Shape Grammars and Grammatical Evolution for Evolutionary Design

Michael O'Neill
Natural Computing Research & Applications Group
University College Dublin
m.oneill@ucd.ie

John Mark Swafford
Natural Computing Research & Applications Group
University College Dublin
johnmarksuave@gmail.com

James McDermott
Natural Computing Research & Applications Group
University College Dublin
jamesmichaelmcdermott@gmail.com

Jonathan Byrne
Natural Computing Research & Applications Group
University College Dublin
jonathanbyrn@gmail.com

Anthony Brabazon
Natural Computing Research & Applications Group
University College Dublin
anthony.brabazon@ucd.ie

Elizabeth Shotton
School of Architecture, Landscape & Civil Engineering
University College Dublin
elizabeth.shotton@ucd.ie

ABSTRACT

We describe the first steps in the adoption of Shape Grammars with Grammatical Evolution for application in Evolutionary Design. Combining the concepts of Shape Grammars and Genetic Programming opens up the exciting possibility of truly generative design assist tools. In this initial study we provide some background on the adoption of grammar-based Genetic Programming for Evolutionary Design, describe Shape Grammars, and give a brief overview of Grammatical Evolution before detailing how Grammatical Evolution used Shape Grammars to successfully rediscover some benchmark target structures.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]; I.2.2 [**Automatic Programming**]; F.4.2 [**Grammars and Other Rewriting Systems (D.3.1)**]

General Terms

Algorithms, Experimentation

Keywords

shape grammars, evolutionary design, grammatical genetic programming, grammatical evolution

1. INTRODUCTION

The natural process of biological evolution has served as inspiration for the development of powerful problem solving tools in the form of Evolutionary Algorithms. Of particular note are the Genetic Programming variants which are now capable of routine human-competitive performance, e.g. in the area of Analog Circuit Design [24]. Some of the evolved solutions have even passed human tests of innovation by being patentable in their own right. The long term objective of the research, of which this study forms the seed, is to extend and examine a powerful and already well recognised grammatical approach to Genetic Programming, Grammatical Evolution, to challenging Architectural design problem environments. This will be achieved through the development of a more evolvable and rich representation, Shape Grammars [45]. Shape Grammars allow us a natural way to encode human domain knowledge into the evolutionary/generative process, and they are proving invaluable in the area of Grammatical Design for example by the Integrated Design Innovation Group at Carnegie Mellon University to explore the essence of the design of products ranging from Harley Davidson motorcycles to Cars and Coffee Makers [8, 50]. To date, this powerful Shape Grammar formalism has not been combined with an advanced grammatical evolutionary algorithm such as Grammatical Evolution, and the proposed research addresses this important gap.

The remainder of this paper is structured as follows. The following Section 2 provides some background on the adoption of Genetic Programming for Evolutionary Design, we then expose the use of Shape Grammars as a Design Language in Section 3. A brief overview of Grammatical Evolution is provided in Section 4 before we outline the experiments undertaken and finally draw conclusions in Sections 5 and 6 respectively.

2. BACKGROUND

Evolutionary Computation has clearly demonstrated its potential for Evolutionary Design producing solutions that are competitive, and in some cases superior to those developed by human experts resulting in patentable inventions [24, 49, 5]. As such, the real world application domain of Design (in particular Analog Circuit Design [24]) has been a proving ground for the abilities of an artificial evolutionary process and has led to arguably the first routinely, human-competitive form of Machine Learning.

The combination of an Evolutionary Algorithm coupled to a Grammatical Representation (or Design Language) is a

GECCO'09, July 8–12, 2009, Montréal Québec, Canada.

particularly powerful and novel departure in recent years [19]. Examples of research at this nexus of Evolutionary Computation and a Grammatical Representation [19, 17, 40, 12] include Genr8 and GENRE approaches.

The grammar-based form of GP as realised in Grammatical Evolution brings a number of strengths over more traditional optimisation methods for the design domain [19, 17, 30]:

- GP handles the search of open-ended structure. The model size and structure is not specified a-priori.
- Search is stochastic. This facilitates the explorative process as it is not limited or biased by the imagination of the human user. This can result in designs which are novel and sometimes counterintuitive. This has significant implications in the architectural design process, as evidenced by recent work by architectural practices such as Greg Lynn and Foreign Office Architects, as it facilitates creativity through unanticipated form generation and removes the bias of preconceptions based on prior typologies or solutions.
- On the flip-side of the innovative, un-biased explorative process embodied in stochastic evolutionary search, if desired, positive bias can be introduced in the form of architectural domain knowledge. This can be achieved by allowing the user to specify proven structural forms, planning constraints, and even aesthetic preferences a-priori. In a grammar-based form of GP such as Grammatical Evolution, domain knowledge can be easily incorporated through the underlying grammatical representation.

The natural process of biological evolution has clearly demonstrated its power to design elegant form and structure in the name of survival. As such, it is natural to turn to algorithms which are inspired by the biological process of evolution to tackle the development of algorithms for design.

A number of open issues (e.g., see [27]) must be addressed to ensure a successful application of EAs and in particular GE to Design, for example,

- Representation: This should be evolvable, adaptable, allow incorporation of domain knowledge, and be general enough to allow the generation of all possible structures. Grammars have been shown to have much potential as they are both amenable to evolutionary search, through a method like GE, and they provide the building blocks of structural information that can be used to allow the generation of any shape [19, 17]. Grammars provide an easy mechanism by which domain knowledge can be encoded, and with a grammar-based approach to GP like GE, we can also subject the underlying grammar to the process of evolution itself [34, 33, 14, 15, 16].
- Evolvability: The population of candidate designs must have the potential to provide constant innovation and improvements [1]. Critical to the efficient operation of any EA, particularly in the application domain of design, is to incorporate mechanisms for modularity, hierarchy and reuse into the algorithm [19, 14]. These mechanisms will allow a more adaptive and efficient exploration of the design search space. In the case of Architectural Design, the application domain of interest in this research, there are clear parallels between these concepts of modularity, hierarchy and reuse during the human driven design process. Once a successful design module is discovered it can be constantly reused, adapted, or re-evaluated relative to a variation in the underlying constraints, and as such the incorporation of such principles into an automated search technique will prove invaluable. The underlying grammatical representation is also key to evolvability, and the main focus of this research will be towards the evolvability of Shape Grammars.
- Evaluation of candidate designs: In Interactive EC (iEC) where the fitness function is driven even partly by a human user a number of significant challenges arise [27, 49]. Of primary concern is the cost of evaluating the evolving designs, as a human-in-the-loop is a bottleneck which severely slows the search process. A novel strategy to overcome this will be investigated in future studies [28].

3. SHAPE GRAMMARS AS THE DESIGN LANGUAGE

The Design Language is represented in the input grammar that GE uses in the construction of a solution (in this case a design). A number of possible Design Languages exist, for example, L-systems [26] have been adopted and extended with Genr8 [17, 40] and Shape Grammars have been explored in their own right [47, 46, 7] and in combination with other Evolutionary Algorithms (e.g., see [12, 3, 2]). In this study we concentrate on shape grammars.

It is clear from earlier research in Grammatical Design that Shape Grammars are a powerful representation that can be used with success in the design of structures [7, 12, 23, 48, 21, 25] and as such they will be the starting point for this research. Shape grammars allow the recognition and transformation of a set of shapes. An example shape grammar illustrating how the nonterminals (primitive shapes) can be transformed is presented in Figure 1. For example, the production rules (transformations) can include movement of a shape along an axis, a description of how a shape can be sub-divided or placed alongside other shapes, and even transformations such as scaling.

To date Shape Grammars have not been combined with the representational and search power of an advanced grammar-based evolutionary algorithm such as Grammatical Evolution. This proposal addresses this important research gap by directly addressing the limitations of the earlier studies on Shape Grammars with Evolutionary Computation. In particular, we will extend earlier research on Shape Grammars [12] to examine their adoption with GE in the first instance to overcome the limitations of a fixed-length and fixed-integer rule encoding and the resulting issue of illegal genotypes being generated.

4. GRAMMATICAL EVOLUTION

Grammatical Evolution (GE) [10, 35, 29, 36, 37, 42] is a grammatical approach to GP, and there is a vast literature on the use of grammars in GP, e.g., see [18, 41, 51, 52, 44]. GE has been adopted for a range of application domains,

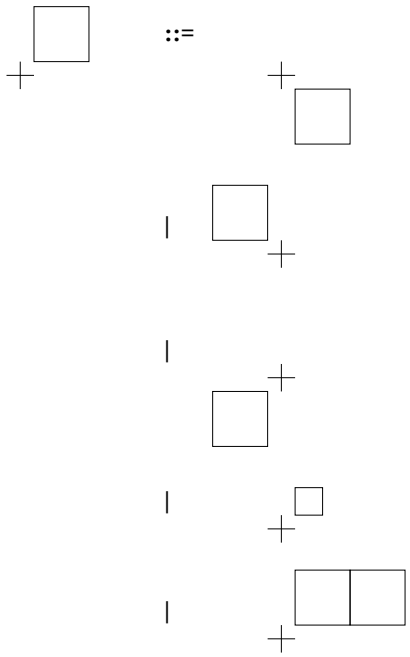


Figure 1: An example Shape Grammar containing a single square non-terminal with five possible productions. The cross-hairs indicates a point of reference to help illustrate what effect each production rule has. The first three rules translate the position of the non-terminal square south, west and south-west respectively. The fourth rule transforms the original square non-terminal by scaling its size by half. The fifth production duplicates the square and places it alongside the original non-terminal.

e.g., financial modeling (e.g., [6, 9])[1]. In addition to the notion of explicit grammars, GE borrows additional principles from molecular biology. The most powerful of these is the genotype-phenotype map. Earlier research in GP has shown some of the potential benefits of such a mapping [4, 20] and GE further exploits this representation to create a highly-modular and flexible approach to program/model induction. An example of this is the fact that alternative search engines can be adopted to explore the model space (e.g. Particle Swarm Optimization and Differential Evolution have been adopted [31, 32]). The flexibility of GE is such that even with the presence of the genotype-phenotype map, traditional tree-based search operators of crossover and mutation can be adopted in place of the genotype search operators effectively transforming GE into a standard form of GP with the grammar used during the initialization of the population [13]. Of course, it is also entirely possible to combine search operators that are focused on both the genotype and phenotype combining the benefits of each approach.

GE uses grammars in a generative sense to guide the construction of phenotypic solution structures. We briefly illustrate an example of such a mapping taking the grammar adopted in this study as an example (see Figure 2).

[1]Further information on GE and pointers to code can be found at `http://www.grammatical-evolution.org` and `http://ncra.ucd.ie`

```
<prog> ::= <term> | <term> <prog>
<term> ::= <var> | <op> <term> | [ <term> ]
  <op> ::= s0 | s1 | s2 | s3 | gro | shrnk
 <var> ::= sqr | crcl
```

Figure 2: The Shape Grammar adopted in this study represented in a more typical GP-like symbolic expression representation.

The behaviour of the operators (`<op>`) adopted in the grammar are as follows; `s0` moves the shape right 10 pixels, `s1` moves the shape down 10 pixels, `s2` moves the shape left 10 pixels, `s3` moves the shape up 10 pixels, `gro` doubles the size of the shape, and `shrnk` halves the size of the shape. `sqr` draws a square shape, `crcl` draws a circle shape, `[` and `]` push and pop the pen's state (i.e., position of where we draw next) onto and off the stack respectively.

We begin the development of each individuals solution structure from a seed known as the Start symbol. In Grammatical Evolution this is typically the first non-terminal symbol encountered in the grammar file. In this case starting from `<prog>` there are two possible rules which can transform it into either `<term>` or alternatively into `<term><prog>`. To decide which rule replaces `<prog>` we consult the genome by reading the next available codon integer value and apply the following mapping function:

$$Rule \ = \ c \,\%\, n$$

where `c` is the the codon integer value, and `n` is the number of choices available for the current non-terminal context. Given a codon integer value of 13, this gives $13 \ \% \ 2 \ = \ 1$. In other words, `<prog>` will be replaced with `<term><prog>`. The mapping continues by taking the left-most non-terminal symbol in the developing solution, examining the grammar to determine if a choice has to be made for that non-terminal and if so, the next codon integer value is read. If the next codon had the value 7, $7 \,\%\, 3 = 1$ resulting in `<term><prog>` being replaced with `<op><term><prog>`. Continuing to read codon values from the example individual in Figure 3 to map the left-most non-terminal will give the expansion in Figure 4.

13	7	24	3	45	10	27	5	----------

Figure 3: An example Grammatical Evolution individual represented as integer codon values.

This solution has the effect of moving the pen's position 10 pixels to the right (s0), drawing a circle at this point (crcl), and then drawing another circle over the first one (crcl). In contrast if the generated solution had the form `[s0 crcl] crcl` this would have moved the pen 10 pixels to the right, drawn a circle, and then the pen state is reset to the center of the image and the second circle is drawn in this location, resulting in two circles side-by-side.

5. EXPERIMENT DESCRIPTION

In this study we wish to determine if it is possible to rediscover structures using a Shape Grammar approach with

```
<op><term><prog>
 |
 |
 V
s0 <term><prog>
     |
     |
     V
s0 <var><prog>
     |
     |
     V
s0 crcl <prog>
           |
           |
           V
s0 crcl <term>
           |
           |
           V
s0 crcl <var>
           |
           |
           V
s0 crcl crcl
```

Figure 4: A sample partial Grammatical Evolution mapping of an individual using the Shape Grammar adopted in this study.

Grammatical Evolution. To this end we apply the Shape Grammar in Figure 2 to three separate targets of increasing difficulty, which are outlined in Figures 5, 6 and 7.

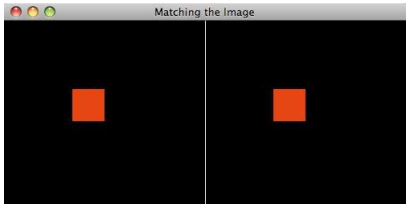


Figure 5: The first (easy) benchmark target tackled in this study. The target is depicted on the left and the best evolved structure is depicted in the right half of the image.

We adopted the GEVA v1.0 software [38, 39] for these experiments and the parameter settings were employed as outlined in Table 1.

Evolution in this case was driven by an automated fitness function as we are attempting to rediscover known benchmark target structures. The fitness function adopted minimises fitness and is:

$$fitness = \sum_{i=0}^{i=n-1} if\ (t_i\ !=\ p_i)\{fitness = fitness + 1;\}$$

t_i is the pixel value at target image index i, p_i is the pixel value of the evolved image at index i, with a total image size

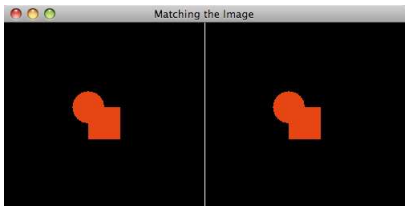


Figure 6: The second (medium difficulty) benchmark target tackled in this study. The target is depicted on the left and the best evolved structure is depicted in the right half of the image.

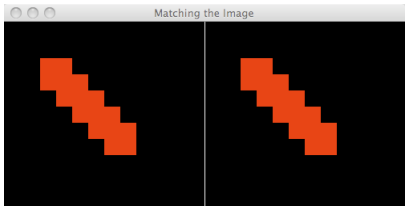


Figure 7: The third (hard) benchmark target tackled in this study. The target is depicted on the left and the best evolved structure is depicted in the right half of the image.

Table 1: Grammatical Evolution parameter settings adopted for each of the three benchmark problem instances.

Parameter	Value
integer codon mutation	0.02
initialisation	Ramped-half-and-half
generations	100
population size(problem1, 2, & 3)	200, 200, 2000
selection	tournament
tournament size	3
replacement	generational
elitism	5 individuals
crossover 1-pt	0.7
max wraps	3

n is 250×250 pixels. If the target pixels value is different from the evolved solution we punish fitness by adding one to the overall fitness. This is a rather simple fitness measure which does not pay respect to the shapes structure, so for example, if the target solution contained a single circle and the evolved solution contained a circle but at a different location, the fact that the solution found a circle is not rewarded in this case. Future work will investigate more sophisticated fitness measures.

Fitness plots showing both best and average fitness averaged over thirty independent runs on each target are given in Figures 8, 9 and 10 for the easy, medium and hard instances. As can be observed from the average best fitness plots, all runs find a solution to each problem with the given parameter settings.

Samples of successful solutions to each of the targets are provided in Figure 11. It is worth noting in the easy problem instance that the target square is not centered at the de-

```
s2 s3 s2 s3 gro sqr crcl shrnk crcl s2 s3 gro crcl gro s3 s2 sqr


s3 s3 s2 s0 gro s2 crcl s1 shrnk s2 s1 s3 crcl s0 shrnk gro s2
sqr gro s3 gro sqr s2 s0 sqr sqr s2 s2 gro s3 shrnk s3 [ crcl ]


gro gro sqr shrnk s2 s3 sqr gro s3 s2 sqr shrnk s2 s3 [ s2 crcl ]
sqr gro s2 s3 sqr gro s1 s2 s3 shrnk s2 s3 crcl [ s3 shrnk [ s3 s2
[ [ [ s3 gro s2 [ sqr ] ] ] ] ] ] [ s3 crcl ] s3 sqr
```

Figure 11: A sample evolved solution for the easier target (top), for the medium target (middle), and the hardest target (bottom).

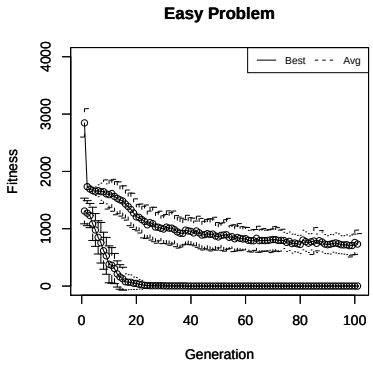


Figure 8: Fitness plot of the best and average population fitness averaged over thirty independent runs on the easier target.

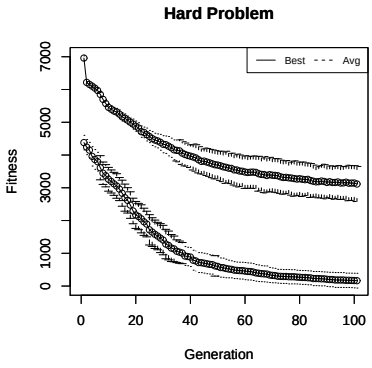


Figure 10: Fitness plot of the best and average population fitness averaged over thirty independent runs on the harder target.

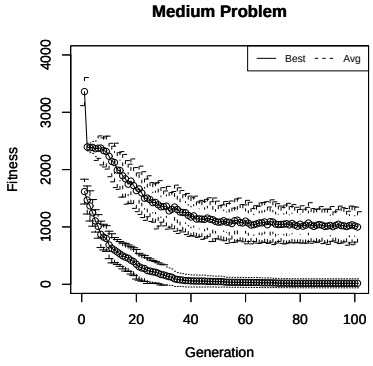


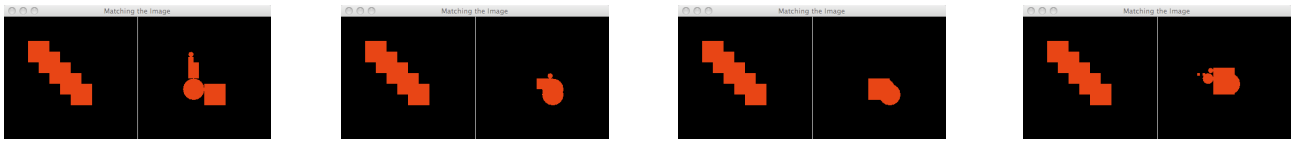
Figure 9: Fitness plot of the best and average population fitness averaged over thirty independent runs on the medium difficulty target.

fault pen starting position. In the simplest case, a successful solution to this problem must reposition the square before drawing it, and it must also apply two `gro` transformations to create a square of the correct size. In the evolved solution for the easier problem instance presented in Figure 11 it can be noted that two circles and three squares are drawn, so there is some redundancy in the solution, however all the shapes fall inside a perfectly sized and positioned square, and as such it scores perfect fitness.

For the harder problem instance, Figure 12 illustrates randomly generated solutions from generation zero. Figure 13 samples some of the best of generation solutions from a successful run. The gradual variation of the solution towards the target is evident. The leftmost solution is very close to the ideal target but has some extra structure in the middle squares. Gradually these extra structures are moved about until eventually the ideal solution is found.

6. CONCLUSION & FUTURE WORK

We presented a study which examined the utility of Shape Grammars with Grammatical Evolution for Evolutionary Design. Much potential exists for the application of Evolutionary Computation and in particular Grammatical approaches to Genetic Programming to Design, with many examples of human-competitive and even superior performance in the literature. Advantages of Genetic Programming approaches to design problems include its ability to navigate the space of open-ended structures, its stochastic, innovative nature, and the ease with which domain knowledge can be incorporated into the generative process.



```
gro gro shrnk gro s3 s1 sqr s2 s3 s2 s2 s2 crcl shrnk s0 s3 s3 sqr s3 s1 s3 sqr s3 shrnk sqr s3 crcl crcl

s0 gro s3 shrnk s3 crcl s2 s1 gro s2 sqr gro s0 s1 crcl shrnk s3 sqr gro crcl crcl

gro gro s1 s3 gro shrnk crcl shrnk s3 gro s2 s2 sqr s0 s0 s2 shrnk crcl shrnk gro gro crcl shrnk s0 sqr
s2 sqr sq

shrnk s2 gro s3 s2 s3 s2 s3 crcl s0 s3 s0 s1 gro s2 gro sqr s0 s2 s1 s0 gro shrnk crcl s1 s3 gro shrnk
s2 sqr shrnk s2 s1 s0 crcl s3 s2 s2 crcl shrnk shrnk sqr s2 sqr crcl
```

Figure 12: Examples of randomly generated generation zero individuals on the harder problem instance (top). The corresponding phenotypic expressions are also provided in order of appearance of the solutions from left to right (bottom).

Figure 13: Samples of best of generation individuals from a successful run on the harder problem instance. The target is on the left with the evolved solution on the right of the figure. The gradual progression towards the target is evident.

This last point is particularly relevant to the grammatical approaches to Genetic Programming.

Three benchmark problems of increasing complexity were examined, and Grammatical Evolution adopting Shape Grammars successfully found the target solution in each case. This study is very encouraging for our continued research in this area where we will expand upon the primitive shapes and operators which can be adopted with the grammar, investigate more sophisticated automated fitness functions which capture more global information on the shape structures themselves. In addition, we will examine an interactive interface which will allow the human designer to directly interface with the evolving population, acting for the selection operator and also allowing intervention by allowing modification of the evolving solutions with the subsequent phenotype-genotype map to enable further evolution. At the time of writing we are also extending this research to 3D and more realistic design problems.

Acknowledgments

This publication has emanated from research conducted with the financial support of Science Foundation Ireland.

7. ADDITIONAL AUTHORS

Ciaran McNally (School of Architecture, Landscape & Civil Engineering, University College Dublin, email:ciaran.mcnally@ucd.ie), and
Martin Hemberg (Harvard Medical School, email:martin.hemberg@childrens.harvard.edu).

8. REFERENCES

[1] Altenberg, L. (1994). Evolution of Evolvability in Genetic Programming. In Advances in Genetic Programming, Chapter 3, pp.47-74. MIT Press.

[2] Ang, M.C., Chau, H.H., McKay, A., De Pennington, A. (2007). Combining Evolutionary Algorithms and Shape Grammars to Generate Branded Product Design. In Design Computing and Cognition '06 Part 7, pp.521-539. Springer.

[3] Abimbola O. Asojo. (2001). Exploring algorithms as form determinants in design. In Proceedings international space syntax symposium, volume 3, Atlanta, USA.

[4] Banzhaf, W. (1994). Genotype-phenotype-mapping and neutral variation- A case study in genetic programming, in Lecture Notes in Computer Science 866, *Parallel Problem Solving from Nature III*, pp. 322-332. Springer.

[5] Bentley, P. (Ed). (1999). Evolutionary Design by Computers. Morgan Kaufmann.

[6] Brabazon, Anthony and O'Neill, Michael (2006), *Biologically Inspired Algorithms for financial Modelling*, Springer.

[7] Brown, K. (1997) Grammatical Design. IEEE Expert, March-April, pp. 27-33.

[8] Cagan, J., Vogel, C.M. (2001). Creating Breakthrough Products: Innovation from Product Planning to Program Approval. Prentice Hall.

[9] Dempsey, I. (2007). Grammatical Evolution in Dynamic Environments. PhD Thesis. University College Dublin.

[10] Dempsey, I., O'Neill, M., Brabazon, A. (2009).

Foundations in Grammatical Evolution for Dynamic Environments. Springer.

[11] Frazer, J. (1995). An Evolutionary Architecture. Architectural Association, London.

[12] Gero, J.S. (1994). Evolutionary Learning of Novel Grammars for Design Improvement. AIEDAM, 8(2):83-94.

[13] Harper, R. and Blair, A. (2005). A structure preserving crossover in grammatical evolution, in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, D. Corne, Z. Michalewicz, M. Dorigo, G. Eiben, D. Fogel, C. Fonseca, G. Greenwood, T. K. Chen, G. Raidl, A. Zalzala, S. Lucas, B. Paechter, J. Willies, J. J. M. Guervos, E. Eberbach, B. McKay, A. Channon, A. Tiwari, L. G. Volkert, D. Ashlock, and M. Schoenauer, Eds., vol. 3. Edinburgh, UK: IEEE Press, 2-5 Sep., pp. 2537–2544.

[14] Hemberg, E., Gilligan, C., O'Neill, M., Brabazon, A. (2007). A Grammatical Genetic Programming Approach to Modularity in Genetic Algorithms. In LNCS 4445 Proceedings of EuroGP Tenth European Conference on Genetic Programming. Valencia, Spain, pp.1-11. Springer.

[15] Hemberg, E., O'Neill, M., Brabazon, A. (2008). Grammatical Bias and Building Blocks in Meta-Grammar Grammatical Evolution. In Proceedings of the IEEE World Congress on Computational Intelligence, Hong Kong. IEEE Press.

[16] Hemberg, E., O'Neill, M., Brabazon, A. (2008). Altering Search Rates of the Meta and Solution Grammars in the mGGA. In LNCS 4971 Proceedings of EuroGP 2008, Naples, Italy, pp.362-373. Springer.

[17] Hemberg M., O'Reilly U-M. (2004) Extending Grammatical Evolution to Evolve Digital Surfaces with Genr8. In LNCS 3003 Proc. of the European Conference on Genetic Programming, pp.299-308, Springer.

[18] Hicklin, J. (1986). Application of the genetic algorithm to automatic program generation. MSc Thesis, University of Idaho, Moscow, ID.

[19] Hornby G.S., Pollack J.B. (2001) The advantages of generative grammatical encodings for physical design. In Proc. of Congress on Evolutionary Computation, pp.600-607, IEEE Press.

[20] Keller, R.E. and Banzhaf, W. (1996). Genetic programming using genotype-phenotype mapping from linear genomes into linear phenotypes, in *Genetic Programming 1996: Proceedings of the First Annual Conference*, J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo, Eds. Stanford University, CA, USA: MIT Press, 28–31 Jul., pp. 116–122.

[21] Knight, T.W. (1980). The generation of Hepplewhite-style chair back designs. Environment and Planning B, Vol.7, pp.227-238.

[22] Knight, T.W. (1993). Color Grammars: the Representation of Form and Color in Design Leonardo, Vol. 26,pp.117-124.

[23] Koning, H., Eizenberg, J. (1981). The language of the prairie: Frank Lloyd WrightÕs prairie houses, Environment and Planning B Vol. 8, pp.295-323.

[24] Koza, John R., Keane, Martin A., Streeter, Matthew J., Mydlowec, William, Yu, Jessen, and Lanza, Guido. (2003). Genetic Programming IV. Routine Human-Competitive Machine Intelligence. Boston, MA: Kluwer Academic Publishers.

[25] Li, Andrew I-kang. (2002). Algorithmic Architecture in Twelfth-Century China: The Yingzao Fashi, pp. 141-150 in Nexus IV: Architecture and Mathematics, eds. Kim Williams and Jose Francisco Rodrigues, Fucecchio (Florence): Kim Williams Books.

[26] Lindenmayer, A. (1968). Mathematical models for cellular interactions in development. Parts I and II. Journal of Theoretical Biology, Vol. 18, pp. 280-315.

[27] McCormack, J., (2005) Open Problems in Evolutionary Music and Art, LNCS 3449 Applications of Evolutionary Computation, Proc. Of EvoMUSART 2005, pp.428-436. Lausanne, Switzerland. Springer.

[28] McDermott J., Griffith N., O'Neill M. (2007) Evolutionary GUIs for Sound Synthesis. In LNCS4448 Proc. EvoMUSART2007 The Fifth European Workshop on Evolutionary Music and Art Springer Giacobini et al. (Ed.'s), pp.547-556, Valencia, Spain. Springer.

[29] O'Neill, M. (2001).*Automatic Programming in an Arbitrary Language: Evolving Programs in Grammatical Evolution*. PhD thesis, University of Limerick, 2001.

[30] O'Neill, M., Brabazon, A., (2008) Evolving a Logo Design using Lindenmayer Systems, Postscript and Grammatical Evolution, IEEE Congress on Evolutionary Computation 2008, Hong Kong, China.

[31] O'Neill, M., Brabazon, A. (2006). Grammatical Swarm: The generation of programs by social programming. Natural Computing, Vol.5, No.4, pp.443-462.

[32] O'Neill, M., Brabazon, A. (2006). Grammatical Differential Evolution. In Proceedings of IC-AI, pp.231-236. CSREA Press.

[33] O'Neill, M., Brabazon, A. (2005). mGGA: The meta-Grammar Genetic Algorithm. In LNCS 3447, Proceedings of the European Conference on Genetic Programming EuroGP 2005, pp.311-320, Lausanne, Switzerland. Springer.

[34] O'Neill, M., Ryan, C. (2004). Grammatical Evolution by Grammatical Evolution: The Evolution of Grammar and Genetic Code. In LNCS 3003 Proceedings of the European Conference on Genetic Programming EuroGP 2004, pp.138-149, Coimbra, Portugal. Springer.

[35] O'Neill, M., Ryan, C. (2003). *Grammatical Evolution: Evolutionary Automatic Programming in an Arbitrary Language*. Kluwer Academic Publishers.

[36] O'Neill, M., Ryan, C. (2001). Grammatical Evolution, *IEEE Trans. Evolutionary Computation*. 2001.

[37] O'Neill, M., Ryan, C., Keijzer M., Cattolico M. (2003). Crossover in Grammatical Evolution. *Genetic Programming and Evolvable Machines*, Vol. 4 No. 1. Kluwer Academic Publishers, 2003.

[38] O'Neill M., Hemberg E., Gilligan C., Bartley E., McDermott J., Brabazon A. (2008). GEVA - Grammatical Evolution in Java (v1.0). UCD School of Computer Science Technical Report UCD-CSI-2008-09. Available from `http://ncra.ucd.ie/geva/`.

[39] O'Neill M., Hemberg E., Gilligan C., Bartley E., McDermott J., Brabazon A. (2009). GEVA: Grammatical Evolution in Java. SIGEVOlution, 3(2):17-22.

[40] O'Reilly, U-M., Hemberg M. (2007). Integrating generative growth and evolutionary computation for form exploration. Genetic Programming and Evolvable Machines, Vol. 8, No. 2, pp.163-186. Springer.

[41] Poli R., McPhee N.F., Langdon W.B. (2008). A Field Guide to Genetic Programming. Published via http://lulu.com and freely available at http://www.gp-field-guide.org.uk.

[42] Ryan, C., Collins, J.J., O'Neill, M. (1998). Grammatical Evolution: Evolving Programs for an Arbitrary Language. *Proc. of the First European Workshop on GP*, 83-95, Springer-Verlag.

[43] Sass, L. (2006). Wood Frame Grammar: A generative system for digital fabrication. International Journal of Architectural Computing. 4(1):51-67.

[44] Ratle, A., Sebag, M. (2000). Genetic Programming and Domain Knowledge: Beyond the Limitations of Grammar-Guided Machine Discovery. In LNCS 1917 Proceedings of Parallel Problem Solving from Nature PPSN VI, pp.211-220. Springer.

[45] Stiny, G. (1980). Introduction to shape and shape grammars. Environment and Planning B Vol.7, pp. 349-351.

[46] Stiny, G. (1991) The Algebras of Design. Research in Engineering Design 2(3):171-181.

[47] Stiny, G., Gips, J. (1972). Shape Grammars and the Generative Specification of Painting and Sculpture. In Proceedings of IFIP Congress71, pp.1460-1465. North-Holland.

[48] Stiny, G., Mitchell, W.J. (1978). The Palladian grammar, Environment and Planning B, Vol.5, pp. 5-18.

[49] Takagi, Hideyuki. (2001). Interactive Evolutionary Computation: Fusion of the Capabilities of EC Optimization and Human Evaluation. Proceedings of the IEEE, Vol.89, No,9, pp.1275-1296.

[50] Vogel, C. M., Cagan, J. and Boatwright, P.B.H. (2005). The Design of Things To Come. Prentice Hall.

[51] Whigham, P.A. (1996). Grammatical Bias for Evolutionary Learning. PhD Thesis, University of New South Wales, Australian Defence Force Academy, Canberra, Australia.

[52] Wong, M.L., Leung, K.S. (2000). Data Mining Using Grammar Based Genetic Programming and Applications. Kluwer Academic Publishers.