Combining Structural Analysis and Multi-Objective Criteria for Evolutionary Architectural Design

Jonathan Byrne, Michael Fenton, Erik Hemberg, James McDermott, Michael O'Neill, Elizabeth Shotton, and Ciaran Nally

Natural Computing Research & Applications Group
University College Dublin, Ireland
{jonathanbyrn,michaelfenton1}@gmail.com,
m.oneill@ucd.ie

Abstract. This study evolves and categorises a population of conceptual designs by their ability to handle physical constraints. The design process involves a trade-off between form and function. The aesthetic considerations of the designer are constrained by physical considerations and material cost. In previous work, we developed a design grammar capable of evolving aesthetically pleasing designs through the use of an interactive evolutionary algorithm. This work implements a fitness function capable of applying engineering objectives to automatically evaluate designs and, in turn, reduce the search space that is presented to the user.

1 Introduction

Design can be described as a purposeful yet explorative activity [7]. Initially the designer must explore the search space to find a concept or form that is capable of fulfilling the design specification. Once the form has been chosen, the design process focuses on satisfying the constraints of the original design specification. At the centre of this process there is a conflict between form and function. While these two attributes of a design are not mutually exclusive, there can be a trade off when realising a design.

In this paper we look at the specific case of architectural design. For a structure to be created it requires the combined effort of both architects and engineers. The heuristics an architect uses to evaluate a design are not the same as a structural engineer. Architects evaluate all aspects of the design, from broader issues of internal and external relationships to more detailed aesthetic measures such as material use, texture and light. Engineers evaluate the integrity of the structure itself. To oversimplify, architects are concerned with spaces, engineers are concerned with forces.

This study is a continuation of our previous work that primarily focused on the aesthetic qualities of conceptual design [19]. Through the use of design grammars and an interactive fitness function we have shown that Grammatical Evolution (GE) [18] is capable of creating surprising and innovative designs. Conversely,

C. Di Chio et al. (Eds.): EvoApplications 2011, Part II, LNCS 6625, pp. 204–213, 2011.

the focus of structural evolutionary design has primarily been concerned with engineering constraints. The objective nature of engineering constraints lend themselves to implementation as a fitness function and allow a design to be optimised accordingly [15].

Our work combines the formal approach of architecture with the constraints of engineering. The advantages of this approach are twofold. First, conceptual designs can be optimised to increase their functionality and strength while reducing the amount of material used. This will, in turn, make the designs more credible and realisable. Second, assigning an objective fitness to a design also provides a mechanism for grouping designs. The user may then select which areas of the search space they find the most interesting and thus accelerate convergence on aesthetically pleasing designs.

This paper is organised as follows. Section 2 is a summary of related research in this area. A description of our approach to design generation and analysis is given in Section 3. The two experiments carried out using this system are described in Section 4 and Section 5 and their results are examined. Our conclusions and future work are discussed in Section 6.

2 Previous Work

Computers are ubiquitous in design but they are typically used as an analytical aid rather than as a generative tool. Computer applications are employed after the conceptual design process has been completed. With a few notable exceptions, the computer is not used to explore the search space of possible designs. This section discusses previous work in design generation.

2.1 Conceptual Evolutionary Design

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 [22]. Parametric design tools have now been introduced into more mainstream design software. There is the Grasshopper plug-in for the Rhino modelling system [9] and Bentley Systems have implemented a program called Generative Components [6].

An evolutionary approach to conceptual design exploration is implemented in GENR8 [20]. 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 to a series of metrics, symmetry, undulation, size smoothness, etc. The user is able to weight these metrics according to their preference. Our approach builds on this work. We use a grammar for generating designs and a combination of automatic and users evaluation to drive the evolutionary process.

2.2 Evolutionary Structural Design

Structural engineers seek to find ways for a structure to resist the applied stresses while also reducing material usage and cost. Evolutionary Computation (EC) naturally lends itself to these problems and, accordingly, there has been a large amount of work in this area. Many of the earliest EC applications were focused on optimising structures [15]. The computational cost of structural analysis meant that early papers focused on greatly simplified structures, such as two dimensional trusses [11]. As computational power increased, so did the scope of the applications. Structures such as bridges [25], electricity pylons [23], and even whole buildings [14] have been optimised using EC. A list of applications is covered extensively in the literature of Kicinger [15]. Structural optimisation falls into three categories. The overall layout of the system (topology), the optimal contour for a fixed topology (shape) and the size and dimensions of the components (sizing). Our work focuses on the topological optimisation, although the modular nature of our approach could be adapted for optimising the other categories. This possibility is discussed in greater detail in Section 6.

2.3 Interactive Evolutionary Computation

Interactive Evolutionary Computation (IEC) was developed as a means of assigning fitness when no objective metric could be defined. Human interaction has allowed EC to be applied to problems such as music and computer graphics, and to act as an exploratory tool as opposed to its primary function as an optimiser. A more complete list of interactive applications can be found in [1] and [24].

3 Experimental Setup

Our system is comprised of four parts, an evolutionary algorithm, a design grammar, structural analysis software and a multi-objective fitness function. This section describes our approach to generating and evaluating designs.

3.1 Grammatical Evolution

Grammatical Evolution is an evolutionary algorithm that is based on GP [18]. It differs from standard GP by representing the parse-tree based structure of GP as a linear genome. It accomplishes this by using a Genotype-Phenotype mapping of a chromosome represented by a variable length bit or integer string. The chromosome is made up of codons eg:(integer based blocks). Each codon in the string is used to select a production rule from a Backus Naur Form(BNF) grammar. Production rules are selected from the grammar until all non-terminal rules are mapped and a complete program is generated. The advantage of using a grammar is that it is possible to generate anything that can be described as a set of rules. Grammars are capable of generating strings, mathematical formulas, pieces of programming code and even whole programs. The grammar used for our experiments is described in Section 3.2. Another advantage of applying

GE to design is that generative processes, like the mapping process in GE, are required for the production of designs that can scale to a high level of complexity [13].

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. In our previous experiment, evaluation was provided solely from user interaction. The grammar was created with no consideration for the structural soundness of the resulting bridges. Despite this, it was possible to compare the relative performance of bridges in the grammar by applying a pre-determined loading.

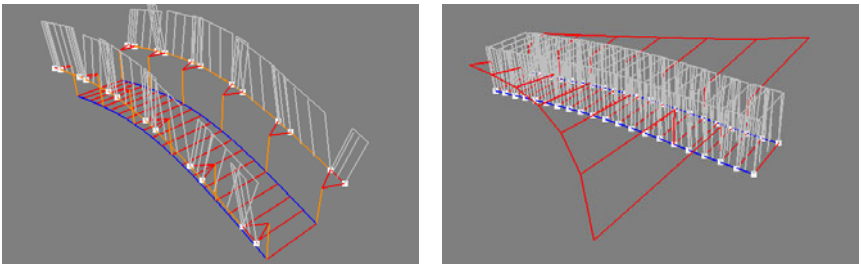
The size of the grammar meant that it could not be appended to the paper. The grammar is available online at [16]. The grammar creates graphs using networkx [10], a python class for studying complex graphs and networks. Three desirable characteristics for a design generator are modularity, regularity and hierarchy [12]. We implement these characteristics using the novel method of higher order functions. Our work in this area is discussed in greater detail in [17]. For structural analysis to be performed on the bridges, a mechanism was required for specifying the loads on the structure. Our approach was to add attributes to the existing grammar. This allowed us to label components depending on the function that created them. Labelling meant that forces could be assigned to the structures automatically and accordingly, that different forces could be applied to different parts of the bridge. An example of this can be seen in Figure 1.

3.3 Structural Analysis

The ability to analyse structures as computable models is achieved by using Finite Element Methods [8]. Instead of calculating the partial differential equation for a whole design, a continuous structure is discretised into an approximating system of ordinary differential equations. The approximation can then be solved using numerical approximation methods for differentiation such as Euler's method or the Rung-Kutta method. Our designs are particularly suited to Finite Element Analysis (FEA) as the structures are already discretised into a series of interconnected beams. To analyse our designs we are using San Le's Free Finite Element Analysis (SLFFEA) [21]. This software is freely available for download and has been used by engineering companies in industry.

3.4 Multi-Objective Fitness Function

Design usually involves satisfying several (possibly contradictory) objectives. Multi-objective evolutionary algorithms (MOEAs) have been shown to be a useful approach for finding the best compromise when tackling a multi-objective problem [26]. Instead of weighting the objectives and allowing an evolutionary



(a) Force being applied to the handrail (b) Force being applied to the walkway

Fig. 1. Different Magnitudes of Stresses Being Applied to the Handrail and Walkway

algorithm to converge on a single global optimum, the algorithm builds a pareto-front of the individuals that maximise the given objectives. Using fronts can aid the design process by presenting the user with several pareto-equivalent designs and letting them select the design that most closely matches their requirements. We are using a GE implementation of the NSGA2 algorithm [3] as our selection mechanism. The algorithm uses a fast non-dominated sorting algorithm to calculate which individuals are on the front and then group the other individuals in the population relative to this group. Normally MOEA applications are only concerned with the individuals on the pareto-front, we intend to investigate in Section 5 whether the grouping property of the NSGA2 algorithm could also be of benefit for guiding the search process.

4 Optimising Designs Using Structural Analysis

This experiment aimed to test whether an evolutionary search was capable of generating designs that minimised the stress in a structure and reduced the amount of material used. It was carried out using the implementation described in Section 3 and the bridge grammar described in Section 3.2. The experimental settings were: Population size = 100, Generations = 50, No. of Runs = 30, Mutation Rate = 1.5%, Crossover Rate = 70%, Selection Scheme = Tournament, Tournament Size = 3, Replacement Scheme = NSGA2.

The material from which the bridge was constructed was small scale air dried oak sections with a moisture content of 20% or more. The structural qualities of this wood were taken from the British Standards BS-EN-338-2003 as a grade D30 class of timber [2]. The material qualities were then assigned to the bridge beams for SLFFEA analysis. For stresses on a structure to be calculated, we must first assign fixed points and loaded beams. Normally this is done manually by the user. Our approach automated this by using attributes in the grammar, as described in Section 3.2. The bridges were subjected to a uniformly distributed load (UDL) of 5kN/m upon the walkway itself and a separate 1kN/m load was applied to the handrails. The loads for the bridge were taken from [4].

While we tried to replicate a load that a bridge might be subjected to during actual usage, the main purpose was to compare how well the bridges performed relative to other bridges generated by the design grammar.

There were two constraints placed on the designs, one of which was stress based and one that was based on material usage. The stress constraint in the fitness function calculated the maximum stress on each beam in the design, this was then averaged over the whole structure and the selection pressure aimed at reducing it. If a beam failed then the bridge was assigned a default fitness of 100,000. This meant that high stress designs were removed from the population and the fitness pressure aimed at reduced stresses over the structure as a whole. The material constraint aimed at reducing the number of beams used in a structure. This fitness metric is opposed to the stress constraint as one method for reducing the average stress on the beams is by adding more unnecessary beams. By adding a penalty for the total weight of material used, it can force the algorithm to simplify the design. Reducing the total weight of material used also translates into direct savings when manufacturing an instance of the design.

4.1 Optimisation Results

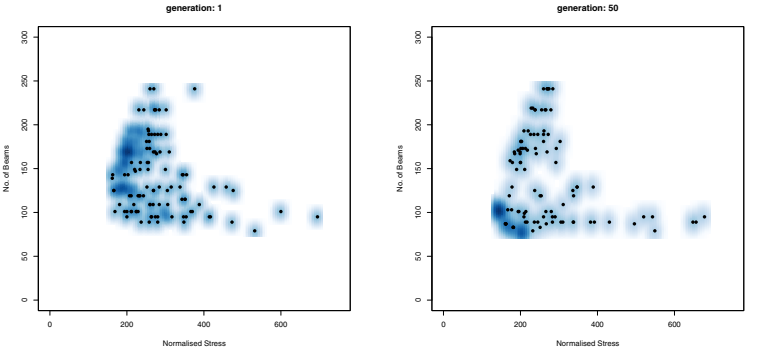
The results for the experiment are shown in Figures 2 and 3. It is clear that the fronts are moving toward a pareto-optimality over the course of 50 generations, as shown in Figure 2. There is a 43% reduction in the material used (Figure 3(a)) and a reduction of the average maximum stress placed on the structure of 41% (Figure 3(b)) after 50 generations. The results show that using structural analysis and an MOEA can significantly reduce the stresses and self weight of a design.

The results show that our system is capable of evolving structures that increasingly satisfy the constraints specified in our multi-objective fitness function. This is very important for trying to move a design from a mere concept to something that could be actualised. This is a challenge that faces engineers and architects on a daily basis and a GE based approach such as this has the potential to help solve this problem.

5 Categorising Designs Using Structural Analysis

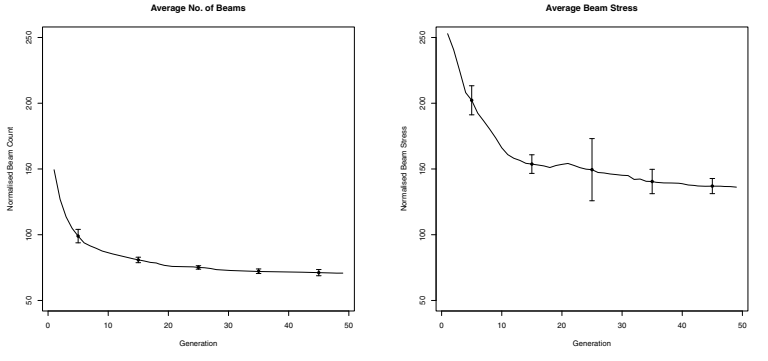
Our intention in implementing this software is not to exclusively optimise designs but to allow the architect to interactively evolve designs that they find aesthetically pleasing. To this end, it is imperative to bias the algorithm towards designs that the user finds interesting and appealing. The design process is not about optimisation and, as such, designers are often interested in designs that do not lie on the pareto front.

In this experiment we used the settings described previously except that we only allowed each run to be executed for a single generation. Instead of using the NSGA2 algorithm to optimise the bridge designs, it is used to group the bridge designs by realisability. The grouping created by the fast non-dominated sort are shown in different colors in Figure 5. The user selects the design grouping they



(a) Generation 1 (b) Generation 50

Fig. 2. Scatter plot with a density estimation function that shows the progression of front over 50 generations



(a) Normalised Average of Beams Used (b) Normalised Average of Stress

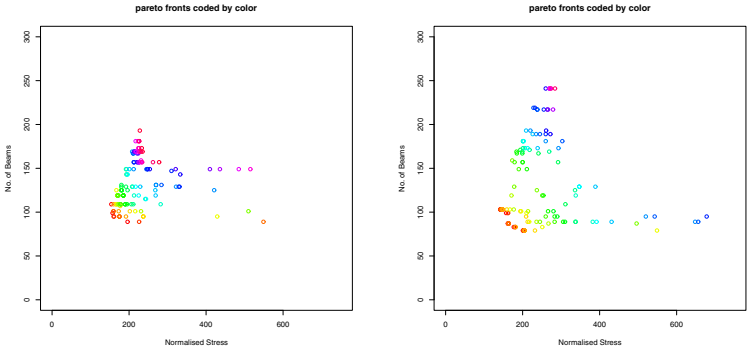
Fig. 3. The fitness minimisation average of the population over 50 generations

find the most interesting and so direct the pareto-front using selection pressure. By categorising designs by their place on the fitness landscape we can accelerate convergence onto more appealing areas of the search space.

In our experiment, we randomly selected designs from the first two fronts and the last two non-empty fronts.To generate images of the designs we used an open source mesh viewer developed by the INRIA called medit [5]. An online survey was then conducted on the designs. The survey consisted of presenting two designs,side by side, and asking the user to select which design they found most aesthetically pleasing, as shown in Figure 4. If you wish to see more of the designs, there is a link to the survey at [16]. If the user had no preference for a particular design they can indicate this with the no preference button. The presentation of the images were randomised so that there was no bias for which

(a) (b) (c) (d)

Fig. 4. Sample bridges from the survey and survey layout (d)



(a) (b)

Fig. 5. Color differentiated fronts produced by the NSGA2 fast non-dominated sorting algorithm at generation 1 (a) and generation 50 (b)

side the images appear on. The survey was carried out by post-graduate students and volunteers from the school of architecture. This survey was authorised by the ethics committee and the head of the school of Architecture.

5.1 Categorisation Results

The survey was completed by 28 individuals and consisted of 2800 evaluations. The users showed a preference of 55.9% for bridges from the end of the non-dominated sort compared to a 36.84% preference for non-dominated bridges from the front of the sort. The users had no preference on 7.26% of the designs. This shows an aesthetic preference for designs that do not fulfill the engineering constraints. The results imply that the engineering constraints that we chose for this experiment are in opposition to aesthetic preferences. Although aesthetic preference is a purely subjective quality, the inclination towards unconstrained designs could be because of their unusual and unexpected configurations rather than the "ordinary" nature of structurally sound designs.What is most interesting about this result was that there was no selection pressure on the population. The fast non-dominating sort was applied to randomly generated individuals.

As can be seen in Figure 5(a) compared to 5(b), this is the worst case scenario. If a population was evolved over several generations the difference between individuals greatly increases, which would aid categorisation of the population.

The results indicate that the fast non-dominated sort in the NSGA2 algorithm has a secondary purpose; It can be used to group designs by how well they meet the objectives. This mechanism could greatly speed up IEC by allowing the user to choose between groups rather than selecting individuals.

6 Conclusion and Future Work

In this paper we encoded material and physical constraints into a fitness function and showed conceptual designs could be evolved towards those objectives. This is step towards making conceptual designs more realisable. We also showed that multi-objective fitness functions could be used for more than optimisation. By automatically categorising the designs and then presenting those categories to a user for evaluation, the MOEA could drastically reduce the search space presented to the user during IEC.

Our future work intends to encode other aesthetic constraints such as smoothness, curvature, etc and allow the user to select objectives that they would most like to see in the presented design. The modular structure of our software makes it possible to lock the topology of a chosen bridge design and focus on the optimisation of either the shape of the beams or the sizing and material the beams are constructed from.

Acknowledgments

We would like to thank Andrea McMahon and Brian Lynch for their unceasing support while writing this paper. This research is based upon works supported by the Science Foundation Ireland under Grant No. 08/RFP/CMS1115 and the Graduate Research Education Programme in Sustainable Development, jointly funded by IRCSET and IRCHSS.

References

1. Banzhaf, W.: Interactive evolution. In: Back, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation, ch. C2.9, pp. 1–6. IOP Publishing Ltd., Oxford University Press (1997)
2. British Standards Institution: BS EN 338-2003: Structural Timber Strength Classes. BSI, London (2003)
3. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2), 182–197 (2002)
4. Fenton, M.: Analysis of Timber Structures Created Using A G.E-Based Architectural Design Tool. Master's thesis, University College Dublin, Ireland (2010)
5. Frey, P.J.: MEDIT:interactive mesh visualization. 0 RT-0253, INRIA (December 2001), http://hal.inria.fr/inria-00069921/en/

6. Generative Components, http://www.bentley.com/getgc/
7. Gero, J.S.: Creativity, emergence and evolution in design. Knowledge-Based Systems 9(7), 435–448 (1996)
8. Glaylord, E., Glaylord, C.: Structural engineering handbook. McGraw-Hill, New York (1979)
9. Grasshopper, Generative Modeling with Rhino, http://www.grasshopper3d.com/
10. 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)
11. Hoeffler, A., Leysner, U., Weidermann, J.: Optimization of the layout of trusses combining strategies based on Mitchels theorem and on biological principles of evolution. In: Proceeding of the 2nd Symposium on Structural Optimisation, Milan, Italy (1973)
12. Hornby, G.S.: Measuring, enabling and comparing modularity, regularity and hierarchy in evolutionary design. In: Proceedings of GECCO 2005 (2005)
13. Hornby, G.S., Pollack, J.B.: The advantages of generative grammatical encodings for physical design. In: Proceedings of the 2001 Congress on Evolutionary Computation CEC 2001, pp. 600–607. IEEE Press, Los Alamitos (2001)
14. Kicinger, R., Arciszewski, T., DeJong, K.: Evolutionary design of steel structures in tall buildings. Journal of Computing in Civil Engineering 19(3), 223–238 (2005)
15. Kicinger, R., Arciszewski, T., Jong, K.D.: Evolutionary computation and structural design: A survey of the state-of-the-art. Computers and Structures 83(23-24), 1943–1978 (2005)
16. Link to the bridge grammar, http://i.imgur.com/0vsDh.png
17. 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)
18. O'Neill, M.: Automatic Programming in an Arbitrary Language: Evolving Programs with Grammatical Evolution. Ph.D. thesis, University Of Limerick, Ireland (2001)
19. O'Neill, M., McDermott, J., Swafford, J.M., Byrne, J., Hemberg, E., Shotton, E., McNally, C., Brabazon, A., Hemberg, M.: Evolutionary design using grammatical evolution and shape grammars: Designing a shelter. International Journal of Design Engineering (in press)
20. 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
21. San Lee's Free Finite Element Analysis, http://slffea.sourceforge.net/
22. Shea, K., Aish, R., Gourtovaia, M.: Towards integrated performance-driven generative design tools. Automation in Construction 14(2), 253–264 (2005)
23. Shea, K., Smith, I., et al.: Improving full-scale transmission tower design through topology and shape optimization. Journal of Structural Engineering 132, 781 (2006)
24. Takagi, H.: Interactive evolutionary computation: Fusion of the capabilities of EC optimization and human evaluation. Proc. of the IEEE 89(9), 1275–1296 (2001)
25. Topping, B., Leite, J.: Parallel genetic models for structural optimization. Engineering Optimization 31(1), 65–99 (1988)
26. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans. Evolutionary Computation 3(4), 257–271 (1999)