10,000 Iterations

Computation as a Tool
for Schematic Design

by

Samuel Palacio Gutierrez

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in
partial fulfilment of the requirements for the degree of

Master of Architecture

Carleton University
Ottawa, Ontario

ABSTRACT //

Design as an exploratory method is an iterative process cycling through analysis, proposal, evaluation, and refinement. The dominant way of communicating ideas through drawings and models is limited by the static nature of the media. As an alternative approach, how can computational methods be used as tools for assisting in preliminary design?

10,000 Iterations studies computation as a supplementary tool for schematic design by developing an evolutionary model that generates optimized layouts according to the architect's criteria. This process, due to its computational nature, is limited to the aspects of design that can be expressed mathematically.

A simple design brief is developed as a critical method for refining the effectiveness and feasibility of this tool. The layouts generated by this process give the architect function driven material to consider for further development early on in the design process.

ACKNOWLEDGEMENTS //

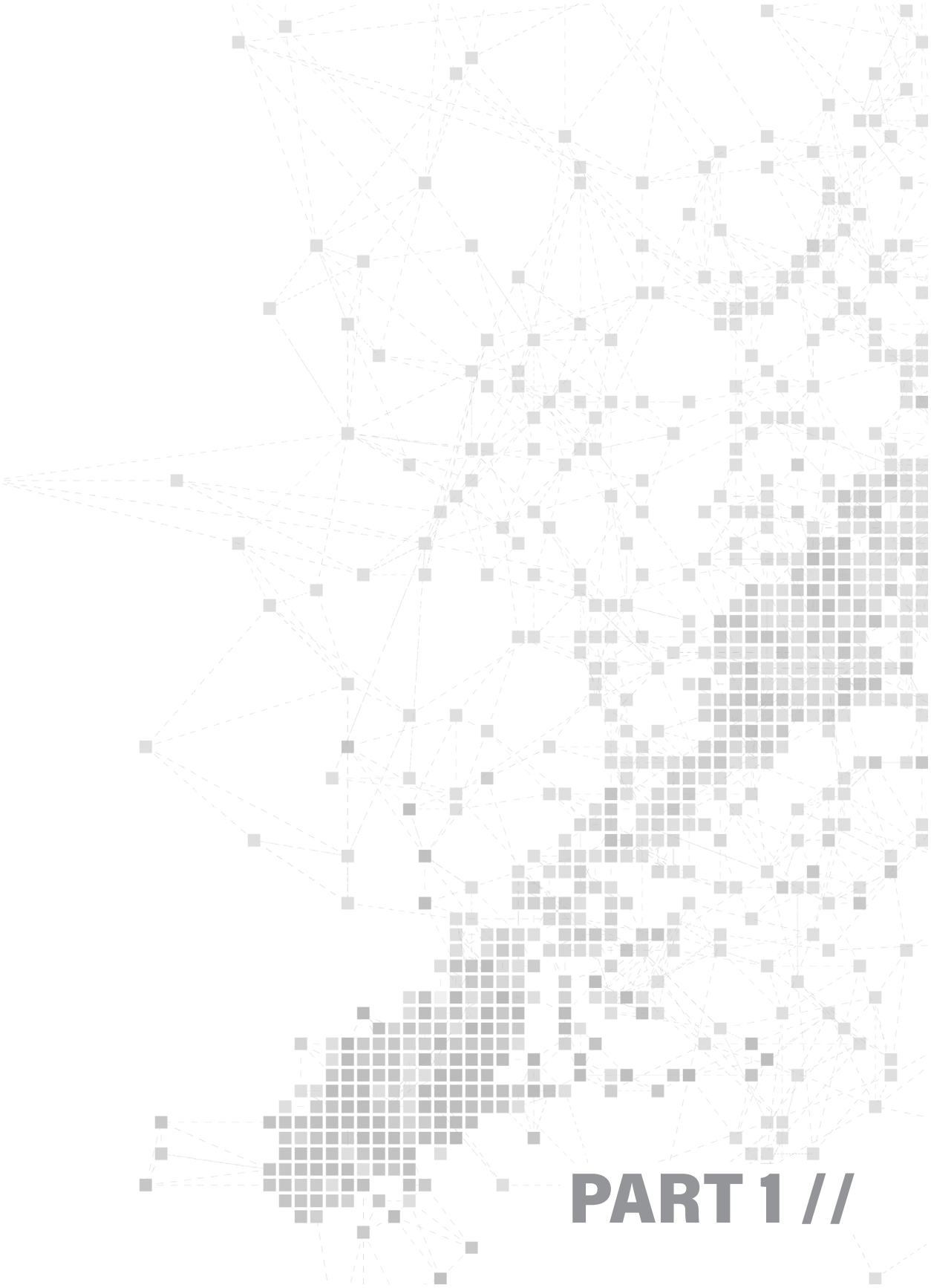
I would like to first thank my family, for their unconditional support and encouragement in everything I do. Mom, thank you for checking up on me and making sure I was taking care of myself. Dad, thank you for the many long discussions on what must have seemed like the most random of topics. I hold your wisdom and advice in the highest regard.

Thank you to my advisor, Professor Mariana Esponda for her dedicated review, her patience, and her unwaivering support throughout the course of this thesis.

A special thanks to Alison for her insight, criticism, and encouragement. You inspire me in everything I do.

TABLE OF CONTENTS //

LIST OF ILLUSTRATIONS //



PART 1 //

1.0 PROLOGUE //

10,000 Iterations details the development of a computational model for generating schematic layouts. Along the way, the thesis also explores what aspects of design can be computed and how design knowledge can be abstracted into computational terms by using the model for the schematic development of a simple design brief.

As part of a hybrid design approach, the computational model produces layouts for the architect to consider based on their own criteria. The layouts are intended to be starting points, catalysts for conversation that show the possibilities of the project.

1.1 DESIGN & PERFORMANCE //

Architecture is a technical response to an intricate system comprised of a project's requirements, context, and constraints (Figure 1). Fundamentally as a discipline it must address mankind's basic and higher-level needs. It must directly provide shelter, security, and the space for social interaction while also acting as a catalyst for self-fulfillment and actualization.[1] Hence, architecture, in its effort to best address all criteria through a single technical object is a complex solution which demands analysis and development.

1 Michael S. Bitterman, "Intelligent Design Objects (IDO) A Cognitive Approach For Performance-Based Design" (Ph.D., Delft University of Technology, 2009), 9, https://repository.tudelft.nl/islandora/object/uuid%3A785ecfa1-93e3-4186-9a48-006f7e004525.

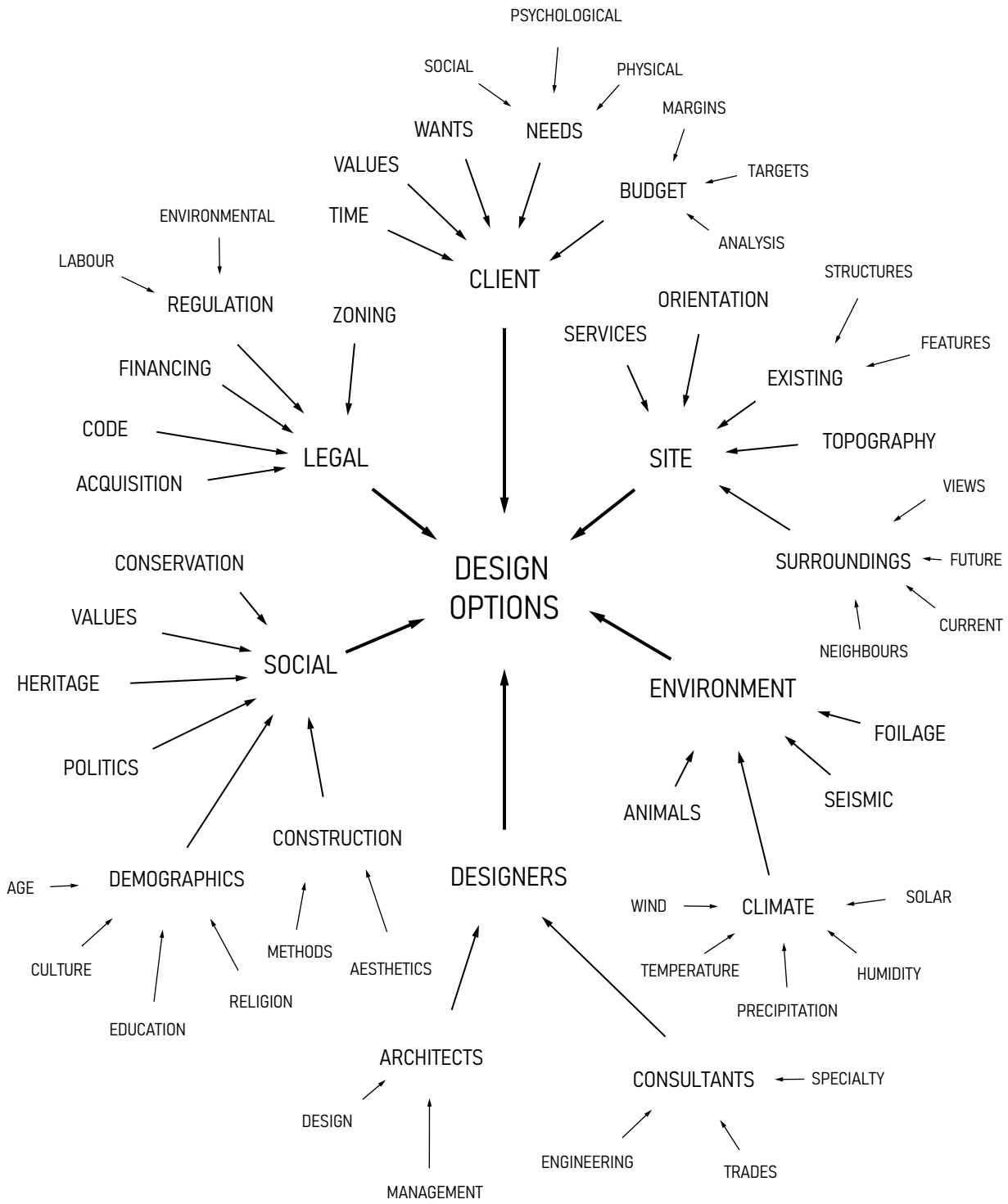


Figure 1.
An example of a simple architectural system.

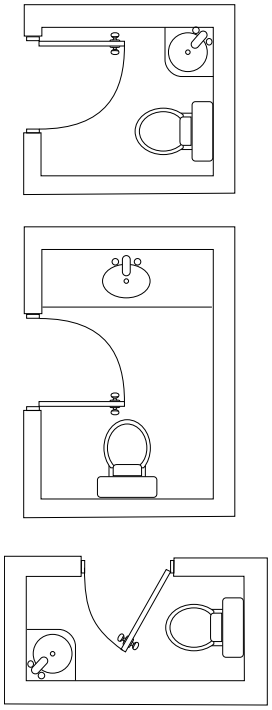


Figure 2.
To illustrate the softness of criteria: all three options are "small washrooms", so how does one pick between them?

Design is the iterative method for forming and refining complex solutions. The ill-defined nature of architectural problems makes design a challenging task. Projects have a multitude of soft criteria, each having various possible solutions, and each solution itself having many ways of being executed. The softness of criteria refers to the gradient of acceptability surrounding specified requirements, making it difficult to describe and recognize good solutions (figure 2).[2] [3] Evaluating a design's quality becomes even more difficult when weighing aesthetic qualities against functional or social ideals.

2 Bitterman, "Intelligent Design Objects", 25.
3 William J. Mitchell, "Three Paradigms for Computer-Aided Design", Automation In Construction 3, no. 2-3 (1994): 239-245, doi:10.1016/0926-5805(94)90023-x.

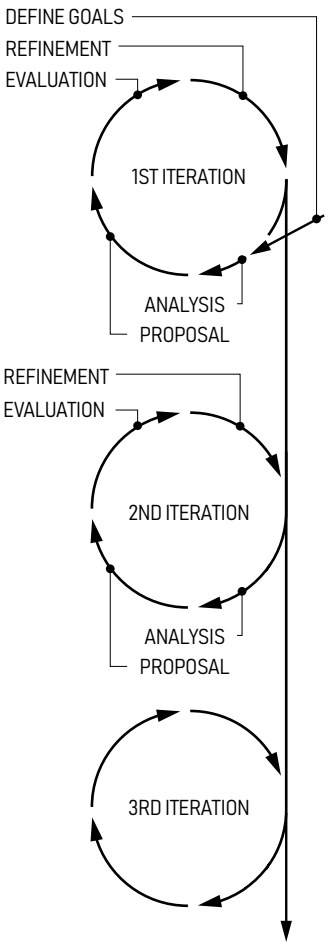


Figure 3.
The design process.

Thus, the design process is also a method for learning, during which the architect explores the system's intrarelationships through an iterative process cycling through analysis, proposal, evaluation, and refinement (figure 3). With each cycle, the system is adjusted to compensate for what was learned in the past cycle. As some criteria will inevitably have conflicting requirements, the final design must resolve conflict through compromise. Finding this balance requires holistic evaluation of the design across alpha, beta, and gamma sciences (figure 4). The alpha sciences are concerned with the artistic and subjective. Beta sciences are the rational and objective, and gamma sciences focus on the interests of society and culture.[4]

4 Sevil Sariyildiz et al., "Knowledge Model For Cultural Analogy In Design And Design Education", in Local Values In A Networked Design World - Added Value Of Computer Aided Architectural Design (repr., Delft: Delft University Press, 2004), https://cumincad.architexturez.net/doc/oai-cumincadworks-id-avocaad-2003-10.

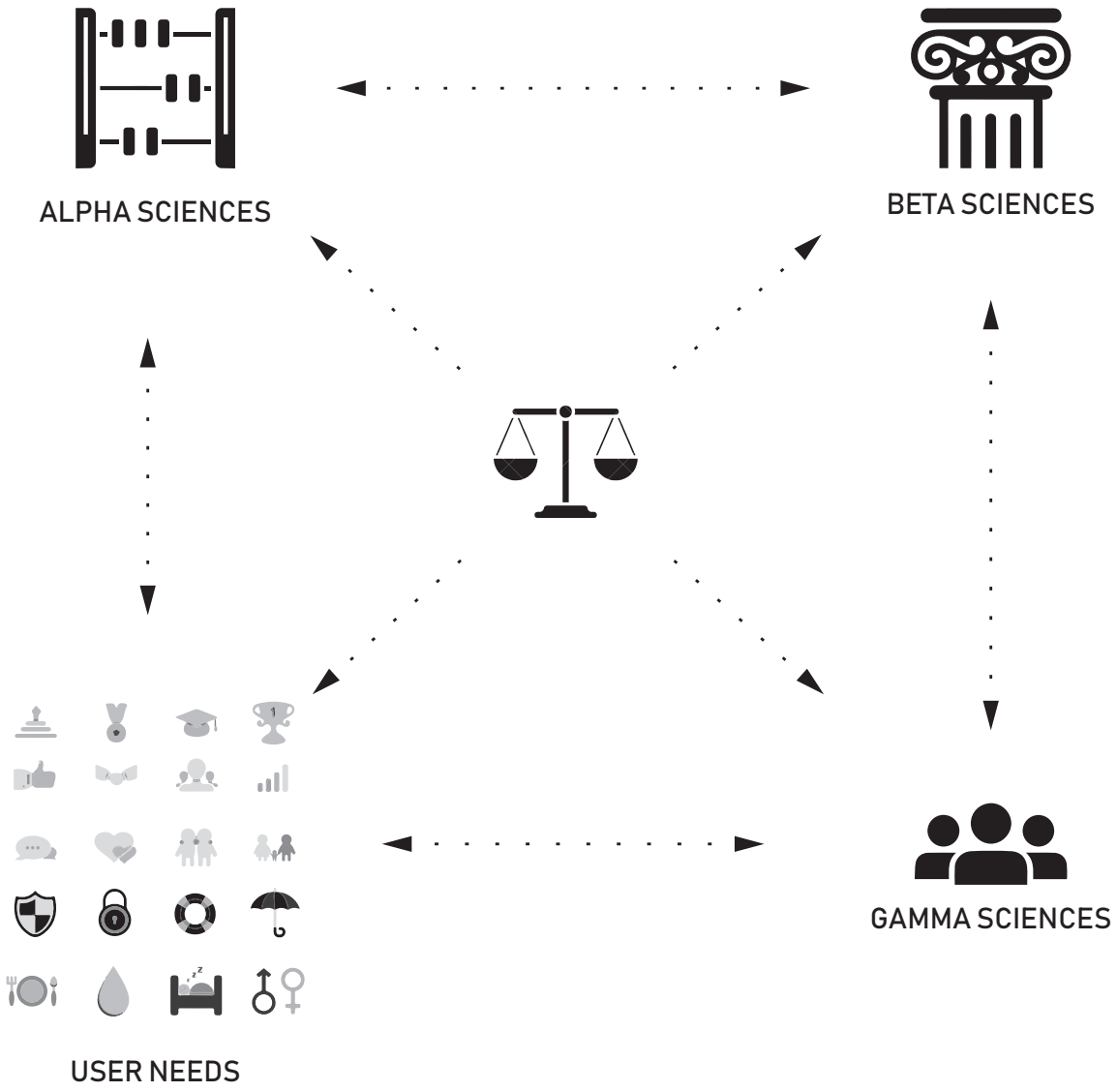


Figure 4.
Balancing Alpha, Beta, and Gamma sciences with user needs.

The architect's role then, is to define the system and establish relationships between its elements that fulfill the project's needs in an aesthetically pleasing, rationally sound, and socially aware manner. This is done through a design process led by the architect's intuition and experience. As methods for performance evaluation, architects abstract ideas into (analog or digital) models and drawings.[5] Although the static nature of these media makes them ideal for critical review and communication, it also makes the process of iterating and refining arduous.[6] These methods are further limited in that they are simplified versions of a solution, and therefore cannot provide holistic feedback.

5 Bitterman, "Intelligent Design Objects", 14.
6 Ibid.

1.2 COMPUTATION //

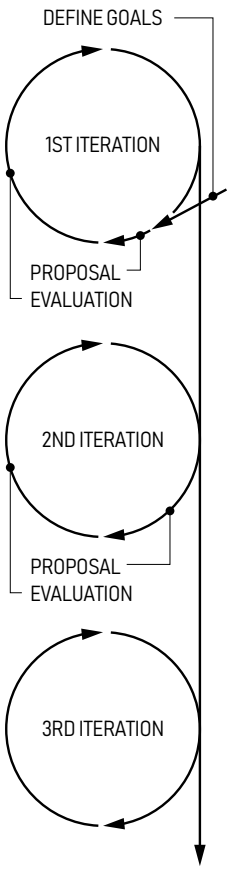


Figure 5.
The computational process.

Computation can assist in the schematic design process by rapidly generating suggestions for the architect to consider. For the purposes of this thesis, computation refers to the use of algorithms for spatial organization. The computer loops between two phases in developing a scheme: proposal and evaluation (figure 5). The proposal phase in the second iteration and onwards all take the evaluation from the previous phase into account. The cycle repeats until improvement stops or a target is met. The computer can simultaneously carry out hundreds of these processes.

While computational methods take advantage of the computer's rapid processing and vast memory for quick and accurate iteration, only the parts of the design system that can be described mathematically can be considered. Design requirements such as areas, proximity, and cost can be easily translated into computational terms, but accounting for qualities like spatial atmosphere and material combinations presents a much greater challenge due to their subjectivity. As such, computation is limited to the schematic planning of rational and quantitative elements.

For the computer to judge and refine its iterations, the architect must define performance for each design element. This involves translating the architect's knowledge into computational terms, essentially a set of simple design rules. The computer can then gauge a design's performance based on how well it adheres to the principles described in the rules.

1.3 THESIS QUESTION //

While this thesis relies on academic sources, the primary method of research is design. Specifically, the design of a computational process that it itself generates spatial layouts. In designing this computational process, it is critical to ask: what aspects of design can be computed? and how can design knowledge can be expressed computationally? 10,000 Iterations begins a discussion on how computation can be used as an effective method for assisting in schematic design. The layouts generated by the process are meant to provide the architect with a starting point. It is then up to the architect to develop the non-computable aspects of design as they see fit.

1.4 CLARIFYING TERMINOLOGY //

The informal use of digital computing in combination with a lack of standard terminology in architectural software, when compared to computer science, has resulted in the development and acceptance of an ambiguous lexicon.[7] The informality of computational education in architecture is especially noticeable in general discussions. Terms like parametric, algorithmic, and scripting are used interchangeably, suggesting that the technicality of those processes is not understood.[8] This section will clarify concepts and terminology that are essential in developing the computational model.

7 Aasholm, "Incessant Replication", 38.
8 Ibid.

1.4.1 COMPUTER AIDED VS COMPUTATIONAL //

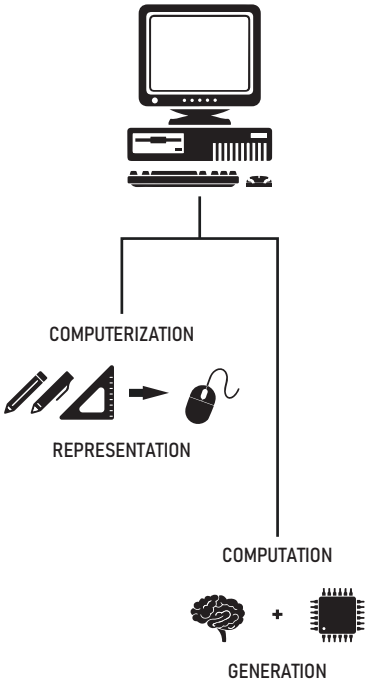


Figure 6.
CAD is a computerization of representation methods while computation is generative.

Computer Aided Design (CAD) refers to the production and representation of a design's geometry. Through these objectives, CAD ends up being a symbolic representation of data, and as such the tools found in CAD are all centered around an efficient workflow for representation.[9] Computation takes a different approach than CAD; it is defined as "the representation and use of knowledge to support or carry the synthesis of designs".[10] The fundamental difference between a computational approach and a computer aided one is that the computational method has the capability for innovation; it is an expedited exploration that returns new information. Computer aided design is a method for representation and workflow efficiency.

9 Ipek G. Dino, "Creative Design Exploration by Parametric Generative Systems In Architecture", METU Journal of the Faculty of Architecture 29, no. 1 (2012): 211.
10 Ibid, 211.

1.4.2 ALGORITHMS //

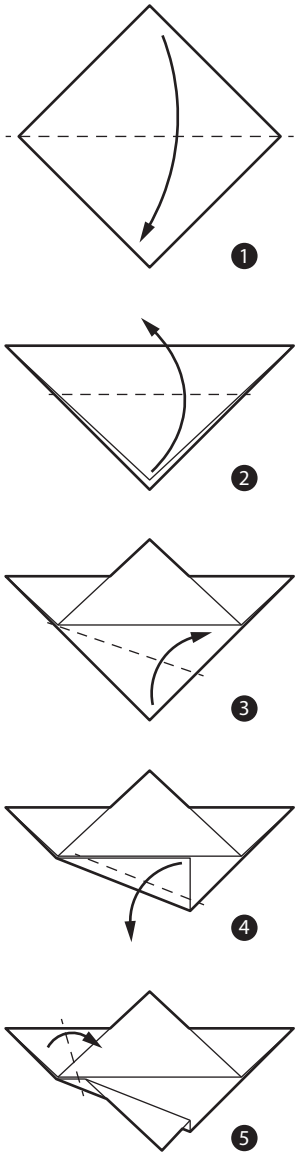


Figure 7.1
Origami is an example of an analog algorithm (continued in figure 7.2).

An algorithm is a finite set of procedures defining a succession of operations for the solution of a given problem. It is a predetermined sequence of explicit, elementary instructions described in an exact, complete yet general manner.[11] Algorithms do not have to be digital; figure 7 shows an example of an analog algorithm. For example, recipes, assembly instructions, or a "how-to" video can all be considered algorithms.

The main difference between analog and digital algorithms is the actuator, the entity that performs the logic. Typically, in analog settings, the actuator is a person capable of interpreting ill-defined instructions (ones that contain ambiguity). For example, in a baking recipe, "add a large amount of chocolate chips" or "bake until done" are both ambiguous instructions. How much is "a large amount"? What defines "done"? A computer needs to have "large" and "done" clearly defined, since it cannot interpret ambiguity.

11 Ahlquist and Menges, Computational Design Thinking, 11.

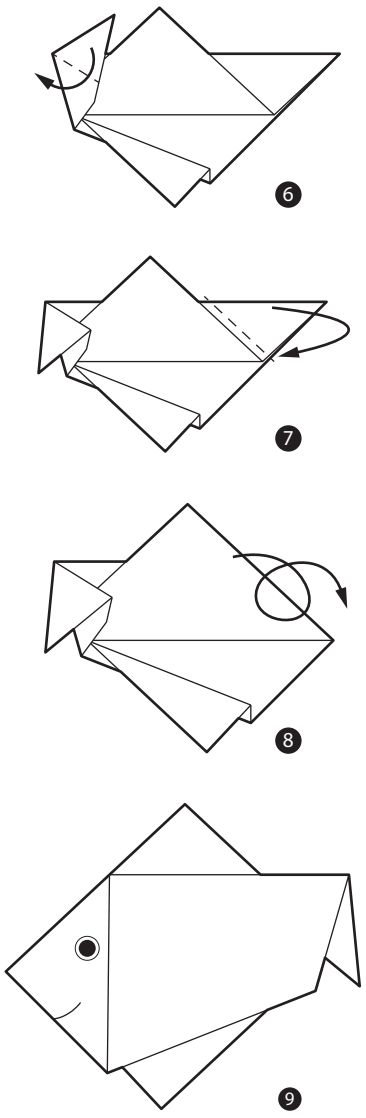


Figure 7.2
Origami is an example of an analog algorithm.

The term algorithm can also refer to the textual signs of a computer code.[12] When an algorithm grows to a complexity that incorporates many boolean operators and user inputs, and the clarity of its linear process is obscured by complexity, the term program is better suited than algorithm. A program has a constant discourse between its logic and its user.[13]

12 Ahlquist and Menges, Computational Design Thinking, 11.
13 Ibid.

1.4.3 SCRIPTING //

Scripting is a nonspecific term that can refer to low or high-level engagement with the computer. For example, both customizing the settings in a computer program and writing its code can be described as scripting.[14] Scripting then, is a significantly deeper engagement between the user and the computer than typical scenarios. Another way of looking at scripting is "the means by which the user gives highly specific instructions to the computer".[15] Note that this definition starts to blur the line between algorithm and script.

The difference between scripting and programming is the formality and complexity involved.[16] Simpler automation tasks in a non-professional setting are usually referred to as scripting. A computer scientist writing complex software is usually described as programming.

14 Mark Burry, Scripting Cultures (repr., New York, NY: John Wiley & Sons, 2013), 8.
15 Burry, Scripting Cultures, 9.
16 Aasholm, "Incessant Replication", 44.

Scripting is a valuable design tool as it allows the designer to use the software in ways that may not have been foreseen by its developers. As software becomes a limiting factor to the built environment, scripting provides an escape to software limitations by allowing the user to break away from the constraints of the user interface and predefined commands.[17]

17 Burry, Scripting Cultures, 9.

1.4.4 PARAMETRIC DESIGN //

The term parametric can refer to project parameters, the design method, the software used, or the stylistic movement parametricism.[18] Parametric design is a type of generative design, a method that emphasizes process over the final product and is defined by its procedural logic.[19] Both Le Corbusier's Five Points of Architecture and Louis Sullivan's plates describing the processes for reproducing floral ornamentation can be seen as methods for generative design.[20]

Parametric design is distinguishable within generative design as being exclusively digital. Within the digital environment, it focuses on the transformation of and relationships between parameters, typically through mathematical operations, to produce a design. As a result, adaptations are instantly reflected in the final design.[21] The adaptability of parametric design makes it an appealing design method for architects.

18 Davis, "Modelled on Software Engineering", 18-29.
19 Dino, "Creative Design Exploration by Parametric Generative Systems In Architecture", 207.
20 Ibid, 209.
21 Ibid, 208.

The definition for parametric design overlaps with the definition for algorithms, as they are both procedural methods. The difference between parametric and algorithmic seems to be a question of user interface. Parametric design tends to be implemented within CAD software and shows a geometrical representation of its logic while algorithmic processes are usually represented in text.

1.4.5 SYSTEMS THEORY //

Systems theory is a way of describing global behavior by understanding the complex and nonlinear interactions and reciprocities between the elements in a system.[22] Applied within architecture, systems theory dismisses notions of architecture being comprised of static, isolated entities, instead favoring the viewpoint that form is derived from the culmination of systems and their context.

Computational design is essentially managing the relationships in a systemic model (figure 1) to produce designs that are direct responses to the project's context. Adjusting the relationships in the model produces designs with different priorities.[23] The architect is responsible for defining the system's contents and their relationships. Examples of what a system might include are zoning and code requirements, site topography, local climate, and preferred views.

22 Ahlquist and Menges, Computational Design Thinking, 15.
23 Ibid.

1.4.6 MODELS //

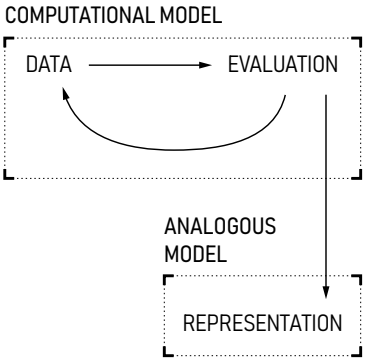


Figure 8
The computational model produces the analogous model.

This thesis deals with two models: the computational model, and the analogous model. The computational model is a set of algorithms developed in Grasshopper that describe the design process. Its written nature provides a holistic view of the logic and decision-making process, facilitating editing and refinement. To graphically communicate design proposals to the architect, the computational model produces analogous models (figure 8).

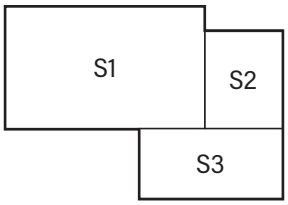


Figure 9
An example of the analogous models produced by the computational model. S1, S2, and S3 represent space 1, space 2, and space 3 respectively.

Analogous models represent the set of original properties with a second set of properties.[24] An example would be an architectural detail, where material properties are represented through hatch patterns. Applied to architecture and design, analogous models are static, and can only show a single design option.[25] The analogous models produced are schematic layouts, simplified diagrams illustrating spatial relationships (figure 9).

24 Broadbent, Design In Architecture, 89.
25 Ibid, 38.

PART 2 //

2.0 METHODS CONSIDERED //

The different algorithms that are suitable for schematic layout generation can be organized into three distinct categories: primitive, heuristic, and metaheuristic. In all three categories the algorithms aim to generate a layout that best meets the design and topological constraints established by the architect. Design constraints describe how to identify appropriate solutions and the topological constraints describe the desired relationships between elements in the layout.

2.0.1 PRIMITIVE METHODS //

Primitive methods require a complete understanding of the task at hand and use an exhaustive search method. This brute force approach computes all possible solutions and then selects the best one based on criteria established by the architect. This resource intensive method is limited to simple problems as more complex problems undergo combinatorial explosion, making it impossible to compute all solutions.[26]

26 Kalay, Architecture's New Media, 278.

2.0.2 HEURISTIC METHODS //

Heuristic methods mimic the architect's ability to use analogies and experience for guiding the search process in a non-exhaustive manner.[27] As such, heuristic methods find acceptable solutions relatively quickly and cheaply.[28] However, since computational methods require explicit instructions, any logic from analogies and experience must be translated to computational terms.[29]

27 Leonora Bianchi et al., "A Survey On Metaheuristics For Stochastic Combinatorial Optimization", Natural Computing 8, no. 2 (2008): 239-287, doi:10.1007/s11047-008-9098-4, 242.
28 Mitchell, Computer-Aided Architectural Design, 53.
29 Kalay, Architecture's New Media, 256.

2.0.3 METAHEURISTIC METHODS //

Metaheuristics are higher level algorithmic procedures for generating solutions to optimization problems. They are general constructs that can be applied to different types of problems. For example, the same metaheuristic algorithm can be used for space planning, cost estimation, and shortest route problems. Instead of being guided by the architect's experience and intuition, metaheuristics draw analogies from natural processes for guidance.[30] Independence from human intuition and tradition gives metaheuristics the possibility of discovering novel designs shrouded by human experience.[31]

Metaheuristic algorithms were selected as the preferred computational method since the ill-defined nature and combinatorial possibilities of space planning are outside the scope of primitive methods. The possibility of novel design generation and superior handling of ill-defined problems with incomplete information makes metaheuristics more promising than heuristic methods.

30 Bianchi et al., "A Survey On Metaheuristics ", 243.
31 Kalay, Architecture's New Media, 278.

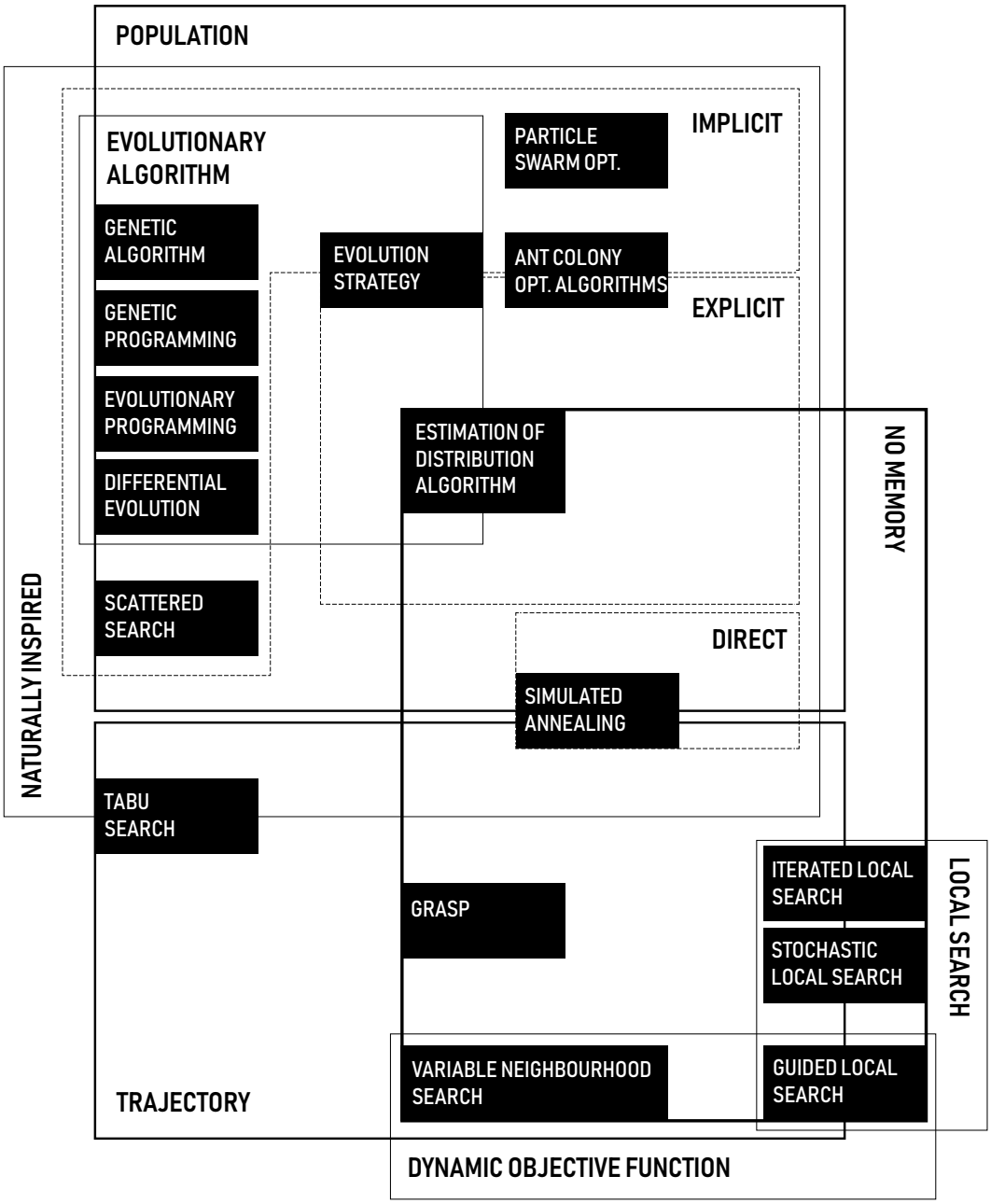


Figure 10.
Mapping of the main metaheuristic strategies.

Figure 10 shows a mapping of the main metaheuristic strategies.[32] Evolutionary algorithms are of particular interest for their ability to handle complex problems with many variables, and for their customizability.[33] The can be further subcategorized as Evolutionary programming, Evolutionary strategies, and Genetic Algorithms.

Genetic algorithms were selected as the as the computational method due to their population-based approach and use of memory. Specifically, Grasshopper's Galapagos component was chosen for its accessibility, user interface, and integration with CAD in Rhinoceros software.

32 Aasholm, "Incessant Replication", 183.
33 Clemens Heitzinger, "Simulation And Inverse Modeling Of Semiconductor Manufacturing Processes" (Ph.D, repr., Vienna University of Technology, 2002).

2.1.0 THE COMPUTATIONAL MODEL //

The goal of the computational model is to produce functional layouts for the architect to consider. However, the model does not technically "produce" these layouts, it "finds" them. As such, the computational model requires a space to search, a search method, and an idea of what it's looking for. Accordingly, the model is divided into three respective parts: the combinatorial model, the genetic algorithm, and the fitness function.

2.1.1 A SPACE TO SEARCH //

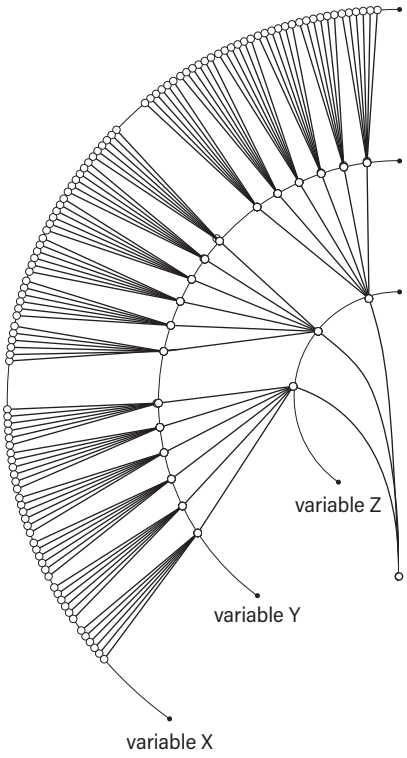


Figure 11
An example of a state action graph. Each white circle represents a unique combination of variables. If X,Y, and Z each have 6, 6, and 3 possible values, the model has 108 possible "states" or designs.

The combinatorial model is a parametric framework for defining the project's phase-space and selecting designs within it. Phase-space is an n-dimensional space that contains all the designs the architect wishes to consider. To define it, the architect must first specify how many spaces the layout will have, then the domain of each space's position and dimensions. In searching for functional solutions, the genetic algorithm adjusts the combinatorial model's parameter values, effectively moving and resizing spaces in the layout. Figure 11 is an example of a state action diagram. It shows all the possibilities a combinatorial model embodies. The genetic algorithm can only select states contained within the scope of the combinatorial model.

2.1.2 A SEARCH METHOD //

Galapagos, the genetic algorithm, is the only part of the computational model that is not developed by the architect. Its search for a functional layout simulates a population evolving over time through natural selection. The population is comprised of individuals, each representing a design, and is divided into generations. The design's properties, such as location and size, are described in the individual's genome. The most fit individuals as determined by the fitness function are selected to survive and populate the next generation. As such, fit and unfit traits are carried forward and discarded respectively during evolution.[34] [35]

34 David Rutten, "Galapagos: On The Logic And Limitations Of Generic Solvers", Architectural Design 83, no. 2 (2013): 132-135, doi:10.1002/ad.1568.
35 David Rutten, "Navigating Multi-Dimensional Landscapes In Foggy Weather As An Analogy For Generic Problem Solving", in International Conference

2.1.3 A DESCRIPTION //

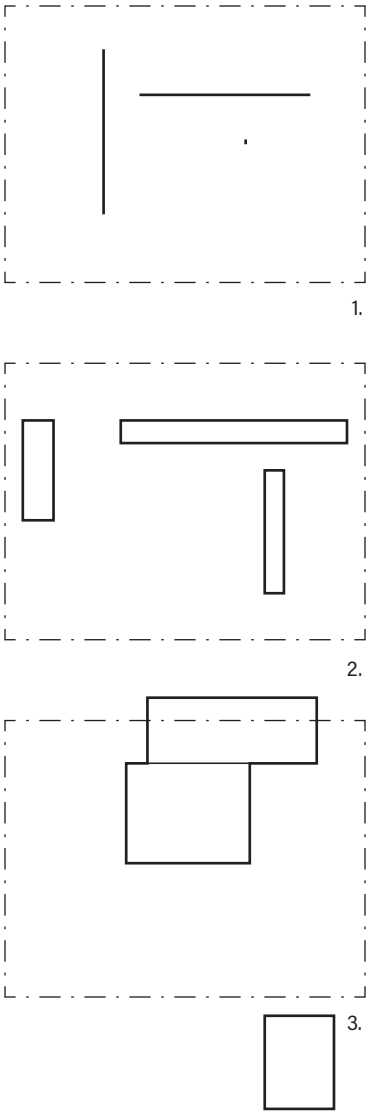


Figure 12
Examples of terrible solutions.
1. The spaces are 1 dimensional.
2. The spaces are poorly proportioned and far apart.
3. The spaces are not on the site.

Before the computational model can begin its search, it needs to know what to look for. The fitness function describes a solution's quality based on the architect's criteria. As projects are unique, the architect must establish the needs of each project prior to developing the model. Once those needs have been established, the architect must generally describe the properties of the designs which meet said needs. The descriptions must be specific enough that undeniably terrible solutions are always rejected (figure 12), but broad enough to fit a variety of different proposals.

The fitness function has 2 parts: a set of design rules and a corresponding set of weighting values. The design rules are the architect's descriptions of proposals that meet the needs of the project translated into computational terms. The weighting values are used to prioritize different rules. To determine the fitness of a layout, the computational model compares the degree to which the layout meets the design rules. The resulting partial fitness values are scaled according to the weighting scheme before being totaled to produce a single fitness value.

2.2 DESIGN BRIEF //

To simulate real world goals and constraints, a hypothetical design brief has been developed. The brief's intentional simplicity is meant to create a test project that can serve as a critical method for crafting and refining the computational model. In its simplicity, the test project strikes a balance between clearly defined objectives and formal flexibility while staying manageable with the available resources.

The brief features a freelance graphic designer as the client. They own a house in Old Ottawa South with a relatively large backyard and are seeking to take advantage of Ottawa's coach house bylaws to expand her business by building an independent office space in their backyard.[36]

36 Section 142 of Zoning By-Laws and Section 3.1 of Ottawa's Official Plan.

2.3.0 DESIGN RULES //

After interviewing the client and considering the project's systemic context, seven design rules have been established to generally describe the desired set of solutions. They constrain the computational model enough to consistently produce results relevant to the project, while maintaining enough flexibility to explore different design options.

The rules are as follows: area matching, proportion matching, spatial containment, spatial overlap, programmatic adjacency, programmatic connectivity, and footprint consolidation. The following sections will explain the reasoning behind each rule and elucidate their logic. As the rules are set by the architect, the definition of what makes a design fit is entirely dependent on their judgement.

2.3.1 AREA & PROPORTION MATCHING //

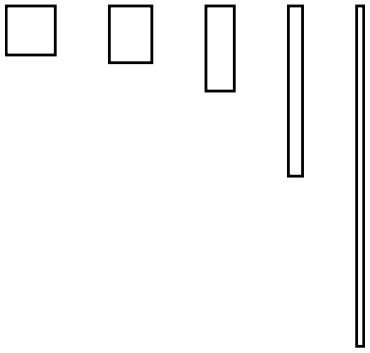


Figure 16
All these spaces meet the same area target, but have no ratio target. Without a proportion target, all of these spaces are considered equally fit.

The first two rules speculate the size and proportion of each space by comparing their current properties to a set of target areas and ratios. The targets are developed by the architect depending on the needs of the project. The fitness of each space reflects the difference between its current area and ratio and the corresponding targets. During evolution, the genetic algorithm will attempt to minimize the difference.

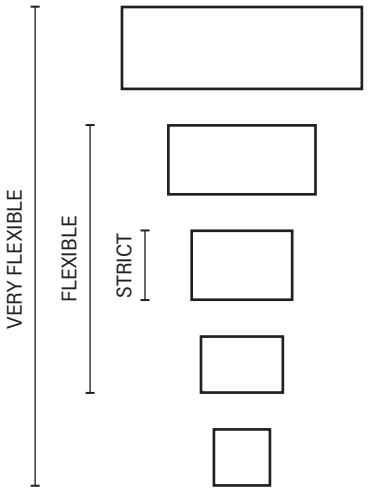


Figure 17
An example of how the weighting scheme changes the range of acceptable solutions.

Using the weighting scheme, the architect can specify how much each space will adhere to each rule. Meaning that the spaces' flexibility, the acceptable amount of deviation from the targets, is adjustable. This "softens" the criteria and is essential for design exploration. Figure 17 shows an example. The middle rectangle is the target area and proportion. As the weighting places less emphasis on the targets, the script will accept solutions farther away from the center.

2.3.2 SPATIAL CONTAINMENT //

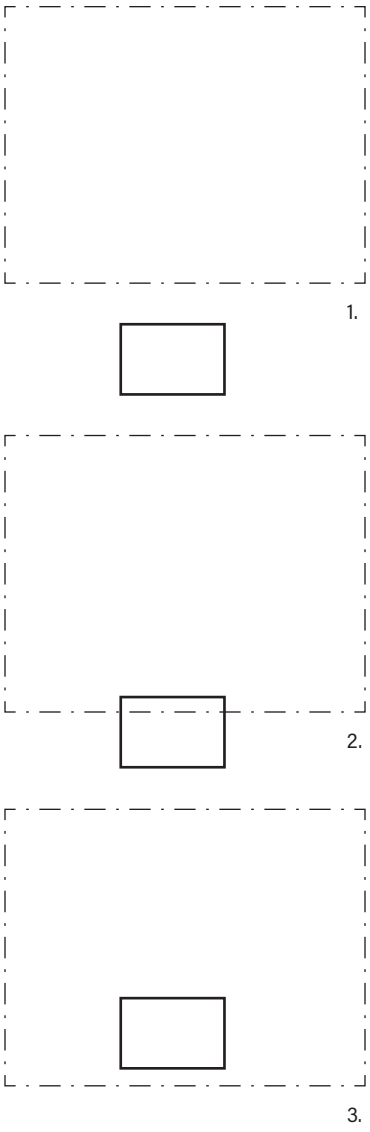


Figure 18
Progressive overlap. As the space comes within the boundary, the fitness increases.
1. Worst
2. Better
3. Best

The third rule ensures that all spaces are within the site's developable area. The architect establishes the developable area from the applicable bylaws and by assessing any special conditions on the site. The result is a polyline that broadly represents where the architect wants to build.

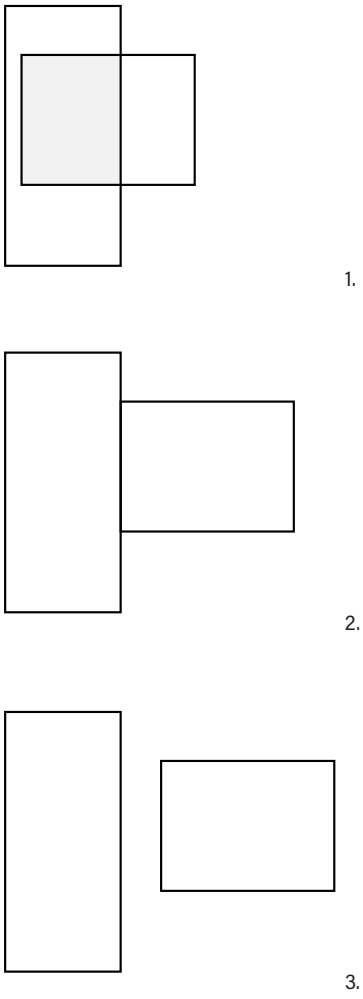
As the genetic algorithm iterates, the fitness function constantly tests if the spaces are within the developable area. The greater the overlap between a space and the developable area, the better its fitness. This progressive approach (figure 18) was chosen over a boolean evaluation (a yes – no evaluation) as it allows for finer compromise between this rule and the others. Meaning, the script might propose fit solutions that challenge the boundaries of the developable area.

Within the developable area there is a need for establishing forces that repel spaces away from undesirable locations. For example, it may be desirable for a porch to be in the sun, or for the whole layout to be on higher ground. It was attempted to incorporate these factors into the computational model with the same dynamic and scalable functionality as the rest of

the rules. However, the time required to run the script increased more than eightfold.

As a result, these factors have been statically incorporated into the developable area polyline. For example, areas that are in shade the whole year have been excluded from the developable area. Unfortunately, this means that dynamic weighting has not been applied to solar considerations due to the available hardware.

2.3.3 OVERLAP PREVENTION //



1.

2.

3.

Figure 19
The three stages of overlap.
1. Overlapping
2. Edge overlap
3. Not overlapping

The fourth rule ensures spaces don't overlap. There are 3 states of overlap: overlapping, edge overlap, and not overlapping. This rule prevents spaces from overlapping in the same progressive manner as the third rule. The additional rules needed to guide spaces into edge overlap are discussed in section 2.2.4.

By separating spaces by function, this rule inherently limits the script to producing certain layout typologies. So far, these typologies have been described as functional since their goal is to meet a set of quantitative criteria established by the architect. The computational model's difficulty in considering the social, political, and spiritual aspects of human life means that conceptual projects that seek to comment on these aspects are best left for the architect to develop.

2.3.4 PROGRAMMATIC ADJACENCY & CONNECTIVITY //

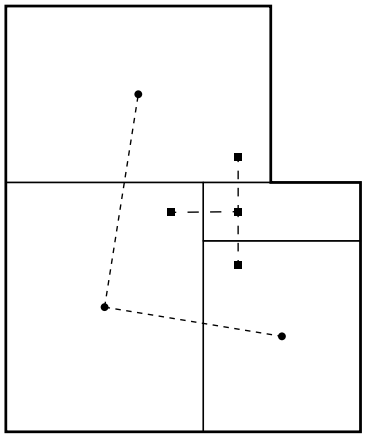


Figure 20
Connectivity and adjacency. The short dashed line represents adjacency and the long dashed line represents connectivity.

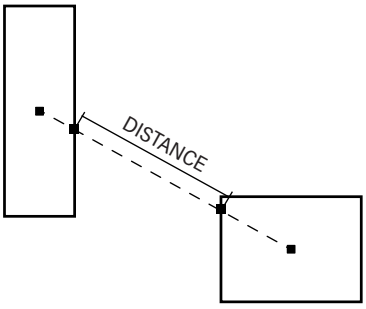


Figure 21
Fitness is calculated by taking the distance between each pair of spaces.

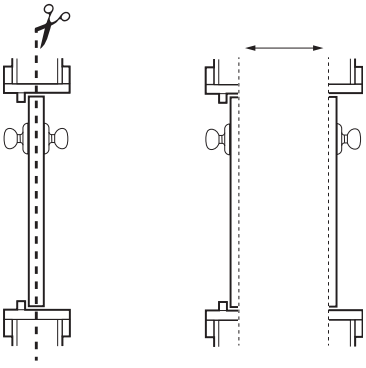


Figure 22
Fitness is calculated by taking the distance between each pair of spaces.

The next two rules pull spaces together into the edge overlap condition according to a hierarchy scheme set by the architect. The scheme specifies which proximities should be prioritized. The adjacency rule affects spaces which need to be close together but not connected (figure 20, short dashed line). Their fitness is determined by the shortest distance between each pair of spaces (figure 21).

The connectivity rule addresses spaces which the architect has determined need to be physically connected; spaces that require a door or aperture between them (figure 20, long dashed line). As a result, this rule places connection points on each pair of spaces and minimizes the distance between the points. Essentially, each point represents half a door (figure 22). To minimize the distance, the genetic algorithm can move both the spaces and their connection points.

To ensure feasible connections are generated, the architect must describe what they consider to be an acceptable connection. Like in the spatial containment rule, this is a broad definition meant to guide the genetic algorithm away from undesirable designs

without specifying a specific type of solution. Fitness for connected spaces is determined by the distance between each pair of points, and the degree to which the points meet the description of an acceptable connection.

2.3.5 FOOTPRINT CONSOLIDATION //

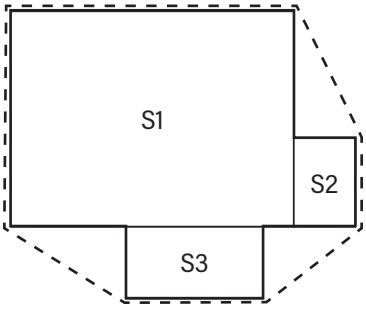


Figure 23
The convex hull of this layout is represented by the dashed line. It is formed by connecting the outermost points of the layout.

Figure 24
An example of a compacted layout. Figure 23 is an example of a non-compacted layout.

Through this last rule, the architect stipulates how compact the footprint of the layout should be. It provides a method for the architect to generally match the local vernacular. For example, in rural areas the architect might want the layout to be more spread out, and in the inner city for the layout to be more rectilinear.

To determine a layout's consolidation the genetic algorithm compares the area of the layout's convex hull to the sum of the spaces' areas. In a completely compacted layout, the area of the convex hull and the sum of the spaces' areas are the same.

The architect adjusts the importance of this rule with the weighting scheme. Tuning the importance of this rule through trial and error is critical. If it's overly prioritized, the convex hull restricts the spaces from moving freely early in the genetic algorithm's iterations, consistently produces unfit solutions.

2.3.6 WEIGHTING SCHEME //

The weighting scheme is a set of multiplier values that prioritize different design rules by scaling their fitness. The process is as follows: first, the script determines the degree to which the current design complies with each design rule. This determines the design's partial fitness values, the fitness for each rule. Then, each partial fitness value is emphasized or understated by multiplying it with it's corresponding weight. This is done for each of the partial values, allowing for the architect to set what parts of the design are negotiable, and what parts must be strictly adhered to. The partial fitness values are then totaled to produce the design's overall fitness.

The architect has access to the partial and overall fitness values for each design. Meaning, the architect has a numerical description of how each design — as a whole, and in parts — is meeting the targets. This feedback will be referred to as metadata, and is useful for refining the script. If the computational model isn't producing desired outcomes, looking into the metadata reveals which rules need to be adjusted, or if additional rules are needed.

PART 3 //

3.0 PROJECT SPECIFICS //

The computational model was developed and refined in two phases: Version 1.0 and 2.0. Section 3 will go over the specifics of each version, how the project requirements were determined, what assumptions were made, and what targets were used. Samples of the designs and the corresponding weighting used by each version will also be shown.

Since the site and the developable envelope are common to both versions, they will be covered first before discussing the scripts.

3.1.1 SITE //

When deciding where the project should be located, it was important to select a site that would test the boundaries of the script while maintaining a level of realism in the project. To find sites that met these requirements, a neighborhood analysis of Ottawa was conducted. It investigated the demographics of different neighborhoods, the existing and upcoming housing typologies, the zoning bylaws, and the typical lot characteristics of each neighborhood.

The analysis looked for a neighborhood where density was being added through infill or low-rise development, the residents could afford to build, and where the backyards were large enough to accommodate different design options while meeting Ottawa's coach house guidelines.[37]

The analysis revealed that the Old Ottawa South neighborhood (figure 25) met all the targets. The selected site is on the South-West end of the neighborhood, located at 76 Seneca street (figure 26).

37 City of Ottawa, "How to Plan Your Coach House In Ottawa" (repr., Ottawa: City of Ottawa, 2017).

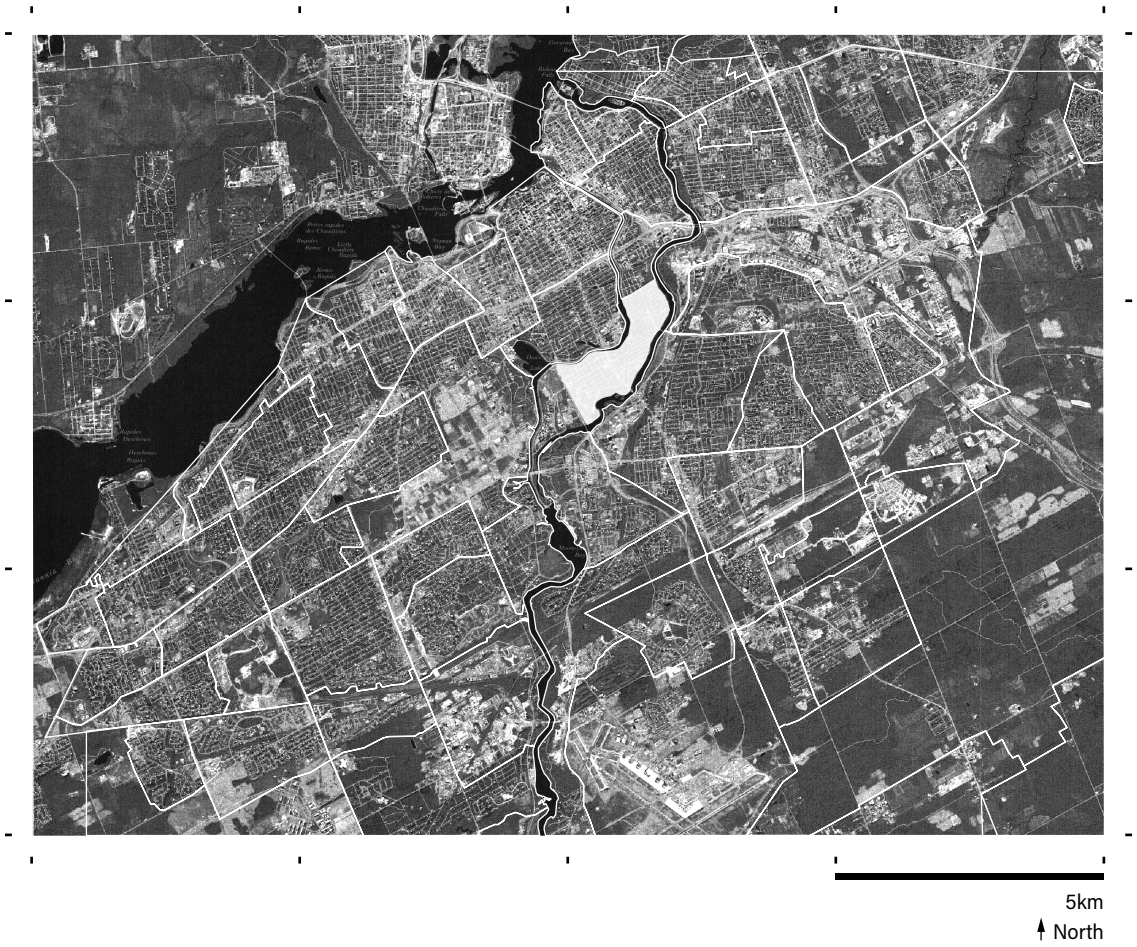


Figure 25.
Ottawa's neighborhoods. Old Ottawa South is highlighed in white.

500m
North
Site

Figure 26.
Old Ottawa South. The site is highlighted in red.

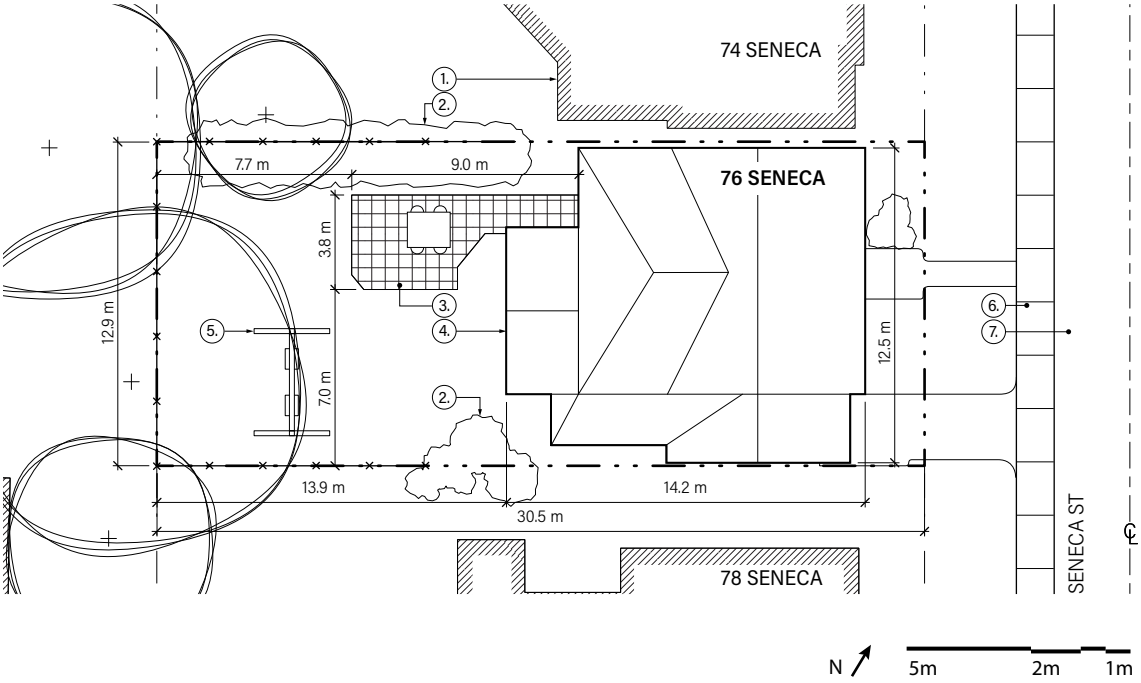


Figure 27.
Site Diagram.

As shown in figure 27, the lot measures 12.9m by 30.5m and has a two-story detached house. The rear yard has a depth of 13.9m and is shaded by several large trees. Most of the yard is unoccupied except for a small hardscaped dining area, a swing set, and hedges. The swings will be removed, but the dining area and hedges will be kept.

3.1.2 DEVELOPABLE ENVELOPE //

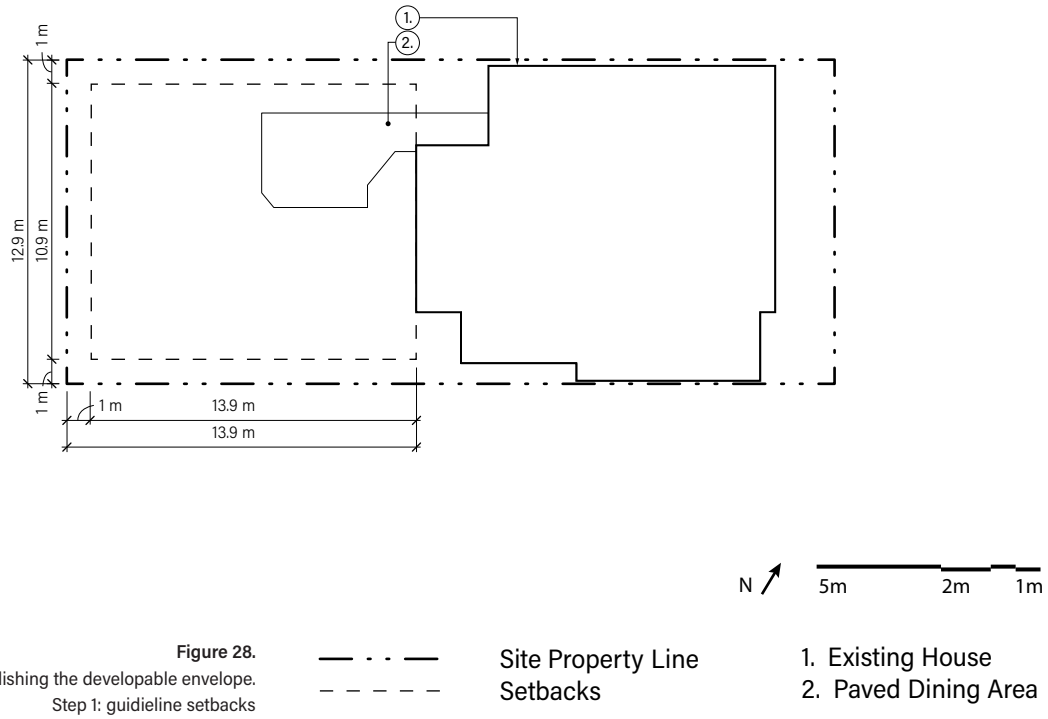


Figure 28.
Establishing the developable envelope.
Step 1: guidieline setbacks

The site's developable envelope was established in 3 steps. First, the setback requirements from Ottawa's coach house guidelines were applied to the site (figure 28).[38] The guidelines state that structures must be between a maximum of 1m and a minimum of 4m from the extents of the rear yard. A maximum of 1m has been used. [39]

38 Ottawa, "How to Plan Your Coach House".
39 Ibid, 17.

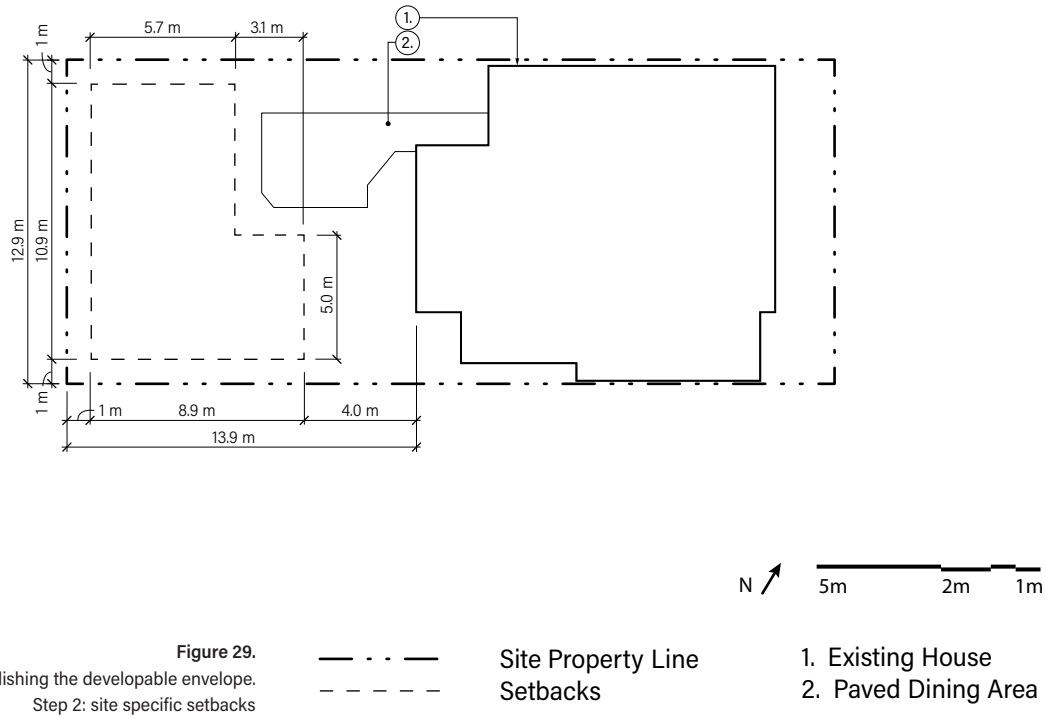


Figure 29.
Establishing the developable envelope.
Step 2: site specific setbacks

Then, additional setbacks specific to the site were added. These include a 1m setback from the paved area and a 4m setback from the house. Figure 29 shows the modified developable envelope.

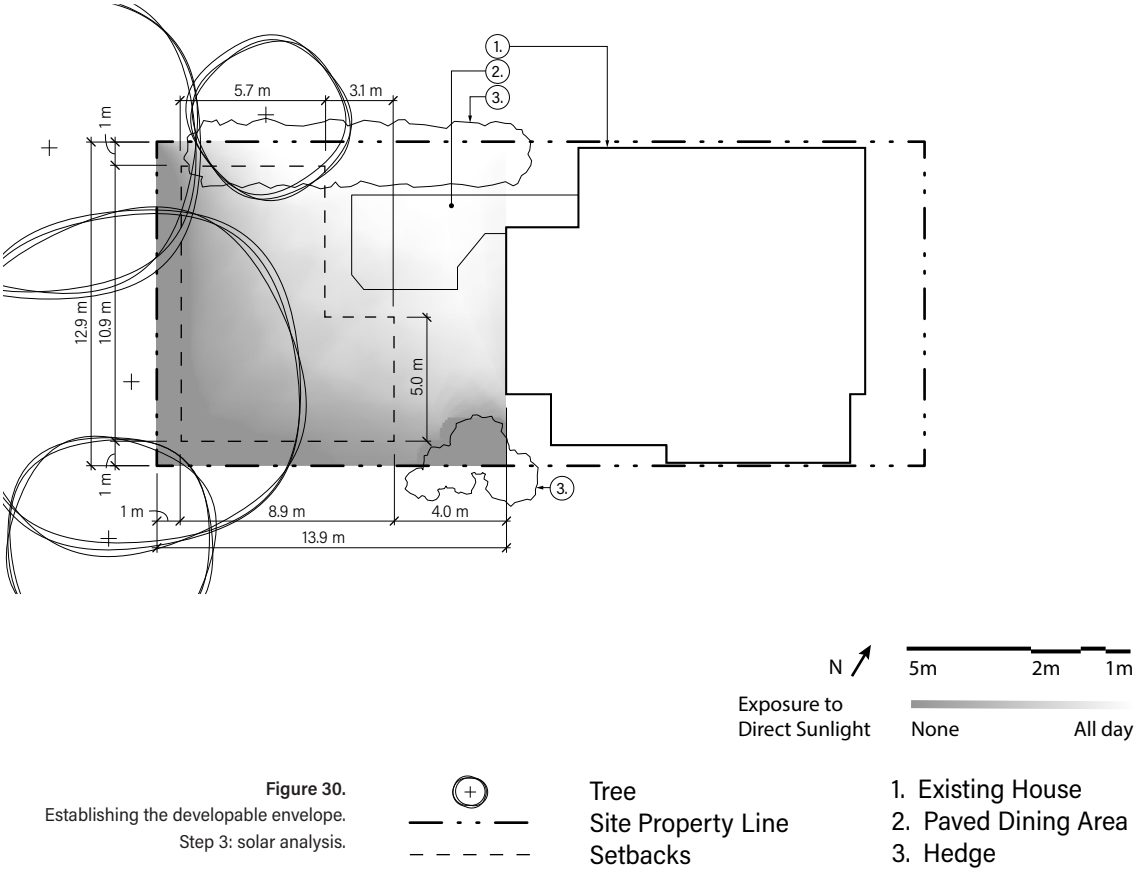


Figure 30.
Establishing the developable envelope.
Step 3: solar analysis.

Lastly, a solar study was conducted to find any areas of the site that are shaded year-round (figure 30). This study took the neighboring houses and surrounding trees, hedges, and fences into account. The solar study shows that the most shaded areas are outside of the developable envelope. As such, the setbacks shown in figures 29 and 30 will be used to define the area of the site the algorithm will be limited to.

3.2 THE CLIENT'S CRITERIA //

- Natural light
- Skylights
- Treadmill
- Desk
- Meeting area
- Open concept
- Printer area
- Big work table
- Tv near big table
- Washroom
- Kitchenette
- Storage for devices
- Sample room
- Broom closet
- Industrial
- New york loft
- Modern
- Visually spacious
- Contemporary
- Swiss

Figure 31.
The client's critera as it was said in the interview.

OBJECTS
- Big work table
- Desk
- Skylights
- Treadmill
- Tv near big table

SPACES
- Broom closet
- Kitchenette
- Meeting area
- Printer area
- Storage for devices
- Sample room
- Washroom

QUALITIES
- Contemporary
- Industrial
- Modern
- Natural light
- New York loft
- Open concept
- Swiss
- Visually spacious

Figure 32.
The client's critera sorted by objects, spaces, and qualities.

The client was interviewed and asked to describe an ideal home office space while keeping financial and spatial limits in mind. The information from the interview was then used to determine the project's criteria. Figure 31 shows the criteria listed by the client, in the order it was mentioned. Figure 32 shows the same criteria, sorted into spaces, objects, and qualities.

As the interview continued, the client was adamant about maximizing daylight but made it clear that they were flexible on the qualitative characteristics of the space. They also provided ideal or specific measurements for many of the furniture pieces and spaces. These measurements were used to calculate the spatial requirements in the upcoming sections.

3.3.0 DEFINING PROJECT REQUIREMENTS //

OPEN OFFICE	▪ Desk	2100 x 2710
	▪ Work table	1960 x 2110
	▪ Built in storage	600 x 2000
	▪ Treadmill	2540 x 1360
	▪ Kitchennette	600 x 1000
	Area + 15% for circulation:	17.4 m^2
WASHROOM	▪ Toilet	-
	▪ Sink	-
	Approx. area:	2.5 m^2
CLOSET	▪ Samples	-
	▪ Supplies	-
	▪ Storage	-
	Approx. area:	1.5 m^2
	Total req'd area:	21.4 m^2

Figure 33.
Calculating the spatial requirements for the project based on the client's criteria. All linear measurements are in mm.

Version 1.0 was focused on understanding the technicalities of computation and creating a working script. To facilitate this, the client's requirements were distilled into the fewest possible spaces that were still representative of the relationships found in more complex designs. This would allow for a scalable model to be developed. The simplified model divided the client's criteria into 3 distinct spaces: open office, washroom, and storage.

Figure 33 shows how the area requirement for each space was calculated. The client provided specific measurements for many of the requirements in the open office. The only requirements given for the washroom were for it to be small, and for it to be a two-piece washroom. The client gave a specific area target for the closet based on the amount of material samples they typically carry and their storage needs.

3.3.1 RULE SPECIFICS //

This section will go over the specific conditions and assumptions used to determine the targets for each design rule and the resolution of the combinatorial model. Figure 34 shows a summary of this section.

Area and ratio targets

The area target for the open office space was determined by adding together all the spatial requirements of the furniture and the kitchen as specified by the client, plus 15% of the total area for circulation (figure 33). For the storage space and the washroom, the area targets were determined by asking the client about their storage needs, and if they wanted a shower in the washroom.

Spatial containment

The spatial containment rule was set to penalize any spaces that were not completely within the developable envelope.

Spatial overlap

The spatial overlap rule was set to heavily penalize any overlap between spaces.

Programmatic adjacency and connectivity

The washroom and storage space were set to have an adjacency requirement between them. The office was set to have a connectivity requirement to both the washroom and storage space.

Footprint consolidation

The footprint consolidation rule was set to prefer compact, rectilinear footprints.

Domains and resolution

The domains of each space's width, length, and location parameters were set to exclude values that were significantly removed from the desired results. For example, the washroom's width and length domains are set to include values from 1m up to 3m. Anything less than 1m is uncomfortable, and anything larger than 4m is too big for the requirements of the project. The resolution, the step size between parameter values, of each domain is set to 0.5m. This reduces the combinatorial possibilities of the project without over constricting the number of possible designs.

	Office	Washroom	Closet
Area Targets	17.4 m^2	2.5 m^2	1.5 m^2
Ratio Targets	1.6	1.6	1.6
Spatial Containment	Within Developable Envelope		
Overlap Prevention	No Overlap Allowed		
Programmatic Adjacency	None	Closet	Washroom
Programmatic Connectivity	Washroom and Closet	Office	Closet
Footprint Consolidation	Prefer Rectilinear		
Width Domain	3.0m - 6.0m	0.5m - 3.0m	0.5m - 3.0m
Length Domain	3.0m - 10.0m	0.5m - 3.0m	0.5m - 3.0m
Width Resolution	0.5 m		
Length Resolution	0.5 m		
Position Resolution	0.5 m		

Figure 34.
Summary of the rule specifics covered in section 3.3.1.

3.3.2 V1.0 SEARCH PROCESS //

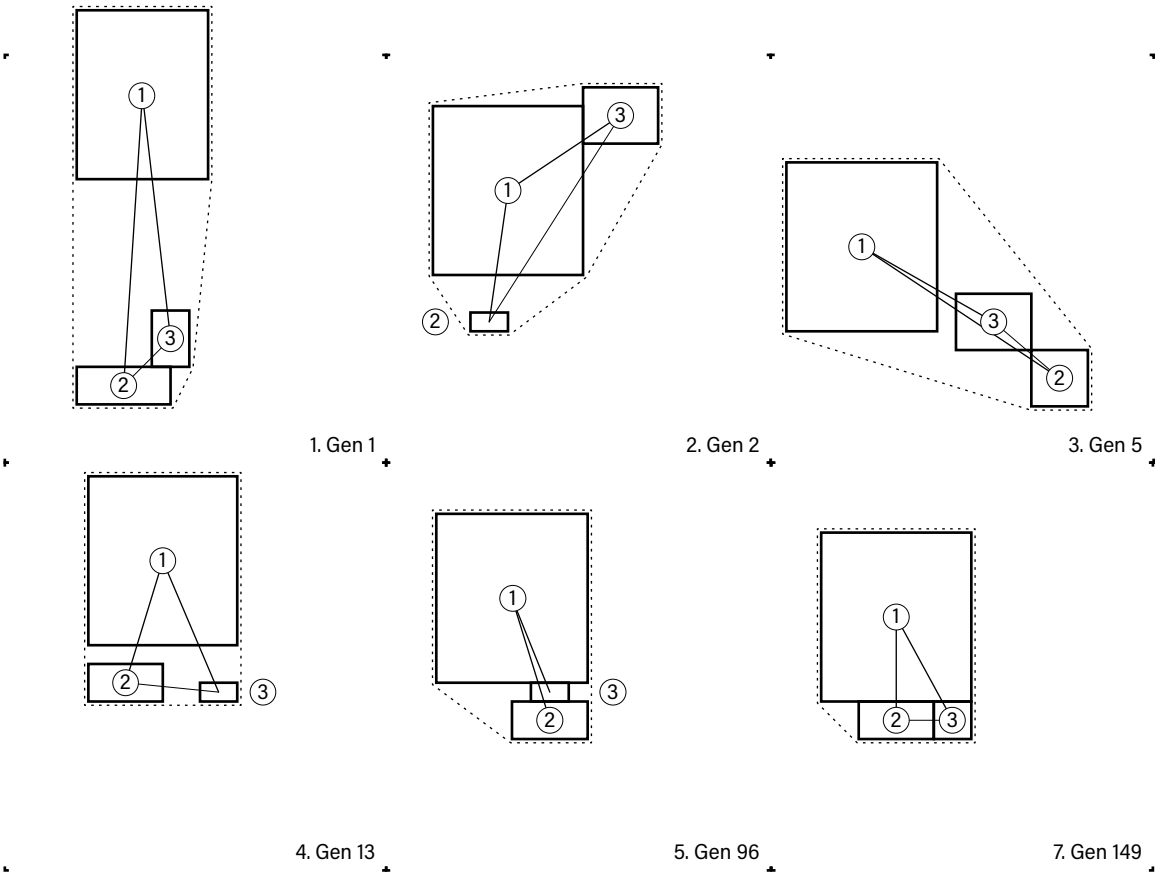


Figure 35.
The computational model's evolutionary process.

Figure 35 shows an example of version 1.0's evolutionary process from start to finish. As the generations pass, the spaces in the model are organizing themselves according to the criteria that has been developed by the architect in the computational model. The starting point is determined by the genetic algorithm through an initial random sampling of the parameters.

3.3.3 V1.0 WEIGHT TESTING //

Version 1.0's simplicity was essential for developing the script and learning how to use it. This next section will show the process of testing the computational model by flexing the weighting scheme. Figures 37 – 42 show different schematic proposals, each emphasizing a separate rule during computation. This process serves as a critical method, confirming each rule's effects on the layout and revealing any unforeseen side-effects.

Figure 36 shows the result of the computational process when the overlap prevention rule is not prioritized. The lack of distinction between inside and outside breaks the rest of the logic. For the purposes of this example, figures 37 - 42 will all have the overlap prevention rule prioritized as a baseline.

After testing the individual effects of the rules, different schemes that balance all the rules at once were explored. Figures 43 – 45 show how the outcome of the computational process can vary when the weighting scheme is balanced differently.

Notes:

The overlap prevention rule allows the script to distinguish inside from outside. Without this distinction, the spaces tend to converge. Furthermore, when the spaces overlap, the area targets become invalid as the same area is being counted twice.

Although the end result may be a valid design possibility, it fails to meet the goals of the research. Going forward, this rule is emphasized significantly over all others.

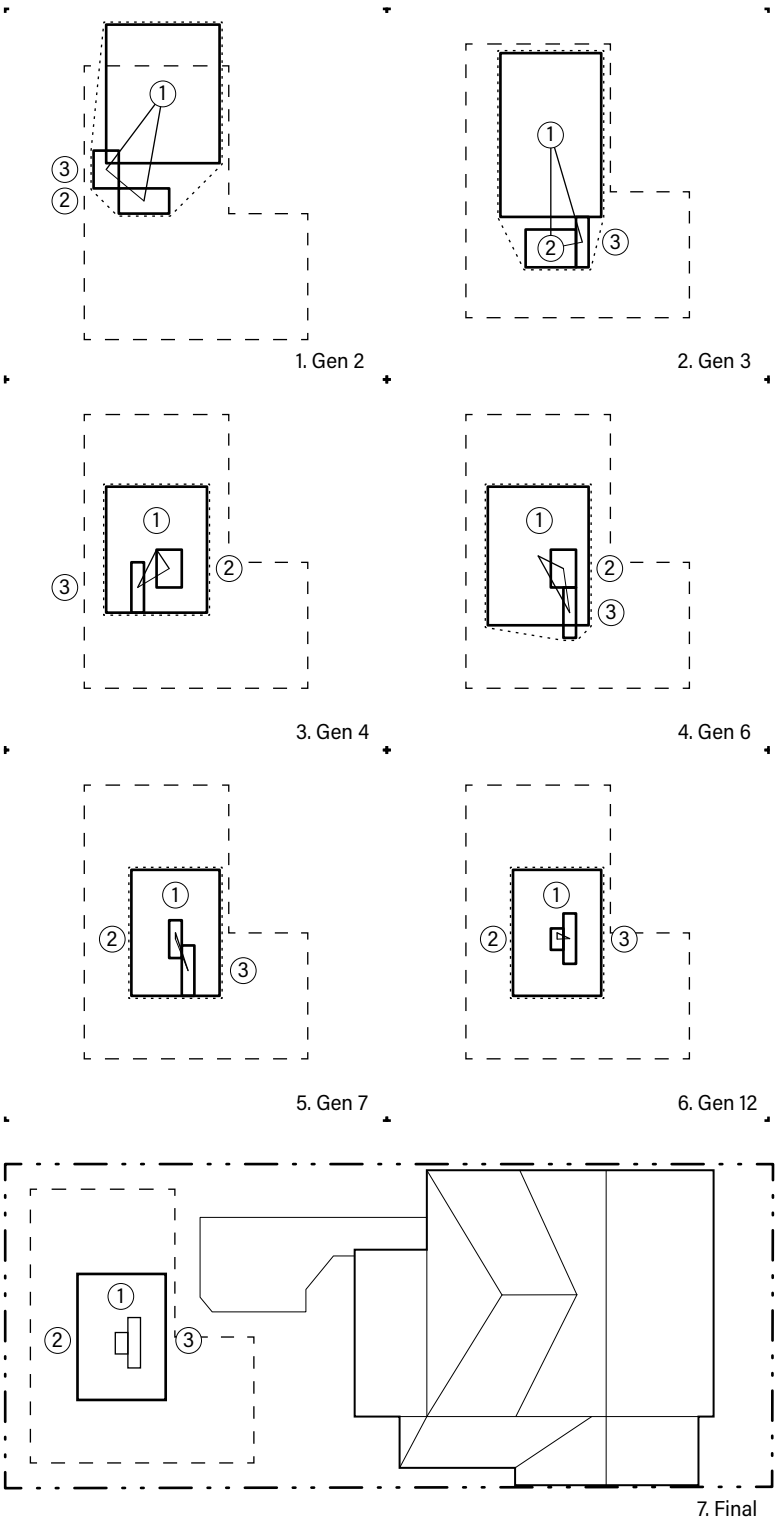


Figure 36.
If the overlap prevention rule is not prioritized, the computational model does not work.

Notes:

Figure 37 prioritizes the area matching rule. The fact that the spaces all ended up within the developable envelope at the end of the computational process is a coincidence.

At the end of the process, a perfectly fit solution was not found. Meaning, the spaces did not end up meeting the area targets. This is because the coarseness of the length and width resolution don't allow for the spaces to ever be the desired size. Going forward, it should be confirmed that area targets are within the extents of the combinatorial model.

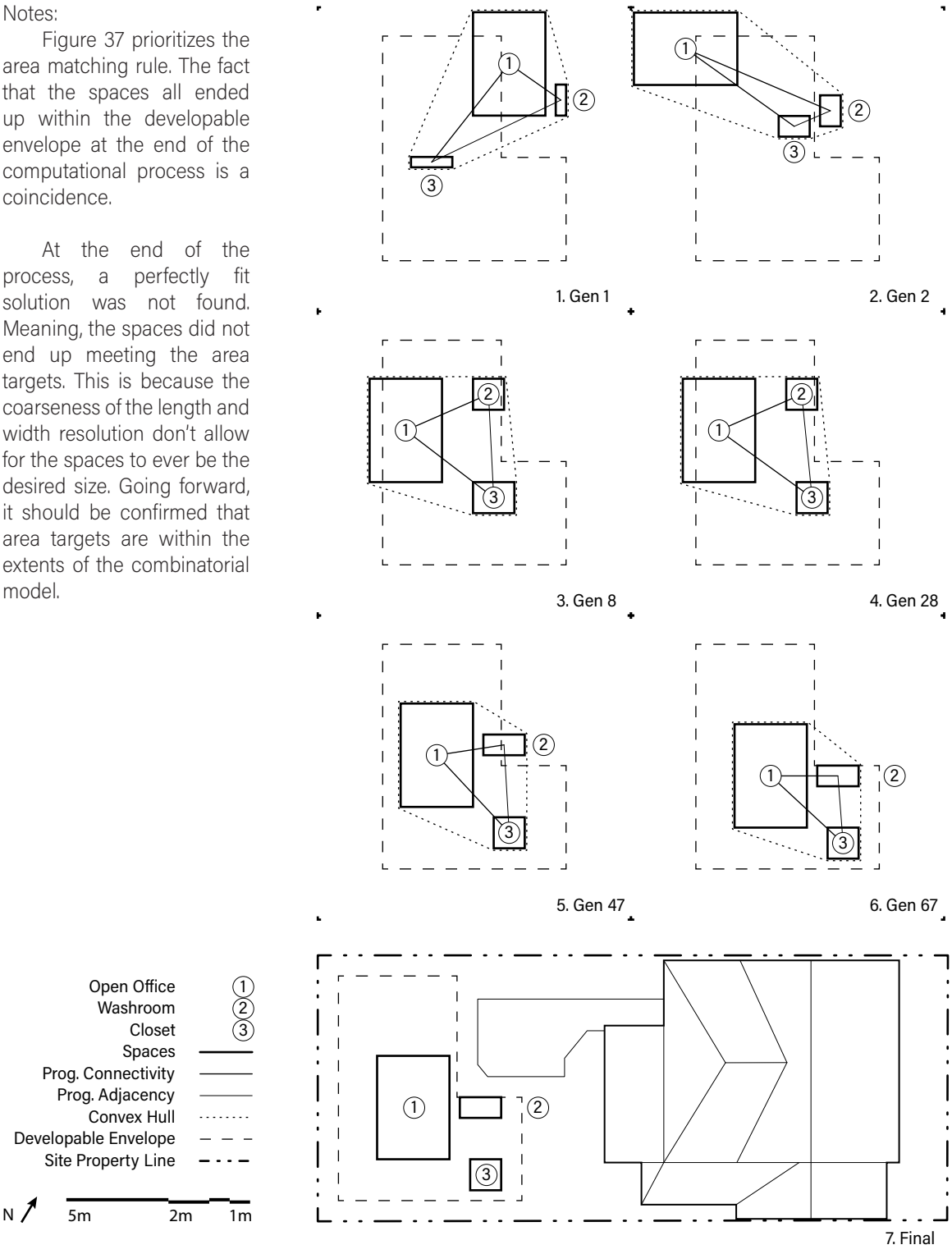


Figure 37.
Prioritizing the area matching rule.

Notes:

Figure 38 prioritizes the proportion matching rule. The script is set to read proportions both horizontally and vertically, meaning that the orientation of a space will not affect it's ratio.

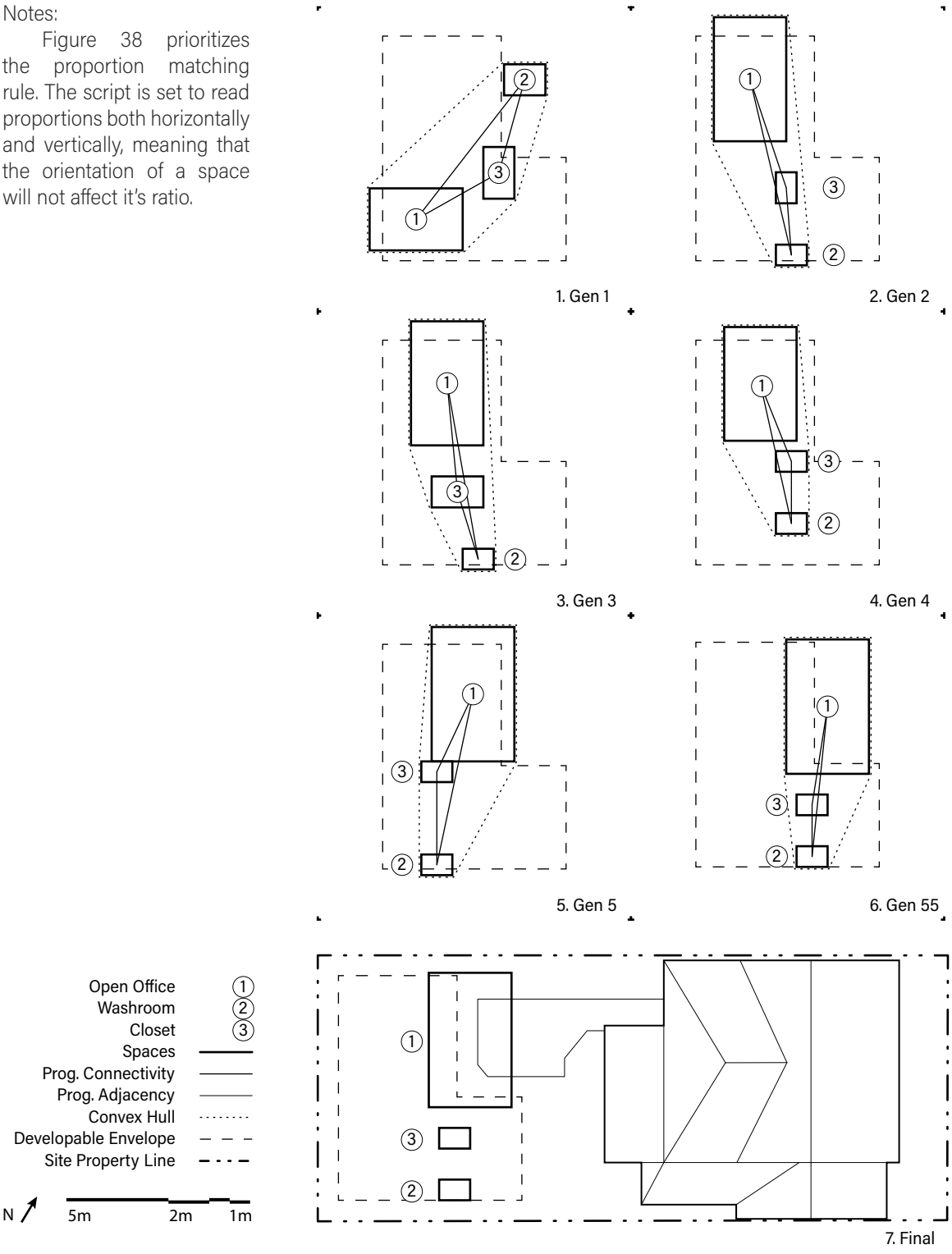


Figure 38.
Prioritizing the proportion matching rule.

Notes:

Figure 39 prioritizes the spatial containment rule. Since the only goal here was to fit all the spaces within the developable envelope, a soluution was found in the first generation. The script considers all the iterations shown equally fit, since there is no other rules being used.

An unforseen side effect of this rule is that the script will shrink spaces to their minumum ddimensions since smaller spaces are more likely to be within the developable envelope.

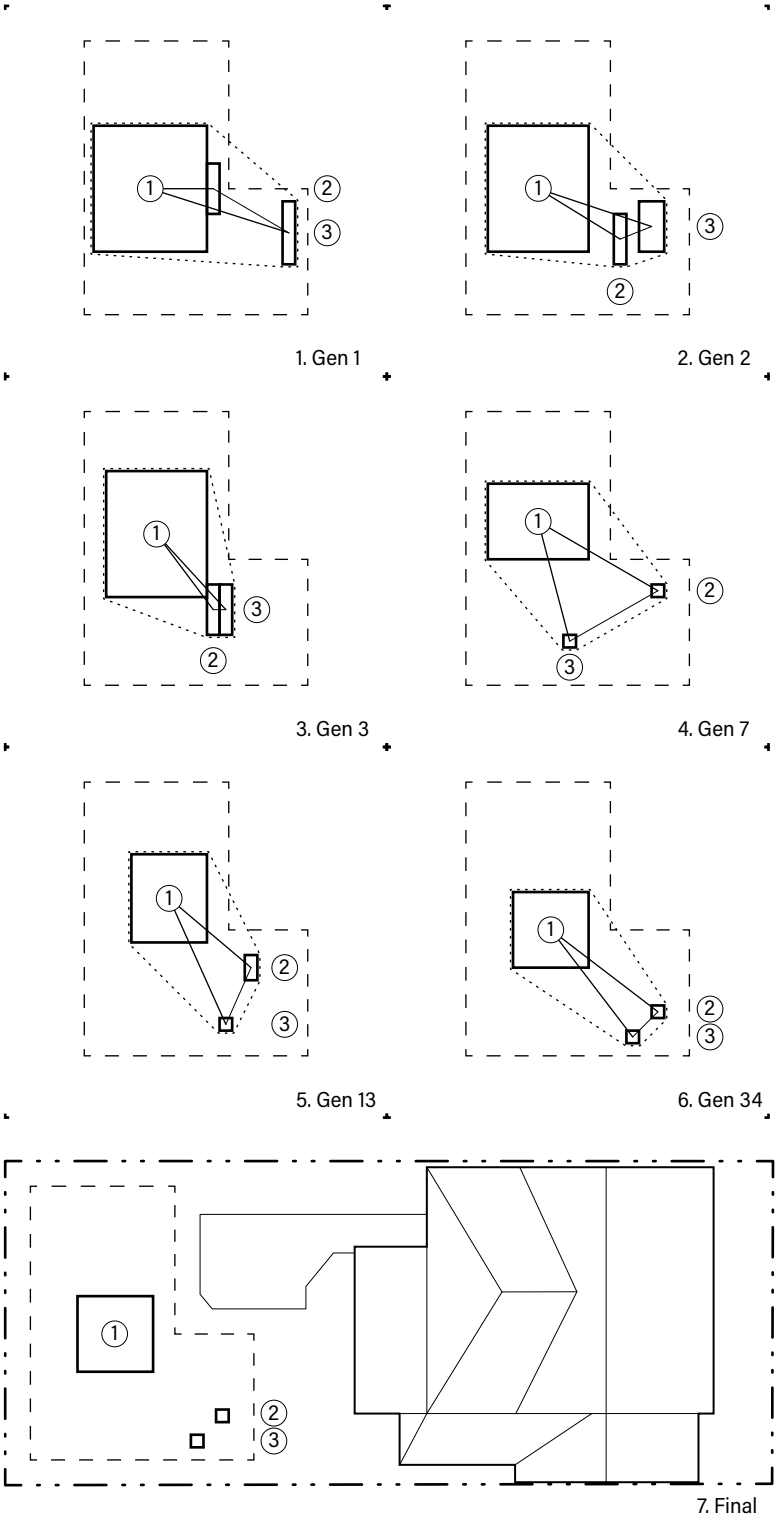


Figure 39.
Prioritizing the spatial containment rule.

Notes:

In Figure 40, the programmatic connectivity rule has been emphasized. An unforseen side effect of distance based rules is that the spaces tend to shrink to their minimum dimensions.

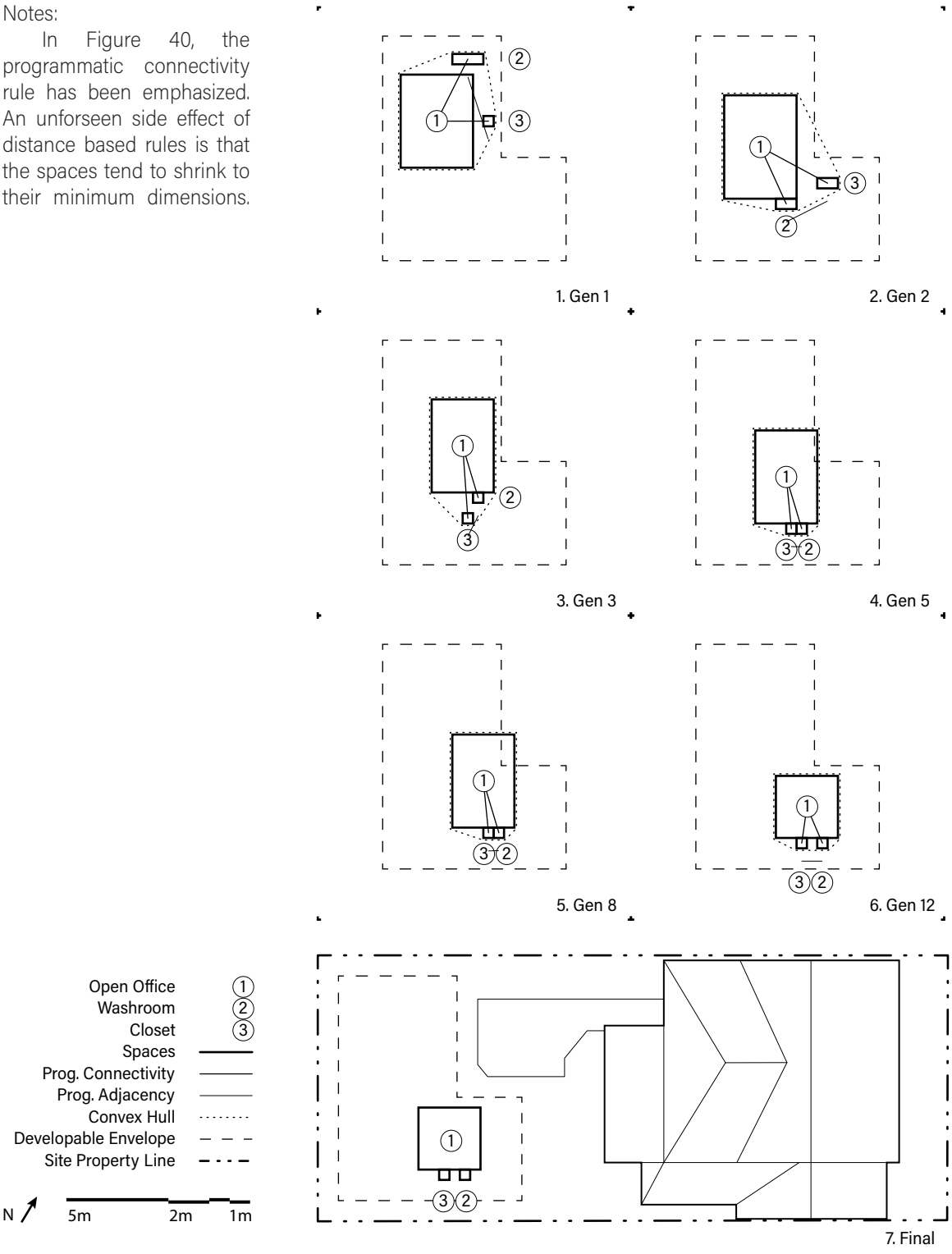


Figure 40.
Prioritizing the programmatic connectivity rule.

Notes:

Figure 41 prioritizes the programmatic adjacency rule. Since the only requirement was for spaces 2 and 3 to be adjacent, the solver found a fit solution almost instantly.

All 6 steps to right are considered perfect solutions by the solver. Interestingly, as the solver continued to iterate even after finding fit solutions, the spaces all shrank to their minimum dimensions.

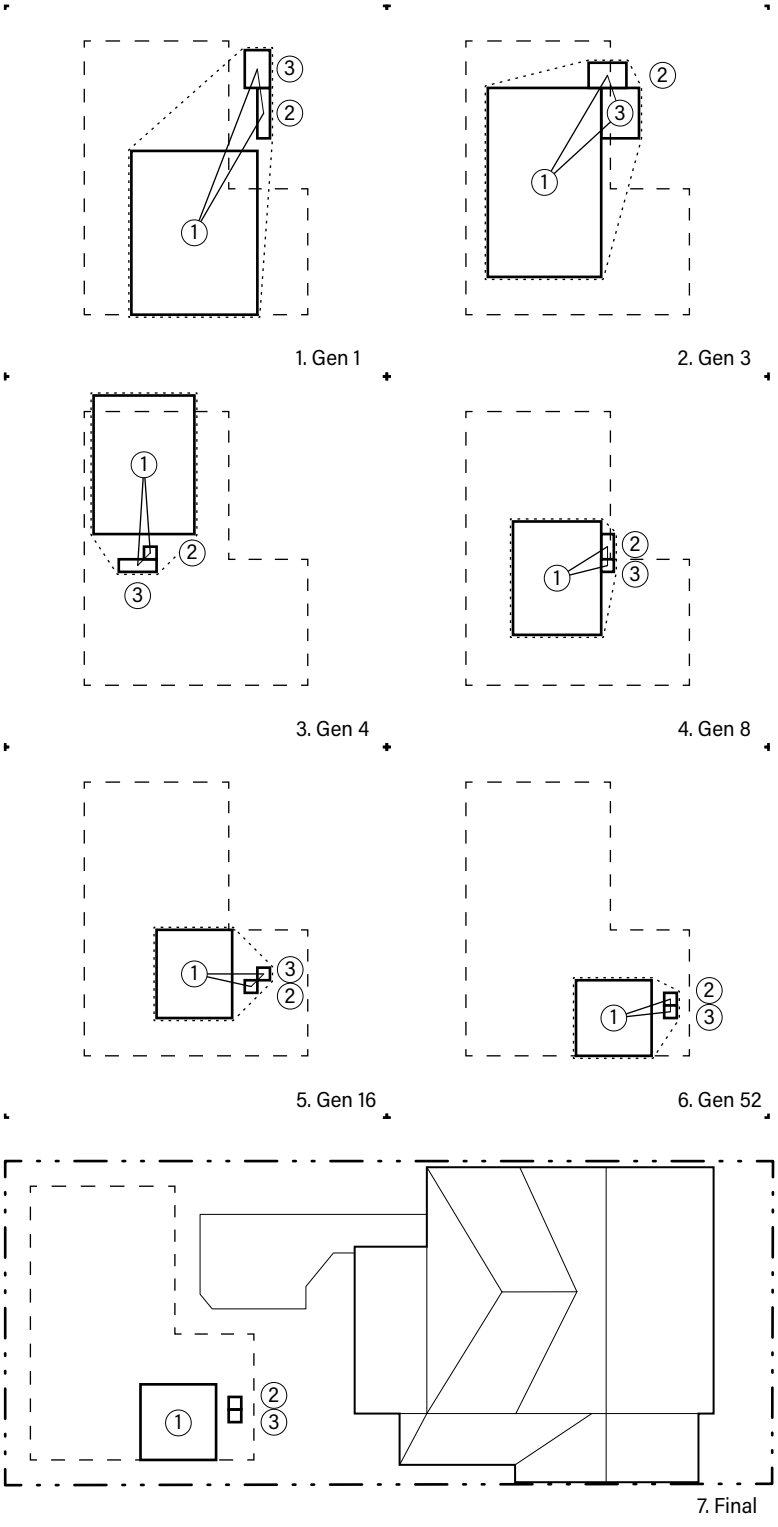


Figure 41.
Prioritizing the programmatic adjacency rule.

Notes:

Figure 42 prioritizes the footprint consolidation rule. Currently, the rule mostly focuses on straightening the convex hull. Perhaps more emphasis should be placed on the differences between the sum of the area of the spaces and the area of the convex hull. This would priotitize a rectangular layout.

This rule seems to constrict how much the spaces can move. It should be lightly weighted, otherwise the variable exploration that tends to occur in the first generations will be heavily restrained. This will prevent the solver from finding fit solutions.

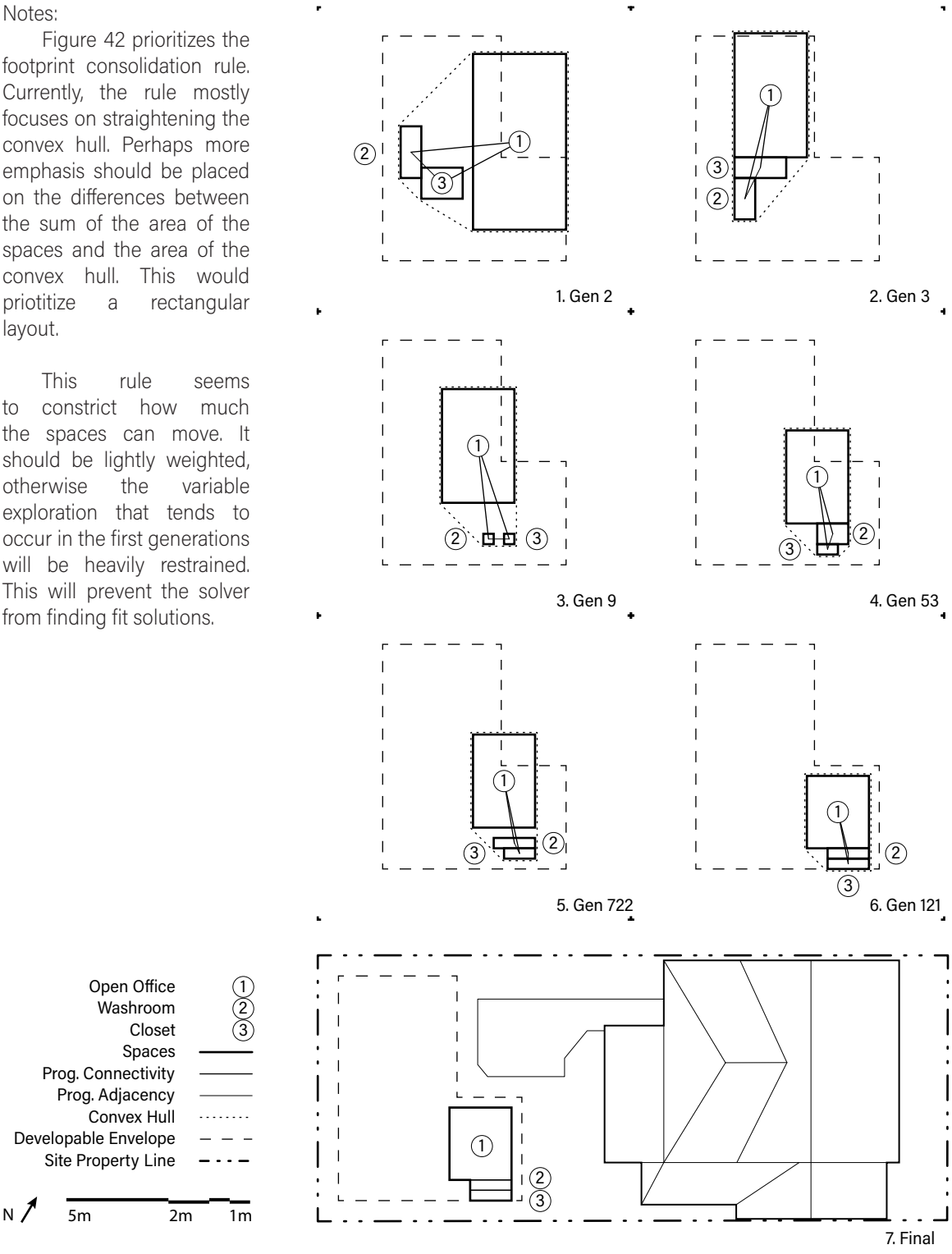
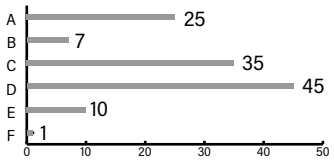


Figure 42.
Prioritizing the footprint consolidation rule.

Notes:

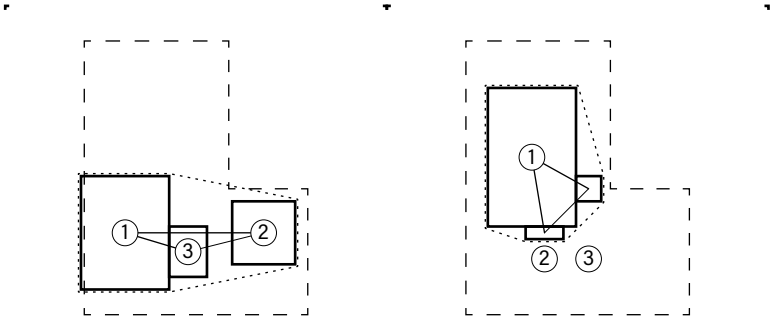
Figures 43 - 45 explore how differently balanced weighting schemes affect the layout.

Figure 43 focuses on area matching and programmatic connectivity while leaving room for flexibility in the proporption and footprint consolidation rules.



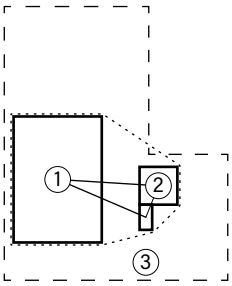
Weighting Values

A. Prog. Conn.	D. Spatial Inclusion
B. Prog. Adj.	E. Proportion Match
C. Area Match	F. Footprint Cons.

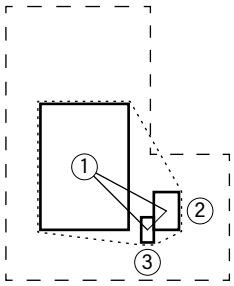


1. Gen 1

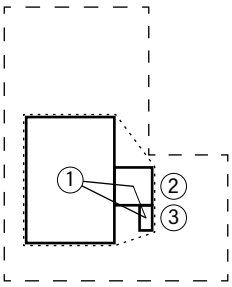
2. Gen 3



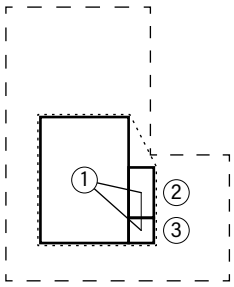
3. Gen 4



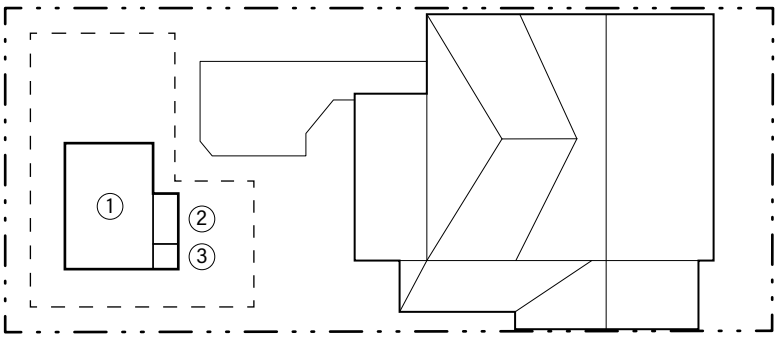
4. Gen 9



5. Gen 11



6. Gen 24



7. Final

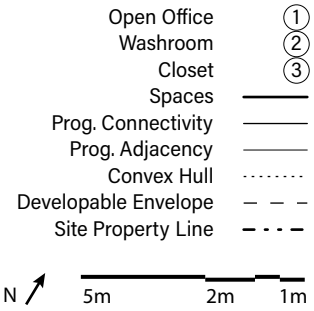
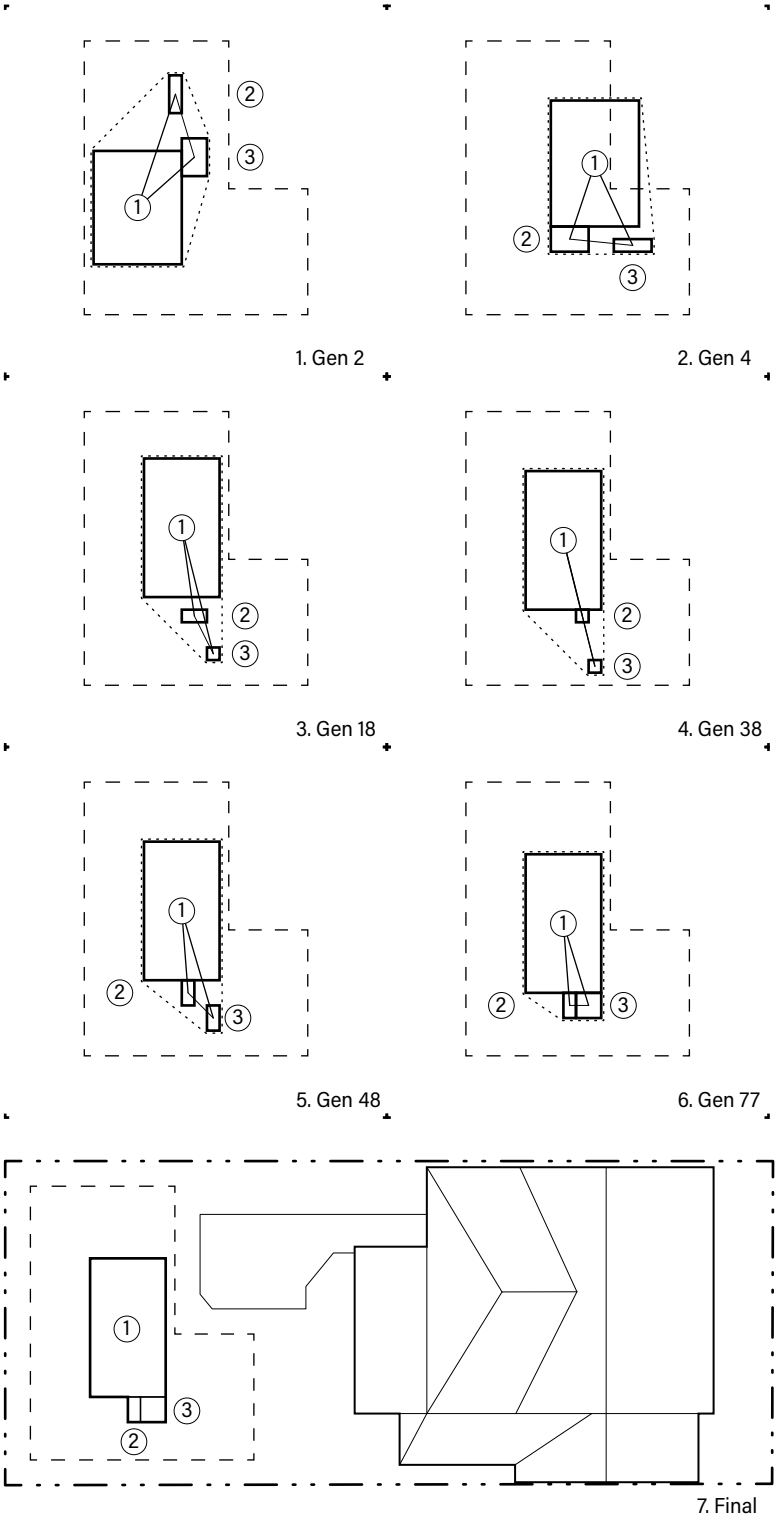
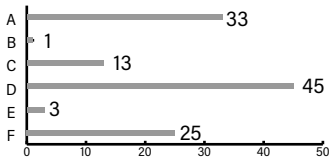


Figure 43.
Balancing the design rules with the weighting scheme to produce different designs.

Notes:

Figure 44 places emphasis on programmatic connectivity and footprint consolidation while leaving the rest of the rules relatively flexible.





Weighting Values

A. Prog. Conn.	D. Spatial Inclusion
B. Prog. Adj.	E. Proportion Match
C. Area Match	F. Footprint Cons.

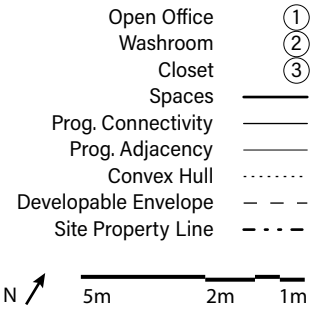
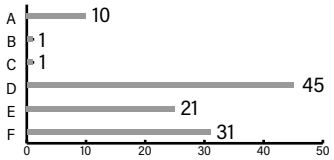


Figure 44.
Balancing the design rules with the weighting scheme to produce different designs.

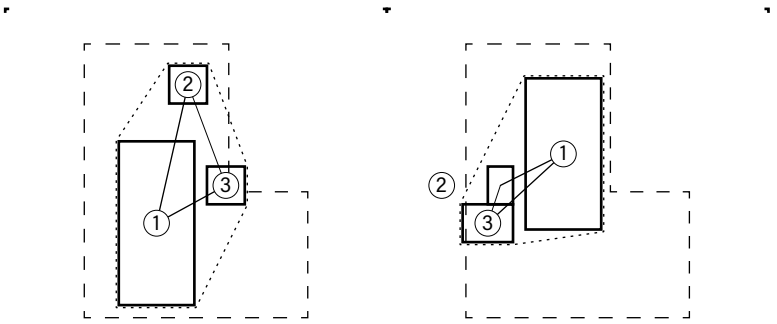
Notes:

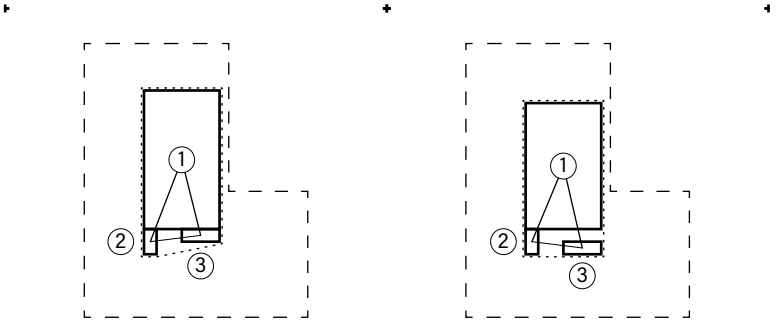
Figure 45 prioritizes proportion matching and footprint consolidation while leaving the rest of the area and adjacency rules at the minimum.

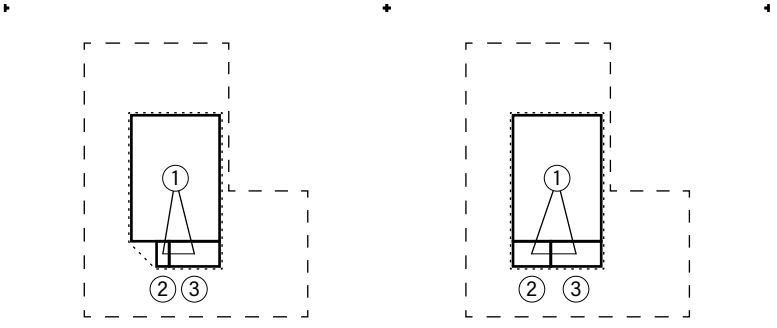


Weighting Values

A. Prog. Conn.	D. Spatial Inclusion
B. Prog. Adj.	E. Proportion Match
C. Area Match	F. Footprint Cons.







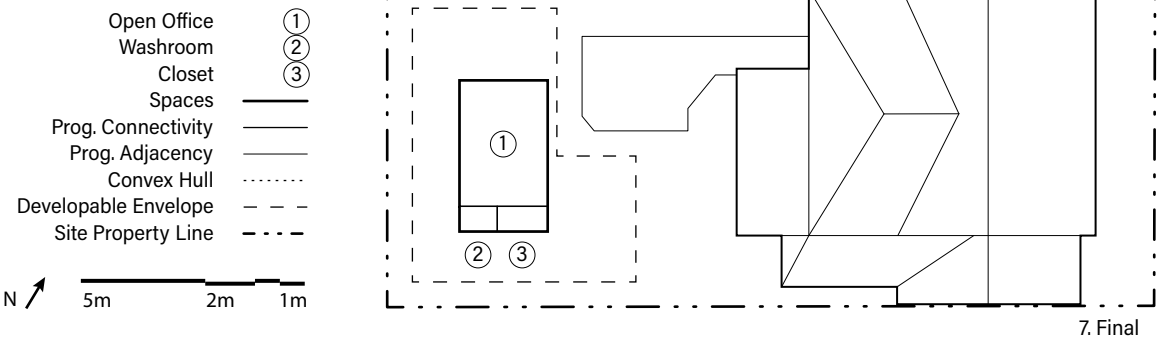


Figure 45.
Balancing the design rules with the weighting scheme to produce different designs.

3.3.4 V1.0 CONCLUSIONS //

At this point in the research, the baseline computational model was functional. Experimenting with the model and reviewing the results was an excellent method for learning to use the model and determining what improvements were needed. Below are the key points from the testing process, these will be used to improve the model in the Version 2.0.

Using the model

1. The spatial overlap rule must be absolutely be prioritized above all others, any flexibility in this rule distorts the fitness of other rules.

2. The ratio rule needs added flexibility to allow for proper design exploration, otherwise the model produces the same design every time. The same applies to the area rule but to a lesser extent.

3. In penalizing the difference in area between the convex hull and the spaces, the footprint consolidation rule has the side effect of constricting how much the spaces can move. Meaning that the footprint consolidation rule will compact solutions before they can develop. As such, it should be weighted very lightly.

Model adjustments

1. The weighting scheme should be further elaborated so that each space and relationship can be individually weighted as a subcategory of each rule. For example, this new scheme would allow for each space's area target to be weighted separately, where as currently all area targets are weighted equally.

2. The model is lacking a rule or a strategy to define the main entrance.

3. The space's current length and width resolutions make it impossible for some of the spaces to meet their area targets. The resolution should be increased so that the targets can be met.

4. The logic for the programmatic connectivity and adjacency rules needs to be adjusted to prevent corner joins, and to better determine which edges should be brought together.

3.4.0 DEFINING PROJECT REQUIREMENTS //

OUTDOOR	▪ Entrance	-
	▪ Door swing	-
	Approx. area:	4 m²
FOYER	▪ Circulation space	-
	Approx. area:	4 m²
WASHROOM	▪ Toilet	750 x 1200
	▪ Sink	750 x 1000
	Approx. area:	3 m²
CLOSET	▪ Samples	-
	▪ Supplies	-
	▪ Storage	-
	Approx. area:	2.25 m²
WORK SPACE	▪ Desk	2100 x 2710
	▪ Built in storage	600 x 2000
	▪ Treadmill	2540 x 1360
	Area + 15% for circulation:	12.0 m²
KITCHEN & MEETING	▪ Meeting table	1960 x 2110
	▪ Kitchen	1500 x 2000
	Area + 20% for circulation:	8.5 m²

Figure 46.1.
Calculating the spatial requirements for the V2.0 based on the client's criteria. All linear measurements are in mm.

While Version 1.0 focused on the technicality and functionality of the model, version 2.0 focuses on quality of what the computational model produces. Essentially, version 2.0 takes the baseline model and refines it to a point where it is capable of producing a variety of distinct designs that both explore the possibilities of the site and meet the client's requirements.

To improve the functionality of the model, the observations detailed in section 3.3.4 were implemented into the current version. To increase the design possibilities of the model, the open office space from version 1.0 was divided, and outdoor and foyer spaces were added.

The open office space was split into two: a work area and a kitchen and meeting area. This allows for the possibility of a public – private division in the plan. To define the main entrance an outdoor area was added. This space could be something small like front steps, or something larger like a porch. Lastly, a foyer was added to support the entrance, serve as an access point to the washroom and closet, and to potentially act as a buffer between the office and meeting spaces.

SUMMARY

▪ Outdoor	4.0 m^2
▪ Foyer	4.0 m^2
▪ Washroom	3.0 m^2
▪ Closet	2.25 m^2
▪ Workspace	12.0 m^2
▪ Kitchen & Meeting area	8.5 m^2
Total area:	**33.75 m^2**

Figure 46.2.
Calculating the spatial requirements for the V2.0 based on the client's criteria. All linear measurements are in mm.

In summary, version 2.0 handles 6 spaces: outdoor area, foyer, washroom, closet, workspace, and the kitchen and meeting area. Figure 46 shows how the area requirement for each space was calculated. The same information provided by the client and described in section 3.3 was used to inform the area calculations. The outdoor area, foyer, washroom, and closet will all be very flexible in their area requirements. As such, their calculated area targets are approximate.

3.4.1 ADDITIONAL DESIGN RULES //

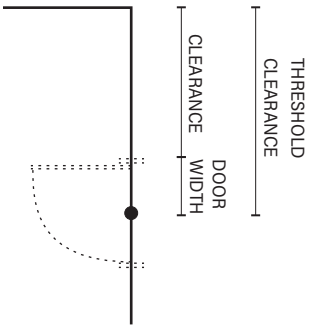


Figure 47.1
Illustrates how the threshold clearance rule is calculated. The door is speculative and represented by the point. The point is the same connection point used for the programmatic adjacency and connectivity rules.

Two additional rules were added to the computational model. The first will be referred to as threshold clearance; it is complimentary to the programmatic connectivity and adjacency rules. The threshold clearance rule prevents corner joins and restrains the joining point between two spaces from being too close to the ends of their respective edges (figure 47.1). The driving concept behind this rule is to ensure that the joining point has enough clearance on either side to physically fit a door or opening. As such, each point has a clearance parameter for the architect to input a value that represents the speculative width of the aperture plus any further clearances.

The second rule is complimentary to the footprint consolidation and programmatic adjacency and connectivity rules. It will be referred to as the single envelope rule. Without taking distances or areas into account, it penalizes the solution that fail to have singular footprints. In other words, the rule encourages the computational model to produce solutions where all the spaces are connected or adjacent. This rule was added to supplement the footprint consolidation, adjacency, and connectivity rules.

3.4.2 IMPROVED WEIGHTING SYSTEM //

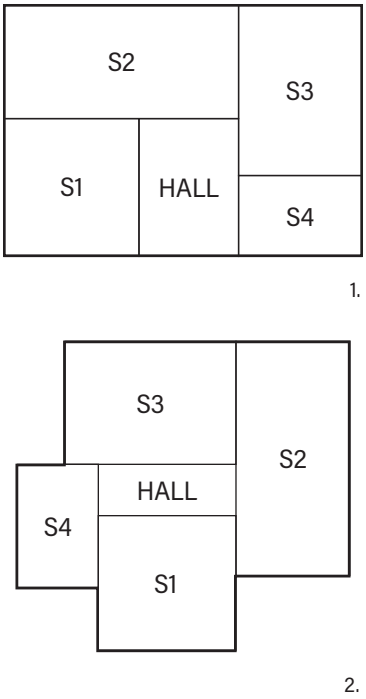


Figure 47.2
All the spaces except the hall have strict area and ratio targets. This allows the hall to adapt to specific design conditions.

The improved weighting system in version 2.0 allows the effect of each rule to be calibrated individually for each space. For example, the hallway can have a very flexible area target, while the other spaces have very strict area targets. This makes the hallway malleable, allowing for the other rules to adapt it to the specific conditions of the design (figure 47.2). This extended scheme allows the architect to specify the characteristics of a fit design with much greater detail by controlling which spaces have negotiable qualities.

3.4.3 RULE SPECIFICS //

This section will go over the specific conditions and assumptions used to determine the targets for each design rule and the resolution of the combinatorial model. As most of the conditions are unchanged from version 1.0, only the new conditions will be covered. Figure 48 shows a summary of this section.

Area and ratio targets

The area target for the outdoor space was set to comfortably accommodate a small table and chair, giving the client the option to work outside. The foyer's area target was set to comfortably accommodate 4 people arriving at once. For the storage space and the washroom, the area targets were determined by asking the client about their storage needs, and if they wanted a shower in the washroom. The workspace's area target was determined by adding together all the spatial requirements of the furniture as specified by the client, plus 15% of the total area for circulation. The area target for the kitchen and meeting area was also determined by totaling the spatial requirements as specified by the client. However, since it is a space meant to accommodate multiple people, an additional 20% was added to the total area to account for circulation.

Programmatic adjacency and connectivity

The washroom and storage space were set to have an adjacency requirement between each other. The foyer was set to have a connectivity requirement to all other spaces.

Threshold clearance

The threshold clearance rule was set to account for all doors being 914mm wide, and for an additional ___ mm of clearance from corners.

Single envelope

The single envelope rule was set to prefer layouts with a single envelope, layouts where the spaces are all grouped together.

	Outside Area	Foyer	Washroom	Closet	Workspace	Kitchen & Meeting Area
Area Targets	4.0 m^2	4.0 m^2	3.0 m^2	2.25 m^2	12.0 m^2	8.0 m^2
Proportion Targets	1.6	2	1.8	1.2	1.6	1.6
Spatial Containment	Within Developable Envelope					
Overlap Prevention	No Overlap Allowed					
Programmatic Adjacency	None	None	Closet	Washroom	None	None
Programmatic Connectivity	Foyer	All	Foyer	Foyer	Foyer	Foyer
Threshold Clearance	Assume 1100 mm door + 20m mm clearance from corners					
Footprint Consolidation	Prefer Rectilinear					
Single Envelope	Prefer Single Envelope					
Width Domain	1.0m - 4.0m	1.0m - 4.0m	1.0m - 2.5m	0.5m - 2.0m	2.5m - 7.0m	2.5m - 7.0m
Length Domain	1.0m - 4.0m	1.0m - 4.0m	1.0m - 2.5m	0.5m - 2.5m	2.5m - 7.0m	2.5m - 7.0m
Width Resolution	0.5 m		0.25 m		0.5 m	
Length Resolution	0.5 m		0.25 m		0.5 m	
Position Resolution	0.5 m		0.25 m		0.5 m	

Figure 48.
Summary of the rule specifics covered in section 3.4.3.

3.4.4 SCHEMATIC PROPOSALS //

Compared to version 1.0, version 2.0 of the computational model offers much finer control of how the rules affect the spaces in the layout. This allows for many different scenarios, each prioritizing different rules and varying the flexibility of each space to be set up. With this new amount of control, the script was ready to be used for schematic exploration. The following section illustrates some of the many scenarios that were explored.

The following section uses version 2.0 of the computational model to explore the project's design possibilities while meeting the client's requirements. A testing phase like in section 3.3.3 was not needed due to the similarities in data management between version 1.0 and 2.0. As such, figures 49 - 54 are a collection of handpicked outputs from the model, intended to show the range of its capabilities.

The final iteration, shown on the site, is an interpretation of the script's final output. This step shows how the architect can adapt ideas brought forth through during the computational process to add features outside of the computational model's scope.

Notes:

Figure 49 prioritizes programmatic relationships and has moderate ridigidity in area targets while being very flexible with respect to proportions.

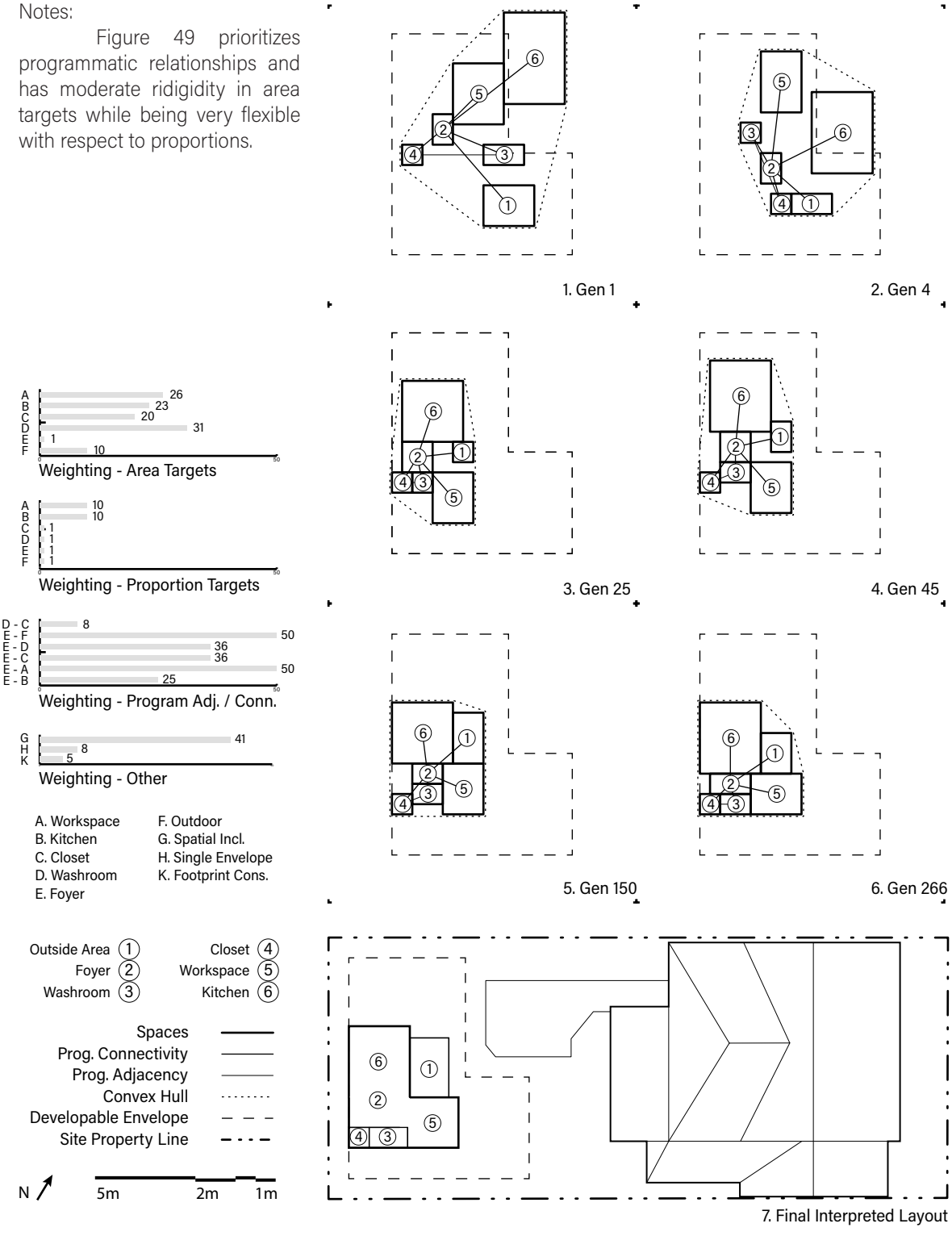
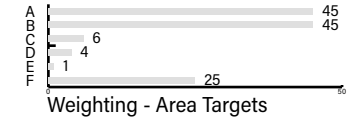
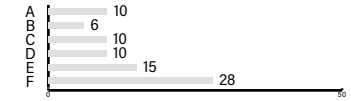


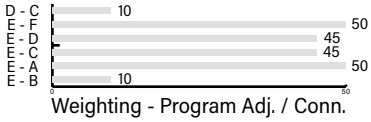
Figure 49.
Schematic iterations, interpretations, and associated weighting values.

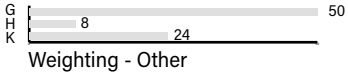
Notes:

Figure 50 prioritizes programmatic relationships while remaining relatively flexible in all other rules.

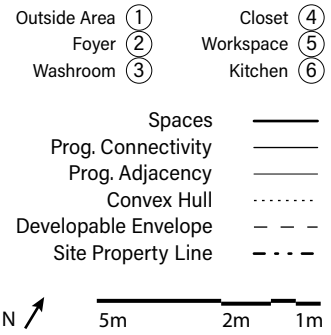


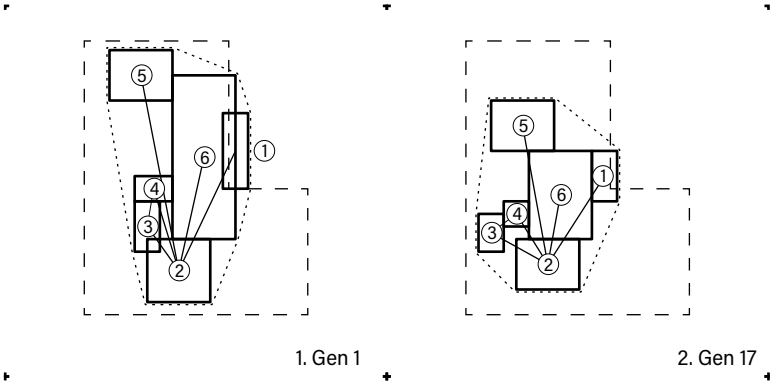


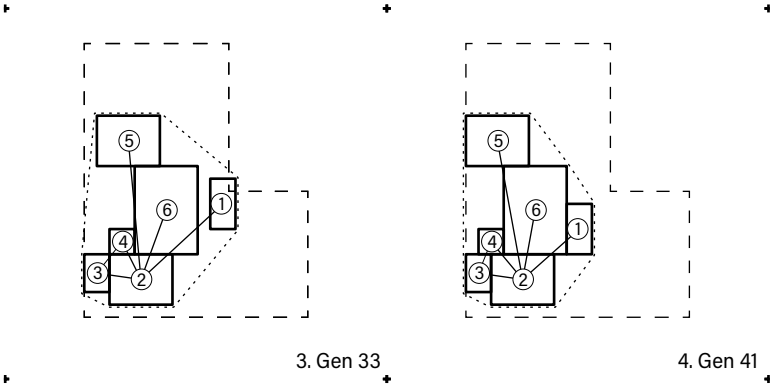


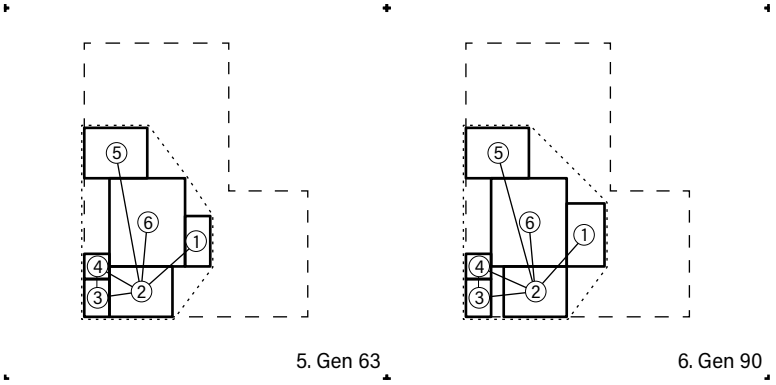


A. Workspace
B. Kitchen
C. Closet
D. Washroom
E. Foyer
F. Outdoor
G. Spatial Incl.
H. Single Envelope
K. Footprint Cons.









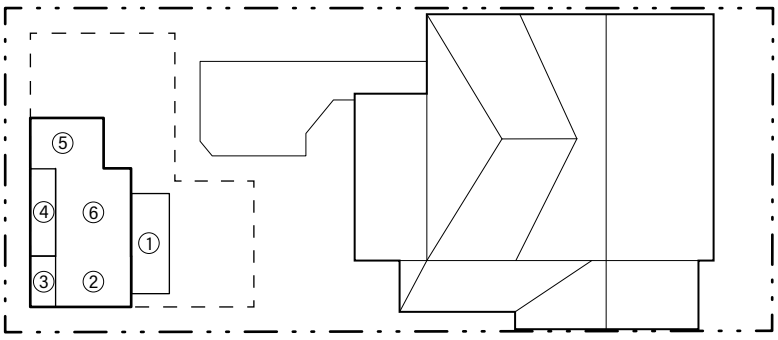
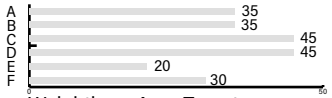


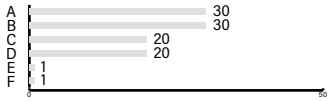
Figure 50.
Schematic iterations, interpretations, and associated weighting values.

Notes:

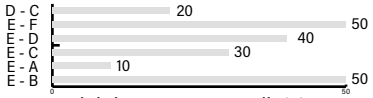
Figure 51 has a heavy emphasis on all the targets except for those corresponding to the foyer and the outdoor area.



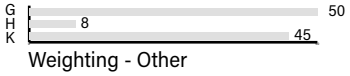
Weighting - Area Targets



Weighting - Proportion Targets



Weighting - Program Adj. / Conn.



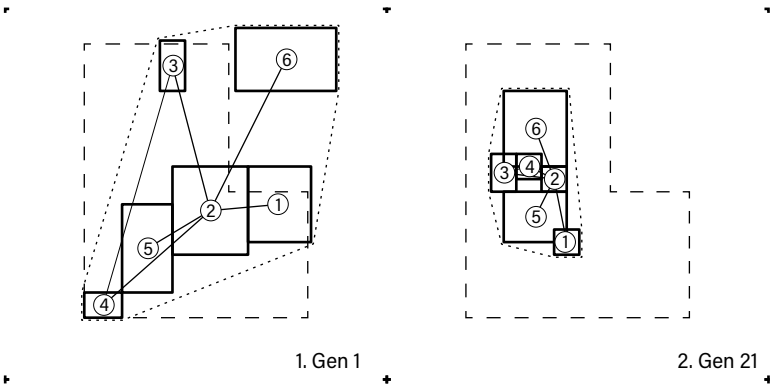
Weighting - Other

A. Workspace
B. Kitchen
C. Closet
D. Washroom
E. Foyer
F. Outdoor
G. Spatial Incl.
H. Single Envelope
K. Footprint Cons.

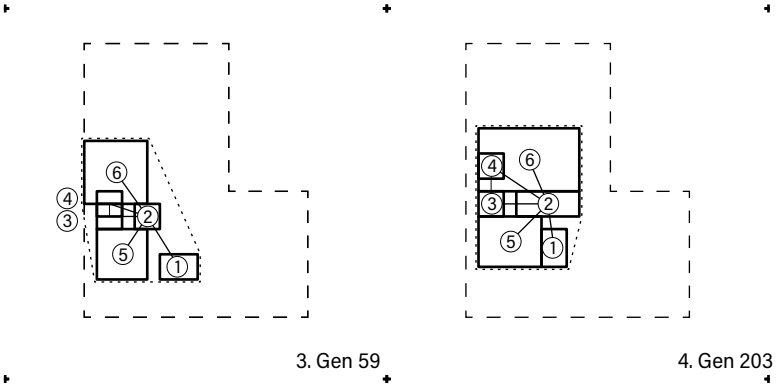
Outside Area 1
Foyer 2
Washroom 3
Closet 4
Workspace 5
Kitchen 6

Spaces
Prog. Connectivity
Prog. Adjacency
Convex Hull
Developable Envelope
Site Property Line

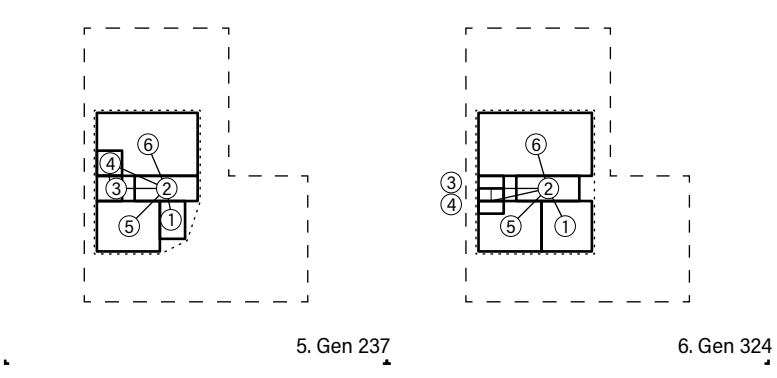
N
5m 2m 1m


1. Gen 1

2. Gen 21


3. Gen 59

4. Gen 203


5. Gen 237

6. Gen 324

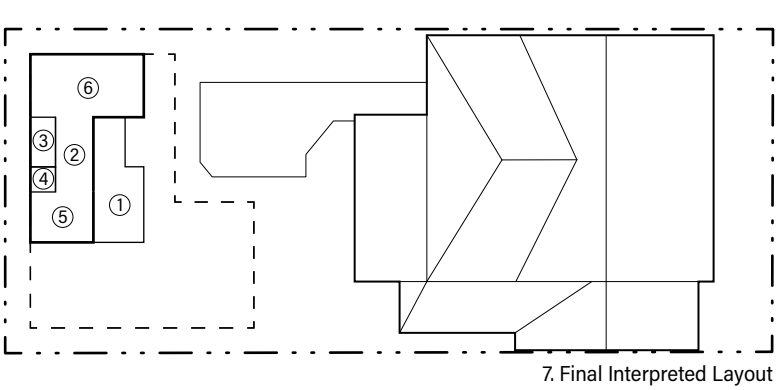
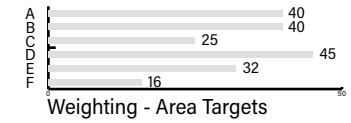

7. Final Interpreted Layout

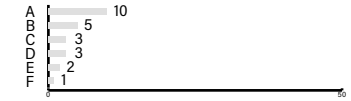
Figure 51.
Schematic iterations, interpretations, and associated weighting values.

Notes:

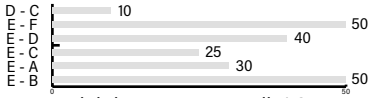
Figure 52 emphasizes areas and programmatic connections, while being lenient with proportions and footprint consolidation.



Weighting - Area Targets



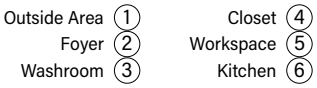
Weighting - Proportion Targets

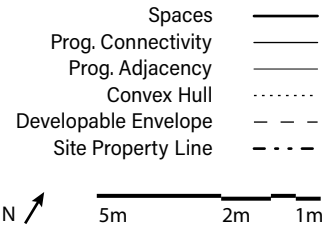


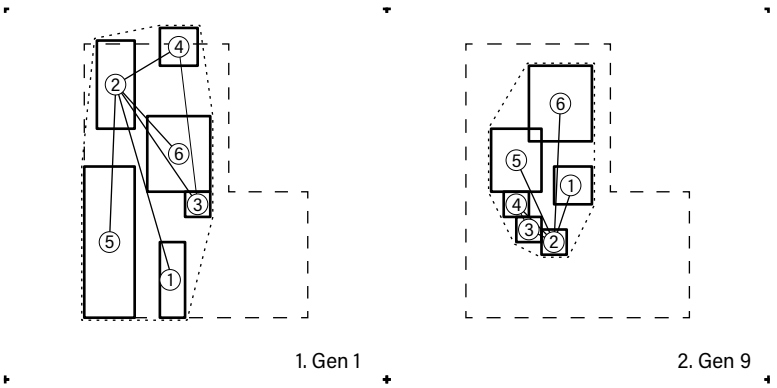
Weighting - Program Adj. / Conn.

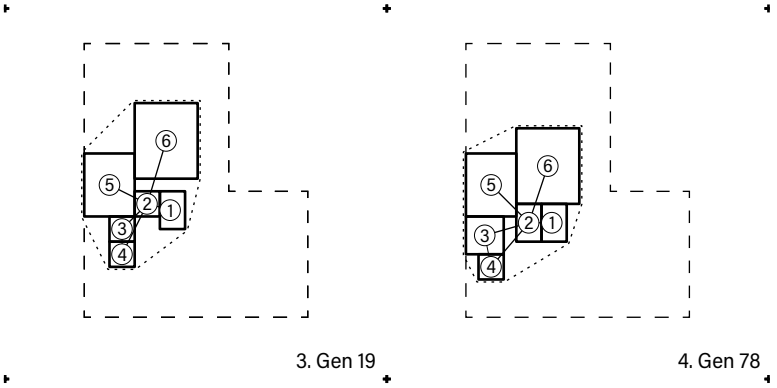
Weighting - Other

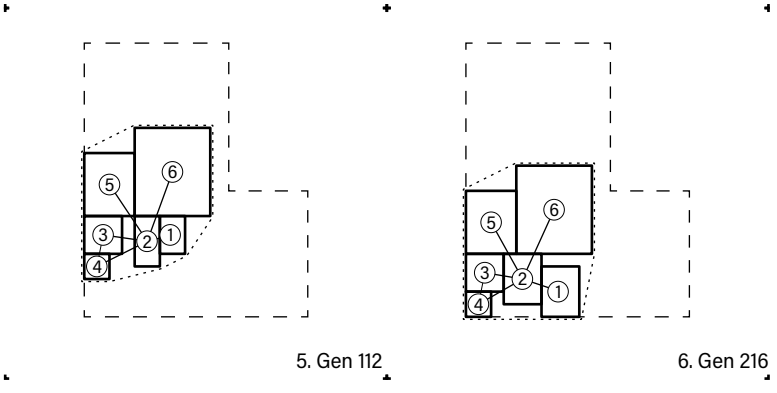
A. Workspace
B. Kitchen
C. Closet
D. Washroom
E. Foyer
F. Outdoor
G. Spatial Incl.
H. Single Envelope
K. Footprint Cons.











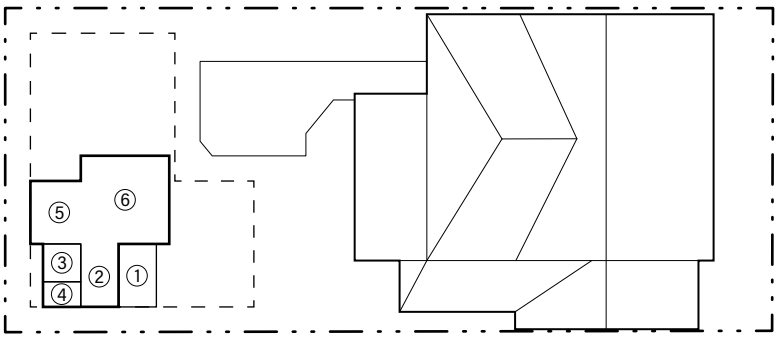

7. Final Interpreted Layout

Figure 52.
Schematic iterations, interpretations, and associated weighting values.

Notes:

Figure 53 prioritizes areas, proportions, and footprint consolidation while having some flexibility in programmatic connections.

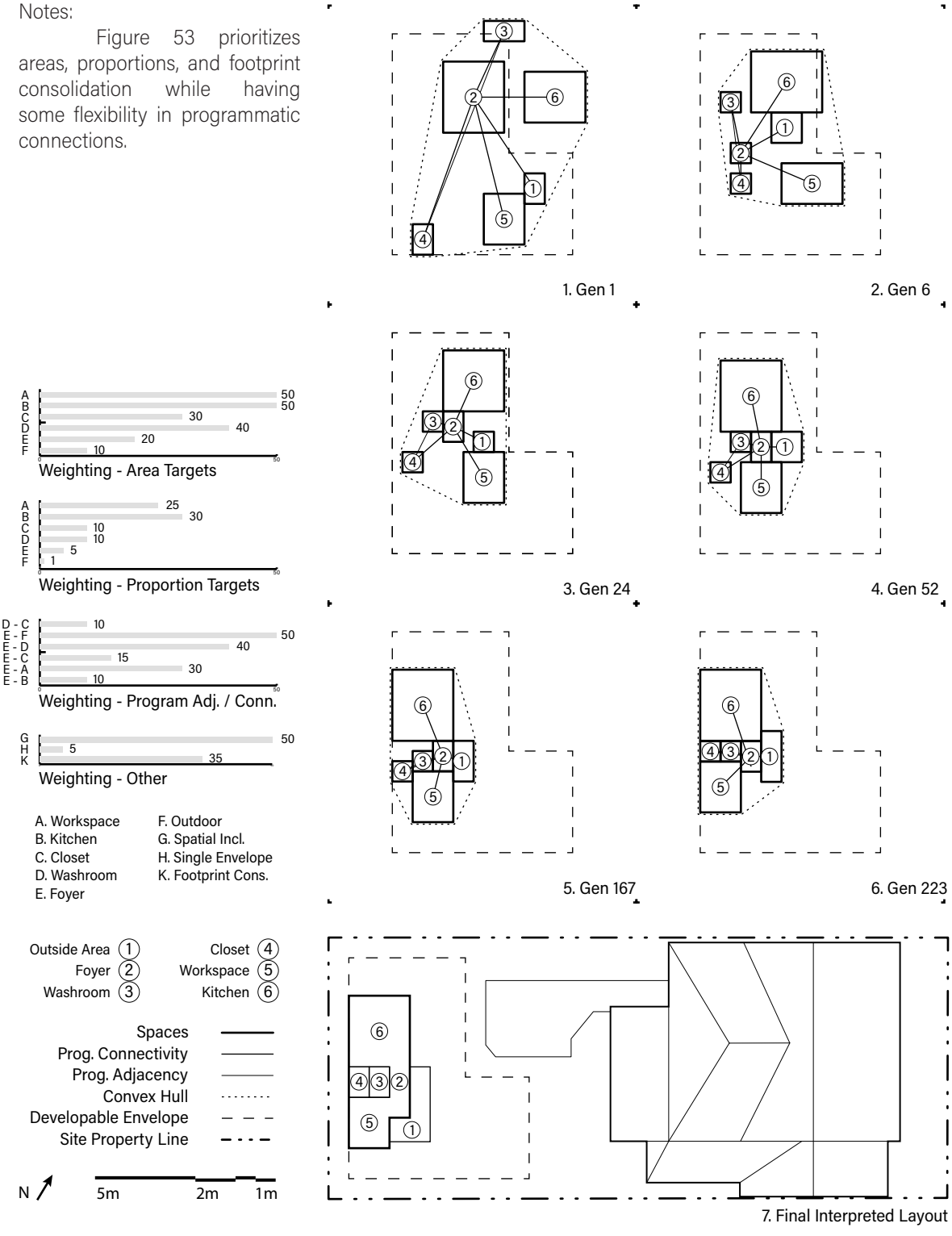
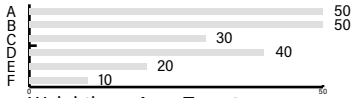


Figure 53.
Schematic iterations, interpretations, and associated weighting values.

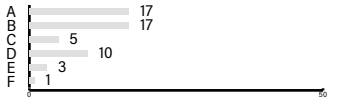
Notes:

Figure 54 prioritizes area targets, and programmatic connections while allowing flexibility in the space's proportions.

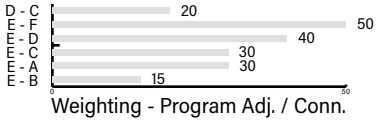
The overlaping closet and kitchen space in the final iteration indicates that the spatial overlap rule needs to be weighted heavier, or that the footprint consolidation rule needs to be lightened



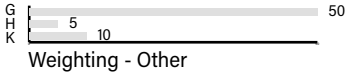
Weighting - Area Targets



Weighting - Proportion Targets



Weighting - Program Adj. / Conn.



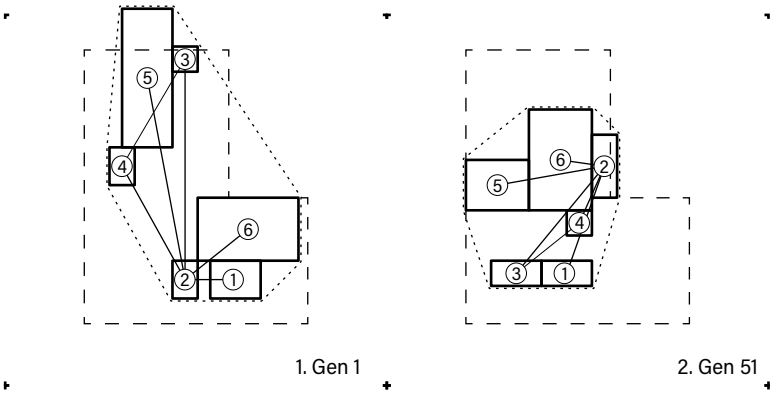
Weighting - Other

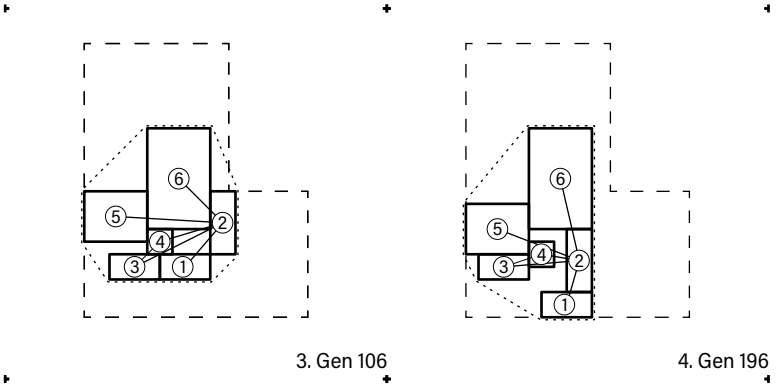
A. Workspace
B. Kitchen
C. Closet
D. Washroom
E. Foyer
F. Outdoor
N. Spatial Incl.
O. Single Envelope
P. Footprint Cons.

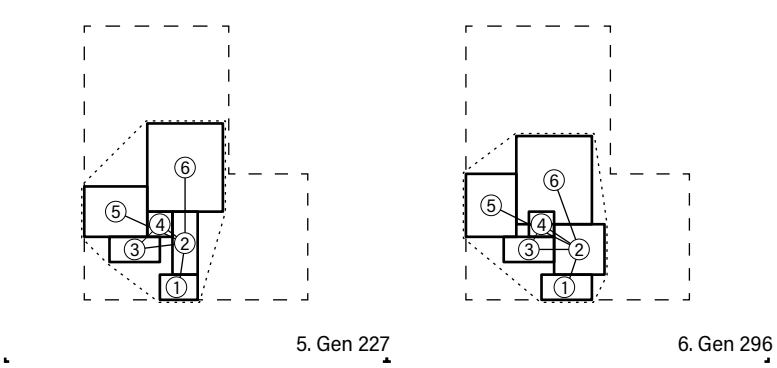
Outside Area 1
Foyer 2
Washroom 3
Closet 4
Workspace 5
Kitchen 6

Spaces
Prog. Connectivity
Prog. Adjacency
Convex Hull
Developable Envelope
Site Property Line

N
5m 2m 1m







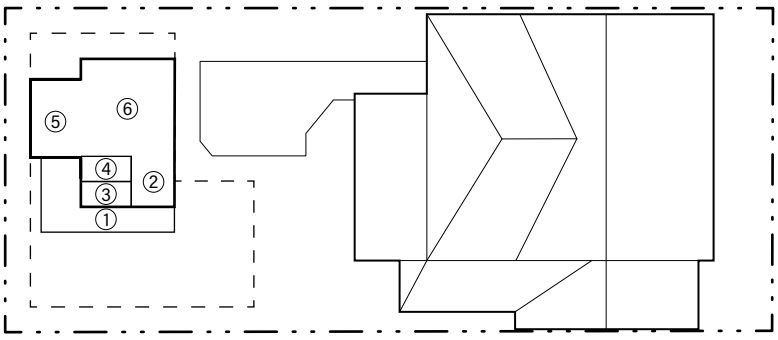

7. Final Interpreted Layout

Figure 54.
Schematic iterations, interpretations, and associated weighting values.

3.4.5 COMPARISONS & METADATA //

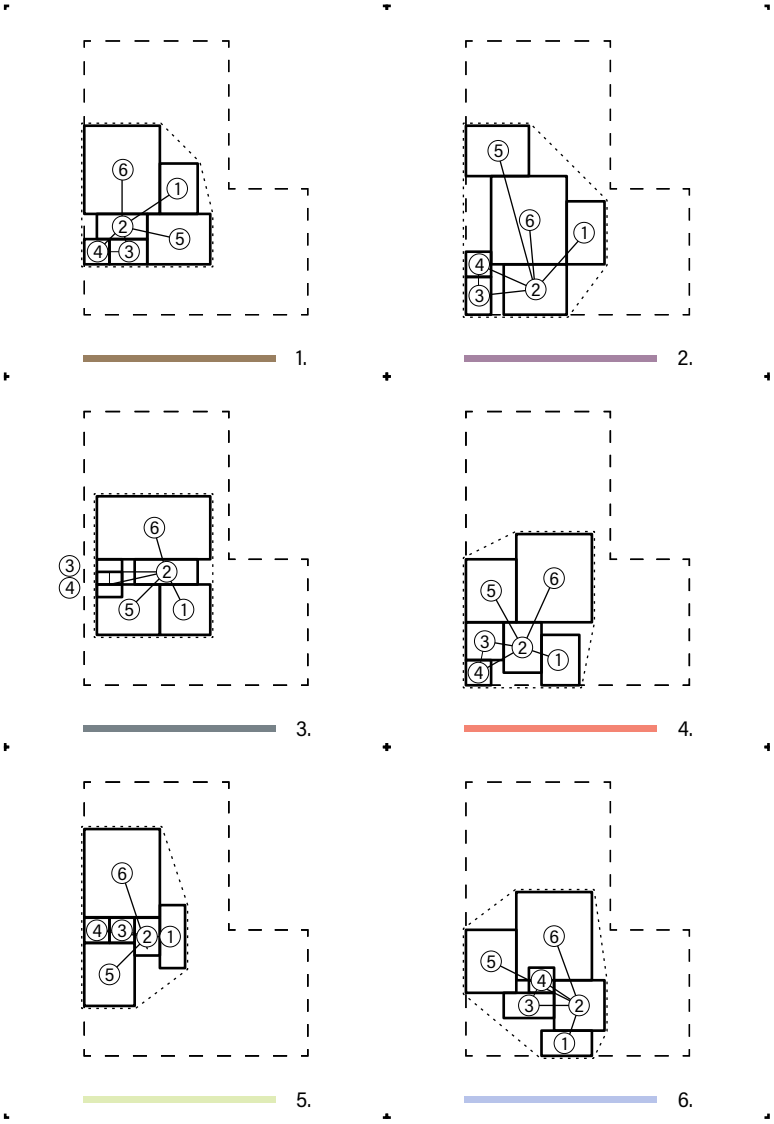
For each design the computational model produces, an accompanying set of metadata is also generated. The metadata is useful for comparing multiple designs to each other. Figure 55 shows geometric and data comparisons of the six explorations from section 3.4.4.

The metadata is a numerical description of the designs. It includes the fitness, dimensions, locations, areas, targets, and weighting of the overall design and of each space. With the metadata, the architect can quickly sort designs based on specific qualities. For example, if the architect wants to only look at solutions where the area of the workspace is between 10 m^2 and 15 m^2, it is simply a question of setting up a filter.

Notes:

If multiple powerful computers are available, the script could be distributed amongst them to produce multiple solutions simultaneously. Since the metadata provides a numerical, non-geometric view of each design, the number of options the architect can consider at once increases dramatically. The combination of rapid exploration through distributed computing and metadata for easily comparing large amounts of designs would allow the architect to explore the schematic possibilities of project very quickly.

The chart on the bottom right shows the weighting values used for each design. Different priorities in the computational model produce different layouts. As the weighting value increases, the degree to which the relevant design rule must be met increases.



A. Workspace
B. Kitchen
C. Closet
D. Washroom
E. Foyer
F. Outdoor
G. Spatial Incl.
H. Single Envelope
K. Footprint Cons.

Outside Area 1
Foyer 2
Washroom 3
Closet 4
Workspace 5
Kitchen 6

Spaces
Prog. Connectivity
Prog. Adjacency
Convex Hull
Developable Envelope
Site Property Line

N
5m 2m 1m

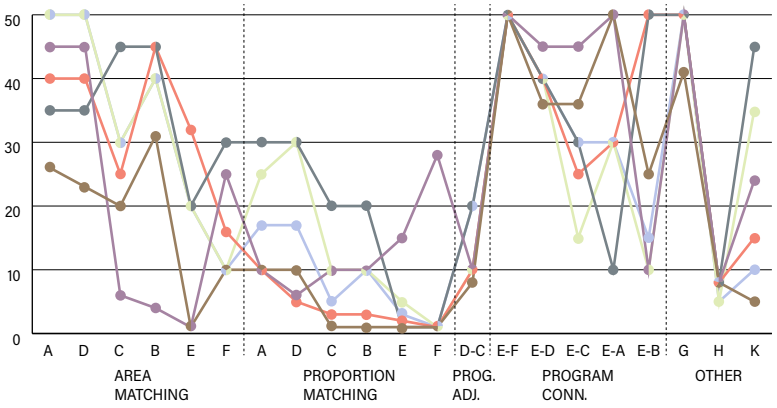


Figure 55.
Geometric and data comparisons of the schematic proposals.

Notes:

Figure 57 shows a high-level fitness summary, it is useful for comparing multiple spaces in a non-geometric manner. As the quality of the solution increases, its fitness approaches 0.

Parts 1-9 are partial fitness values, showing the degree to which each design rule has been met. Part 10 shows the total fitness for each design.

Only the final iterations from figure 55 have been taken into consideration. These graphs should be read in conjunction with the weighting graph from figure 56. As the weighting values change, the fitness of a design also changes. If the weights allow for more flexibility, the design's fitness improves.

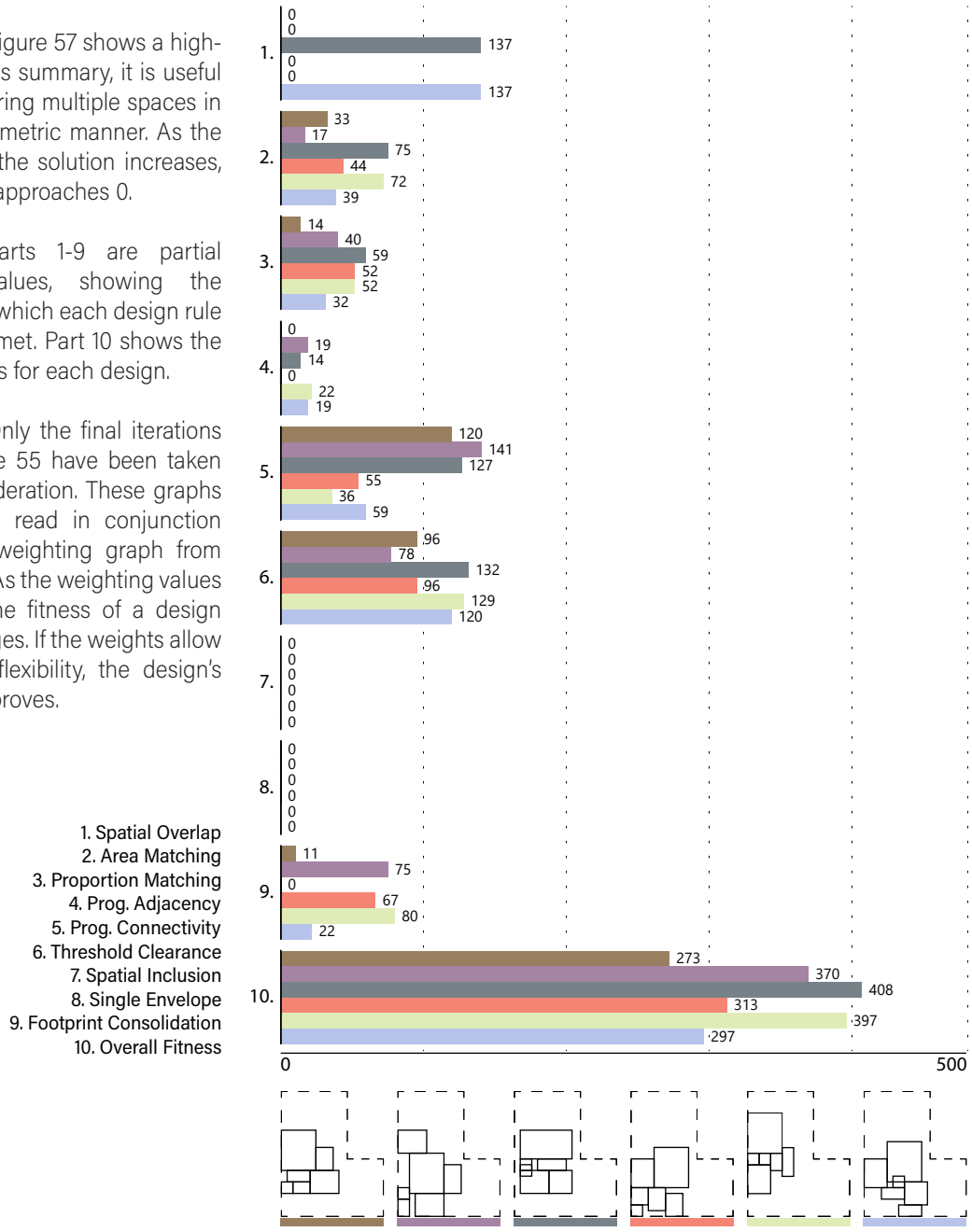


Figure 57.
Fitness summary of the schematic proposals.

3.4.5 V2.0 CONCLUSIONS //

The adjustments made to the computational model between version 1.0 and 2.0 make it capable of producing designs that explore the possibilities of the project while meeting the established constraints. The following section reflects on the computational process as experienced while testing version 2.0 of the model. It covers the challenges faced in this specific phase of the thesis and suggests improvements that could be made to the model.

Using the model

1. Grasshopper was selected for its user interface, however, the interface itself eventually becomes a limiting factor. The inability to set up routines ties the user's presence to the productivity of the process. Meaning that the user needs to be present to start, end, and save each run of the computational process. This prevents one from, for example, leaving the machine to compute multiple runs of the script overnight.

2. As expected, the runtime for version 2.0 was longer than its predecessor due to the increased number of variables. However, the runtime varied tremendously. The solutions computed in figures 49 – 54 took between

20 – 90 minutes. There seemed to be no guarantee of design quality the longer the script ran for.

Model adjustments

1. The computational model is lacking a rule or strategy that situates the project on the site. Something that picks where within the developable envelope the project should lie.

2. The model is also lacking a rule to line up spatial divisions within the footprint of the project.

3. Changing the programmatic relationships between spaces is an arduous process where it is easy to make mistakes. It would be beneficial to try and simplify this process.

Technical difficulties

As stated at the beginning of the thesis, the outputs from the model are meant to spark ideas, to initiate conversations about the directions the project could take, all while providing layouts and data as starting points. Results from testing the model have confirmed this expectation in both the final outputs and

in the progress iterations.

The final outputs are layouts that best meet the requirements outlined by the architect. However, regardless of how rigid the design rules are set and how suitable the requirements seem on paper, they will occasionally fail to develop cohesive solutions when computed. As the examples from figures 49 - 54 show, the script rarely produces flawless solutions. There's almost always spaces that could be better aligned, spaces that should be rotated, or spaces that are slightly overlapping. This can be fixed by adjusting the existing logic, adding more design rules, and with more careful weighting in the computational model.

Yet, in the progress solutions when the model is iterating, temporary serendipitous design states rise and fall. In these temporary states, designs that break the rules are generated. Occasionally, these solutions show ideas that are outside the set constraints. For example, during testing a design was observed where the closet was between the outside space and the foyer. Since the outside space is used to define the entrance, this begged the question, what if the closet were a mudroom?

Typically, as the model continues to iterate these designs are replaced by ones with higher fitness, designs which conform to the rules.

During the iterative process such a vast amount of solutions are generated that it is very difficult to catch any novel rule-breaking designs. This task becomes impossible if the script is being run on multiple computers at once. Furthermore, the amount of data produced by the computational process makes it difficult to search through it. On average, version 2.0 of the script would compute between 200-400 generations of solutions per run. With each generation having 100 individual solutions, and each solution having at least 10 metadata items, the total number of values from each computational process would range between 200,000 – 400,000 per run. This amount of data is painfully slow to process in Grasshopper, and even starts to push the limits of Excel spreadsheets. If the computational model were developed further, the increased complexity would eventually exceed the limits of Excel. To develop a truly effective computational model, it would need to be made with custom software capable of addressing these issues.

CONCLUSIONS //

10,000 iterations began with the intent of exploring the different possibilities of a project without ignoring its systemic context. Specifically, the thesis sought to generate function driven layouts: layouts focused on spatial organization and iteration developed through computational methods.

To achieve this, the thesis went through three stages: research, script development, and script refinement. As the research phase progressed, it became clear that the most significant challenge of using generative methods would be describing design logic in computational terms. Accordingly, the rest of the thesis explored the issue through the development and refinement of the computational models.

Through the course of this thesis project, it became increasingly evident just how complex design, as a process, is. Defining the functional aspects of design, such as spatial proximities or sizes is a straightforward matter since the objective is clear. However, the question of defining or explaining creative design, design that goes beyond the functional, persists and perhaps is something that cannot be defined computationally. The topic was

explored through the incorporation of the weighting scheme. By giving the computational model the ability to compromise between design elements, it begins to approach design that goes beyond the functional.

Although the thesis used a specific strategy to represent spatial planning: the organization and development of rectangles representing spaces, the solutions that were discarded for not conforming to the framework have their own value if seen under different criteria. These outliers could be interpreted as parti models under a different exploratory strategy. For example, instead of defining spaces as rectangles, one could have free floating walls and define programs as the spaces in between the walls. This strategy could be used to produce more innovative or unique designs, but also requires more involvement and interpretation from the architect. Namely, defining the fitness of this type of layout is significantly more challenging than what was attempted in the thesis. Meaning that perhaps the only way to find these promising outliers is to manually search through the tens of thousands of iterations produced each time the script is run. At that point, the goal of using computational methods for exploration is lost.

There is also something to be said for the iterative process itself. During computation, the script shows the best solution at that point in time. The solutions are in a limbo state where they partially break and comply with the design rules. During the process then, there are serendipitous moments where the script shows solutions that the architect may identify as having potential. However, the script cannot recognize that potential because the solution is outside of the domain set by the rules. As soon as a solution with higher fitness, one that is within those bounds comes along, it replaces the previous one. Due to the huge amount of iterations that are produced each time the script is run, it is easy to miss these moments of serendipity and they get lost in the data.

The research undertaken during this thesis project has shown that computation as a tool for schematic design has significant potential when used as an exploratory method. Its process, final outputs, and rejected solutions all have value and serve to show – either through a straightforward diagram, or through an abstracted parti – the possibilities of the project. Regardless of the chosen approach, computation as an

architectural tool must be a hybrid strategy due to the ill-defined nature of architectural problems and the softness of their criteria; one where architect and computer work together in generating and developing ideas.

Next Steps

If the thesis project were to continue, development of three distinct areas would be prioritized. First, the practical aspects of this process would continue to be refined. As computationally generated layouts have clear applications in function driven architecture, it is worth pursuing this avenue.

The second area that would be studied is how computation can be used as a type of division of labor. Undoubtedly, computation excels when applied to well-defined problems. As such, a set of hyper specific tools for architects to delegate the menial or repetitive parts of design to would be researched. The underlying goal would be that if the architect does not have to do these tasks, they would have additional time to work on the more challenging or important parts of the project.

Lastly, other methods for representing – and thus

defining – layouts, would be studied. This thesis defined spaces as rectangles with clear bounds. However, there are other strategies for defining spaces, such as particles or gradients. Such strategies require entirely new set of design rules and logic and could therefore produce a completely different set of solutions.

GLOSSARY //

Acronyms

CAD

Computer Aided Design

Key Terms[40]

Algorithm

A procedure used to return a solution to a question or to perform a particular task - through a finite list of basic and well-defined instructions.[41]

Boolean

A binary variable that can have one of two possible values, 0 (false) or 1 (true).

Code

(Computing) Program Instructions.

Compute

Reckon or calculate (a figure or amount).

---

40 All definitions taken from Oxford Dictionary unless otherwise stated.

41 Tedeschi, Arturo, Fulvio Wirz, and Stefano Andreani. AAD, Algorithms-aided design: parametric strategies using Grasshopper. Brienza, Italy: Le Penseur Publisher, 2014. p.22

Domain

The set of possible values of the independent variable or variables of a function.

Fitness

1. The quality of being suitable to fulfill a particular role or task.
2. An organism's ability to survive and reproduce in a particular environment.

Generative

Relating to or capable of production or reproduction.

Heuristic

Proceeding to a solution by trial and error or by rules that are only loosely defined.

Parameter

A numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation.

Permutation

Each of several possible ways in which a set or number of things can be ordered or arranged.

Stochastic

Having a random probability distribution or pattern that may be analyzed statistically but may not be predicted precisely.

Syntax

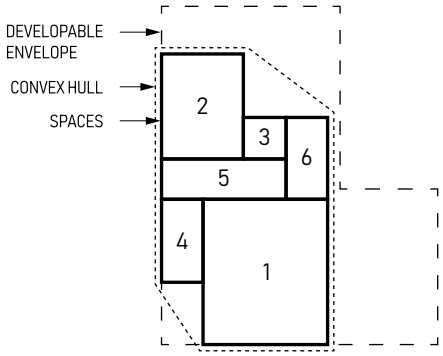
The structure of statements in a computer language.

Topology

1. The study of geometrical properties and spatial relations unaffected by the continuous change of shape or size of figures.
2. The way in which constituent parts are interrelated or arranged.

APPENDIX A //

This appendix shows 40 different layouts produced by the computational model and their respective metadata. The targets for each layout are consistent, the only thing that has been varied in each layout is the weighting scheme. The diagram below explains how to read each figure.



1. WORKSPACE
2. KITCHEN & MEETING AREA
3. CLOSET
4. WASHROOM
5. FOYER
6. EXTERIOR SPACE

SPATIAL OVERLAP

AREA MATCHING

PROPORTION MATCHING

PROGRAMMATIC ADJACENCY

PROGRAMMATIC CONNECTIVITY

THRESHOLD CLEARANCE

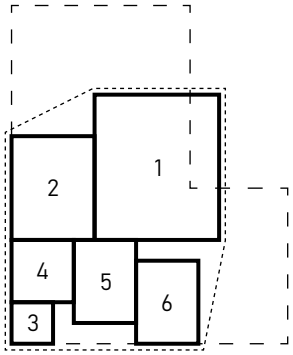
SPATIAL INCLUSION

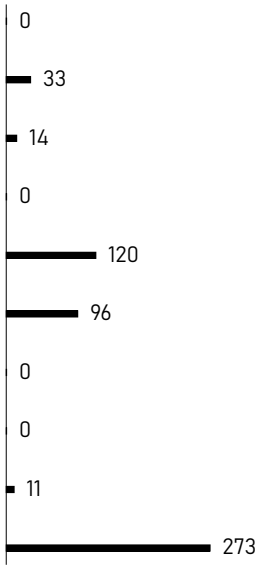
SINGLE ENVELOPE

FOOTPRINT CONSOLIDATION

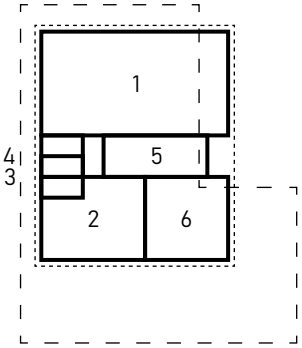
OVERALL FITNESS

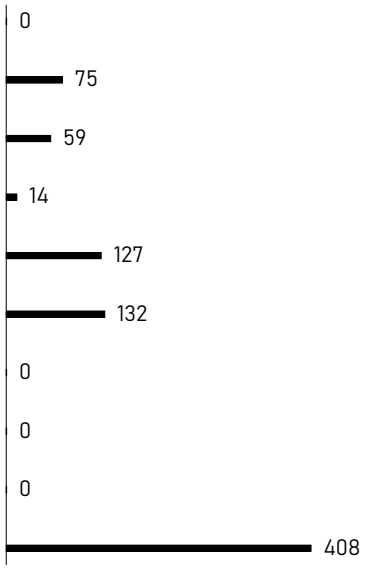
Iteration Number



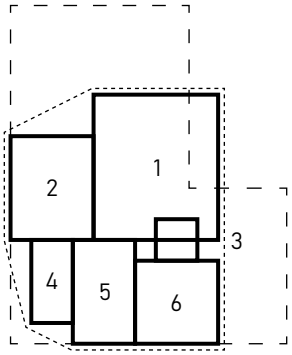


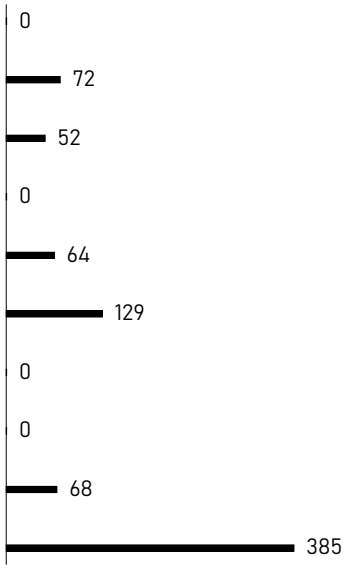
1



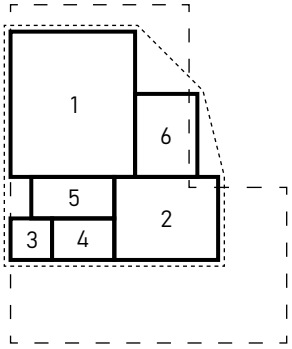


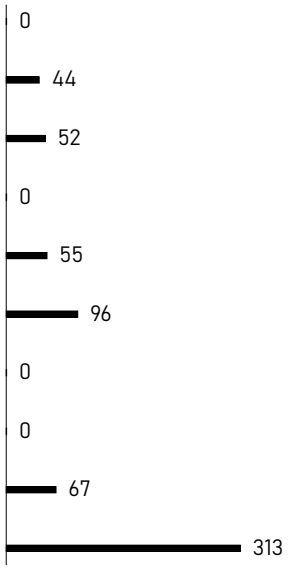
2



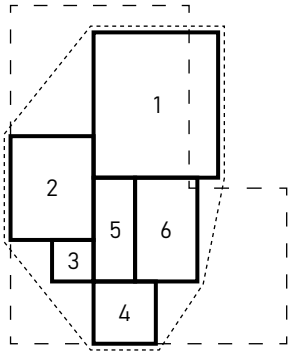


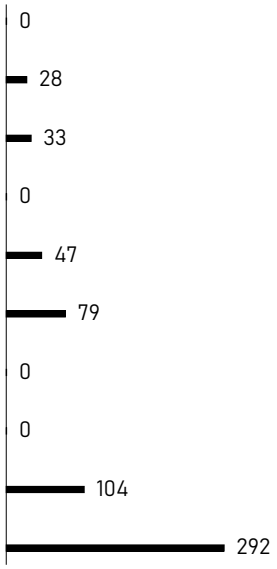
3



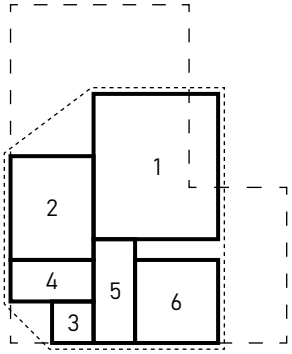


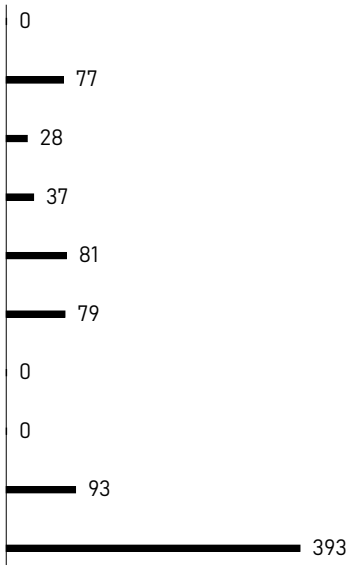
4



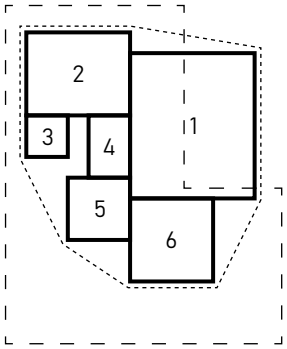


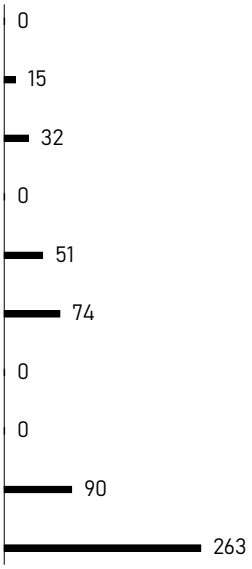
5



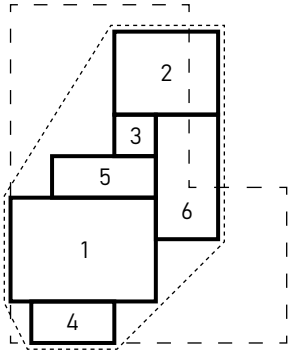


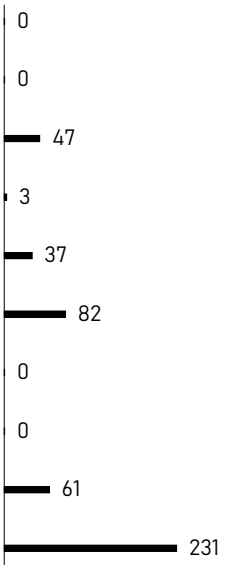
6



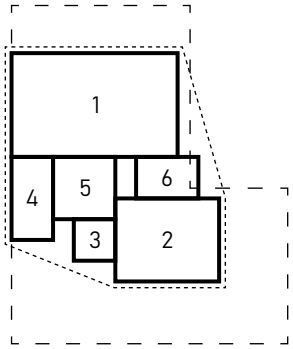


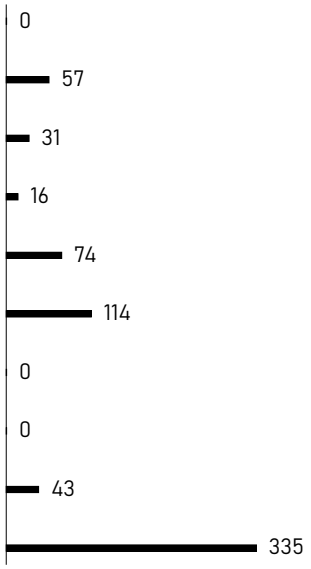
7



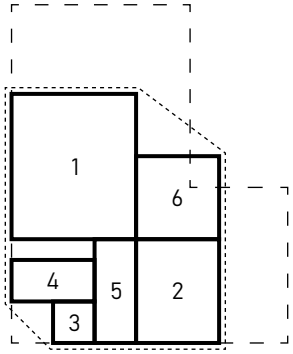


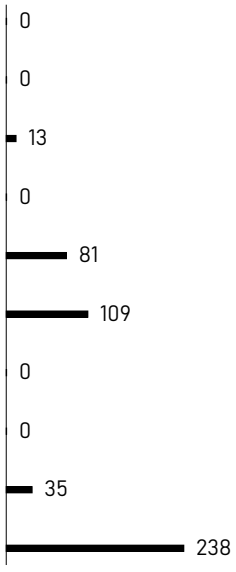
8



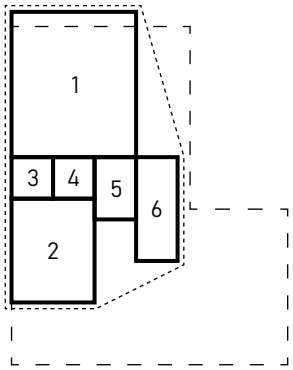


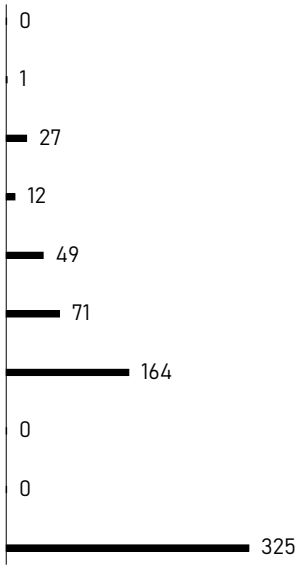
9



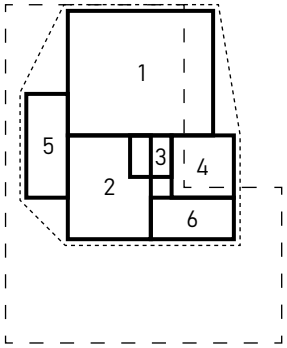


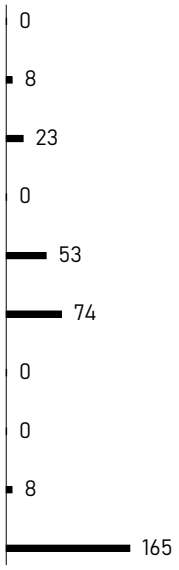
10



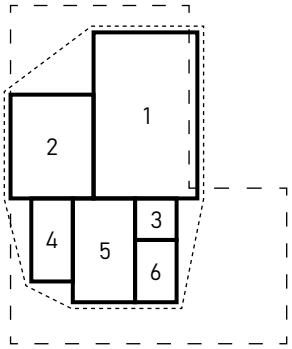


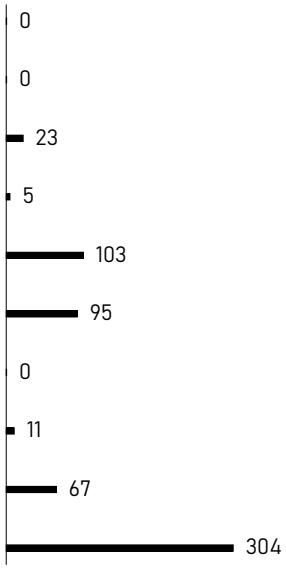
11



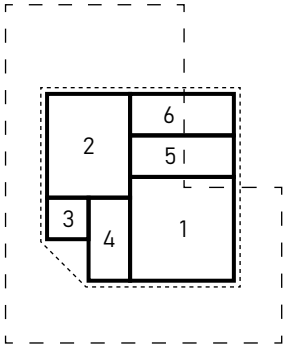


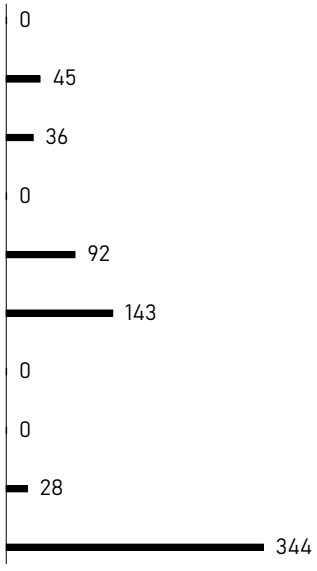
12



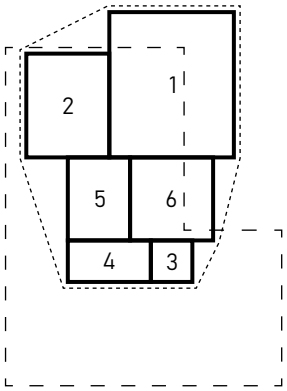


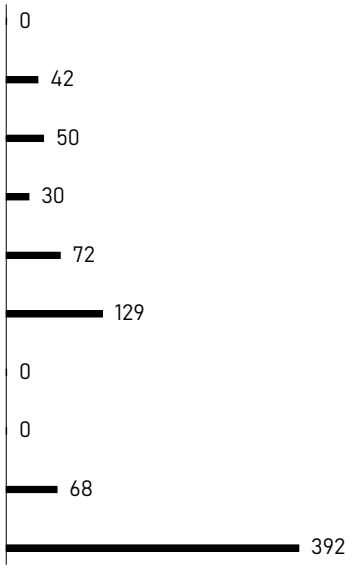
13



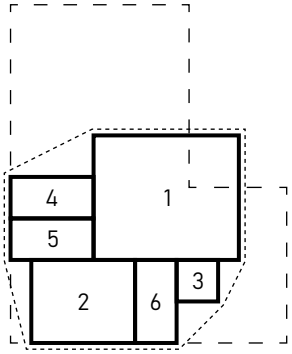


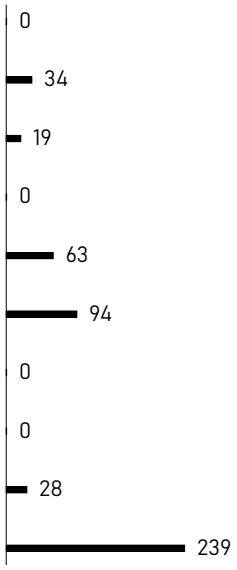
14



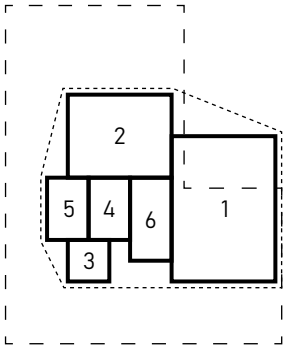


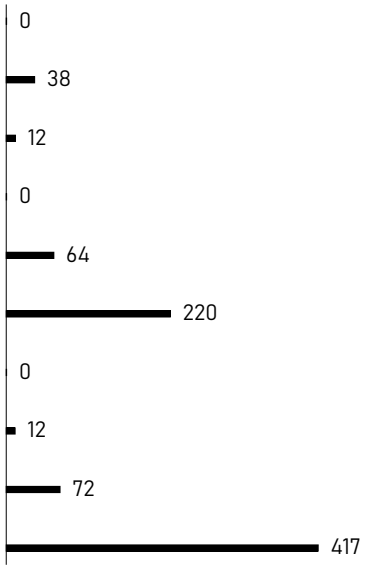
15



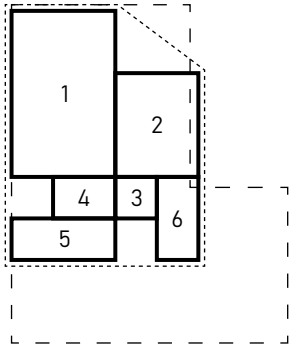


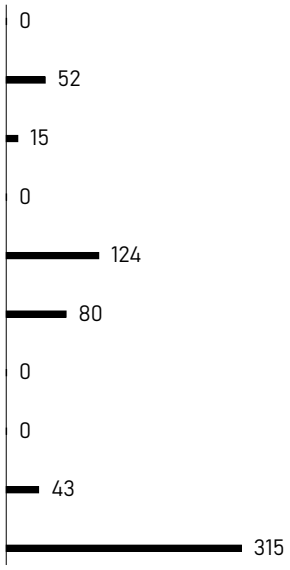
16



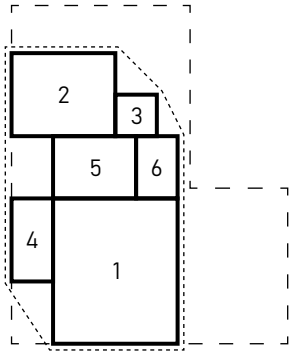


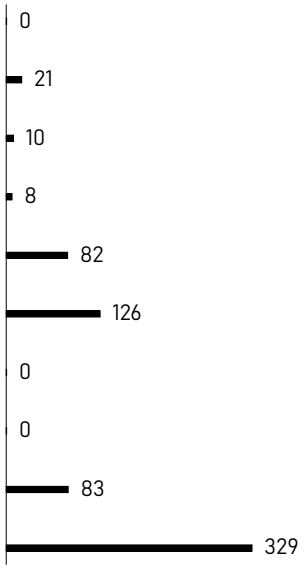
17



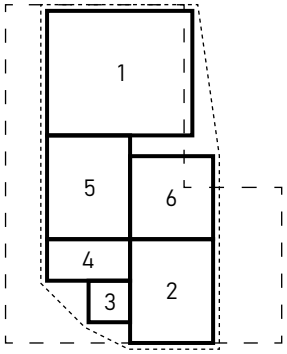


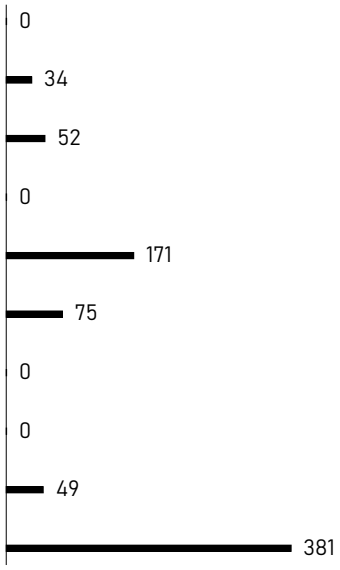
18



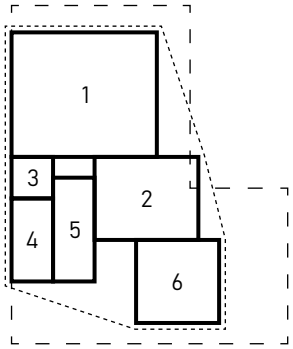


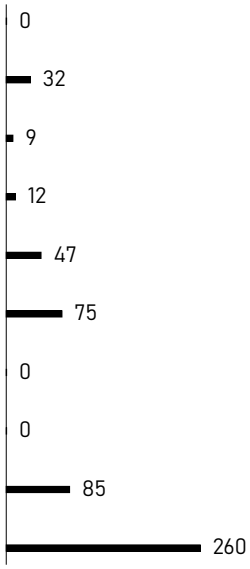
19



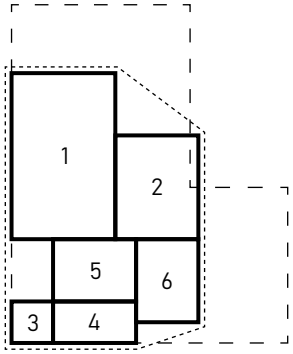


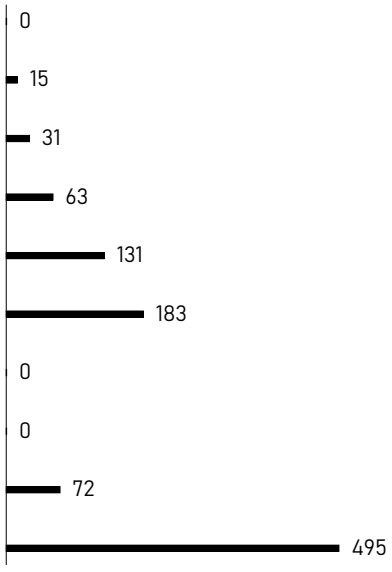
20



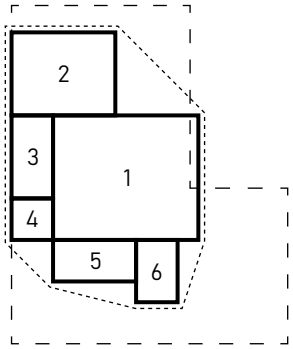


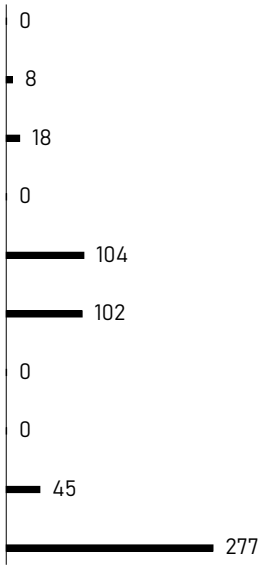
21



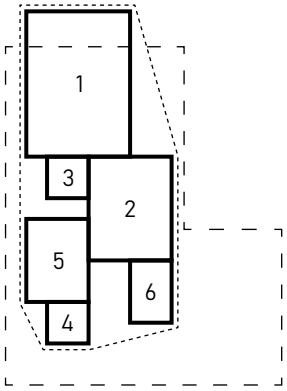


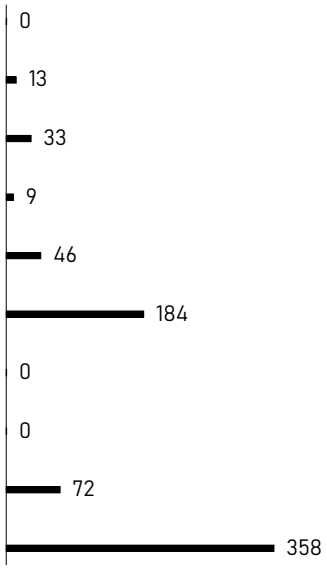
22



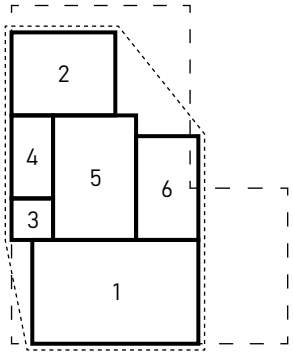


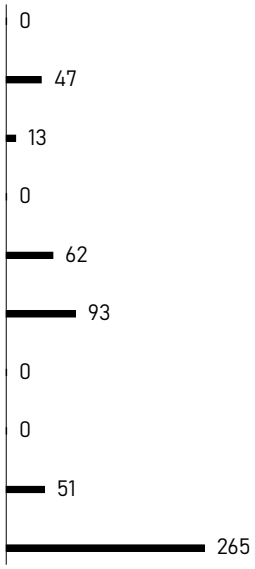
23



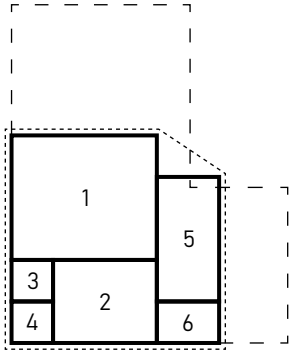


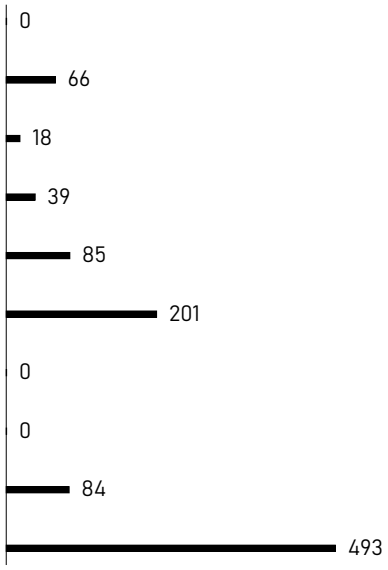
24



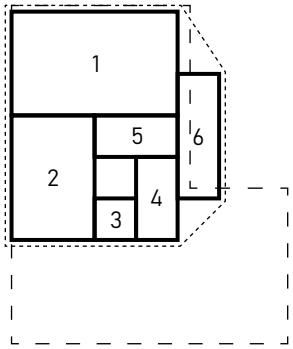


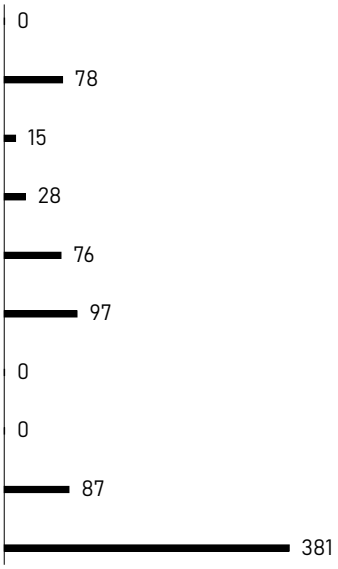
25



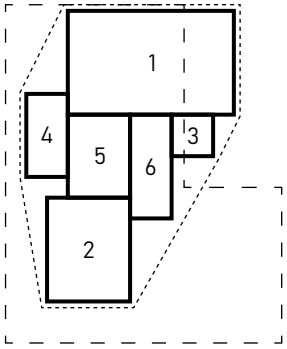


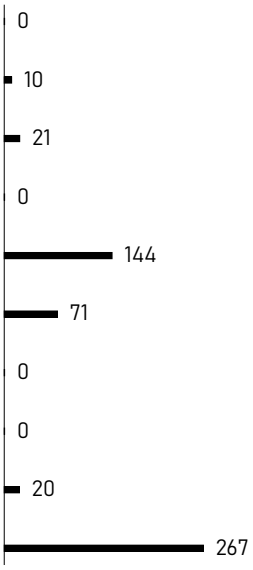
26



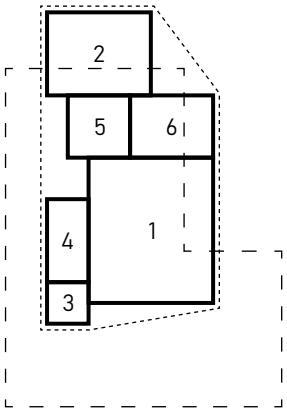


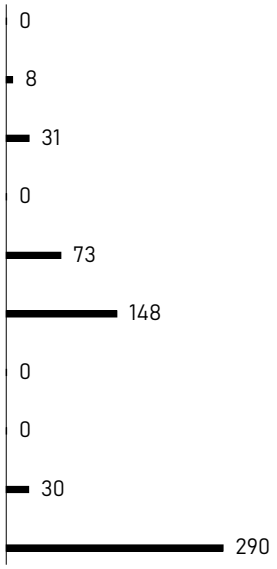
27



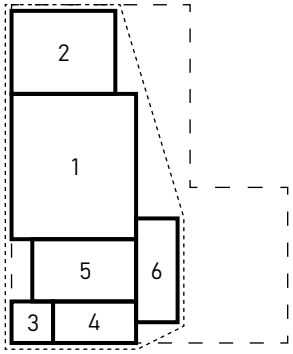


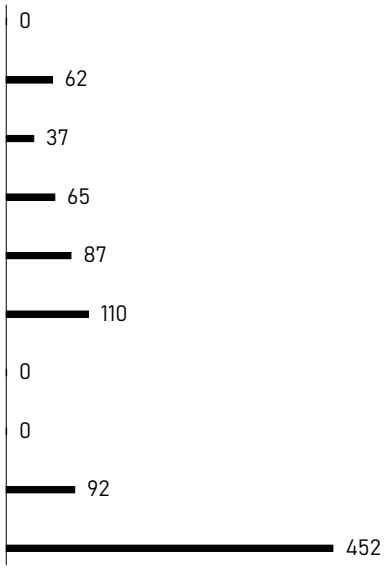
28



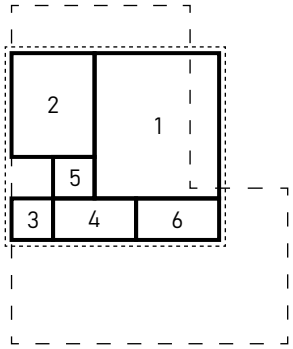


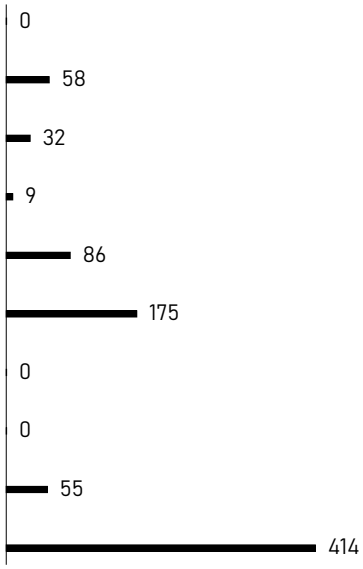
29



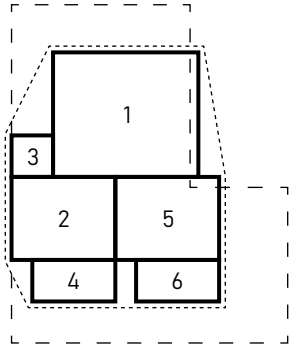


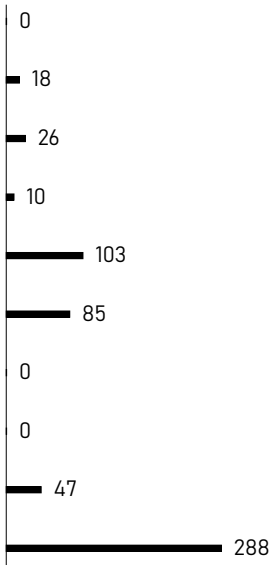
30



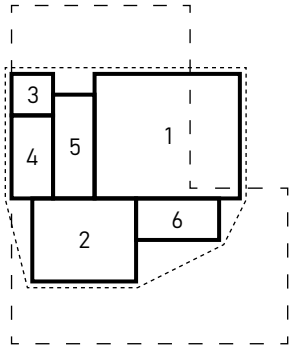


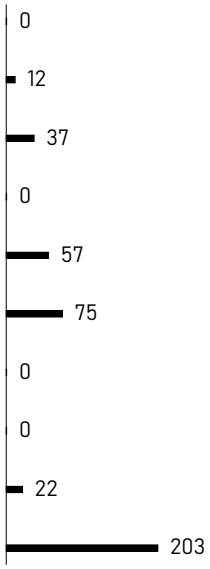
31



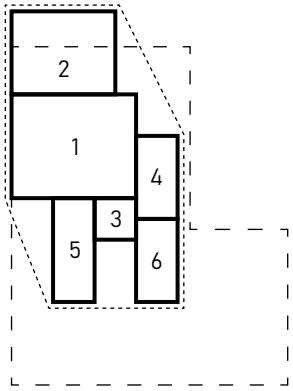


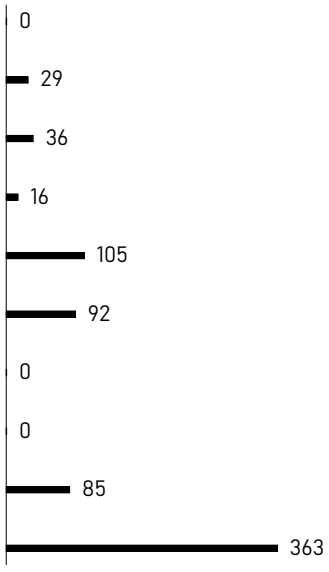
32



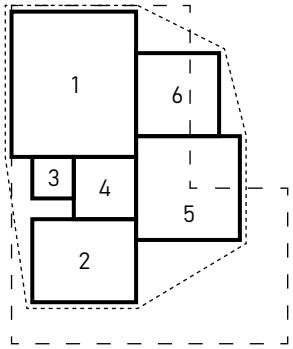


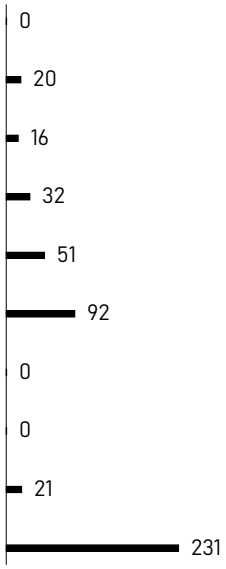
33



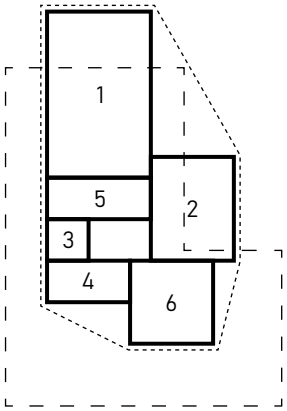


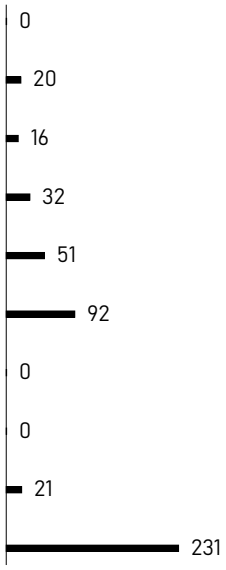
34



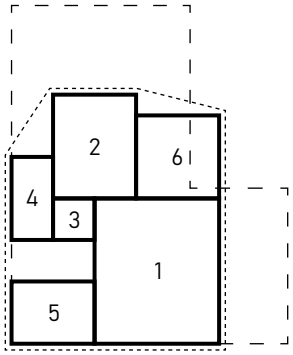


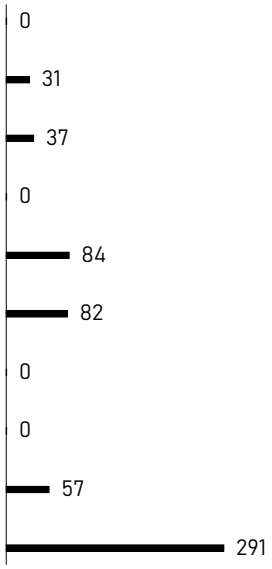
35



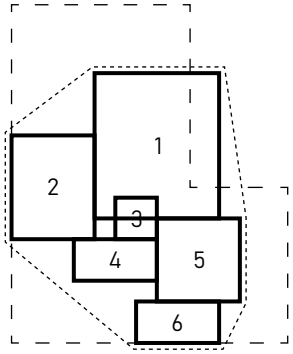


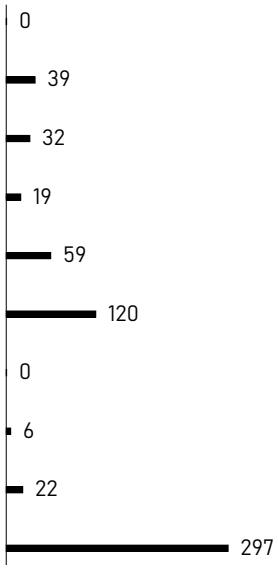
36



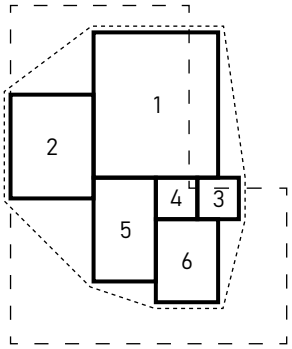


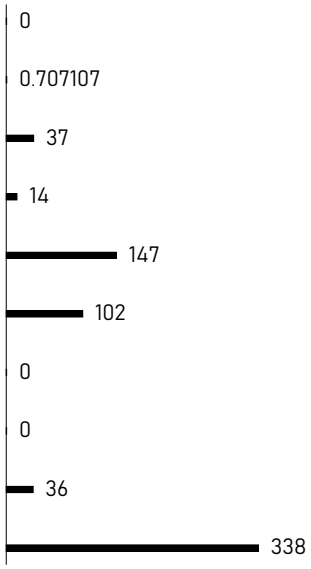
37



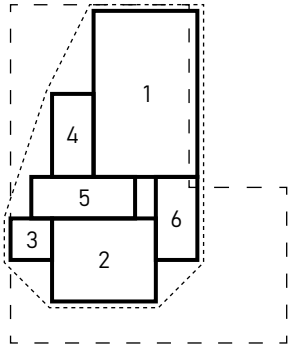


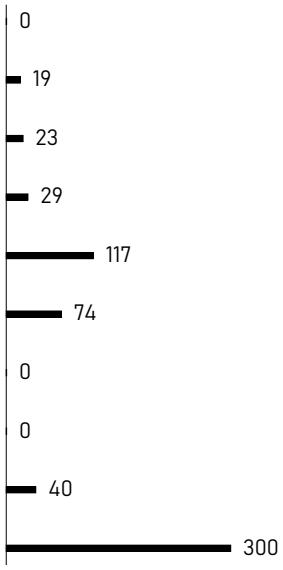
38


1
2
4
3
5
6


0
0.707107
37
14
147
102
0
0
36
338

39


1
4
5
6
3
2


0
19
23
29
117
74
0
0
40
300

40

BIBLIOGRAPHY //

Aasholm, Ron. "Incessant Replication Computational Floor Plan Generation". Master's Thesis, Reprint, Aalto University, 2015. https://repository.tudelft.nl/islandora/object/uuid%3A785ecfa1-93e3-4186-9a48-006f7e004525

Ahlquist, Sean, and Achim Menges. *Computational Design Thinking*. Reprint, Chichester: John Wiley & Sons, 2011.

Alexander, Christopher. *Notes On The Synthesis Of Form*. Reprint, Cambridge, Mass. [u.a.]: Harvard Univ. Press, 1964.

Arvin, Scott A, and Donald H House. "Modeling Architectural Design Objectives In Physically Based Space Planning". *Automation In Construction* 11, no. 2 (2002): 213-225. doi:10.1016/s0926-5805(00)00099-6.

Asanowicz, Aleksander. "Evolution Of Computer Aided Design: Three Generations Of CAD". In *Architectural Computing From Turing To 2000: 17Th Ecaade Conference Proceedings*, 94-100. Reprint, Liverpool: University of Liverpool, 1999.

Bertalanffy, Ludwig von. *General System Theory*. Reprint, New York: Braziller, 1969.

Bitermann, Michael S. "Intelligent Design Objects (IDO) A Cognitive Approach For Performance-Based Design". Ph.D Diss, Reprint, Delft University of Technology, 2009. https://repository.tudelft.nl/islandora/object/uuid%3A785ecfa1-93e3-4186-9a48-006f7e004525.

Broadbent, Geoffrey. *Design In Architecture: Architecture And The Human Sciences*. Reprint, London: Fulton, 1988.

Burry, Mark. *Scripting Cultures*. Reprint, New York, NY: John Wiley & Sons, 2013.

City of Ottawa. "How To Plan Your Coach House In Ottawa". Reprint, Ottawa: City of Ottawa, 2017.

Coates, Paul. *Programming. Architecture*. Reprint, Hoboken: Taylor & Francis, 2010.

Davis, Daniel. "Modelled On Software Engineering: Flexible Parametric Models In The Practice Of Architecture". Ph.D, Reprint, RMIT University, 2013.

Derix, Christian. "Mediating Spatial Phenomena Through

Computational Heuristics". In *ACADIA*, 61-66, 2010. http://papers.cumincad.org/cgi-bin/works/paper/acadia10_61.

Dino, Ipek G. "Creative Design Exploration By Parametric Generative Systems In Architecture". *METU JOURNAL OF THE FACULTY OF ARCHITECTURE* 29, no. 1 (2012): 207-224. doi:10.4305/metu.jfa.2012.1.12.

Heitzinger, Clemens. "Simulation And Inverse Modeling Of Semiconductor Manufacturing Processes". Ph.D, Reprint, Vienna University of Technology, 2002.

Kalay, Yehuda E. *Architecture's New Media*. Reprint, Cambridge, Mass.: MIT Press, 2004.

Koizumi, Kenkichiro. "Technology At A Crossroads: The Fifth Generation Computer Project In Japan". *Historical Studies In The Physical And Biological Sciences* 37, no. 2 (2007): 355-368. doi:10.1525/hsps.2007.37.2.355.

Rodrigues, Eugénio, Adélio Rodrigues Gaspar, and Álvaro Gomes. "An Evolutionary Strategy Enhanced With A Local Search Technique For The Space Allocation Problem In Architecture, Part 2: Validation And Performance Tests". *Computer-Aided Design* 45, no. 5 (2013): 898-910. doi:10.1016/j.cad.2013.01.003.

Rodrigues, Eugénio, Adélio Rodrigues Gaspar, and Álvaro Gomes. "An Evolutionary Strategy Enhanced With A Local Search Technique For The Space Allocation Problem In Architecture, Part 1: Methodology". *Computer-Aided Design* 45, no. 5 (2013): 887-897. doi:10.1016/j.cad.2013.01.001.

Mitchell, William J. "Three Paradigms For Computer-Aided Design". *Automation In Construction* 3, no. 2-3 (1994): 239-245. doi:10.1016/0926-5805(94)90023-x.

Mitchell, William J. *Computer-Aided Architectural Design*. Reprint, New York: Van Nostrand Reinhold, 1979.

Rutten, David. "Galapagos: On The Logic And Limitations Of Generic Solvers". *Architectural Design* 83, no. 2 (2013): 132-135. doi:10.1002/ad.1568.

Rutten, David. "Navigating Multi-Dimensional Landscapes In Foggy Weather As An Analogy For Generic Problem Solving". In *International Conference On Geometry And Graphics*, 2019.

https://ieatbugsforbreakfast.files.wordpress.com/2014/08/manuscript-david-rutten.pdf.

Sariyildiz, Sevil, Özer Ciftcioglu, Bige Tunçer, and Rudi Stouffs. "Knowledge Model For Cultural Analogy In Design And Design Education". In *Local Values In A Networked Design World - Added Value Of Computer Aided Architectural Design*. Reprint, Delft: Delft University Press, 2004. https://cumincad.architexturez.net/doc/oai-cumincadworks-id-avocaad-2003-10.