Evolution of Sun-Shades Outside Building Façades

DISSERTATION SUBMITTED FOR THE PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF PHILOSOPHY IN INFORMATION TECHNOLOGY IN THE DEPARTMENT OF COMPUTER SCIENCE AT THE UNIVERSITY OF CAPE TOWN

by

Leon Coetzee

Supervised by:

Dr Geoff Nitschke

January 2023

Contents

1 Abstract

The research objective behind this study is to compare 'traditional' architectural sun-shades with evolved sun-shades to determine which best blocks direct sunlight from entering a window. Two geographical locations are tested along with two façade conditions for each. The sun path on the summer solstice provides the projected sun rays, measured every fifteen seconds. The sun-shades are made up of points in three-dimensional space that form a 'point cloud'. The points can be connected to form a surface and from there a geometric form.

An Evolutionary Strategy, using self-adaptation, evolves the points within the point cloud to generate the sun-shade.

Fitness for each point is determined by the number of sun rays the point can block from striking the window surface; furthermore, the point may not obstruct the view from the window given certain conditions. The mean fitness for ten 'traditional' architectural sun-shade solutions, represented as point clouds, is compared to the mean fitness of the evolved sun-shade point cloud.

This study provides two contributions to this field; firstly it provides a method with which to measure the fitness of 'traditional' sun-shades solutions and compares them with evolved solutions, secondly it provides a form for the solution. Architecturally, the form this evolved sun-shade takes becomes interesting as an Evolutionary Algorithm is employed as an approach to form-finding.

Finally, some possible improvements and modifications are further discussed.

Acknowledgements

I would like to thank:

- My Supervisor, Dr Geoff Nitschke, for his patience and guidance.
- My wife, Maggie for her mathematical advice and support together with our children Maia and Louie.
- My brother, Derek and Vlad Constantinescu, for their assistance using R.
- My work colleagues for their understanding and input.

2 Introduction

This case study explores using an Evolutionary Algorithm to design an optimum sun-shading device for application in Architecture. Sun-shading devices are external elements on a building façade that typically shield a window opening with the intention of blocking direct sunlight from entering and warming the interior. To assist in keeping the work focused on evolutionary computation, the scope of the study shall be limited to only consider the successful reduction of solar radiation on a surface. This research does not consider light quality or comfort levels (glare) for building occupants. Over time, various approaches and solutions have been shown to produce effective devices for certain conditions. Both how to approach this design task (with assistance on how to analyse the conditions under which such a device should function) and how to select a solution from an existing set of 'typical' sun-shades have been well documented and are typically used as architectural 'precedent studies' when determining a solution. This case study sees these solutions as 'traditional' sun-shades, examples of which can be seen in Figure 1.

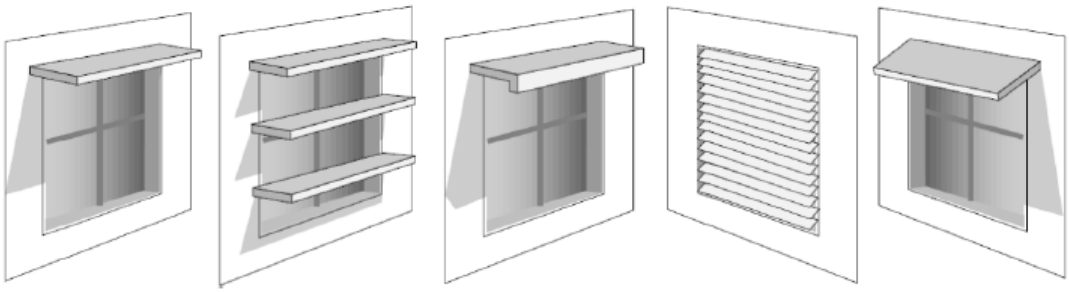


Figure 1: Traditional sun-shades proposed in 'Tips for Daylighting with Windows' (O'Conner et al.)

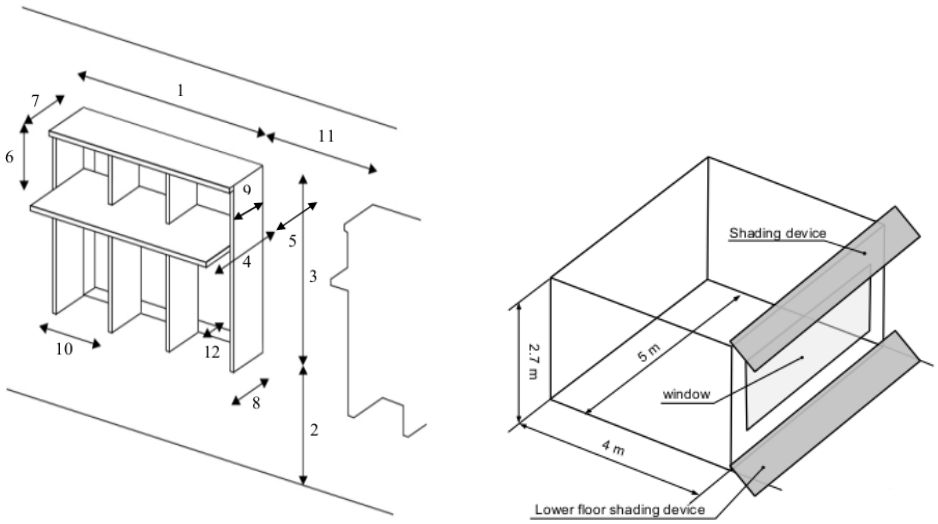
Current research of a similar scope as that carried out in this case study considers, among others; façade design optimisation, the optimisation of a given (pre-designed) shading device, the geometric optimisation of fenestration (window layout) within a façade, the optimal design of a building envelope[i], the optimisation of the shape of a free-form building and the optimised design of an energy efficient building façade.

Façade design optimisation: The optimisation of a façade design seeks to balance allowing natural daylight to enter the building (to provide adequate internal lighting) while also minimising the resultant glare which would cause discomfort for the users. This research is based around creating a natural daylight system with the hope that this can reduce the energy costs of the building. The optimisation solution is centred on the design of a window with integrated sun-shades and uses a simulation of sky brightness to determine internal luminance values.

[i]The Building Envelope can be seen as the element that separates the interior from the exterior of a building. The envelopes' ability to function as a barrier to better manage the internal climate of a building is what is of interest.

Optimisation of a (given) shading device: The optimisation of a given sun-shading device looks to address the increase in heat build up within a building while being restricted in following existing building regulation guidelines that concern the placement together with a broad outline of a device but is lacking in details concerning the choice of sun-shades that may be used. This research does not alter the form of the device, instead, using the results from an energy simulation based on readings from within a test space, the existing device is 'optimised' by altering the position of components (relative to the building surface) - how the device may be rotated and by how wide or thick the components may be (this needs to fit within the building regulations).

Geometric optimisation of fenestration: The geometric optimisation of windows on a façade is inspired by the fenestration of Notre Dame du Haut - Ronchamp Chapel by Le Corbusier. The aim of this study being to reduce the size of window openings on a façade to minimise heat build up within. This work looks at optimising the design of a window layout by affecting the shape, amount and position of window openings. The façade is divided into cells which drive the optimisation of openings, however, certain constraints are implemented (window opening aspect ratios and amounts) to allow the designer to exercise control over the solution. Examples of façade and sun-shading device optimisation approaches can be seen in Figure 2.



(a) Façade design optimisation for daylight (Torres & Sakamoto)

(b) External shading device optimisation (Manzan & Pinto)

Figure 2: Optimisation of sun-shading façade and devices

Optimal design of a building envelope: Optimum building envelope design is broader in scale - the building as a whole is considered within an environment. Typically software simulation is used to mimic the performance of a building, the results of which can be used to create a 'Green Building' (a high performance, energy efficient, low environmental impact building). Given both the scale of the work (the entire building) and the many variables to be considered in the optimisation solution, this research

tends to be rich and complex.

Optimisation of a free-form building shape: Research on the shape optimisation of free-form buildings hopes to increase solar radiation within the building, set in a cold climate, by optimising the shape of a free-form building where solar radiation, the 'shape coefficient' and 'space efficiency' are considered. Parametric[ii] modelling creates the free-form building model while a genetic algorithm is used to maximise solar radiation gain and space efficiency while minimising the shape coefficient[iii] this can be visualised in Figure 3.

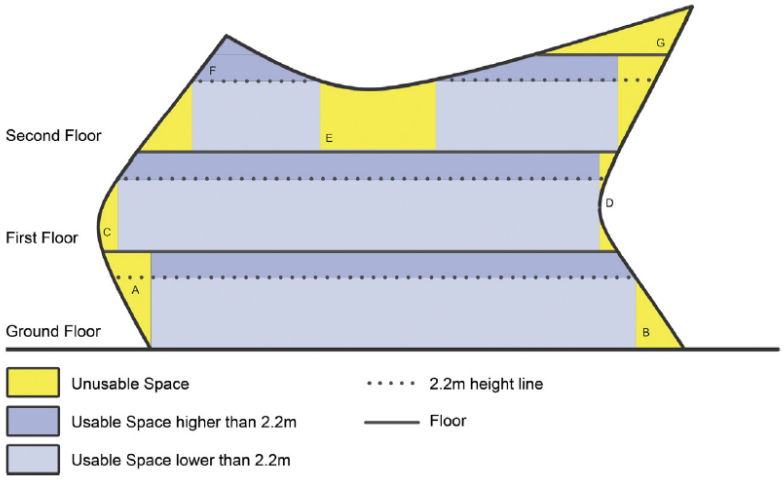


Figure 3: Optimum building envelope design - shape coefficient visualisation (Zhang et al.)

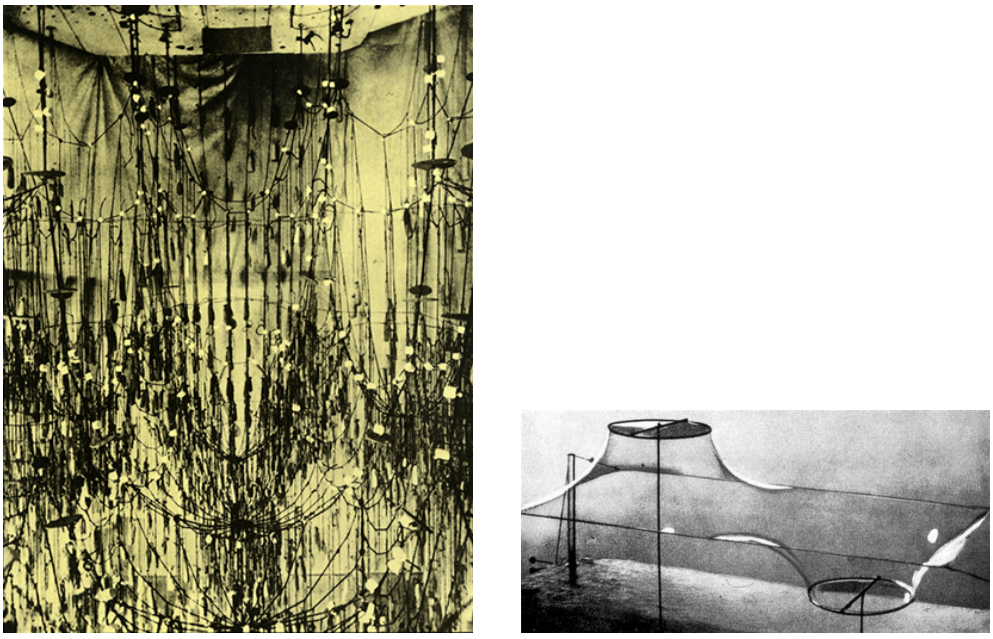
This case study - Evolution of Sun-Shades Outside Building Façades - is unique in that it considers the form the final, optimised sun-shade will take. Architecturally, this is known as 'form-finding' and it is a valid design strategy employed by the Architect (or designer) in a scenario where they cannot anticipate what shape the final product will take when solving a given problem. **Can the evolved form of a sun-shade perform better than existing 'traditional' sun-shades?** The Form-finding in this case study is more mature when compared to other research in that other sun-shade optimisation research typically modifies existing elements or alters a given layout by keeping parts constrained and moving components - lengthening, shortening or rotating as needed. Whereas this case study starts with a point cloud and evolves this volume to a final, unknown, form. Furthermore, what makes this research unique is that it determines a method with which to evaluate the fitness of traditional sun-shades and then uses this fitness value to compare traditional sun-shades with the newer geometrically evolved

[ii]Parametric refers to a software modelling method that allows the user to access parameters that make up the model. This is often 'dynamic' as changes to parameters are immediately reflected in the model.

[iii]The shape coefficient is the ratio between the area of a building's external surface and its internal volume.

sun-shade. Finally, the Evolutionary Algorithm, in this case study, is permitted to function fully; no 'designer' intervention is made, no parameter tweaks are permitted and the algorithm functions fully until the final generation is reached (no population 'restart' is required). This research uses a 'pure' Evolutionary Algorithm, not an implemented 'plug-in' or software generated solution. No limit is set on the evolutionary process as it functions separately from an environmental simulation which typically require simplified or pre-calculated data (often contained within a database).

Regarding 'Form-finding'; this approach to architectural design has a strong heritage going back to Antoni Gaudí's 'funicular'[iv] and being aptly demonstrated by Frei Otto's 'Soap Bubble'[v] experiments. Figure 4 illustrates these original methods of Form-finding.



(a) Antoni Gaudí Funicular for the Colònia Güell chapel (Martinell)

(b) Frei Otto Experimenting with Soap Bubbles (Zexin)

Figure 4: Architectural Form-finding: Antoni Gaudí and Frei Otto

While Evolutionary Algorithms are perfectly suited to the 'form-finding' design method - where they evolve a solution to suit a set of given fitness parameters with the resulting artefact being unknown - in architectural research they are typically used to optimise a series of given parameters. However, these results see a general 're-arranging of existing elements' rather than the creation of something new and

[iv]The funicular was a series of chains suspended from the ceiling. The form the chains took on visually illustrate the structural forces that would be experienced in his chapel design (shown upside down). The structure can be designed from this knowing it can accommodate the distribution of forces.

[v]Frei Otto coined the phrase 'Form-finding', which he used to describe his method of using soap bubbles to determine the shape the tensile roof would take given the structure and supports he designed for the 1972 Olympic Stadium in Munich - Olympiastadion.

possibly unexpected.

3 Literature Review

3.1 Understanding and creating Form

One of the earliest discussions on Form is D'Arcy Wentworth Thompson's On Growth and Form [59]. In this work Thompson seeks to remove the 'romantic'[vi] approach to the study of organic life and replace it with the more 'modern' scientific method underpinned by 'mathematical processes'. He starts by indicating that the forms living things take are constrained by physical laws while growth is to be studied in conjunction with form and can be seen as either an increase in size - 'magnitude' - or a gradual change over time (the slow development of a structure). When considering magnitude, it is proposed that this be a scalar value (something represented by a number) and is connected to the relationship between parts - most probably as a ratio. The 'Principal of Similitude' is important - as originally noted by Galileo [58] - where the effect of scale has limitations [vii]. Notably, as something increases proportionately in size it will eventually collapse under its own weight. To address this - either proportions need to be altered (often resulting in something 'clumsy' or inefficient) or the material used needs to change (to something stronger). When considering growth, this can be seen as a vector in that it is the change of magnitude in a direction, typically the magnitude of one direction to a second; length to height. As these magnitudes vary across time intervals you get growth. Where time is involved you see a 'rate of growth', this also plays a part in determining the form an organism will take. Thompson wraps up by concluding that, among things, *"Every growing organism, and every part of such a growing organism, has its own specific rate of growth, referred to a particular direction. It is the ratio between the rates of growth in various directions by which we must account for the external forms of all, save certain very minute, organisms."*

The Architect, Christopher Alexander, while reflecting on the state of the architectural

[vi] The approach to studying organic life at this time was to see 'something more' in natural artefacts that neither physics nor mathematics could explain. Among the idea's at that time were that pedigrees and blood-relationships formed part of the classification system, embryology could define the life history of a race or that bird migration patterns could reveal lost lands (islands or sunken continents). This was all underpinned by the idea of 'the final cause', the 'great design' or 'end purpose'.

[vii] Regarding the Principal of Similitude, Galileo is quoted as follows: *"From what has already been demonstrated, you will plainly see the impossibility of increasing the size of structures to vast dimensions either in art or in nature; likewise the impossibility of building ships, palaces or temples of enormous size in such a way that their oars, yards, beams, iron-bolts, and, in short, all their other parts will hold together; nor can nature produce trees of extraordinary size because the branches would break down under their own weight; so also it would be impossible to build up the bony structures of men, horses, or other animals so as to hold together and perform their normal functions if these animals were to be increased enormously in height; for this increase in height can be accomplished only by employing a material which is harder and stronger than usual, or by enlarging the size of the bones, thus changing their shape until the form and appearance of the animal suggests a monstrosity."* [32]

profession, sought to develop a 'systematic' approach to creating form, which he sees as the ultimate goal of design. Alexander does not see the world as homogeneous but that it is instead irregular. Consequently, as the world tries to fit this irregularity, form is generated. Alexander underscores this by quoting D'Arcy Thompson where form is seen as the 'diagram of forces' for these irregularities. In his early work [3], Alexander sets about proposing an approach to creating form by incorporating Logic and Mathematics into the architectural design process; he appreciates this is a difficult prospect for the architectural profession and explains:

"It is not hard to see why the introduction of mathematics into design is likely to make designers nervous. Mathematics, in the popular view, deals with magnitude. Designers recognize, correctly, that calculations of magnitude only have strictly limited usefulness in the invention of form, and are therefore naturally rather skeptical about the possibility of basing design on mathematical methods. What they do not realize, however, is that modern mathematics deals at least as much with questions of order and relation as with questions of magnitude. And though even this kind of mathematics may be a poor tool if used to prescribe the physical nature of forms, it can become a very powerful tool indeed if it is used to explore the conceptual order and pattern which a problem presents to its designer. Logic, like mathematics, is regarded by many designers with suspicion. Much of it is based on various superstitions about the kind of force logic has in telling us what to do."

Alexander starts by understanding that the designer's 'intuition' will not assist very much in the creation of form and that, furthermore, this intuitive process has essentially become guided by 'general principals' which, over time, have formed the 'root' of architectural theories. However, Alexander argues, these general principals are no longer fit for purpose - if anything, they lead the designer further astray as new concepts get developed to deal with more complex problems. Alexander sees these concepts as being the result of 'arbitrary historical accidents' and consequently do not help the designer to find a 'well-adapted' solution.

Alexander outlines this scenario:

"The modern designer relies more and more on his position as an 'artist', on catchwords, personal idiom, and intuition - for all these relieve him of some of the burden of decision, and make his cognitive problems manageable. Driven on his own resources, unable to cope with the complicated information he is supposed to organize, he hides his incompetence in a frenzy of artistic individuality. As his capacity to invent clearly conceived, well-fitting forms is exhausted further, the emphasis on intuition and individuality only grows wilder."

Alexander's 'systematic' approach to creating form is based in the mathematical world of set theory where design criteria ('misfits') get abstracted into elements within a mathematical set, this becomes more complex as designs increase in complexity and includes nested sets. The way these elements interact with each other - interfere, conflict, concur - determines the character of the system. Together with this original set of design criteria is a second set that indicates the links between elements, this set identifies the interaction between elements.

The two combined sets - elements and links - form a 'linear graph' the structure of which is called a 'field'. This can be broken down using a method called 'decomposition' to show subsets and hierarchies. This final diagram becomes 'the program' as it contains

the instructions to create the final form.

To understand how he extracts design criteria and creates connecting links, Alexander does go into some detail discussing why creating form outside of this method fails - he looks at how vernacular (similar to 'community build') architecture approaches form-making and contrasts this with Modern architecture. He identifies 'unselfconscious' and 'self-conscious' approaches to design - where the first is generally successful while the latter needs support. He creates 'rules' for using his system (what makes a fit criteria and how sets work), he discusses methods for decomposing the structure and ends with a final diagram - a pattern.

In his introduction to the revised edition, Alexander indicates he focused too much on discussing processes while leaving the diagrams that supported this discussion to the end. When revisiting this work the value of the diagrams struck him. These 'patterns' are an abstraction of components interacting within a system while at the same time being isolated from other systems.

Using these diagrams allowed for multiple, small, independent systems to be combined to form a whole by connecting their component ideas together. He had the realisation that, because these systems are 'abstract and individual', they could be combined in multiple ways for infinite designs. The power of his diagrams led him to realise the content he had written was simply a 'formal and complicated' way to get to his diagrams/patterns. He summarises; *"If you understand the need to create independent diagrams, which resolve, or solve, systems of interacting human forces, you will find that you can create, and develop, these diagrams piecemeal, one at a time, in the most natural way, out of your experience of buildings and design, simply by thinking about the forces that occur there and the conflict between these forces."*

This idea of patterns has had a massive impact across multiple disciplines, such that Alexander goes on to acknowledge that from the 'method' section in this work 'a whole academic field has grown up - Design Methods'. Alexander continues: *"... and I have been hailed as one of the leading exponents of these so-called design methods. I am very sorry that this has happened, and want to state, publicly, that I reject the whole idea of design methods as a subject of study, since I think it is absurd to separate the study of designing from the practice of design."*.

Alexander next produced a three-set series of books further refining his idea of patterns; The Oregon Experiment (1975), A Pattern Language: towns, buildings, construction (1977) and The Timeless Way of Building (1979). Alexander sees 'A Pattern Language' as the second 'volume' of a whole with 'The Timeless Way of Building' forming the first 'volume' indicating that the second volume may be seen as a 'sourcebook' for the 'timeless way' while the first volume covers its practice and origin [4].

Reception to this body of work within the architectural community has been mixed - this is clear in recent studies hoping to revisit the theories. Calling for a balanced re-evaluation of this work, Ritu Bhatt indicates how it was initially rejected as it was seen to be 'deterministic and authoritarian' by academics [10]. Similarly, Dawes and Ostwald indicate that while this is probably the most widely read architectural treatise, the content is not discussed - there is a lack of critical engagement with the theory to challenge the ideas and assumptions they contain [21]. Dawes and Ostwald go on to indicate that the three-set series can be seen as 'canonical texts' and they identify this

work as Alexander's 'second theory' of architecture.

3.2 Computers in Architectural design

In the early to mid 1990's, two important works focused on connecting architectural practice to computing. At that time computers were typically seeing use as drawing aids allowing for an increase in the production of drawings but computationally saw little use in the application of design (CAD)[viii].

In the first work Mitchell [45] considered the 'logic' underpinning computer algorithms and translated that to the 'logic' underpinning architectural theory. Where 'rules' could be applied to generate architectural components (or designs), so too could they be converted into algorithms for computer application. Mitchell considered the 'Greek Orders' [29], specifically the construction of a classical column. An example of the Doric order (after Fletcher) can be seen in Figure 5. Mitchell believed they can be classified according to certain rules - originally extracted from the use of grammar (article noun verb). He used this grammar to construct 'a sentence' which becomes 'classical rules'. The designer works down a hierarchy, from the abstract to the refined item (which, it is hoped, can go on to become an architectural detail drawing). Mitchell clarifies this idea: *"All this works because a classic order is, like a sentence, a linear sequence of elements (running bottom to top instead of left to right). Thus we can embed a grammar in an algebra in which the sole operation is concatenation of sequences of elements."* This is illustrated in Figure 6 and thereafter Figure 7. From this starting point, to better address *"two and three-dimensional form"*, Mitchell expands what can be done using algebra (with the introduction of parameters) and works through grammar tables to increase the scope of work to include 'sophisticated (architectural) grammar' to generate floor plans of entire (Palladio style) buildings. Here he incorporates Shape Grammar and later Artificial Intelligence into architectural design and references the seminal work of Stiny [56] and McCarthy [44].

[viii] CAD has two meanings; originally meaning Computer Aided Design, later to also mean Computer Aided Drawing (drafting). Computer Aided Drawing is the most common meaning today - this is where drafting work has become digital, this term is not flattering. Computer Aided Design is an older term and was used when computers were seen to be of assistance in the design process.

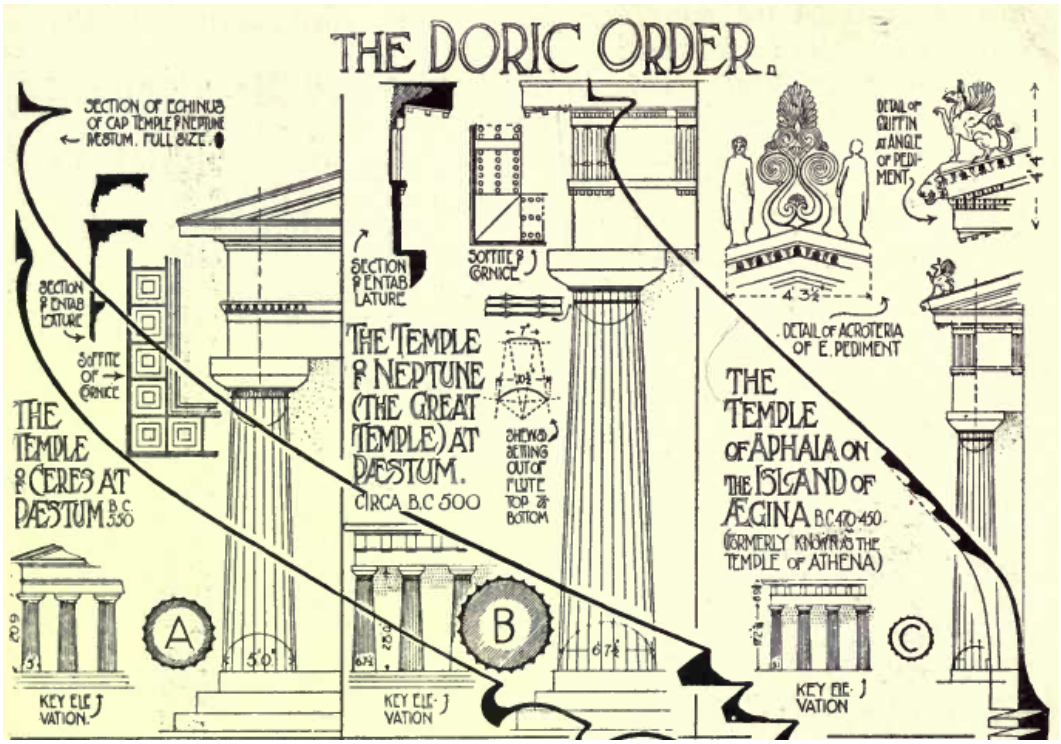


Figure 5: Greek Orders - Doric Order (Fletcher)

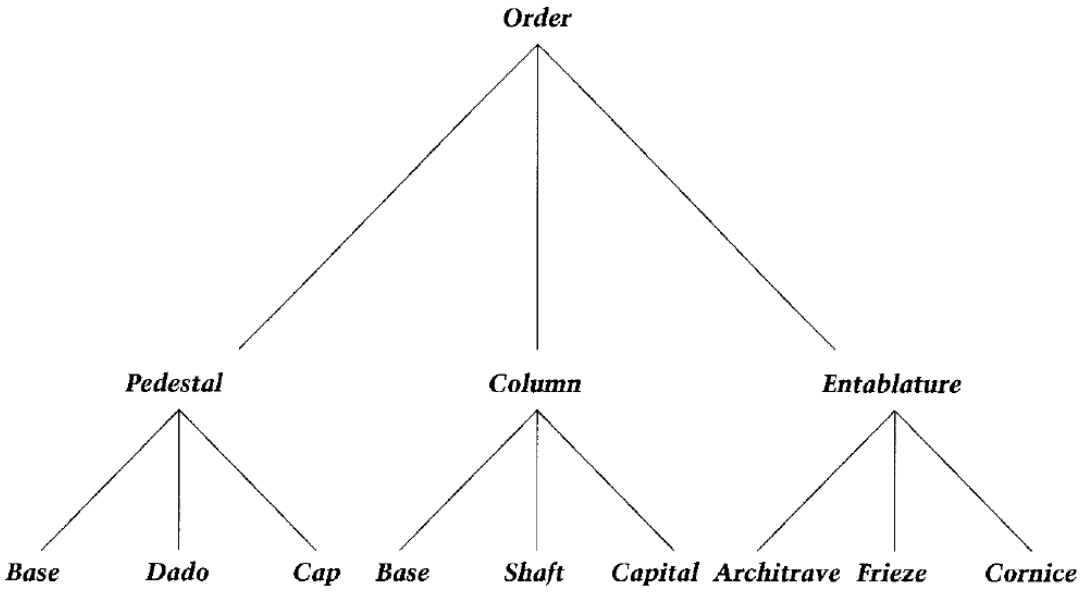


Figure 6: Recursive grammar for an architectural column (Mitchell)

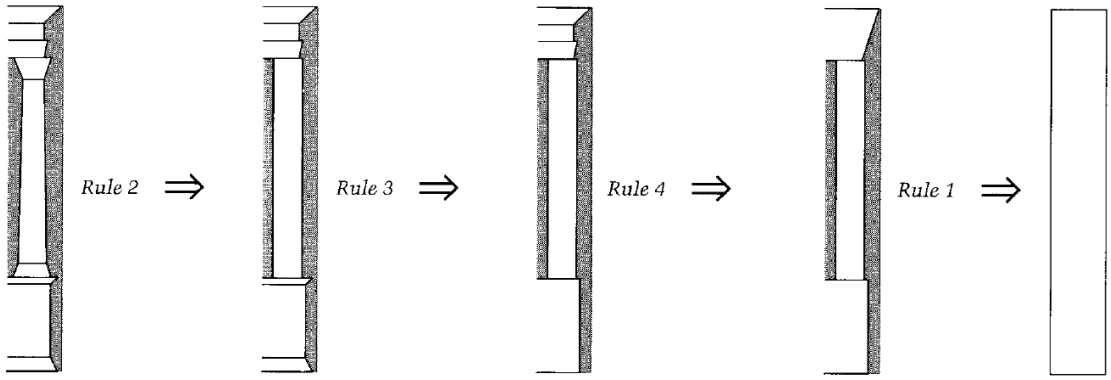


Figure 7: Recursive grammar applied as a linear sequence of elements (Mitchell)

While Mitchell focused on logical application, in the second work Frazer [31] focused on evolution. As part of a research module at the AA School (Architectural Association), an incredible amount of research is showcased. The starting point was the design and development of the tools they believed they needed to carry out their research (at the time this was undertaken, computing power and software was limited and computer graphics techniques were in their infancy). The idea was to 'start from first principals' with the benefit that they could create solutions to support their research (rather than use commercial software that may not quite assist as expected). Frazer started by building a computer and logic circuits (after referencing the Turing Machine, The Von Neumann Machine, Parallel Processors and finally Neural Networks). Computer modelling was introduced - a virtual representation of an actual construction. This allowed research ideas to be developed, described, visualised and evaluated for environmental performance. When creating their environment for simulation, their research found the understanding, and consequently the application, of solar geometry by architects was erroneous. (This was due to the lack of an understandable mental model). A family of solar tools (with physical models) was created and incorporated as computer models. Researchers understood at this point that user input required an upgrade; digitisers were considered along with using physical models as a means of providing input. From this flowed the creation of a self-replicating computer and out of that a machine-readable grid board. Figure 8 shows a three-dimensional version created out of cubes (later to be reduced in size).

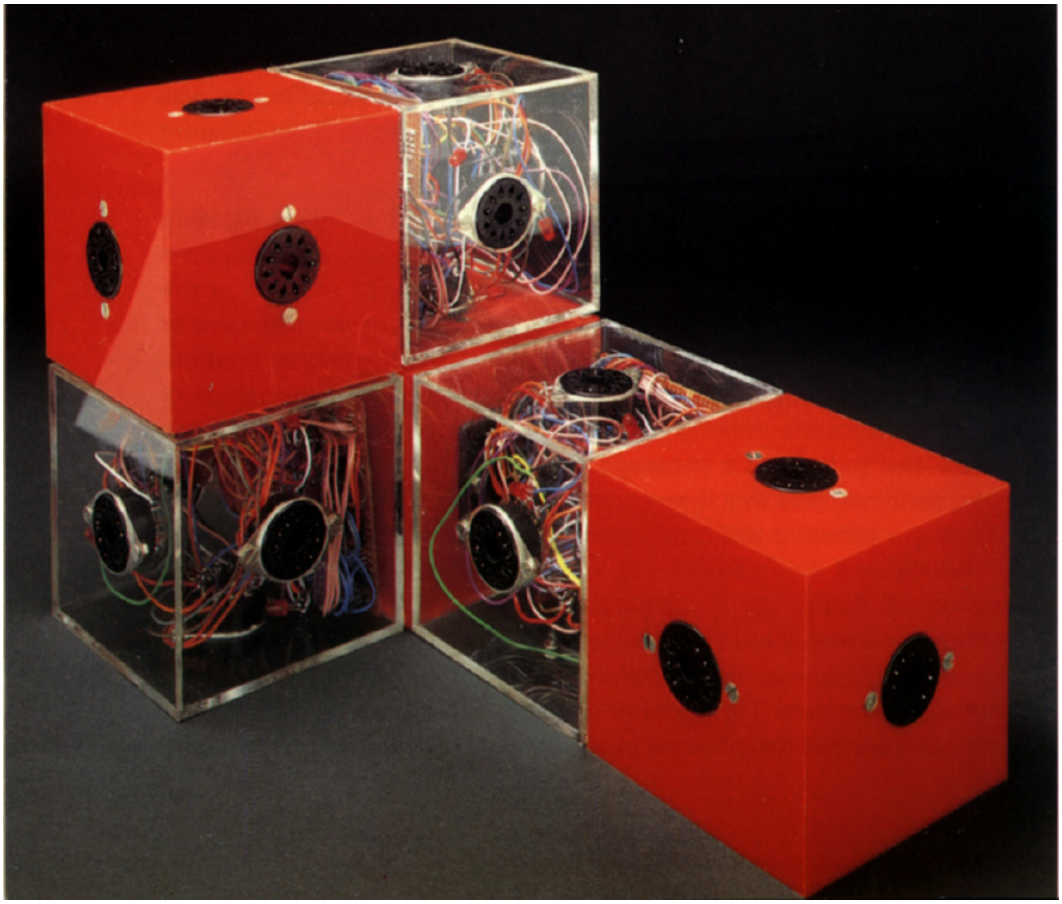


Figure 8: Three-Dimensional machine readable grid, 1980 (Frazer)

The 'Generator Project' was set up as a consequence of the three-dimensional machine readable grid - here building components ('a kit of parts' - enclosures, walkways, screens, services) could be re-arranged to meet changing client needs. A computer program was used to drive the layout of the components to create new arrangements thereby satisfying (changing) client needs. Importantly, each part had embedded electrical components making the building a 'reconfigurable array processor' - where the processor and building would have identical configurations. Frazer saw this as an Intelligent Building as it could reconfigure itself to suit user needs (and even 'got bored' if there was an inadequate need for change). 'Embedded intelligence' and 'learning from ideas' came out of the Generator Project - consciousness. As Frazer indicates; *"I am not interested in the argument about whether computers are actually intelligent, alive or conscious, but as a mental exercise it is interesting to consider a building to be conscious at least in the sense of being able to anticipate the implications of its actions, as any good environmental control system should."* 'The Universal Constructor' was the next 'intelligent model' - using cubes as their constructor (cubes being more universal than building components.). The cubes could have a 'state' and this could be mapped to an environmental aspect (sound, wind, and so on), furthermore each cube could communicate with the other. The flow of logic could pass through the cubes with the entire structure functioning as an input/output device - input could be the location or configuration of the cube while output would be a displayed message. Using this displayed message, the construction would speak

to an observer (who could add/remove cubes as requested by the constructor). It was expected that this tool could be used as a new design method (it could function as a three-dimensional cellular automata responding to a building site condition). *"The 'Universal Constructor' was an interactive self-organising environment"*, a working model is shown in Figure 9.

Figure 9: Universal Constructor - a working model functioning as 'interactive self-organising environment' (Frazer)

Frazer's research group next looked at Polyautomata for use as a generative tool (this was due to the simplicity they offered). Cellular Automata allowed researchers to understand complex behaviour derived from simple rule sets. Wolfram's cellular

automata [63] and Conway's 'Life Game' [19] were studied and incorporated - cellular automata as 'finite state machines' were seen as a class of computer with the ability to simulate computational operations. Self-replicating automata (after Von Neumann's thought experiment [46]) were tested with the hope of increasing the complexity of forms being developed. Strongly influenced by Conway's 'Life Game', 'Artificial Life' was the next field the research group looked at, noting: *"Our model of architecture exhibits the characteristics of metabolism, epigenesis, self-reproduction and mutability, which are generally agreed to be requirements of life"*. Frazer makes it clear, this research sees architecture as a form of artificial life while others were searching for behaviour to consequently claim artificial life.

Next, Frazer looks at Evolutionary Methods; identifying a simple use - where performances are measured and a slightly improved solution is selected, while all others are rejected. He indicates this is useful when one plans to optimise *"already satisfactory solutions, where only a small variation of quality is required"*. A more sophisticated evolutionary method is required where, either a well developed solution does not exist or where the possible improvement is radically different. Holland's [36] adaptive model is investigated where a series of artefacts are produced from a set of operators while the environment influences the selections made. From this Holland developed generalised reproductive and genetic operators resulting in Genetic Algorithms - giving researchers the ability to determine solutions not 'imagined' or to solve problems that were not initially fully understood.

Biomorphs - proposed by Dawkins [22] - were evaluated alongside genetic algorithms; but Frazer determined that the latter would be more suitable for design applications. As evolutionary processes could result in diverse products, so too could architectural problems be subjected to these same mechanics and have varied solutions generated. In the works published, an interesting design for a sun-shade was produced, see Figure 10.

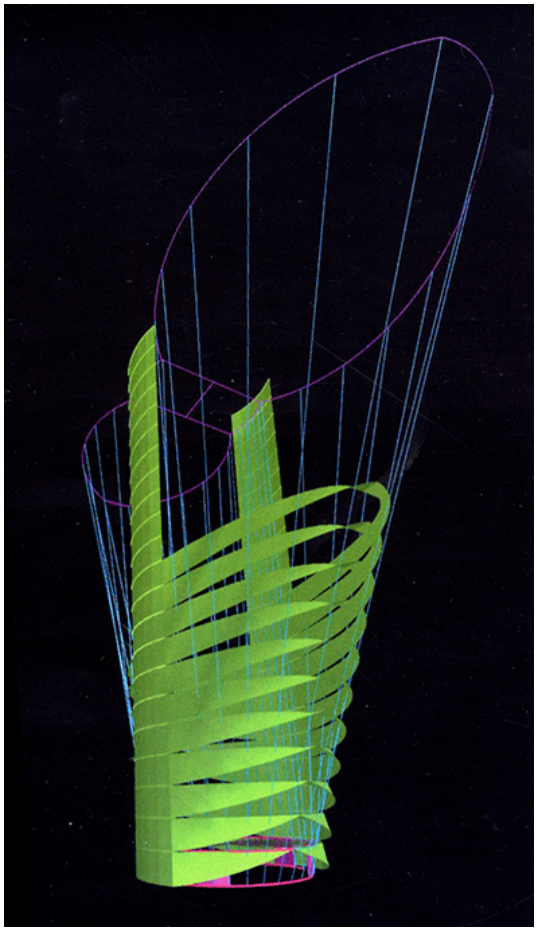


Figure 10: Solar Study evolving a wrap-around sun-shade (Frazer)

Influenced by Biomorphs, researchers modified Genetic Algorithms to form 'hierarchical genetic algorithms' where *"one algorithm learned the criteria for selecting the outcome of the next"*, this was done as positive results were achieved when a researcher intervened during selection where the initial selection process was either ill-defined or had conflicting selection criteria.

Finally, Frazer 'locates' this research within the practice of Architecture and then discusses the tools they developed and how to use them to generate 'a new model of the generative architectural process'. Speaking on how a CAD (Computer Aided Design) methodology could be used, Frazer believes that the approach to architectural design in general is unsatisfactory and should not be imitated. He also feels that to mimic a human process (architectural design) via a machine is not using the machine to its full potential. Finally, Frazer indicates his concern for designers who use programs they feel are 'useful' as this software only addresses the designer's 'sensibilities' and does not explore problem solving and producing results.

While discussing the 'Specific Model' his new 'generative architectural process' will use, Frazer notes the establishment of a computer environment for the model; an architectural genetic search space. The datastucture within this space is made up of points - also called motes - that have metadata attached such that each point understands its location and position relative to others. It is also made up of unique

properties (descriptions of forms it may map to) and a history of how it came to be where it is and what it is. The mote does not move but instead the metadata moves across 'a field of motes'. This field extends indefinitely and motes that surround a structure have environmental properties (that could affect the behaviour and performance of the structure).

Frazer notes that the advantage of these motes is in representation and visualization; but this is also the disadvantage as they may be misunderstood to be simply geometrical.

3.3 Architectural Form-Finding (beginnings and theory)

Central to this case study is the architectural notion of form-finding. When considering the idea of 'form-finding' as an architectural design method, Antoni Gaudí's system of constructing a funicular [43] may be considered a good starting point. In the late 1800's, Gaudí [65] constructed his funicular - models made up of a series of connected chains with weights suspended from the ceiling. Under the influence of gravity, the various inter-connected chains - when settled (static) - would represent the final form the forces in the building will take. This, in turn, dictates where structural elements need to be, for example; columns to carry the load of domes or arches. Figure 11 shows a reconstructed scale model of Gaudí's hanging model for the Colònia Güell chapel.

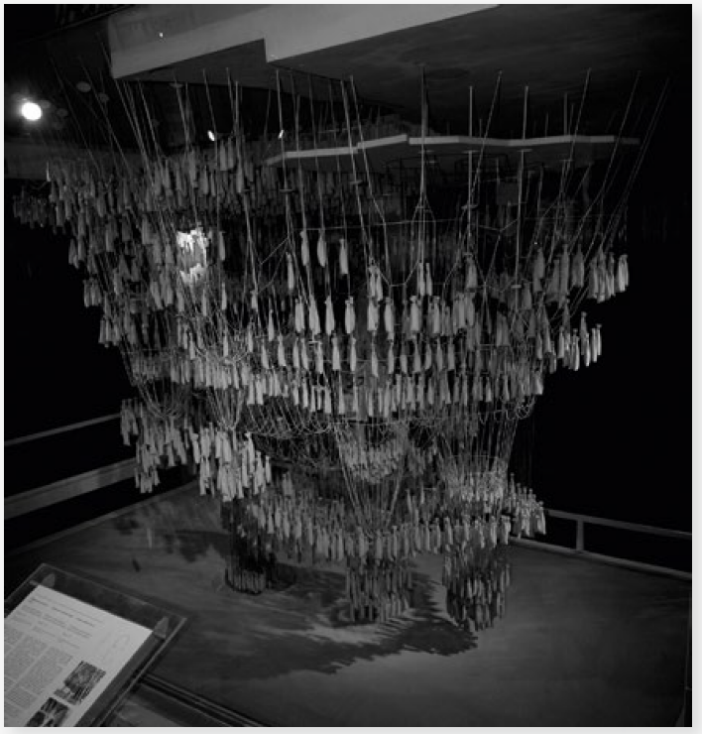


Figure 11: 1:15 scale reconstruction of Gaudí's hanging model (Burry)

Frei Otto, in the 1970's, used a conceptually similar approach to find the final form his tensile roof structure would take for the Munich Olympic Pavilion roofs; Figure 12 shows the final construction in situ. Otto notes that in the 1950's, the mathematics of hanging structures was quite complicated, forcing him to consider using a physical

Figure 12: Frei Otto and Günther Behnisch, Olympic Stadium, Munich, 1972 (Burry)

model. He created tensile supports as a model and used soap films to form the skin, in essence reverse tension-loading the suspended forms. This allowed him to determine the way the final roof structure would look given the support system he supplied. It was Frei Otto who coined the phrase 'form-finding' and explained how it works as a design method [34]. For Otto, form-finding is a design method using 'practical processes' that make use of the 'self-organisation of materials' under 'external forces'. (By practical processes, Otto means built artefacts, self-organisation of materials refers to the natural shape materials tend to take for equilibrium to become static - similar to the angle at which a pile of sand settles.) The forms available as solutions are controlled by the choice and definition of the conditions under which form-finding takes place. Otto 'unites the logic of material' and structure.

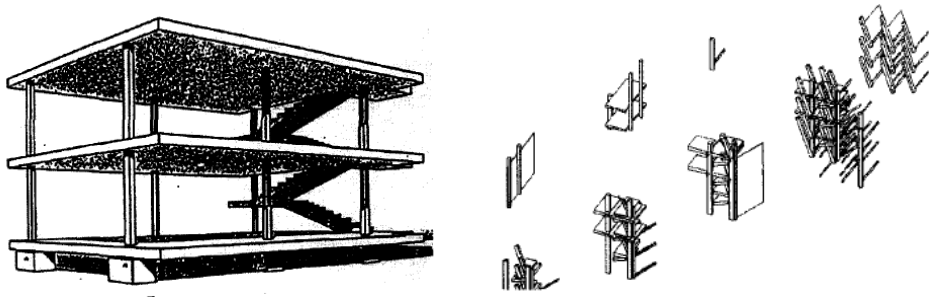
The work pioneered by Gaudí and Otto is now considered part of the more recent 'architectural style' of 'Parametricism'[ix]. While Schumacher is credited with 'discovering' Parametricism (Parametric Architecture), Burry links Parametricism to the earlier work of Gaudí and Otto [14]. Burry, who continued the work on Gaudí's unfinished 'Basílica de la Sagrada Família', sees Gaudí as a parametric thinker. By 'parametric thinker', Burry means a *"creative mind capable of thinking and acting parametrically"*. He draws similarities to the 'Plastic Architecture' of the 1920's which he believes is effectively 'parametrically variable ('plastic') architecture'. Under plastic architecture, the focus became that of 'elements' with the following given as examples; 'function, mass, plane, time, space, light, colour, material...'. 'Elements' here equate to 'variables'

[ix]In 2009, Schumacher wrote that the innovations produced by Parametricism would replace those that had been generated by the Modernist movement. While taking care to clarify that Parametricism is an architectural style the techniques used are interesting; they include 'animation, simulation, form-finding, parametric modelling and scripting' . [55]

in the parametric paradigm.

According to Burry, Van Doesburg's 1924 Manifesto (Towards a Plastic Architecture) [61] was talking about 'parametrically variable architecture' all along. This manifesto shifts architecture away from (rigid architectural) styles and the imitations thereof and instead proposes an approach to architecture in a fresh way; namely that architecture is composed of building elements. Interestingly, in the Manifesto, Van Doesburg notes: *"The new architecture is formless and yet exactly defined; that is to say, it is not subject to any fixed aesthetic formal type. ...The functional space is strictly divided into rectangular surfaces having no individuality of their own. Although each one is fixed on the basis of the others, they may be visualised as extending infinitely. Thus they form a co-ordinated system in which all points correspond to the same number of points in the universe. It follows from this that the surfaces have a direct connexion to infinite space."* Van Doesburg is, in essence, discussing Euclidean space providing a link between 'Plastic Architecture' and computer modelling.

Coates and Makris build upon the work of Mitchell using the shape grammar of form and genetic programming [17]. They experiment with Le Corbusier's 'Domino House'[x] as that conceptual house structure easily lends itself to shape grammar. The 'house' gets defined as an s-expression, the phenotype of which is then evolved to show the various possibilities inherent in the morphology as supplied originally by Le Corbusier. Representation is in the form of geometrical 3D objects - 'blocks', that get evolved to form a Domino House prototype. Figure 13 illustrates Le Corbusier's idea of the Domino House and part of the evolved structural work. Pottmann provides a thorough textbook on architectural geometry and would be a starting point when determining how to represent geometry [51].

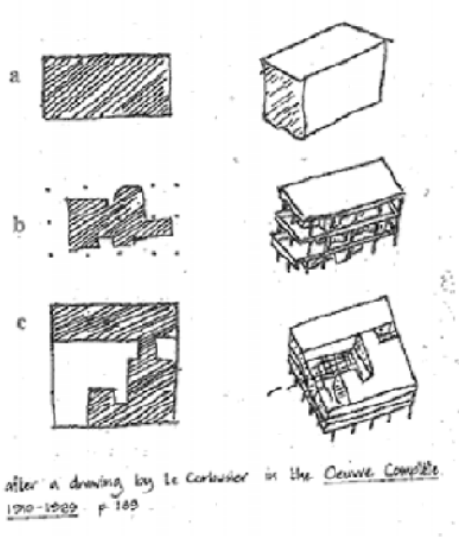


(a) Domino House (Le Corbusier) (b) Stick and Slab primitive objects

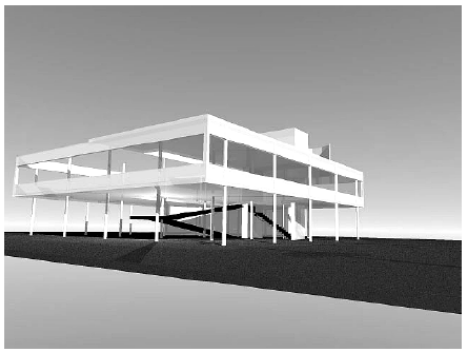
Figure 13: Shape Grammar of form with Genetic programming (Coates & Makris)

Again, using Le Corbusier because of his 'distinct compositional techniques', Asojo converts elements that occur in the 'White Villas' to 'program rules' [5]. He scripts the rules in CAD software to generate architectural 3D elements sequentially. He then also converts the elements to shape grammar and follows Coates and Makris to use genetic programming to generate 3D models. As Asojo explains: *"In genetic programming the*

[x]The domino house was an idea Le Corbusier developed that was a concrete frame structure with vertical columns and a reinforced slab (requiring no beams for support).



(a) Sketch Designs (Le Corbusier)



(b) Generated Prototype after Le Corbusier sketch designs

Figure 14: Genetic Programming incorporating Le Corbusier's techniques (Asojo)

basic idea is that architecture results from the multiplication of simple relationships. The range of moves available when exploring by hand are limited. The use of a recursively defined generative grammar using genetic programming allows for recombination and embedding of morphological moves to any level of complexity required." The results are illustrated in Figure 14.

While Mitchell and Frazer have made significant contributions to introduce and establish roles for computing in architectural design thereby allowing Coates and Makris to explore this further (with all producing interesting 3D models and forms), architectural theory still frowns upon this body of work. Broadly speaking, Evolutionary Algorithms are not easily accommodated in both the practice and theory of architecture. De Landa refers to Evolutionary Algorithms as 'Genetic Algorithms' and has attempted to embed this approach to design within architectural theory using the work of Gilles Deleuze as a starting point [25]. These 'genetic algorithms' are discussed in the context of how they assist with artistic design. The concern De Landa addresses is the architect's concern that an algorithm can design and consequently (following from this), what value (or burden) this could place on the architect as a (chief, artistic) designer. De Landa goes on to determine where in the process of evolving an artistic design solution the role of the designer is crucial; namely, the designer needs to consider the search space ensuring it is rich enough to allow for surprising (or even shocking) solutions - typically solutions the designer has not already considered. Da Landa believes that by considering the philosophical work of Gilles Deleuze, the architect will be able to establish a rich search space. The architect is required to consider 'population thinking', 'intensive thinking' and 'topological thinking' that Da Landa believes will together form a 'new conception of the genesis of form'.

At a similar time Kolarevic [39] was exploring the impact the 'Information Age' would have on how architecture is designed, seeing it as a challenge that needed to be addressed. This challenge is to move architecture away from its traditional (building construction)

technological underpinning to computational innovation where 'digital media' is used as a generative tool (rather than as a representational tool) to generate form - digital morphogenesis[xi]. Following Kolarevic, **Computational Architecture** is conceptually divided into various categories, namely; Topological Architecture (topological space), Isomorphic Architecture (isomorphic surfaces), Animate Architecture (motion kinematics and dynamics), Metamorphic Architecture (keyshape animation), Parametric Architecture (parametric design) and Evolutionary Architecture (genetic algorithms) - each of the 'new, emergent dimensions of architecture' is then discussed. **Topological Architecture** is defined by its departure from Euclidean geometry, represented in the Cartesian space, towards the geometry of continuous curves most typically found in Non-Uniform Rational B-Splines (NURBS) where the topology is represented by parametric functions (rather than equations). **Isomorphic Architecture** is also removed from Euclidean geometry (and Cartesian space) and uses isomorphic surfaces ('metaballs' or 'blobs') to create geometry (isomorphic surfaces have fields of influence that can attract/repel - a surface is generated where the field intensity is equal). **Animate Architecture** uses animation software to produce both force and motion simultaneously[xii] while form is produced, this results in producing a change in form together with the forces that shape it. **Metamorphic Architecture** uses 'keyshape animation' where the state of the geometry is recorded per animation keyframe and software computes the 'inbetween' frame shape changes. The modelling forms that are keyframed vary; typically bounding box deformations, spline curves or deformations along a path. **Parametric Architecture** concerns itself with changing exposed parameters resulting in a final shape. Equations can be used to link parameters to provide relationships between parts or objects. Parametric design requires procedural ('algorithmic') descriptions of geometry. **Evolutionary Architecture** follows the work of Frazer and he is quoted: *"architectural concepts are expressed as generative rules so that their evolution and development can be accelerated and tested by the use of computer models. Concepts are described in a genetic language which produces a code script of instructions for form generation. Computer models are used to simulate the development of prototypical forms which are then evaluated on the basis of their performance in a simulated environment. Very large numbers of evolutionary steps can be generated in a short space of time and the emergent forms are often unexpected."* Kolarevic concludes that computational architecture provides strong design exploration coupled to a high level of unpredictability which work well with form-finding, something this approach seems to be made for. Figure 15 shows an example of the work produced by 'Animate Architecture'.

[xi] Digital morphogenesis is also referred to as **computational architecture** where form generation and transformations are derived through computational processes

[xii] Typically animation software produces forces separately and they are fed into the final animation.

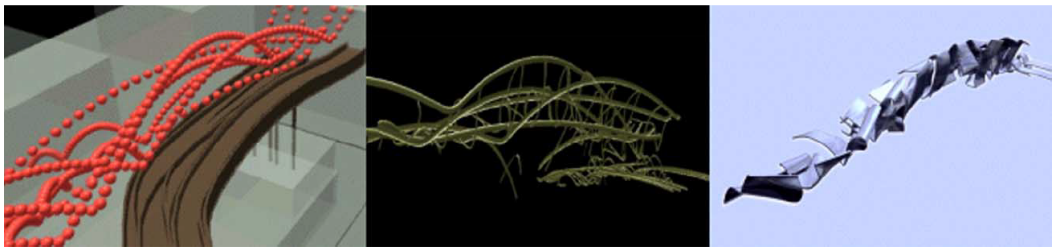


Figure 15: Animate Architecture: Greg Lynn's work - Port Authority Bus Terminal, New York (Kolarevic)

Written in 2010 as an historical overview and to strengthen the link between experiments and architectural theory, Brian Holland [35] traces "Evolutionary Thinking" in the field of Architecture. The original architectural approach, when looking at nature, was to mimic biological systems; as he notes, the *"strategic application of adaptive biological design principals to man-made systems"*. This has formed the basis for the architectural theory of **Biomimetics**. Holland documents that at around the same time as Darwin was publishing 'On the Origin of Species' [20] the 'Revue Générale de l'Architecture' created the term *'Organic Architecture'*, now known in architectural theory as **Organicism**. The direct connection to Darwin's work is unclear; organic architecture related architecture to 'the organised life of animals and vegetables'.

The initial architectural approach to Evolution was to be analytical. It becomes a metaphor within the history of architecture with which to classify designs and buildings and chart their change over time. Interest is with growth, anatomy and classification systems.

'Contemporary' interest in evolutionary systems starts with Frazer's 'An Evolutionary Architecture' - discussed previously. Here the focus is on form generation. Frazer is credited as being the first to try and develop a conceptual and computational system that could be used to design architectural forms.

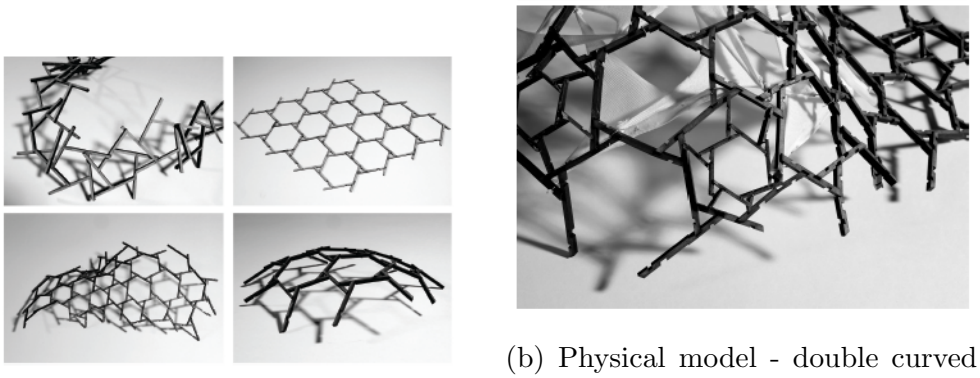
Holland acknowledges the 'subjective selection' designers have incorporated in their evolutionary selection methods and references De Landa's discussion on the role of the Architect in an evolutionary design model. He credits De Landa with locating Genetic Algorithms in architectural theory and for establishing a theoretical framework to discuss these algorithms.

Holland's survey ends with the work done at MIT in 2001 on the Maya software plug-in - Genr8. This plug-in incorporates evolutionary computation, generative computation and an environmental model. Surface geometries are grown using three dimensional L-systems under the effect of various forces (wind, gravity, and so on). The evolutionary module allows for populations of surfaces to be examined over generations and specifically allows the selection process to be automated or subjectively managed by the user.

Agkathidis [1] looks at how *Generative Design Methods* can be implemented and taught at an Undergraduate educational level as **Biomorphic**[xiii] **Design**, testing if this is suitable for the Undergraduate Design Studio. It should be noted that the

[xiii] Agkathidis describes the term Biomorphic as coming from the world of art and used when describing the work of the sculptor Henry Moore. The term is connected to fluid, organic shapes in art, architecture

studio does encourage digital exploration and fabrication. Questions asked were; can this design method be integrated into a studio course, what are the strengths and weaknesses, does this methodology produce innovative solutions (and does this make students more employable and does it help in developing their design skills) and is this field appropriate for an Undergraduate level of study? The course follows three phases; **Analysis** (group work with precedent studies focused on nature where the intention of the design and appropriate parameters are defined), **Morphogenesis** (group work where abstract generative models are produced referencing the parameters defined previously) and **Metamorphosis** (where the abstract, generative prototypes are translated into 'traditional' architectural content - floor plans, sections, details, models and visualisations). Across various projects, Morphogenesis sees Grasshopper[xiv] used to generate prototypes, the product of which is 3D printed, CNC milled or laser-cut, an example of student work is shown in Figure 16.



(a) Generated Shell types

(b) Physical model - double curved shell with suspended membrane

Figure 16: Student work, Design Experimentation 2: Bird's Nest (Agkathidis)

Agkathidis concludes that using generative methods in an Undergraduate Studio has a positive impact as high levels of design innovation was noted while keeping the level of applied knowledge appropriate to Undergraduate students. It was felt that this course was successful in moving students out of the 'classical (architectural) curricula' to explore more innovative design methods, this connects to the needs of the job

and design. Specifically in Architecture the term is mainly used to refer to nature inspired or naturally occurring patterns and shapes.

[xiv] Grasshopper - algorithmic modeling for Rhino `http://www.grasshopper3d.com/`. Grasshopper started as a plug-in to Rhinoceros 3D (Rhino), but has since been built-in to the software. It is a visual programming environment that allows various modules to be installed with each module performing certain specialised tasks. As it is a visual programming environment, users can place components on a canvas and plug-in various components to programme a solution, the result of which is shown in the Rhino three-dimensional viewport. Among the top downloads are 'Lunchbox' for exploring machine learning, mathematical shapes and panelling, 'Kangaroo Physics', a live physics engine and 'Ladybug Tools' for environmental design (sun paths, shadow studies, wind and computational fluid dynamics).

market. Agkathidis concludes: *"the strict boundaries between terms such as biomimicry, biomorphism, zoomorphism, geomorphism, anthropomorphism, organicism and bionics are becoming ever blurrier. Emerging computational tools and design techniques, such as generative, algorithmic and parametric design, in combination with digital technologies like CNC fabrication and 3D printing, are embracing nature as a source of inspiration, and will allow constructive new synergies between biology and architecture in the years to come."*

A final observation on an architectural approach to form is by Bonnemaison [13] while discussing the work of a Graduate Seminar serving to outline more recent research on (Architectural) '**Organicism**'[xv]. Central to this work is the acknowledgement that the whole is larger than the parts (consequently nature needs to be deeply understood). This central theme is understood to be common in the scientific approach to organicism where it is not enough to discuss the parts, one needs to understand the context of the whole and how the parts function within it. To 'locate' Organicism within architectural theory, D'Arcy Thompson's work [59] on natural sciences is seen as the starting point as this work is commonly referenced by architects when considering the growth of structure and geometrical patterns. Frank Lloyd Wright is seen to be prominent in this movement where, at the time, organicism was termed 'organic' architecture. Methods for using wood and coloured glass, in combination, generated forms creating geometries that, in turn, could be repeated at varied scales to create a whole. This is considered typical of organic architecture (organicism) inspired by the natural sciences. Bonnemaison next discusses additional examples where architects were inspired by nature and so, in effect, practice organicism; Le Corbusier's modular series (referencing the Fibonacci sequence), Christopher Alexander (to create A Pattern Language [4]), Frei Otto's 'aesthetics' (from his soap film), Pier Luigi Nervi and Felix Candela (double curved concrete surfaces), Santiago Calatrava (metal arches, animal bones) and Zaha Hadid (metal and glass, complex shell forms). Bonnemaison believes the 'philosophy of organicism' offers ideas, methods and practice that can be used within the field of architecture and beyond ('digital technologies' and robotics) to transcend simply mimicking nature.

3.4 Architectural Solar controls - shading devices (Sun-Shades)

Managing the internal temperature of buildings has always been part of the architectural design process; in summer the intention is to minimise internal heat build-up while in winter this is desirable. The condition of the building envelope, the size of openings, the building's orientation on site together with consideration of local weather conditions play a role in finalising the design of the building. This approach to design has recently gained prominence in order to produce environmentally sound buildings - Green Architecture - that typically have small energy footprints and incorporate more passive (as opposed to mechanical) climate controls.

Solar controls play a central role in managing the impact of climate on buildings, consequently passive climate controls and shading devices become important. Originally,

[xv] Bonnemaison describes Organicism as *"the art of learning from nature to create beautiful architecture"* and goes on to say that organicism can be seen as an approach to 'invention' and 'interpretation' that draws from nature.

solar charts were used to determine the solar angle (compass direction) and azimuth (height above the horizon) of the sun at given times throughout the year; an example of a solar chart and how it would be used is shown in Figure 17 (this information is from the SKAT Climate Responsive Buildings [33] series). Using the chart for your geographic location, you can look up the position of the sun on a certain day and at a particular time (or across a time frame). You will receive the solar angle and azimuth for that scenario and can design your building to manage sunlight accordingly. Initially these charts were supplied in published tabular forms [53], but are now available electronically with additional data (like weather patterns)[xvi]. Furthermore, this electronic dataset can now be 'piped' into simulation software to test various design parameters or it can be used to extract precise solar details[xvii].

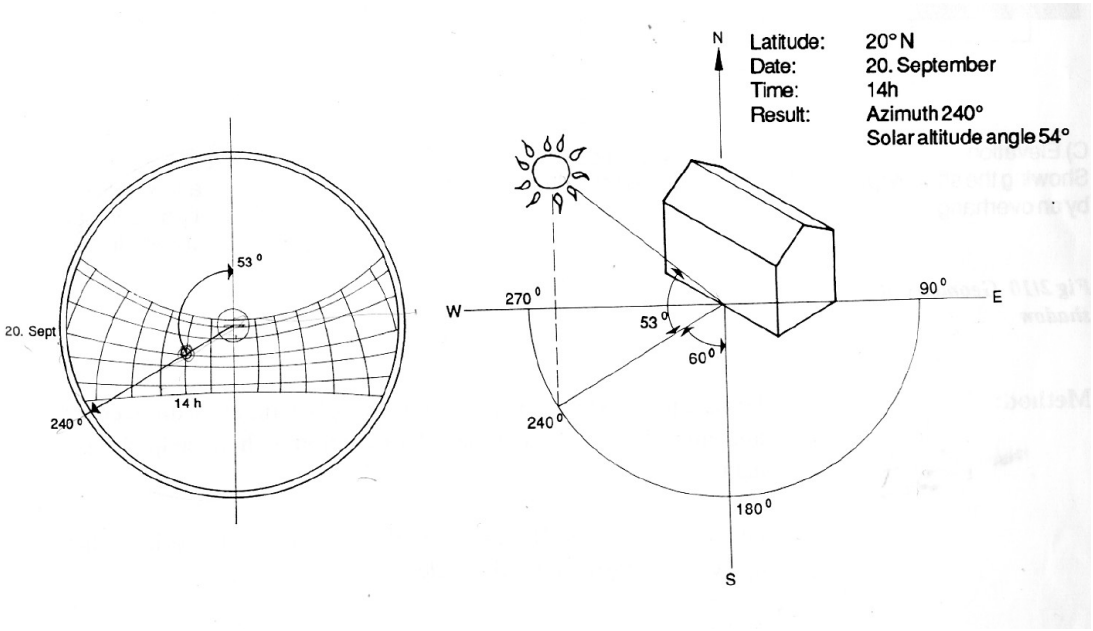
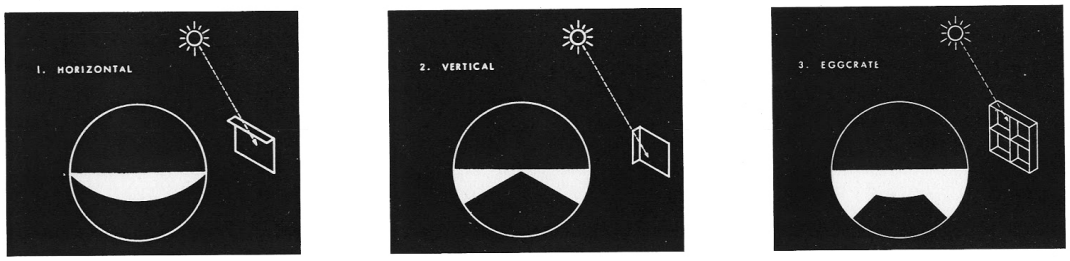


Figure 17: Typical Solar chart and its usage (example from SKAT Climate Responsive Building)

Once the solar angle and azimuth is known, a shading device - or sun-shade - can be determined for the given circumstances. In an important work, central to designing shading devices, Olgyay and Olgyay [48] proposed a method that has the designer colour in the solar chart and this, based on the resulting 'pattern', could be used to determine which sun-shade type best suits their design needs. Typical patterns are illustrated in Figure 18; sun-shade types (solutions) include the horizontal sun-shade (similar to a roof overhang, but above the window opening), the vertical sun-shade (like a vertical fin to the sun side of the window opening) and the 'eggcrate' sun-shade (a combination of the previous two to form a box shape in front of the window).

[xvi] EnergyPlus. Weather data. `https://energyplus.net/weather`

[xvii] Ladybug: A plugin for environmental analysis. `https://rhino.github.io/addons/ladybug.html`



Scheme of horizontal, vertical, and eggcrate devices.

Figure 18: Sun-shade solutions (Olgyay & Olgyay)

Olgyay and Olgyay define the effectiveness of a shading device as the proportion of the vertical surface covered by the shading device during the 'overheated' times versus the proportion of the vertical surface free from cover during the 'underheated' times. 'Overheated' times are essentially the summer periods when it is best to ensure no solar radiation penetrates the building interior while 'Underheated' times are the winter periods in which it is best that the building be heated passively through solar radiation.

It may be easier to visualise this by considering a north facing façade with a roof overhang; of interest is the surface shaded by the overhang in summer versus that free from shade in winter (when considering a building in the Southern Hemisphere).

After Olgyay and Olgyay, the effectiveness of a shading device (for summer shading) can be described as:

$$Sp = \frac{So}{Ro}.100\% \tag{1}$$

Where: Sp is the summer shading performance So is the intercepted direct solar radiation (Btu) during the overheated time Ro is the full amount of direct solar radiation (Btu) to strike the surface during the overheated time

The yearly effectiveness of a shading device is described as:

$$He = \frac{(So - Su)}{Ro}.100\% \tag{2}$$

Where: He is heat efficiency. This is obtained by deducting direct solar radiation losses, for the 'underheated' time (Su), from that of the 'overheated' time (So).

To evaluate a shading device for any locality and when facing any direction, we are interested in the 'shading effect' (Se). This is a ratio between 'heat efficiency' and 'shading performance' where heat efficiency considers the yearly balance for the shading device while shading performance is an optimum value for summer. This average value gives the formula:

$$Se = \frac{Sp + He}{2} \tag{3}$$

That gives us the final formula:

$$Se = \frac{So - \frac{Su}{2}}{Ro}.100\% \tag{4}$$

In the late 1990's, a similar follow-up guide after Olgyay and Olgyay's 'Solar control and Shading devices' was published, titled 'Tips for Daylighting with windows: The Integrated Approach' [47]; it was revised and published again in 2013 by Alastair Robinson and Stephen Selkowitz. This guide, acting as a quick-reference, supports designers with numerous topics of which 'Shading Strategy' is most relevant to this research. Certain 'Key Ideas' are presented, namely; designers are encouraged to use an external shading device to keep out unwanted solar heat, **horizontal forms for south facing windows** (north facing in the southern hemisphere), attempts to be made to design the building footprint to shade itself (if external shading devices are not acceptable), using **vertical forms on east and west windows**, **west and south windows to receive priority with regards to shading** (west and north in the southern hemisphere) and finally 'glare relief' should also be considered when working on shading. Their proposed shading options are illustrated in Figure 1, in this case study referred to as 'traditional' sun shades.

3.5 Research on daylighting, solar radiation, energy efficient buildings and form-finding

The work discussed in this section outlines research of an equivalent nature to this case study where daylighting, solar radiation and energy efficient buildings are concerned.

3.5.1 Façade design optimisation for daylight with a simple genetic algorithm

Torres and Sakamoto [60] examine the applicability of a genetic algorithm to optimise a daylight system both to maximise energy savings and provide adequate interior lighting. The test is applied to a building façade where parameters can be changed - some parameters are changed simultaneously.

The results are analysed in the Radiance software package[xviii]. Fitness is evaluated by how much of the annual lighting requirement can be replaced by daylight. Several trials were run over 200 generations

To optimise the overall layout, components of the façade are predefined with values extracted as parameters - typical parameters include; window width, 'light-shelf' depth, number of windows and number of shades per window. Encoding sees each real-number being an alleles and the real-parameter vector a chromosome. This is illustrated in Figure 19.

Due to computational limitations, a subset of meteorological data is used together with absolute elitism to achieve faster convergence in fewer generations. Additionally, twenty one parameters were used as this allowed the search space to become overpopulated which assisted in achieving a fitter model in a faster time frame.

Tokyo, Japan, was used for the daylight simulation model and four 'observers' took simplified measurements for every time step. The control model featured random blind positioning (open or closed) that could reasonably model a building occupant's

[xviii] Radiance - a validated lighting simulation tool. `https://www.radiance-online.org/`

behaviour, the Radiance software calculation was taken for one in every twenty days throughout the year.

The Genetic Algorithm also saw simplification; populations were limited to 10 individuals. To avoid convergence to local maxima as a consequence of absolute elitist selections, 3 random individuals were included with the breeding group. Fitness was the proportion of total lighting that could be excluded (instead being replaced by natural daylight).

Different breeding methods were used: initially one parameter value was mutated. Three recombination methods were tested. Simple cross-over involved swapping one parameter value between two individuals. Additionally, interpolation and extrapolation cross-overs were tested where one parameter value was replaced by a value proportional to the fitness of individuals.

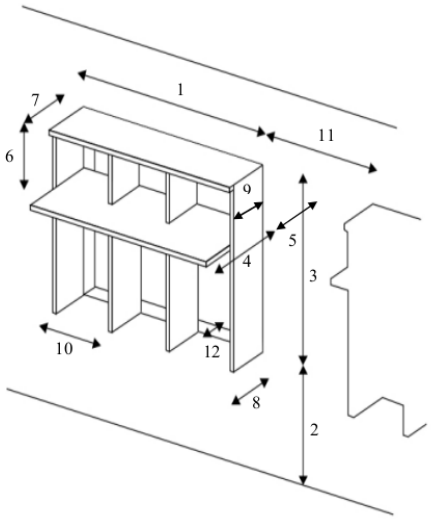


Figure 19: Façade design optimisation. (Torres & Sakamoto)

Results showed a fast increment of fitness slowing after 100 generations. This translates to the main elements of the façade changing initially with small changes in size taking place later. As daylight determined fitness, windows remained at the maximum height and width, protection elements above the light shelf disappeared (as no glare was caused) while vertical fins protected observers from lower altitude sun.

Torres and Sakamoto effectively use a genetic algorithm to re-arrange and resize existing components on a façade. No new artefact is created, but interesting discoveries are made (elimination of components above the light shelf). To assist with computation, a number of simplifications are made - populations are small and studies are done on certain days, rather than sequentially. Approaches within the Radiance software package together with the daylight calculations also saw simplification.

3.5.2 Genetic optimization of external shading devices

Manzan and Pinto [42] use genetic optimisation to design an external shading device for a window resulting in different optimal geometries for the panel. These devices were evaluated in simulations performed using the Radiance software package.

The research aims to address an Italian building regulation that requires compulsory external shading devices for buildings with over 1 000 square metres of total surface area. The goal being to reduce energy usage required for lighting, heating and cooling. Design of the shading device is not specified within the building regulations.

The following concerns need to be addressed in the design of the external shading device; the orientation of the façade, size of the window(s), importance of heating and cooling and the impact on internal lighting conditions together with the size and position of the shading device.

Manzan and Pinto describe how, typically, climatic conditions are separated from

lighting analysis in that shading device design usually does not consider both aspects simultaneously. Following Franzetti's 2004 work [30], fourteen parameters are identified to perform computations on within the simulation.

The test is on a room with a south facing window (direct sun); the window is 4 metres wide and 1.5 metres high while the room is 5 metres wide, 4 metres in depth and 2.7 metres high, see Figure 20. Different glazing systems are used with 8 photocells in the room for measurements (two rows of four). Six conditions are tested: standard glass with no reveal (glass flush with outside surface of wall), standard glass with reveal (glass set back in opening, not flush with outside wall) and standard glass with no shading device - this is repeated with high performance glass characteristics.

The shading device itself is a flat panel running the length of the room, inclined horizontally. The genetic algorithm optimises the panel using four parameters: shading height (start of panel measured from floor surface), shading width (how wide the panel can become - moving away from the surface of the building), angle, distance to wall surface.

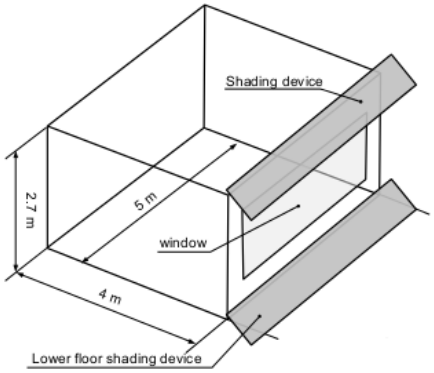


Figure 20: Optimization of external shading devices. (Manzan & Pinto)

The energy simulation software (ESP-r) [28] required modification to work with inclined flat panels. The simulation (ESP-r is connected to the Radiance simulation) gets inserted into the optimisation loop - this results in large numbers of simulations being run. To assist with computation, a user defined method is employed: measurements of a defined shading device created in Radiance is used to drive the artificial lighting calculation.

modeFRONTIER software[xix] using the 'workflow designer' tool (a visual programming environment) allowed for optimisation algorithms to be selected and tweaked. The optimisation method built-in to the modeFRONTIER software is the MOGA II algorithm - considered strong when working with multi-objective problems[xx].

MOGA II is briefly discussed when being compared to the NBI algorithm [54] for multi-objective optimisations. MOGA-II uses 'smart multisearch elitism' (for 'robust' and 'directional crossover') to allow for fast convergence. Encoding in MOGA-II is the same as in classical genetic algorithms (binary encoding where each chromosome is represented as a string of 1 or 0). For reproduction, an individual is selected - the selection of individuals can use any schema but typically local tournament (with random steps in a toroidal grid) is used. This initial individual then takes a 'random walk' (of an assigned number of steps - the number of steps remaining fixed during the optimisation run and proportional to the population size). Other individuals met in the first walk are marked as candidates for first parents while a second random walk, again from

[xix] modeFRONTIER is the leading software solution for simulation process automation and design optimization. `https://engineering.esteco.com/modefrontier/`

[xx] MOGA II is implemented within the modeFRONTIER software and is an improvement on MOGA - Multi-Objective Genetic Algorithm [50]. MOGA II follows a stochastic method.

the starting point, creates a list of second parents. When the set of candidates are established, the fittest is chosen for reproduction. At each step of reproduction, one of four reproduction operators is chosen and applied to the current individual. The four operators being; classical crossover, directional crossover (this assumes a 'direction of improvement' that can be detected by comparing fitness values of two individuals - the direction of improvement is evaluated by comparing the fitness of the individual (i) from generation (t) with fitness of its parents (generation t-1)), mutation and selection. A novel operator, 'evolutionary direction crossover', was introduced to replace the classical crossover operator (this new operator showed good performance as a complex multimodal function). A new individual is created by moving in a randomly weighted direction that lies within those given by individual and its parents.

Optimisation considered 16 individuals in 100 generations. Limitations were placed on the (horizontal) length of the shading device and view for the building occupants. Fitness is based on the lowest value achieved for primary energy requirements - this was in the order of 100 kWh/m^2 kilowatts.

After a few generations convergence was achieved. The results indicated that a wider shading device produced reduced energy consumption. If the panel width is constrained, a higher angle (panel rotated downwards) offers the best solution.

Manzan and Pinto's work is encased in a software application making it difficult to explore the specific changes they experimented with (within the visual programming environment) to better understand the changes they implemented as they ran their experiment. Additionally, they optimise a pre-defined design rather than explore new forms. They do, however, use simulations to verify results and as noted, include both climatic and lighting conditions in their optimisation.

3.5.3 Geometric optimization of fenestration

Wright and Mourshed [64] consider window shape when working to reduce the impact of energy use for heating and cooling a building. They believe most studies concern themselves with 'regular geometric shape and position' of windows in a façade. Their study uses a genetic algorithm to optimise the state of a cell on a façade to drive the design of fenestration.

Optimisation considers the shape, amount and position of windows in a façade. To achieve this, the façade is divided into rectangular cells each being either solid or glass. It is hoped that solutions generated could be architecturally similar to the 'wall of light' (south façade) of Le Corbusier's Chapel of Notre Dame du Haut.

While the genetic algorithm sets the number and location of cells on the façade the user is given access to parameters than can assist in the design - they act as constraints in the algorithm. The design constraints are: amount of windows, area of the window surface, the window aspect ratio, density of windows on the façade and the 'centre of gravity' for each window.

A façade 15 cells wide by 8 cells high is considered. A collection of adjoining cells is considered a window - three such windows are identified in Figure 21.

The genetic algorithm features Gray binary encoding (variables are binary encoded), tournament selection (with a stochastic ranking algorithm determining fitness), 'uniform' crossover (100% probability for chromosome crossover, 50% for binary gene crossover),

one gene mutation per chromosome and elitism (only the single best solution of the previous generation is used for the new solution). The algorithm ends after 1 000 generations for unconstrained optimisation and 3 000 generations for constrained optimisations (as they are considered more demanding); 30 individuals make up the population size (population size is 're-initialised' should a population collapse take place before completion of the search).

To a greater or lesser degree, the optimisations outlined above work within specific set boundaries (parameters being predefined) going so far as to include 'control over the form' - i.e. how the results will look.

While Wright and Mourshed hope to consider the design of the fenestration on a façade - being unhappy with 'regular geometric shape and position' - the grid system they place on the surface enforces geometric shape and position. They fail to open the façade to free-form, unconstrained solutions. Instead, they choose to define a window using certain metrics and then force regular geometry onto the results. Furthermore, they incorporate optimisations that allow the user to control the form - not quite allowing for form-finding with results that are unpredictable.

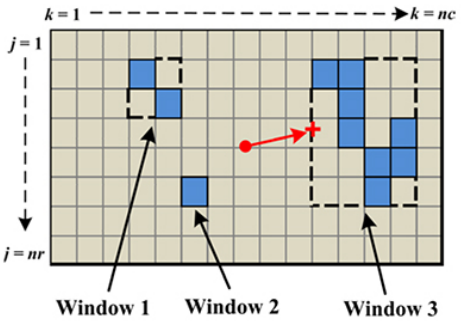


Figure 21: Geometric optimisation of fenestration. (Wright & Mourshed)

3.5.4 Optimal Building Envelope Design - History, current status, new trends

Huang and Niu [38] provide a survey of optimal building envelope design (the design of which is seen as important in Green buildings). Central to building envelope design is energy consumption - by better managing energy consumption one can save on building running costs, reduce the impact of the building upon the environment or address comfort for the building occupants. As computing power has increased, so too has simulation software improved, broadening the scope for further research. However, the parameters that require testing during simulations are many - resulting in lengthy simulation times. As a consequence of this, an interest in 'systematic and effective optimisation processes' for sustainable building design has developed. The building performance model[xxi] together with the algorithms selected, play a role in optimisation. In the survey, Huang and Niu outline the development of algorithms to optimise building envelope design and highlight Evolutionary Algorithms as being the most used optimisation method; found in 60% of the research surveyed with Genetic Algorithms dominating. Huang and Niu go on to say: *"The GA and its modifications are considered*

[xxi] Building performance is interested in HVAC (Heating, ventilation and air conditioning) and building envelope design; the latter having more parameters requiring optimisation. Furthermore, the relationship between parameters and to the performance of the building envelope is complex.

to be the best choices for solving building design optimization problems." A survey of approaches to, and modifications of, Genetic Algorithms was undertaken and a table drawn up, this is shown in Figure 22.

Different modification approaches applied for the improvement of the genetic algorithm's performance.

Names of researchers	Modification approach	Major finding	Reference
Znouda et al.	Applying elitism selection instead of traditional wheel selection. Introducing an immigration procedure into the mutation process.	The speed and the accuracy of the genetic algorithm search engines were largely improved.	[62]
Palonen et al.	Applying an omni-optimizer, a simulated binary crossover operator and a polynomial mutation in crossover and mutation operations.	The stopping criterion definition was very important for the reduction of computing time.	[63]
Kämpf and Robinson Ramallo-González and Coley	Application of hybrid covariance matrix adaptation evolutionary strategy and HDE algorithms.	The covariance matrix was used for decorrelation of design variables. Although the calculation of the covariance matrix was time-consuming (compared with the computing time used for annual building simulation), it was acceptable.	[64,65]
Eisenhower et al. Stavrakakis et al. Geyer and Schlüter	Applying multiple algorithms for building envelope optimization. Applying a meta-model approach to avoid repeating simulation cases. Introducing design space exploration into the traditional response surface method to further improve the method's accuracy.	The proposed approach could save around 80% of the computing time with a negligible change in the degree of error. The degree of error could be reduced by more than 70% when design space exploration was utilized.	[66,59,67]
Bucking et al.	Considering mutual information (dependency between variables) during the evolution of the design cases.	With the consideration of mutual information, the data mining with genetic algorithm optimization could save 40% of the computing time and improve the accuracy by 25%.	[68]
Junghans and Darde	Combining a genetic algorithm and a simulated annealing algorithm.	In 1/3 of the optimization runs, the accuracy was improved by at least 5%, but the computing time did not change significantly.	[69]
Yu et al.	Applying the genetic algorithm into the back propagation neural network.	A small relative error (of 1.7% for energy consumption and 2.1% for indoor thermal comfort hours) was reported on the prediction.	[60]

Figure 22: Table - approaches to and modifications of Genetic Algorithms (Huang & Niu)

Huang and Niu conclude that for simple optimisation, gradient-deterministic direct searches were used; while for complex optimisation, Evolutionary Algorithms were used and were found to run faster and perform more accurately. Of the Evolutionary Algorithms, Genetic Algorithms were most used as it was found they were widely applicable and performed with high speed and accuracy. Most research conducted in this field considered the optimisation of energy-related performance; similarly, most research centred on single-objective optimisation. Visual and thermal comfort was less researched. Less multi-objective optimisation was performed and this was mostly done optimising thermal and energy performance.

It is clear, Evolutionary Algorithms play a role in architectural design - but from the survey results - this seems mostly focused on performance optimisation. The exploration of form, in the context of this survey, is not common.

3.5.5 Shape optimization of free-form buildings

Zhang et al. plan to heat buildings in a cold climate by using passive means [67]. They focus on shape optimisation to assist with this task. They start by generating a 'free-form' shape using controllable parameters - this allows the shape to be modified by means of multi-objective optimisation. They use Rhinoceros[xxii] with the Grasshopper plug-in as the software to generate the parametric free-form shape; Grasshopper grants

[xxii] Rhino (Rhinoceros3D) software `https://www.rhino3d.com/`

access to the free-form model's parameters. Ladybug (itself a Grasshopper plug-in) opens the model to climatic simulation - the data of which is contained in an EnergyPlus file (.epw). A Genetic Algorithm is used to perform multi-objective optimisation, the GA is embedded as an additional Grasshopper plug-in - Octopus[xxiii]. As the authors note: *"Octopus applies evolutionary principles to parametric design and solution finding, allows for a multi-objectives optimization process, and allows for the production of a set of trade-off solutions among each objective. Importantly, Octopus still allows the whole optimization process to be visual and controllable."*

Zhang et al. work to shape-optimise a community centre in Shenyang. They view the community centre's main function as that of a sports centre/cultural space and consider this the most important constraint around which optimisation should take place. Secondary spaces are arranged around this dominant space and generally include movement. In addition, outdoor spaces are required for activities; the area to the west is for parking (as traffic dominates that side of the site), to the south side an open space is accommodated as both a playground and to keep shadows created by neighbouring buildings away from the new building design, see Figure 23. Taken together, this information is used to generate the variables that will govern shape generation. The building shape is made out of two dominant curves, one to constrain the plan and one the form of the volume. Rhino is used to generate the form which contains rectangular volumes that act as curve control point limits (see Figure 3). Next, the form was tweaked; the west entrance area was enlarged to create a vestibule and allow more traffic while the south side surface (together with the forms generated on the roof) was modified to allow more solar radiation to penetrate the building to increase heat build-up.

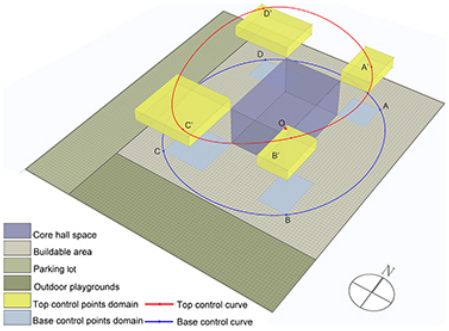


Figure 23: Free-form building shape optimisation. (Zhang et al.)

For optimisation, variable increment steps were set (to 0.1) and co-ordinate points defined to match X, Y and Z Cartesian values. The time range was set to match the months requiring heating (November to March). Overall, 20 variables controlled the building shape generation, some with constraints applied (for example: the shape co-efficient variable, when sufficiently decreased, will result in the control points for the curve controlling the plan to approach a maximum and effectively become a constant). An additional building, in the shape of a cube, is simulated and tested against as a reference building.

Zhang et al., when discussing the advantages of Genetic Algorithms, go on to say; *"The genetic algorithm is often used in simulation-based optimization problems because it presents several advantages: GA does not require the objective function to be continuous; it is a global search technique that can escape from local optima more easily, and it can find multiple Pareto solutions for a multi-objective optimization problem in one run."*

[xxiii] Grasshopper plug-in: Octopus (multi-objective evolutionary optimization) `https://www.food4rhino.com/en/app/octopus`

These observations themselves are taken from Wang, et al. [62]

The Genetic Algorithm had a population size of 50 with the maximum number of generations being 100. The crossover rate was set to 0.8, mutation probability to 0.1, mutation rate to 0.5 and elitism to 0.5 within the Grasshopper Octopus plug-in. The Octopus plug-in uses algorithms based on SPEA-2 [68] and HypE [7] from ETH Zurich, implemented using Pisa software[xxiv] [11].

It was found that after 100 generations, 84 non-dominated solutions were created forming the Parento frontier and expressed as a curved surface, the Pareto frontier is shown in Figure 24. Each solution had strengths and all were found to be valid final designs. The best and worst solutions were compared resulting in 10 final solutions that provided the best optimisation in terms of solar radiation, shape co-efficient and space efficiency, together with the Cartesian values that make up the free-form shape - the base control points.

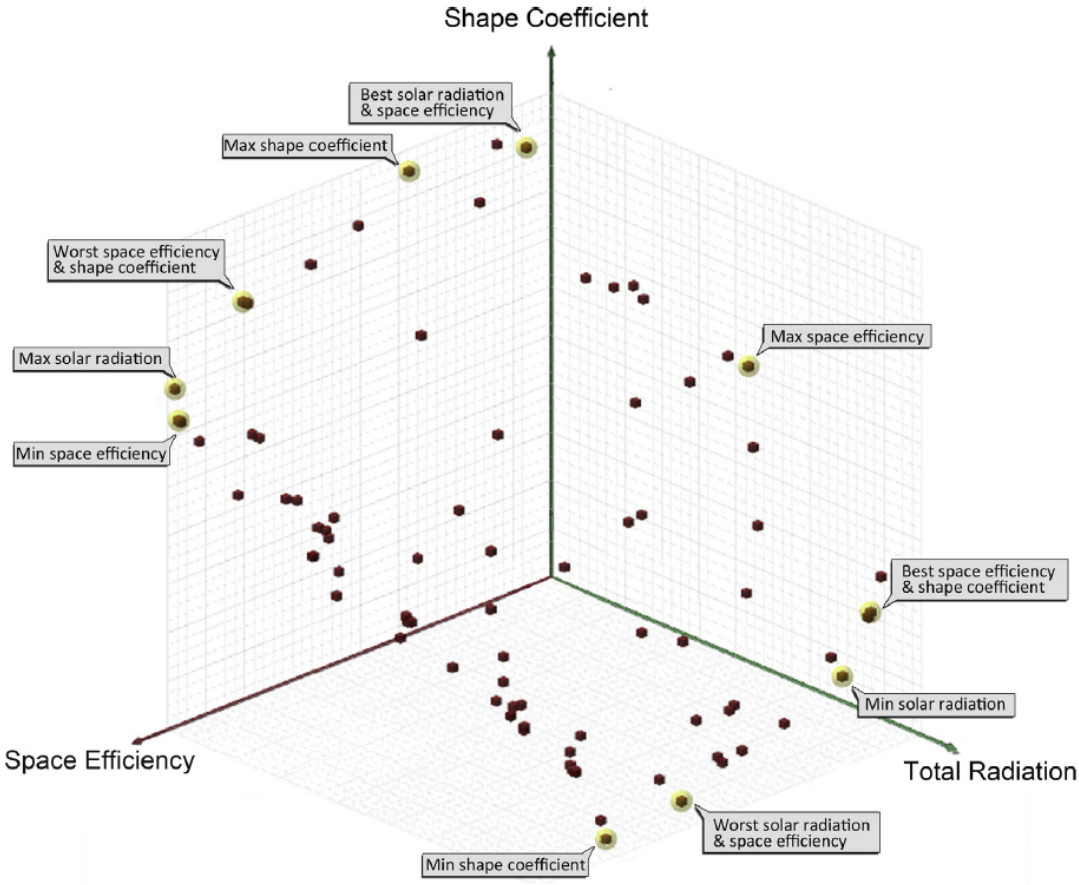


Figure 24: Pareto Frontier of the optimization. (Zhang et al.)

The free-form building resulting from the optimisation was shown to have better characteristics than the reference cube with improved solar radiation gain (53% more),

[xxiv] PISA - A Platform and Programming Language Independent Interface for Search Algorithms: `https://sop.tik.ee.ethz.ch/pisa/`

a lower minimum shape co-efficient value (20% less; indicating better energy saving related to the building shape) and higher space efficiency (more than 95%).

The results are clearly better, but the greatest advantage was found to be that the Octopus plug-in generated three-dimensional models for each solution. A three-dimensional, free-form shaped building was available for evaluation for every Pareto frontier solution; every optimal shape could be visually evaluated. In architectural design this is found to be very valuable; the ability to find the optimal building shape by balancing between solar radiation gain, shape co-efficient and shape efficiency, then to have this shape available as a three-dimensional model. This research links parameter optimisation with shape optimisation and produces a series of artefacts for study.

3.5.6 Robotic Form-Finding and Construction

Zexin and Mei [66] revisit architectural form-finding and apply this to a robotic arm to digitally generate the resultant shapes. Figure 25 shows the forms created when using architectural projection as a generator of form-finding.

Zexin and Mei start by discussing the development of form-finding, from Gaudí to Frei Otto, then proceed to discuss 'traditional' architectural representation (drawings) in this context, noting: *"From the 'nonlinear system theory' to the 'parametric design', the computational technology has brought a completely new medium to the architectural design, and it's even altering the fundamental theories and thinking processes behind it. With the popularization of digital software, the mechanics of how architects work have changed; less and less are using traditional architectural drawings as a from-finding tool."* Interestingly, Zexin and Mei add this of Gaudí's work: *"Through using this completely new method – which is based on motion dynamics [statics] and gravity, Gaudí invented his own architecture languages and achieved his greatest architectural goal - to perfect and go beyond Gothic style."*

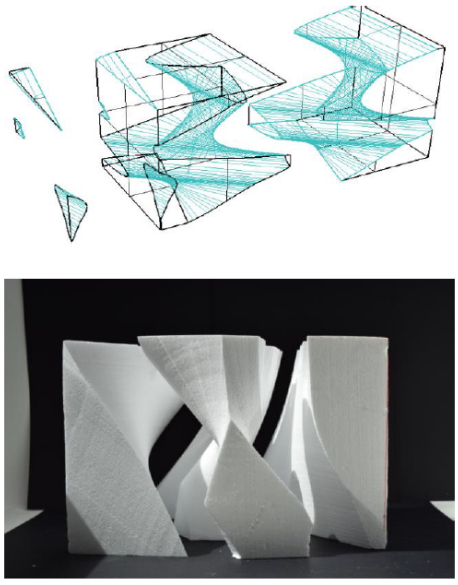


Figure 25: Cut foam following the logic of architectural projection. (Zexin & Mei)

It is in this sense that 'traditional architectural drawings' get discussed. Traditional architectural drawings, using orthographic projections[xxv], it is

[xxv] Orthographic drawings use parallel projections to transfer the building information to a plane. Parallel projections project lines perpendicular to the drawing plane and use true shapes and sizes with no distortion to represent surfaces. As each plane can only show one projection, 'multiview drawings' are used to show the building in three-dimensions creating 'plan' views (a 'top' orthographic projection cast on the horizontal picture plane) and elevation views (front/side orthographic projections cast on vertical picture planes). These conventions are well documented [52] [15] and are typically taught in

argued, have been in use since the Renaissance.
While architects were originally directly involved in building construction, since the Renaissance this connection was removed instead seeing architects work on drawings that connect to construction. Over time, the drawing methods of projection led to influence the forms reproduced in architecture. Zexin and Mei propose there is a disconnect between architectural drawings and the ability to express new, complex, 3D forms. They question how traditional drawings can be redefined or reused for form-finding. Their solution is to analyse and consider the impact history and the evolution of architectural drawings and paintings (specifically their projections) have had on form. Working through parallel projections in a cubic space to perspectives with vanishing points, they end with **Stereotomy** - a combination of parallel projection and perspective projection, illustrated in Figure 26. It is believed this is a 'multi-dimensional composite' projection which can lead to unpredictable forms as 'the workspace has been changed'.

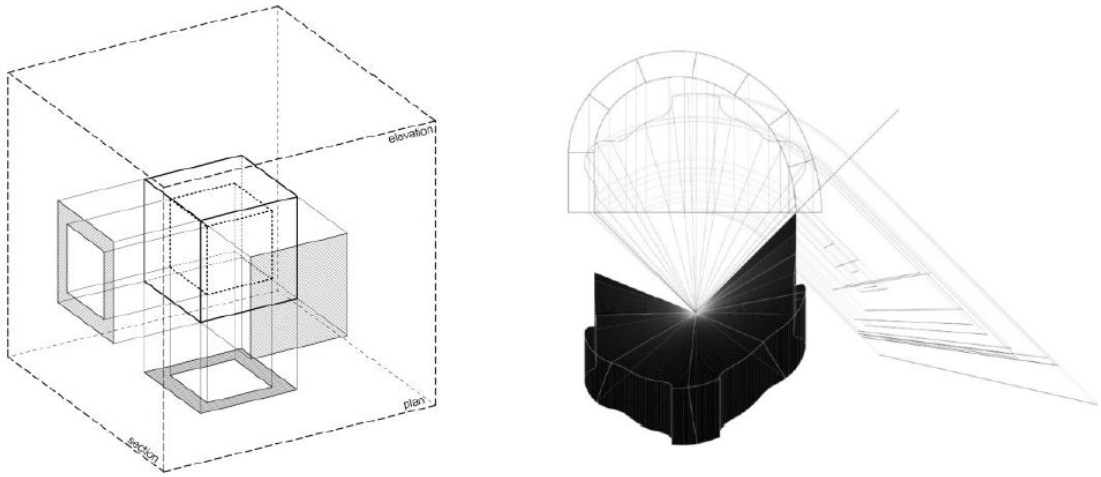


Figure 26: Architectural projection - traditional and stereotomy with perspective. (Zexin & Mei)

A six-axis, articulated, robotic arm is employed to 'connect drawings with construction'. The movement of the arm is controlled by computer instructions and these follow the idea of inverse kinematics. Robotic arms are finding use in various architectural applications where they lay components to create a parametric artefact - curved walls, domes, and so on. Here the robotic arm is used with a hot-wire cutter to slice ruled surfaces from foam cubes. The robotic arm is simulated in Rhino using a Grasshopper plug-in where the trajectory of the arm creates a 'ruled surface', a straight line sweeping through two curves. Manipulating these curves (the rails) is fundamental to this form finding experiment. A cylindrical projection is used to control the curves; the curves are mapped to an unrolled cylinder using a Cartesian co-ordinate system.

Zexin and Mei are not clear on what determines the rail shapes that get projected to the cylindrical surface that in turn form the ruled surfaces cut from the foam. Similarly,

foundation courses.

after 'deconstructing' various architectural drawing projections and redefining a new workspace, they return to the more predictable Cartesian co-ordinate system and a relatively simple cylindrical projection.

3.5.7 Comments regarding the Research in daylighting, solar radiation, building envelopes

Torres and Sakamoto blend using a genetic algorithm with daylight simulation software to produce an architectural artefact with an optimised result. Their research best demonstrates how an evolutionary algorithm can be combined with architectural components and simulated results to modify architectural designs. This research provides a sound introduction on how an architectural problem can be translated and incorporated into an evolutionary algorithm. An architectural problem is converted into a set of parameters which, in turn, are seen as real-number variables. In the evolutionary algorithm these are seen as alleles where the 'real-parameter' vectors become chromosomes. Unfortunately, given their computational demands - by incorporating an environmental simulation - the complexity of the evolutionary algorithm suffers (21 parameters within a part of a façade, a population of 10 individuals, absolute elitism where 3 individuals provide for more variation in breeding over 200 generations).

Manzan and Pinto, in a later study, also combine an evolutionary algorithm with a daylight simulation but use a 'software tool' (modeFRONTIER) to implement the evolutionary algorithm. This results in a more complex daylight simulation together with better integration but at the expense of using a 'pure' evolutionary algorithm, instead a 'preset' implementation is in place and a software interface guides the changes. The Evolutionary Algorithm used in the software application is a multiobjective Genetic Algorithm. Four input parameters are used initially covering a broad range that are optimised to a narrower set, these are then optimised again for a final result (a population of 16 individuals is used across 100 generations). This work is significant for the complex simulation(s) used to optimise the shading device while the genetic algorithm simply supports the different configurations allowing the designer to select the best implementation given a set of circumstances.

Wright and Mourshed, like Torres and Sakamoto, combine a genetic algorithm with an architectural optimisation problem (window layout and opening sizes). However, unlike Torres and Sakamoto, they build in an aspect that allows the designer to control the result. In the 'unconstrained optimisation' solution the genetic algorithm's optimised results indicated a bias to place windows high up and to the west of the façade. This position makes sense, given the constraints supplied - optimum area, low energy use, good natural light and a reduction in heating requirements. The 'constrained aspect ratio' solution resulted in fewer windows but with them having a similar surface area. The solution where the 'number of windows' in the façade is constrained does not compete with solutions that generate the lowest energy use. The genetic algorithm used random initialised variables with Gray binary encoding of these variables, binary tournament selection (with fitness being assigned by a stochastic ranking algorithm), uniform crossover (100% probability of chromosome and 50% probability of binary gene crossover) resulting in 1 gene mutation per chromosome, elitist selection (the single best solution of the previous generation included in the next). The population size

was of 30 individuals and the search continued until unique solutions were found (1000 for unconstrained optimisation and 3000 for constrained optimisation). Overall, the research indicates that the genetic algorithm worked correctly, but the designer could change the look of the layout if what was produced was seen as undesirable. This is unfortunate as having unexpected results allows designers to interrogate why these solutions could be best given the constraints supplied.

Huang and Niu provide a comprehensive survey of building envelope design research focused on parameter optimisation combined with simulated performance. What stands out is how dominant Genetic Algorithms are in this field together with research concerning the performance of these algorithms within this field. This is significant as typically building envelope design takes precedent; here Huang and Niu indicate research done where the performance of the algorithm is important. This indicates quite a mature research field where both the problem being studied is addressed (building envelope design) together with the tools used to accomplish this research (in this case, Genetic Algorithms).

Zhang et al. work to implement a 'framework' they call: Modelling - Simulation - Optimization. Their idea is to move seamlessly between modelling a building while using a genetic algorithm to optimise the resultant shape. They see this as 'a performance driven approach to find a solution' to a given design problem. A consequence of this approach is that they use 'off the shelf' software[xxvi] to implement their framework; Rhinoceros with the Grasshopper plug-in (a visual programming environment) creates the model and exposes parameters for shape optimisation. The model is 'dynamic' in the sense that changes can take place 'on the fly' - this feeds into a simulation tool (using the Radiance lighting software and the Grasshopper plug-in, Ladybug). The results of the simulation are sent to another Grasshopper plug-in, Octopus, a multi-objective genetic algorithm. This plug-in allows certain settings to be modified to drive the genetic algorithm, however, the most impactful aspect is how this connects to the model driving the parameters directly. This is due to how the plug-ins work together within Grasshopper. Their ability to work directly with the genetic algorithm and manage the evolutionary process is secondary to the integrated software producing an outputted artefact.

Zexin and Mei discuss traditional architectural representation methods (drawings with orthographic views) and work to 'subvert' them to explore form-finding using a robotic arm. However, the method they finally use to produce an artefact is, in essence, a ruled surface - one of the earliest modelling options available for initial solid modelling CAD software programmes. Something in their work breaks down - it could be the constraint imposed upon the work by the robotic arm. This work is difficult to understand as the result produced seems to be unconnected to their initial discussion on blending traditional and 'modern' drawing projection methods.

[xxvi] One is reminded of Frazer when he was starting his research and indicated his preference for starting from first principals and avoiding commercial software was preferable.

3.6 General Comments

The potential use of computers in Architectural research has a long history that references core design aspects starting with Thompson breaking form into mathematical qualities to Alexander's view of opposing design intuition with a mathematical approach relying on the application of logic. This being further extended by Alexander in his work to break down complex problems into interlinked subsystems (or patterns). Mitchell, quite early on, powerfully connects computers to architecture by drawing parallels between the logic underpinning algorithms and that of architectural languages or 'architectural grammar'. However, the iconic link to Evolutionary processes is centred on the work by Frazer but detailed access to particular methods are lacking - instead an overview of what was accomplished is noted. After Frazer research work seems to become more conservative - the era of big ideas seems to have passed. More direct approaches seem to follow; exploring Le Corbusier's form grammar and connecting his rather 'formulaic' approach to form by employing shape grammar, as shown by Coates and Makris and later Asojo. Similarly, revisiting the original protagonists of form-finding, Gaudí and Otto who explored form by creating, revived an interest in computational form-finding and spawned associated architectural theories and movements (Organicism, Computational and Parametric Architecture to name a few). But architectural acceptance of this work seems rare, even after it was incorporated into architectural theory (De Landa). Instead, it seems Evolutionary Algorithms at this time are extensively used in architectural research where they dominate multi-objective optimisation, as would be typically required in Building Envelope design and simulation. There is far less traction in architectural design application - where they would be ideally suited to support architectural form-finding.

4 Methods

4.1 Introduction

This case study hopes to answer the question concerning whether an evolved architectural sun-shade can perform better than a traditional architectural solution when attempting to block sun rays from entering a window opening.

This work considers the creation of an architectural sun-shading device by using an Evolutionary Algorithm to evolve a point cloud to best protect a window opening within a wall surface and is a more robust exploration of the initial work previously documented [18]. Figure 27 demonstrates the result of applying an Evolutionary Algorithm[xxvii] to a point cloud while Figure 28 represents the point cloud as a meshed surface (viewed from below). This evolved solution is then compared to ten 'traditional' architectural sun-shades to verify if an evolved sun-shade can produce a better performing sun-shade compared to the forms currently in use. In addition, architecturally, this case study is interesting as an Evolutionary Algorithm is used as a form-finding device.

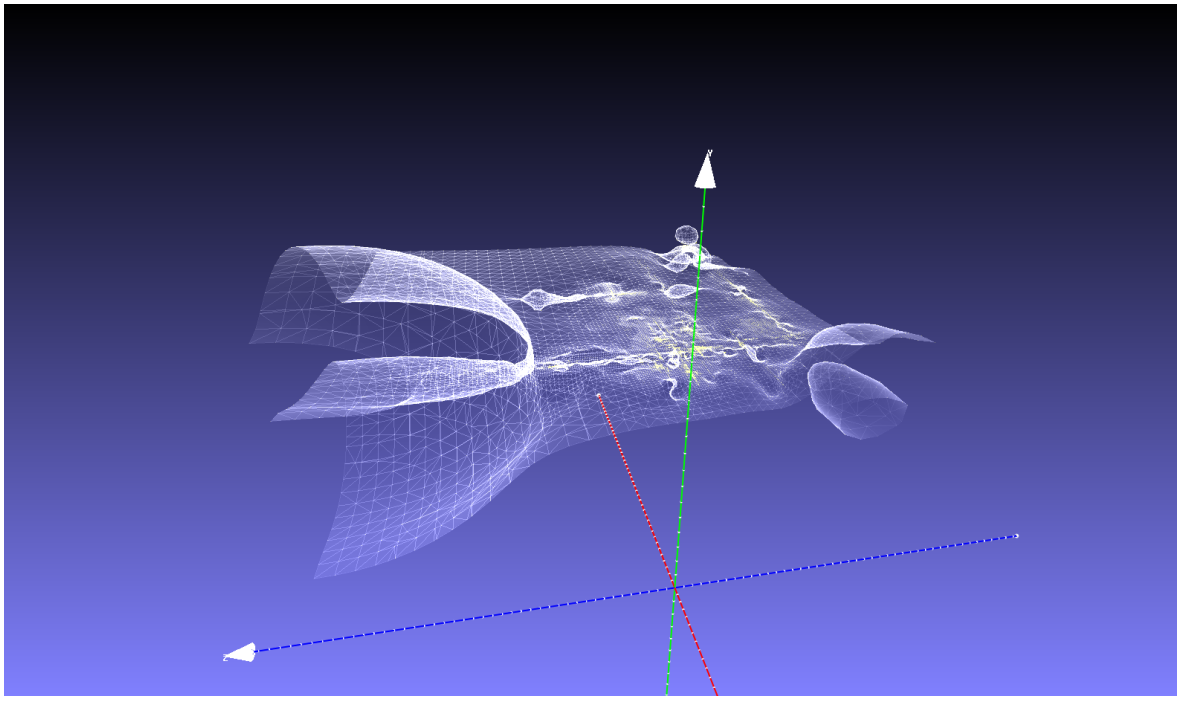


Figure 27: Point cloud of an evolved sun-shade in MeshLab. (Author)

[xxvii]This particular image was generated by the 'Amsterdam East' first run evolutionary algorithm. Of particular interest is the curved area (to the left in the image) located furthest from the wall surface (to the right) and towards the sunrise shielding the window opening against early horizontal direct sun rays.

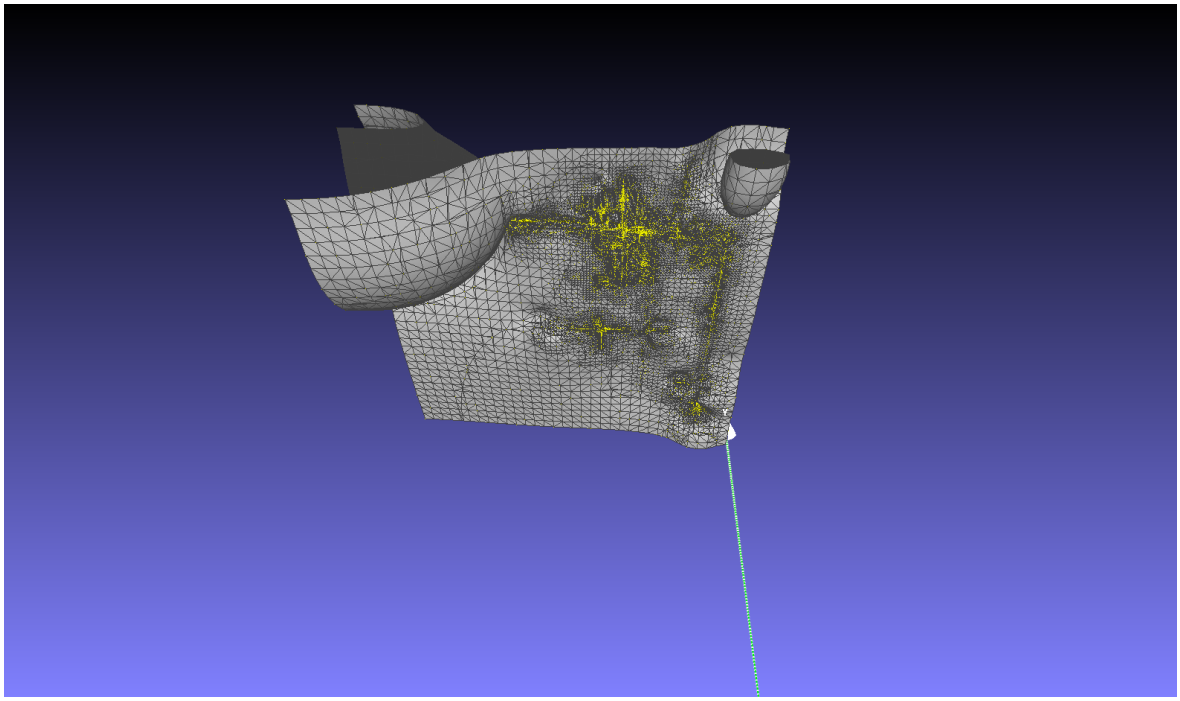


Figure 28: Surfaced point cloud of the evolved sun-shade (viewed from underneath) in MeshLab. (Author)

The workflow to produce the meshed sun-shade can be outlined as follows; a random point cloud (using Euclidean Cartesian co-ordinate values) is generated within a 'bounding box' set outside a wall opening. Each point within the point cloud is then evaluated by testing how it interacts with solar values (the altitude and the compass direction of the sun). The point is 'saturated' with values as they are measurements taken at fifteen second intervals across the Summer Solstice. The points that best keep out the sun - the most values - are kept to finally produce a resultant point cloud. Architecturally, MeshLab [16] can then be used to visualise the resulting mesh[xxviii], allowing 3D printing. (This can be incorporated into a scaled architectural model representing a building design or printed directly for installation in-situ, Figure 45 shows an evolved sun-shade in context).

This case study, however, focuses on the Evolutionary Algorithm required to evolve the point cloud together with establishing a method with which to judge the fitness of traditional sun-shades to compare fitness values.

4.2 Background

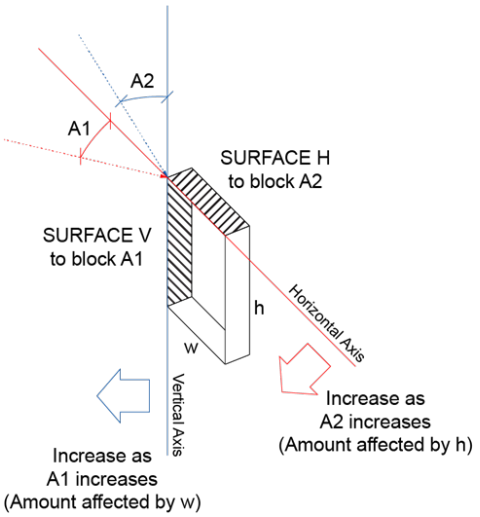
To determine if a point in space can block a ray of sunlight from striking a surface, we need to consider how sunlight is approached in building design, specifically solar angles.

[xxviii] Additional meshed point clouds can be found in **Experiments and Results** for a visualisation of other evolved sun-shades.

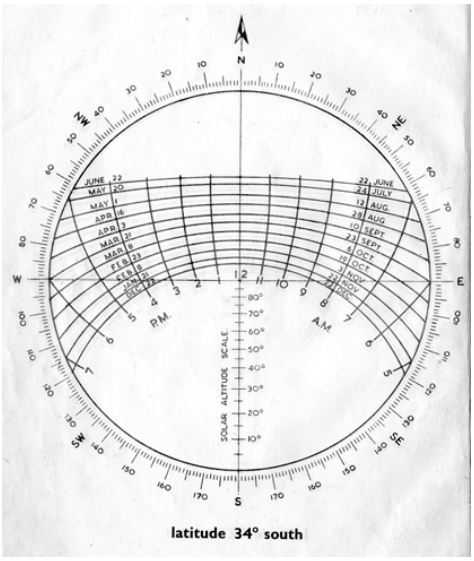
4.2.1 Solar Angles

Typically, in Architecture, the approach to working with Solar Angles starts by dividing the solar angle (the sunlight penetrating the building) into two vectors, a horizontal and vertical vector. Figure 29a labels these vectors A1 (horizontal) and A2 (vertical). The goal behind visualising the solar angle as two vectors is to allow it to better fit with architectural drawings; the horizontal axis (A1) correlates to a horizontal plane and relates to the building floor plan drawing while the vertical axis (A2) correlates to a vertical plane, most often a building side view (a drawing called an elevation) or a vertical slice through the building (called a section).

To assist with the deconstruction of a solar angle into the horizontal and vertical vectors, a solar protractor is used (this device would be comparable to using a logarithmic table book to determine logarithmic values).



(a) Solar Angle as two vectors. (Author)



(b) Solar protractor issued by the South African National Building Authority for use at latitude 34° south, i.e. Cape Town. (SANBA, 1981)

Figure 29: Solar angle vectors and the solar protractor

Figure 29b illustrates a solar protractor issued by the South African National Building Authority (1981) [53] for use at latitude 34° South, i.e. Cape Town.

The protractor is often copied onto a transparency sheet and is then used by placing the 'crosshairs' (the intersection of the North/South and East/West axis) at the area under examination on a drawing floor plan; for example, the outer wall surface at the centre of a window. The protractor is then swivelled around this 'crosshair' and aligned to North on the floor plan.

The month containing the solar angle under review is noted and marked by the horizontal curving bands within the protractor while the hours of the day are marked by the vertical curved lines (as noted in the protractor) with midday coinciding with the North/South axis.

The angle of the building, relative to North, determines the initial horizontal solar angle (A1) for the side of the building in question. This gives a visual indication of which part of the building will receive sunlight at the time under review. To determine this starting angle, a line is extended along the building edge until it reaches the outside rim of the solar protractor giving the horizontal solar angle. The hour line cut by the building edge indicates the time from which that façade will experience direct sunlight. The hours (where they connect with the month line) can then be marked and lines drawn from the 'crosshair' through the hour marks will connect with the outside rim of the protractor to indicate the horizontal angle which changes over time. (This is shown in Figure 30 (represented by the line) and in Figure 34 in an application.)

To extract the vertical solar angle (A2), a compass is positioned at the intersection of the 'crosshair' and the distance is set to the hour/month intersection. This radius, when swivelled down, intersects a vertical 'solar altitude scale' value. This angle indicates the height of the sun above the horizon at the given time (on the given month). This is indicated by the circle in Figure 30.

Environmental factors, like neighbouring buildings or mountains will have an impact on the solar altitude scale measurement. It may be that on the 22nd of December, at 4pm, the solar altitude scale indicates an angle of 37 degrees above the horizon while the sun is at 95.5 degrees (5.5 degrees south of direct West[xxix]). But this could coincide with an external environmental factor and make the reading irrelevant. It's useful to work out pre-existing angles caused by environmental factors as this allows the study to be constrained within limits.

[xxix]The traditional solar protractor measures values in two halves. North is 0 degrees then going down to East or West the protractor marks their values as 90 degrees. This differs from the digital reading where the values are read clockwise with East as 90 degrees, South as 180 degrees and West as 270 degrees.

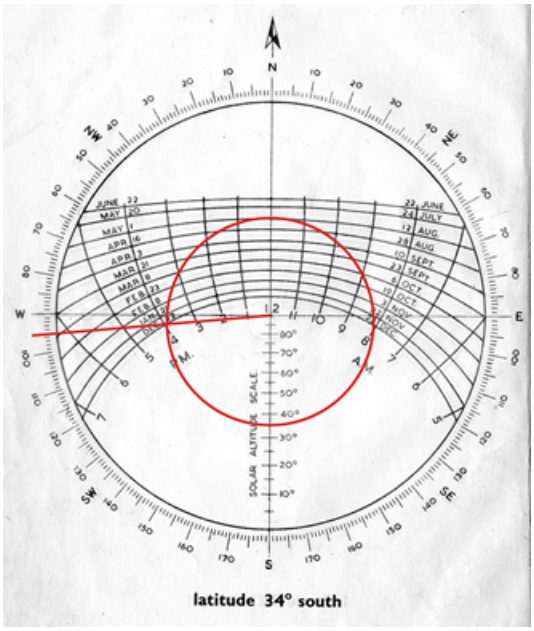


Figure 30: Solar protractor marked to indicate how the vertical solar angle is retrieved (Author)

4.2.2 Solar values as an example

To illustrates the above method, while simultaneously providing a sample solution space, the following example may be considered. This focuses on the Upper Campus at the University of Cape Town, then a sample building and thereafter a space within this building (an office). The sample building, as a floor plan, is shown in Figure 31 while Figure 32 indicates the space within the floor plans - here we are only interested in the West façade (or condition). In-situ measurements may be taken to ensure both the general accuracy of the work and to supply constraints to the solution space.

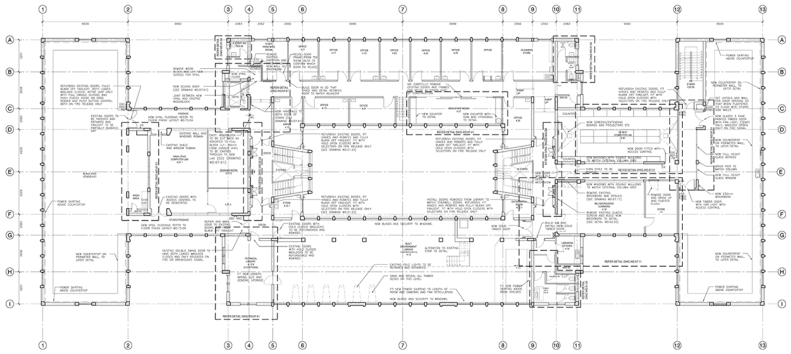


Figure 31: Sample building, West façade is to the top. (Author, after work by MLH Architects)

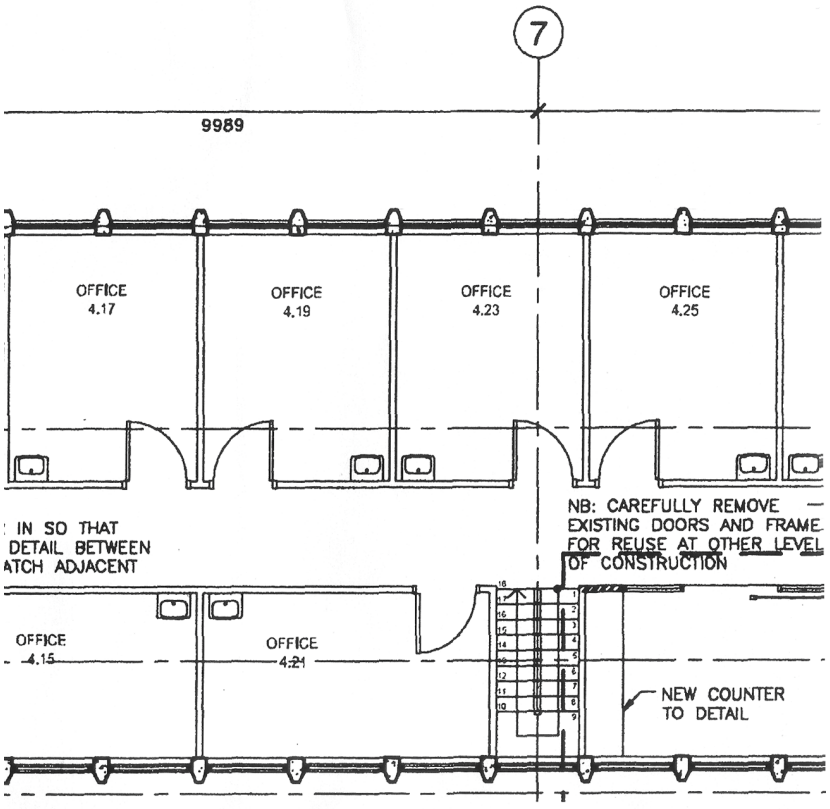


Figure 32: Office spaces within a sample building, Upper Campus, UCT. (Author, after work by MLH Architects)

The sample building is located to the south of Upper Campus, with a higher storied building to the West and then a mountain behind that. In-situ measurements confirm that the higher storied building will restrict solar penetration while the mountain is located further South West and does not affect the study as the building intervenes earlier. The sample building is on a lower elevation as the University of Cape Town's Upper Campus is built on the slope of a mountain.

The building footprint is rectangular with the long axis running North/South resulting in the longer façades being on the East and West side. Geographic Information System (GIS) data indicates that the sample building is oriented 22.8-degrees North-East (Figure 33). This results in the West façade receiving direct sunlight slightly before midday when the sun would be directly North.

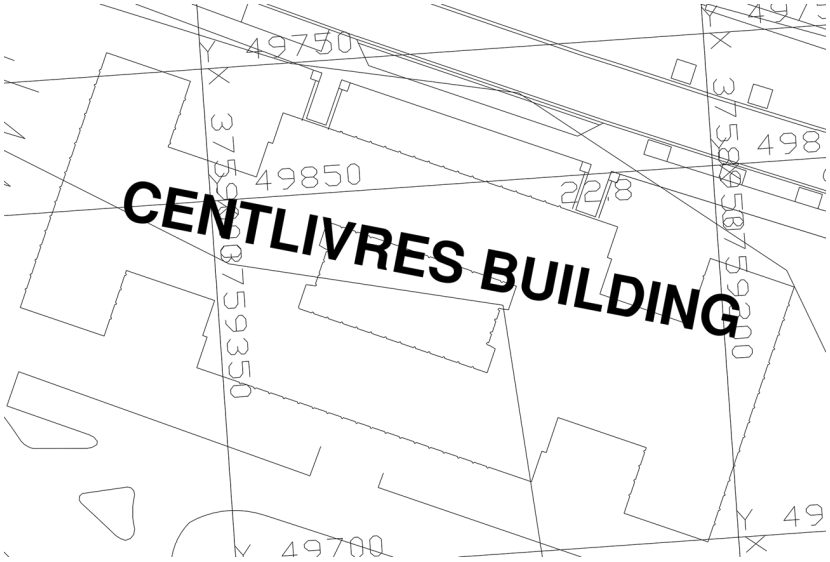


Figure 33: Sample building, Footprint and Orientation with GIS co-ordinate system. (Author)

Using the Solar Protractor and the working method outlined above, measurements can be taken of the sample building's West façade (Figure 34).

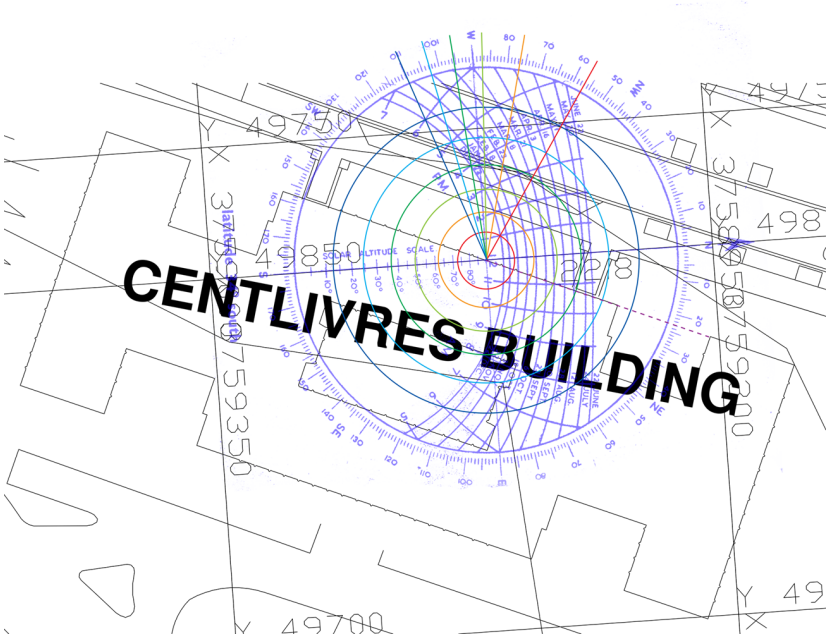


Figure 34: Sample building with solar protractor superimposed. (Author)

The measurements considered in this example are those that occur on 22 December in the Solar Protractor as the solar penetration is at its most extreme at that time. Concentrating on the West façade, the solar protractor is positioned with the centre on the wall surface where the window would be and aligned to North indicating that the sun strikes the façade at approximately 11.40am with the solar altitude angle at

approximately 79 degrees. This rises to 90 degrees overhead at midday while being at 22.8 degrees relative to the façade due to the building being oriented by this angle relative to North on site. By 1pm the sun has dropped to 73 degrees in altitude at 57 degrees past north (79.8 degrees relative to the façade) - marked in red. By 2pm the sun is lower at 62 degrees altitude and 76.5 degrees past north (89.3 degrees relative to the façade - almost at a right angle) - marked in orange. By 3pm the altitude is 49 degrees at 87.5 degrees West (110.3 degrees to the façade) - marked in lime green and by 4pm the altitude is at 37 degrees and 96.5 degrees West (118.3 degrees to the façade) - marked in green. The higher storied building blocks the sun from 30 degrees - so by 5pm the sun, at an altitude of 24.5 degrees, is behind the building and does not penetrate the sample building. These readings are illustrated in Table 1.

Time:	Altitude:	Angle (to façade):	Angle (compass):
11.40 am	79°	0°	22.8° North East
12 pm	90°	22.8°	0° North
01 pm	73°	79.8°	57° North West
02 pm	62°	89.3°	76.5° West North West
03 pm	49°	110.3°	87.5° West
04 pm	37°	118.3°	95.5° West
05 pm	24.5°	125.8°	103° West South West

Table 1: Table of values extracted from the Solar Protractor: West façade

Considering the data obtained from the Solar Protractor and the orientation of the sample building on the site, we can conclude: the occupants of this building, with offices along the West façade, will experience the sudden appearance of bright, sharp sunlight entering their office at a high angle during the late morning. This light will recede slightly as the sun moves closer to the vertical with window sills being illuminated and then the sunlight will start entering deeper in to the office once again. Over time the sun will move rapidly across the window opening with more direct light entering the office space as the sun moves down towards the horizon. This will cause glare to the occupants as the sun rays become more horizontal and the sun becomes perpendicular to the façade - making these offices difficult to work in (due to glare and heat build-up). Suddenly the office will be cooled and the glare removed as the sunlight is blocked by the neighbouring building.

4.2.3 Digital Solar Protractor

'Traditional' solar protractors have been replaced by 'digital' solar protractors. It is no longer necessary to print out and superimpose drawn protractors upon floor plans, as noted in the example above.

Solar and weather data has been gathered over time and centralised for downloading by the United States Department of Energy. African solar and weather data is available and divided by region with ZAF representing South Africa. Within this region two main geographical locations are represented, namely; Cape Town and Johannesburg. The data sources are available as an IWEC file type and can be accessed at this link: `https://energyplus.net/weather-region/africa_wmo_region_1/ZAF` Solar and weather data for Amsterdam is also available at this link: `https://energyplus.net/weather-location/europe_wmo_region_6/NLD/NLD_Amsterdam.062400_IWEC`

Various software packages can make use of this data source (it is most commonly used to drive simulations) but, within Architecture, a three-dimensional modelling package (Rhino3D) with a visual programming environment (Grasshopper) is typically used.

The programming environment loads a module (or plug-in) that can deal with the appropriate data provided. In this case study Ladybug is used within Grasshopper to access the IWEC solar/weather file.

Using 'nodes' within Ladybug, it is possible to extract information from the file to obtain a 'Sun Azimuth' and a 'Sun Altitude' - on the solar protractor this equates to the compass direction and altitude, previously noted as A1 and A2. This is shown in Figure 35.

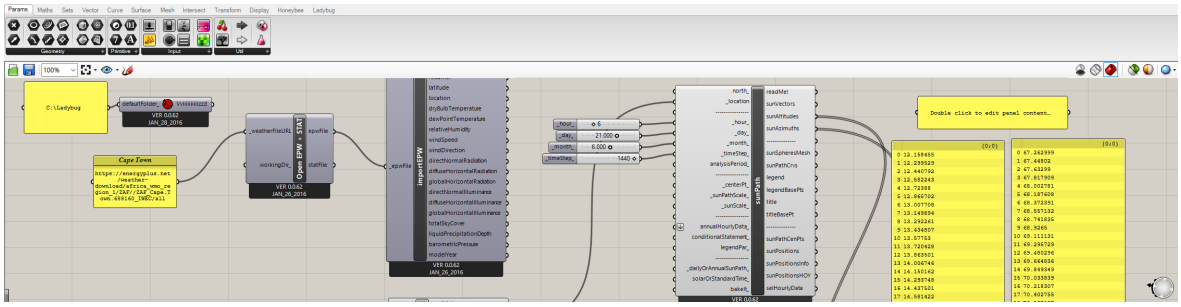


Figure 35: A Digital Solar Protractor - Grasshopper with LadyBug. (Author)

This figure demonstrates how the IWEC file series is imported on the left and 'piped out' as an EPW file. This is then fed into an importer to set the environment. A 'sunpath' element is added next and has the location (Amsterdam, off screen) fed to the location attribute. Four numerical sliders are added to represent the hour, month, day and time, they feed into appropriate attributes. From the Sun Path operator the Sun Altitude and Sun Azimuth can be extracted, visible in the yellow text output boxes shown on the right.

This method was used to extract solar data for Cape Town and Amsterdam. A value for time was set to retrieve this information - altitude and azimuth - across a selected day for every minute, this being the smallest value for extraction. This was then converted to fifteen second intervals. Following this process it was possible to determine the sun's position at fifteen second intervals across the longest summer day in each location. This information was stored in a CSV file for reading by a python script. **Appendix 3** contains the initial values extracted for Cape Town and Amsterdam from sunrise. The values are considerable in length so only the first few readings are charted - from 4am until 6.37am for Cape Town and from 3am until 5.27am for Amsterdam.

4.2.4 The window opening and 'bounding box'

In setting up the parameters within which the point cloud could be generated, the sample building on the Upper Campus of the University of Cape Town (as discussed previously in 4.2.2) was referenced. A proposed single window opening was considered where the measurements were taken in millimetres. A rectangle is imagined as the office space with the window in the outer surface (apart from considering the internal sill height for viewing, internal office details are not considered as this case study only focuses upon external work with the window surface referenced). As two conditions are studied, this office with window surface may be rotated such that the window surface faces the appropriate direction. The width of the window was set to be 1 300 millimetres, the height set to 1 700 millimetres and the depth - representing the wall thickness, was set to 200 millimetres.

External to the window is a slightly larger rectangular volume to contain the point cloud - the 'bounding box'. This is set to be 10% wider than the window area on either side with the height extending above and below the window area by 20% and the depth set to project away from the wall surface by two thirds of the bounding box height. The bounding box values are set relative to the window area to allow for the point cloud volume to be constrained to the window dimensions (should the window surface be modified, the point cloud bounding box is updated accordingly). The dimensions surround the window surface symmetrically so as not to affect form generation by building in an 'offset' of some sort to one edge.

'Standing within the office' and 'looking out the window' would require an unobstructed view. For the purpose of having a viewable lower area to the window, a measurement of one third the window height was considered (so the base third of the window looking through the point cloud was not to be obstructed by points as the viewing area would be blocked). Similarly, some sun-shade types require running the vertical height of a window (for example, a vertical fin). To allow for this potential form to be accommodated, each vertical edge within the viewing area permits points to 'encroach' by a width of one fifth of the total window width (this dimension space is further extended outwards by the dimensions of the bounding box offset). In an architectural sense, to simplify point generation and better visualise the values of the point cloud, zero is located at the bottom left of the 'bounding box' and is considered flush with the wall surface, making all values positive.

4.2.5 Calculating sun rays

The fitness of an individual (point within the point cloud) is based on the total number of sun rays the point can intercept and block from striking the window surface with the higher value being most fit. The Altitude and Azimuth readings derived from the digital solar protractor are used as input values to determine the vector of the sun ray. This vector is projected from the point to test if the sun would strike the window surface, and as such, would be blocked by the point (i.e. project a shadow).

To determine this vector, the origin point of this Cartesian space remains the same as that of the bounding box. In the Cartesian co-ordinate system, X has been used to indicate the values running horizontally along this wall surface (looking perpendicularly at the wall and increasing from left to right), Y is the height going up the wall while Z is

the value projecting directly away from the wall surface towards the viewer ('outside' as a positive value). Two calculations are performed to determine the shadow projection; first a horizontal calculation, thereafter a vertical calculation from this horizontal location for the final position. Figure 36 considers this first calculation, where the XZ co-ordinate values (or plane) represents an architectural 'top' view. The solar protractor value of 'Horizontal Angle' (Azimuth) is represented by angle H.

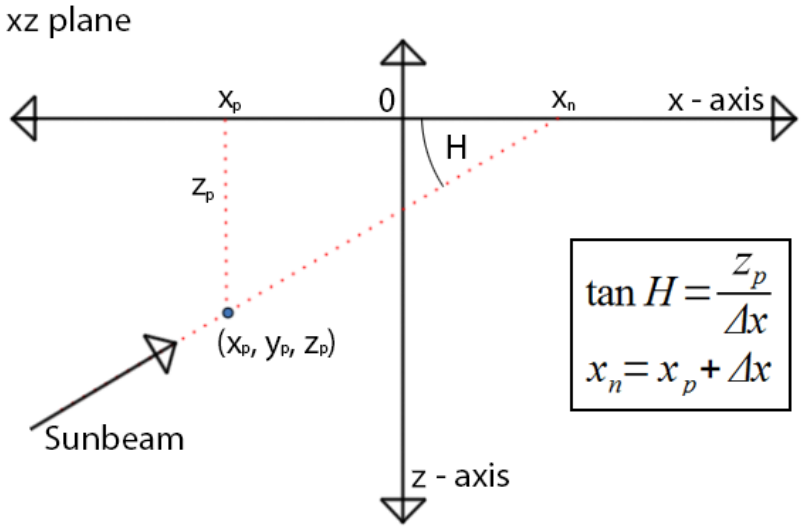


Figure 36: Sunbeam triangulation calculation - XZ (Author)

The next figure, Figure 37, considers the vertical position following from the previous horizontal position where YZ co-ordinate values (or plane) represents an architectural 'side' view. The solar protractor value of 'Altitude' is represented by angle V (vertical).

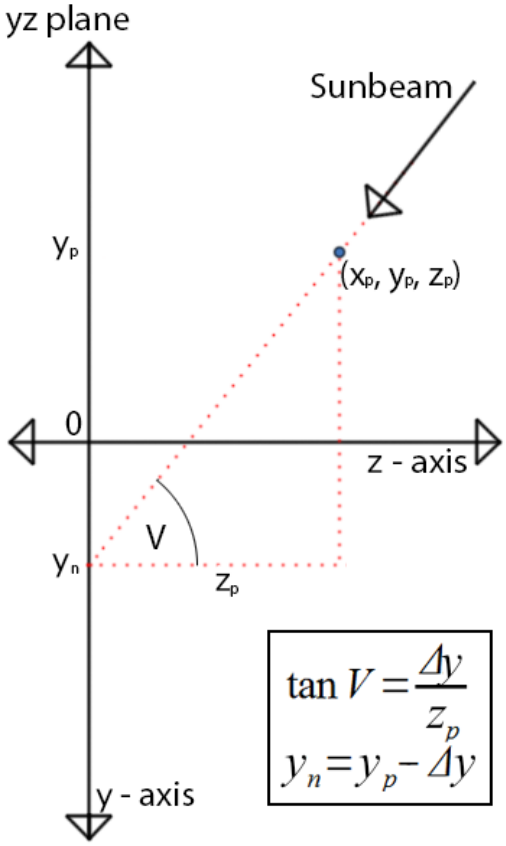


Figure 37: Sunbeam triangulation calculation - YZ (Author)

Calculating the normal
Given values:
H – the angle to the façade
V – the altitude
r – the distance from the generated point to the centre of the Cartesian space
Regarding H - since the solar angle is provided in Cartesian space together with North representing an angle of zero degrees, it is imperative to normalise this angle based on the direction the façade is facing. Figures 38 – 41 demonstrate the basic calculation to be performed based both on this facing direction as well as the hemisphere in which the solar angle is observed.

The action to 'normalise' the solar angle is to provide a mathematical basis where this angle has been cast into the same Cartesian plane. This allows the same fitness test to be performed regardless of the building façades' facing.

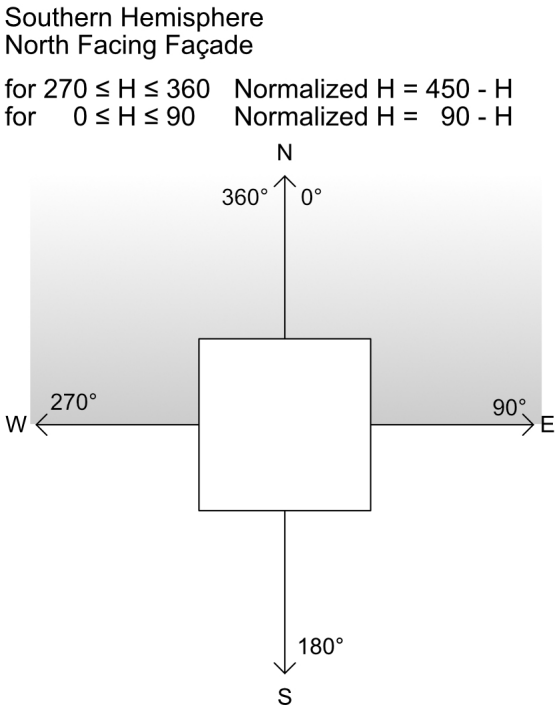


Figure 38: Casting the solar angle from a Southern Hemisphere projection on a North facing façade into a range between 0° and 180° (Author)

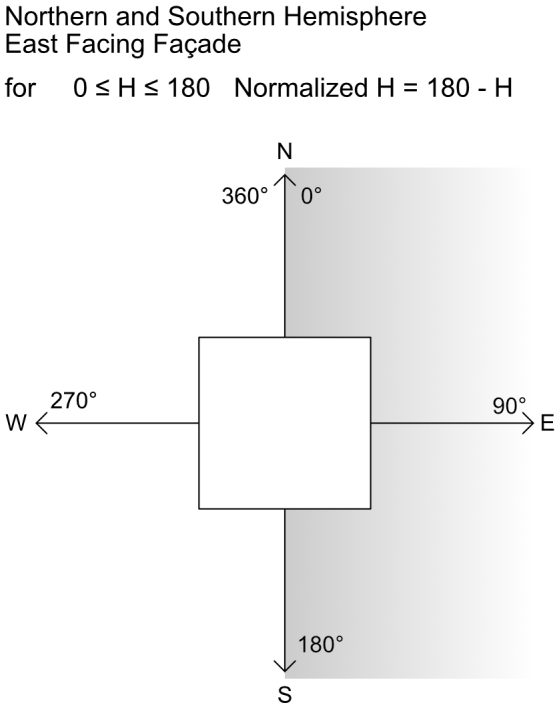


Figure 39: Casting the solar angle from a Northern or Southern Hemisphere projection on an East facing façade into a range between 0° and 180° (Author)

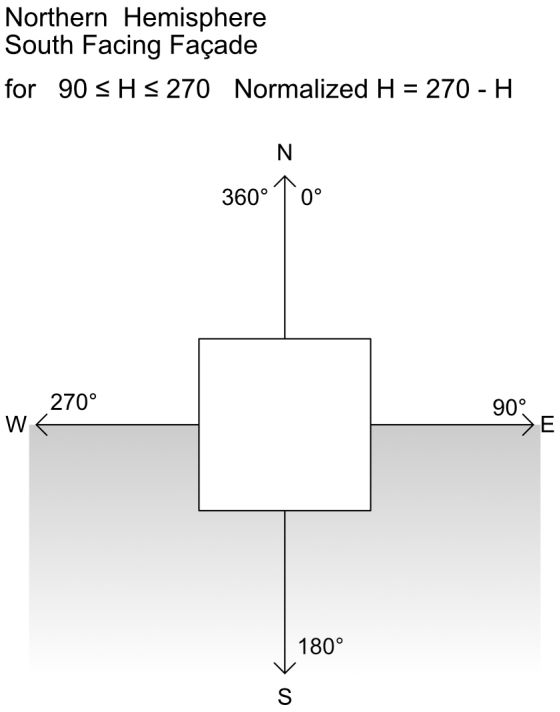


Figure 40: Casting the solar angle from a Northern Hemisphere projection on a South facing façade into a range between 0° and 180° (Author)

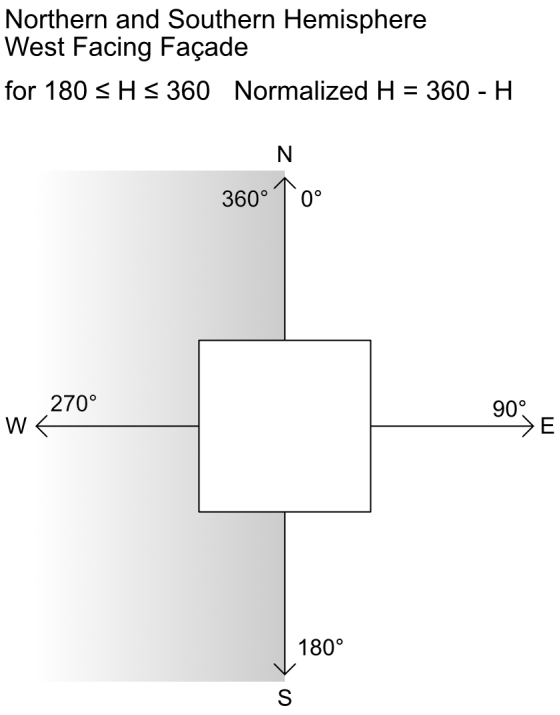


Figure 41: Casting the solar angle from a Northern or Southern Hemisphere projection on a West facing façade into a range between 0° and 180° (Author)

Figure 36 calculates the projected shadow of the generated point in space, where a successfully blocked ray of light would cast a shadow upon the window and therefore fall within the 'fit' dimensions within the point cloud. To calculate the difference in X to determine point X_n by using known values:

X_p – a straight projection of the generated point X_{pr}
Z_p – the distance the generated point Z_{pr} is away from the façade
H – the normalised solar angle as calculated in Figures 38 – 41, above

We can establish the equation of

$$\tan H = \frac{Z_p}{\Delta X} \text{ where } \Delta X = X_n - X_p \tag{5}$$

It is therefore given that

$$X_n = \frac{Z_p}{\tan H} + X_p \tag{6}$$

Figure 37 calculates the height of the projected shadow using a similar method. Using the known variables to calculate the projected point:

Y_p – a straight projection of the generated point Y_{pr}
Z_p – the distance the generated point Z_{pr} is away from the façade
V – the solar altitude

We can establish the equation of

$$\tan V = \frac{\Delta Y}{Z_p} \text{ where } \Delta Y = Y_p - Y_n \tag{7}$$

It is therefore given that

$$Y_n = Y_p - \frac{\tan H}{Z_p} \tag{8}$$

The projected point Z will always be 0, as the Cartesian plane of projection is located on the window where the shadow is cast.

It is now possible to consider the Evolutionary Algorithm that could be used to evolved the point cloud (generated using 'architectural' values).

4.3 Evolutionary Algorithms

Evolutionary Algorithms, when employed in the Architectural profession for research, tend to see Genetic Algorithms dominate [38]. It is understood this is connected to how they represent Evolutionary processes, making it easier for Architectural Professionals to understand in order to better adapt these processes to fit their building research criteria [25]. Similarly, Evolutionary Programming and Evolution Strategies see less use in this profession, this could be connected to their more complex methods (for example, the use of self-adaptation). Some researchers have taken a specialised approach, dedicating their work to explore the potential of certain specific algorithms (for example, Coates' focus on Genetic Programming [17]).

4.3.1 General Outline

This case study looks at using **Evolution Strategies** to optimise Cartesian real-value parameters that make up points within a point cloud with the final optimised result producing a sun-shade that blocks the most direct sunlight from striking upon a window surface. The sunlight values are taken on the Summer Solstice as this is the extreme case for optimisation. Furthermore, two conditions (East and West) are optimised in two locations, Cape Town and Amsterdam.

Following De Jong [23], a single, fixed-size population within a fixed environment and using fixed methods for reproduction and mutation is tested. Each individual (a point) is a fixed-length vector in the form [fitness, X co-ordinate (within the point cloud), Y co-ordinate (within the point cloud), Z co-ordinate (within the point cloud), self-adaptive value (for X), self-adaptive value (for Y), self-adaptive value (for Z), individual ID, parent ID, generation ID]. **Appendix 4** provides two sample tables indicating the details of numerous individuals.

4.3.2 Why Evolution Strategies for the algorithm?

Thomas Bäck, in his Introduction to the Handbook of Evolutionary Computation [6] , identifies the following notable characteristics for each Evolutionary Algorithm:

Genetic Algorithms place the emphasis on recombination as the search operator with mutation as a background operator and typically use proportional selection and binary representation for individuals.

Evolution Strategies use normally distributed mutations to modify real-value vectors with an emphasis on mutation and recombination as essential operators for searching through the search space and strategy parameter space simultaneously. The selection operator is deterministic with parent and offspring sizes usually differing from each other. Regarding representation, Evolutionary Strategies **need no encoding step** - the genotype space is the same as the phenotype space $\mathbb{R}^n$.

Evolutionary Programming emphasises mutation but does not incorporate recombination. When approaching real-valued optimisation problems, normally distributed mutations are used and the evolutionary process is extended to strategy parameters (this being similar to Evolution Strategies). The selection operator is probabilistic with most applications involving real-valued vectors (even though the algorithm was originally intended to evolve finite-state machines).

Like Bäck, Eiben and Smith [26] also note that Evolutionary Strategies are generally useful for parameter optimisation where **representation of object parameters is obvious** - in other words, no encoding step is needed between the genotype and phenotype.

Eiben and Smith continue to indicate that the Evolutionary Strategies used by Rechenberg and Schwefel started out as **a shape optimisation algorithm**. Initially used to verify a flat plate as an expected global optimum and then being used to design a bent pipe with the result being unexpected, but better, when compared to other solutions [24].

This case study also centres around **shape optimisation** to determine the best sun shade design for a given scenario and confirms an important aspect behind this study, namely; to potentially produce an unexpected but better result when compared

to what is currently used in the field. In terms of shape optimisation, this case study essentially undertakes the same task, shape exploration.

Eiben and Schoenauer [27] confirm Evolutionary Strategies were implemented in experiments as they have a strong application to real-valued parameter optimisation (making use of Gaussian mutation).

In summary, the ideal Evolutionary Algorithm for this case study was to have understandable representation of object parameters (so no 'encoding'); it would be best to have a legacy of shape optimisation (seen as being similar to shape exploration) where unexpected results could be produced; it was expected to be used on continuous parameter optimisation (points saturated with solar values) and it was hoped that the algorithm would have strong architectural form-finding potential - from this the requirement would be that offspring mutation should be significant (possibly an evolutionary algorithm where this is emphasised) and over time it may be best to have these mutation values change. Following Eiben and Smith, Evolution Strategies is clearly the best suited algorithm for application in this case study being strong in all aspects noted above.

4.3.3 Fitness

Fitness for each individual is represented as an integer value recording the amount of sun rays that the individual blocks from striking a window surface where the higher value indicates higher fitness. As noted in 4.2.5, a sun ray is projected every 15 seconds from the individual to test if this strikes the window surface. It should be noted that it is not expected that every sun ray will be blocked as they vary significantly in their projection - from the horizontal to vertical. Furthermore, it is not possible to achieve 100% fitness given the sun's path through the point cloud placement (consider that the sun rays are almost parallel to the façade at times).The values that make up the sun rays were extracted using a digital solar protractor (as outlined in 4.2.3) with the total amount of sun rays for the Summer Solstice collected into CSV text files, one for each location, and read line by line to compute if the sun ray at that time is blocked from striking the window surface, see **Appendix 3**. Under certain conditions a point in 3D space was simply assigned a fitness value of zero, being totally unfit. This was done where the point was located outside the point cloud bounding box or if it was located within the 'viewing area'.

4.3.4 Building façade

The the sample building, discussed in 4.2.2, was based on a building on the Upper Campus of the University of Cape Town and was abstracted to provide the façade. This building is considered singled storied, rectangular and cardinally aligned with longer façades to the East and West. No terrain features (mountain slopes or nearby buildings) are considered as this allows for all solar values to be assessed while also permitting for better value correlation between the two geographical regions. This building was 'placed' in both Cape Town and Amsterdam and the Summer Solstice in both locations provide the solar values used for fitness. Summer Solstice is considered to be 22 December for Cape Town and 21 June for Amsterdam. The window opening

of the original building was maintained (see 4.2.4) and flat wall surfaces are used (no architectural decorations).

4.3.5 Bounding box - point cloud

As the first generation of the population forms a point cloud, this is contained and then evolved within the bounding box that will also ultimately contain the evolved sun-shades (discussed in 4.2.4). Consideration for the volume of the bounding box was the need to be big enough to not limit potential forms resulting in unexpected sun-shade solutions. Similarly it can accommodate vertical 'fin' forms while the depth could accommodate various traditional sun-shade types ('eggcrates') and still be structurally sound.

4.3.6 Population

The point cloud within the bounding box forms the total population. Consequently, the population size was determined by considering the mesh density that could be created. It was hoped that a higher value - a larger population - would result in a more detailed (or 'nuanced') mesh; this is important considering the form-finding potential of the geometry. A population of 20 000 individuals was considered acceptable for creating a detailed mesh that, in turn, would be a visualisation of a sun-shade. To produce enough data to evaluate the effectiveness of the Evolutionary Strategy, 100 generations were considered adequate. The parent population is initialised randomly within the bounding box. As noted in Table 3 and Table 4, each individual is assigned an object parameter, a Cartesian co-ordinate value of X, Y and Z, together with an evolvable strategy parameter for each co-ordinate. This initial generation is ranked by fitness and the top half (50%) provides a pool of potential candidates that can be selected as parents. Of this group, 2 000 individuals are randomly selected with each producing 10 offspring. The offspring can receive negative evolvable strategy parameters. Selected parents and offspring combine to form the next generation and are tested for and ranked by fitness. Using elitist selection, the top half are once again made available as potential parents in a selection pool with 2 000 randomly selected individuals producing the next offspring (this repeats for 100 generations). Overall, this is done for two façade conditions, East and West and in two geographical locations, Cape Town and Amsterdam. 10 'runs' of each generation is performed per façade condition in each location to gather enough data to evaluate the Evolutionary Strategy.

4.3.7 Recombination

Recombination is additive as the parent object parameters are combined with the child's evolvable strategy parameter, this being subject to adaptive step mutation.

4.3.8 Mutation

Initially the object parameters were mutated using a normal distribution (with an expected value of zero) through a random Gaussian vector, however the mutations

were too strong. Following this, self-adaptive mutation was implemented following the guidelines as discussed by Beyer and Schwefel [9].

4.3.9 Summary of ES

Representation: Real-value vectors
Recombination: intermediary
Mutation: Self-adaptive, logarithmic normal
Parent Selection: Uniform Random
Survivor Selection: $(\mu + \lambda)$

4.4 Pseudocode - Evolution Strategies

The following Pseudocode outlines what the Evolutionary Strategy is required to do (this follows from Beyer [8]).

$(\mu + \lambda) - ES$

1) $g = 0$
2) while $g < 100$;
   a) if $g == 0$;
      • initialise $\left(\mathfrak{P}\binom{(0)}{P} := \{\mathfrak{a}_1, ...\mathfrak{a}_{20000}\}\right)$
      • test fitness
        - if within view zone;
          * fitness = 0
        - else;
          * read sun text data
          * calculate sunray position on window surface
            ' if on glass, fitness = fitness +1
            ' else;
      • sort by fitness
      • save ALL numpy array
      • g + 1
   b) else;
      • read in ALL numpy array
      • slice top 50% from ALL numpy array
        - save as allParents numpy array
      • randomly select 2 000 individuals from allParents numpy array
        - save as Parents numpy array
      • duplicate Parents numpy array
        - save as thisGen numpy array
      • for each vector in Parents numpy array
        - extract object parameters
        - set counter = 0
          * while counter ¡ 10;
            ' extract evolvable strategy parameters (randomly set as neg. value)
            ' calculate child evolvable strategy parameters
            ' set evolvable child strategy parameters
            ' mutate child object parameters
            ' (test fitness) : if outside bounding box, set fitness = 0; else
            ' (test fitness) : if within viewing zone, set fitness = 0; else
            ' read sun text file
            ' calculate sunray position on window surface
            ' if on glass, fitness = fitness + 1;
            ' append child to thisGen numpy array
        - counter + 1
      • save as final thisGen numpy array

- sort thisGen numpy array
- save thisGen numpy array to ALL numpy array
- g + 1

At (1) a generation counter is initialised and set to zero. (a) while the counter is zero, generate parents. Each vector contains a fitness value, X, Y and Z Cartesian co-ordinate values, X, Y and Z self adaptive values and three tracking/identity values - individual ID, parent ID and generation ID (this is used to track who gave birth to the individual, how they may vary and for troubleshooting and logging). Upon creation each parent is tested for fitness and assigned a fitness value. They are all contained in a Numpy array and sorted by fitness value. (b) focuses on subsequent generations. The top half of the parents are selected to ensure a healthy variety of candidates. Of this group, two thousand (10% of the total population) are randomly selected to produce ten offspring. This group is duplicated as they form the next generation together with their offspring. Each offspring has their object parameters and mutation adaptive step calculated (these are inherited through generations). Each is tested for fitness by reading in solar data from a text file and checking to see if the sun ray is blocked from striking a window surface. Certain cases are tested first where zero fitness is applied, these are; outside the bounding box and within the 'view parameters' (the view out the window must not be obscured). The children are appended to the numpy array. This full generation - parents and children - is sorted for fitness and saved to be used for the next generation cycle. The generation counter is increased by one. After one hundred generations the Evolutionary Strategy stops.

4.5 Point cloud to mesh

Various text files are logged while the Evolutionary Algorithm runs over the evolutionary cycle of 100 generations. One text file captures every individual found in each generation, another captures each individual making up every tenth generation and a third captures the individuals within the initial and final generations. This allows the evolutionary process to be 'read'. The text files are 'cleaned' and 'formatted' and exported as CSV files using a Python script. The CSV file is further processed to extract Cartesian co-ordinate values, X Y and Z, for an entire generation. These values, converted to a TXT file, can be read into **MeshLab**. Should the first generation of Cartesian co-ordinate values be displayed in MeshLab a point cloud filling the bounding box will be shown (see Figure 1 in a separate paper; second right image in the earlier work [18]). The final generation Cartesian co-ordinate values form a point cloud representing the final, evolved sun-shade (discussed further in the next section **Experiments and Results**).

To convert these point clouds to surfaces within MeshLab, they need to be sampled - Poisson Disk Sampling (with 'Base Mesh Subsampling' activated) was used. Thereafter their normals needed to be set using 'Compute Normals for Point Set' (with 'flip normals' deactivated). Finally the mesh is generated using 'surface reconstruction' - Screened Poisson Surface Reconstruction. Making it easier to view a complicated, highly detailed mesh within MeshLab, certain visual cues are best set. 'Show Axis' is activated to orient the view to suit the building surface. 'Enable Shadow Mapping' and 'Enable Screen

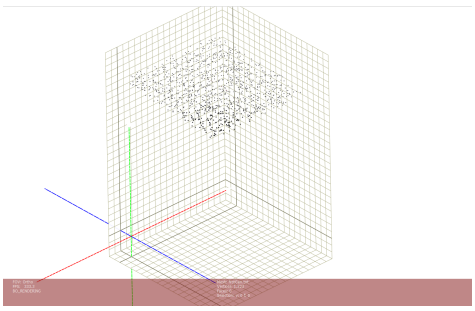
Space Ambient Occlusion' allows the 3D object to stand out from the environment. Setting the material to 'glass' while deactivating 'faces' allows the sun-shade to be visualised three-dimensionally with less distractions making for a clearer evaluation.

4.6 Traditional Sun-Shade 'fitness'

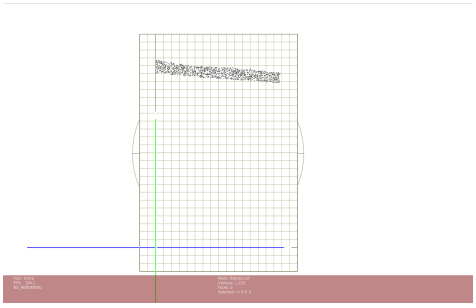
As this case study seeks to determine if an evolved sun-shade can outperform a traditional solution typically used in Architecture; a method was required to determine the 'fitness' of a traditional sun-shade such that it could be compatible with the Evolutionary Strategy and allow for comparison. To accomplish this, existing initial random point clouds were selected as a solution space for each traditional sun-shade. These initial point clouds contain 20 000 random individuals with existing fitness values, this having already been determined by the number of sun rays each point had blocked. Again, using point cloud editing software, MeshLab, the points that make up a traditional sun-screen can be isolated and extracted.

There are 10 sample 'traditional' sun screens under consideration; for each, an Evolution Strategies' initial population point cloud was chosen randomly. Thereafter, points within this point cloud that do not make up the geometry of the sun screen are deleted. This 3D point cloud is then saved as an XYZ file (a text file). Occasionally, the point cloud may require several edits, each being saved as an XYZ file. These files are then combined in a text editor where duplicates are removed, resulting in a comprehensive 3D point cloud as a text file. This is then re-opened in MeshLab where the software 'cleans and repairs' the point cloud by removing duplicate vertices. Thereafter the point cloud is sampled (using 'point cloud simplification' and Poisson disk sampling). The final result is saved as an XZY (text) file.

Figure 42 shows two views of the point cloud that make up the 'Sloped Overhang' traditional sun-shade within Meshlab while Figure 43 illustrates the traditional 'Horizontal Overhang' sun-shade point cloud.



(a) Traditional sun-shade - sloped overhang in MeshLab (Author)



(b) Traditional sun-shade - sloped overhang in MeshLab side view (Author)

Figure 42: Traditional sun-shade 'Sloped Overhang' - 3D point cloud in MeshLab

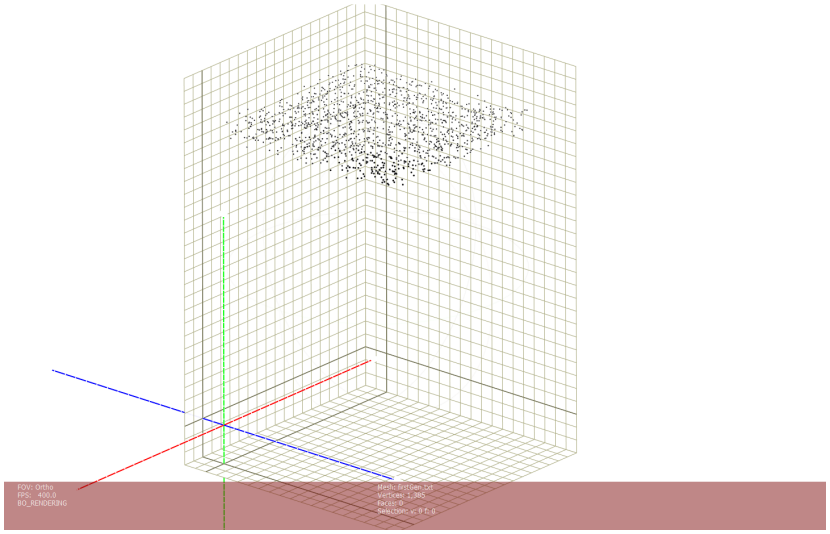


Figure 43: Traditional sun-shade 'Horizontal Overhang' - 3D point cloud in MeshLab (Author)

Using the extracted points that make up the sun-shade, it is possible to determine a fitness value for each traditional sun-shade. **Experiments and Results** outlines which initial random point cloud was assigned to each traditional sun-shade and lists the fitness values produced for each.

5 Experiments and Results

5.1 University of Cape Town - Data Management

The results produced in this case study follow the University of Cape Town's approach to Data Management and curation. The details can be found at this link: `https://www.uct.ac.za/research-support-hub/research-data/managing-research-data`. At the time of documenting this case study, the raw data produced by the Python scripts as Numpy arrays together with traditional sun-shade fitness results is made available for researchers. The data will be available, as the University of Cape Town determines, via ZivaHub using this link `https://zivahub.uct.ac.za/`.

5.2 Traditional Sun-shades Fitness results

In their 1997 work, O'Conner et al. [47] discuss traditional sun-shades for use in architectural design. Their illustrations of these sun-shades are combined and presented in Figure 44. The following sun-shades can be noted: (top left to right) Horizontal Overhang, Horizontal Overhang 3 louvres, Horizontal Overhang dropped edge, Horizontal 15 sloped, Horizontal Overhang sloped and (bottom left to right) Horizontal Overhang louvred edges, Horizontal Overhang of louvres, Vertical louvre-fin 4 angled and Horizontal 19 horizontal. A tenth sun-shade is added - the Vertical louvre-fin (this is simply a vertical fin located to the sun side of the window opening).

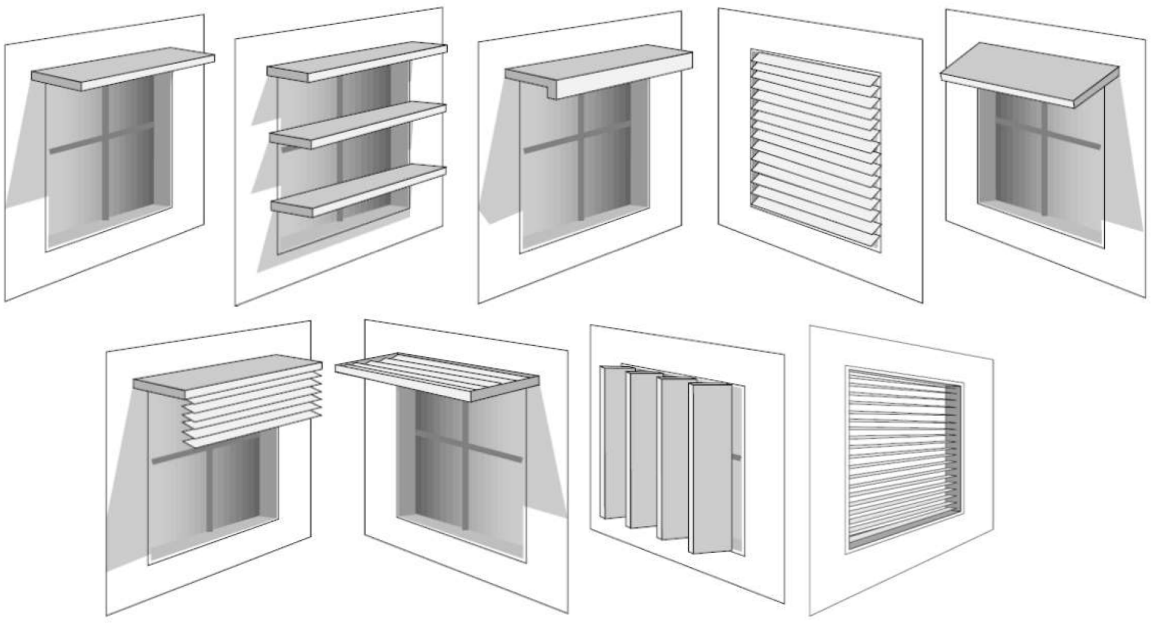


Figure 44: Traditional window sun-shades - combined image (O'Conner et al.)

As noted previously, Figure 43 illustrates the traditional 'Horizontal Overhang' sun-shade, as show above, placed within a point cloud (using the dimensions of the point cloud bounding box).

Each traditional sun-shade was randomly placed within the initial point clouds generated during an Evolutionary Strategy run before any evolutionary changes were effected but after having fitness per point calculated. The points that make up the points clouds are randomly generated and vary between each run.

The following initial point clouds were used to sample points for the fitness values of each traditional sun-shade:

1. 'Horizontal Overhang' - sampled in the Amsterdam East, third run
2. 'Horizontal Overhang dropped edge' - sampled in the Cape Town West, fifth run
3. 'Horizontal Overhang louvred edges' - sampled in the Cape Town East, first run
4. 'Horizontal Overhang 3 louvres' - sampled in the Amsterdam East, eighth run
5. 'Horizontal 19 horizontal' - sampled in the Amsterdam West, seventh run
6. 'Horizontal Overhang of louvres' - sampled in the Cape Town West, tenth run
7. 'Horizontal Overhang sloped' - sampled in the Amsterdam West, second run
8. 'Horizontal 15 sloped' - sampled in the Amsterdam East, tenth run
9. 'Vertical louvre-fin' - sampled in the Cape Town East, eighth run
10. 'Vertical louvre-fin 4 angled' - sampled in the Cape Town West, second run

The fitness value of the remaining points were calculated. This allowed for fitness comparison between traditional and evolved sun-shades. The results are illustrated in Table 2.

	'Traditional' sun-shade system	Normalised Fitness	Mean Fitness	Full Fitness
1	Horizontal Overhang	0,32	634,27	1984,00
2	Horizontal Overhang dropped edge	0,28	470,69	1711,00
3	Horizontal Overhang louvred edges	0,27	468,57	1710,00
4	Horizontal Overhang 3 louvres	0,22	442,12	1984,00
5	Horizontal 19 horizontal	0,03	57,76	1985,00
6	Horizontal Overhang of louvres	0,27	456,01	1711,00
7	Horizontal Overhang sloped	0,34	673,86	1985,00
8	Horizontal 15 sloped	0,08	149,99	1984,00
9	Vertical louvre-fin	0,25	432,53	1710,00
10	Vertical louvre-fin 4 angled	0,26	442,56	1711,00

Table 2: Traditional Sun-shades - fitness values.

'Full' fitness indicates the total amount of sun-rays that pass through the point cloud and strike the window surface and that can be blocked for the given scenario (East/West and Cape Town/Amsterdam). The total amount of sun-rays available for blocking. The 'mean' fitness value indicates the mean value of run rays that are blocked by the point

cloud population - this is the 'typical' best fitness value that can be expected under the circumstances. The 'Normalised' fitness value is a range between zero (no sun rays blocked) and one (all sun rays that pass through the point cloud and could be blocked).

It can be seen that the traditional 'Horizontal Overhang sloped' sun-shade has the highest normalised fitness value of 0,34 followed closely by the traditional 'Horizontal Overhang' with a normalised value of 0,32. (The overhang being a common solar shading device).

5.2.1 Traditional Sun-shades Fitness - comments

As Olgyay and Olgyay noted, and as is common practice in Architecture, sun-shades with horizontal forms are best for direct sun facing façades (South/North - as per hemisphere). Similarly, vertical sun-shades are expected to be prominent solutions where the sun rays are anticipated to be at a low angle to the façade, typically East/West and at sunrise/sunset. Furthermore, the vertical shades are situated to the side of the window where the sun is expected to reach the the highest point (the North side of a window opening in the Southern Hemisphere). As this case study is limited to East and West façades, it is expected that vertical fins would be the prominent solution. However, as Table 2 indicates, the Horizontal Overhang sloped, followed closely by the 'traditional' Horizontal Overhang (or roof eave) are found to be the most fit sun-shades of the Traditional types.

5.3 Evolved Sun-shades mesh results

Figure 45 shows an evolved sun-shade in context. The view is from the direction of the sun and the sample window opening is lightly indicated as a background. The view is composited from a MeshLab scene.

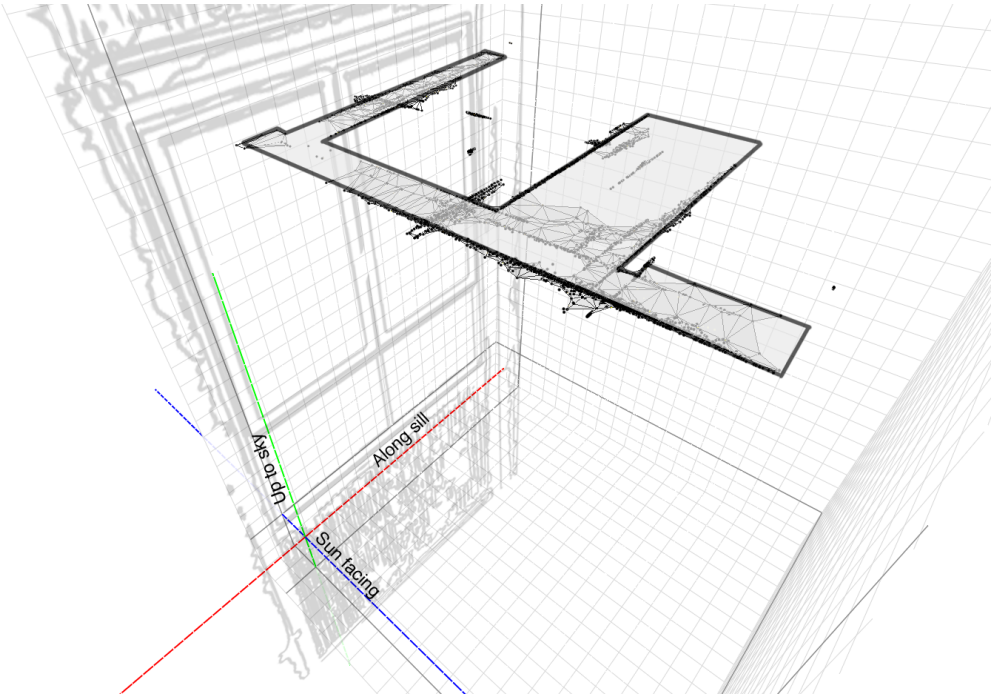


Figure 45: Evolved sun-shade in-situ. (Author)

MeshLab was used to surface point clouds in order to better visualise the sun-shades that were evolved. The Red (X) axis runs along the wall surface while the Green (Y) axis indicates 'up' and the Blue (Z) axis 'outwards'. The following images, below, indicate four typical evolved solutions.

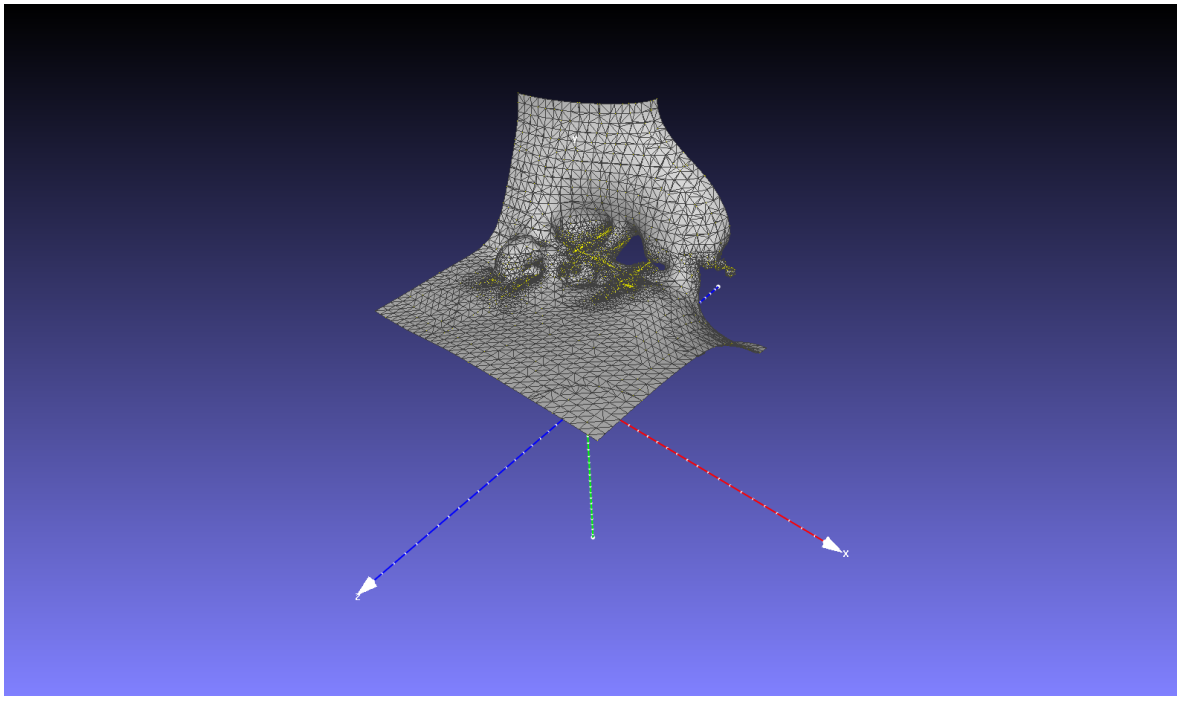


Figure 46: MeshLab generated sun-shade for the Cape Town East first run. (Author)

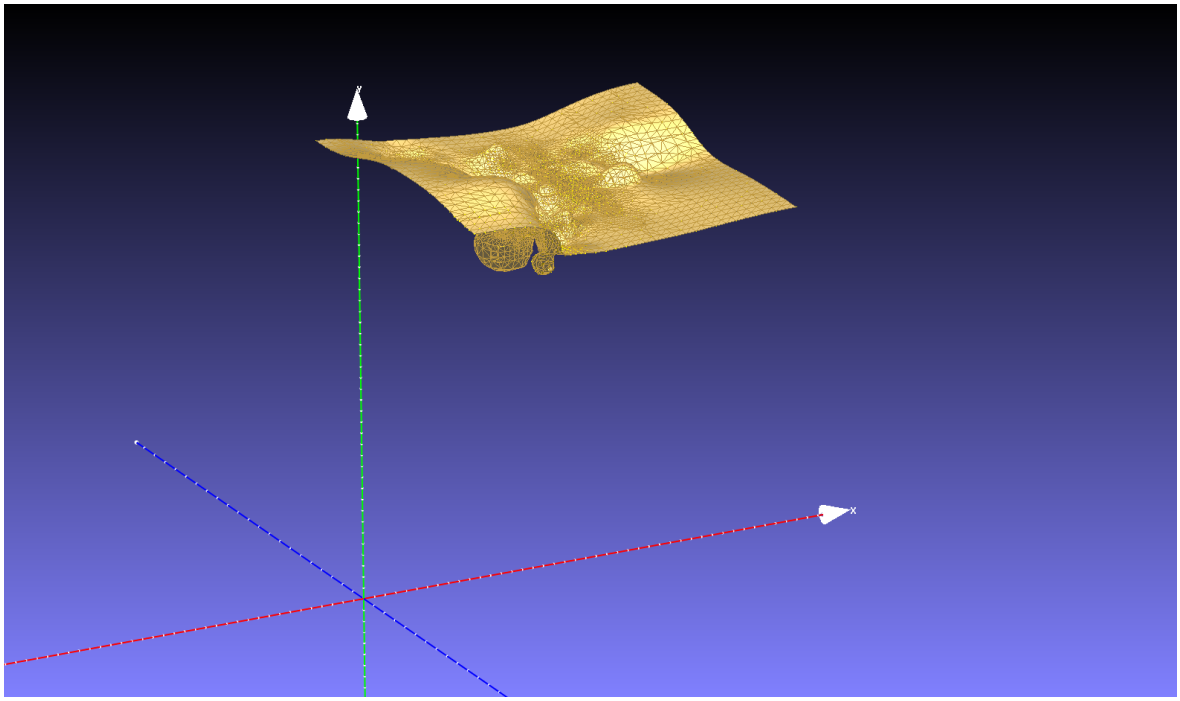


Figure 47: MeshLab generated sun-shade for the Cape Town East fourth run. (Author)

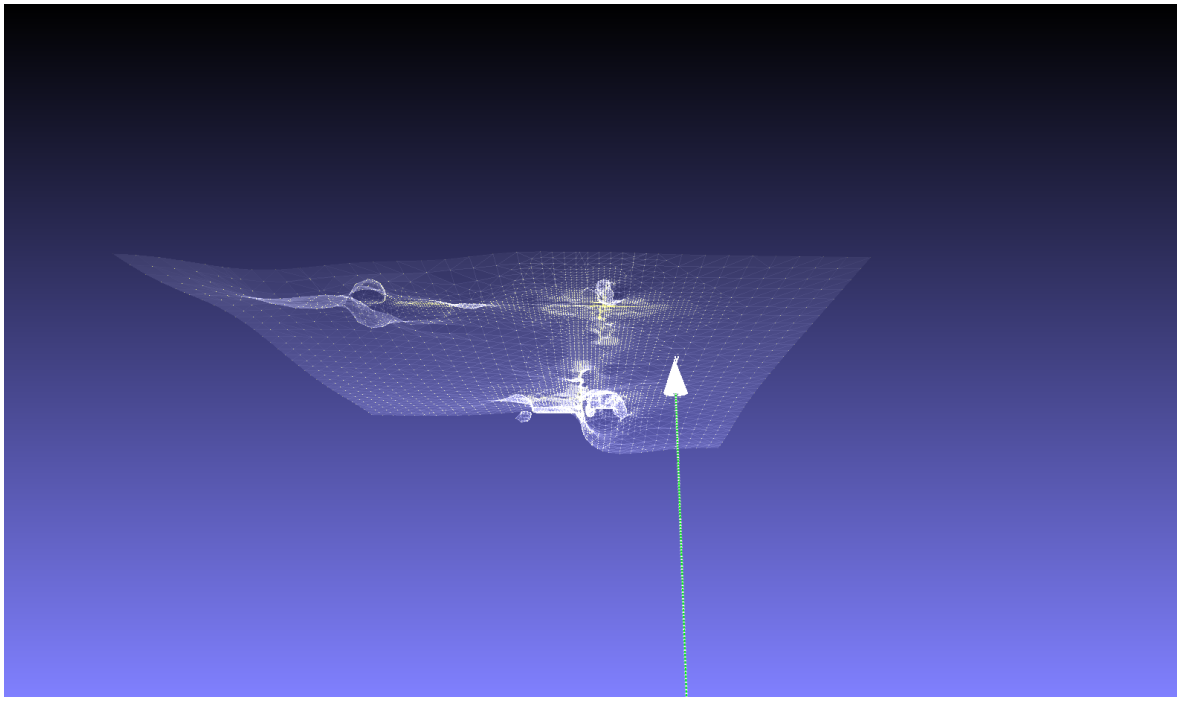


Figure 48: MeshLab generated sun-shade for the Amsterdam East second run. (Author)

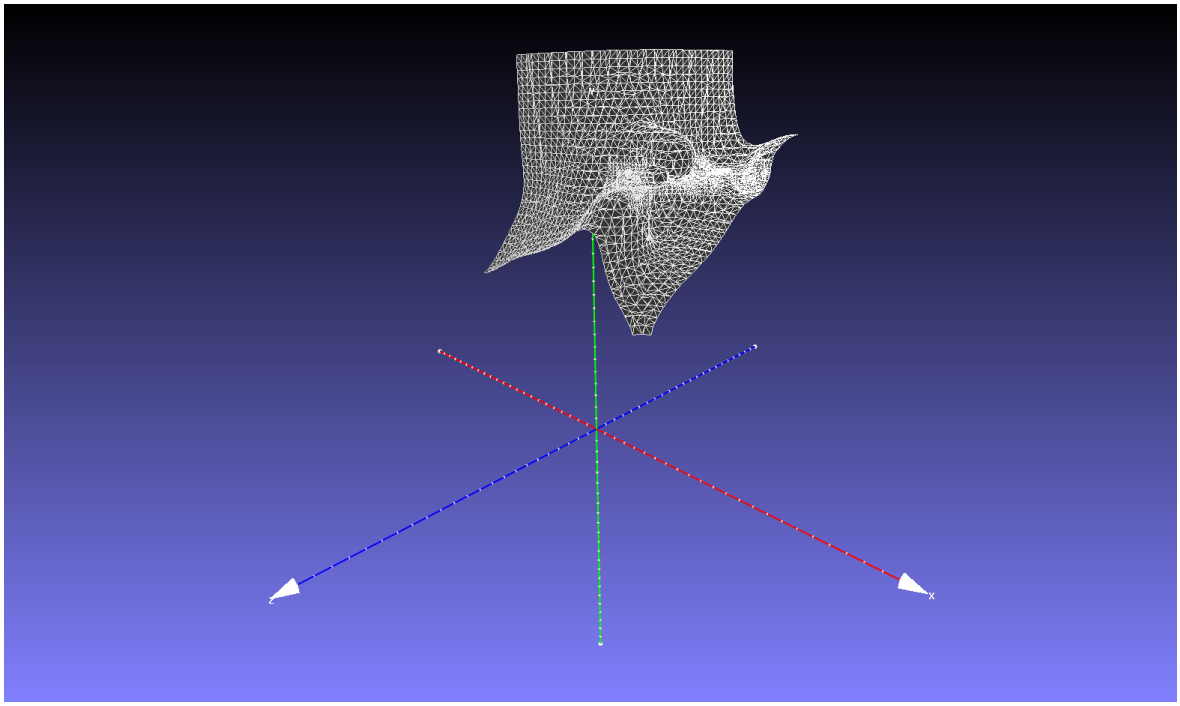


Figure 49: MeshLab generated sun-shade for the Amsterdam East third run. (Author)

5.3.1 Evolved Sun-shades meshes - comments

The evolved meshes do not neatly align with the traditional categories of vertical fins, horizontal overhangs or 'eggcrates'. Instead, more organic, 'shell like' shapes emerge. Surprisingly, most evolved sun-shades are strongly horizontal. Figure 45 is purely horizontal but contains a 'leading edge' towards the sunlight to possibly track sun rays as they swing North.

East façades: All sun-shades in this section need to deal with sunrise where sun rays typically enter the building horizontally and then rise in altitude to angle down through the window openings into the interior (as the angle increases the light penetration is decreased in the interior).

Figure 46 combines both vertical and horizontal elements with a lower 'overhang' (the horizontal form being lower) and a vertical shield to prevent sun rays entering. A bulged, rounded area is present on the vertical surface in the direction of the sun, then curves in and re-appears in the direction of the sun forming vertical fins that are rounded but connected to the whole. An organic 'eggcrate'.

Figure 47 uses a different display setting in MeshLab and indicates a high horizontal 'overhang' with a 'bulb' below and towards the sunlight to possibly block out the early rising sun.

Figure 48 displays the evolved sun-shade in an 'x-ray' view to better indicate the details of the mesh. This is a largely horizontal sun-shade but again has a lower bulge that appears cylindrical. The density in the mesh, shown as concentrations in white, indicate where points within the point cloud localised. This could be a visual indication of a local maxima within the evolved solution space. (The mesh contains more detail as more points are present in that position).

Figure 49 illustrates a vertical surface at a higher level but that this then extends horizontally and in the most outward facing area is 'snapped' to form an inverted 'V' where a vertical screen of sorts rises then falls across the window so as not to obstruct the view. There is a certain 'bias' towards one side of the 'vertical' screen.

5.4 Evolution Strategies - fitness results

As noted previously, the fitness for each individual is represented as an integer value and indicates the number of sun-rays the individual blocks from striking a window surface. In Cape Town, on December 22nd, this equates to 3 241 tests (where the sun rises at 5am and sets at 7.52pm). In Amsterdam, on June 21st, this equates to 3 969 tests (where the sun rises at 4am and sets at 8.59pm). It should be noted that it is not expected that every sun ray will be blocked as they vary significantly in their projection - from the horizontal to vertical. In Cape Town, of the 3 241 tests performed, only 1 710 were eligible (meaning they could strike the window surface through the point cloud) for the East façade and 1 711 for the West façade. In this case study, these values are labelled 'Full Fitness'. Similarly, in Amsterdam, of the 3 969 tests preformed, only 1 984 were eligible for the East façade and 1985 for the West façade. In addition, given the placement of the point cloud and the individual points within, relative to the window surface, it is not possible to achieve 100% fitness even within the 'Full Fitness' value.

There are three levels of detail to consider when focusing on the Evolution Strategies results. The first, available in **Appendix 4**, notes **individual fitness values** within a generation. This is a 'detailed' view of the Evolutionary Strategy at the individual level. Table 3 indicates a sample from the initial population for the Cape Town West façade 6th run while Table 4 indicates the same sample at the final generation. Both Tables show a small selection of the individuals rather than the whole population (which numbers several thousand, the raw results being available in the data repository).

Next, **Appendix 5** summarises **mean initial fitness values per generation**. This indicates a broader view of the fitness results and maps the mean value from the initial generation until full fitness for that location is achieved and tracks this across all the sample runs. Table 5 reflects the results for Amsterdam East.

Lastly, the final **mean fitness value achieved per Evolutionary Algorithm** run is examined (section 5.4.3). This is illustrated in both graphical and Tabular form (**Appendix 6**).

5.4.1 Individual Fitness values

The results that track the entire run (Appendix 4) indicate varied and healthy fitness among initial individuals. This first generation seems highly diverse - enough to indicate a large range of points scattered within the point cloud, as one would expect. However, over time certain parents and their offspring dominate - this is evident when tracking the parent/individual identities. At most, a parent is present for approximately five generations before being entirely replaced by the offspring. Final generations are made up of individuals produced within that generation with an occasional immediate parent among them. Looking at Table 4, offspring of parent 669 dominate the top fitness values

together with the offspring of parent 670. Additionally, there seems to be a convergence to a local maxima as points converge in the search space and small changes take place (reading this via object parameters of Cartesian values). Reflecting on these co-ordinate values, parents and offspring are located closely together while sets of parent/offspring groups, 669 and 670, are also located close to each other within the point cloud with only the Z value showing a difference.

This was also noticed in the initial test runs and modifications were made at that time, this Evolutionary Algorithm reflecting the updated changes (see 4.3.8).

This 'granular' view also provides an indication of the evolvable strategy parameters (here as self-adaptive values) used as the mutation strength. The first generation contains larger and generally positive values while the final generation reflects smaller increments with a combination of negative and positive values. Evolvable Strategy Parameters are further discussed in 4.3.7.

5.4.2 Fitness values per Generation

Appendix 5 displays the results across the first 10 generations (out of 100) and compares the performance of the algorithm over 10 runs (to ensure the results are consistent). The table indicates the mean fitness achieved per generation. The top of this table indicates mean and the normalised version across the 10 runs.

The fitness values are constant across the different runs - where each optimise a newly generated point cloud. Consistently, among the various runs, full fitness is achieved by the tenth generation with the normalised value effectively being achieved by the fifth generation.

Architecturally the first five generations are extremely valuable as this is where 'form-finding' seems to take place. Using MeshLab to visualise the point cloud across each of these early generations and then animating the images makes it possible to understand what appears to be 'self-organisation' as the points shift and the surface of the sun-screen appears. By the tenth generation (and increasing in ten generations), the animated sun-screen hardly changes. Small movements are noticed but the form is already set by the tenth generation.

This does, however, indicate how effective the Evolutionary Strategy is at optimising form, confirming the reason why this algorithm was selected initially.

5.4.3 Mean fitness per Evolution Strategies run compared to traditional sun-shade

The following figures graph the normalised fitness achieved per generation to provide an overall reading of how the Evolutionary Strategy performed in each scenario where the mean fitness is constant and reached by each run. The normalised value indicates total fitness that could be achieved ('full fitness'). The red bar running horizontally across the graph indicates the best fitness achieved by the 'traditional' sun shades (refer to Table 2).

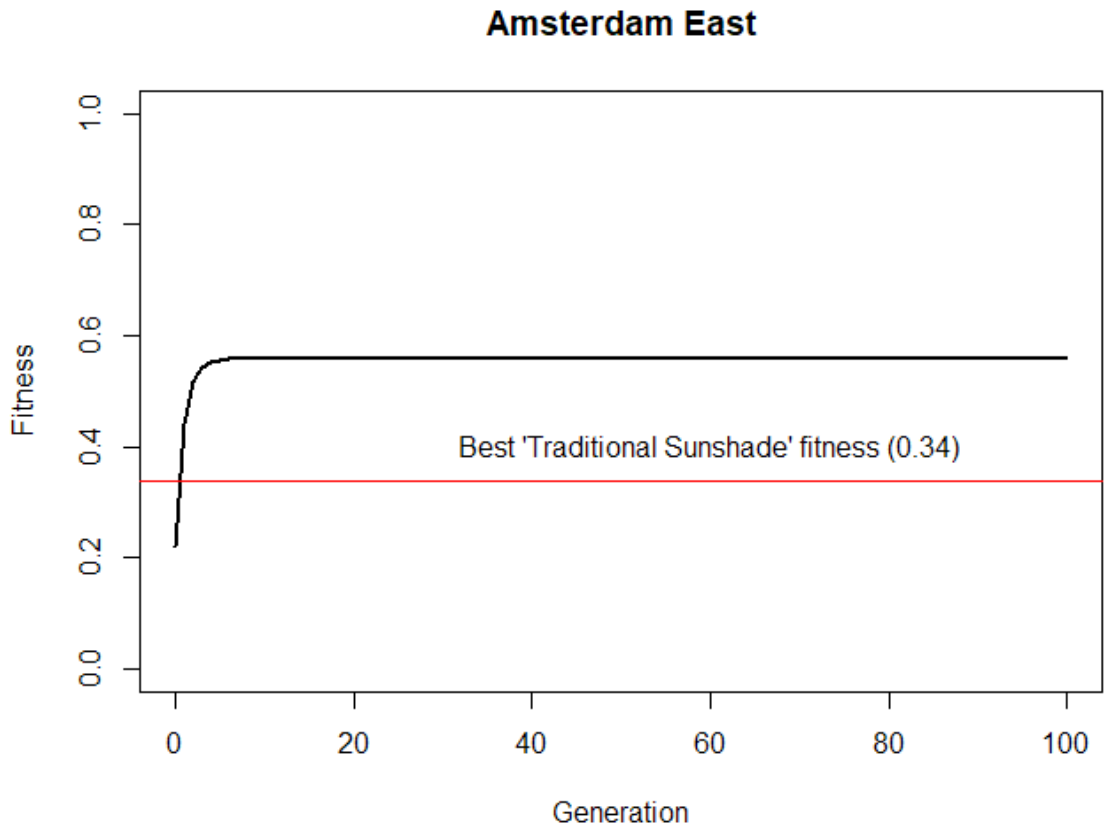


Figure 50: Evolutionary Strategy - Amsterdam East fitness. (Author)

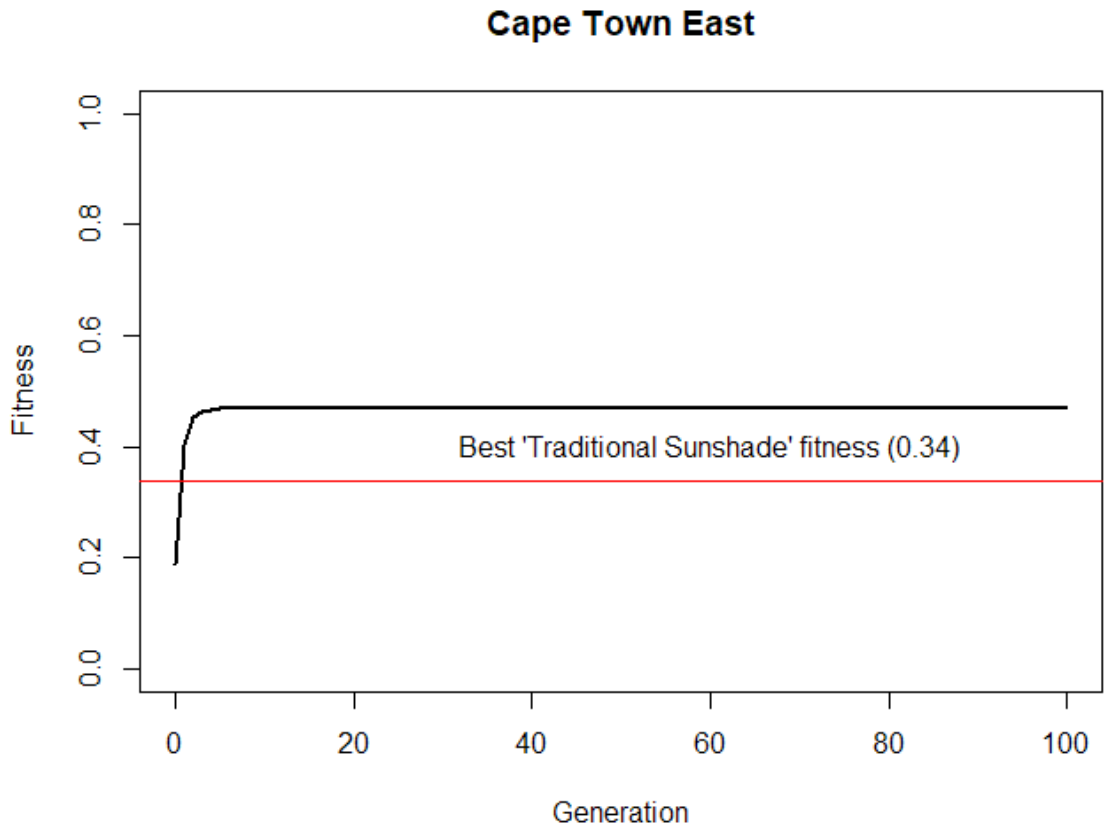


Figure 51: Evolutionary Strategy - Cape Town East fitness. (Author)

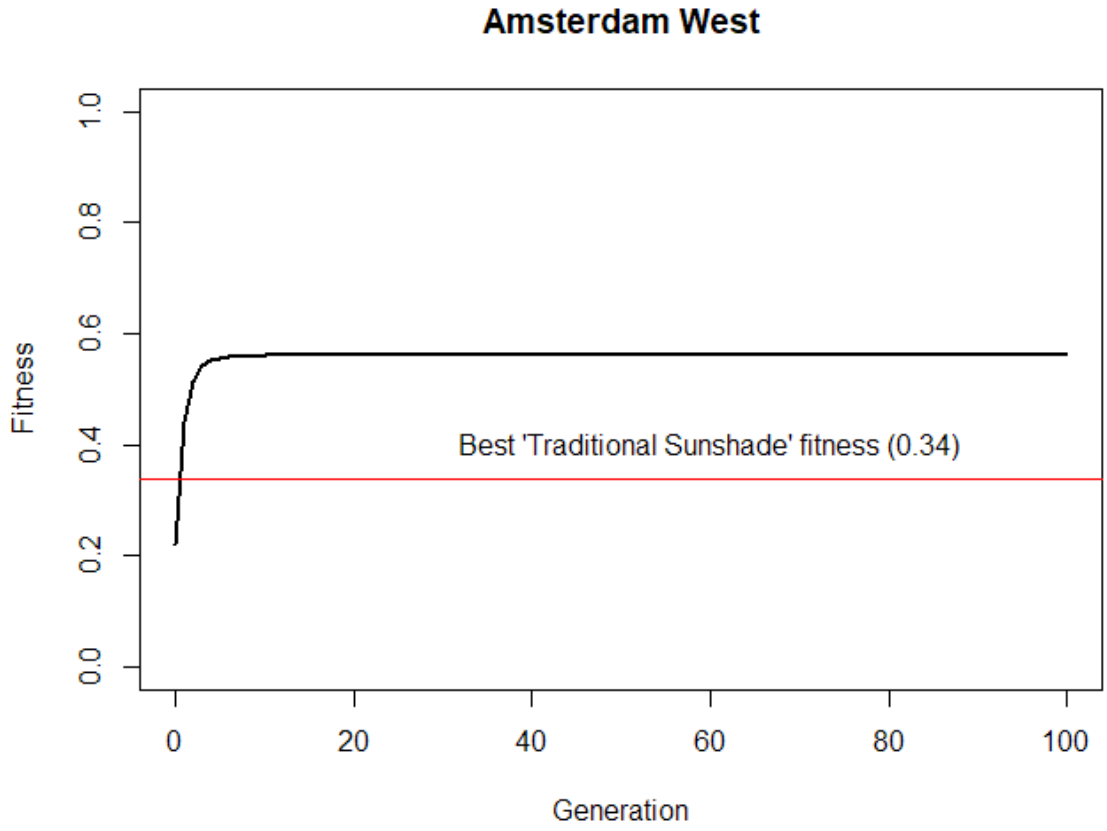


Figure 52: Evolutionary Strategy - Amsterdam West fitness. (Author)

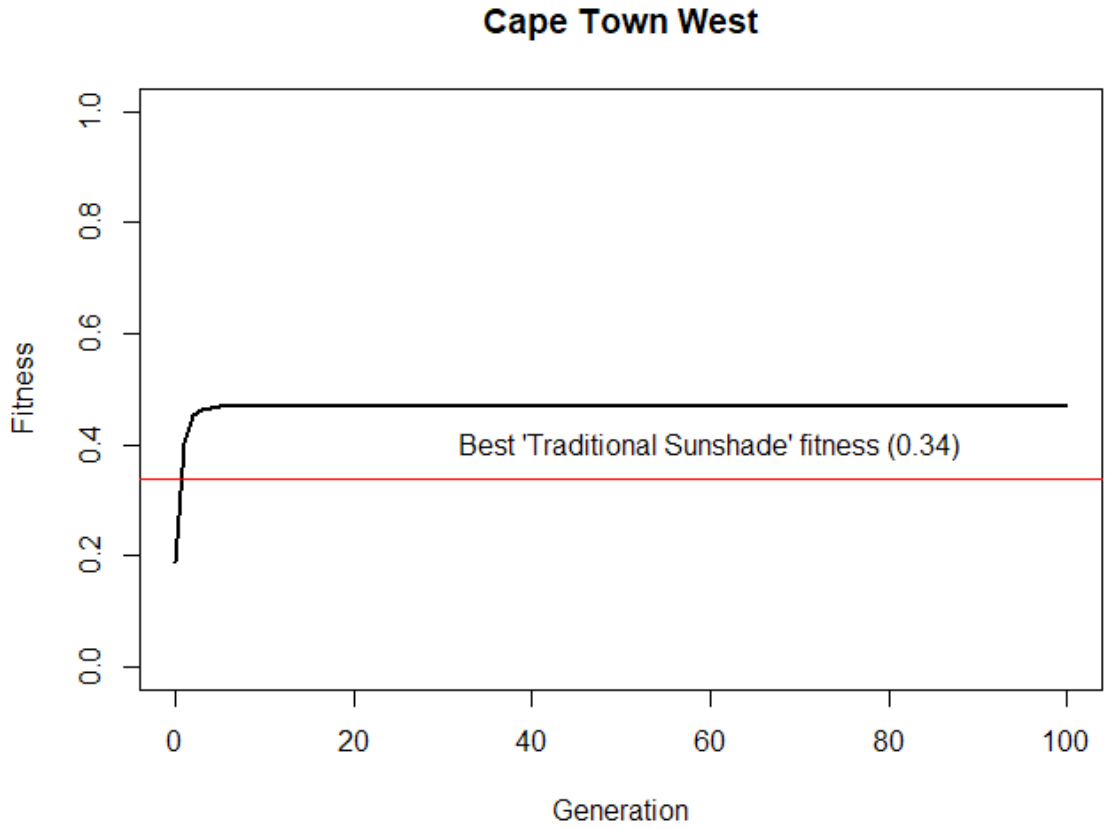


Figure 53: Evolutionary Strategy - Cape Town West fitness. (Author)

In each instance, the Evolution Strategies' mean fitness surpasses that of the best 'traditional' sun-shade before the fifth generation, indicating the extreme efficiency of this algorithm for shape optimisation. The fitness values continue to climb rapidly, and as confirmed previously, reach peak fitness by the tenth generation. The level of fitness achieved by the evolved sun-shades are significantly higher than the 'traditional' sun-shades. It is interesting to note that both the Amsterdam normalised fitness values are higher than those found in Cape Town. Torres and Sakamoto noted in their research that by overpopulating the search space, they managed to achieve a fitter model. It could be that as Amsterdam could test fitness across a longer time frame they could achieve a similar result ('overpopulation') and hence produce the higher normalised fitness values.

Architecturally the case study is interesting during the high fitness climb as most 'form-finding' takes place at this time. Thereafter the changes are small enough to be insignificant.

The tables in **Appendix 6** indicate the comparative fitness values between the two conditions in the case study. Tables 9 and 10 outline the values found for the East façade while Tables 11 and 12 outline the values for the West façade. As noted above, both Southern Hemisphere façade's have lower fitness values but are similar in total, while the Northern Hemisphere values are generally higher - this could be the result

of both a longer period of sun being experienced at the summer solstice and a larger spread of solar values (i.e. solar angle - the compass value).

Again, all values are normalised where zero indicates no fitness value and one indicates complete fitness ('full fitness'). Using these values, we can test the fitness of evolved sun-shades against the fitness of 'traditional sun shades.'

5.5 Hypothesis testing - single sample T test (two tailed)

To determine if the evolved sun-shades perform better than the traditional sun-shades, a single sample T test is employed. This case study starts with the assumption that the traditional sun-shades are the best - the null hypothesis - until there is evidence that indicates otherwise (the alternative hypothesis). A two tailed test is applied as the Critical Value (c) may appear on either side of the bell-shaped distribution being a 'less than' or 'greater than' value ($\frac{\alpha}{2}$).

Normalised fitness values for the ten traditional sun-shades are compared to the normalised fitness values for the ten evolutionary algorithm runs for each condition.

The Traditional sun-shades produce the following values; the (normalised) Mean for the 10 'traditional' sun-shades is 0,23 (with 0.100 as the standard deviation). This leads to:
H_0 (the null hypothesis) $\mu = 0,23$
H_a (alternate hypothesis) $\mu \neq 0,23$

A significance value of 5% is used.
$\alpha = 0.05$

A sample size of 10 is used.
N = 10

The following values are the mean fitness values per condition:
Amsterdam East: $\overline{x} = 0.56$
Cape Town East: $\overline{x} = 0.47$
Amsterdam West: $\overline{x} = 0.56$
Cape Town West: $\overline{x} = 0.47$

Standard deviation:
$s = 0$ for all samples

Following the single sample T test

$$t = \frac{\overline{x} - \mu}{\frac{s}{\sqrt{n}}} \tag{9}$$

The following may be evaluated:
Amsterdam East:

$$t = \frac{0,56 - 0,23}{\frac{0}{\sqrt{10}}} \tag{10}$$

Cape Town East:

$$t = \frac{0,47 - 0,23}{\frac{0}{\sqrt{10}}} \tag{11}$$

Amsterdam West:

$$t = \frac{0,56 - 0,23}{\frac{0}{\sqrt{10}}} \tag{12}$$

Cape Town West:

$$t = \frac{0,47 - 0,23}{\frac{0}{\sqrt{10}}} \tag{13}$$

Critical value (c) = 2,262 (positive and negative values)
This value from a Table where $\frac{\alpha}{2}$ is 0,025 and df (degrees of freedom) is sample size - 1 or $(n-1)$) resulting in $n = 9$.

Comparison to the Critical Value (c): $t \sim c$
Where $t > c$ reject H_0 in favour of H_a at 95% significance level.

5.5.1 Comments and observations

The evolved sun-shades produce a standard deviation of zero as after 100 generations the final values are so refined that there is no variance in fitness. While the sample standard deviation (s) is zero, t will be undefined. What affect this has on the bell curve is that the value will be very much increased with the curve being quite narrow. This indicates a statistically significant difference. I would conclude that as t is a higher value ($t > c$) it falls within the 'rejection' range of the distribution. This test statistic is so extreme that it makes it improbable if the null hypothesis is true. Consequently, I reject the null hypothesis at a 95% significance level; namely, that traditional sun-shades have the best fitness for task. The evolved screens are fitter by over two standard deviations. This indicates that it is extremely likely that our evolved sun-shade fitness values will surpass those of the traditional sun-shades. It would be worth noting that having a standard deviation of zero could be an anomaly (possibly due to rounding errors in the graphing software) but is most likely the result of continued evolution resulting in maximum fitness.

6 Discussion

6.1 Form-Finding

When the initial parameters were set, a rectangular volume with a rectangular opening in the base was envisaged. However, results from the case study indicate forms along the top of the volume dominate. This was unexpected. Furthermore, vertical fins seem to be replaced by smooth curves bulging out from the horizontal sun-shade, a far subtler solution than a dedicated 'device' or surface.

These form-finding results may point to the fact that horizontal sun rays change rapidly over time with points lower in the point cloud (that would block rapidly changing sun rays) not having particularly high fitness values as less rays pass through those points when striking the window surface given how fast the sun moves. Instead, it seems that horizontal forms dominate as those points intercept and block more (near vertical) sun rays as the sun at a higher angle moves slower, indicating more rays are available to be blocked and hence a higher fitness is achieved.

From the above, it would seem reasonable to imagine the points that make up vertical fins effectively block less sun rays simply because the sun path moves rapidly where they function with the consequence that fewer sun rays provide lower fitness values.

Could it be that the points within the point cloud do not experience the same potential to block the sun rays as the movement of the sun is not constant, given how variably the sun altitude changes (fast at sunrise, slower at midday)? From this, an amended approach could be proposed. Or will this amendment significantly alter the existing results and produce a more predictable solution; are we interfering with the design process?

An amended approach to address the domination of horizontal forms could be to focus on points not having the 'same potential' to block sun rays. This could be accomplished by 'grading' the points within the point cloud with regards to their ability to participate in blocking sun rays. One may see this as a 'heat map' where points that experience rapid sun ray change could be 'colder' while those who block the sun as it lingers at the noon apex would be 'hotter'. From this, after establishing 'bands' (or ranges) for grading the 'heat map', one could determine fitness levels per band or 'grade' as a relative value. For example, the 'green' band (or grade) would have high fitness while blocking less sun rays and could compete equally with the 'red' band where more sun rays are present allowing for more opportunities to block the sun rays on a window surface and hence have higher fitness values as a whole. A 'weighting' or 'equalisation' system of some sort would assist in distributing these new relative high fitness values across the point cloud.

Interestingly, this observation could lead to 'metadata' being connected to each point within the point cloud harking back to the 'motes' discussed by Frazer [31], the work that originally inspired the creation of the point cloud in the first place.

The converse is also applicable, namely, that the above 'heat map' interferes with a perfectly functional approach and is, instead, forcing the 'form-finding' result back to more familiar territory. This view would advocate that the implemented solution is entirely valid with one needing to realise that more sun rays tend to be near vertical

and generate this resulting form as a consequence.

6.2 Evolution Strategies

As discussed briefly (5.4.2), when using MeshLab on one run, every tenth generation was used to create a surface mesh and was saved as an image. These 'static' rendered images were combined to create an animated file where every image is 'held' and duplicated as an animation frame to create the animated clip. From this, it was visually possible to understand how the sun-shade was formed using the Evolution Strategies.

The initial evolution was too rapid to follow (this is confirmed by the graphs in **Experiments and Results** in Figures 50, 51, 52 and 53, where full fitness is achieved within ten generations). To better understand the initial evolution, a change was implemented in the animated image to, in effect, 'slow down' the first few generations. Each of the first twenty generations were saved and rendered as static MeshLab images. This was then combined with the other images saved for every tenth generation thereafter and was used to create a second animated clip.

From this clip it is possible to see an interesting evolution in the sun-shade, almost resembling a 'self-organising entity' over the initial generations. Thereafter the main form settles and lots of rapid, but very small, changes take place. These final changes appearing inconsequential.

This seems to confirm the graphed results that indicate rapid changes in fitness initially and then less in later generations. This was present in the first implementation of the Evolutionary Strategy where the top 10% of the population would be selected to reproduce. It was felt that this elitism rushed to a local optimum too soon with the solution space not being sufficiently explored. To address this, the top half of the parents were made eligible for selection to improve upon diversity. In addition, mutation was initially based upon a normal distribution (around zero) with a random Gaussian vector; this was also adjusted to be more inline with Beyer Schwefel's [9] proposed values for strategy parameters with attention paid to self-adaptation with the recommended learning parameter. While an improvement, it still seems the solutions are too narrow. Effectively generating a working solution within the first ten generations out of one hundred seems premature. Either the search space could have been explored further or the number of generations required could have been reduced.

6.3 General observations

Architecturally, in order to generate an artefact (a surface mesh) this case study required being located within a Cartesian co-ordinate system. From this starting point, real-value vectors would be evolved requiring no encoding. In addition, visualising the solution space when working with a point cloud, seems intuitive. Similarly, intermediary recombination is understandable as it applies to real-value parameters and allows for offspring to be near or around parental variable values. The produced artefact saw improvement when self-adaptation was incorporated making Evolution Strategies well suited to this case study.

However, again architecturally, the exploration of the solution space seemed to

have narrowed after the tenth generation. As Eiben Smith [26] note, (μ, λ) is generally preferred for Evolution Strategies where this case study instead uses $(\mu + \lambda)$. It could have been better to discard all parents in order to leave the local optima and in addition, discard outdated solutions (because the fitness optimum moves). This could explain the multiple small and insignificant changes that the animation indicates takes place in later populations. Eiben Smith also go on to note how $(\mu + \lambda)$ hinders self-adaptation as 'misadapted' strategy parameters may survive across a number of generations, exacerbated where there are children with bad strategy parameters and selection is elitist. Elitism was recognised as a concern once 10% of the parents were initially selected - hence the modification to 50% of parents later being considered eligible for selection. Looking at both the results printed for each generation and the visual representation contained within the animation, mutation and recombination could be reconsidered. Of course, this must be seen in combination with survivor selection as that would have a significant impact. It seems children are rather close to their parents. In later generations this is visible as a large group of individuals ranked by fitness alongside each other - essentially a successful parent alongside successful offspring[xxx]. However, as noted previously, the generation count does not go far back indicating that most individuals are recent; this being clearly seen in **Appendix 4** 'Individual Fitness values', specifically Table 4, the last three columns. These columns represent the individual, the parent and the generation. What is clearly demonstrated for this final generation is how many children are ranked side by side - Parent 669 and Parent 670 having many offspring present.

[xxx]In the point cloud, this would be a cluster of individual points which, in MeshLab, do not significantly contribute towards generating a mesh surface.

7 Conclusion and Future Work

7.1 Conclusion

This case study clearly demonstrates it is possible to use an Evolutionary Algorithm to Architecturally explore and generate form. Where previous studies typically consider surface elements that are 're-arranged' [60] [64] or have their geometry 'transformed' (lengthened, rotated and so on) [42], what makes this work stand out is that it explores 'pure' Evolutionary Algorithms that, when combined with open source meshing software, explore form to produce artefacts that can be examined three-dimensionally or to scale as printed three-dimensional models. No pre-coded proprietary software solutions are required where the user is removed from how the Evolutionary Algorithm functions. Furthermore, meshed solutions can be dynamically studied as they can draw directly upon the evolved point cloud; in more recent studies this dynamic link with software to produce an artefact was considered a strong point to promote the use of propriety solutions [67]. In addition, this case study determined a method that could be employed to ascertain the fitness values of 'Traditional' sun-shades to allow for their assessment alongside those generated by the Evolutionary Algorithm within a Cartesian co-ordinate system. This allowed for both the Traditional and Evolved sun-shade to be evaluated in terms of fitness to determine the best form required to block direct sunlight under the same conditions.

7.2 Future Work

Given the above, a number of changes can be suggested and indeed additional avenues of exploration can be considered in future work, outlined below.

7.2.1 Explore other façade conditions

In this case study, the East and West façades conditions were explored, however, the characteristics of the sun path in these circumstances are fairly similar. Namely, the sun starts low and horizontal then rotates to the North/South façade until almost directly overhead but flush with the wall surface with the inverse taking place on the other façade. Studies of the North/South façade would require a different solution as the sun starts at a high angle overhead but flush with the wall surface then twisting to be directly perpendicular to the façade but at a higher angle before lowering again and moving past the façade. Typically horizontal overhangs dominate these traditional solutions, it would be interesting to compare this with the evolved solutions; are they similar or have vertical fins of some sort been added too?

7.2.2 Changes to the Evolutionary Algorithm

The algorithm itself may see modification; the number of individuals per generation is sufficient (the points that make up the point cloud is good) but it may be that exploration within the solution space is too abrupt. It could be that the shape optimisation strength

of the Evolution Strategies implementation does not require a large number of generation to achieve a result - this is a strength as a large amount of solutions may be explored in a shorter time. Or it may be that there was a convergence to a local maxima, typically a consequence of elitism. If this is the case, solutions could include over populating the search space or increasing variation in the breeding group. As noted above, discarding parents could also address this.

Parameter settings could require modification. Kramer [40], in his survey of self-adaptive parameters, discusses 'adequate parameter settings' and notes the impact this has on the success of the optimisation process. Following this survey, it would seem that self-adaptive control of mutation strength is still an optimal approach.

7.2.3 Add simulation to improve upon the idea of fitness

What makes this case study differ significantly from those considered, is the connection to an environmental simulation. To determine the fitness of an individual, an alternate method of counting sun-rays blocked was used. This worked well within the context of the Evolutionary Algorithm but is a less mature solution that can be improved upon by linking the Evolutionary Algorithm to a simulation. Architecturally, this makes more sense but does require a more 'architecturally' accurate model, as noted previously, to incorporate an energy and lighting simulation that could produce in-situ results allowing these to be interpreted in an architectural way (a good example of this would be heat efficiency [30]). An approach to connecting the evolved sun-shade to a simulation would be to set up a three-dimensional environment that functions within an active simulation engine. The environment can be set up to contain objects that would be architecturally relevant to the simulation; this may be neighbouring buildings as block models, the building wall surface with the window opening, possibly a volume behind this opening to represent a room to allow for measurements to be taken (this follows the approach used by Manzan and Pinto [42]). This environment can be established to allow for the evolved sun-shade 3D mesh model to be imported such that certain values remain consistent (the direction of the wall surface and position of the window opening relative to the evolved sun-shade). This will allow for consistency in results and the rapid exploration of multiple sun-shades as they are all located in the correct position and orientation. Pre and post sun-shade measurements can be taken and compared. The 3D environment could be established within Rhino 3D or Trimble SketchUp software, both allow for mesh models to be imported. For Rhino 3D, the simulation would typically be driven by Grasshopper plug-ins (as noted in the studies investigated). For SketchUp, OpenStudio[xxxi] takes the place of Grasshopper plug-ins and allows for simulations to be active. Using SketchUp and OpenStudio is another common approach and is further explored by Al-Zubayi [2] and Bojic et al. [12]. Once the environment is set, solutions may be tested and measured against various parameters, namely solar heat gain, glare and so on.

[xxxi] OpenStudio: An open source integrated analysis platform `https://openstudio.net/`

7.2.4 Reconnecting to 'original' Form-finding principals

This case study was inspired by, and is an interpretation of, the idea of 'a field of motes' as initially described by Frazer [31]. This conceptual idea was then applied to implement the Architectural notion of form-finding. Where there is significant scope for future work would be to 'reconnect' to the original idea of form-finding, as discussed by Burry [14], by working to incorporate physics and material properties (also referred to as the 'self-organisation of materials') into future case studies. This, in turn, connects to the original work of Antoni Gaudí and Frei Otto where the characteristics of the material used in the construction combine with gravity to play a role in determining the final form. Piker [49] appreciated this connection between gravity and form-finding by making use of the Grasshopper plug-in, Kangaroo, to introduce physics into the Rhinoceros 3D modelling world. This case study could be progressed by locating the point cloud within a physics engine. Nvidia's PhysX could be a platform on which to build this research [37] with Havok offering another option [41]. It should be noted that there are several physics engines with the web-based company, G2 inc, ranking the software [xxxii]. In addition, the points may be modified to better fit Frazer's idea of motes where each mote could be a point containing metadata of some sort. Frazer mentions each mote being aware of others, in a sense this could be a development of 'meta-ball' surfaces (briefly summarised by Pottmann et al. [51]) but now contained within an environment that applies an effect (Templet [57] could be a starting point).

7.2.5 Meshing and surfacing of the point cloud

MeshLab [16] was used to create a surfaced model from the final, evolved point cloud; this being the form-finding artefact (the sun-shade). However, this surface model requires slight modification to allow for 3D printing[xxxiii] and rendering in visualisations. The MeshLab product is a single skinned, triangular surface mesh where the surface normal plays a role (typically making the product 'invisible' when viewed 'from behind' - in the direction of the normal vector). To resolve this, Architectural 3D modelling would attempt to 'add thickness' to the surface to create a 'solid volume' where there is a surface viewable in all directions. Architecturally this also allows for additional consideration regarding dimensions (the thickness) and construction methods - this rapidly goes beyond making the object visible and crosses into considering how it would be constructed and with what materials. Furthermore, triangular meshes are no longer preferred being replaced by polygon meshes where subdivision surfaces are preferred and are dominating the modelling tasks. Pottmann et al. [51] again provides a significant overview of this field. Future work would possibly include revisiting the use of MeshLab: in this case study, duplicate points were removed within the software to reduce complexity and certain surface sampling techniques were implemented but this could be more thoroughly explored. In addition, future work can be focused on maturing

[xxxii] `https://www.g2.com/categories/physics-engine`

[xxxiii] Wikiversity provide a good sample 'step-by-step' process to follow. That can be found at this link: `https://en.wikiversity.org/wiki/MakerBot/Cleaning_Up_Point_Cloud_Meshes_in_Meshlab_For_3D_Printing`.

a solution that can take a point cloud and produce a simpler polygon surface mesh. This would need to include the reduction of points (possibly using a distance threshold), and a robust method to create polygon surfaces (that may require the addition of certain points to better form a surface; possibly starting with a large polygon surface and subdividing this).

References

[1] Agkathidis, A. Implementing biomorphic design. In *The 34th International Conference on Education and Research in Computer Aided Architectural Design, eCAADe* (2016), pp. 291–298.

[2] Al-Zubaydi, A. Y. Building models design and energy simulation with Google SketchUp and OpenStudio. *Journal of Advanced Science and Engineering Research 3*, 4 (2013), 318–333.

[3] Alexander, C. *Notes on the Synthesis of Form*, vol. 5. Harvard University Press, Cambridge, MA, USA, 1964.

[4] Alexander, C. *A Pattern Language: towns, buildings, construction.* Oxford University Press, Oxford, UK, 1977.

[5] Asojo, A. O. Design algorithms after Le Corbusier. CumInCAD, ACADIA Quarterly, Vol. 19, 4 (2000), 17-24.

[6] Back, T., Fogel, D. B., and Michalewicz, Z. *Handbook of Evolutionary Computation.* IOP Publishing Ltd., Bristol, UK, 1997.

[7] Bader, J., and Zitzler, E. Hype: An algorithm for fast hypervolume-based many-objective optimization. *Evolutionary computation 19*, 1 (2011), 45–76.

[8] Beyer, H.-G. *The Theory of Evolution Strategies.* Springer Science & Business Media, Dordrecht, Netherlands, 2001.

[9] Beyer, H.-G., and Schwefel, H.-P. Evolution strategies - a comprehensive introduction. *Natural computing 1*, 1 (2002), 3–52.

[10] Bhatt, R. Christopher Alexander's Pattern Language: an alternative exploration of space-making practices. *The Journal of Architecture 15*, 6 (2010), 711–729.

[11] Bleuler, S., Laumanns, M., Thiele, L., and Zitzler, E. Pisa - a platform and programming language independent interface for search algorithms. In *International Conference on Evolutionary Multi-Criterion Optimization* (2003), Springer, pp. 494–508.

[12] Bojic, M., Skerlic, J., Nikolic, D., Cvetkovic, D., and Miletic, M. Toward future positive net-energy buildings. *EXPRES 2012* (2012), 49.

[13] Bonnemaison, S. *Resurgence of Organicism.* Dalhousie Architectural Press, Halifax, Nova Scotia, Canada, 2019.

[14] Burry, M. Antoni Gaudí and Frei Otto: Essential precursors to the parametricism manifesto. *Architectural Design 86*, 2 (2016-03-01), 30,35.

[15] Ching, F. D. *Architectural graphics.* John Wiley & Sons, Hoboken, NJ, USA, 2015.

[16] Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., and Ranzuglia, G. MeshLab: an Open-Source Mesh Processing Tool. In *Eurographics Italian Chapter Conference* (2008), V. Scarano, R. D. Chiara, and U. Erra, Eds., The Eurographics Association.

[17] Coates, P., and Makris, D. Genetic programming and spatial morphogenesis. *AISB Symposium on Creative Evolutionary Systems*, Edinburgh College of Art and Division of Informatics (AISB'99), University of Edinburgh, March 1999

[18] Coetzee, L., and Nitschke, G. Evolving optimal sun-shading building façades. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (New York, NY, USA, 2019), GECCO '19, Association for Computing Machinery, p. 393–394.

[19] Conway, J. The Game of Life. *Scientific American 223*, 4 (1970), 4.

[20] Darwin, C. *The Origin of Species: by Means of Natural Selection or the Preservation of Favored Races in the struggle for Life*, vol. 2. Modern library, New York City, NY, USA, 1872.

[21] Dawes, M. J., and Ostwald, M. J. Christopher Alexander's a Pattern Language: analysing, mapping and classifying the critical response. *City, Territory and Architecture 4*, 1 (2017), 1–14.

[22] Dawkins, R. Blind watchmaker biomorphs. In *The Pattern Book: Fractals, Art, And Nature*. World Scientific Publishing, Hackensack, NJ, USA, 1995, pp. 9–11.

[23] De Jong, K. *Evolutionary computation: a unified approach*. MIT Press, Cambridge, MA, USA, 2006.

[24] De Jong, K., Fogel, D. B., and Schwefel, H.-P. A history of evolutionary computation. *Handbook of Evolutionary Computation A 2* (1997), 1–12.

[25] DeLanda, M. Deleuze and the use of the genetic algorithm in architecture. *Architectural Design 71*, 7 (2002), 9.

[26] Eiben, A., and Smith, J. *Introduction to Evolutionary Computing (natural computing series)*. Springer-Verlag, Berlin, Germany, 2008.

[27] Eiben, A., and Schoenauer, M. Evolutionary computing. *Information Processing Letters 82*, 1 (2002), 1–6.

[28] Energy Systems Research Unit, Glasgow, S. ESP-r (a general purpose, multi-domain simulation environment).

[29] Fletcher, B. *A history of Architecture on the Comparative Method*. T Batsford Ltd., London, UK, 1958.

[30] Franzetti, C., Fraisse, G., and Achard, G. Influence of the coupling between daylight and artificial lighting on thermal loads in office buildings. *Energy and Buildings 36*, 2 (2004), 117–126.

[31] Frazer, J. *Themes VII: An Evolutionary Architecture*. Architectural Association, London, UK, 1995.

[32] Galilei, G. *Dialogues concerning two new sciences.* Dover Publications, Mineola, New York, USA, 1914.

[33] Gut, P., and Ackerknecht, D. *Climate responsive buildings: appropriate building construction in tropical and subtropical regions.* SKAT Foundation, Gallen, Switzerland, 1993.

[34] Hensel, M., Menges, A., and Weinstock, M. Emergence: morphogenetic design strategies, 2004.

[35] Holland, B. Computational organicism: Examining evolutionary design strategies in architecture. *Nexus Network Journal 12*, 3 (2010), 485 – 495.

[36] Holland, J. H. Genetic algorithms. *Scientific American 267*, 1 (1992), 66–73.

[37] Hongpan, N., Yong, G., and Zhongming, H. Application research of PhysX engine in virtual environment. In *2010 International Conference on Audio, Language and Image Processing* (2010), pp. 587–591.

[38] Huang, Y., and Niu, J.-l. Optimal building envelope design based on simulated performance: History, current status and new potentials. *Energy & Buildings*, 117 (2016), 387–398.

[39] Kolarevic, B. Digital morphogenesis and computational architectures. *Construindo n (o) espaço digital, PROURB, Universidade Federal do Rio de Janeiro, Rio de Janeiro* (2000), 98–103.

[40] Kramer, O. Evolutionary self-adaptation: a survey of operators and strategy parameters. *Evolutionary Intelligence 3* (2010), 51–65.

[41] Mac Mathuna, D. Playing Havok with physics. *Physics World 16*, 8 (2003), 10.

[42] Manzan, M., and Pinto, F. Genetic optimization of external shading devices. In *Proceedings of 11th international IBPSA conference, Glasgow, Scotland* (2009), pp. 27–30.

[43] Martinell, C. *Gaudí - His Life. His Theories. His Work.*, 1975 ed. The MIT Press, Cambridge, MA, USA, 1967.

[44] McCarthy, J. Generality in artificial intelligence. *Communications of the ACM 30*, 12 (1987), 1030–1035.

[45] Mitchell, W. J. *The Logic of Architecture. Design, computation and cognition.* The MIT Press, Cambridge, MA, USA, 1990.

[46] Neumann, J., Burks, A. W., et al. *Theory of self-reproducing automata*, vol. 1102024. University of Illinois Press, Urbana, IL, USA, 1966.

[47] O'Conner, J., Lee, E., Rubinstein, F., and Selkowitz, S. Tips for daylighting with windows: The integrated approach. Technical Report, Lawrence Berkeley National Laboratory, Berkeley, CA, USA, No. PUB-790. 1997.

[48] Olgyay, A., Olgyay, V., et al. *Solar control and shading devices*. Princeton University Press, Princeton, NJ, USA, 1976.

[49] Piker, D. Kangaroo: form-finding with computational physics. *Architectural Design 83*, 2 (2013), 136–137.

[50] Poloni, C., and Mosetti, G. Aerodynamic shape optimization by means of a genetic algorithm. In *the 5th international symposium on computational fluid dynamics* (1993), Japan Society of computational fluid dynamics, pp. 279–284.

[51] Pottmann, H. *Architectural Geometry*, vol. 10. Bentley Institute Press, Exton, PA, USA, 2007.

[52] Reekie, R. F. *Draughtsmanship*. Edward Arnold & Co., London, UK, 1961.

[53] Richards, S. J. *Solar Charts for the Design of Sunlight and Shade for Buildings in South Africa*. National Building Research Institute, CSIR, Brummeria, Pretoria, RSA, 1981.

[54] Rigoni, E., and Poles, S. NBI and Moga-II, two complementary algorithms for multi-objective optimizations. In *Dagstuhl Seminar Proceedings* (2005), Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[55] Schumacher, P. Parametricism: A new global style for architecture and urban design. *Architectural Design 79*, 4 (2009), 14–23.

[56] Stiny, G. Introduction to shape and shape grammars. *Environment and planning B: planning and design 7*, 3 (1980), 343–351.

[57] Templet, R. M. *Game Physics: An Analysis of Physics Engines for First-Time Physics Developers*. PhD thesis, California State University, Northridge, 2020.

[58] Thompson, D. W. Galileo and the principle of similitude. *Nature 95*, 2381 (1915), 426–427.

[59] Thompson, D. W. *On Growth and Form*, vol. 2. Cambridge University Press, Cambridge, UK, 1942.

[60] Torres, S., and Sakamoto, Y. Façade design optimisation for daylight with a simple genetic algorithm. *Building Simulation Proceedings* (2007).

[61] Van Doesburg, T. Towards a plastic architecture. *de Stijl*, vol. 12 Studio Vista London, 6/7 (1924), 78-83.

[62] Wang, W., Rivard, H., and Zmeureanu, R. An object-oriented framework for simulation-based green building design optimization with genetic algorithms. 5–23.

[63] Wolfram, S. Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena 10*, 1-2 (1984), 1–35.

[64] Wright, J. A., and Mourshed, M. Geometric optimization of fenestration. *Building Simulation conference* (2009).

[65] Zerbst, R. *Antoni Gaudí*, Architecture and Design series ed. Taschen Publications, Cologne, Germany, 1993.

[66] Zexin, S., and Mei, H. Robotic form-finding and construction based on the architectural projection logic. In *IOP Conference Series: Materials Science and Engineering* (2017), vol. 216, IOP Publishing, p. 012058.

[67] Zhang, L., Zhang, L., and Wang, Y. Shape optimiszation of free-form buildings based on solar radiation gain and space efficiency using multi-objective genetic algorithm in the severe cold zones of china. *Solar Energy*, 132 (2016), 38–50.

[68] Zitzler, E., Laumanns, M., and Thiele, L. Spea2: Improving the strength pareto evolutionary algorithm. *TIK-report 103* (2001).

8 Appendix 1 - Source Code

Python Source Code for Evolutionary Strategy written for Python 3 with Python 2.7 compatibility. (Comments outlining tests code and log files for evaluating results and testing the script have been removed to keep the content clearer.)

```
# EVOLUTIONARY STRATEGY
# This code run: Amsterdam Sun: South Facade
# Supervisor: Geoff Nitschke
# May 2017 implementation
# This was written in Python 3 but as the
# HPC only supports Python 2.7, has been rewritten
# to include Python 2.7 compatibility

# General implementation:
# Parents randomly selected from top half of
# population (top 10 percent too elitist)
# Sun angle read in for every 15s
# More complexity - ES works better with more complexity
# Test run: 9 500 pop, 100 Generations
# Actual run: 20 000 pop, 100 Generations
# this to create finer, smoother results
# Generate Point Cloud (adults)
# Evaluate fitness - each ray, each 15s
# (Most rays blocked = higher fitness)
# (Certain positions - obstructing view - auto unfit)
# Rank them
# OLD --- Keep top 10 percent of parents - (too elite) ---
# CURRENT Keep top half of parents
# Randomly select 200 parents from top half, discard rest
# For each parent, generate 10 mutated offspring,
# mutations based on normal distribution (Gaussian)
# [Normal Distribution] The horizontal value ranges
# from -150 to +150, this value is
# added/subtracted from X Y Z value
# New population ranked by fitness, repeat



# sample = int(np.random.normal(loc=0.0,scale=150))
```

```
# ** the number 150 can change,
# tweak as needed (Its based on
# the smallest measurement halved,
# thinking 200mm/2)

# Do it for each X,Y and Z

# Amsterdam, Cape Town, mid-summer,
# 15 seconds over 1 day
# 100 generations

# Mesh model (we want to see the mesh changes)
# Evaluate temperature

# Originally, replaced 'guessed' value
# in gaussian mutation

# Imports
import random
from random import randint
import sys
import math
import numpy as np
import csv
from datetime import datetime
import functools, random, itertools

# Written Output files:
# The final Point Cloud file (matrix; x, y, z)
output = open('ES-3v7_output.txt', 'w')
# A file to time the algorithm
timer = open('ES-3v7_timer.txt', 'w')
# A file to create 3D mesh, every X (10th) generation
outputXgen = open('ES-3v7_outputXgen.txt', 'w')

# Numpy to show full values in matrix
```

```
np.set_printoptions(threshold=np.nan, precision=3, suppress=True,
 linewidth=200)

# Variables
gen_count = 100  # 100 (Generation Count)
pop_size = 20000  # 9 500 OR 20 000 (Population Size)
xGen = 10  # The number of children each parent spawns
xGen3Dcounter = 10  # Write every 10th generation to outputXgen file
x1 = 0  # a counter (X can be anything)

# USER MEASUREMENTS, bounding box and fitness created from it
# 'Window Surface': actual measurements are (millimetres):
# x = 650x2 (1300), y = 850x2 (1700), z = 200
width = 1300 # range for width (X)
height = 1700  # range for height (Y)
depth = 200  # range for depth (Z)

# 'Bounding Box' of point cloud:
# actual measurements increased on each side
boundingWidth = int(width + (width/10)*2) # 10 precent on either side
boundingHeight = int(height + (height/5)*2) # 20 percent either side
boundingDepth = int((boundingHeight/3)*2) # two thirds bounding height
surfaceOffset = int(boundingDepth/10) # keep points off surface window
sampleScale = int(boundingDepth/3) # scale value - bell curve (depth/3)

# Window Fitness range (0,0 is located on corner of bounding box,
# this offsets it inwards)
startX = int(boundingWidth - width)/2 # 10 percent inwards
endX = int(startX + width)  # length of window width (+ 'offset')
startY = int(boundingHeight - height)/2 # 20 percent inwards
endY = int(startY + height)  # height of actual window height
third = int(startY + (height/3)) # 1/3 of window height

## ES IMPLEMENTATION
N = 10   # N variances vary individually (Global learning Rate)
c = 1     # Beyer Swefel, section 27, pg 29; c=1 a reasonable choice
tau_0 = (c/math.sqrt(2*N)) # Global Learning Rate
```

```
tau = (c/math.sqrt(2*(math.sqrt(N)))) # Child Learning Rate
# Global Learning Rate (Tau0) 1/square root of 2 * n
# Child Learning Rate (Tau) 1/square root of 2 square root of n

# Symbols for easier use in algorithm
g = 0
p = pop_size
w = int(boundingWidth + 1)  # width (X)
h = int(boundingHeight + 1)  # height (Y)
d = int(boundingDepth + 1)  # depth (Z)
# t = int(pop_size / 10) # Original setting - top 10 percent
t = int(pop_size / 2) # To select 50 percent of population
tminus = t-1
tenth = int(pop_size / 10)

# Code to not randomly select same row number again
def random_no_repeat2(random_func):
    already_returned = set()
    while True:
        i = random_func()
        if i not in already_returned:
            already_returned.add(i)
            yield i

# Initialise the 2D Numpy Array (0 = row, 10 = columns)
# Format: column 1: fitness | column 2: X | column 3: Y | column 4: Z
# | (step size) column 5: mutation strength sigmaX | column 6: sigmaY
# | column 7:sigmaZ | column 8: Individual | column 9: which x10 cycle
# | column 10: generation
database = np.empty((0, 10), int)
currentdatabase = np.empty((0, 10), int)

# To read in a file and making that compatible with Python 2.x and 3.x
if sys.version_info[0] == 2:  # Not named on 2.6
    access = 'wb'
    kwargs = {}
else:
```

```
    access = 'wt'
    kwargs = {'newline':''}

# FITNESS FUNCTION
def fitness(x, y, z):
    fitness = 0
    # Read in text
    # V, vertical | H, horizontal
    for row in sunPosition:
        currentline = str(row[0]) # read lines return list of strings
        V, H = currentline.split(",") # string split to two values

        V = V.replace(',', '')  # format string remove comma
        V = V.replace(' ', '')  # format string remove whitespace
        H = H.replace(',', '')  # format string remove comma
        H = H.replace(' ', '')  # format string remove whitespace
        V = float(V)  # convert V (string) to V float
        H = float(H)  # convert H (string) to H float

        # X Fitness
        # cannot divide TAN by 0
        if z == 0:
            xOnSurface = x
        else:
            xDifference = z / math.tan(math.radians(180-H))
            xOnSurface = int(x + xDifference)

        # Y Fitness
        if z == 0:
            yOnSurface = y
        else:
            yDifference = math.tan(math.radians(V)) * y
            yOnSurface = int(y - yDifference)

        if (startX <= xOnSurface <= int(endX) and startY <= yOnSurface
         <= int(endY)):
            fitness = fitness + 1
```

```
        else:
            fitness = fitness + 0

    return fitness

# get current time stamp
print(str(datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
timer.write(str(datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
timer.write('\n')

# Generate parents
while g < gen_count:
    if g == 0:  # PARENTS GENERATED
        while x1 < p:
            # Fitness - cut from here and placed in test function
            rw = random.randrange(0, w)  # Random int 0 to width+1
            rh = random.randrange(0, h)  # as above, 0 to height+1
            rd = random.randrange(1, d)  # as above, 1 to depth+1
            # (tangent in fitness calculation cannot div by 0)
            # mutation strengths
            sigma_0w = math.exp(tau_0*(np.random.normal(0,1)))
            sigma_0h = math.exp(tau_0*(np.random.normal(0,1)))
            sigma_0d = math.exp(tau_0*(np.random.normal(0,1)))

            if rd < surfaceOffset:
                fitt = 0
            elif (rh <= third) and (int(startX + width/ 10) <= rw
             <= int(endX - (width / 10))):
             # if below a 1/3, to be on edges (10 percent)
                fitt = 0
            else:
                sunPosition = csv.reader(open('Amst15_Sun-EAST.txt',
                **kwargs), delimiter=' ',
                quotechar='|')  # read in sun location file

                fitt = fitness(rw, rh, rd)
```

```
            # empty array appended w a parent as generated
            database = np.append(database, np.array([[fitt, rw, rh,
             rd, sigma_0w, sigma_0h, sigma_0d, x1, g, g]]), axis=0)

            x1 = x1 + 1

        # sort parents by first value in array (fitness)
        database = database[database[:, 0].argsort()[::-1]]

        # For troubleshooting, keep track of original parents
        original_database = database
        output.write('Starter generation (sorted initial parents)\n')
        output.write(str(original_database))
        output.write('\n')
        output.write('Numpy Array details: (ROWS,COLUMNS) - '
        + str(original_database.shape) + '\n') # for checking
        output.write('Numpy Array dimension details: '
        + str(original_database.ndim) + ' dimension\n') # for checking
        output.write('\n')

        # slice array to keep top X percent (now set to 50 precent)
        ### database = database[0:t]

        outputXgen.write(str(original_database))
        outputXgen.write('\n')

        # write out database content to numpy propriety file
        np.save('array.npy', database)

    else:  # OFFSPRING = PARENTS

        random_gen = random_no_repeat2(functools.partial(random.randint
        , 0, tminus)) # from top half the population
        values = list(itertools.islice(random_gen, t))
        # select 50 percent of whole population as parents

        for i in range(tenth):
```

```
# Each parent creates a tenth of total pop offsping

    database = np.load('array.npy')
    steps = i - 1
    r50 = values[steps]
    currentParent = database[r50, None, :]
   # choose from the random selection generated at the start,
   # 10 parents

    # I have an individual parent now to pull out their values
    # X, Y and Z
    parentX = currentParent[0, 1]
    parentY = currentParent[0, 2]
    parentZ = currentParent[0, 3]

    # I have an individual parent now to pull out their
    # sigma_0 values
    parentSigmaX = currentParent[0, 4]
    parentSigmaY = currentParent[0, 5]
    parentSigmaZ = currentParent[0, 6]

    thisGen = np.empty((0, 10), int)

    # ADAPTIVE
    for a in range(xGen):
        # 50 percent chance of negative value
        switch1 = randint(0, 1)
        if switch1 == 1:
            parentSigmaX = parentSigmaX * -1

        switch2 = randint(0, 1)
        if switch2 == 1:
            parentSigmaY = parentSigmaY * -1

        switch3 = randint(0, 1)
        if switch3 == 1:
            parentSigmaZ = parentSigmaZ * -1
```

```
# child sigma x parent sigma
childSigmaX = parentSigmaX*
(math.exp(tau*(np.random.normal(0,1))))
childSigmaY = parentSigmaY*
(math.exp(tau*(np.random.normal(0,1))))
childSigmaZ = parentSigmaZ*
(math.exp(tau*(np.random.normal(0,1))))

# X Y Z values changed by adaptive step
childX = parentX + childSigmaX
childY = parentY + childSigmaY
childZ = parentZ + childSigmaZ

if (childZ < int(surfaceOffset)) or (childZ >
 int(boundingDepth)):
    fit = 0
elif (childX != abs(childX) or childX >
 int(boundingWidth)) or (
        childY != abs(childY) or childY >
         int(boundingHeight)):
    fit = 0
elif (childY <= third) and (int(startX + width / 10)
 <= childX <= int(endX - (width / 10))):
   # if below a 1/3, needs to be on edges (10 percent)
    fit = 0
else:
    sunPosition = csv.reader(open('Amst15_Sun-EAST.txt'
    , **kwargs), delimiter=' ', quotechar='|')
     # read in sun location file

    # fittest.write('X Y and Z are valid
    # - fitness function can run\n')
    fit = fitness(childX, childY, childZ)

# Children collected into an array
# - this array is new every time per adult
```

```
                thisGen = np.append(thisGen, np.array([[fit, childX,
                 childY, childZ, childSigmaX, childSigmaY, childSigmaZ,
                  a, i, g]]), axis=0)

            parentChilddatabase = np.concatenate((currentParent,
             thisGen), axis=0)

            currentdatabase = np.concatenate((currentdatabase,
             parentChilddatabase), axis=0)

        # sort
        sorteddatabase = currentdatabase[currentdatabase[:, 0]
        .argsort()[::-1]]
        currentdatabase = np.empty((0, 10), int)
        # clear database for each cycle

        if g % xGen3Dcounter == 0:
            outputXgen.write(str(g) + '\n')
            outputXgen.write(str(sorteddatabase[0:p]))
            outputXgen.write('\n')

        output.write('Next generation of parents and children,
         generation ' + str(g) + '\n')
        output.write(str(sorteddatabase[0:p]))

        output.write('\n')
        output.write('Numpy Array details: (ROWS,COLUMNS) - '
         + str(sorteddatabase[0:p].shape) + '\n')  # for checking
        output.write('Numpy Array dimension details: '
         + str(sorteddatabase.ndim) + ' dimension\n')  # for checking
        output.write('\n')

        finaldatabase = sorteddatabase

        np.save('array.npy', finaldatabase)

    g += 1
```

```
# f.write(repr(database))
print('Done')
output.write('Done')
print(str(datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
timer.write(str(datetime.now().strftime('%Y-%m-%d %H:%M:%S')))
```

9 Appendix 2 - Source Code cleaning output

This Python script cleans the input file to produce a CSV file for processing and evaluation.

```
# This python program reads in the output file of the script above
# and cleans the formatting, outputting a CSV file. This file is used
# to evaluate the algorithm and graph results. This script is clumsy
# and could be refined - it does produce the required results.

import re

# Read in generated file
inputFile = open("input.txt")
contents = inputFile.read()
# remember to close the file
inputFile.close()

#print(contents)

# clean white space on both sides
content1 = contents.replace('[[  ',',')
content2 = content1.replace(' ',',')
content3 = content2.replace(']]',',')
content4 = content3.replace(']',',')
content5 = content4.replace('[',',')
content6 = content5.replace(',,,,,',',')
content7 = content6.replace(',,,,',',')
content8 = content7.replace(',,,',',')
content9 = content8.replace(',,',',')
#content6 = re.sub('],',',', content4)

# Could clean white spaces if software requires it
#content6 = re.sub(' ', ',', content5)

content = content9
# print(content)

# Write out clean file
```

```
outputFile = open("output.csv", "w")
outputFile.write(content)
# remember to close the file
outputFile.close()
```

10 Appendix 3 - Solar calculations (every 15s)

Cape Town	22-Dec				
Time stamp	Azimuth	Solar Angle	Time stamp	Azimuth	Solar Angle
4am	no data	no data	0.45	1.91470725	117.2132095
5am			10	1.960767	117.179063
0	0.128775	118.558356	0.15	2.006883	117.1449893
0.15	0.17431	118.5235305	0.3	2.052999	117.1109155
0.3	0.219845	118.488705	0.45	2.099115	117.0768418
0.45	0.26538	118.4538795	11	2.145231	117.042768
1	0.310915	118.419054	0.15	2.19140325	117.0087658
0.15	0.35651	118.3843065	0.3	2.2375755	116.9747635
0.3	0.402105	118.349559	0.45	2.28374775	116.9407613
0.45	0.4477	118.3148115	12	2.32992	116.906759
2	0.493295	118.280064	0.15	2.37614775	116.8728283
0.15	0.5389495	118.2453938	0.3	2.4223755	116.8388975
0.3	0.584604	118.2107235	0.45	2.46860325	116.8049668
0.45	0.6302585	118.1760533	13	2.514831	116.771036
3	0.675913	118.141383	0.15	2.56111375	116.737176
0.15	0.7216265	118.1067893	0.3	2.6073965	116.703316
0.3	0.76734	118.0721955	0.45	2.65367925	116.669456
0.45	0.8130535	118.0376018	14	2.699962	116.635596
4	0.858767	118.003008	0.15	2.74629975	116.6018063
0.15	0.90453925	117.9684903	0.3	2.7926375	116.5680165
0.3	0.9503115	117.9339725	0.45	2.83897525	116.5342268
0.45	0.99608375	117.8994548	15	2.885313	116.500437
5	1.041856	117.864937	0.15	2.93170525	116.4667165
0.15	1.0876865	117.8304948	0.3	2.9780975	116.432996
0.3	1.133517	117.7960525	0.45	3.02448975	116.3992755
0.45	1.1793475	117.7616103	16	3.070882	116.365555
6	1.225178	117.727168	0.15	3.1173285	116.3319035
0.15	1.2710665	117.6928008	0.3	3.163775	116.298252
0.3	1.316955	117.6584335	0.45	3.2102215	116.2646005
0.45	1.3628435	117.6240663	17	3.256668	116.230949
7	1.408732	117.589699	0.15	3.303168	116.1973658
0.15	1.454678	117.5554058	0.3	3.349668	116.1637825
0.3	1.500624	117.5211125	0.45	3.396168	116.1301993
0.45	1.54657	117.4868193	18	3.442668	116.096616
8	1.592516	117.452526	0.15	3.4892215	116.0631005
0.15	1.638519	117.4183068	0.3	3.535775	116.029585
0.3	1.684522	117.3840875	0.45	3.5823285	115.9960695
0.45	1.730525	117.3498683	19	3.628882	115.962554
9	1.776528	117.315649	0.15	3.6754885	115.9291058
0.15	1.82258775	117.2815025	0.3	3.722095	115.8956575
0.3	1.8686475	117.247356	0.45	3.7687015	115.8622093

Cape Town	22-Dec				
Time stamp	Azimuth	Solar Angle	Time stamp	Azimuth	Solar Angle
20	3.815308	115.828761	7	5.69092	114.505083
0.15	3.86196725	115.7953793	0.15	5.7380865	114.4723345
0.3	3.9086265	115.7619975	0.3	5.785253	114.439586
0.45	3.95528575	115.7286158	0.45	5.8324195	114.4068375
21	4.001945	115.695234	8	5.879586	114.374089
0.15	4.0486565	115.6619183	0.15	5.92680125	114.3414005
0.3	4.095368	115.6286025	0.3	5.9740165	114.308712
0.45	4.1420795	115.5952868	0.45	6.02123175	114.2760235
22	4.188791	115.561971	9	6.068447	114.243335
0.15	4.23555425	115.5287205	0.15	6.11571075	114.210706
0.3	4.2823175	115.49547	0.3	6.1629745	114.178077
0.45	4.32908075	115.4622195	0.45	6.21023825	114.145448
6am			10	6.257502	114.112819
0	4.375844	115.428969	0.15	6.304814	114.080249
0.15	4.422659	115.3957835	0.3	6.352126	114.047679
0.3	4.469474	115.362598	0.45	6.399438	114.015109
0.45	4.516289	115.3294125	11	6.44675	113.982539
1	4.563104	115.296227	0.15	6.49410975	113.9500273
0.15	4.60997025	115.2631055	0.3	6.5414695	113.9175155
0.3	4.6568365	115.229984	0.45	6.58882925	113.8850038
0.45	4.70370275	115.1968625	12	6.636189	113.852492
2	4.750569	115.163741	0.15	6.68359625	113.820038
0.15	4.79748625	115.1306833	0.3	6.7310035	113.787584
0.3	4.8444035	115.0976255	0.45	6.77841075	113.75513
0.45	4.89132075	115.0645678	13	6.825818	113.722676
3	4.938238	115.03151	0.15	6.8732725	113.690279
0.15	4.98520575	114.9985153	0.3	6.920727	113.657882
0.3	5.0321735	114.9655205	0.45	6.9681815	113.625485
0.45	5.07914125	114.9325258	14	7.015636	113.593088
4	5.126109	114.899531	0.15	7.06313725	113.5607478
0.15	5.17312675	114.8665988	0.3	7.1106385	113.5284075
0.3	5.2201445	114.8336665	0.45	7.15813975	113.4960673
0.45	5.26716225	114.8007343	15	7.205641	113.463727
5	5.31418	114.767802	0.15	7.25318875	113.4314428
0.15	5.36124775	114.7349315	0.3	7.3007365	113.3991585
0.3	5.4083155	114.702061	0.45	7.34828425	113.3668743
0.45	5.45538325	114.6691905	16	7.395832	113.33459
6	5.502451	114.63632	0.15	7.44342625	113.3023613
0.15	5.54956825	114.6035108	0.3	7.4910205	113.2701325
0.3	5.5966855	114.5707015	0.45	7.53861475	113.2379038
0.45	5.64380275	114.5378923	17	7.586209	113.205675

Cape Town	22-Dec				
Time stamp	Azimuth	Solar Angle	Time stamp	Azimuth	Solar Angle
0.15	7.633849	113.1735008	0.3	9.5960245	111.8648735
0.3	7.681489	113.1413265	0.45	9.64410625	111.8332153
0.45	7.729129	113.1091523	28	9.692188	111.801557
18	7.776769	113.076978	0.15	9.74031225	111.769947
0.15	7.8244545	113.0448583	0.3	9.7884365	111.738337
0.3	7.87214	113.0127385	0.45	9.83656075	111.706727
0.45	7.9198255	112.9806188	29	9.884685	111.675117
19	7.967511	112.948499	0.15	9.9328515	111.643555
0.15	8.01524175	112.9164328	0.3	9.981018	111.611993
0.3	8.0629725	112.8843665	0.45	10.0291845	111.580431
0.45	8.11070325	112.8523003	30	10.077351	111.548869
20	8.158434	112.820234	0.15	10.12555925	111.517354
0.15	8.20620975	112.7882208	0.3	10.1737675	111.485839
0.3	8.2539855	112.7562075	0.45	10.22197575	111.454324
0.45	8.30176125	112.7241943	31	10.270184	111.422809
21	8.349537	112.692181	0.15	10.318434	111.3913408
0.15	8.3973575	112.6602203	0.3	10.366684	111.3598725
0.3	8.445178	112.6282595	0.45	10.414934	111.3284043
0.45	8.4929985	112.5962988	32	10.463184	111.296936
22	8.540819	112.564338	0.15	10.51147525	111.2655135
0.15	8.58868375	112.5324293	0.3	10.5597665	111.234091
0.3	8.6365485	112.5005205	0.45	10.60805775	111.2026685
0.45	8.68441325	112.4686118	33	10.656349	111.171246
23	8.732278	112.436703	0.15	10.70468125	111.139869
0.15	8.78018675	112.4048455	0.3	10.7530135	111.108492
0.3	8.8280955	112.372988	0.45	10.80134575	111.077115
0.45	8.87600425	112.3411305	34	10.849678	111.045738
24	8.923913	112.309273	0.15	10.898051	111.014406
0.15	8.97186525	112.277466	0.3	10.946424	110.983074
0.3	9.0198175	112.245659	0.45	10.994797	110.951742
0.45	9.06776975	112.213852	35	11.04317	110.92041
25	9.115722	112.182045	0.15	11.09158325	110.889122
0.15	9.163718	112.1502885	0.3	11.1399965	110.857834
0.3	9.211714	112.118532	0.45	11.18840975	110.826546
0.45	9.25971	112.0867755	36	11.236823	110.795258
26	9.307706	112.055019	0.15	11.28527675	110.764014
0.15	9.35574475	112.0233118	0.3	11.3337305	110.73277
0.3	9.4037835	111.9916045	0.45	11.38218425	110.701526
0.45	9.45182225	111.9598973	37	11.430638	110.670282
27	9.499861	111.92819	0.15	11.47913125	110.6390808
0.15	9.54794275	111.8965318	0.3	11.5276245	110.6078795

Amsterdam	21-June				
Time stamp	Azimuth	Solar Angle	Time stamp	Azimuth	Solar Angle
3am	no data	no data	0.45	1.23378975	51.4918475
4am			10	1.263673	51.540791
0	0.08324	49.574323	0.15	1.29363725	51.58968725
0.15	0.1123735	49.62370625	0.3	1.3236015	51.6385835
0.3	0.141507	49.6730895	0.45	1.35356575	51.68747975
0.45	0.1706405	49.72247275	11	1.38353	51.736376
1	0.199774	49.771856	0.15	1.41357525	51.785225
0.15	0.22899225	49.821189	0.3	1.4436205	51.834074
0.3	0.2582105	49.870522	0.45	1.47366575	51.882923
0.45	0.28742875	49.919855	12	1.503711	51.931772
2	0.316647	49.969188	0.15	1.53383625	51.98057425
0.15	0.34595	50.01847175	0.3	1.5639615	52.0293765
0.3	0.375253	50.0677555	0.45	1.59408675	52.07817875
0.45	0.404556	50.11703925	13	1.624212	52.126981
3	0.433859	50.166323	0.15	1.65441725	52.17573675
0.15	0.463246	50.215557	0.3	1.6846225	52.2244925
0.3	0.492633	50.264791	0.45	1.71482775	52.27324825
0.45	0.52202	50.314025	14	1.745033	52.322004
4	0.551407	50.363259	0.15	1.77531775	52.3707135
0.15	0.580878	50.412444	0.3	1.8056025	52.419423
0.3	0.610349	50.461629	0.45	1.83588725	52.4681325
0.45	0.63982	50.510814	15	1.866172	52.516842
5	0.669291	50.559999	0.15	1.896536	52.5655055
0.15	0.69884525	50.60913525	0.3	1.9269	52.614169
0.3	0.7283995	50.6582715	0.45	1.957264	52.6628325
0.45	0.75795375	50.70740775	16	1.987628	52.711496
6	0.787508	50.756544	0.15	2.01807025	52.760114
0.15	0.817145	50.8056315	0.3	2.0485125	52.808732
0.3	0.846782	50.854719	0.45	2.07895475	52.85735
0.45	0.876419	50.9038065	17	2.109397	52.905968
7	0.906056	50.952894	0.15	2.13991775	52.9545405
0.15	0.9357755	51.0019335	0.3	2.1704385	53.003113
0.3	0.965495	51.050973	0.45	2.20095925	53.0516855
0.45	0.9952145	51.1000125	18	2.23148	53.100258
8	1.024934	51.149052	0.15	2.26207825	53.14878575
0.15	1.0547355	51.19804325	0.3	2.2926765	53.1973135
0.3	1.084537	51.2470345	0.45	2.32327475	53.24584125
0.45	1.1143385	51.29602575	19	2.353873	53.294369
9	1.14414	51.345017	0.15	2.38454875	53.342852
0.15	1.17402325	51.3939605	0.3	2.4152245	53.391335
0.3	1.2039065	51.442904	0.45	2.44590025	53.439818

Amsterdam	21-June				
Time stamp	Azimuth	Solar Angle	Time stamp	Azimuth	Solar Angle
20	2.476576	53.488301	0.15	3.75175	55.4660935
0.15	2.50732875	53.5367395	0.3	3.783249	55.514107
0.3	2.5380815	53.585178	0.45	3.814748	55.5621205
0.45	2.56883425	53.6336165	31	3.846247	55.610134
21	2.599587	53.682055	0.15	3.877818	55.65810675
0.15	2.63041625	53.73044975	0.3	3.909389	55.7060795
0.3	2.6612455	53.7788445	0.45	3.94096	55.75405225
0.45	2.69207475	53.82723925	32	3.972531	55.802025
22	2.722904	53.875634	0.15	4.004174	55.8499575
0.15	2.75380925	53.92398475	0.3	4.035817	55.89789
0.3	2.7847145	53.9723355	0.45	4.06746	55.9458225
0.45	2.81561975	54.02068625	**5am**		
23	2.846525	54.069037	0	4.099103	55.993755
0.15	2.87750575	54.1173445	0.15	4.1308175	56.0416475
0.3	2.9084865	54.165652	0.3	4.162532	56.08954
0.45	2.93946725	54.2139595	0.45	4.1942465	56.1374325
24	2.970448	54.262267	1	4.225961	56.185325
0.15	3.00150425	54.3105315	0.15	4.25774675	56.233178
0.3	3.0325605	54.358796	0.3	4.2895325	56.281031
0.45	3.06361675	54.4070605	0.45	4.32131825	56.328884
25	3.094673	54.455325	2	4.353104	56.376737
0.15	3.125804	54.50354675	0.15	4.38496025	56.424551
0.3	3.156935	54.5517685	0.3	4.4168165	56.472365
0.45	3.188066	54.59999025	0.45	4.44867275	56.520179
26	3.219197	54.648212	3	4.480529	56.567993
0.15	3.25040225	54.6963915	0.15	4.5124555	56.615768
0.3	3.2816075	54.744571	0.3	4.544382	56.663543
0.45	3.31281275	54.7927505	0.45	4.5763085	56.711318
27	3.344018	54.84093	4	4.608235	56.759093
0.15	3.37529725	54.8890675	0.15	4.64023125	56.8068295
0.3	3.4065765	54.937205	0.3	4.6722275	56.854566
0.45	3.43785575	54.9853425	0.45	4.70422375	56.9023025
28	3.469135	55.03348	5	4.73622	56.950039
0.15	3.500488	55.08157575	0.15	4.76828575	56.9977375
0.3	3.531841	55.1296715	0.3	4.8003515	57.045436
0.45	3.563194	55.17776725	0.45	4.83241725	57.0931345
29	3.594547	55.225863	6	4.864483	57.140833
0.15	3.625973	55.27391725	0.15	4.89661775	57.18849375
0.3	3.657399	55.3219715	0.3	4.9287525	57.2361545
0.45	3.688825	55.37002575	0.45	4.96088725	57.28381525
30	3.720251	55.41808	7	4.993022	57.331476

Amsterdam	21-June				
Time stamp	Azimuth	Solar Angle	Time stamp	Azimuth	Solar Angle
0.15	5.0252255	57.37909925	0.3	6.3589725	59.324491
0.3	5.057429	57.4267225	0.45	6.39183975	59.3717625
0.45	5.0896325	57.47434575	18	6.424707	59.419034
8	5.121836	57.521969	0.15	6.457638	59.4662725
0.15	5.1541075	57.5695555	0.3	6.490569	59.513511
0.3	5.186379	57.617142	0.45	6.5235	59.5607495
0.45	5.2186505	57.6647285	19	6.556431	59.607988
9	5.250922	57.712315	0.15	6.589426	59.655194
0.15	5.2832615	57.759865	0.3	6.622421	59.7024
0.3	5.315601	57.807415	0.45	6.655416	59.749606
0.45	5.3479405	57.854965	20	6.688411	59.796812
10	5.38028	57.902515	0.15	6.72146925	59.84398575
0.15	5.41268675	57.95002875	0.3	6.7545275	59.8911595
0.3	5.4450935	57.9975425	0.45	6.78758575	59.93833325
0.45	5.47750025	58.04505625	21	6.820644	59.985507
11	5.509907	58.09257	0.15	6.85376525	60.032649
0.15	5.54238075	58.140048	0.3	6.8868865	60.079791
0.3	5.5748545	58.187526	0.45	6.92000775	60.126933
0.45	5.60732825	58.235004	22	6.953129	60.174075
12	5.639802	58.282482	0.15	6.98631275	60.2211855
0.15	5.67234225	58.3299245	0.3	7.0194965	60.268296
0.3	5.7048825	58.377367	0.45	7.05268025	60.3154065
0.45	5.73742275	58.4248095	23	7.085864	60.362517
13	5.769963	58.472252	0.15	7.11910975	60.40959675
0.15	5.8025695	58.5196595	0.3	7.1523555	60.4566765
0.3	5.835176	58.567067	0.45	7.18560125	60.50375625
0.45	5.8677825	58.6144745	24	7.218847	60.550836
14	5.900389	58.661882	0.15	7.25215475	60.59788525
0.15	5.93306125	58.709255	0.3	7.2854625	60.6449345
0.3	5.9657335	58.756628	0.45	7.31877025	60.69198375
0.45	5.99840575	58.804001	25	7.352078	60.739033
15	6.031078	58.851374	0.15	7.385447	60.78605225
0.15	6.0638155	58.89871275	0.3	7.418816	60.8330715
0.3	6.096553	58.9460515	0.45	7.452185	60.88009075
0.45	6.1292905	58.99339025	26	7.485554	60.92711
16	6.162028	59.040729	0.15	7.51898375	60.9740995
0.15	6.1948305	59.08803375	0.3	7.5524135	61.021089
0.3	6.227633	59.1353385	0.45	7.58584325	61.0680785
0.45	6.2604355	59.18264325	27	7.619273	61.115068
17	6.293238	59.229948	0.15	7.6527635	61.16202825
0.15	6.32610525	59.2772195	0.3	7.686254	61.2089885

11 Appendix 4 - Individual fitness details

Table 3 indicates a sample from the initial population for the Cape Town West façade 6th run while Table 4 indicates the same sample at the final generation. Both Tables show a small selection of the individuals rather than the whole population (which numbers several thousand).

Fitness	X	Y	Z	X self-ad.	Y self-ad.	Z self-ad.	Indiv.	Parent	Gen.
804.	1030.	2041.	603.	0.989	0.998	1.167	19999.	0.	0.
804.	347.	2042.	296.	1.11	0.78	1.23	8606.	0.	0.
804.	1288.	2038.	817.	1.411	1.07	1.11	4845.	0.	0.
712.	879.	1141.	1282.	0.963	1.315	1.217	6426.	0.	0.
711.	514.	1135.	553.	0.713	1.084	1.079	17916.	0.	0.
711.	1476.	1325.	394.	1.004	1.282	0.587	14954.	0.	0.
6.	313.	1140.	1568.	0.841	1.026	1.121	11108.	0.	0.
5.	254.	1660.	1556.	0.96	0.686	1.426	18128.	0.	0.
5.	222.	1778.	1242.	0.703	1.044	0.778	8395.	0.	0.
5.	233.	967.	744.	0.931	1.113	0.881	11494.	0.	0.
5.	201.	2116.	1126.	0.735	0.593	0.691	12015.	0.	0.
5.	1511.	2276.	178.	1.323	1.273	0.829	19921.	0.	0.
5.	222.	2094.	1452.	0.97	0.837	0.851	14818.	0.	0.
5.	188.	1407.	618.	0.966	1.033	0.717	8585.	0.	0.
4.	213.	1441.	911.	0.848	0.954	1.037	15996.	0.	0.
4.	236.	747.	575.	1.16	1.186	0.961	19214.	0.	0.
4.	193.	2131.	1008.	0.705	1.297	0.906	17313.	0.	0.
3.	163.	1044.	261.	1.443	0.979	0.861	16900.	0.	0.
3.	209.	1720.	1042.	1.115	0.97	1.105	7980.	0.	0.
3.	356.	919.	1557.	1.029	0.952	0.781	5219.	0.	0.
2.	213.	2366.	1498.	0.977	1.178	0.833	16682.	0.	0.
2.	239.	1655.	1389.	0.708	0.856	1.106	6755.	0.	0.

Table 3: Fitness Ranked sample of individuals - 1st Gen, Cape Town West, 6th run (Author)

Fitness	X	Y	Z	X self-ad.	Y self-ad.	Z self-ad.	Indiv.	Parent	Gen.
804.	879.8	2036.9	936.8	0.42	-0.046	0.539	9.	1999.	99.
804.	851.8	2035.9	547.9	-0.	0.005	0.133	8.	669.	99.
804.	851.8	2035.9	547.6	-0.	0.005	-0.149	0.	669.	99.
804.	851.8	2035.9	547.9	-0.	0.006	0.119	1.	669.	99.
804.	851.8	2035.9	547.6	0.	-0.005	-0.191	2.	669.	99.
804.	851.8	2035.9	548.1	-0.	0.003	0.361	3.	669.	99.
804.	851.8	2035.9	548.0	-0.	0.003	0.27	4.	669.	99.
804.	851.8	2035.9	547.8	0.	-0.004	0.076	5.	669.	99.
804.	851.8	2035.9	547.6	0.	0.006	-0.124	6.	669.	99.
804.	851.8	2035.9	547.8	-0.	-0.002	0.079	7.	669.	99.
804.	851.8	2035.9	547.5	-0.	-0.003	-0.267	9.	669.	99.
804.	849.7	2036.5	726.6	0.51	0.001	-7.978	8.	670.	99.
804.	849.2	2036.5	734.6	-0.44	-0.	11.941	9.	977.	98.
804.	848.9	2036.5	746.1	-0.23	-0.	11.477	0.	670.	99.
804.	849.6	2036.5	747.9	0.46	0.001	13.325	1.	670.	99.
804.	849.5	2036.5	719.1	0.29	0.	-15.468	2.	670.	99.
804.	848.9	2036.5	745.6	-0.27	-0.001	11.034	3.	670.	99.
804.	849.9	2036.5	720.0	0.71	-0.	-14.613	4.	670.	99.
804.	849.6	2036.5	721.1	0.48	0.	-13.501	5.	670.	99.
804.	848.7	2036.5	743.9	-0.41	0.	9.343	6.	670.	99.
804.	851.8	2035.9	547.8	0.	0.005	0.17	6.	568.	98.
804.	476.6	2039.6	205.3	-0.25	-0.001	0.	9.	668.	99.
804.	476.7	2039.6	205.3	-0.15	0.001	0.	8.	668.	99.
804.	476.6	2039.6	205.3	-0.26	0.002	0.	7.	668.	99.
804.	940.4	2043.0	1077.0	-11.42	0.001	-102.229	1.	667.	99.

Table 4: Fitness Ranked sample of individuals - Final Gen, Cape Town West, 6th run (Author)

12 Appendix 5 - Fitness values per generation per ES run

One hundred generations were tested for fitness across ten separate run cycles. For Amsterdam East a total of 1 984 sun rays could be blocked, this would achieve full fitness, but this amount would be unlikely for points making up the point cloud of a sun-shade. Instead, as Table 5 shows, a maximum fitness of 1 111 was achieved. This represents a normalised value of 0,56. This maximum fitness was achieved by the tenth generation and maintained until the final one hundredth generation.

AMST. E											
Full Fit.	1984										
Gen.	0	1	2	3	4	5	6	7	8	9	10
Norm.	0,22	0,44	0,52	0,54	0,55	0,56	0,56	0,56	0,56	0,56	0,56
Mean	435	876	1023	1075	1096	1104	1108	1110	1110	1111	1111
Run 1	431	880	1030	1080	1097	1104	1108	1109	1110	1110	1111
Run 2	437	874	1020	1073	1093	1101	1106	1108	1109	1110	1111
Run 3	438	877	1028	1078	1097	1105	1108	1110	1110	1111	1111
Run 4	439	879	1022	1075	1096	1103	1107	1109	1110	1111	1111
Run 5	428	863	1011	1065	1091	1102	1107	1109	1110	1111	1111
Run 6	437	879	1023	1074	1095	1104	1108	1110	1111	1111	1111
Run 7	433	876	1027	1078	1097	1105	1108	1109	1110	1111	1111
Run 8	438	881	1028	1078	1097	1104	1108	1110	1110	1111	1111
Run 9	432	870	1018	1073	1096	1105	1109	1111	1111	1111	1111
Run 10	437	883	1026	1077	1098	1105	1109	1110	1111	1111	1111

Table 5: Amsterdam East - Mean Fitness value per generation.

Table 6 indicates the results achieved for the Amsterdam West condition where 1 985 sun rays represents complete fitness. A maximum of 1 113 sun rays were blocked, a normalised value of 0,56. This was accomplished by the thirteenth generation and maintained until the final one hundredth generation.

AMST. W											
Full Fit.	1985										
Gen.	0	1	2	3	4	5	6	7	8	9	10
Norm.	0,22	0,44	0,51	0,54	0,5524	0,56	0,56	0,56	0,56	0,56	0,56
Mean Fit.	436	875	1021	1075	1096,6	1106	1110	1111	1112	1112	1112
Run 1	437	880	1020	1077	1099	1108	1111	1112	1112	1112	1112
Run 2	437	882	1022	1075	1097	1105	1109	1111	1112	1112	1112
Run 3	438	876	1024	1076	1097	1106	1109	1111	1112	1112	1112
Run 4	437	876	1024	1078	1096	1105	1109	1111	1112	1112	1113
Run 5	435	871	1017	1069	1093	1102	1107	1110	1111	1112	1112
Run 6	436	870	1014	1070	1095	1106	1111	1112	1112	1112	1113
Run 7	436	883	1029	1077	1098	1106	1110	1111	1112	1112	1113
Run 8	436	870	1020	1075	1096	1106	1110	1111	1112	1112	1112
Run 9	433	868	1022	1077	1097	1106	1110	1112	1112	1112	1112
Run 10	431	873	1022	1076	1098	1107	1111	1112	1112	1112	1112

Table 6: Amsterdam West - Mean Fitness value per generation (first 10 generations shown).

Cape Town saw less sun rays being blocked as a result of a shorter summer solstice. As table 7 indicates, full fitness required 1 710 sun rays to be blocked for the East façade. The highest fitness evolved blocked 803 sun rays - a normalised value of 0,47. This was accomplished by the eighth generation and maintained until the one hundredth generation.

CT EAST										
Full Fitness	1710									
Generation	0	1	2	3	4	5	6	7	8	100
Norm. Fitness	0,19	0,4	0,45	0,46	0,467	0,47	0,47	0,47	0,47	0,47
Mean fitness	324,8	689	773	791	797,9	801	802	803	803	803
Run 1	321	681	772	791	797	800	802	802	803	803
Run 2	328	688	771	790	797	801	802	803	803	803
Run 3	328	693	774	792	799	801	802	803	803	803
Run 4	324	692	774	792	798	801	802	802	803	803
Run 5	326	694	774	791	798	801	802	803	803	803
Run 6	320	681	771	791	798	801	802	803	803	803
Run 7	325	683	772	791	798	801	802	803	803	803
Run 8	325	688	773	792	798	801	802	802	803	803
Run 9	324	691	773	791	797	801	802	803	803	803
Run 10	327	696	777	793	799	801	802	803	803	803

Table 7: Cape Town East - Mean Fitness value per generation across 10 runs.

Cape Town West required a total of 1 711 sun rays to be blocked to achieve full fitness. As Table 8 indicates, the best achieved fitness value was 804 sun rays blocked for a normalised value of 0,47. This was achieved by the eighth generation and maintained until the one hundredth generation.

CT WEST										
Full Fitness	1711									
Generation	0	1	2	3	4	5	6	7	8	100
Norm. Fitness	0,19	0,40	0,45	0,46	0,47	0,47	0,47	0,47	0,47	0,47
Mean fitness	324,5	689	774	792	798,7	802	803	803	804	804
Run 1	324	687	773	792	799	802	803	803	804	804
Run 2	326	695	775	792	799	802	803	803	804	804
Run 3	327	688	771	791	798	801	803	803	804	804
Run 4	325	692	775	792	798	802	803	803	804	804
Run 5	321	689	774	792	799	802	803	804	804	804
Run 6	324	687	774	792	799	802	803	803	804	804
Run 7	324	687	774	793	799	802	803	803	804	804
Run 8	326	699	776	792	799	802	803	803	804	804
Run 9	322	679	771	792	799	802	803	804	804	804
Run 10	326	691	772	792	798	801	803	804	804	804

Table 8: Cape Town West - Mean Fitness value per generation across 10 runs.

13 Appendix 6 - Fitness per façade

The following tables indicate the comparative fitness values between the two conditions in the case study. Tables 9 and 10 outline the values found for the East façade while Tables 11 and 12 outline the values for the West façade. Both Southern Hemisphere façade's have lower fitness values but are similar in total, while the Northern Hemisphere values are generally higher - this is connected to both a longer period of sun being experienced at the summer solstice and a larger spread of solar values (i.e. solar angle - the compass value).

		Amsterdam East	
run	Normalised Fitness	Mean Fitness	Full Fitness
1	0.56	1111.00	1984.00
2	0.56	1111.00	1984.00
3	0.56	1111.00	1984.00
4	0.56	1111.00	1984.00
5	0.56	1111.00	1984.00
6	0.56	1111.00	1984.00
7	0.56	1111.00	1984.00
8	0.56	1111.00	1984.00
9	0.56	1111.00	1984.00
10	0.56	1111.00	1984.00
	0.56	MEAN	
	0.00	Standard Deviation	

Table 9: Amsterdam East façade - comparison between façade fitness values.

Cape Town East

run	Normalised Fitness	Mean Fitness	Full Fitness
1	0.47	803.00	1710.00
2	0.47	803.00	1710.00
3	0.47	803.00	1710.00
4	0.47	803.00	1710.00
5	0.47	803.00	1710.00
6	0.47	803.00	1710.00
7	0.47	803.00	1710.00
8	0.47	803.00	1710.00
9	0.47	803.00	1710.00
10	0.47	803.00	1710.00
	0.47	MEAN	
	0.00	Standard Deviation	

Table 10: Cape Town East façade - comparison between façade fitness values.

		Amsterdam West	
run	Normalised Fitness	Mean Fitness	Full Fitness
1	0.56	1113.00	1985.00
2	0.56	1113.00	1985.00
3	0.56	1113.00	1985.00
4	0.56	1113.00	1985.00
5	0.56	1113.00	1985.00
6	0.56	1113.00	1985.00
7	0.56	1113.00	1985.00
8	0.56	1113.00	1985.00
9	0.56	1113.00	1985.00
10	0.56	1113.00	1985.00
	0.56	MEAN	
	0.00	Standard Deviation	

Table 11: Amsterdam West façade - comparison between façade fitness values.

		Cape Town West	
run	Normalised Fitness	Mean Fitness	Full Fitness
1	0.47	804.00	1711.00
2	0.47	804.00	1711.00
3	0.47	804.00	1711.00
4	0.47	804.00	1711.00
5	0.47	804.00	1711.00
6	0.47	804.00	1711.00
7	0.47	804.00	1711.00
8	0.47	804.00	1711.00
9	0.47	804.00	1711.00
10	0.47	804.00	1711.00
	0.47	MEAN	
	0.00	Standard Deviation	

Table 12: Cape Town West façade - comparison between façade fitness values.