottleneck for the evolutionary algorithm. There is an additional burden on the algorithm to intuit what the user actually desired from their selections. Our approach makes the assumption that when a user finds a design they like that they want to explore that area of the search space. Introducing a bias has been shown to improve evolutionary search [33]. Selectively applying an operator

allows the user to bias the search toward exploring a specific area of the search space, depending on the locality of the operator. To enable the users to do this we allow them to apply mutation operators directly to the individuals. We have developed new operators that interact with an individual with varying degrees of locality [8]. Our approach implements a novel interface for allowing the user to direct the search using mutation and we use both preexisting and newly created metrics to evaluate the results at different phenotypic levels. We will now explore if these can aid an evolutionary design process.

This paper is organised as follows. Related research in computer generated design and interactive evolutionary computation is given in Section 2. A description of Grammatical Evolution and the generative processes in GE is given in Section 3. Our experimental setup, grammar choice and experimental design are described in Section 4. The results are shown and explained in Sections 5, 6 and 7. Finally, we discuss our conclusions and future work in Section 8

2 Related Research

This section discusses previous work using computers for architectural design generation and the applications of interactivity in evolutionary computation.

2.1 Computer Generated Architectural Design

While computers are almost universally applied in architectural design, 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 was a successful attempt at implementing parametric design and the results have been used to design a structure in the inner courtyard of Schindler house [28]. Parametric design tools have now been introduced into more mainstream design software. There is the Grasshopper plug-in for the Rhino modelling system [13] and Bentley Systems have implemented a program called Generative Components [12].

An evolutionary approach to conceptual design exploration is implemented in GENR8 [23]. 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.

2.2 Interactive Evolutionary Computation

Human interaction was originally introduced to the evolutionary process for problems where no objective fitness function could be found. Interactive fitness functions have allowed EC to be applied to problems such as music and computer graphics, and to act as an exploratory tool as opposed to being limited to optimisation. One of the earliest attempts to introduce human evaluation was the work by Richard Dawkins called Biomorphs [10]. This work simulated the evolution of 2-D branching structures made from sets of genetic parameters, where the user selects the individuals for reproduction.

After Biomorphs there was an increase of research in the field of both interactive genetic algorithms (IGA) and interactive genetic programming (IGP). The seminal paper by Karl Sims [29] showed that basic user interaction was capable of creating complex and beautiful artwork. Sims used human interaction to create images, three-dimensional textures, and, by adding an extra dimension for time, animation. IGAs have since been applied to fields as diverse as music generation [5, 20], anthropomorphic symbols [11], 3-D lighting [1], and aircraft frames [25] to name a few. Interactivity has also been used in conjunction with GP for form design. Aesthetic problems have a subjective element in their evaluation that is difficult to define in an algorithmic fitness function. This has led to the development of several form design tools [24, 4]. A more complete list of applications of interactivity can be found in the literature by [3] and [31]. Our approach to interactive evolution differs from the approaches mentioned above. Instead of utilizing user evaluations to direct the evolutionary search our approach allows the user to manipulate the genome directly. As such, it is categorised as active intervention [31].

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 [21]. 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 which are then applied to generate a program. The chromosome is made up of codons. Each codon in the string is used to select a production rule from a Backus Naur Form (BNF) grammar. The BNF represents a language in the form of production rules. Each rule is comprised of non-terminals that map to either terminals, non-terminals or both, depending on the production rules. A simple example BNF grammar that could be used for symbolic regression is given in Figure 1. `<expr>` is the start symbol from which all programs are generated. The grammar states that `<expr>` can be replaced with either one of `<expr><op><expr>` or `<var>`. An `<op>` can become either +, or -, and a `<var>` can become either x, or y.

```
<expr> ::= (<op><expr><expr>)        (0)
         | <var>                     (1)
<op>   ::= +                         (0)
         | -                         (1)
<var>  ::= x                         (0)
         | y                         (1)
```

Figure 1: A simple BNF grammar

The integer codons decide which rule is chosen by simply calculating the modulus of the codon value with the number of rules. This can be represented with the following formula:

$$Rule(idx) = Codon\ Value\ \%\ Num.\ Rules \tag{1}$$

By iterating through the codons the BNF rules are applied and a derivation tree is built. This in turn generates a string from the grammar which represents the program.

3.1 Generative processes in EC

As shown in the previous section, GE uses a mapping process to create output. In GP this is called the genotype to phenotype mapping. The genotype to phenotype mapping is a terminology that has been adapted from the field of biology. A genotype is an encoding upon which the evolutionary operators of mutation and recombination act. An example of this would be human DNA. Changes in the genotype are translated into changes in the phenotype. A phenotype is an observable characteristic of an individual. An example of this in humans would be eye-color or height. Selection is performed on the phenotype. This biological concept was introduced to the field of EC as a method for separating the search and solution space [2] and as a metaphor for describing the representations and mapping processes.

It could be argued that in traditional tree-based GP there is no generative process. The trees that define the programs are directly manipulated by the operators, therefore there is no generative stage. Although, this is not the complete picture. The tree itself could be viewed as a genotypic encoding and the output of the program as the phenotype. Furthermore, in Genetic Algorithms there is a mapping of the evolved integer string translating it into a meaningful application. This highlights that a genotype to phenotype to fitness mapping exists in all forms of evolutionary computation except the most basic.

In GE there are several stages in the mapping process, each with its own observable characteristics. As each stage also contains a version of the final instantiation, it can be classified as a phenotype. The three main phenotypic stages are shown in Figure 2. The integer list is translated into a derivation tree using a BNF grammar defined in Figure 1 and the mod rule. The terminal

rules that make up the finished string are shaded. The string is then evaluated to produce the final output phenotype, in this case a 3 dimensional plot of a plane.

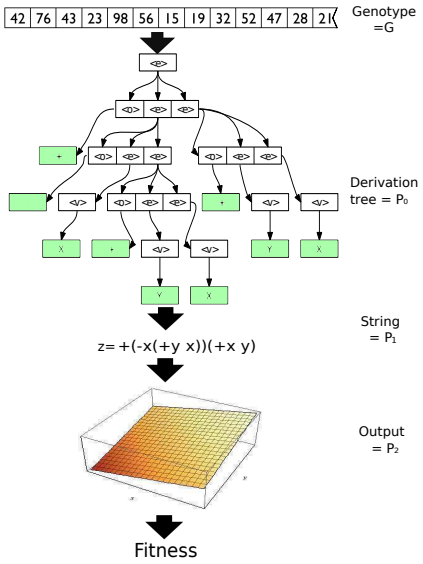


Figure 2: stages of derivation in GE

how well neighbouring genotypes correspond to neighbouring phenotypes, also known as locality, has been described as a key element in Evolutionary Computation. To study locality, it is necessary to define a metric on the search space. In a genotype-phenotype mapping representation. In his work, Rothlauf claimed that for two different search spaces (e.g., genotypic and phenotypic search space) it is necessary to define two different metrics [26]. As we are interested in determining how locality is affected on the different phenotypic stages of the generative process, we need to use a different metric for each one.

3.2 Mutation and Locality in GE

Standard GE mutation can be divided into two types of events, those that are structural in nature and those that are nodal. A nodal mutation changes a single leaf of the derivation tree. A structural mutation changes one or more internal nodes of the derivation tree (and zero or more leaves). This can result in a change to the length of the phenotype. This work was originally shown in [7]. In order to expose the impact of mutation on derivation tree structure we will use the binary grammar as shown in Figure 1. This allows us to condense codons (elements in the string representing the individual) to single bits.

We can then construct genomes with binary valued codons to construct sentences in the language described by the above grammar. Consider all genomes of length two codons (2^2 of them) and draw an edge between genomes that are

a Hamming distance of one apart. If we then present the corresponding partial derivation trees resulting from those genomes we see the arrangement outlined in Fig. 3. In this particular example we see that a mutation event at the first codon impacts a non-terminal rule that corresponds to a new derivation tree structure. Here we define a new derivation tree structure as being one that has changed in length, that is, it contains more non-terminal symbols than its neighbour. Mutations from 00 to 10 (and vice versa) and from 01 to 11 (and vice versa) result in these structural changes. Whereas the remaining mutation events result in node relabelling.

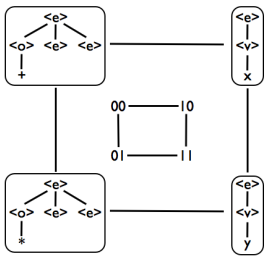


Figure 3: The 2D neighbourhood for the example grammar (i.e., using the first two codons).

Extending the genomes by an additional codon we can visualise the Hamming neighbourhood between the 2^3 genomes both in terms of codon values and partial phenotype structures. These are illustrated in Fig. 4. Again, we see a clear distinction between mutation events that result in structural and non-structural modifications.

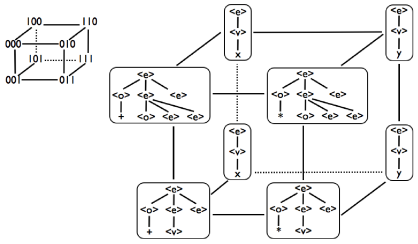


Figure 4: The 3D neighbourhood for the example grammar (i.e., using the first three codons).

Mapping these codons back to the grammar we see that structural mutations occur in the context of a single non-terminal symbol, `<e>`. We can see from this grammar that this non-terminal alone is responsible for structural changes, as it alone can change the size of the developing structure. The rules for the `<o>` and

<v> non-terminals are non-structural as they simply replace an existing symbol without changing structural length.

Effectively we can now decompose the behaviour of mutation into two types of events. The first are events that are structural in their effect and the second are those which are nodal in their effect. By logical extension we could consider both types of events as operators in their own right, and therefore define a *structural mutation* and a *nodal mutation*. It should be noted, however, that these events are only a specialisation of standard GE mutation, as it is possible for both types of events to occur during standard application of GE mutation to an individual's genome. In order to understand the impact of change arising from mutation events in GE we need metrics of distance between the different stages. The metrics we use in this experiment are discussed in Section 6

4 Experimental Setup

The goal of the experiment is to compare whether separating mutation events allows the user to navigate the search space more efficiently. This experiment was run using Architype, an interactive design generation tool based on GE. Architype provides an interface for selecting designs for mutation or crossover. The Architype interface was adapted for this experiment as shown in Figure 5. There is a target bridge on the right hand side and nine bridges to select from on the left hand side. When the user selects a bridge, their choice is saved in a green box in the top left frame and 8 mutated variations are made of it. The user can keep selecting bridges until they think they have matched the target. Twenty four volunteers participated in this experiment and the experiment itself was approved by the Ethics Committee in ***. The grammar used in this experiment is described in Section 4.1. The subjective nature of aesthetics makes evolutionary design search a difficult area to quantify. Instead of asking the user to find a design that they like, we specify a target design that they must evolve towards.

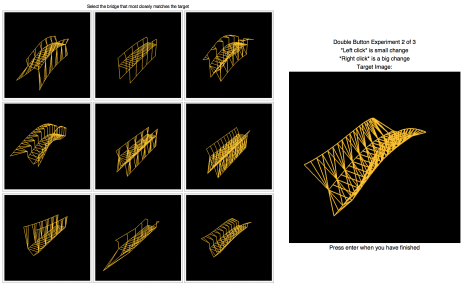


Figure 5: A screen-shot of the interactive GUI

4.1 Design Grammar

The grammar was originally conceived based on a brief provided to third year students in the *** 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 is available on-line at [6]. The grammar creates graphs using networkx [14], a python class for studying complex graphs and networks. Three desirable characteristics for a design generator are modularity, regularity and hierarchy [15]. We implement these characteristics using the novel method of higher order functions. Work in this area is discussed in greater detail in [19].

4.2 Experimental Design

The aim of the experiment is to compare standard integer mutation against a combination of nodal and structural mutation. There is no crossover or selection used in this experiment. All variants are created exclusively by mutation events. The mutation operators do not work probabilistically, instead they select a codon from within the coding region of the genome and increment it by one. As a codon is only used when choosing between rules, incrementing it by one means that it will always encode for a new rule, thus ensuring a genomic change in the individual. It also means that the hamming distance between mutation events is exactly one.

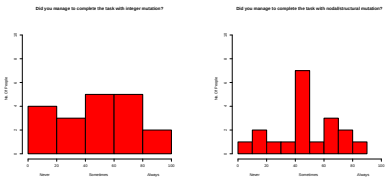
To avoid confusion we decided to simplify the allowed user input. The user is limited to either the left mouse button (LMB) exclusively, or both mouse buttons. When the input is exclusively LMB, standard integer mutation is applied. When the user has a choice of both buttons, the LMB causes nodal mutation and the RMB causes structural mutation.

The user was allowed two trial runs to familiarise themselves with the interface and the different effects of the mutation operators. Finally, after completion of the trial the user was asked to complete six experiments. The first three of the experiments used integer mutation to match three targets and the next three experiments used nodal and structural mutation to match the same three targets. A fixed random seed was used for each experiment so that all participants would be presented with the same initial designs. The time of each selection and the individual selected were recorded. The user had a time limit of five minutes to complete each task.

Upon completion of the experiment the user was presented with every selection they had made and was asked to select the design that most closely matched the target. After completion of the experiment, the participant was asked to complete a short survey. The survey may be viewed online at [32].

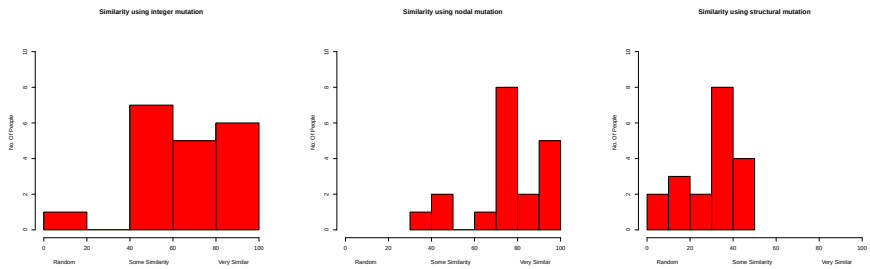
5 Survey Results

The survey results are shown in Figures 6 through 9. The survey recorded user opinions on various aspects of the task using the Likert scale [18]. The X axis for the histogram represents their level of preference. Zero percent represents strongly disagree whereas one hundered percent represents strongly agree. The Y axis represents the numbers of user who agreed with that preference. There is a significant number of people who felt they successfully completed the task using the structural/nodal mutation operators (Figures 6 and 8). The users also noticed a large difference in locality of the operators (Figure 7). The users found nodal mutation generated more similar variations than integer mutation and that structural mutation produced the least amount of similarity.



(a) Integer (b) Nodal/Structural

Figure 6: Did you manage to complete the task?

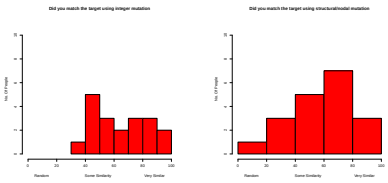


(a) Integer (b) Nodal (c) Structural

Figure 7: Were the variations similar?

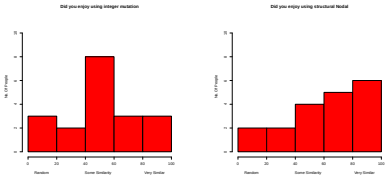
6 Distance Metrics

To analyse how closely the user matched the target we used a selection of distance metrics to evaluate the user's choices. Each distance metric was chosen to record the distance during a different phenotypic level of the mapping process.



(a) Integer (b) Nodal/Structural

Figure 8: Did you match the target?



(a) Integer (b) Nodal/Structural

Figure 9: How would you describe the task?

6.1 Tree Edit Distance

O'Reilly [22] proposed an approach, called edit distance, with the main goal of having a metric that specifies the degree of dissimilarity between two individuals in the form of tree-like structures. The idea of edit distance is to calculate the minimum cost (number of moves) that is required to transform a given tree to a target tree step by step. For this purpose, the author defined the use of three types of edits: (a) Substitution: changing a node into another, (b) Insertion: adding a node within the tree and (c) Deletion: removing a node from the tree. O'Reilly's approach was inspired by the work reported in [27, 30]. This distance is notable because it is closely aligned with a mutation operator defined in the same paper. In GE this metric is applied to calculate a distance at the derivation tree stage of the mapping.

6.2 Levenshtein Distance

Levenshtein distance is a metric for comparing two strings, or in our case, generated computer programs. This has been used as a metric to compare representations in GP [17] [16]. The Levenshtein distance between two strings is defined as the minimum number of edits needed to transform one string into the other, with the allowable edit operations being insertion, deletion, or substitution of a single character. In our study the output string is tokenised and the levenshtein metric is applied to phenotype symbols.

6.3 Euclidean Distance

Euclidean distance is defined as the straight-line distance between two points on the same plane. It is derived from Pythagoras's theorem and is given by the formula

$$distance = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}.$$

An approach was developed for applying the euclidean metric to our bridge designs that satisfied the mathematical conditions expected of a metric. When comparing two designs, the design with the most nodes is selected. Each node in the larger design is then iterated through and the nearest node in the smaller design is found. The distance between these nodes is then added to the total distance between designs.

This metric satisfies the metric conditions of non-negativity, symmetry and the triangle inequality. As our approach only compares point on the design the 'identity of indiscernibles' condition is not fulfilled as the edges between nodes has no impact on the distance. We decided not to include node edges in the distance as sub graph isomorphism is an NP Complete problem [9]. The Euclidean metric provides a distance for the final stage of the output, the graph representing the bridge itself.

6.4 Distance Metric Results

The results for the distance metrics are shown in Figures 10 through 15. The Figures show the time taken (in seconds) on the X axis and the distance from the target on the Y axis. The smaller the value on the Y axis, the more closely the result matched the target. The graphs show the best result obtained from the user over the course of the run. As shown in the graphs, there is little or no improvement over time. This result was further confirmed by simulating the user input using random selection. A T-Test was performed and no significant difference was found between the user data and the randomly generated data. We only show the results for the first target image as the other target graphs show similar results. The results are discussed further in Section 7

7 Discussion

The results from Section 6.4 appear to prove the Null hypothesis, that the user is unable to direct search using interactive Operators and selection. This is a highly contentious conclusion to draw as many years of interactive evolutionary computation studies have shown the opposite. If this is not the case then there are two possible causes for the results.

7.1 Choice of Metrics

How a human evaluates the similarity of two designs is a subjective measurement. Our survey showed that users often based how similar they found bridges

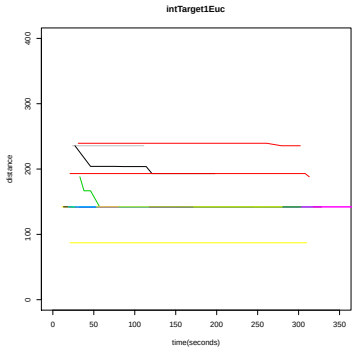


Figure 10: Integer euclidean distance for target 1

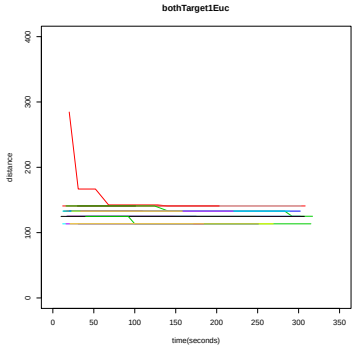


Figure 11: Nodal/Structural Euclidean distance for target 1

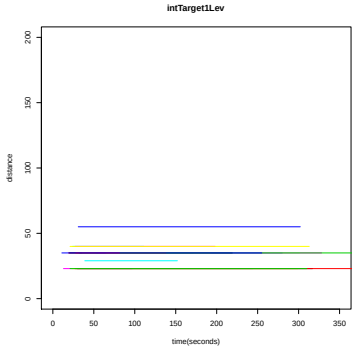


Figure 12: Integer Levenstein distance for target 1

on individual features or parts of the design, such as the handrail or the curve

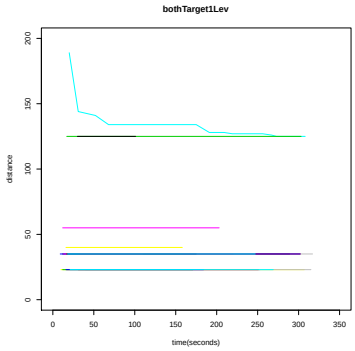


Figure 13: Integer Levenstein distance for target 1

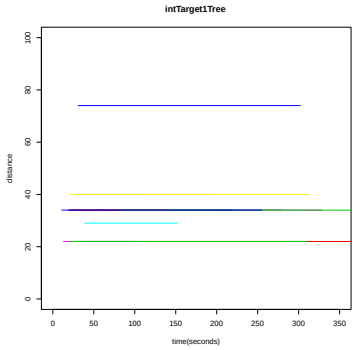


Figure 14: Integer Tree distance for target 1

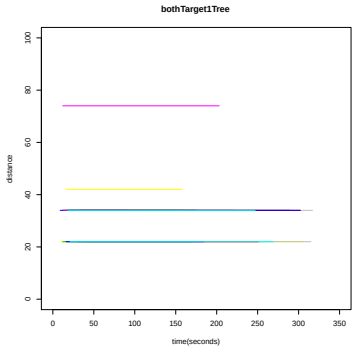


Figure 15: Integer Tree distance for target 1

of the walkway. While the metrics comparing bridges at earlier stages of the

mapping process (tree edit and Levenstein distance) would have great difficulty recording small changes on components, Euclidean distance should have recorded some improvement. While the metrics are not perfect and human selection is subjective, some improvement should have been detectable.

7.2 Methodology

The experiment was constructed so as to facilitate precise logging of input. Pilot studies were completed to ensure the interface was usable and that certain concerns were addressed such as saving previous designs and allowing a single design to be repeatedly mutated. What was not addressed is the question of how to best facilitate the user's exploration of the search space. On average, a user made 17 selections during the course of an experiment. This equates to a hamming distance of 17 from their original selection. To assume that a significant improvement could be made in this short distance was optimistic.

Although the design selections were presented to the users at the end of the run, many were frustrated by the inability to go back to a good design. By forcing the user to follow a specific evolutionary path, our experiment severely limited the user and added to their frustration when they "lost" a good design.

The locality of the nodal and structural operators may not have translated into localised changes in the output. Although the operators are localised on the derivation tree level, the mapping process of GE could magnify the impact it has on the bridge. There is also the problem that what constitutes a small or a large change depends on the user's personal preference.

The grammar also complicated what was already a difficult task. Several participants complained of identical bridges being generated. This was the result of very small changes being made to the Bezier curve defining either the handrail or the walkway. Such changes fall below the threshold of a Just Noticable Difference (JND). JND is a concept from cognitive psychology that was developed by the founder of psychophysics, Ernst Heinrich Weber. 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.

8 Conclusions and Future Work

In this paper we explored the application of novel mutation operators by the user. We discussed locality at different stages of the GE mapping process and examined whether this could have a beneficial effect on search. While the survey showed that the user's found a clear difference between the operators, the distance metrics showed no evidence of this being beneficial to search. As shown in the discussion section, this is the result of our approach to the interface.

It became clear from this experiment that allowing the user to apply the operators is not enough, the user has to be able to interact with the search space in a meaningful way. After the feedback we received and our analysis of

the results, we intend to re-implement the interface to allow the user to explore the search space more efficiently. We also intend to examine whether the metrics adequately reflect similarity and locality.

9 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/RFP/CMS1115 and 08/IN.1/I1868.

References

[1] K. Aoki and H. Takagi. Interactive GA-based design support system for lighting design in 3-D computer graphics. *Trans. of IEICE*, 81:1601–1608, 1996.

[2] Wolfgang Banzhaf. Genotype-phenotype-mapping and neutral variation—a case study in genetic programming. In *Proceedings of Parallel Problem Solving from Nature III*, volume LNCS 866, pages 322–332. Springer, 1994.

[3] Wolfgang Banzhaf. Interactive evolution. In Thomas Back, David B. Fogel, and Zbigniew Michalewicz, editors, *Handbook of Evolutionary Computation*, chapter C2.9, pages 1–6. IOP Publishing Ltd. and Oxford University Press, 1997.

[4] Peter J. Bentley and Una-May O'Reilly. Ten steps to make a perfect creative evolutionary design system. In *GECCO 2001 Workshop on Non-Routine Design with Evolutionary Systems*, 2001.

[5] J. Biles. GenJam: A genetic algorithm for generating jazz solos. In *Proceedings of the International Computer Music Conference*, pages 131–131. INTERNATIONAL COMPUTER MUSIC ASSOCIATION, 1994.

[6] Link to the bridge grammar. http://i.imgur.com/0vsDh.png.

[7] J. Byrne, M. O'Neill, and A. Brabazon. Structural and nodal mutation in grammatical evolution. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 1881–1882. ACM, 2009.

[8] Jonathan Byrne, James McDermott, Edgar Galván López, and Michael O'Neill. Implementing an intuitive mutation operator for interactive evolutionary 3d design. In *IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2010.

[9] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[10] Richard Dawkins. *The Blind Watchmaker*. Longman Scientific and Technical, Harlow, England, 1986.

[11] N. Dorris, B. Carnahan, L. Orsini, and LA Kuntz. Interactive evolutionary design of anthropomorphic symbols. In *CEC2004: Proceedings of the 2004 Congress on Evolutionary Computation*, volume 1, 2004.

[12] Generative components. http://www.bentley.com/getgc/.

[13] Grasshopper, generative modeling with rhino. http://www.grasshopper3d.com/.

[14] Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, Pasadena, CA USA, 2008.

[15] Gregory S. Hornby. Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In *Proceedings of GECCO '05*, 2005.

[16] Christian Igel. Causality of hierarchical variable length representations. In *Proceedings of the 1998 IEEE World Congress on Computational Intelligence*, pages 324–329, Anchorage, Alaska, USA, 5-9 May 1998. IEEE Press.

[17] R.E. Keller and W. Banzhaf. Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes. In *Genetic Programming 1996: Proceedings of the First Annual Conference*, pages 116–122, Stanford University, CA, USA, 1996. MIT Press.

[18] R. Likert. A technique for the measurement of attitudes. *Archives of Psychology*, 22(140):1–55, 1932.

[19] James McDermott, Jonathan Byrne, John Mark Swafford, Michael O'Neill, and Anthony Brabazon. Higher-order functions in aesthetic EC encodings. In *2010 IEEE World Congress on Computational Intelligence*, pages 2816–2823, Barcelona, Spain, 2010. IEEE Press.

[20] J.M. McDermott. *Evolutionary Computation Applied to the Control of Sound Synthesis*. PhD thesis, University of Limerick, 2008.

[21] Michael O'Neill and Conor Ryan. *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers, 2003.

[22] Una-May O'Reilly. Using a distance metric on genetic programs to understand genetic operators. In *IEEE International Conference on Systems, Man, and Cybernetics: Computational Cybernetics and Simulation*, volume 5, 1997.

[23] Una-May O'Reilly and Martin Hemberg. Integrating generative growth and evolutionary computation for form exploration. *Genetic Programming and Evolvable Machines*, 8(2):163–186, June 2007. Special issue on developmental systems.

[24] Una-May O'Reilly and Girish Ramachandran. A preliminary investigation of evolution as a form design strategy. In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles E. Taylor, editors, *Proceedings of the Sixth International Conference on Artificial Life*, pages 443–447, University of California, Los Angeles, 26-29 June 1998. MIT Press.

[25] IC Parmee and CR Bonham. Towards the support of innovative conceptual design through interactive designer/evolutionary computing strategies. *AI EDAM*, 14(01):3–16, 2000.

[26] Franz Rothlauf. *Representations for Genetic and Evolutionary Algorithms.* Physica-Verlag, 2nd edition, 2006.

[27] D. Shasha and K. Zhang. Fast Parallel Algorithms for the Unit Cost Editing Distance Between Trees. In *SPAA '89: Proceedings of the First Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 117–126, New York, NY, USA, 1989. ACM.

[28] K. Shea, R. Aish, and M. Gourtovaia. Towards integrated performance-driven generative design tools. *Automation in Construction*, 14(2):253–264, 2005.

[29] Karl Sims. Artificial evolution for computer graphics. In *SIGGRAPH '91: Proceedings of the 18th annual conference on computer graphics and interactive techniques*, pages 319–328, New York, NY, USA, 1991. ACM.

[30] M. Tacker, P. F. Stadler, E. G. Bornberg-Bauer, I. L. Hofacker, and P. Schuster. Algorithm Indepedent Properties of RNA Secondary Structure Predictions. *European Biophysics Journal*, 25(2):115–130, 1996.

[31] Hideyuki Takagi. Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. *Proc. of the IEEE*, 89(9):1275–1296, 2001.

[32] Link to the user survey. http://i.imgur.com/BYRdL.png.

[33] P.A. Whigham. *Grammatical Bias for Evolutionary Learning.* PhD thesis, University of New South Wales, Australian Defence Force Academy, Canberra, Australia, 1996.