# Evolutionary parametric analysis for optimizing spatial adjacencies

**Christopher Boon[1], Corey T. Griffin[1], Nicholas Papaefthimiou[2], Jonah Ross[2], Kip Storey[2]**

[1]Portland State University, Portland, OR
[2]ZGF Architects, Portland, OR

ABSTRACT: This research utilizes parametric and evolutionary software in order to create an analytical system intended to enhance efficiency within groups of complexly interconnected architectural program elements. The authors have developed a parametric definition that can diagrammatically arrange spatial volumes. These volumes represent the various building functions or 'programs' in terms of square footage. The driving parameter for these experiments is adjacency; if two functions in a building need to be connected, they should be adjacent to one another. The degree of adjacency is here viewed as the distance between the centers of programmatic masses. The distance between programs is scored as a numerical value, the lower the value, the closer the programs, and the better the score. The resulting model contains a complete set of programs all arranged within a larger volumetric container. This container represents the building shell. Each arrangement represents an individual iteration. The script automatically creates individuals and store their information in groups referred to as 'generations'. The evolutionary process improves the proximity score over the course of many generations. The resulting spatial diagrams produced autonomously by the computer are sorted, and the computer determines fitness by minimizing the numerical value of the total distance of all interconnected programmatic elements. This type of approach offers a unique system of analysis that can create an incredibly extensive range of unique and otherwise unexplored solutions when faced with problems in developing complex order for spatial relationships. This script is a significant leap forward over existing research in this area with the ability for users to define an irregular shaped site boundary, input multiple stories, relate program elements to external adjacencies (views, parking, etc.) and handle an unlimited number of program elements. This paper uses a three-story hospital on a sloping site with fifty program elements to demonstrate the efficacy of this approach.

KEYWORDS: Programming, Evolution, Parametric, Spatial Adjacency

## INTRODUCTION

Emerging parametric technologies are opening new opportunities in architecture. Generally it is seen primarily as an engine to drive formal explorations and renderings. Its implications however are larger and it is possible to employ it at many stages in the design process. During the initial design stage, much of what is explored involves theoretical concepts. The work is expressed diagrammatically. If the concept can be distilled to its parameters, then it is possible to begin including parametric analysis. This type of analysis will allow designers to develop a much wider range of options in a much shorter timeframe. This study intends to explore the ability of evolutionary parametric modeling as a diagram-building tool. For a building to be efficient it must meet the needs of it occupants. The needs of the occupants are determined during the design process by a complex relationship of programmatic and physical situations. These situations can be measured and used as metric data; which can be translated into parameters.

## 1.0 PARAMETRIC AND EVOLUTIONARY DESIGN

### 1.1. Parametric design

Roller (1991) discusses the way specific information storage (parametric information) in combination with construction process has the potential to capture an extremely true version of design intent, while the benefits of iterative variation in the design process are examined. Motta (1999) focuses on computer aided, intelligent modelling systems, illustrating their widespread application and value as tools that have the potential for reuse. Precedents in parametric design highlight the importance of establishing a workflow that produces successful results in this field.

### 1.2. Evolutionary design

Lenski, et al. (1999) explores the ability of digital models to evolve like organisms in nature. Through experimentation, they attempt to find the best possible way to create a digital genome, and how influences that are inherent in nature, such as mutation, can affect the population. The authors use diagrams to help explain the methods that can achieve the fittest results.

### 1.3. Previous parametric tools for programming

Scrawford (2010) makes use of a detailed set of spreadsheets to create architectural geometry based on programmatic relationships. The tool arranges based on relationships marked within the information on the spreadsheets, and translated into individual spaces that organize themselves according to hierarchy. Definition is displayed to observe logic. The architecture firm NBBJ created a video of their parametric conceptual design tool for architectural practice that allows designers to quickly organize and understand complex architectural programs in three dimensions (Syp, 2010). The project uses spring physics to create gravity between the various elements, which are represented by spherical volumes.

### 1.4. Goals for a new optimization tool

This paper explores the utilization of Grasshopper (a Rhino Plugin) as an analytical and planning tool to graphically represent a 3-dimensional analysis of adjacency requirements in program and spaces. The purpose of this paper and research project is to explore and innovate new ways of further utilizing the technology in evolutionary algorithms or physics modelers to reduce inefficiencies in the design process as well as optimize floor-plan layouts. The tool developed in this research uses digital, generative tools to create spatial diagrams using adjacency requirements. The resulting tool can be used as an aid in the massing of buildings that have inherently complex programmatic relationships.

The ideal outcome will have the ability to reduce inefficiencies in the design process and optimize floor plan layouts in a single motion. The rapid iterative process takes ideas further in less time by letting the computer tirelessly work away at a problem that has a defined set of parameters created by the designer. This type of approach offers a unique system of analysis that can create an incredibly extensive range of unique and otherwise unexplored solutions when faced with problems in developing complex order for spatial relationships.

## 2.0 Methodology

### 2.1. Design directive

Programming for a building like a health care facility is an especially difficult job. This is due to the high level of complexity in spatial relationships and high number of programmatic elements. This is an ideal building program to test the tool on. A hospital also requires a high level of efficiency due to the nature of its function in society.

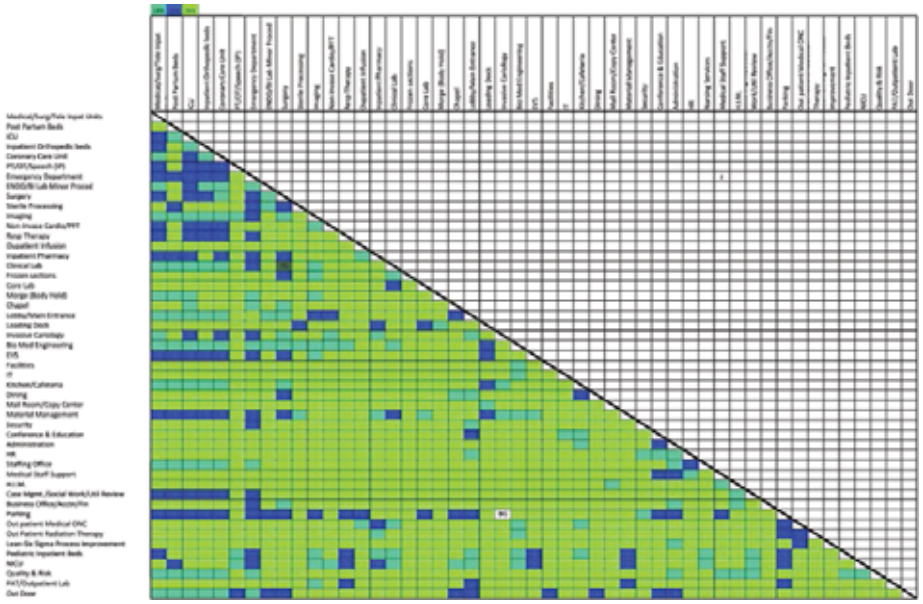

**Figure 1**: This matrix shows the various programmatic elements for a hospital and the relationship with others. There are three different levels of importance indicated by tones. Source: ZGF Architecture

ZGF has a great amount of experience designing hospitals. With expertise, one can rely on research and intuition to achieve efficiency in spatial relationships. At the beginning of this research, ZGF was in process of designing a hospital, and there were program adjacency charts generated at client meetings and these resources were made available as sample test figures. The numbers on the charts represent the various needs of the client. At meetings, the design team and clients develop a matrix of all the discrete programmatic elements. The matrix forms a hierarchy of space in terms of priority and ranks the adjacency requirement of each element in relation to all other elements in the chart. A space like the emergency room would take priority over the gift shop for example, and the two have no adjacency need.

## 2.2. Tool development

The definition begins by input of some simple geometric parameters, thereby establishing the programmatic volumes that will be used in the iterative process. These first steps are the customizable portions of the tool. Each unique architectural situation requires some unique combination of factors that this research attempts to address to some extent in terms of spatial proxemics. The potential floor plan space is all that is actually drawn by the user. The ground floor of the building or site outline is first drawn and then additional floors can be added of any shape. Floor to ceiling height is adjustable. Any number of floors is allowed and there can be multiple sites separated from one another. Any shape can be used to describe the boundary conditions. The tool will automatically fill in the shapes with a three dimensional, rectilinear grid that can also be adjusted as desired.

The geometry all occurs within a three dimensional grid. This begins to describe a structural system. All of the cells in the grid are marked with a single point. These points are all contained in a cloud. Each programmatic element is also built around a single point. The programmatic central points are here referred to as 'origins'. Within the list of programs itself there exists an explicit hierarchy. That is to say, the first program on the list dominates the second, which dominates the third and so on. The first program is placed in the grid by selecting one of the points in the cloud. The selected point becomes the center of the mass, which forms as a pixelated sphere around that point. Once a program occupies a space, all points within it are subtracted from the cloud. The next program repeats the process until all the programs are massed out. The proximity is determined by measuring the distance between origin points. The relationships are weighted in terms of their needs by multiplying the distances by various factors, the higher priority the adjacency need, the higher the multiplication factor. The tool calculates the total value of all relationships by performing mass addition. This figure is the fitness score. The evolutionary process tries to minimize this number as much as possible. The idea is that a lower score creates a closer proximity and a more effectively adjacent situation.

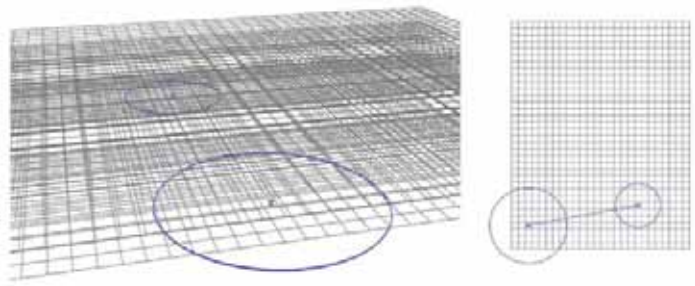

**Figure 2**: Screenshots showing the origin points selected in the grid and the relative floor area of each. Source: Author

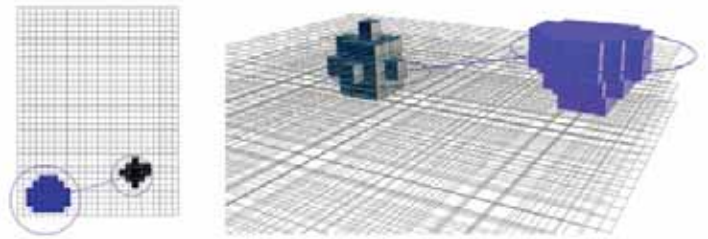

**Figure 3**: Areas are translated into "pixelated spheres". Source: Author

To create the evolutionary element in the process, the computer is allowed to manipulate all the origin points to create iterations. The point cloud within the shell is a list, and Galapagos can iterate using the sliders connected to that list.

## 2.3. Tool demonstration and steps

A series of generic tests help to demonstrate how the system works and where it needs refinement. This test was run with nine programmatic elements. Those are broken into three groups represented by three differently colored groups. Each group member wants to be adjacent to the others in the same group. The site outline for this experiment is for square blocks. There are four floors to be occupied. The Grasshopper definition (Figure 4) is implemented using the steps outlined in Figure 5 and described in more detail below.

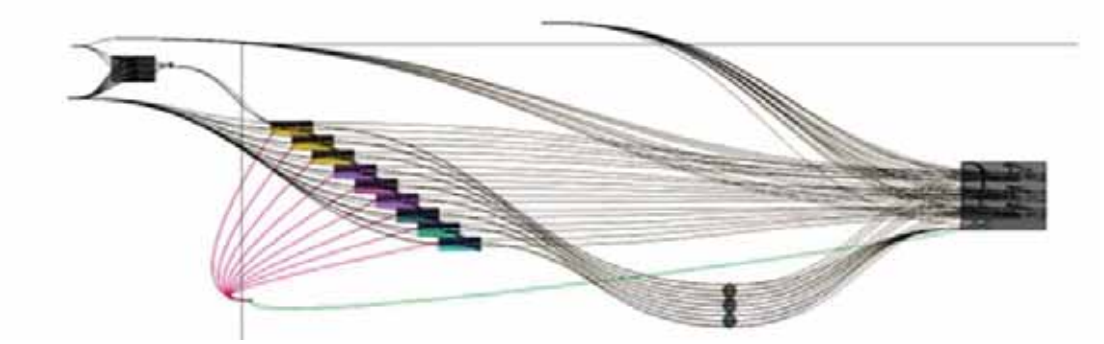

**Figure 4**: Grasshopper definition, screenshot colors represent program groupings. Source: Author

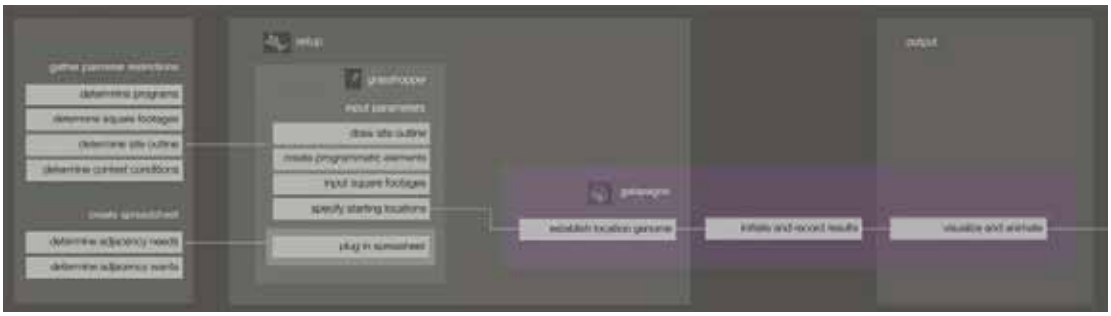


**Figure 5**: Diagram outlining the software programs and steps used to generate adjacency diagrams. Source: Author

*Step 1 – Establish site-specific initial geometry and offset levels*
The starting parameters are customizable. The user sets up a grid that the spaces are built within. The number of cells and size of cells can be set to a specific coarseness. The program can also build the grid inside of a two-dimensional site outline to accommodate non-rectilinear sites. To account for multiple levels the grid can be offset vertically any number of times and the floor-to-floor height is adjustable (Figure 6).

**Figure 6**: Point cloud formation, derived from initial geometry: site outlines and offset floors. Source: Author

*Step 2 – Assign hierarchy of spaces with list order*
When the program activates, the order of the list of program elements is the order of importance. The first element in the list is ranked as the most important. This hierarchy is applied when spaces are divided and reassigned giving priority to the most important program elements.

*Step 3 – Input square footage and set initial locations for each program element*
The square footage for each space is set using a slider in the Grasshopper definition. The various program elements are assigned an initial origin point in the grid for a starting location. This point is center of the pixelated sphere, which represents a program element in the definition.

*Step 4 – Generate adjacency lines between program elements*
The definition creates a line connecting the origin points for every adjacency requirement. A lower value indicates a closer proximity and more optimized adjacency. The definition adds up the total distance of all adjacency lines, which can be weighted to create greater attraction or repulsion in order to prioritize certain adjacencies over others or ensure two program elements are isolated from one another. The Galapagos evolutionary solver will try to minimize this total distance to improve fitness.

*Step 5 – Run Galapagos and record visualizations of every iteration*
The Galapagos “genome” is solely made of the origin location sliders, which allows it to rearrange the location of the program elements within the grid in order to improve fitness by creating the lowest total

adjacency line distance (Figure 7). The areas inside the program circles are translated into pixels on the grid and extruded into three-dimensional volumes by the Grasshopper definition. All iterations are recorded in sequence along with their fitness score. The results (Figure 8) can be exported as an animation.


**Figure 7**: Graph depicts fitness variation(y) over time and generation(x). Source: Author

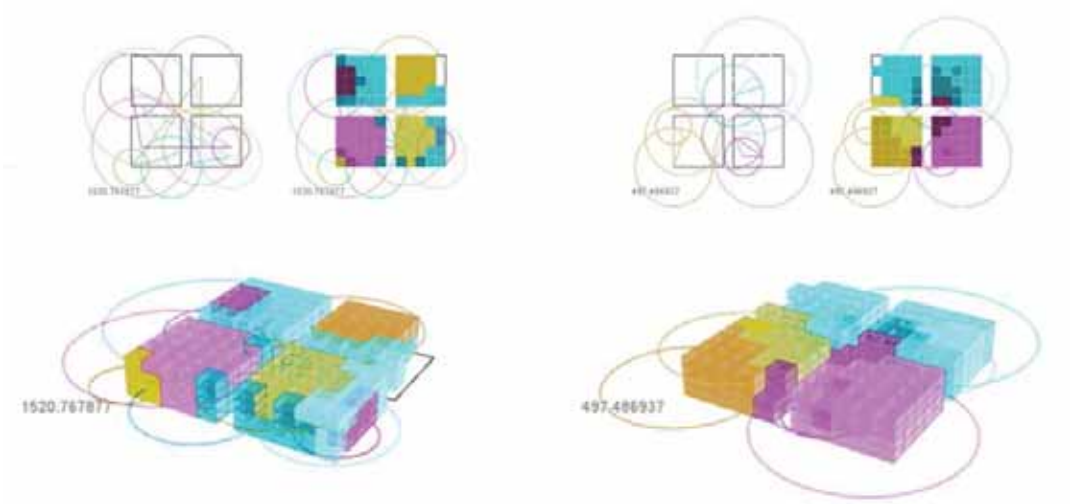


**Figure 8**: Least fit result (left) and most fir result (right). Source: Author

## 3.0. CASE STUDY

Using thirty-six program elements (Figure 9) and the adjacency matrix for an actual hospital project (Figure 1), a hypothetical sloping site with an irregular outline was chosen to test the Evolutionary Parametric Program Adjacency Tool (EPPAT).

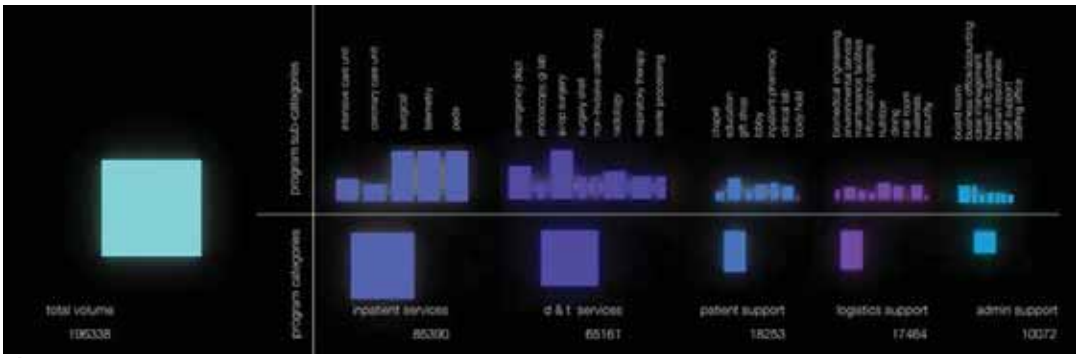


**Figure 9**: Program elements and relative square footages for a hospital. Source: Author

### 3.1. Tool deployed

Figures 10-13 depict the Grasshopper definition for this specific set of programs and adjacencies as well as the resulting Galapagos generated iterations, using the steps outlined in Section 2.2

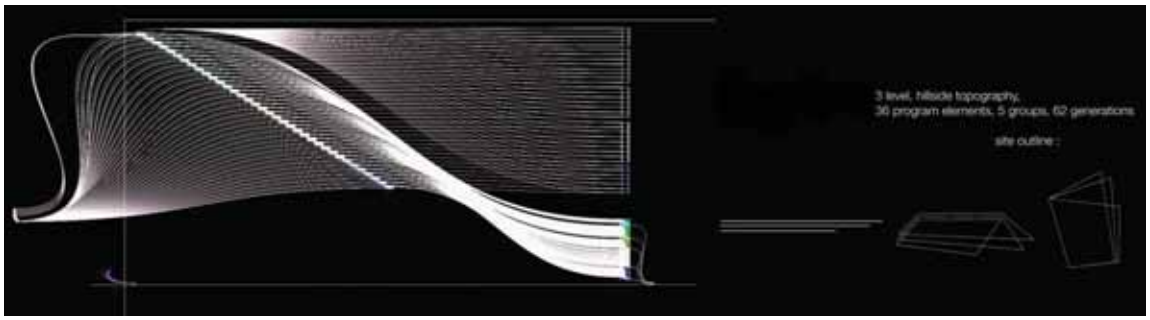


**Figure 10**: Grasshopper definition (left) and site constraints (right). Source: Author

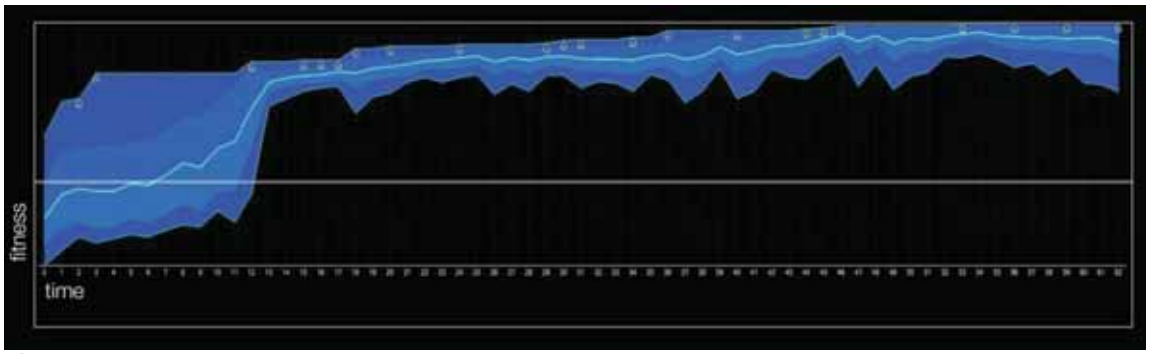


**Figure 11**: Graph depicts fitness variation(y) over generation number (x). 62 generations took 8 hours. Source: Author

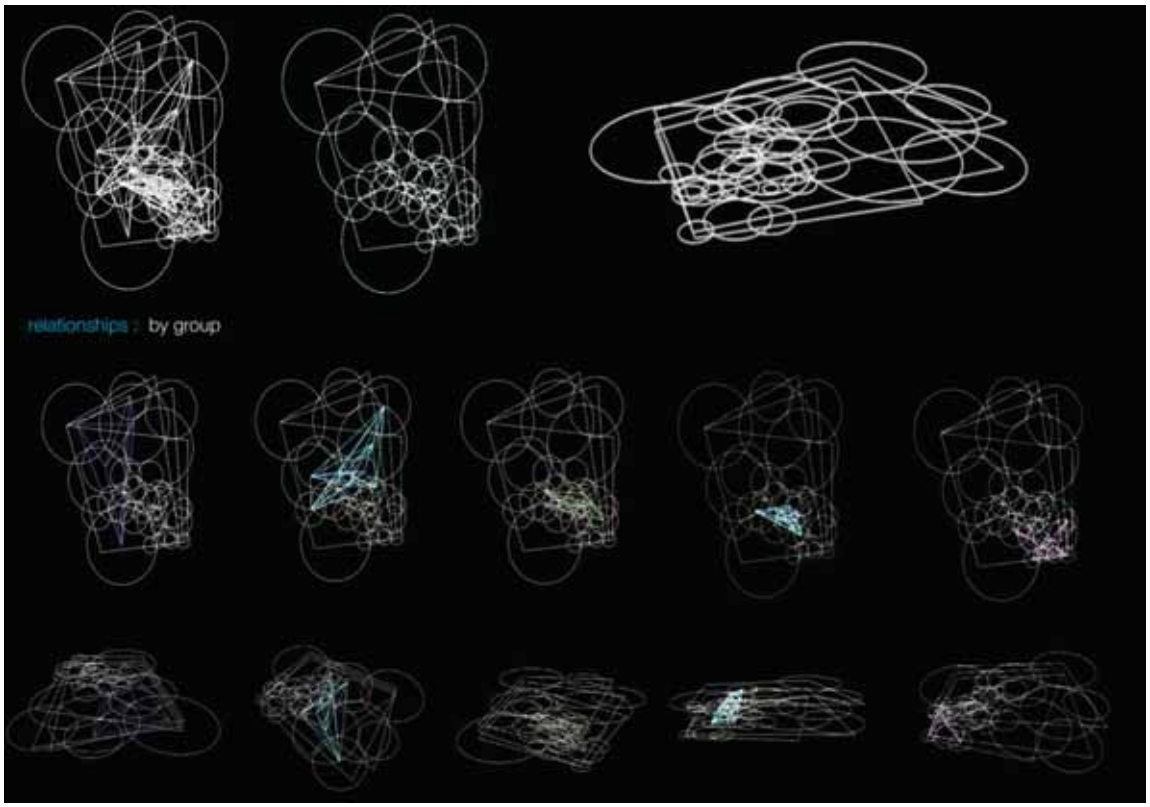


**Figure 12**: Generations, highlighting the geometries and distances used to optimize adjacencies and relationships by program type ("group"). Source: Author

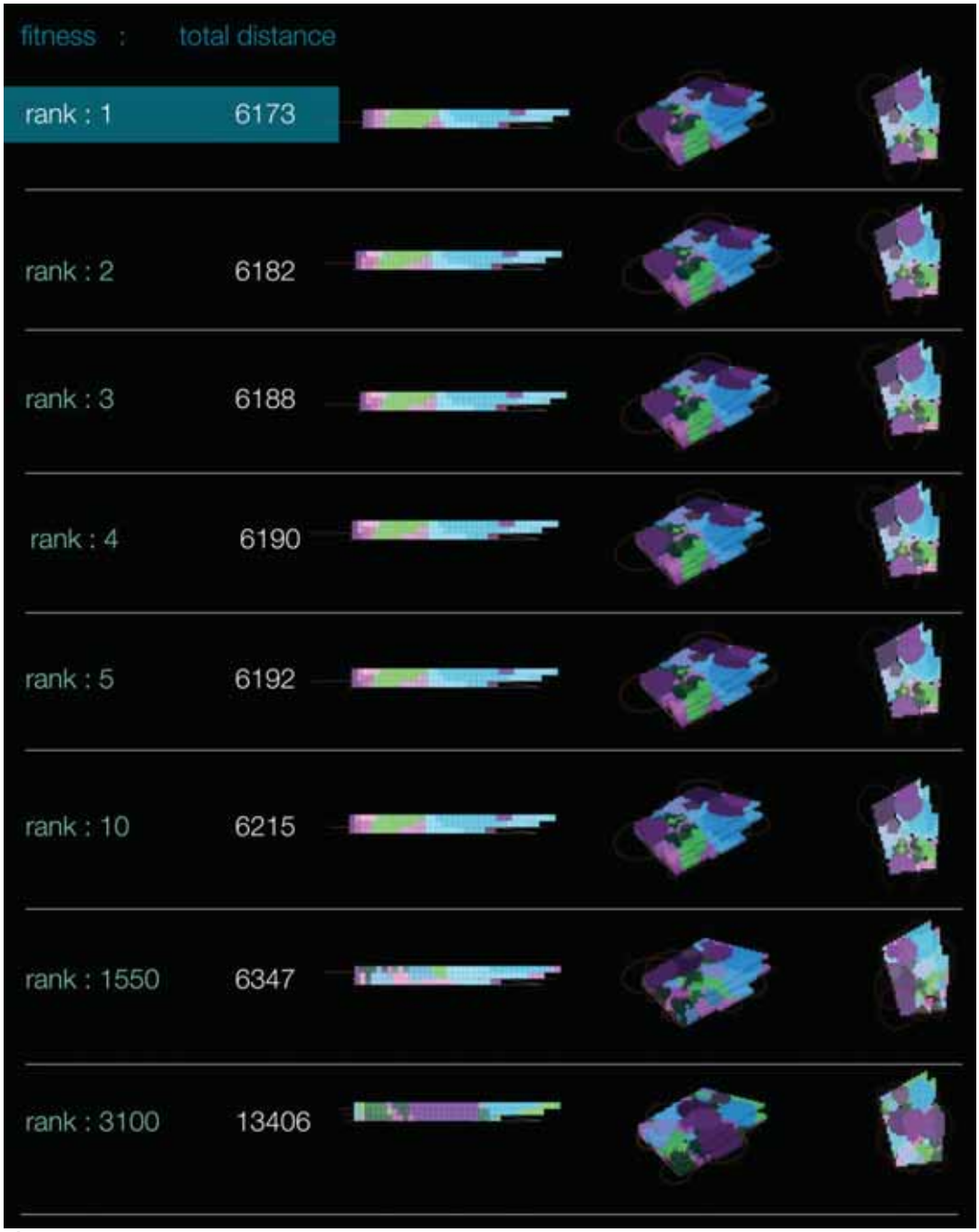

**Figure 13**: Least fit result (bottom) and most fit result (top) in section, isometric and plan. Source: Author

## CONCLUSION

The tool can be applied in a wide variety of situations in the early design phases. It can give designers metric data to inform their process and justify their decisions. The adaptability of the tool is critical to its usefulness. It has been simplified in terms of the scripting to a large degree but could perhaps still be refined to produce faster results. One observation from the experiments however is that the best way to save time is by sacrificing certain amounts flexibility. The more customizable the tool becomes, the more complex the calculation.

The tool can run indefinitely though it reaches a point where the results show no serious improvement. At these points it is best to either stop the process and take best result, or continue by restarting the process,

and hope for a different type of mutation to take hold of the evolutionary direction. There are clearly many ways to achieve similar results within the structure of the parametric tools, so this is just one solution to a problem that can be answered in many ways. Better definitions are ones that can accomplish complex moves within a few commands, this makes the load on the computer exponentially lighter and therefore it produces more results and is easier to use.

The introduction of the evolutionary process is extremely useful in creating iterative tools. Evolutionary algorithms are the way in which computer can independently handle time consuming iterative work. The ideal situation for the tool involves a combination of parametric operations with evolutionary solving. Then the tool is actually thinking and becomes more like a partner or double.

Creating the range for the results is the task of the designer, and the narrower the range, the more meaningful results are. In other words, narrowing the field where variables can occur creates more pertinent results. The end results of this definition are diagrammatic, so they serve as a source of information that can be observed as a metric then incorporated into design. It could be expanded into a more literal space, with doors and windows etc., and the results could become more directly applicable to building design in the later stages.

In general, the evolutionary modeling methods could be utilized on a much wider range of situations. Here research explored adjacency but any parameter or combination of parameters could be driving the model process. The evolutionary process enhances this practice. As a diagramming tool, parametric engines allow designers to start their process from an informed standpoint. The authors believe these methods are sound, but it will take further tests that lead to actual buildings that would later have to be evaluated to find out if the methods yield actual results that are effective.

## ACKNOWLEDGEMENTS

The authors would like to thank the Oregon Community Foundation for funding the seminars in which this research was conducted. The authors would also like to thank ZGF Architects for their collaboration on this research.

## REFERENCES

Lenski, R.E., Ofira, C., Collier T.C., Adami, C. 1999. “Genome Complexity, Robustness and Genetic Interaction in Digital Organisms” in *Nature*, Vol 400. (Macmillan Magazines Ltd.) 661-664

Motta, E. 1999. *Reusable Components for Knowledge Modeling: Case Studies in Parametric Design Problem Solving*. Amsterdam, IOS Press.

Roller, D. 1991. “An approach to computer aided-parametric design” in Computer-Aided Design Vol.23, Issue 5. Böblingen, Hewlett Packard.

Samuel, Flora. 2010. *Le Corbusier and the Architectural Promenade*. Basel: Birkhauser.

Scrawford. 2010 “Grasshopper Space Planner.” last modified 18 Jan. http://lmnts.lmnarchitects.com/parametrics/grasshopper-space-planner/

Syp, Marc. 2010. "Architecture: Realtime Physics for Space Planning." Vimeo. N.p., last modified 3 Mar. http://vimeo.com/15563685