RESPONSIVE ALGORITHMS

An investigation of computational processes in early stage design

FRANO. BAZALO¹, TANE J. MOLETA²
¹,² Victoria University of Wellington, Wellington, New Zealand
{tane.moleta}@vuw.ac.nz

Abstract. An algorithm is a process of addressing a problem in a finite number of steps. In the context of architectural design, algorithmic thinking means taking on an interpretive role to understand the results in relation to design criteria, knowing how to modify the code to explore new options, and speculating on further design potentials. The application of algorithms within architecture often addresses the developed design stages, primarily to optimise structure, test environmental performance or to resolve complex construction. This research aims to explore algorithmic tools with a focus on early stage design. This design stage is often developed using traditional processes and is where algorithmic applications have been less successfully executed. The objectives are to algorithmically explore the areas of space planning, programme layout, form finding and form optimisation within early stage architectural design. Through the combination of a range of diverse algorithms, this research has an ultimate aim of integrating a computational workflow into practice at the early design stage.

Keywords. Computational design, Early stage design

1. Computation in design practice

Computational processes have been largely inducted into architectural practice as a drafting technique to simply digitise preconceived designs, not as a true partner in the process of conception (Picon, 2010). “Architects use the computer as a virtual drafting board making it easier to edit, copy and increase the precision of drawing” (Terzidis, 2006). However in order to reduce the time spent on the transition from the early design stage to more pre-
cise stages, increasingly more architects are starting to use CAD pro-
grammes in all stages with mixed results. As Aliakseyeu et al observe “The
down side of this practice is that the use of such precise programmes in the
early design stage tends to limit the creativity and can encourage poor design”
(Aliakseyeu, Martens, & Rauterberg, 2006).

The core of architectural thinking is still totally dependent upon the de-
signer’s intuition (Picon, 2010). Despite this, a review of current literature
reveals that increasingly architects are developing computational tools that
create opportunities in design process, fabrication and construction (Peters,
2013, p. 10). For the past 10 years, emerging computational tools and tech-
niques have had an increasing impact on architectural design (Agkathidis,
2012). Architects and students of architecture are experimenting with ways
to embed digital methods into the design process, exploring new possibilities
and challenges. We are moving rapidly from an era of being aspiring expert
users, to one of being adept digital toolmakers (Burry, 2011, p. 8). Firms
such as MOS and Facit Homes have fully integrated computation into prac-
tice and the actual design process. In these firms there is no separation be-
tween design intent and computational technique. Computation is used in a
natural or unconscious way (Peters, 2013, p. 11). Despite digital processes
increasingly being adopted extensively throughout practices, many architects
in practice still prefer to use paper and pen or scale models during the early
design stage. (Gross and Yi-Luen Do, 1996; Aliakseyeu, 2003). Use of the-
se forms of media offers the required flexibility, speed and natural (intuitive)
interaction (Aliakseyeu, Martens, & Rauterberg, 2006). This way of working,
however, has been cited by Aliakseyeu et el as creating an interruption in the
flow of the design process, since the architect has to transfer his/her design
into computer-aided design specifications after the early design stage.

2. Potential for Computation in Architecture

An algorithm is a process of addressing a problem in a finite number of steps.
It is an articulation of either a strategic plan for solving a known problem or
a stochastic search towards possible solutions to a partially known problem
(Terzidis, 2006). Theoretically, as long as a problem can be defined in logi-
cal terms, a solution may be produced that will address the problem’s de-
mands (Terzidis, 2006). Algorithmic thinking means taking on an interpret-
tive role to understand the results of the generating code, knowing how to
modify the code to explore new options, and speculating on further design
potentials (Peters, 2013). Leach observes, “many people have misgivings
about the term parametric” (Leach & Schumacher, 2012). Peters states much
of the industry’s ill-fated impression of parametric design has come from
both the increase of availability of these tools, paired with an uneducated understanding of the algorithmic processes behind the “black box” (Peters, 2013). Many critics blacklist “parametrically” designed buildings from the beginning as scripting often veers towards complexity of otherwise quite simple things (Burry M., 2011, p. 40). Views on the potential for computation to transform architectural design range extensively. My position aligns with that of Peters where computation can assist the architect in the design process, extending the designer’s abilities to deal with highly complex situations (Peters, 2013, p. 10). Ultimately, they could help change the way architectural design is undertaken in an increasingly complex, data-driven society. The potential lies in the possibility for factual parameters to be incorporated in the design process alongside the designer’s intuition and formal predisposition: parameters that deal with population growth, sustainability, site constraints and regulations, and other issues allowing the exploration of all possible solutions to given problems (Agkathidis, 2012).

2. Research Proposal

This research proposes that a computational design workflow provides designers with a greater capacity to explore designed outcomes. It argues complex design issues can be interpreted simultaneously while generating results that otherwise might not have been reached using traditional methods. Algorithmic design tools offer the power of computational iteration, which enables multiple design permutations to be undertaken in a short time span. Moreover, I propose that some of the problems associated with a computational approach to design, in particular the capacity to enable subjective input from the designer, can be addressed. The focus of this research is on the early stages of design. The aim is to explore how a combination of algorithms can be used for the simultaneous creation, analysis and optimisation of design ideas. The early design stage includes space planning, programme layout, form finding and form optimisation as defined in the literature reviews by Burry, Peters, Picon, and Terzidis.

3. Algorithmic Space Planning

Space planning is required in every architectural design process with varying levels of complexity. Quick bubble diagrams are often the method of choice in the preliminary stages (Karlen, 2009). They allow schematic visualisation of the program required while starting to establish adjacencies, relationships, connections and functions. In an effort to grant this traditionally manual process a level of algorithmic intuition, circle packing was analysed. This is the study of the arrangement of circles so that no overlapping occurs and the cir-
cles are packed as tightly as possible. Packing is referred to as the subdivision of space in which as little space as possible, or none, is left over. This experimental simulation process began by calculating the arrangement of a group of given circles based on their sizes. As the simulation ran, the circles were drawn to an anchor point and arranged themselves based on size and collisions with each other. With the particles arranged, Voronoi and metaball algorithms were combined to accurately divide each area into cells that equated to the size of the circle representing it. While this packed the circles as tightly as possible, there was no consideration to a boundary. To achieve this, a boundary curve was included in the simulation. The algorithm was relatively effective in packing the cells, however there was little control over how they were arranged spatially.

Figure 1. Circle Packing

4. Space Syntax

The circle packing physics simulations yielded interesting results, however in terms of optimising architectural programme, it had a complete inability to establish and maintain programmatic relationships. This is a key role in early stage design. As far as the algorithm was concerned, the solutions were optimised based on efficient packing, though it was far from optimised in terms of architectural planning. In order for the algorithm to organise architectural programme, it had to recognise relationships, connections and adjacencies between programmes.

Space Syntax theories were introduced to the circle-packing algorithm in order to unite simple space planning guidelines with the formal optimisation of an self-organising early stage design tool. Space Syntax is a set of theories and techniques designed to analyse spatial configurations. It was conceived in the late 1970’s by Bill Hillier and Julienne Hanson as a tool to help architects simulate the likely social effects of their designs (Hillier & Vaughan, 2007). Space Syntax’s intended use was an analytical tool (The
Bartlett, 2007) but paired with generatively defined algorithms, these theories have the potential to not only perform post analytical assessments but to define and optimise spatial configurations based on adjacency rules. By embedding simple rubrics as with manual space planning, the algorithm is capable of recognising primary and secondary connections in which it can aim to optimise with the help of the A* shortest path algorithm. For example: ‘a’ has to be connected to ‘b’, ‘b’ to ‘g’ and ‘x’ to ‘y’.

Experiments were conducted testing the instinctual capabilities and limits of these Space Syntax processes. Striving for an intuitive processing language, the algorithm developed a capability to understand simple rules, for example; a store requires an ATM directly adjacent or seating must be located next to a platform etc. In practice and in theory, these simple statements work for orthodox situations, however architecture is usually the contrary with complex programmatic relationships. To help counter this, a review and input stage was encrypted into the process. This permits the designer to make executive decisions that might be unique to the situation, outside of the algorithms programmed understanding. It was then possible to add or overwrite connections that are predetermined by the algorithm. This is key in allowing the designer to “design” and not the algorithm to merely determine.

As many briefs outline, there are often several extremely strict parameters while the rest of the programme are “free” to be organised according to the parameters set in the equation. By combining the results of Space Syntax algorithm with Kangaroo physics simulation, the graph could be adapted by the user with several anchor points. These anchor points can represent programme with specific location requirements. This manual input permits the Space Syntax diagram to be tweaked or altered towards a more usable form with a simple drag and drop of the mouse. Using the pre-established spring lengths, the graph remains in an optimised form while the overall composition can be manipulated.

Figure 2. Space syntax simulation
5. From two to three dimensions

As a tool intended for early stage architectural design, it was a necessity that self-organisation worked in three-dimensions. The crucial progression was to establish an effective process for spatialising these previously two-dimensional algorithmic processes. An example of three-dimensional circle packing is the highly intriguing interaction of bubbles: forming and optimising. This natural process of organisation and optimisation appears to share many parallels with some of the overarching goals I was hoping to explore in intuitive, algorithmic early stage design. Scientists have observed that when bubbles form clusters, their boundaries merge to form a new partition wall perpendicular to the loci of the bubble. Coincidently, this process is replicated in 3D Voronoi diagrams. More interestingly, when two or more bubbles meet, they proceed to adopt a shape which encases the original volume of air while reducing the sum of their surface areas as small as possible. This process of form-finding through preservation and optimisation provided the stimulus to explore these parallels. By drawing inspiration from these natural procedures, a dynamic process of optimisation, ordering and adaption was combined with an underlying logic of establishing 3D architectural programme and form for early stage design techniques. In the same way that bubbles interact with each other based on intuitive logic, rules needed to be embedded into the programme to allow the logic to optimise the layout interactively and dynamically. Where 3D Voronoi presents its flaws, is through its legibility of architectural space. The irregular dodecahedron-like forms present disobliging spaces, principally due to the lack of horizontal planes, in particular the ground planes. In an effort to target this spatial illegibility, a simple logic was developed to architecturally interpret these cells. By scheduling each cell to be sectioned at horizontal intervals, parallel floor planes are generated within the bounds of the Voronoi arrangement. This simple algorithmic process brings an inhabitable order to this otherwise chaotic mathematical structure.

Fig 3. Voronoi Cells sectioned to create floor plates
6. Optimising form

By combining the earlier physics simulations and organisational logic with three-dimensional Voronoi logic, a complex amalgamation of algorithms was explored using Space Syntax rubrics to tie together the processes. As this algorithm is intended to be a tool for early stage design, it was important not to restrict the form-finding process to a singular shape or aesthetic. It was an important aspect to remain flexible and dynamic in the same way the process is. For this reason, the form can be re-parameterised using spheres, cubes, cuboids, metaballs, Voronoi cells, or a combination, depending on the overall desired aesthetic and programme requirements. To unify this collective geometry, Bailey & Katzenstein’s “shrink wrap” definition was adapted in order to re-skin the algorithmic outcome. This algorithm runs simultaneously with the form finding process, encasing the internal geometry within a unified boundary. As a result, the external form is a direct representation of the programme it contains.

Due to pragmatic planning requirements, many architectural briefs require selected programme to be located specifically on the site. With these parameters in mind, the algorithm includes the ability for target points to be assigned to any programme with strict instructions, ensuring the simulation arranges the programme around these points. Any un-specified programme is free to optimise within the bounds of the Space-Syntax relationships and site. This provides a workflow pragmatically suited to the way in which early stage design is often undertaken. Additionally, an attractor spine or boundary can also be used to influence the spatial arrangement of programme in conjunction with any anchor points.

Figure 4. Envelope and Programatic relationships pre and post optimisation

The simultaneous calculation of algorithms result in an external envelope containing all internal programme. This presents an initial early stage design
which can be developed further with consideration into form, materials, light, construction, details etc. The ease and speed of this process provides the ability to explore many configurations and leaves more time for the pragmatic resolution of the design.

7. Algorithm Refinement

In order for these algorithms to be a useful tool for early stage design, the process and user interface had to be significantly refined. While the Grasshopper3D interface, in particular the sliders, are a great way of manipulating input parameters, they have their limits. The main concern is the complexity of the required parameters, often requiring multiple inputs to include data such as programme, size, names, relationships and levels. Each input type requires a unique input and this quickly becomes excessive and tedious to control.

As a method of refining the input of data, a spreadsheet was linked to the script. Within the spreadsheet, data is established and the seamless flow between Excel and Grasshopper result in a significantly more powerful and user-friendly tool. All programme, names, levels, sizes and relationships are established within the frontend spreadsheet interface while the algorithms are calculated in Grasshopper interpreting the data and work towards optimisation based on the previous form-finding procedures. While the spreadsheet streamlined how data is inputted, it was crucial to find the balance between an intuitive self-organising structure and the added input of the user to customise and form the final output. Following Piker’s analysis of physics simulations in architectural design, he notes that one great advantage of physically based methods is the natural feel we associate with real-world behaviours (Piker, 2013). This ability to intuitively interact with digital mass lends itself well to the design process and was important to include in this tool for early stage design. Instead of using points to attract the programme, as was used in the previous algorithm, boxes (or any geometric form) represent the programme. Any geometry that is manually moved pre-simulation remains static and provides anchor points for the simulation to work around. This provides the ability to input any empirical data or strict programmatic relationships while the remaining programme is optimised based on the given parameters. Additionally, boxes can be manually moved during simulation to influence and manipulate the structural composition. This constant feedback loop between algorithm and user provides the required intuitive and natural process required for early stage design.
5. Conclusion

This research was fundamentally concerned with the relationship of contemporary digital design tools and how they can be used to augment design outcomes for early stage design. Through a myriad of technical algorithmic studies, the usefulness of algorithmic processes in early stage design has been extensively explored. In response to the research question, the final algorithm has the ability to enhance the design workflow within early stage design by harnessing several aspects existing methods lack: The algorithm has a capacity to interpret raw data from multiple strings and process simultaneously. These parameters reach a complexity generally requiring the designer to conduct the development in chronological phases as each parameter is considered individually. What the amalgamation of algorithms has achieved, differentiating it from typical approaches is to simultaneously process data while an immediate visual response is generated. This combination permits a concurrent process of creation, analysis and optimisation of early stage design problems.

Refinement of the algorithm and the use of spreadsheet simplify the process and permit an ease of use, allowing extremely dynamic outputs to be generated extensively and rapidly. The refined algorithm grants an ability to achieve an extensive array of design options in a reduced timeframe. Differentiating between existing algorithmic self-organisation techniques, the method developed permits an empirical level of input from the designer, allowing outputs to be tweaked and enhanced based on the perspectives and opinions of the creator. This mediates an interesting, yet balanced juxtaposition between human and computer. Most intriguingly, computation has the potential to provide inspiration and go beyond the intellect of the designer through the generation of unexpected results (Peters B., 2007). The idea of sketching by algorithm is encapsulated in this research by the rubric nature
of transforming data into formal and spatial outputs. This unexpected, however reasoned, characteristic is what captures the potential for computational design tools in the early stage architectural design process.

Figure 6. Optimised form shown in context

Reference


