METAHEURISTICS IN ARCHITECTURE

T. Strobbe[1], P. Pauwels[1], R. Verstraeten[1] and R. De Meyer[1]

[1] Ghent University, Belgium

Abstract This paper outlines our findings concerning the use of constraints as design drivers in a design exploration process and investigates a possible application of a heuristic search and optimization method in architecture as a means for constraint solving. Fundamental theoretical research will cover these two aspects, accompanied with an appropriate test-case.

Keywords CAD, Generative design, Metaheuristics, Genetic Algorithm

1 INTRODUCTION

Computer-aided design (CAD) has been extended in various ways during the last decades. This evolution resulted in worldwide adoption in the domain of architecture, engineering and construction (AEC), making CAD systems an essential tool for AEC specialists. However, research has shown that recent developments in CAD, such as building information modelling (BIM) and simulation-based design, have mainly affected the later stages of the design process [1]. In these stages, the designer's concept is already fixed and the influence of recent CAD tools is then limited to very specific fields, most often related to building performance only. However, there is an increasing demand for modelling tools that allow the designer to explore essential variations in an early outlet and scheme design phase [2][3].

The paper is organized as follows. Section 2 presents a summary of existing applications of metaheuristics in architectural design and outlines the research gap. Section 3 provides a study that reviews genetic algorithms. Section 4 presents our proposed design method, accompanied with a simple test case. Section 5 concludes the paper.

2 METAHEURISTICS FOR ARCHITECTURAL DESIGN

2.1 The usage of constraints in generative design systems

A generative design system, operating on top of a parametric CAD system, may provide this kind of functionality, as it enables the designer to produce a solution space instead of one single solution by defining parameters, relations and constraints. This relatively new technique has been introduced in some of the world's leading architectural practices and academic institutions [4]. The design process starts with an open-ended and multidisciplinary design exploration phase and progresses towards a more specific design solution in the later design phases. Most important in the design exploration process is the notion of 'constraints': they form the boundaries between which a design solution is to be found.

This approach lies at the basis of research initiatives that suggest methods to narrow the solution space by implementing constraints related to performance criteria. One of these initiatives is *GENE_ARCH* [5], a design system in which a generative model is combined with multidisciplinary building simulation software (DOE2.1E), evaluating thermal performance and lighting analysis. "*It was developed to help architects in the creation of energy-efficient and sustainable architectural solutions, by using goal-oriented design, a method that allows to set goals for a building's performance.*" [5]. The software was tested by L. Caldas on actual buildings with a considerable complexity and has proven to be reliable [5].

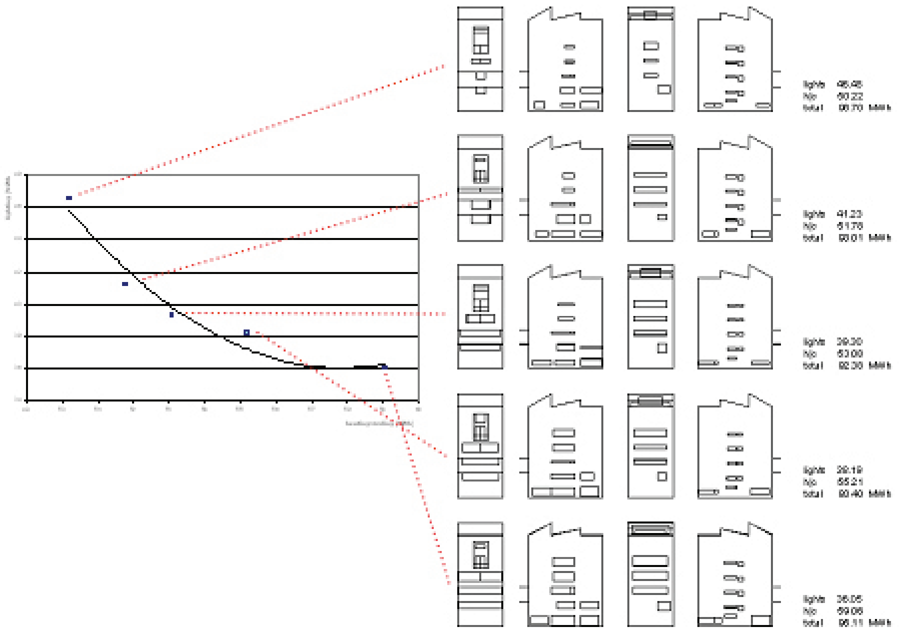

Figure 1. Optimal solutions for Àlvaro Siza's School of Architecture at Oporto using *GENE_ARCH* (Caldas, 2007).

Similar precedents can be found in the field of structural optimisation. A generative structural design system, *eifForm*, is combined with an associative modelling tool, *GenerativeComponents*, using eXtensible Markup Language (XML) as a modelling language for integrating the two tools [6]. A CAD based example that can be used during all stages of the design development process is the *Generative Design Method* (GDM) [7].

Firstly, it is possible to further develop such first initiatives by considering that constraints are not limited to purely geometric requirements, but might as well have topological, material or functional characteristics. A designer has to work within the constraints of the client's budget, brief and government regulations, to accomplish the best compromise from a wide range of design solutions. Secondly, constraints are often considered as restrictive factors in the design, but previous research has shown that constraints can play the role of design drivers as well [8]. This ambiguity can lead to constraints evolving from a limitation in the design process to an effective way for driving the solution space to innovative design solutions.

2.2 Metaheuristics for constraint solving

The design process can be specified as a generation process resulting from the well-defined description of its constraints. To ensure that the best compromise from a wide range of design variations can be found and the designer can find the most eligible outcome, an efficient exploration of the heterogeneous solution space is essential. This solution space can be represented by a performance landscape, in which performance is defined as a predetermined quality, resulting from the combination of several parameters.

Due to the high level of complexity and the vast amount of information, this solution space exploration can be formulated as a combinatorial optimization problem, a topic discussed in computer science and mathematics [9]. Such problems are concerned with the efficient exploration of a discrete set of possible solutions. They can be addressed by the use of a computational optimization method or 'metaheuristic', operating on a population of possible solutions and iteratively trying to improve a candidate solution to meet a desired objective.

The field of metaheuristics is a rapidly growing field of research, due to the importance of combinatorial optimization problems in several disciplines. Also, they are believed to play a more effective role in the future of architecture, as stated in [10]. Some of the most used metaheuristics are '*simulated annealing*' [11], '*ant colony optimization*' [12], '*particle swarm optimization*' or '*neural networks*' [13] and '*genetic algorithms*' (GA) [14]. If the solutions of the optimization problems can be encoded in linear-strings, they can be solved by genetic algorithms. We will discuss such GA's in the remainder of this paper.

3 HOW TO OPTIMIZE YOUR DESIGN WITH GENETIC ALGORITHMS

3.1 Genetic Algorithms

GA's are based on the principles of natural evolution: the inherited characteristics within a population change over generations due to genetic variation, inheritance and natural selection [14]. Some individuals have an advantageous characteristic and therefore have a greater chance of survival making this advantageous characteristic occur more often in the following generations. Also, new characteristics are created by mutation and recombination (genetic operations). Genetic Algorithms use similar principles applied to a population of feasible design solutions. Traditionally, a GA consists of the four following key components [14].

1. Initialization

Initialization consists of the generation of a random population of x individuals.

2. Selection

Selection consists of selecting a part of the population that will evolve to the next generation. The probability that an individual is selected is proportional to his relative fitness, making it more likely that fitter individuals can reproduce. This selection method is called roulettewheel selection or fitness proportionate selection.

3. Reproduction

Reproduction consists of selecting two individuals that are combined through genetic operations such as crossover and mutation. Crossover consists of the exchange of genomes between two individuals which creates offspring. Through crossover the population evolves towards potentially interesting regions of the solution space. Mutation consists of randomly modifying a small part of the new offspring. In this way genetic diversity is maintained, which avoids convergence to local optima.

4. Termination

The iterative process of initialization, selection and reproduction is stopped once a termination condition is reached. This condition can be defined as a maximum number of generations or an acceptable fitness.

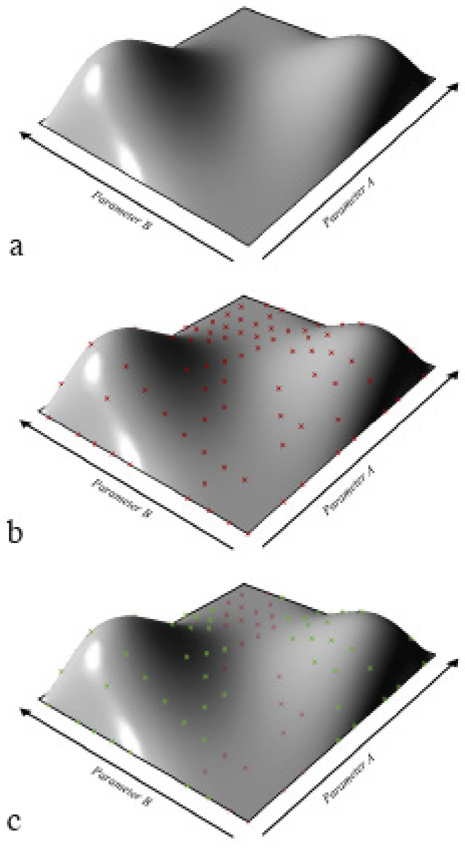

Figure 2. Genetic algorithm process: (a) performance landscape consisting of 2 parameters; (b) random population of *x* individuals; (c) selection of individuals using fitness proportionate selection.

Previous research has already illustrated the large number of applications of GA's in several disciplines [10]. In contrast to what happens in these other disciplines, the usage of evolutionary techniques in the AEC domain is not generally known. Nevertheless, pioneering work of Frazer proposes already the concept of evolutionary architecture as a form of artificial life, subject to evolutionary processes in response to the user and the environment [15]. Also, recent theoretical experiments show increasing interest in the topic of evolutionary algorithms in architectural design [5][7][16]. One theoretical example that explores the use of GA's in architectural design is '*EvoArch*' [16]. This tool is concerned with the architectural layout design problem, which is the finding of the best adjacencies between functional spaces under given constraints. An evolutionary algorithm is proposed to solve this combinatorial optimization problem.

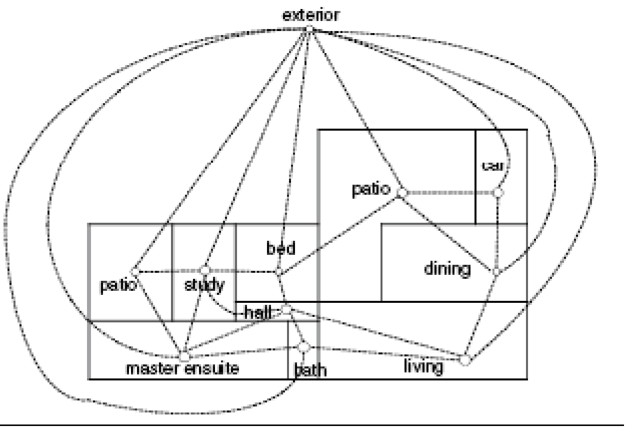

Figure 3. Graph representing architectural space topology of the floorplan created using '*EvoArch*' (Wong, 2009).

3.2 Multidisciplinary Design Optimization

It is possible to extend this concept of metaheuristic optimization methods by combining a number of different solution spaces. This method, applied in other engineering fields such as automobile, aircraft and spacecraft design, is called '*multidisciplinary design optimization*' (MDO). This technique allows designers to simultaneously incorporate several relevant disciplines (structural and thermal analysis, government regulations, economics, ...). The combination of metaheuristics and MDO can increase the efficiency of the design exploration, by taking into account the interactions between the different disciplines. This technique has the potential to solve complex coupled systems by exploring these interrelated disciplines.

4 PROPOSED APPROACH

This paper proposes a designer-driven generative design method applied to the architectural design process and consisting of the following elements:

1. Definition of constraints that have a decisive influence on the architectural design. Constraints are not limited to geometric requirements, but can cover a whole area of different fields.

2. Exploring the solution space, defined by the constraints, using metaheuristics search and optimization methods or '*multidisciplinary design optimization*' (MDO) and allowing the designer to interfere with the process by altering parameters.

3. Selection of the most desirable design solution.

It is important to highlight the central role of (1) the constraints that drive the design and (2) the designer, who can modify the generative design process based on the resultant outcomes by altering the parameters. Besides a design generating tool, the focus is also on constraint evaluating.

In order to test this proposed design method, a number of simple experiments are discussed. These experiments relate to problems where geometry is being generated using evolutionary techniques. These techniques are implemented in a common architectural CAD system, *Rhinoceros* [17], using *Grasshopper* [18], a graphical algorithm editor tightly integrated with *Rhino*'s 3d modelling tools. *Grasshopper* provides an evolutionary solving function, *Galapagos*, which is used to search and optimize the generated design solutions.

4.1 Experiment 1: one fitness function

The goal is to generate a *NURBS* surface using evolutionary techniques by defining a specific set of constraints. The constraints in this first experiment are: (1) four fixed points (invariable) that are part of the surface and (2) two attractor points to which the surface must evolve. The single fitness function is calculated by the distance between the surface and the attractor points. The results of the evolutionary solver after 15 generations (n=15) can be found in the following table.

Fitness: distance from surface to points							
n	0	1	2	3	5	10	15
distance [-]	459,67	442,71	464,06	312,17	232,16	58,65	10,23
curvature [-]	533,11	593,71	385,47	363,21	563,02	775,61	719,34

Figure 4. Distance from surface to attractor points (*n*=0...15).

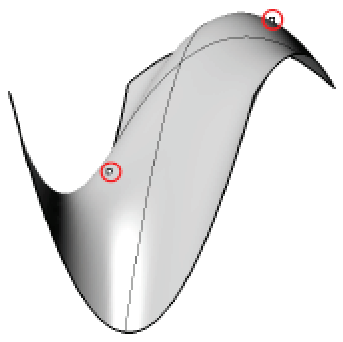

Figure 5. Surface generated using an evolutionary solver in '*Grasshopper*' (n=15).

An acceptable design solution was found relatively quick after 15 generations. As demonstrated in table 1, an exponential progression towards this optimal solution can be observed. Figure 5 shows one of the many possible design solutions, but more constraints are necessary to further narrow the solution space.

4.2 Experiment 2: two fitness functions

A designer can decide that the previously generated design solution is not desirable, because the surface is too strongly curved. Therefore, a second fitness function is integrated, the mean curvature. The total fitness function is implemented as follows.

$$Fitness = d^a x c^b$$

d= total distance between surface and attractor points.

c= mean curvature of the surface.

a,b= correction factors to implement the designer's preference.

Fitness: distance from surface to points & Curvature							
n	0	1	2	3	5	10	15
distance [-]	479,42	404,75	515,11	431,11	321,53	315,88	263,33
curvature [-]	630,58	592,98	477,64	382,98	344,31	230,22	254,89

Figure 6. Distance from surface to points and mean curvature of surface (n=0...15).

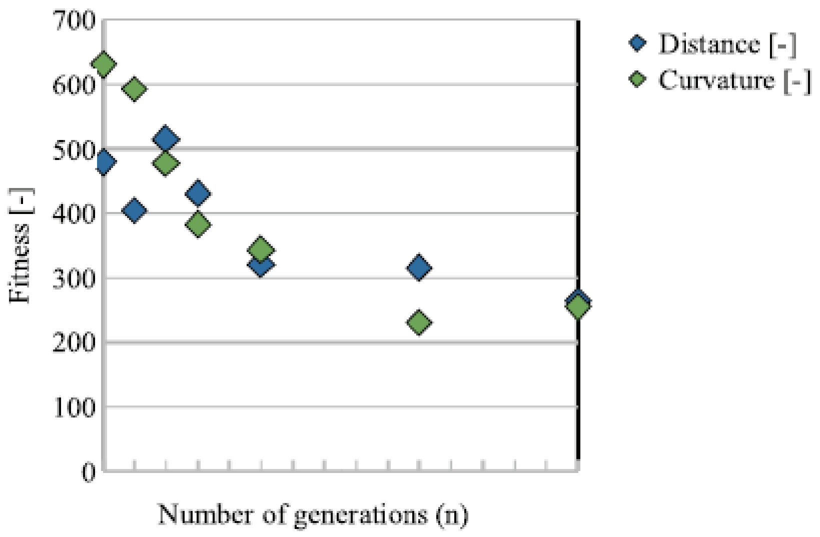

Figure 7. Distance from surface to points and mean curvature of surface (n=0...15).

The final generated surface shows a balanced compromise where both the mean curvature and the distance between the surface and the attractor points are minimized in a relatively quick time. A visualization of this process is demonstrated in figure 8 below.

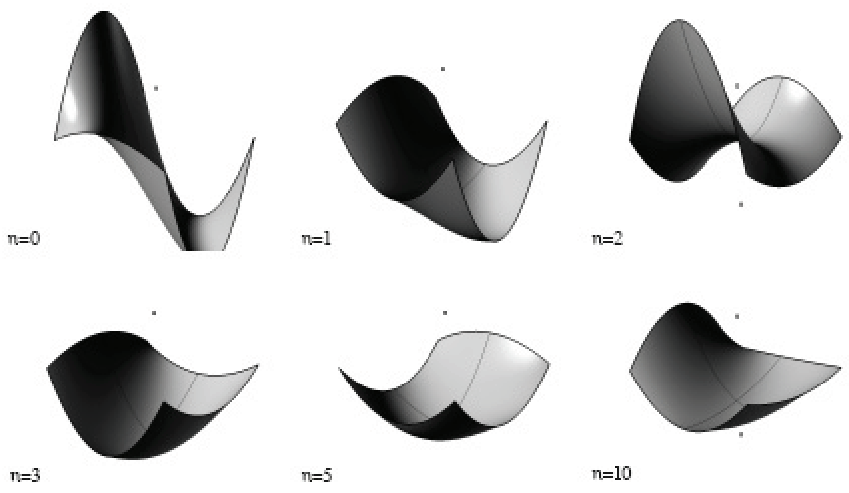

Figure 8. Surfaces generated using an evolutionary solver in '*Grasshopper*' (*n*=0...10)

5 CONCLUSIONS AND FUTURE RESEARCH

This paper describes a design methodology using constraints and a constraint solver to generate design solutions. Theoretical research and a simple test case has proven this a valid method, using existing techniques from other disciplines applied to architectural design. Our goal is to extend these first experiments by considering that constraints are not limited to purely geometric requirements, but might as well have topological, material or functional characteristics.

The final aim is to develop a twofold set of tools that assist the designer in exploring a wide range of design solutions, focusing on both design (1) evaluating and (2) generating methods. Firstly, a method for constraint evaluating is proposed as a way to generate a database of feasible design solutions. This database can be a useful tool for designers to find valuable precedents. Secondly, a designer-driven tool will be developed that can generate possible design variations during the early architectural design phase. This tool does not aim at the automation of the design process, but allows the designer to take into account different design variations, by describing the constraints between which the design solution can be found.

6 REFERENCES

[1] Penttilä H. (2006). Describing the changes in architectural information technology to understand design complexity and free-form architectural expression. ITcon Vol. 11, Special Issue The Effects of CAD on Building Form and Design Quality, pg. 395-408.

[2] Kasik D., Buxton W., Ferguson D. (2005). Ten CAD challenges. In IEEE Computer Graphics and Applications, pg. 81-92.

[3] Salim F. & Burry J. (2009). Software openness: Evaluating parameters of parametric modelling tools to support creativity and multidisciplinary design integration. Spatial Information Architecture Laboratory (SIAL), RMIT University, Melbourne.

[4] Peters B. & De Kestelier X. (2006). The Work of Foster and Partners Specialist Modelling Group. In Proceedings of the Bridges Conference: Mathematical Connections in Art, Music and Science.

[5] Caldas L. (2007). Generation of energy-efficiënt architecture solutions applying GENE_ARCH: An evolution-based generative design system. Advanced Engineering Informatics 22, pg. 59-70.

[6] Shea K., Aish R., Gourtovaia M. (2003). Towards integrated performancedriven generative design tools. Automation in Construction, Volume 14, Issue 2, Education and Research in Computer Aided Architectural Design in Europe (eCAADe 2003), Digital Design, pg. 253-264.

[7] Krish S. (2010). A practical generative design method. Computer-Aided Design, Volume 43, Issue 1, January 2011, pg. 88-100.

[8] Kilian A. (2006). Design exploration through bidirectional modeling of constraints. Department of Architecture, MIT, Massachusetts.

[9] Schrijver A. (2010). A course in Combinatorial Optimization. Department of Mathematics, University of Amsterdam.

[10] Fasoulaki E. (2007). Genetic algorithms in architecture: a necessity or a trend?. In Generative Art Conference 2007.

[11] Kirkpatrick S., Gelatt C., Vecchi M. (1983). Optimization by Simulated Annealing. Science 220 (4598), pg. 671-680.

[12] Dorigo M. (1992). Optimization, Learning and Natural Algorithms. Politecnico di Milano.

[13] Kennedy J., Eberhart R. (1995). Particle Swarm Optimization. In Proceedings of IEEE International Conference on Neural Networks, pg. 1942-1948.

[14] Holland J. (1975). Adaption in natural and artificial systems. University of Michigan Press.

[15] Frazer J. (1995). An evolutionary Architecture. Architectural Association Publications.

[16] Wong S., Chan K. (2009). EvoArch: An evolutionary algorithm for architectural layout design. Computer-Aided Design, Volume 41, pg. 649-667.

[17] Rhinoceros. Modelling tools for designers. Available from: http://www.rhino3d.com (accessed 6th January 2011).

[18] Grasshopper. Generative modelling for Rhino. Available from: http://www.grasshopper3d.com (accessed 6th January 2011).