

Generative Design with Quality Function Deployment for Architectural Layout Design  
Optimization

by

Soojung Yoon

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering  
University of Alberta

© Soojung Yoon, 2024

## **Abstract**

Architectural design problems are inherently complex due to their non-linear nature and the interdependence of many variables. Traditional design methods struggle with the speed and breadth of exploration, often failing to adapt quickly to changing client needs and project timelines. This research addresses these challenges by developing a framework that integrates Quality Function Deployment with generative design to automate and optimize the architectural layout process.

The methodology is segmented into three stages: pre-generative design, generative design, and post-generative design. In the pre-generative design stage, client needs are translated into specific design requirements using Quality Function Deployment. In the generative design stage, generation algorithms are developed, namely bottom-up and top-down method, that are applicable to different problem settings. The generated design solutions are evaluated and optimized through NSGA-II genetic algorithms. The final stage, post-generative design, focuses on refining the chosen designs and converting them into functional Building Information Modelling models.

A case study of designing a single detached house with 3 bedrooms and 2.5 bathrooms is conducted following this methodology. The effectiveness of the integrated Quality Function Deployment and generative design approach is exemplified through the case study. The use of advanced computational tools enables the exploration of numerous design alternatives, efficiently narrowing down to the best solutions that meet predefined criteria.

## **Acknowledgements**

I owe profound gratitude to numerous individuals whose support and guidance were instrumental in the completion of this thesis.

First and foremost, I extend my gratitude to my supervisors, Dr. Mohamed Al-hussein and Dr. Ahmed Bouferguene, for their guidance and insightful feedback throughout my research journey. Their commitment to scholarly excellence has deeply influenced and motivated me.

Additionally, I am thankful to my peers in our research group for their continuous support and motivation. Thanks also go to Jonathan Tomalty and Kendra Hague for their editing of my research.

Above all, my deepest appreciation goes to my family—my parents, my brother, and Doongdoong. Their endless love, sacrifices, and encouragement have been fundamental to my achievements. I am immensely grateful for their presence in my life, providing the motivation and stability needed to pursue my goals.

## Table of Contents

Abstract .....	ii
Acknowledgements .....	iii
List of Tables .....	vii
List of Figures .....	viii
1 Introduction.....	1
1.1 Research background and motivation .....	1
1.2 Research objectives .....	2
1.3 Structure of the thesis.....	4
2 Literature review .....	5
2.1 Architectural layout design .....	5
2.2 Automation of architectural layout design.....	6
2.2.1 Bottom-up methods.....	7
2.2.2 Top-down methods .....	8
2.3 Generative design in architecture.....	10
2.3.1 Advancements in design methodologies.....	10
2.3.2 Multi-objective optimization evolutionary algorithms (MOEAs) .....	15
2.3.3 NSGA-II.....	18
3 Methodology .....	22

3.1	Overview .....	22
3.2	Software – Revit/Dynamo/Generative Design.....	23
3.3	Design requirement analysis .....	24
3.3.1	Quality Function Deployment (QFD) and House of Quality (HoQ) .....	24
3.3.2	Geometric and topological constraints.....	27
3.4	Dynamo script for generation algorithm.....	29
3.4.1	Bottom-up method .....	31
3.4.2	Top-down method.....	33
3.5	Dynamo script for evaluation algorithm .....	35
3.6	Design optimization process .....	37
3.7	Conversion from geometries to BIM elements .....	39
4	Case study .....	42
4.1	Pre-GD: Constraints and objectives.....	42
4.1.1	Building House of Quality .....	42
4.1.2	Geometric and topological requirements .....	49
4.2	GD: Bottom-up method .....	50
4.2.1	Dynamo script.....	50
4.2.2	Details on fitness calculation .....	52
4.2.3	Generative design results .....	54
4.3	GD: Top-down method .....	58

4.3.1	Dynamo script.....	58
4.3.2	Details on fitness calculation .....	59
4.3.3	Generative design results .....	62
4.4	Post-GD: Conversion of geometries into BIM elements .....	62
4.5	Discussion .....	65
4.5.1	Sensitivity analysis: variant parameter setting of genetic algorithm .....	65
4.5.2	Sensitivity analysis: reduced connectivity constraints.....	71
5	Conclusion .....	74
5.1	Summary .....	74
5.2	Contributions.....	76
5.3	Limitations and future work.....	76
	References.....	78

## List of Tables

Table 3-1. Top-down method & bottom-down method.....	29
Table 3-2. Dynamo nodes for generating geometries (Autodesk, n.d.).....	30
Table 3-3. Constraints, variables, and objectives of the bottom-up method.....	31
Table 3-4. Constraints, variables, and objectives of the top-down method.....	34
Table 3-5. Dynamo nodes for converting geometries into BIM elements (Autodesk, n.d.).....	40
Table 4-1. Customer's requirements and their relative importances.....	43
Table 4-2. Technical characteristics and their target and design phase .....	44
Table 4-3. Relationship matrix between customer's requirements and technical characteristics .	46
Table 4-4. Importance weights and ranks of each technical characteristic.....	47
Table 4-5. Scores on each objective of each design outcome.....	56
Table 4-6. Normalized scores for each objective of solution No. 1 and No. 12.....	57
Table 4-7. Scores for each objective across all design outcomes .....	72
Table 4-8. Best and worst scored design outcomes .....	72

## List of Figures

Figure 1-1. Process of Generative Design (Adapted from Nagy & Villaggi, 2018) .....	3
Figure 2-1. The Macleamy Curve (Adapted from Ilozor & Kelly, 2012) .....	5
Figure 2-2. Schematic of bottom-up layout design methods (Adapted from Weber et al., 2022)..	7
Figure 2-3. Schematic of top-down layout design methods. (Adapted from Weber et al., 2022)..	8
Figure 2-4. Evolution of Design Methodologies .....	10
Figure 2-5. Comparison diagram between parametric and conventional design process (Adapted from Eltaweel & SU, 2017) .....	11
Figure 2-6. Heydar Aliyev Center by Zaha Hadid Architects .....	12
Figure 2-7. Generative Design Process (Adapted from Krish, 2011).....	14
Figure 2-8. General scheme of an evolutionary algorithm (Adapted from Eiben & Smith, 2015) .....	16
Figure 2-9. NSGA-II procedure (Adapted from Deb et al., 2002) .....	18
Figure 2-10. Pareto fronts .....	19
Figure 2-11. Crowding distance of a solution.....	20
Figure 3-1. Overview of the methodology.....	22
Figure 3-2. House of Quality (Adapted from Delgado-Hernandez et al., 2007) .....	25
Figure 3-3. An example of floor plan and its graph representation .....	28
Figure 3-4. Flowchart of the bottom-up method.....	31
Figure 3-5. Sequence of the bottom-up method.....	32
Figure 3-6. Flowchart of the top-down method .....	34
Figure 3-7. Sequence of the top-down method.....	35
Figure 3-8. Dynamo script for calculating the fitness.....	36



Figure 3-9. Numeric objective visualized as a geometry.....	36
Figure 3-10. Input/output setting in Dynamo .....	37
Figure 3-11. Sample Generative Design user interface .....	38
Figure 3-12. Visual outputs and numerical scores of Generative Design.....	39
Figure 3-13. A sample Dynamo script for converting curves into walls .....	41
Figure 3-14. Converted Walls and corresponding properties .....	41
Figure 4-1. Correlation matrix (roof of the HoQ).....	45
Figure 4-2. House of Quality for the case study .....	48
Figure 4-3. Dynamo Script for Room Dimension Configurations: (a) fixed dimension, .....	49
Figure 4-4. Justified Plan Graph (JPG) and the room data of the case study .....	50
Figure 4-5. Overview of the Dynamo script for the bottom-up method 1) Space generation section, 2) Space aggregation section, 3) Fitness calculation section.....	50
Figure 4-6. Fitness functions for different objectives.....	52
Figure 4-7. Generative design results .....	55
Figure 4-8. Top 2 solutions selected for comparison using normalization.....	57
Figure 4-9. Overview of the Dynamo script for the top-down method .....	58
Figure 4-10. High-scoring design results from the top-down method.....	62
Figure 4-11. Conversion of selected design into BIM model.....	63
Figure 4-12. Floor plan and 3D view after modification.....	64
Figure 4-13. Floor plan and 3D view after conversion and modification.....	65
Figure 4-14. Performance scores for each population size with generation number 10.....	68
Figure 4-15. Performance scores for each population size with generation number 20.....	68
Figure 4-16. Performance scores by generation numbers.....	68

Figure 4-17. High-scoring designs and their floor plan and 3D view .....	70
Figure 4-18. Justified Plan Graph (JPG) with reduced connectivity constraints.....	71
Figure 4-19. Best and worst scored design outcomes.....	73

# **1 Introduction**

## **1.1 Research background and motivation**

Architectural design problems are “wicked” problems. As defined by Rittel & Webber (1973), wicked problems have no stopping rules and are not binary — right or wrong — but are instead judged on a spectrum from good to bad, unlike mathematical problems or puzzles that have findable solutions. Every wicked problem is unique, each with its own set of specific circumstances and challenges. Additionally, design problems are complex with numerous interdependent factors with multiple possible solutions.

The traditional way of designing has limitations in terms of the speed and breadth of exploration due to its time-consuming nature. Therefore, the process has restricted iteration and flexibility, which are crucial for quickly adapting to client needs and project schedule. Given the inherent complexities and uncertainties in architectural design, such as competing interests, economic factors, and site-specific challenges, leveraging advanced computational tools such as generative design becomes crucial. These technologies enable architects to explore a vast array of design options more efficiently and creatively.

The evolution of architectural tools has transitioned from 2D Computer-Aided Design (CAD) to Building Information Modelling (BIM), and further to parametric design, and now, to the innovative generative design, each stage building upon the last to enhance design capabilities and complexity. Generative design represents a paradigm shift in architectural planning, harnessing algorithms to generate design alternatives based on predefined criteria. Its application has been explored in various studies, demonstrating its ability to produce optimized spatial configurations, sustainable designs, and cost-effective solutions, and so much more.

To ensure that the generative design engine is supplied with the correct constraints and objectives, it is crucial to define these major inputs accurately. Quality Function Deployment (QFD) plays a key role in this process by systematically translating client needs into detailed design specifications using a tool known as House of Quality (HoQ). HoQ is a comprehensive matrix that captures and visualizes the relationship between client needs and the technical means of achieving them, facilitating prioritization and decision-making. By integrating QFD within the architectural design process, the effectiveness of generative design techniques is significantly enhanced for that as mentioned earlier, setting the right input value is crucial in the process. Additionally, QFD improves communication of client expectations to all stakeholders involved, thereby enhancing collaboration, and increasing the efficiency of the design process.

This research aims to develop a holistic framework that utilizes QFD and generative design as the dual pillars of an end-to-end automated layout design process. The goal of the framework is to seamlessly integrate these methodologies into architectural workflows, enhancing their applicability across various design challenges. The algorithm developed in this study provides a flexible framework that can be adapted to various layout design problems rather than being limited to specific problem scenarios.

## **1.2 Research objectives**

This research aims to establish an end-to-end process for automated layout design using Quality Function Deployment (QFD) and generative design. To realize this goal, the approach is divided into three core stages, which are displayed in Figure 1-1.

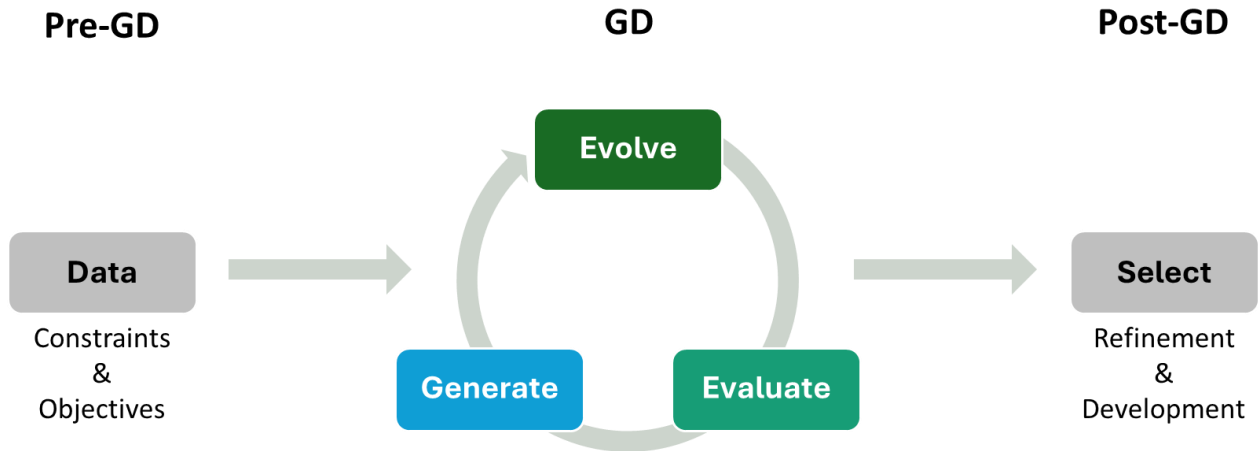


Figure 1-1. Process of Generative Design (Adapted from Nagy & Villaggi, 2018)

### 1) Pre-generative design

The initial stage involves a detailed analysis of design requirements to ensure alignment with client needs. This process utilizes a QFD approach to translate complex client requirements into definitive, actionable design objectives. This stage serves as the foundation for the generative process, ensuring that all generated designs are rooted in a deep understanding of client expectations and practical constraints. For instance, the identified requirements, such as the access between spaces and specific dimensions of spaces, are quantified and become the fitness functions that guide the generative design process to find a set of design solutions that best meet the requirements.

### 2) Generative design

The generative design stage of this research focuses on the development and application of sophisticated algorithms using Dynamo, a visual programming tool for Revit. It involves scripting both generation and evaluation algorithms for the top-down method and bottom-up method each, which will be further discussed in detail in Section 2.2. The algorithms will be tailored for architectural layout planning according to the constraints and objectives. These

algorithms are fed to generative design process with genetic algorithms, specifically NSGA-II, for optimizing the design.

### 3) Post-generative design

The final stage refines the generated layouts to enhance their practical and aesthetic values. Then, they are converted into Building Information Modelling (BIM) elements. By doing so, the designs are not only optimized for performance but are also ready for immediate implementation in construction workflows, bridging the gap between conceptual development and practical application.

The integration of these stages exceeds traditional layout planning by providing a systematic, adaptable, and efficient approach to architectural design, ensuring that solutions are both innovative and aligned with specific client and project requirements.

## **1.3 Structure of the thesis**

This thesis is composed of five chapters. Chapter 1 (Introduction) introduces the research background and motivation. Chapter 2 (Literature Review) presents a thorough literature review on architectural layout design, automated architectural layout design, and generative design in architecture. Chapter 3 (Methodology) details the methods employed in this research to address each stage of the generative design process: pre-generative design, generative design, and post-generative design. Chapter 4 (Case Study) describes the application of these methods in designing a residential layout. Finally, Chapter 5 (Conclusion) summarizes the research findings and contributions, and discuss limitations and suggestions for future work.

## 2 Literature review

### 2.1 Architectural layout design

MacLeamy's time-effort distribution curve, displayed in Figure 2-1, shows that the effort invested in the early stages of design has a greater impact on the overall cost and functional capabilities of a project than the same effort expended in later stages. This implies that making changes and adjustments early in the design process (e.g., during pre-design and schematic design phases) is less costly and more effective than making those changes during construction.

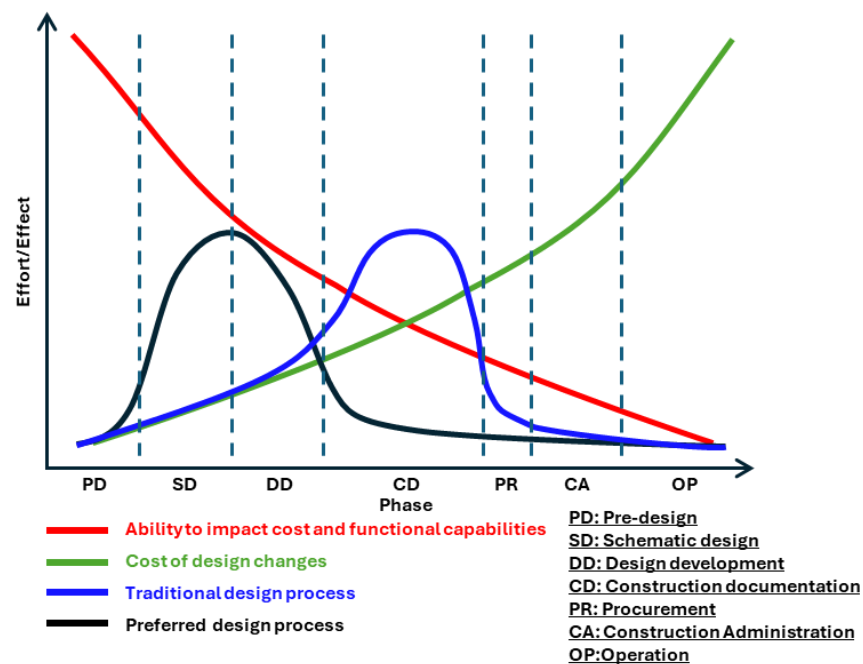


Figure 2-1. The Macleamy Curve (Adapted from Ilozor & Kelly, 2012)

During these initial phases, designers and clients should collaborate closely to identify needs, expectations, and any limitations of the project. By thoroughly addressing client requirements early in the process, the project is more likely to stay within budget, meet functional expectations, and minimize costly changes during construction. However, determining the requirements and constraints itself is complex, and finding their relationships and trade-offs is also challenging.

House of Quality is one of the tools that systematically organize the relationships between requirements and constraints, which will be introduced in Section 3.3.1.

Despite the importance of early design, it is difficult to find the best design not only due to the nature of the design problem being “wicked” problems, but also limited resources such as time, cost, and labour. Moreover, conventional design method may not fully incorporate all relevant factors or adjust to changing requirements, consequently, resulting in a design that does not optimally satisfy the established design criteria (Ko et al., 2023). As artificial intelligence (AI) and machine learning (ML) continue to advance, various research have been conducted to overcome this limitation in architectural layout planning with iteratively generating many different alternatives to assist decision making (Ko et al., 2023).

## **2.2 Automation of architectural layout design**

Weber et al. (2022) categorized computational approaches for automatic space layout generation into three, namely bottom-up methods, top-down methods, and referential methods. Bottom-up methods focus on generating layout designs by aggregating predefined modules or building blocks according to specific spatial, environmental, or structural criteria, which allows for the exploration of various designs. In contrast, top-down methods begin with a given building envelope or massing, and then subdivide and fit spaces within these constraints using geometric and functional considerations. Referential methods, draw upon existing architectural layouts and design precedents, frequently utilizing machine learning algorithms like generative adversarial networks (GANs) to create new designs that mimic learned patterns and styles. In this paper, the bottom-up and top-down methods will be discussed in more detail.



### 2.2.1 Bottom-up methods

Architectural design problems often have specified constraints on the dimension of spaces or adjacency between them. These constraints can be either static or changing throughout the generating process to satisfy the pre-defined fitness of the design output. Weber et al. (2022) acknowledged the adaptability of the bottom-up methods for that they align with traditional design strategies such as mind mapping and bubble diagrams, but also argue that the bottom-up methods may not reach a solution due to their vast exploration area necessitating sophisticated heuristics to efficiently navigate and produce desirable outcomes. There have been different approaches developed by researchers using bottom-up methods for automating the generation of layouts.

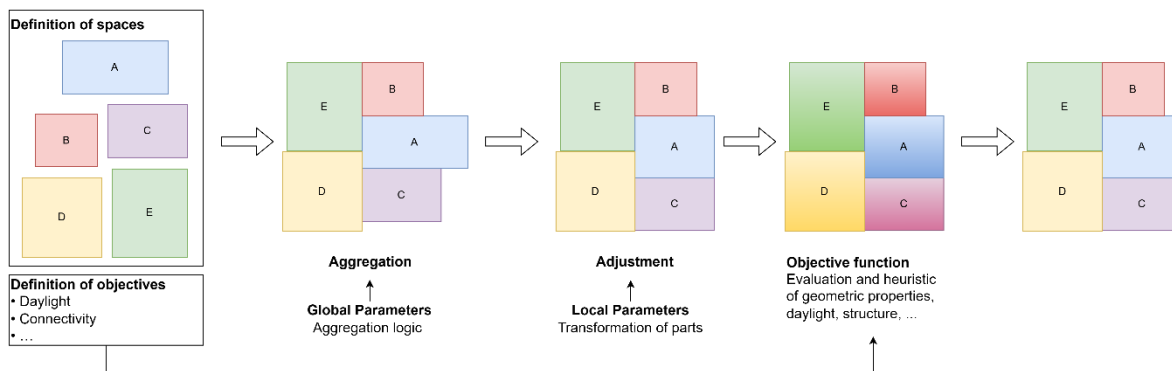


Figure 2-2. Schematic of bottom-up layout design methods (Adapted from Weber et al., 2022)

Merrell et al. (2010) developed a method inspired by traditional architectural design processes. Building layouts were converted from an architectural program generated using Bayesian Network, which contains a list of rooms, their adjacencies, and desired sizes, and optimized with the procedure to minimize a cost function that evaluates the quality of the layout. Chatzikonstantinou (2014) developed a method for creating 3-dimensional architectural layout, which is an extended version of the existing Voronoi subdivision method, having area and weighted adjacency matrix as inputs. Hua (2016) integrated graphical inputs for automated layout generation when the

connectivity and size constraints are considered. Their research's unique approach lies in the ability to generate layouts from patterns within images, accommodating complex shapes and non-orthogonal geometries and allowing for a more intuitive design process. Guo & Li (2017) combined an agent-based system for defining room topologies with evolutionary optimization to refine 3D architectural layouts. Their approach addresses the complexities of multi-storey designs and varying room heights, moving beyond the limitations of traditional 2D planning methods. By ensuring correct topology from the start, the method efficiently narrows the optimization search space. Bahrehmand et al. (2017) introduced a genetic algorithm-based interactive tool for layout planning in computational design projects like games and virtual reality. This tool customizes space arrangements by integrating architectural principles and user preferences, initiated by the designer's primary requirements. Their approach generates personalized layouts with simplified planning process that resembles human-crafted designs.

## 2.2.2 Top-down methods

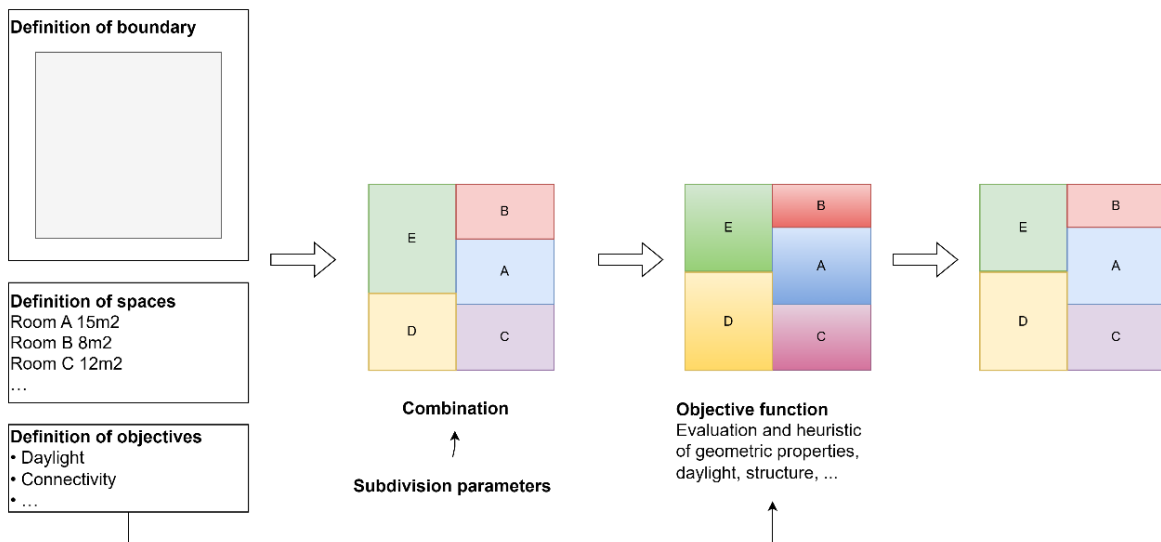


Figure 2-3. Schematic of top-down layout design methods. (Adapted from Weber et al., 2022)

Real-world architectural designs often face constraints on the envelope such as building mass and site boundaries. Top-down methods are suitable for this type of space planning design problems. Unlike the bottom-up method, the iterative optimizations are done under the boundary conditions, which means restricted search space, resulting in more reduced computational challenge than the bottom-up method (Weber et al., 2022). Many researchers developed these methods using subdividing, fitting, and shape packing.

Marson & Musse (2010) used squarified treemaps algorithm, which is a method that organizes hierarchical data by subdividing space into rectangles with aspect ratios as close to 1 as possible to create a house floor plans with semantic details. Koenig & Knecht, (2014) compared two evolutionary algorithm-based methods, dense packing and subdivision algorithms based on generation speed, reliability in finding optimal solutions, and the diversity of solutions produced. Both methods were generally practical, while subdivision had better performance and dense packing was suited for user interaction (Koenig & Knecht, 2014). Nagy, Villaggi, et al. (2017) introduced a new model for tackling space planning challenges in architecture exemplified through designing an exhibit hall, focusing on optimal layout arrangements using genetic algorithm. The model provides guidelines and evaluation techniques to assess the model's suitability for metaheuristic optimization. Saha et al. (2020) introduced a reinforcement learning-based solution to space allocation problems in architecture and urban design. The solution aids in generating building layouts, site parcellation, and massing, allowing for easy analysis.

## 2.3 Generative design in architecture

### 2.3.1 Advancements in design methodologies

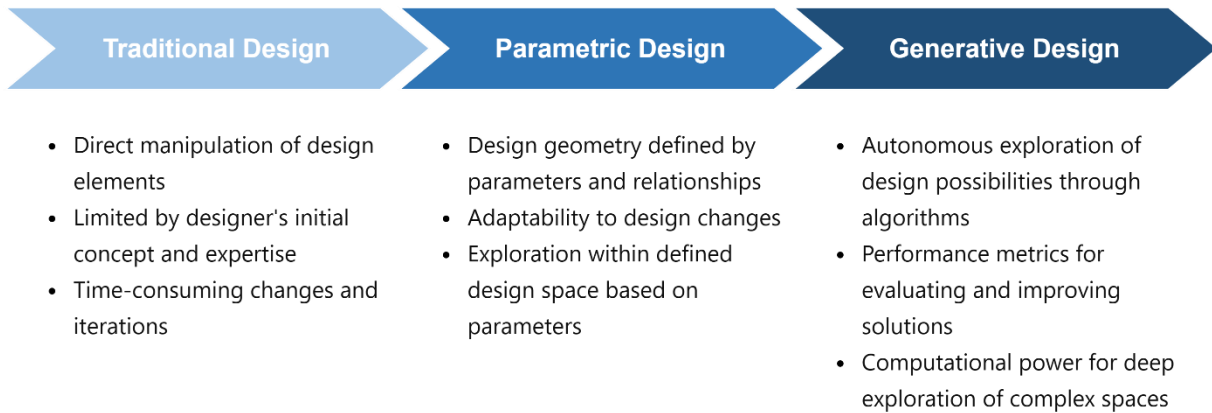


Figure 2-4. Evolution of Design Methodologies

#### Parametric design

As technology evolves, so does the field of design, incorporating new tools and methodologies that reshape how designers conceptualize and explore solutions. The evolution from singular design modelling systems to supportive environments and to advanced generative parametric tools marks a significant shift in design methodologies. Initially, Computer-Aided Design (CAD) was used mainly for 2D and 3D modelling, lacking support for iterative design changes. With the advent of parametric design systems, designers gained the ability to dynamically alter and refine designs, overcoming previous limitations and marking a shift towards more flexible and interactive design processes. (Oxman, 2017).

Parametric design, often referred to as constraint modelling, involves setting up relationships or "constraints" between design elements so that changes to one element automatically update others (Woodbury, 2010). This modelling method goes beyond basic computer-aided drafting or modelling, by allowing complex modifications while maintaining relationships between design

elements. For example, in architectural design, changes to the parameters of a wall can automatically adjust related windows, roofs and other structural elements without manual modification. Eltaweel & SU (2017) described the ability for parametric design to automatically and immediately update the model as a “short cut” to the final model, while conventional design process requiring repetitive manual modification (Figure 2-5).

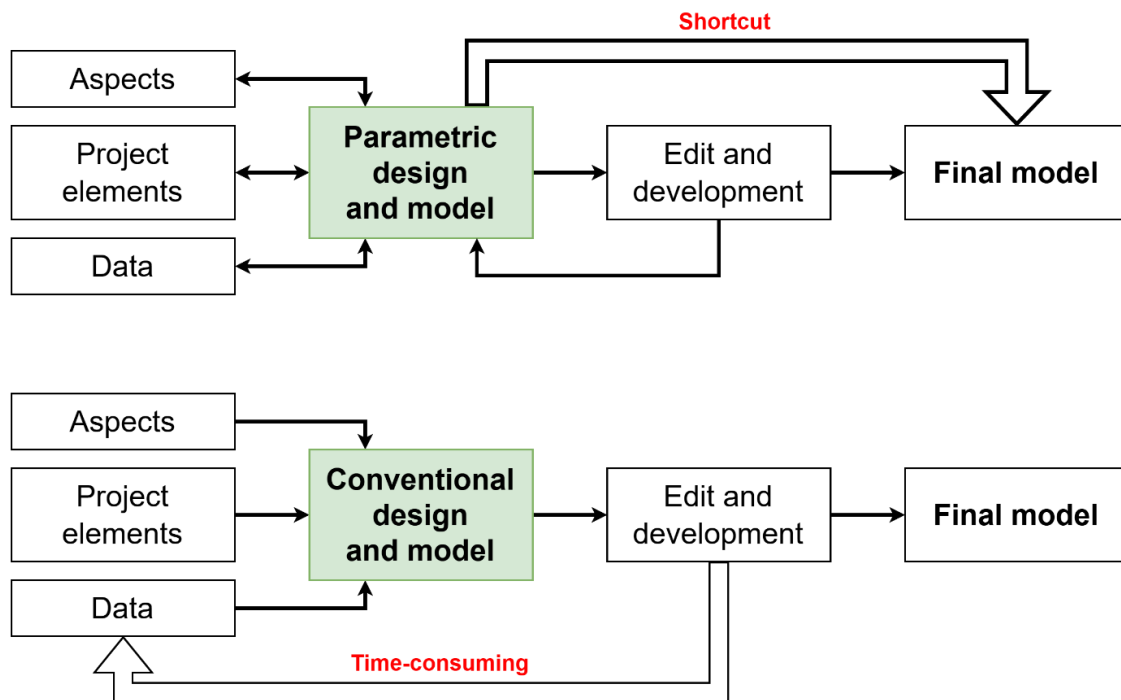


Figure 2-5. Comparison diagram between parametric and conventional design process (Adapted from Eltaweel & SU, 2017)

Building upon this foundation, parametric design allows for the efficient management and variation within architectural models by utilizing the concept of families of elements, where groups of elements differ only in the dimensions of their parts (Monedero, 2000). Furthermore, this technique enables designers to explore a wide range of solutions within a structured design space, moving beyond the traditional approach of crafting a few solutions based on expertise. By

embedding project constraints and objectives into the model, parametric design facilitates the automatic generation of diverse solutions. This shifts the designer's role towards shaping a multi-dimensional design landscape driven by key model parameters, thus vastly expanding design possibilities (Nagy et al., 2017). Consequently, scripting has become an essential skill for designers, whose job is to create a comprehensive framework that encompasses an entire population of possible designs (Reas & Fry, 2007). Figure 2-6 shows one of the most famous parametric designs in architecture, Heydar Aliyev Center in Azerbaijan designed by Iraqi-British architect Zaha Hadid, where curvature of the building's surface was designed using parametric design process (Hufton+Crow, n.d.; Iwan Baan, n.d.).



Figure 2-6. Heydar Aliyev Center by Zaha Hadid Architects (images from Hufton+Crow, n.d.)

## **Generative design**

Generative design extends the capabilities of parametric design which is limited by the need for manual adjustments of parameters to explore different design outcomes. Generative design is defined as a design process utilizing algorithms and computational power to autonomously explore a large range of design solutions (McKnight, 2017). Through automatically navigating within the search space with various combinations of parameter values, the process reduces the reliance on

manual adjustments by designers which were needed in traditional and parametric design process. Parameters defining the design's attributes are manipulated algorithmically, which allows generative design to dynamically generate and optimize geometries based on design goals. The design process involves four steps as follows (McKnight, 2017):

**Step 1: Establishing design parameters and goals**

Experienced designer/engineer set up a parametric model that serves as the foundational framework for the generative design process. Specific metrics, also called fitness functions, are also defined to objectively evaluate each design outcome, as computers do not have innate design intuition to discern good from bad designs (Nagy, Lau, et al., 2017).

**Step 2: Algorithmic generation and performance analysis to create numerous design options**

The model is linked to a search algorithm, such as genetic algorithms, and the first generation of design outcomes is identified.

**Step 3: Iterative refinement of parameters and goals based on results**

Designer/engineer study the first generation of results and modify parameters and goals to refine the algorithm and get the most high-scoring design outcomes.

**Step 4: Manufacturing the final design**

The chosen design is adapted for production

As an example of the process, consider a scenario in architectural design where a facade needs to be covered with tiles. An experienced engineer sets up a parametric model incorporating a tiling algorithm that includes parameters like tile size, shape, and the layout area. Constraints such as alignment rules and material properties are defined to guide the tiling layout. Metrics for evaluating the designs are established in terms of aesthetics, functionality, and cost efficiency. These goals help objectively assess each design's viability since computers cannot intuitively determine good

or bad designs. The tiling algorithm uses these parameters to generate various tiling patterns, automatically adjusting the designs based on the set metrics to optimize aesthetics, function, and cost. Similarly, any generation algorithm, such as those for site layout, daylight optimization or space layout generation as developed in this research, can also serve as a parametric model within the generative design process.

Generative design is particularly beneficial during the conceptual stages of a project, where the design is still being formulated. By enabling the exploration of various design options early on, it can lead to significantly better outcomes compared to making limited optimizations at the later stages of the design process (Krish, 2011). To effectively leverage generative design process, it must integrate precise metrics for assessing each design option, with clear guidelines provided by the designer. This allows the computer to accurately evaluate the efficacy of designs (Nagy, Lau, et al., 2017). Moreover, it's essential to couple the parametric model with a search algorithm, like multi-objective evolutionary algorithms, that manipulates input parameters based on metric feedback to optimize design outcomes and thoroughly explore the design space, progressively refining designs across generations (Murata & Ishibuchi, 1995).

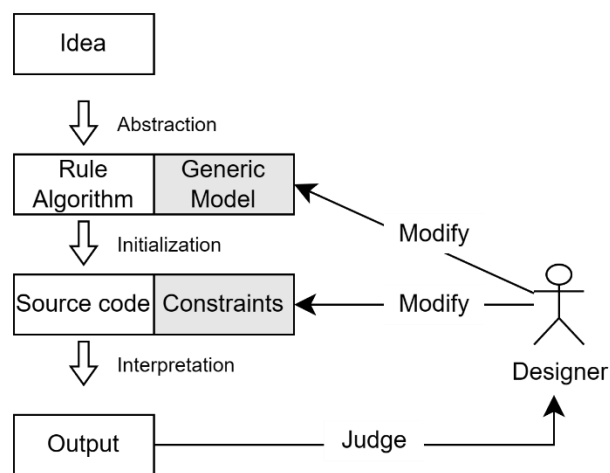


Figure 2-7. Generative Design Process (Adapted from Krish, 2011)



Krish (2011) highlights the interactive nature of the generative design process as the designer constantly guides and refines the process while the computer algorithm generates a range of potential solutions within the predefined design space (Figure 2-7). In other words, the process relies on a partnership between human creativity and computational power, with the designer ensuring that the solutions generated align with both the practical requirements and the creative vision of the project.

### 2.3.2 Multi-objective optimization evolutionary algorithms (MOEAs)

Multi-objective optimization (MOO) is a process of simultaneously optimizing two or more conflicting objectives. Unlike single-objective optimization, which searches for a single optimal solution, MOO aims to find a set of optimal solutions, known as Pareto optimal solutions. These solutions represent trade-offs in which improving one objective may lead to the deterioration of another. Most architectural design problems have multiple objectives and sometimes they conflict with each other, i.e., minimize cost, maximize functionality, etc.

Several multi-objective optimization (MOO) methods are applied when an optimal decision needs to be made in the presence of trade-offs between conflicting objectives. One traditional method for solving MOO is the weighted sum method, which combines all objectives into a single function using the sum of weighted objectives, as shown in the following equations (Yang, 2014).

$$F(x) = w_1 f_1(x) + w_2 f_2(x) + \dots + w_n f_n(x)$$

where,

$$\sum_{i=1}^n w_i = 1, \quad w_i \in (0,1).$$

This method is one of the simplest and most widely used approaches in MOO which is effective when dealing with convex Pareto fronts and can be readily implemented due to its uncomplicated

nature. However, the weighted sum method faces significant limitations, which include the arbitrary selection of weighting coefficients that are dependent on decision maker preferences. This dependency leads to potential biases and suboptimal solutions. The method also has difficulties with non-convex Pareto fronts and demands precise scaling or normalization of objectives to distribute weights correctly. Without such adjustments, it inadequately samples the Pareto front, skewing potential solutions. For more complex issues, more robust methods are preferable (Yang, 2014).

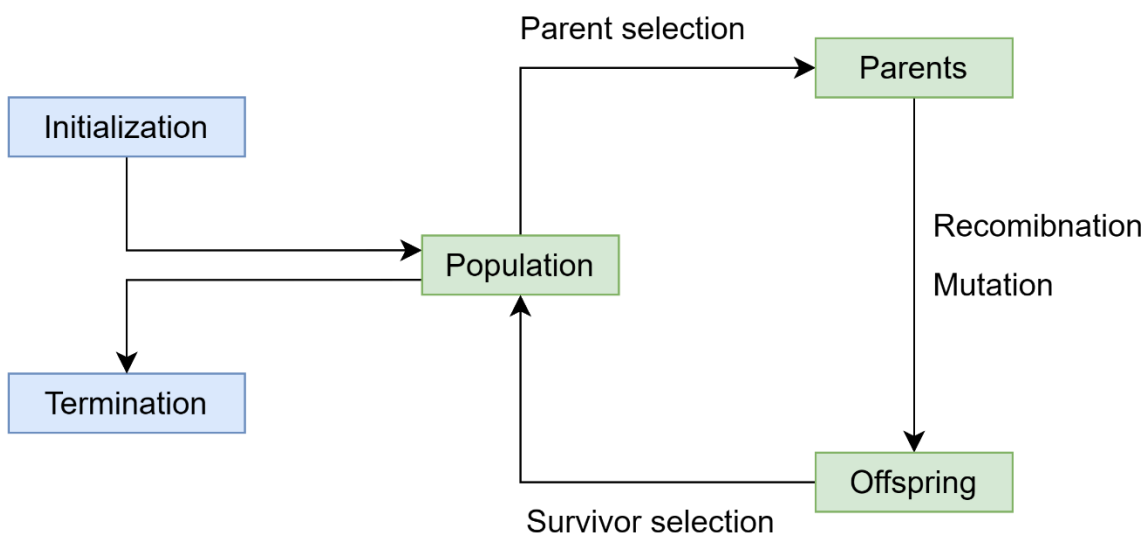


Figure 2-8. General scheme of an evolutionary algorithm (Adapted from Eiben & Smith, 2015)

Evolutionary Algorithms (EAs) are optimization algorithms inspired by natural evolutionary processes, like selection, mutation, and crossover. (Eiben & Smith, 2015). EAs are frequently used to solve MOO problems because they can handle a collection of potential solutions simultaneously, enabling the identification of multiple members of the Pareto optimal set in a single execution (Coello & Lamont, 2004). This is achieved by prioritizing nondominated solutions—those that are not outperformed in all objectives—over dominated ones, thereby focusing on high-quality solutions that represent optimal trade-offs among conflicting objectives. Additionally, EAs employ

strategies such as crowding distance, fitness sharing, and niche formation to maintain diversity within the population. This diversity preservation is crucial as it ensures a comprehensive exploration of the solution space, contrasting with traditional mathematical programming techniques that require multiple separate runs to obtain a comparable spectrum of optimal solutions (Coello & Lamont, 2004).

Genetic Algorithm (GA) is a subset of EAs developed by Holland and his colleagues in their book *Adaptation in Natural and Artificial Systems* (Holland, 1975). Some of the algorithms that utilize genetic algorithm to solve MOO problem include Vector Evaluated Genetic Algorithm (VEGA), the first multi-objective GA proposed by Schaffer, Nondominated Sorting Genetic Algorithm (NSGA), and Strength Pareto Evolutionary Algorithm (SPEA).

VEGA(Schaffer, 1985), the first multi-objective genetic algorithm, works by separately evaluating each objective of a multi-objective problem in different subpopulations, then combining these subpopulations to promote diversity in the solutions. Its implementation is straightforward, yet a significant disadvantage is its tendency to converge toward the extremes of each objective, potentially overlooking balanced solutions (Zitzler & Thiele, 1999a).

NSGA (N. Srinivas & Deb, 1994) improves upon earlier methods by sorting solutions into different levels based on dominance. Each solution is compared with others to identify which are nondominated, grouping them into a hierarchy from best to worst based on these comparisons. This approach offers fast convergence, which is a significant advantage. However, a weakness of NSGA is its sensitivity to the sharing factor  $\sigma_{share}$ , also known as the niche size, which refers to the conceptual area in the solution space where similar solutions are grouped (Coello Coello, 1999).

SPEA (Zitzler & Thiele, 1999b) uses an external archive to store nondominated solutions and assigns a "strength" value to each solution based on how many others it dominates. The method uses a density estimation technique as a method of diversity preservation by avoiding overcrowding, and also uses clustering to manage the external archive keeping only the most representative solutions when it exceeds its capacity. While SPEA is robust and capable of generating diverse, high-quality solutions, it can be computationally intensive and sensitive to parameter settings.

### 2.3.3 NSGA-II

The Fast Nondominated Sorting Genetic Algorithm (NSGA-II) is one of multi-objective GAs first introduced by Deb et al. (2002) that addresses the limitations of the original NSGA. The iterative loop of NSGA-II is described in Figure 2-9.

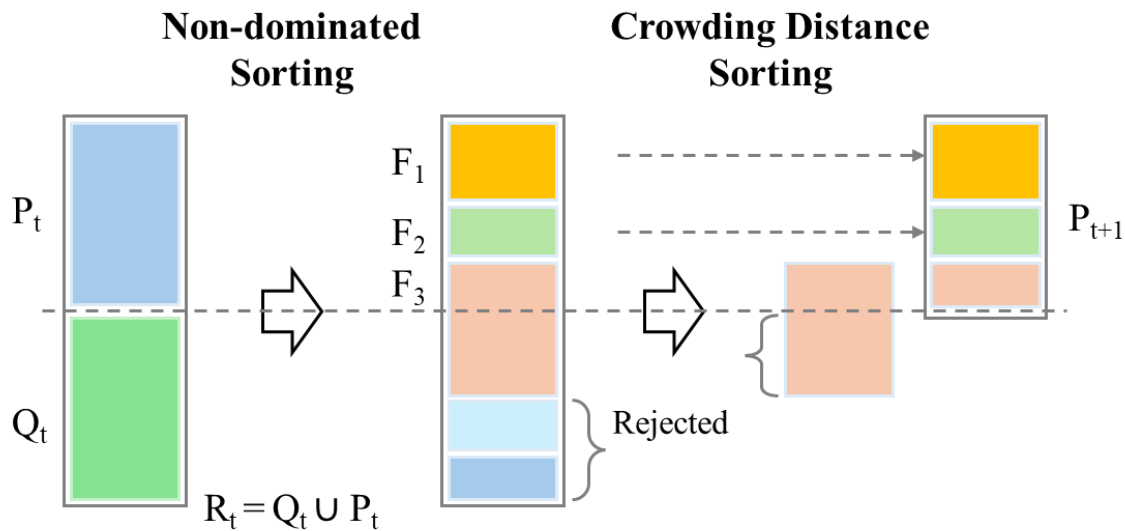


Figure 2-9. NSGA-II procedure (Adapted from Deb et al., 2002)

The initial process is to populate an offspring population  $Q_0$  of size  $N$  using the binary tournament selection, crossover, and mutation. After the first population  $R_t$ , which is  $Q_t \cup P_t$  with the

population size of  $2N$ , is sorted into  $F_1, F_2, \dots, F_i$  in order of best non-domination, solutions from the best set is chosen until the total population size becomes  $N$ . To choose exactly  $N$  population members, the last subsequent set  $F_i$  is sorted using the crowding distance sorting to fill all population slots. The new population  $P_{t+1}$  is then used for selection, crossover, and mutation to create a new  $Q_{t+1}$ . The loop is finished once the sorting procedure has found enough number of fronts to have  $N$  members in  $P_{t+1}$ .

Non-dominated sorting is a classification method used in MOO that sorts potential solutions into different levels, or “fronts”, based on pareto dominance. Each solution is compared against others to identify whether it is dominated, meaning there is at least one other solution better in one objective without being worse in others. Solutions that are not dominated by any other form the first front  $F_1$ , representing the Pareto-optimal set. Subsequent fronts  $F_2$  are determined by temporarily removing solutions of the previous front and repeating the process with the remaining solutions (Deb et al., 2002).

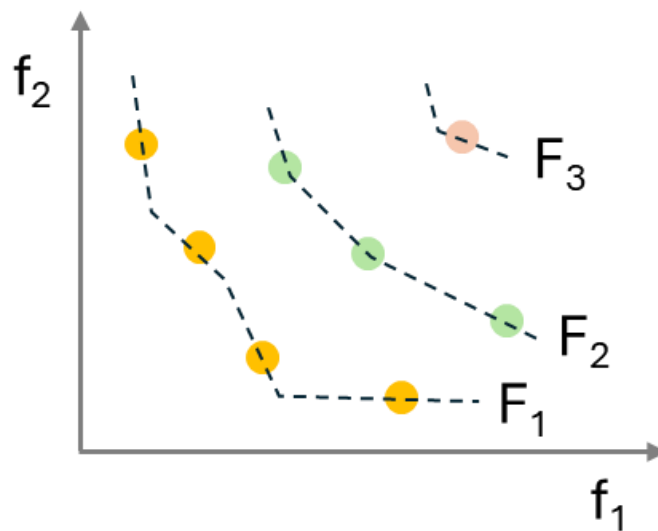


Figure 2-10. Pareto fronts

One characteristic distinguishing NSGA-II to other MOEAs is the fast nondominated sorting. The computational complexity  $O(MN)$  of a MOEA to categorize solutions into Pareto fronts is determined by the number of objectives and the population size, which is represented as  $M$  and  $N$ , respectively. The original NSGA has a computational complexity of  $O(MN^3)$  in the worst case, when there exists only one solution in each front, on the other hand, NSGA-II has more efficient non-dominated sorting method with a complexity of  $O(MN^2)$  (Deb et al., 2002). This improvement significantly reduces computational overhead, especially for large populations.

In order to maintain diversity, crowding-distance sorting is used which does not require user-specified parameters. Density estimation is used to assess solution proximity within the population, where the crowding distance of a solution is the average side length of a cuboid, represented in the Figure 2-11.

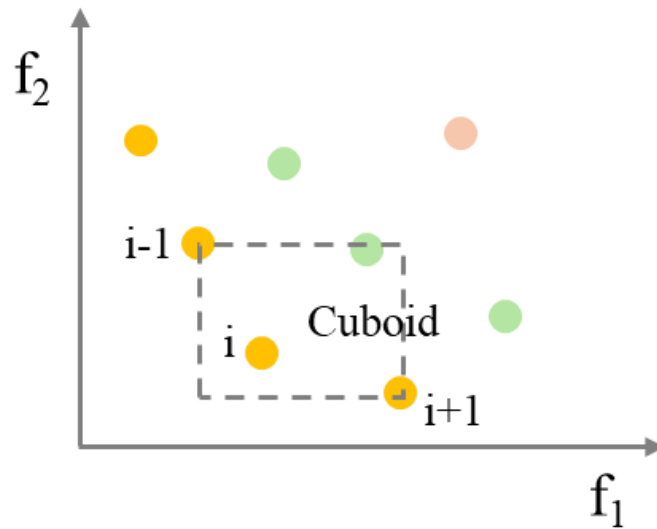


Figure 2-11. Crowding distance of a solution

Each solution has two attributes: nondomination rank( $i_{rank}$ ) and crowding distance( $i_{distance}$ ). The crowded-comparison operator ( $<_n$ ) is shown below, where the solution with a lower(better) non-

domination rank or, in case of a tie in rank, a greater crowding distance is preferred for the next generation. This concept can be expressed as following with the solution  $i$  being preferred to the solution  $j$ .

$$i \prec_n j \text{ if } (i_{rank} < j_{rank}) \text{ or } ((i_{rank} = j_{rank}) \text{ and } (i_{distance} > j_{distance}))$$

The combined parent and offspring populations undergo non-dominated sorting into Pareto fronts. They are then subject to crowding distance sorting within each front in order to select a diverse set of elite solutions for the next generation. The algorithm ensures that the most elite solutions are preserved, directly encouraging the propagation of high-quality solutions to the next generation, and indirectly promoting genetic diversity.

### 3 Methodology

#### 3.1 Overview

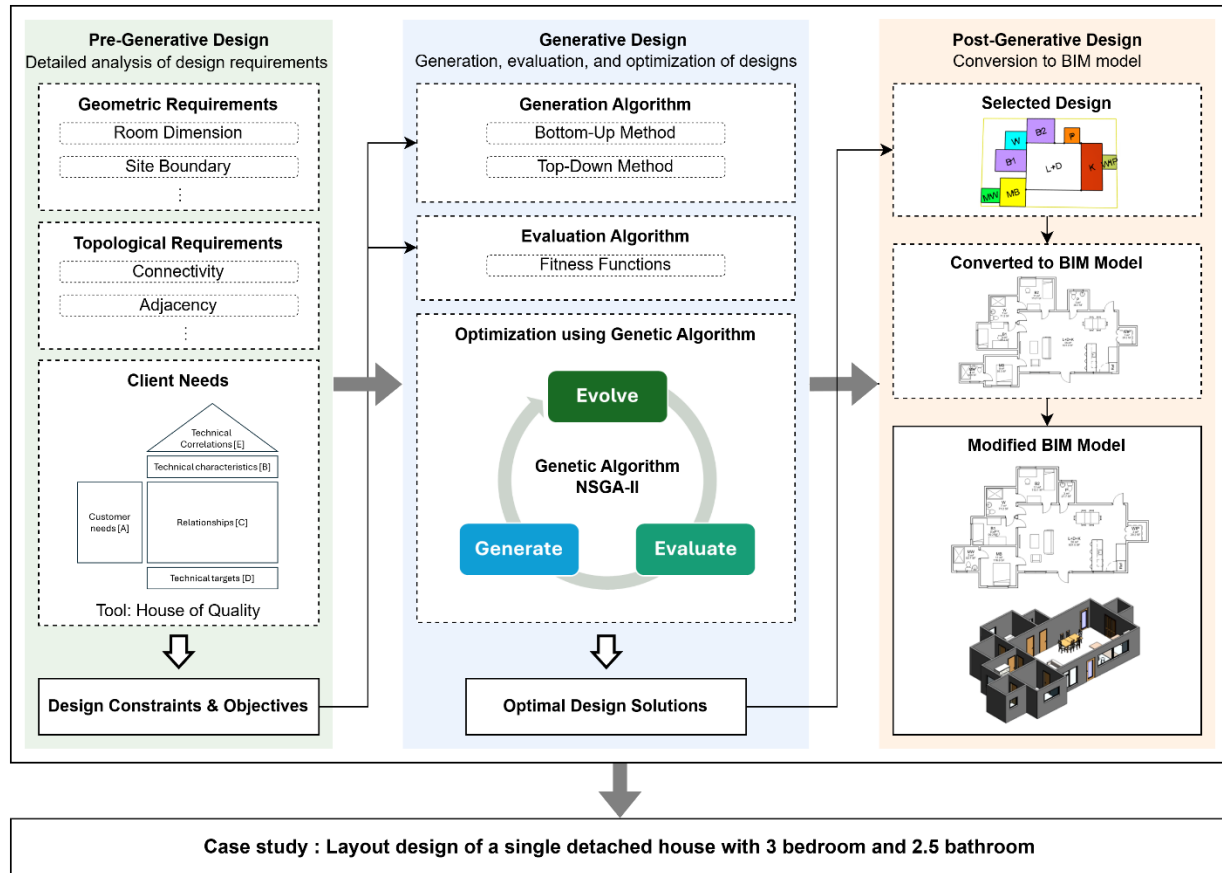


Figure 3-1. Overview of the methodology

An overview of the methodology implemented in this research is presented in Figure 3-1, divided into three stages: pre-generative design, generative design, and post-generative design. Pre-generative design mainly focuses on a detailed analysis of design constraints and objectives that are used in the generation and evaluation algorithms developed for the generative design process. After optimization of the design using genetic algorithm, specifically NSGA-II, the selected optimal design is converted into a Building Information Modelling (BIM) model and modified as needed in the post-generative design process. Modifications required by human intervention may be due to factors such as structural reinforcement, spatial layout reconfiguration, and aesthetic



enhancement. A case study of designing a single detached house with 3 bedrooms and 2.5 bathrooms is conducted using the methodology.

### **3.2 Software – Revit/Dynamo/Generative Design**

The Autodesk product family Revit, Dynamo, and Generative Design are used in this research. Revit is a Building Information Modelling (BIM) software used to design, document, visualize, and deliver architecture, engineering, and construction (AEC) projects that holds the major market share in the industry worldwide.

Dynamo for Revit, a visual programming application, extends Revit capabilities by enabling users to construct custom algorithms. These algorithms can be employed in various applications, ranging from data processing to geometry generation, all in real time without the need for extensive textual programming. Dynamo for Revit was developed to simplify AEC workflows by making Revit's data more accessible to all users, through a user-friendly graphical interface. Dynamo allows users to automate repetitive tasks and foster design exploration by combining its core nodes with custom Revit ones, thereby extending parametric workflows for various applications like documentation, analysis, and design generation. Dynamo was selected for this study due to its capability to generate geometric representations of spaces and seamlessly convert them into BIM elements, enabling an efficient end-to-end process.

Generative Design is another add-on application of Revit for generative design workflows that allows users to generate, evaluate, and optimize designs created from Dynamo script. Generative Design has different types of “solvers”, namely ‘randomize’, ‘optimize’, ‘cross product’, and ‘like this’. ‘Randomize’ produces a predetermined number of design variations by randomly assigning values to each input parameter. ‘Optimize’ conducts an optimization analysis where Generative

Design iteratively refines the design based on the evaluations' outcomes and metrics. This optimization involves creating several 'generations' (or iterations) of a design, each iteration leveraging the input setup from the previous generation to enhance the new design options. For optimization, NSGA-II is used as multi-objective genetic algorithm. 'Cross product' enables the exploration of the full design space by combining every step of each parameter with all other available parameters. 'Like this' directs Generative Design to introduce minor adjustments to the current input configuration. This method allows for the exploration of different variants of a design that is already favourable.

### **3.3 Design requirement analysis**

#### **3.3.1 Quality Function Deployment (QFD) and House of Quality (HoQ)**

When capturing client requirements in design, clients may articulate their needs in broad or subjective terms, which can be open to interpretation and lack the specificity required for technical development. Without a systematic method to translate these requirements into measurable design specifications, there is a risk of misalignment between the client's vision and the final product. Quality Function Deployment (QFD) offers a structured solution to this limitation. QFD refers to an overall concept of converting customer requirements into technical characteristics systematically, ensuring that the development process aligns with the client's expectations. QFD is known to be originated from Japan in the late 1960s and early 1970s, when quality control and improvement were of high importance in the manufacturing industry. Then QFD was introduced to the U.S. approximately 10 years later, and gained significant traction after the article titled "The house of quality" co-authored by Clausing and Hauser was published in the Harvard Business Review in 1988 (Hauser & Clausing, 1988). The initial U.S. case study employing QFD was conducted by Kelsey Hayes in 1986 for designing a coolant sensor (Chan & Wu, 2002).

House of Quality (HoQ) is a foundational tool in QFD that serves as a visual and inter-functional planning matrix, which translates customer needs (WHATs) into design specifications (HOWs).

Figure 3-2 shows the template of HoQ and comprising rooms (Hauser & Clausing, 1988).

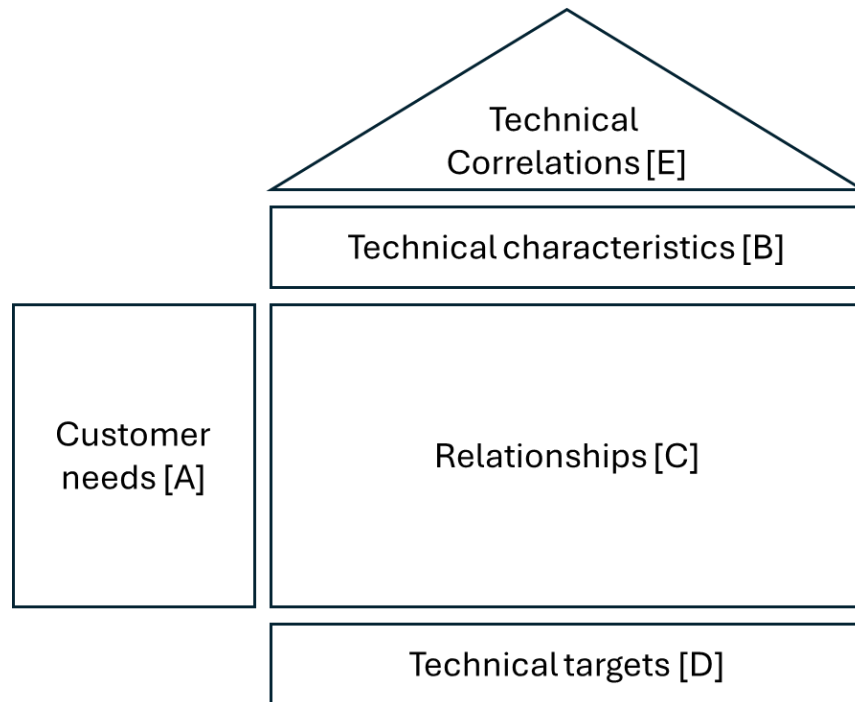


Figure 3-2. House of Quality (Adapted from Delgado-Hernandez et al., 2007)

The following steps are used to build HoQ, with reference to Figure 3-2:

Step 1: Identify the customer's needs (WHATs) [A]. Customers include not only the end users but also all stakeholders of the design. The identification process can be done from interviews or other techniques. Since the needs can contradict each other and have trade-offs, relative importance weights should be set on each need.

Step 2: The design team now determines how to satisfy the customer's needs [B]. Technical characteristics (HOWs) that will affect the customer needs are listed in measurable terms and marked whether the objective is to reduce, increase, or meet the target.

Step 3: Figure the relationship between customer needs and technical characteristics and establish target values [C, D]. The characteristics can affect more than one needs in either positive or negative way and the relationships are marked in the relationships table with symbols.

Step 4: Identify the relationships between HOWs [E]. The roof matrix aids in understanding how one design change can impact other characteristics, necessitating a balance of trade-offs. This process is crucial for creatively solving problems while balancing different objectives.

HoQ first started in the manufacturing industry and has since been used in many other industries including construction in application of the management of design, safety, schedule, etc. Dikmen et al. (2005) conducted a case study that utilized QFD as a strategic tool for post-construction marketing decisions in housing projects. They formed a QFD team to gather customer expectations for a high-rise building in Ankara, Turkey. The findings highlighted that aspects like the housing complex's location, architectural layout, and security features were critical to project success and should be emphasized in marketing. Fargnoli et al. (2020) utilized QFD to assess and improve safety in construction by examining hazard types, hazardous events, and potential consequences. The methodology offered a detailed analysis of safety risks associated with specific tasks and concluded with the need to address minor injuries more effectively. Elhegazy et al. (2021) applied QFD in Egypt's construction industry, focusing on the structural system's optimization. The study highlighted QFD's role in enhancing decision-making for structural systems based on key performance indicators that were aimed at maximizing client satisfaction. Gunduz and Al-Naimi (2022) developed a framework combining the Balanced Scorecard (BSC) and QFD to effectively manage construction projects and reduce delays. By identifying and prioritizing factors from the financial perspective and delay mitigation enablers, the framework helps professionals focus on crucial elements, leading to more efficient resource use.

There is also a study that criticized low awareness of QFD among professionals in the construction field. John et al. (2014) investigated QFD's introduction in the Nigerian construction industry, primarily focusing on design and build projects. The study found inadequate training and management practices. However, those familiar with QFD recognized its benefits for enhancing client satisfaction by effectively capturing and prioritizing client needs. The research suggests that the wider adoption of QFD could significantly improve project outcomes, emphasizing the need for increased awareness and training within the industry.

### **3.3.2 Geometric and topological constraints**

In layout design, or space allocation problem, geometric requirements such as physical dimensions and shapes of spaces, including area, volume, height, and the specific geometry of each space are to be established. Geometric requirements ensure that each space is adequately sized for its intended purpose.

Topological requirements relate to the relationships and connections between different spaces, such as adjacency, connectivity, and circulation. Topological considerations help determine how spaces interact with each other, the flow between areas, and how they are accessed. Transforming these considerations into graph data is an intuitive method that helps visualize and analyze these spatial relationships. Such a graph is called justified plan graph (JPG), in which spaces are represented as nodes, and connectivity or adjacency is represented as edges (Figure 3-3). Generating the graph begins by creating the nodes and defining a core node. A core node functions as a core unit (Koning & Eizenberg, 1981) that serves as the foundational element around which the structure's layout is designed. This core node influences the overall flow within the layout, forming a central hub from which other rooms and zones radiate. Then links are established from this core node to adjacent nodes until all links are added (Lee et al., 2015). JPGs are particularly

useful in architectural planning because they simplify complex spatial systems, allowing for easier layout modification. In regard to the generation algorithms developed in this research, the bottom-up method uses JPG data to incrementally build layouts by focusing on the connectivity (access) between spaces, rather than starting from a predetermined boundary which is the case of the top-down method, the other generation algorithm developed.

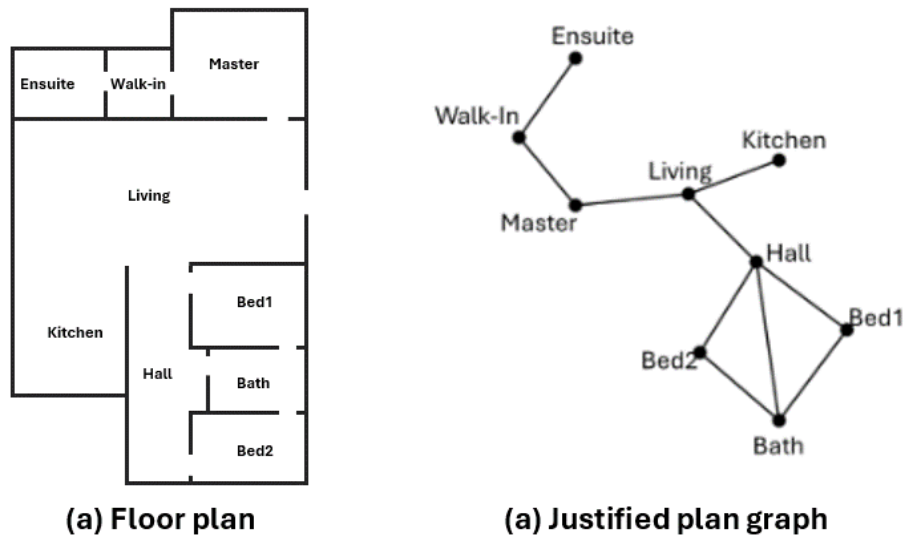


Figure 3-3. An example of floor plan and its graph representation

There have been numerous research studies that have used graph transformation to generate architectural layout designs. For instance, Wang et al. (2018)'s approach derived a dual graph from a floor plan then transformed it into a floor plan using the automatic generation method they developed that has additional transformation rules, such as the addition rule and subtraction rule, to have modified floor plans corresponding to specific requirements. Bisht et al. (2022) developed a software called G2PLAN that takes adjacency graph as input and transforms it into a dimensioned floor plan, utilizing graph theoretical concepts such as biconnected graphs and planar graphs.

### 3.4 Dynamo script for generation algorithm

This chapter describes the flow of the developed Dynamo script for generating algorithms, namely top-down method and bottom-up method. Table 3-1 compares those two methods in terms of their design approach and computational efficiency.

Table 3-1. Top-down method and bottom-down method

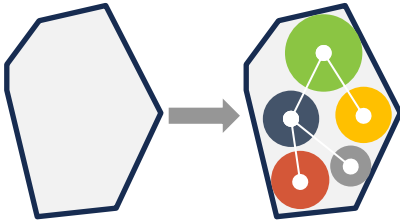
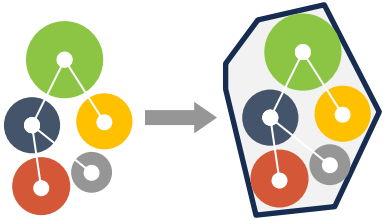
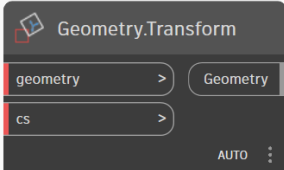
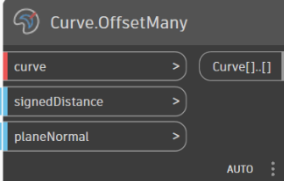
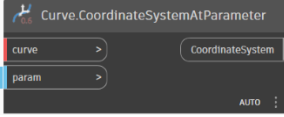
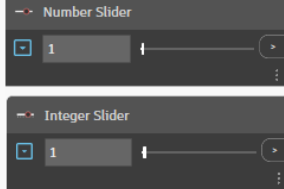
	Top-Down Method	Bottom-Up Method
<b>Schematic Diagram</b>		
<b>Initial Approach</b>	Starts with a given boundary (e.g., building mass)	Starts with individual spaces
<b>Design Logic</b>	Assigns spaces inside the boundary	Aggregates spaces to form a layout
<b>Computational Demand</b>	Less, due to restricted search space	More, due to large search space
<b>Flexibility</b>	Low, limited by a predefined boundary	High, allows for various combinations
<b>Design Strategy</b>	Grid-based system	Connectivity constraint

Table 3-2 shows the definitions of geometric elements and description for some of the nodes that are used in this research's Dynamo script to create and modify geometries:

Table 3-2. Dynamo nodes for generating geometries (Autodesk, n.d.)

Geometry/Node	Description
Point	A location in 3D space using X,Y, and Z coordinates
Curve	A continuous path defined by a sequence of points or a mathematical formula
Rectangle	A rectangle defined by its width, height, and orientation in the coordinate system
CoordinateSystem	A spatial reference frame, consisting of an origin point and direction vectors for axes
	Transform geometry by the given CoordinateSystem's transform
	Offset a Curve by a specified amount
	Get a CoordinateSystem with origin at the point at the given parameter.
	A slider that produces numeric/integer values



### 3.4.1 Bottom-up method

The bottom-up method is used when there is a strong constraint in connectivity between spaces. This method starts from an individual space and builds up to the complete structure. Custom residential homes, healthcare facilities, and retail complexes can be the types of projects that this method is suitable for.

Table 3-3. Constraints, variables, and objectives of the bottom-up method

<b>Constraints</b>	<ul style="list-style-type: none"><li>• Number of spaces to be allocated</li><li>• Dimension of spaces</li><li>• Connectivity between spaces</li></ul>
<b>Variables</b>	<ul style="list-style-type: none"><li>• Coordinates of each space</li></ul>
<b>Objectives</b>	<ul style="list-style-type: none"><li>• Objectives determined by QFD</li></ul>

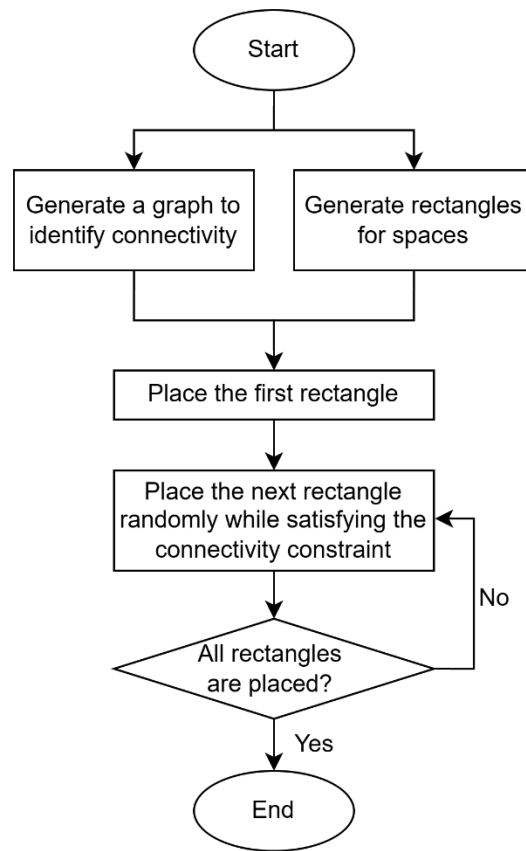


Figure 3-4. Flowchart of the bottom-up method

As discussed in Section 2.2.1, the bottom-up method inherently allows for significant geometric freedom, which typically results in substantial computational cost to explore possible dimensions or coordinates. In this research, the introduction of connectivity requirements as a heuristic constraint from the start effectively limits the search space and reduces the computational cost. Figure 3-4 shows the flowchart of the algorithm.

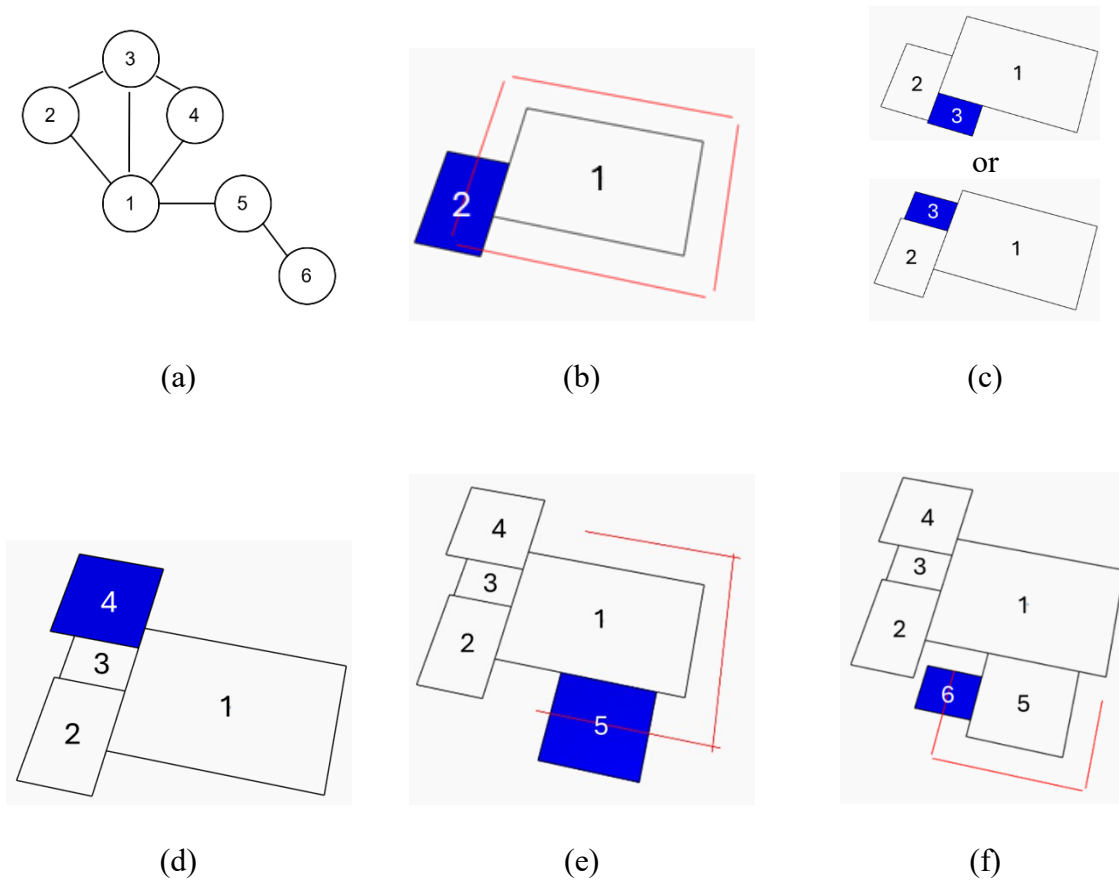


Figure 3-5. Sequence of the bottom-up method

The first step of writing the script for the bottom-up method is to generate a graph from the topological requirement that is to be achieved, shown in Figure 3-5(a). A graph is a great visual representation of the connectivity constraint between spaces and is the foundation of the bottom-up method. For the first rectangle to place, the largest or the one with most connections are

preferred as it serves as a core unit. The second rectangle is placed such that it is positioned at a point along the offset line, marked as red lines in Figure 3-5(b) of the first rectangle, with the offset amounting to half the width ( $w/2$ ) of the second rectangle. The Dynamo node “Curve.CoordinateSystemAtParameter” selects a point on a curve at a specified parameter, ranging from 0 to 1. For example, setting the parameter to 0.5 returns the midpoint of the curve. Additionally, the script developed for the generation algorithm in this study is written to adjust the length of the offset lines so that the minimum length of the shared edge meets a user-defined dimension, referred to as the “door size”. The placement of the rest of the rectangles is determined by their specific connectivity constraints with adjacent rectangles. For instance, in Figure 3-5(c), rectangle 3 is connected to 1 and 2, positioning it at the intersection of their corners to satisfy these connections. Rectangle 4, having connections to Rectangles 1 and 3, is placed at the only possible location that maintains these connections, as shown in Figure 3-5(d). Similarly, rectangles 5 and 6, each connected to 4 and 5 respectively, follow the red offset lines as potential paths for placement based on their single connection, as shown in Figure 3-5(e) and Figure 3-5(f).

### **3.4.2 Top-down method**

The top-down method is used for layout generation when the boundary is a hard constraint. This method works well with buildings that require a uniform layout throughout such as high-rise residential developments due to the repetitive nature of the unit layouts within a fixed building footprint. Figure 3-6 shows the flowchart of the algorithm.

Table 3-4. Constraints, variables, and objectives of the top-down method

<b>Constraints</b>	<ul style="list-style-type: none"> <li>• Site boundary</li> <li>• Number of spaces to be allocated</li> <li>• Dimension of spaces</li> </ul>
<b>Variables</b>	<ul style="list-style-type: none"> <li>• Coordinates of each space</li> </ul>
<b>Objectives</b>	<ul style="list-style-type: none"> <li>• Connectivity</li> <li>• Objectives determined by QFD</li> </ul>

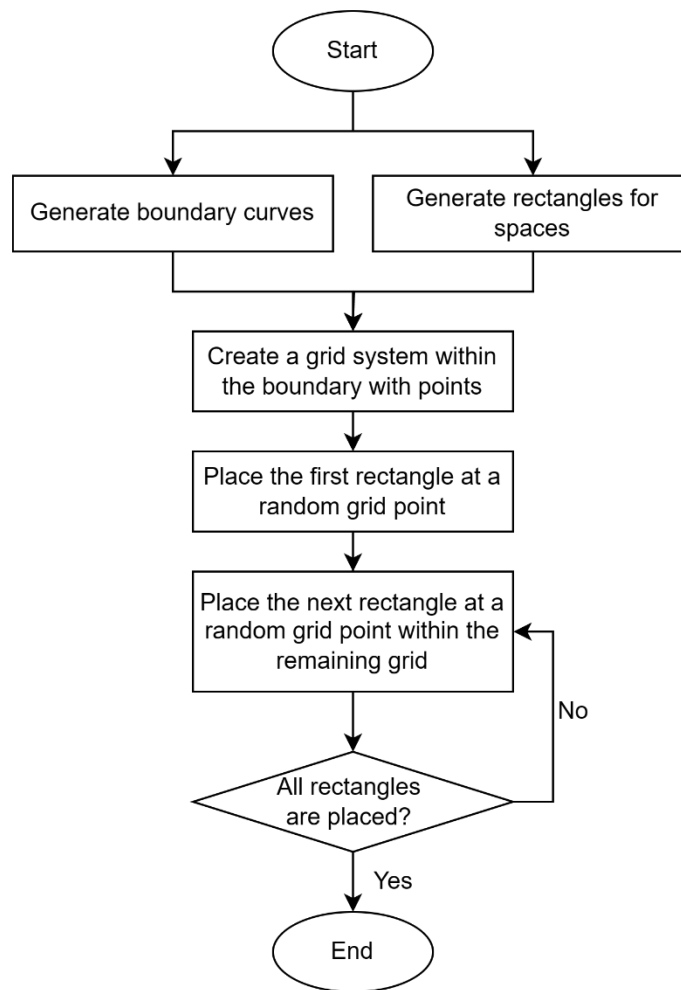


Figure 3-6. Flowchart of the top-down method

The first step of writing the script for the top-down method is to generate curves representing the boundary that each rectangle can be placed inside. Then, a grid system is created by generating points in two-dimensional directions (Figure 3-7(a)). The denser the points are, the larger the

search space is, giving more flexibility in design but at a higher computational cost. After that, selecting the first rectangle to place determines the available points where its center can be positioned, as shown in Figure 3-7(b). The Dynamo code is written in the way to only leave feasible points to keep the rectangle from getting out of the given boundary (Figure 3-7(c)). The points are the variables in this method, working as an input parameter in the generative design process. The engine will explore within the range of the points and choose one point by varying the number slider. The same process is performed until all rectangles are placed in the remaining bounded area, as seen in Figure 3-7 (d)-(f).

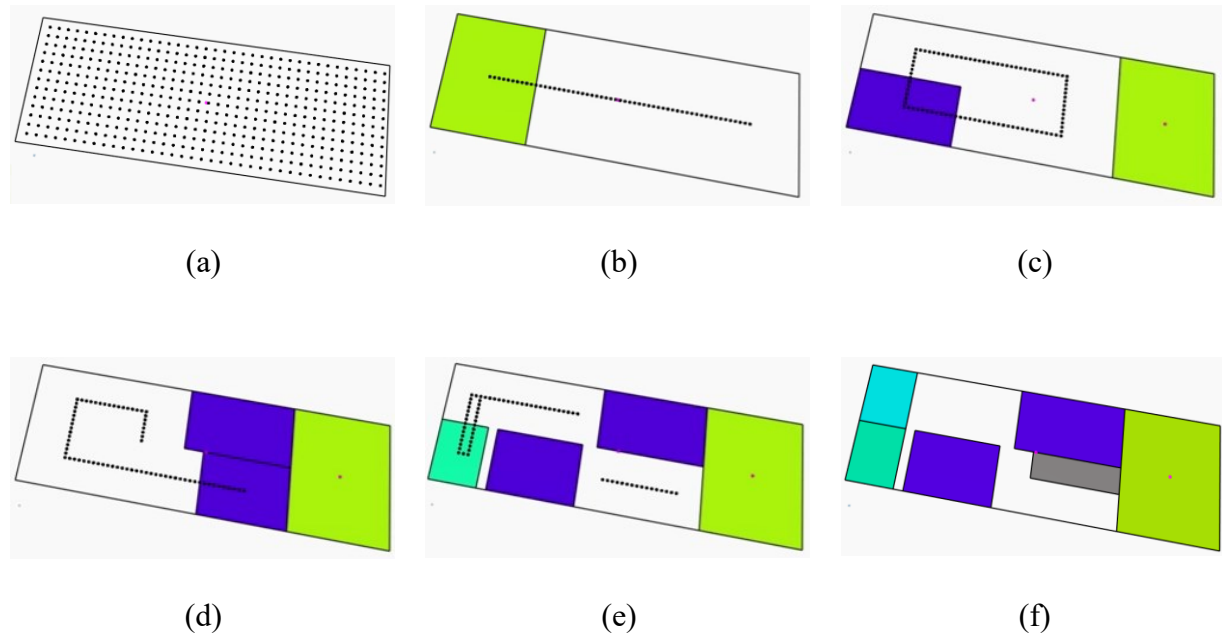


Figure 3-7. Sequence of the top-down method

### 3.5 Dynamo script for evaluation algorithm

The objectives determined from the House of Quality (HoQ), along with the geometric and topological requirements, are foundational to the evaluation algorithm developed in this study. Each objective derived from the HoQ should be converted into a measurable metric that acts as a fitness function in a multi-objective optimization framework.

For instance, if one objective is to have a large south-facing window in the living room, the corresponding fitness function could be defined to maximize the length of the living room wall that is parallel to the x-axis, has a lower y-coordinate (indicating a south-facing wall, based on the orientation settings where higher x and y coordinates represent eastward and northward directions, respectively) and does not share any section with other rooms. In Figure 3-9, the red line indicates the south-facing wall of the living room uninterrupted by other rooms. The dynamo script is written so that if the south-facing wall (A) intersects with other walls (B) of other rooms, the returned value is the length of A minus length of B (Figure 3-8). The lengths of the red lines in Figure 3-9(a) and Figure 3-9(b) represent the lengths of certain walls, which are 8 and 16 units, respectively. Other objectives can also be converted into quantifiable metrics to be fed into generative design process.

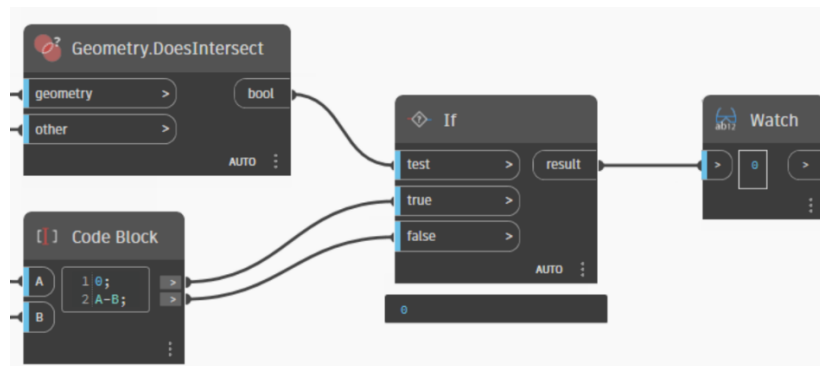


Figure 3-8. Dynamo script for calculating the fitness

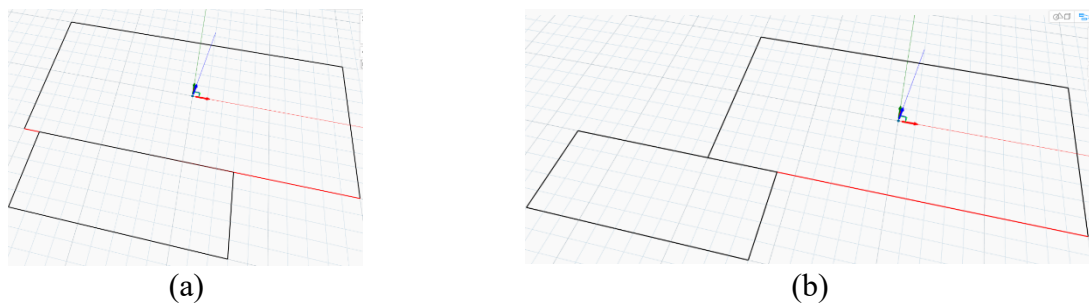


Figure 3-9. Numeric objective visualized as a geometry

### 3.6 Design optimization process

The design optimization process utilizes generative design facilitated by Dynamo and Autodesk's Generative Design software. The optimization process using generative design requires input parameters which enable to randomly generate different design solutions and output parameters that guide the software to achieve better performance in terms of fitness to predefined objectives. The iterative process enables continuous refinement of designs. Adjustments are made based on the performance outcomes, gradually steering the solutions towards optimal design configurations. Numerical scores from each iteration help assess which designs best meet the fitness. To run the Autodesk Generative Design, setting input and output parameters in Dynamo is required as shown in Figure 3-10.

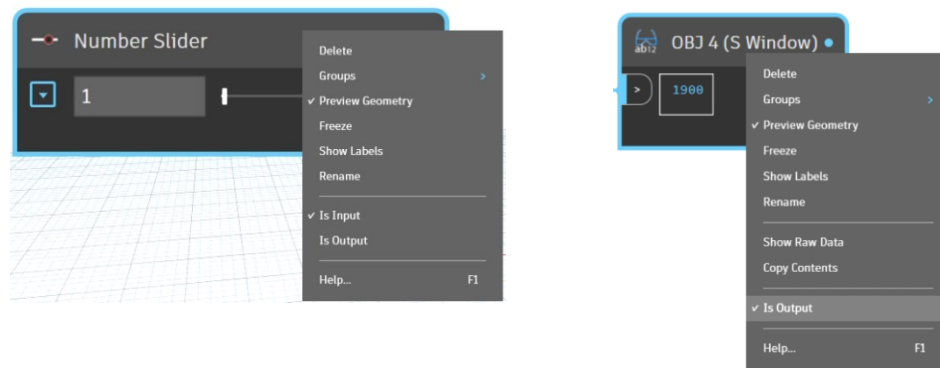


Figure 3-10. Input/output setting in Dynamo

Figure 3-11 displays a sample user interface when 'optimize' is chosen as the generative method. When running the software using the optimization solver, numbers for population, generation and seed must be set. 'Population size' refers to the initial number of solutions for the genetic algorithm to start with, and 'Generations' refers to the rounds of iteration. 'Seed' is a number that uses an internal random number generator to establish a starting point for generation of the initial population and other logic, including crossover, mutation, and selection.

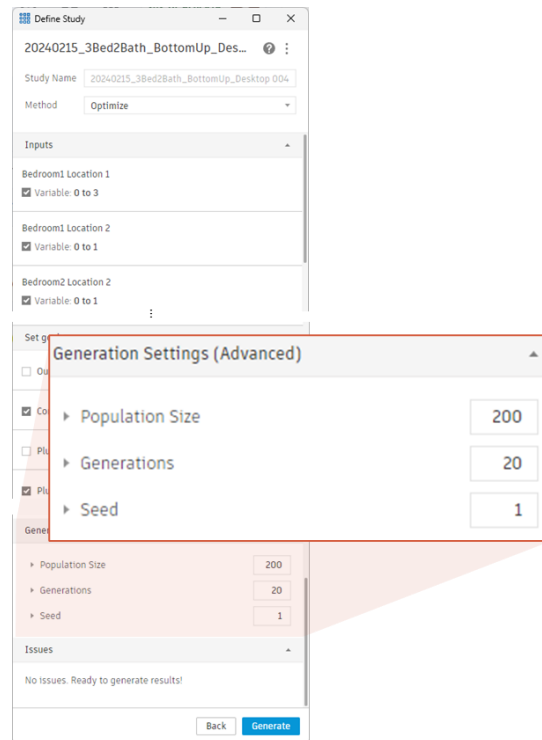


Figure 3-11. Sample Generative Design user interface

After running the software using the optimization solver, visual outputs and numerical scores from each output are generated, as seen in Figure 3-12. The parallel coordinates chart displays and analyzes multi-dimensional data sets, which are particularly useful for simultaneously exploring and comparing multiple design solutions. Each axis represents a different dimension, and each line represents a design, allowing users to quickly identify trends, patterns, and trade-offs between different parameters. Users can interact with the parallel coordinates chart to filter out designs based on specific criteria. By selecting ranges on different axes, users can focus on designs that meet certain thresholds or performance metrics, effectively narrowing down the choice set to the most viable options.





adjusted to reflect essential construction-related attributes that were perhaps overlooked during the generative phase. Additionally, materials are selected to meet specific construction standards and practical needs, ensuring that the final BIM models are not only accurate in their geometric representation but are also fully compliant with regulatory requirements and optimized for real-world application. This transformation allows the geometries generated in Dynamo to evolve into detailed and information-rich BIM models. These models facilitate further architectural processes, such as structural analysis, quantity takeoff, and construction planning, ensuring that the design is not only optimized but also ready for practical applications.

The following are some of the Dynamo nodes that bring geometries into Revit environments.

Table 3-5. Dynamo nodes for converting geometries into BIM elements (Autodesk, n.d.)

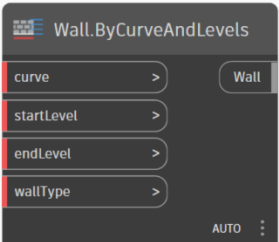
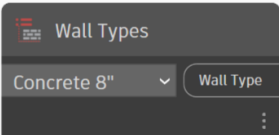
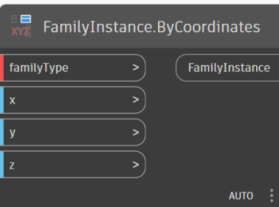
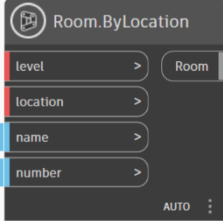
Node	Description
	Create a Revit Wall from a guiding Curve, start Level, end Level, and WallType
	All wall types available in the document
	Place a Revit FamilyInstance given the FamilyType (also known as the FamilySymbol in the Revit API) and its coordinates in world space
	Create a Revit Room Element

Figure 3-13 and Figure 3-14 show the sample Dynamo script, and the result of converting geometries in a Dynamo environment into BIM elements in a Revit environment.

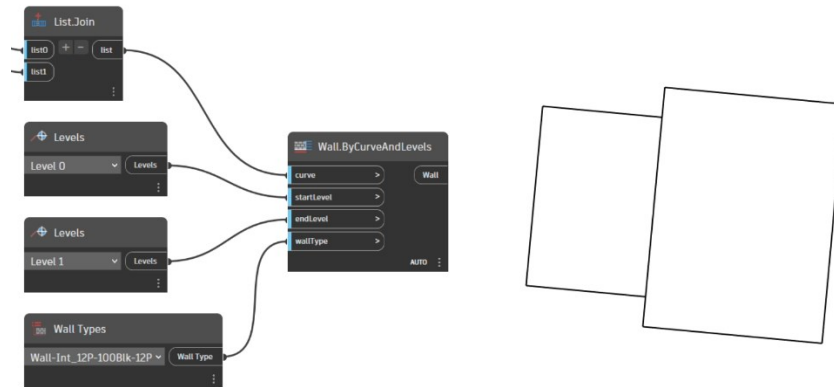


Figure 3-13. A sample Dynamo script for converting curves into walls

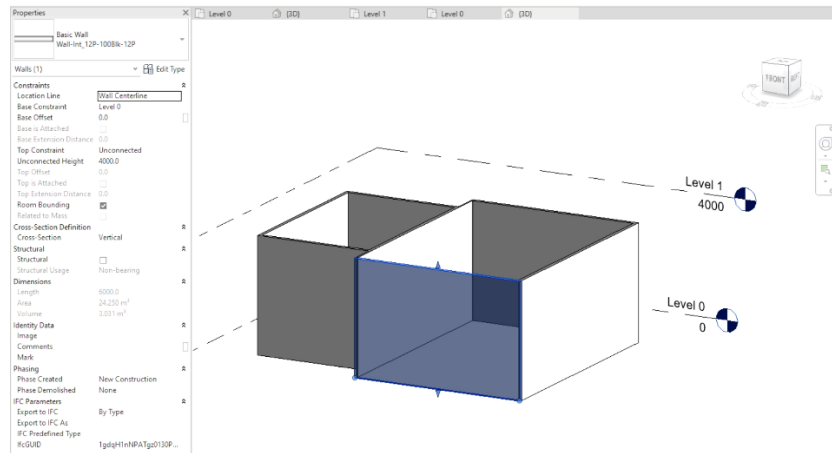


Figure 3-14. Converted Walls and corresponding properties

## **4 Case study**

In this chapter, a case study of designing a single detached house with 3 bedrooms and 2.5 bathrooms is performed using the top-down method and bottom-up method.

### **4.1 Pre-GD: Constraints and objectives**

#### **4.1.1 Building House of Quality**

##### **Step 1: Identify customer's requirements (WHATs)**

The initial step in the design process involves gathering information about the requirements of internal and external customers. Those requirements may include legal regulations such as building codes and safety standards, functional requirements suggested by engineers, or personal preferences of the client. In this study, these needs are assumed based on general assumptions rather than direct client input. Future studies could enhance the validity of these assumptions by conducting detailed interviews with clients to directly assess their preferences. Once the requirements are gathered, customers are also asked to assign each requirement with the relative importance ranging from 1 to 5, 5 being the highest importance.

Table 4-1. Customer's requirements and their relative importances

Space	Requirements	CR <sub>i</sub>	RI
Entrance	Two entrances	CR1	5
Kitchen	Ample countertop space for food preparation	CR2	3
	Efficient workflow between stove, sink, and refrigerator	CR3	4
	Walk-in pantry	CR4	5
	Natural light	CR5	2
	Open concept kitchen	CR6	5
Living Room	Space for full-size dining table	CR7	3
	Natural light & view	CR8	5
	Comfortable seating arrangements	CR9	2
	Fireplace	CR10	1
Powder	Ventilation	CR11	1
Bathroom	Ventilation	CR12	3
	Storage	CR13	2
Master Bed	Walk-in closet	CR14	4
Ensuite	4piece (sink, toilet, shower, tub)	CR15	3

## Step 2: Develop technical characteristics (HOWs)

The next step in this process is to find the technical characteristics for each customer's requirement. Each technical characteristic has a target whether it is aimed to be increased, decreased, or attained. Each is also categorized depending on the design phase that it should be considered. For a kitchen, “adding a walk-in pantry” should be considered in the schematic design phase as it requires an addition of a space, where “ergonomic work triangle layout” can be considered in the design development phase.

Table 4-2. Technical characteristics and their target and design phase

CR <sub>i</sub>	TC <sub>i</sub>	Technical Characteristics	Target	Design Phase
CR1	TC1	One separate entrance to backyard	—	SD
CR2	TC2	Total base cabinet length: minimum 120 inch	▲	DD
CR3	TC3	ergonomic work triangle layout	—	DD
CR4	TC4	Add walk-in pantry	—	SD
CR5	TC5	Install a window facing south	—	SD
CR6	TC6	Open layout with integrated living, dining, and kitchen areas	—	SD
CR7	TC7	Enough space for a full-size dining table near kitchen	▲	SD
CR8	TC8	Install a window facing south	—	SD
CR9	TC9	Dimension of one side: minimum 2.5m	▲	SD
CR10	TC10	Install a fireplace	—	DD
CR11	TC11	Install a fan or a small window	—	SD
CR12	TC12	Install a fan or a small window	—	SD
CR13	TC13	Total base cabinet length: minimum 48 inch	▲	DD
CR14	TC14	Add walk-in closet	—	SD
CR15	TC15	minimum size: 80 sqft	▲	SD

▲: Value to be increased

▼: Value to be decreased

—: Value to be attained

SD: Schematic Design

DD: Design Development

### Step 3: Build the roof of the HoQ (Correlation matrix)

The roof of HoQ is to evaluate the correlations between technical characteristics. The roof enables the design team to recognize how technical characteristics are interdependent in the early design phase. For instance, installing a fireplace (TC10) and installing a fan or a small window (TC11) can create conflicting demands on the design and functionality, in relation to heating efficiency and air quality. The open window can lead to increased heat loss, when the fireplace is supposed to bring heat into living areas. Also, if the airflow from the powder room's ventilation window

### Step 4: Determine the relationship between WHATs and HOWs

45

Table 4-3. Relationship matrix between customer's requirements and technical characteristics

CR <sub>i</sub>	RI	TC <sub>i</sub>														
		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
CR1	5	◆														
CR2	3		◆	◆	◇		◇									
CR3	4			◆	◇		◇									
CR4	5			◇	◆											
CR5	2					◆	◆									
CR6	5						◆									
CR7	3							◆		◆						
CR8	5								◆	◆						
CR9	2									◆	◆					
CR10	1										◆					
CR11	1										◆	◆				
CR12	3												◆			
CR13	2													◆		
CR14	4														◆	
CR15	3															◆

◆ Strong relationship: 3 points    ◆ Medium relationship: 2 points    ◇ Low relationship: 1 point

### Step 5: Calculate importance weight and relative weight of requirements

The importance weight for each technical characteristic is calculated as follows:

$$IW_t = \sum_{i=1}^N RI_i * SR_{it}$$

Where,

$IW_t$ : The importance weight of the technical characteristic t ( $TC_t$ )

$RI_i$ : The relative importance of the customer's requirements



$SR_{it}$ : The strength of relationship between the customer's requirement  $i$  and the technical characteristic  $t$

$N$ : The number of technical characteristics

The calculated importance weight and relative weight are used to rank the technical characteristics.

All or some of these are to be considered as objectives for the optimization.

Table 4-4. Importance weights and ranks of each technical characteristic

	<b>Technical Characteristics (TC<sub>i</sub>)</b>														
	<b>1</b>	<b>2</b>	<b>3</b>	<b>4</b>	<b>5</b>	<b>6</b>	<b>7</b>	<b>8</b>	<b>9</b>	<b>10</b>	<b>11</b>	<b>12</b>	<b>13</b>	<b>14</b>	<b>15</b>
<b>IW</b>	15	9	23	22	6	26	9	15	29	7	3	9	6	12	9
<b>RW</b>	8%	5%	12%	11%	3%	13%	5%	8%	15%	4%	2%	5%	3%	6%	5%
<b>Rank</b>	5	8	3	4	13	2	8	5	1	12	15	8	13	7	8

Figure 4-2 shows the completed HoQ. In this case study, six technical characteristics were ranked the highest five to be achieved. Among these six technical characteristics, five are selected to be achieved and become fitness functions in the generative design process which covers schematic design phase. These five characteristics are:

TC1: One separate entrance to backyard

TC4: Add walk-in pantry

TC6: Open layout with integrated living, dining, and kitchen areas

TC8: Install a window facing south

TC9: Dimension of one side: minimum 2.5m

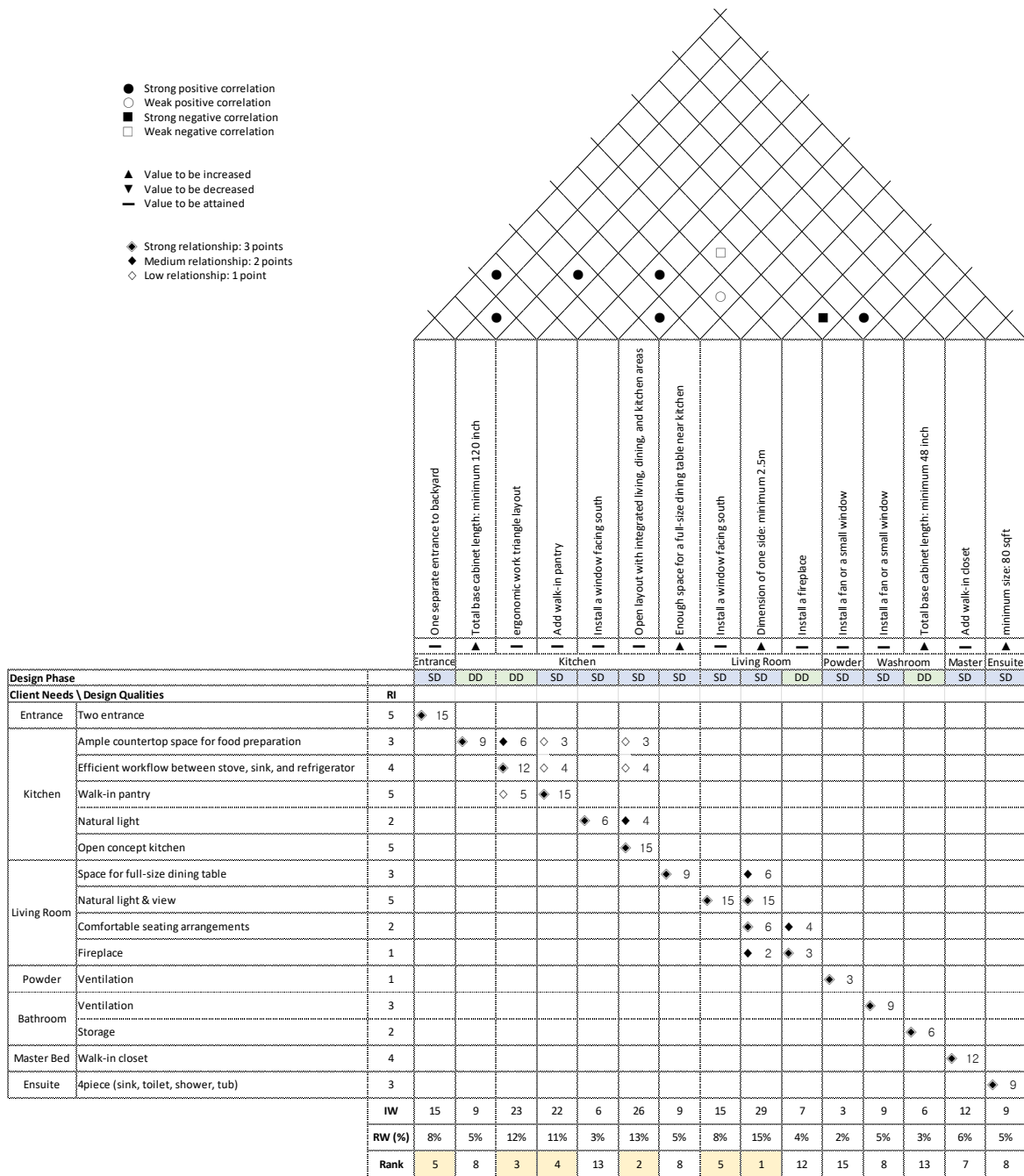


Figure 4-2. House of Quality for the case study

### 4.1.2 Geometric and topological requirements

When designing a single detached house, various geometric constraints must be considered, including the dimensions of the lot and each room. In this case study, the lot dimension is set as one of the constraints within the generative design process. This constraint is handled differently depending on the method: in the bottom-up approach, penalties are applied when this constraint is violated; in the top-down approach, it is set as an input boundary. The dimensions of the rooms are defined as input parameters that can meet requirements through fixed dimensions, fixed dimensions with a 90-degree rotation, or variable dimensions adjusted using an integer slider within a specified range (Figure 4-3). TC9, living room having any dimensions longer than 2.5m, is achieved by setting the minimum value of the integer slider as 2,500.

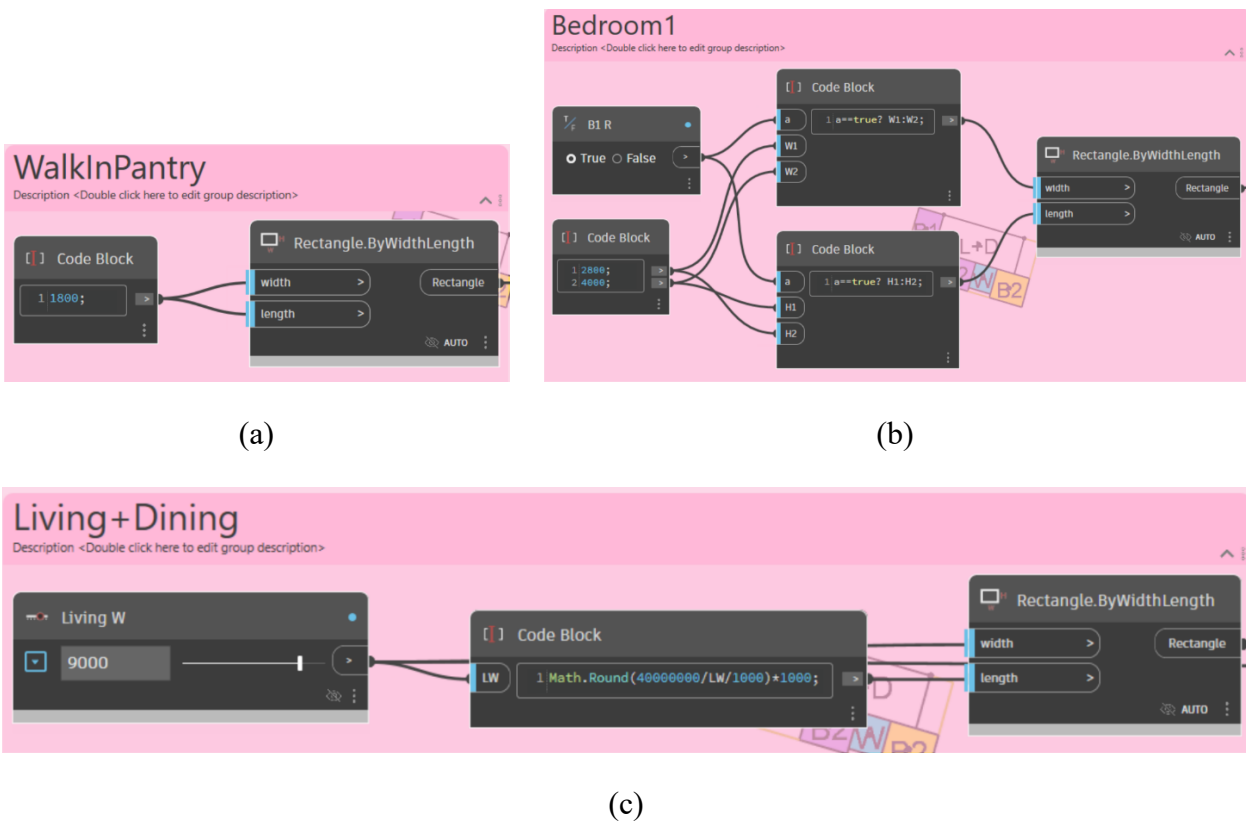
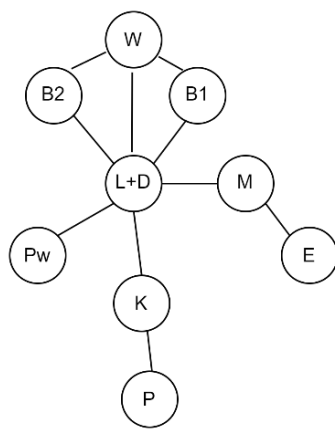


Figure 4-3. Dynamo Script for Room Dimension Configurations: (a) fixed dimension, (b) fixed dimension with 90-degree rotation, (c) variable dimensions adjusted using slider

Connectivity is the main topological constraint in this case study and is approached differently in the bottom-up and top-down methods. The connections between rooms are the input constraint in the bottom-up method while in the top-down method, they are evaluation parameters with penalties applied for non-compliance. Figure 4-4 shows the graph of this case study layout. TC4, adding a walk-in-pantry, is achieved in this stage by making a connection between the kitchen and the walk-in-pantry.



<b>L+D</b>	Living + Dining
<b>B1</b>	Bedroom 1
<b>B2</b>	Bedroom 2
<b>W</b>	Washroom
<b>M</b>	Master Bedroom
<b>E</b>	Ensuite
<b>Pw</b>	Powder Room
<b>K</b>	Kitchen
<b>P</b>	Walk-In-Pantry

Figure 4-4. Justified Plan Graph (JPG) and the room data of the case study

## 4.2 GD: Bottom-up method

### 4.2.1 Dynamo script

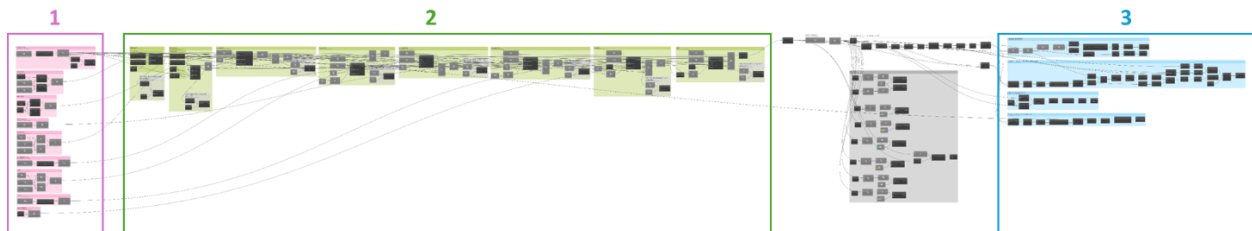


Figure 4-5. Overview of the Dynamo script for the bottom-up method 1) Space generation section, 2) Space aggregation section, 3) Fitness calculation section

Figure 4-5 shows the entire Dynamo script for generating and evaluating the layout using the bottom-up method. The process starts with creating all spaces as rectangles with dimension constraints Figure 4-5(1). The main generation sequence, depicted in Figure 4-5(2), illustrates the aggregation of each space, and the fitnesses of the generated design are calculated in the third section Figure 4-5(3).

#### (1) Space generation

As previously described in section 494.1.2, all spaces are generated as rectangles with either fixed dimensions, fixed dimensions with possible rotation, or variable dimensions.

#### (2) Space aggregation

Starting from L+D, which has the most connections to other rooms, the sequence of space aggregation follows with B1, W, B2, M, E, Pw, K, and finally P. The aggregation using the bottom-up method described in Section 3.4.1 utilizes a variety of Dynamo nodes. Thus, organizing these into custom nodes for packaging significantly helps tidy the Dynamo workspace, and more importantly facilitates reuse across projects.

#### (3) Fitness calculation

In addition to TC4 and TC9 which were already achieved in the process of space generation and aggregation, five more objectives are translated into fitness functions. These functions are used to score or penalize outcomes, guiding the optimization process. For critical requirements, if a design outcome fails to meet a specified threshold, it receives a severe penalty of -10,000 to ensure it is readily excluded (Figure 4-6(a)). For less critical requirements with thresholds, the scoring is adjusted linearly: outcomes meeting or exceeding the threshold receive positive scores or a fixed value of 0, while those that fall short receive progressively decreasing negative scores (Figure

4-6(b)-(c)). For example, if the minimum required length is 10, a length of 15 scores 15, whereas a length of 8 scores -2. The detail of this process is described in the next section, 4.2.2 Details on Fitness Calculation.

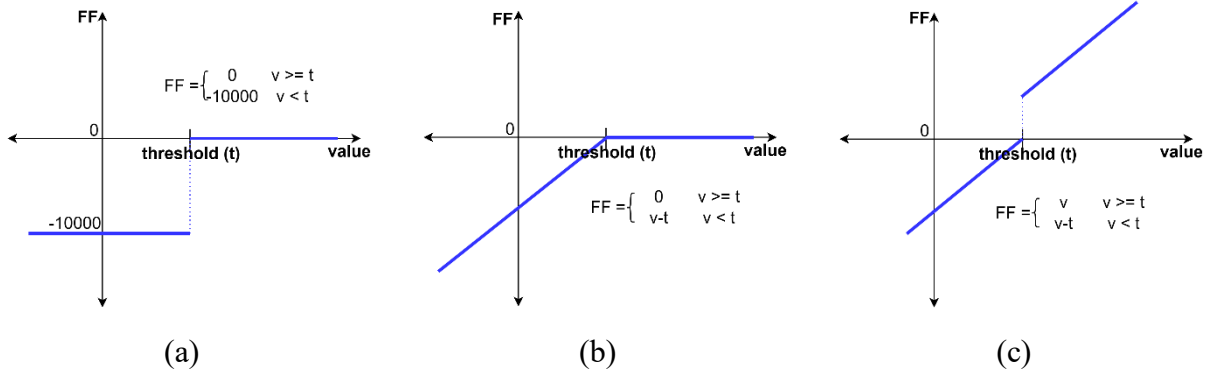


Figure 4-6. Fitness functions for different objectives

The threshold value for each fitness function can be decided by the given conditions of the design problem or by decision makers. The strength of score or penalty can be adjusted according to the particular design problem and its objectives.

## 4.2.2 Details on fitness calculation

### Objective 1: Compactness 1

The first objective is to ensure the compactness of the layout to conform to the lot dimensions of 13,500 mm X 18,000 mm. In the fitness function, this is quantified by evaluating the width and length of the bounding box—a rectangular border enclosing the entire layout—to ascertain how well the layout fits within the specified lot size.

$$\text{Fitness Function (FF)} = \min (A, B)$$

where,

$$A = \begin{cases} -1 \times \max(b_w - 13500, b_l - 13500) & \text{if } b_w > 13500 \wedge b_l > 13500 \\ 0 & \text{otherwise} \end{cases}$$

$$B = \begin{cases} -1 \times \max(b_w - 18000, b_l - 18000) & \text{if } b_w > 18000 \vee b_l > 18000 \\ 0 & \text{otherwise} \end{cases}$$

$b_w$  = width of the bounding box

$b_l$  = length of the bounding box

### **Objective 2: Compactness 2**

This is an additional objective to achieve compactness of the structured layout that complements objective 1. The corresponding fitness function aims to achieve the total number of outer perimeter curves being less than or equal to 30, resulting in rooms being positioned closer to the centroid rather than being sparsely placed.

$$\text{Fitness Function (FF)} = \begin{cases} 0 & n \leq 30 \\ 30 - n & n > 30 \end{cases}$$

$n$  = number of outer perimeter curves

### **Objective 3: One separate entrance to backyard (TC1)**

To quantify the objective of having one separate entrance to backyard, the length of the wall on the north side of the living room or kitchen — whichever is closer to the north assuming the backyard is on the north — must be calculated to be at least 1,000 mm, which is the door size. In this context, the north direction is defined as having a higher y-coordinate.

$$\text{Fitness Function (FF)} = \begin{cases} 0 & \text{if } w \geq 1000 \\ -10000 & \text{otherwise} \end{cases}$$

where,

$$w = \begin{cases} w_l & \text{if } y_l > y_k \\ w_k & \text{otherwise} \end{cases}$$

$y_l$  = y-coordinate of the center of the living room

$y_k$  = y-coordinate of the center of the kitchen

$w_l$  = length of the north-facing wall of the living room uninterrupted by other rooms

$w_k$  = length of the north-facing wall of the kitchen uninterrupted by other rooms

#### **Objective 4: Open layout with integrated living, dining, and kitchen areas (TC6)**

An open layout floorplan requires multiple considerations such as area continuity, visibility, accessibility, and spatial configurations. This study, which focuses on the schematic design phase, assesses the spatial configuration of the living room and kitchen, where fewer surrounding walls indicate an open layout. The fitness function calculates the score based on the length of the shared edge between the living room and kitchen, proportionally adjusting the score based on whether this length meets or falls short of a 2,000 mm threshold.

$$\text{Fitness Function (FF)} = \begin{cases} l & l \geq 2000 \\ l - 2000 & l < 2000 \end{cases}$$

$l$  = length of the shared edge between the living room and the kitchen

#### **Objective 5: Install a window facing south (TC8)**

The south-facing wall of the living room is determined by identifying walls that run along the x-axis with a lower y-coordinate. The maximum length of the south-facing wall uninterrupted by other rooms is calculated by deducting the widths of any blocking rooms from the total width of the living room, as previously described in Section 3.5. If this length is less than the threshold of 1,200 mm, the function returns a negative score.

$$\text{Fitness Function (FF)} = \begin{cases} w_{ls} & l \geq 1200 \\ w_{ls} - 1200 & l < 1200 \end{cases}$$

$w_{ls}$  = length of the south-facing wall of the living room uninterrupted by other rooms

#### **4.2.3 Generative design results**

The generative design process was executed, using NSGA-II for multi-objective optimization, with a population size of 200 and 20 generations, targeting the maximization of all fitness functions (goals).





Table 4-5. Scores on each objective of each design outcome

No.	Compactness 1	Compactness 2	TC1	TC6	TC8
1	0	0	0	4000	7545
2	0	-2	0	4180	8000
3	0	0	0	7000	2170
4	0	-2	0	4180	8000
5	0	0	-10000	3360	8000
6	0	-2	0	7000	5000
7	0	0	0	7000	2170
8	0	-2	0	4180	8000
9	0	0	0	2370	9000
10	0	0	0	6000	7000
11	0	-2	-10000	3380	9000
12	0	0	0	6000	7000
13	0	0	0	6580	2800
14	0	-2	-10000	4385	8000
15	0	0	0	6000	7000
Min	0	-2	-10000	2370	2170
Max	0	0	0	7000	9000

One method for users to evaluate outcomes is based on scores from individual objectives, tailored to their priorities. Multi-objective optimization provides a decision-maker with a range of optimal solutions, allowing for flexible selection based on specific preferences or additional analysis. Objectives 1, 2, and 3 (Compactness 1, Compactness 2, and TC1) act as necessary filters to exclude unviable conditions, ensuring only feasible design solutions are considered for evaluation. Objectives 4 and 5 (TC6, TC8) are measured by length metrics without weights, allowing these metrics to be optimized independently. This approach prevents the optimization engine from favoring solutions that might score higher on these specific metrics but are less effective overall.

Another method to evaluate the outcomes is by calculating overall scores through normalization, adjusting the scores of various objectives to a common scale and aggregating them into a single comprehensive score. This normalized scoring allows for a balanced comparison across diverse design solutions, highlighting the most effective designs by considering all objectives equally. Among the 15 outcomes, No. 1, 12 are selected for comparison using normalization (Figure 4-8).

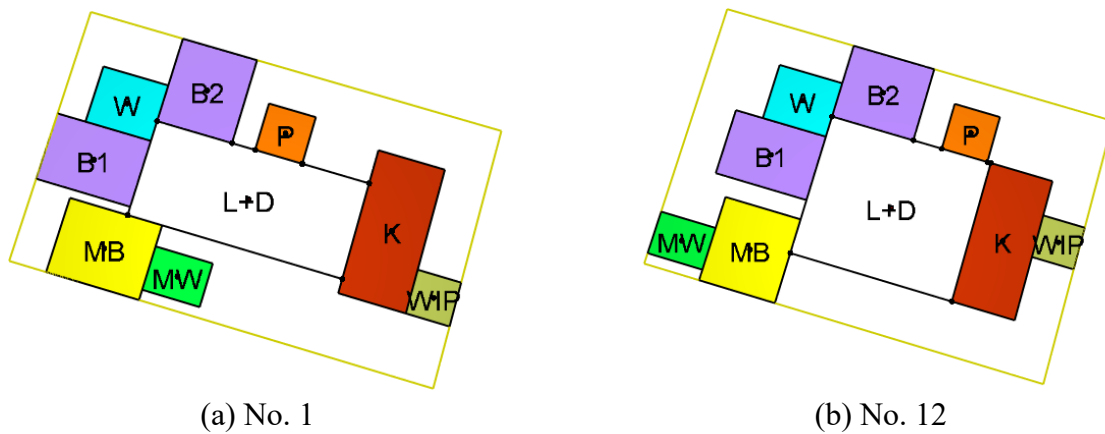


Figure 4-8. Top 2 solutions selected for comparison using normalization

Table 4-6 shows the normalized scores for each objective of solution No. 1 and No. 12. Assuming equal weighting for all objectives, the overall score for each solution is calculated by averaging the normalized scores. Solution No. 12, having a higher overall score than No. 1, is advanced to the next phase of the process, which is its conversion into BIM elements.

Table 4-6. Normalized scores for each objective of solution No. 1 and No. 12

No.	Compactness 1	Compactness 2	TC1	TC6	TC8	Overall
1	-	1	1	0.352	0.787	0.785
12	-	1	1	0.784	0.707	0.873

## 4.3 GD: Top-down method

### 4.3.1 Dynamo script

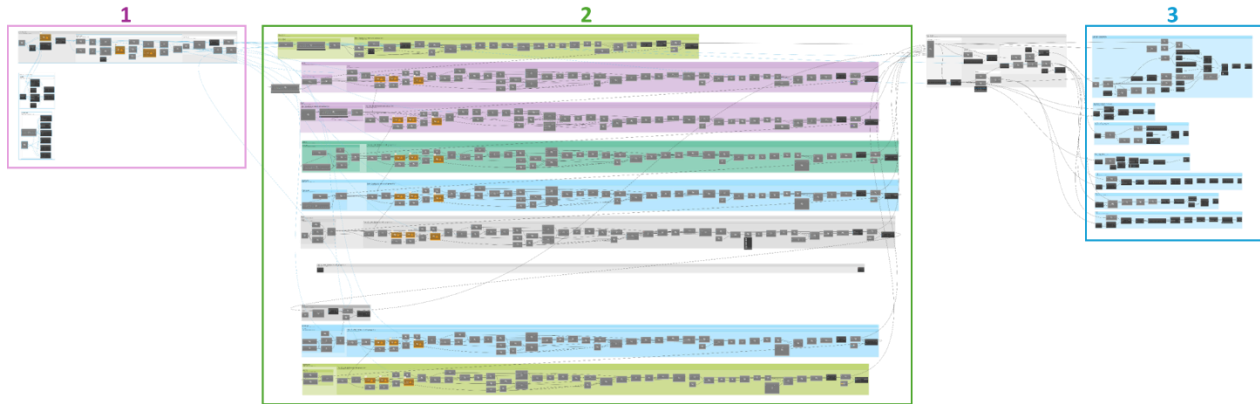


Figure 4-9. Overview of the Dynamo script for the top-down method

1) Grid system generation section, 2) Space allocation section, 3) Fitness calculation section

The Dynamo script for generating a layout using the top-down method is presented in Figure 4-9. The first section is for generating a grid system within the designated house boundary for both the first and second floors (Figure 4-9(1)). Each space is then allocated within the boundary according to the designated floor (Figure 4-9(2)), and the resulting design is evaluated using the defined fitness functions from the requirement (Figure 4-9(3)).

#### (1) Grid system generation

Within the defined house boundary dimensions of 15,000 X 6,000 mm, a grid with 200 mm spacing is created. These points on the grid serve as coordinates for positioning rectangular spaces that represent different rooms of the house. The dimensions of the rooms are multiples of 400 mm to ensure that they can neatly align with the grid's structure without partial overlaps or misalignments. In this generation process, users have the flexibility to adjust the density of the grid system according to their specific design needs and preferences. A denser grid offers finer control over the placement of spaces, allowing for a greater variety of spatial configurations. However,

increasing the density of the grid also leads to a higher computational cost due to the increase of potential placement points, which expands the search space.

## (2) Space allocation

The next process is to allocate space within the predefined boundary. On the second floor, bedroom1, bedroom2, washroom, master bedroom, and ensuite (B1, B2, W, M, and E) are placed, and the rest of the area serves as corridor or resting area. Powder room and walk-in-pantry (P and WIP) are placed on the first floor, the remaining area being an integrated space that combines kitchen and living + dining (K and L+D).

## (3) Fitness calculation

In the top-down method, unlike the bottom-up method, the input constraints are the geometric boundaries of the layout rather than topological requirements. Thus, it's necessary to integrate additional fitness functions to guide the optimization process using the genetic algorithm to achieve topological requirements of the layout. Details are described in the next section, 4.3.2 Details on Fitness Calculation.

### **4.3.2 Details on fitness calculation**

#### **Objective 1: Second floor corridor connectivity**

The objective “corridor connectivity” ensures that the rest of the area after placing all rooms, which becomes the corridor or resting area, are continuous without any room obstructing accessibility. The fitness function applies a significant penalty in different scenarios of generating designs that block these pathways. The first scenario is when there is more than one enclosed space after all allocations are complete. The second scenario is when not all spaces can be placed due to the configuration of previously placed spaces creating infeasible dimensions for subsequent

placements. Additionally, a situation may occur where all other conditions are met but the corridor does not share any edge with other rooms meaning no access to them.

$$\text{Fitness Function (FF)} = \begin{cases} -10000 & s \geq 2 \text{ or } n \neq r \text{ or } e = \text{false} \\ 0 & \text{otherwise} \end{cases}$$

$s$  = number of remaining areas after allocation is finished

$n$  = number of spaces to be allocated

$r$  = number of spaces after allocation is finished

$e$  = Boolean whether the corridor share any edge with all rooms

### **Objective 2: Master bedroom-ensuite connectivity**

Another topological requirement, the connectivity between master bedroom and ensuite, is determined by examining if master bedroom and ensuite share an edge. If they do, the length of the shared edge must be sufficient to accommodate a door.

$$\text{Fitness Function (FF)} = \begin{cases} 0 & l \geq 900 \\ -10000 & l < 900 \end{cases}$$

$l$  = length of the shared edge between M and E

### **Objective 3: First floor connectivity**

Similar to the corridor connectivity objective, the remaining area of the first floor, after placing powder room and walk-in-pantry, must be continuous. If more than one enclosed area remains after allocation or if the narrowest pathway is less than 800 mm in width, the design is penalized.

$$\text{Fitness Function (FF)} = \begin{cases} -10000 & s \geq 2 \\ -10000 & w < 800 \\ 0 & \text{otherwise} \end{cases}$$

$s$  = number of remaining areas after allocation is finished

$w$  = minimum width of the pathway

#### Objective 4: Plumbing adjacency

Another topological requirement added is the adjacency of water-using rooms, washroom, ensuite, and powder room. This ensures that these rooms are positioned close to each other, optimizing plumbing layouts and reducing the distance that water and waste pipes need to travel. The lengths between the three rooms are summed and evaluated.

$$\text{Fitness Function (FF)} = \text{Sum} (d_{WE}, d_{EP}, d_{PW})$$

$d_{WE}$  = Distance between washroom and ensuite

$d_{WE}$  = Distance between ensuite and powder room

$d_{WE}$  = Distance between powder room and washroom

#### Objective 5: One separate entrance to backyard (TC1)

The technical characteristic of placing one separate entrance to backyard is determined by calculating the length of the north-facing boundary wall unobstructed by other rooms. If this length exceeds 1,000 mm, it is deemed to satisfy the condition for having a separate entrance.

$$\text{Fitness Function (FF)} = \begin{cases} 0 & \text{if } w \geq 1000 \\ -10000 & \text{otherwise} \end{cases}$$

$w$  = length of the north-facing wall of the boundary wall uninterrupted by other rooms

#### Objective 6: Open layout with integrated living, dining, and kitchen areas (TC6)

Designing an open layout floorplan focuses on the spatial relationship between the living room and kitchen, with fewer walls indicating a more open layout. The fitness function evaluates the length of the narrowest width of the remaining area at the approximate demarcation between these two spaces.

$$\text{Fitness Function (FF)} = \begin{cases} l & l \geq 2000 \\ l - 2000 & l < 2000 \end{cases}$$

$l$  = length of the narrowest width of the remaining area

### Objective 7: Install a window facing south (TC8)

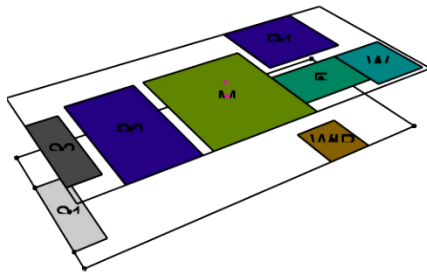
Design with longer south-facing boundary exposed to the outside unobstructed by other scores higher than others.

$$\text{Fitness Function (FF)} = \begin{cases} w_{ls} & l \geq 1000 \\ w_{ls} - 1000 & l < 1000 \end{cases}$$

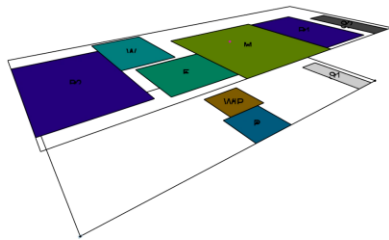
$w_{ls}$  = length of the south-facing boundary wall uninterrupted by other rooms

### 4.3.3 Generative design results

Figure 4-10 shows high-scoring design results and corresponding scores on each objective.



Objective 1	0
Objective 2	0
Objective 3	0
Objective 4	50
Objective 5	0
Objective 6	4400
Objective 7	6400



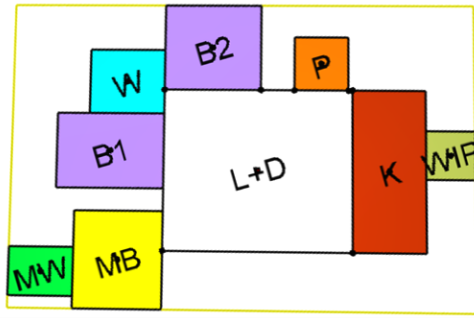
Objective 1	0
Objective 2	0
Objective 3	0
Objective 4	78
Objective 5	0
Objective 6	2400
Objective 7	6400

Figure 4-10. High-scoring design results from the top-down method

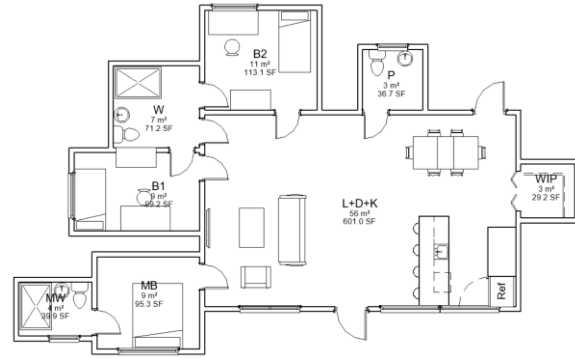
### 4.4 Post-GD: Conversion of geometries into BIM elements

The final phase involves converting the geometries generated in Dynamo into Building Information Modelling (BIM) elements within Revit. In this case study, transforming lines into walls was performed using the Dynamo Node “Wall.ByCurveandLevels” which takes lines, start level, end level, and wall type as inputs.

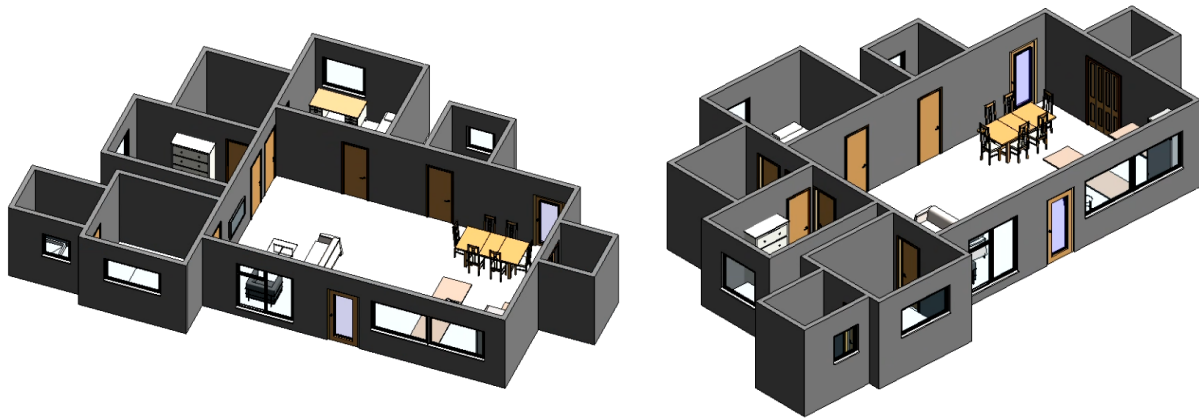




(a) Floor plan as geometric representation



(b) Floor plan after conversion

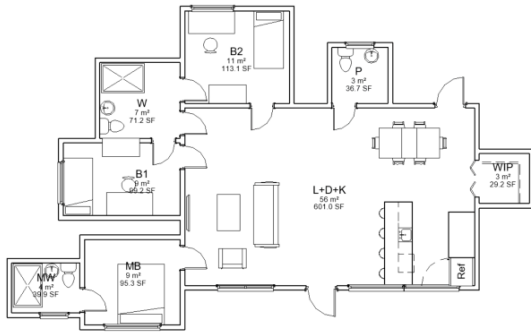


(c) 3D views of the floor plan

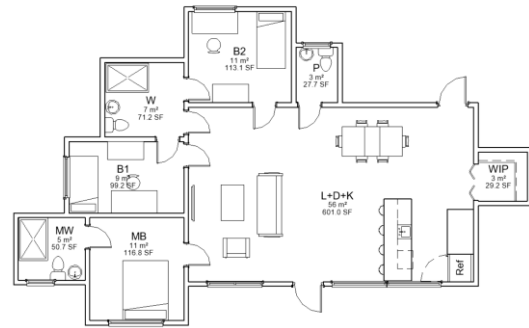
Figure 4-11. Conversion of selected design into BIM model

Figure 4-11(b) illustrates the transformed floor plan with placement of doors, windows, and furniture to visually aid client. Some non-optimal aspects were observed, such as the master bedroom being too far from bedroom 1, creating dead space between them, and a similar issue with the powder room. This highlights that direct transformation without human intervention may not always yield the optimal solution. The designer's expertise and experience are essential for refining the outcome.

Figure 4-12(b) shows the floor plan after modification. The dimensions and configuration of Master bedroom, ensuite, and powder room have been adjusted to remove unnecessary gaps between spaces and enhance compactness.



(a) Floor plan without modification



(b) Floor plan after modification



(c) 3D view of the modified floor plan

Figure 4-12. Floor plan and 3D view after modification

Same procedures, converting and refining, are processed for the selected design outcomes from the top-down method (Figure 4-13).



Figure 4-13. Floor plan and 3D view after conversion and modification

## 4.5 Discussion

### 4.5.1 Sensitivity analysis: variant parameter setting of genetic algorithm

Population size and generation number are important parameters in evolutionary algorithms. Population size refers to the number of individual solutions and it functions as a key parameter that influences competition and diversity within the algorithm, ensuring that only the fittest solutions persist over generations, and generation represents a single cycle of selection, crossover, and mutation within the population (Eiben & Smith, 2015). The impact of those parameters on the performance of evolutionary algorithms has been studied by researchers. Diaz-Gomez & Hougen (2007) explored the significance of selecting an initial population and found that the diversity of

the population impacts the quality of the solutions and computational efficiency. Several factors are considered when generating an initial population such as search space, fitness function, and problem difficulty. Tsoy (2003) found that larger population size with few generations generally perform better than smaller populations. The study revealed that while larger populations can enhance the search process, overly large groups might complicate computation. The research noted that beyond a certain number of generations, improvements become low, suggesting that increasing population size could be more effective than extending generations. In addition to attempts to systematically tune the parameters for the best ones, still many researchers select heuristic approaches to find them as they are problem specific.

When running the Generative Design with optimization solver, population size and generation number is set by the user while crossover and mutation is set by default as 0.8 and 0.4 respectively. To understand the impact of varying generation and population size numbers on the outcomes of the generative design process, a sensitivity analysis was conducted out of the case study using the bottom-up method. This involved changing these parameters to observe the resulting variations in design solutions. By analyzing how different settings influence the quality and diversity of generated designs, this analysis is intended to identify optimal parameter configurations.

A total of 22 generative processes were conducted, with generation numbers set at 10 and 20, and population sizes ranging from 100 to 300 in increments of 20. The scores for each objective were normalized using the global maximum and minimum score across all sets, and then averaged within each set for comparative analysis of the 22 sets.

The normalization process involves the following steps:

Step 1: Identify the global minimum and maximum values for each objective across all sets. Taking the values from the entire dataset across all sets and outcomes rather than within individual outcomes or runs ensures that the normalization scale is consistent across all data points, which is crucial for fair comparison between different runs with potentially different ranges of scores.

Step 2: Apply the normalization formula to each score in every set. This transforms each score to a range between 0 (corresponding to the global minimum) and 1 (corresponding to the global maximum).

$$\text{Normalized Score} = \frac{\text{Original Score} - \text{Global Min}}{\text{Global Max} - \text{Global Min}}$$

Step 3: Calculate the average normalized score for each outcome within a set. This is done by averaging the normalized scores of all objectives for each individual outcome.

Step 4: Determine the overall average score for each set by averaging the average normalized scores of all outcomes within that set.

Figure 4-14 and Figure 4-15 display performance scores derived from the normalized data with varying population sizes and generation numbers. Contrary to expectations, increasing the population size did not enhance outcome quality. Linear trendline analysis yielded R-squared values of 0.0007 and 0.0044 for generation numbers 10 and 20, respectively, indicating no significant linear relationship. Similar results were observed with various trendline options. Likewise, increasing the generation number did not improve performance. Figure 4-16 compares the scores for each population size at generation numbers 10 and 20.

Population Size	Performance Score
G10P100	0.751409
G10P120	0.435709
G10P140	0.756421
G10P160	0.376427
G10P180	0.81466
G10P200	0.451212
G10P220	0.758152
G10P240	0.392438
G10P260	0.714115
G10P280	0.43681
G10P300	0.749385

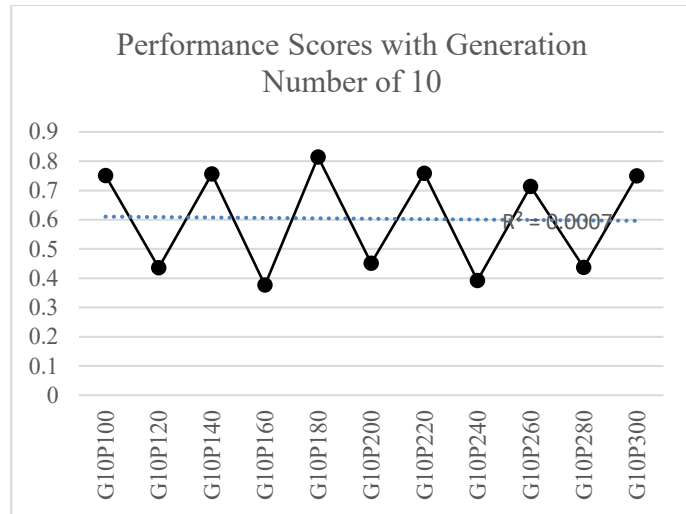


Figure 4-14. Performance scores for each population size with generation number 10

Population Size	Performance Score
G20P100	0.501022
G20P120	0.897396
G20P140	0.305216
G20P160	0.830474
G20P180	0.371314
G20P200	0.784159
G20P220	0.394266
G20P240	0.83085
G20P260	0.832938
G20P280	0.757212
G20P300	0.39396

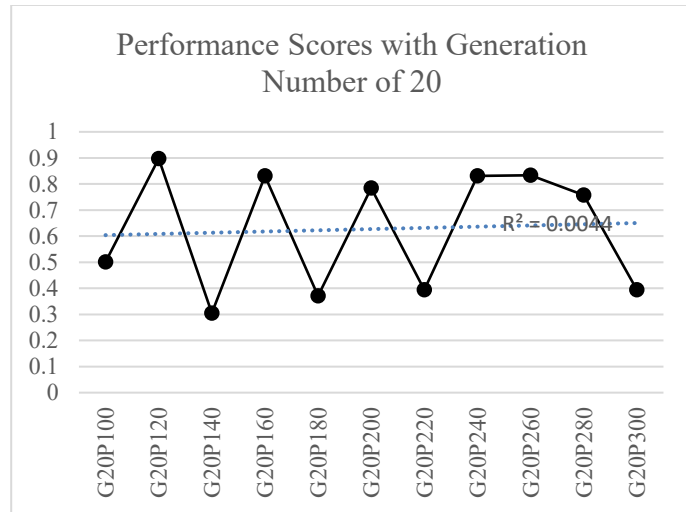


Figure 4-15. Performance scores for each population size with generation number 20

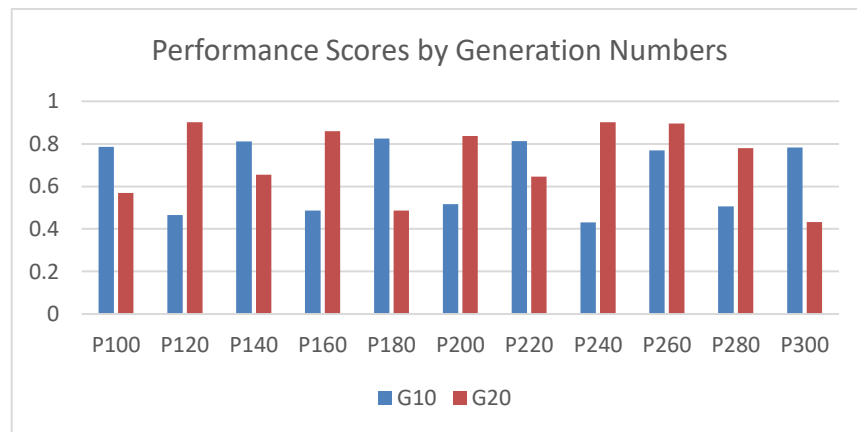


Figure 4-16. Performance scores by generation numbers

There can be multiple reasons for the unexpected results. One possible reason is that the population size is not large enough. If the population size is too small, there may be lack of diversity resulting in premature convergence (Zhang et al., 2009). This can lead to poor quality of final outcomes because it causes the algorithm to settle on suboptimal solutions early on, without sufficiently exploring the solution space for potentially better solutions. Finding the optimal solution is guaranteed if the computer explores and evaluates all possible solutions. This problem-solving technique, which validates every potential candidate, is known as brute-force search or exhaustive search in computer science. However, this case study involves an extraordinarily high number of possible solutions, totaling 144,355,958,537,895,143,424. These are derived from 23 input variables representing the dimensions and locations of each room, each having multiple steps ( $2^6 \times 4 \times 6^3 \times 7 \times 11^5 \times 21^6 \times 27$ ). If we assume that the computer can process one solution in one second, more than 4 trillion years will take to explore all possible solutions, making this option meaningless. Thus, it is very important to find the optimal value for the population size, not too small to not reach the optimal solution and not too large to cost too much computational time and cost.

Another possible reason for the result can be the other parameters in genetic algorithm, crossover rate and mutation rate. Crossover is a process where segments of chromosome from two parent solutions are exchanged to produce new offspring to allow 'good' solutions to potentially develop and mutation is a process making random changes to the chromosome of a solution to maintain diversity within a population of solutions and prevent premature convergence (M. Srinivas & Patnaik, 1994). The crossover rate and mutation rate are set as default as 0.8 and 0.4 respectively in the Autodesk Generative Design as of version 23.2.19.0. As the combination of these parameters

along with population size and generation affects the performance of the genetic algorithm, more sophisticated approaches to find the best combination are required.

The combination of population size of 120 and generation number 20 generated the highest performance scores among all sets. Figure 4-17 displays a few of generative design outcomes from this set. The figure includes both floor plans and 3D views following the conversion and adjustment.

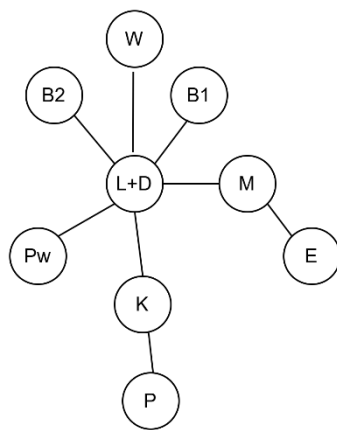


Figure 4-17. High-scoring designs and their floor plan and 3D view



#### 4.5.2 Sensitivity analysis: reduced connectivity constraints

In this section, the implications of relaxed connectivity constraints within the bottom-up approach are explored. The sensitivity analysis conducted aims to assess how these modifications influence the spatial configuration and overall design outcomes. Figure 4-18 shows the justified plan graph (JPG), which has fewer connectivity constraints compared to those presented in the case study of Section 4.1.2. Connections between B1 and W as well as B2 and W were removed.



<b>L+D</b>	Living + Dining
<b>B1</b>	Bedroom 1
<b>B2</b>	Bedroom 2
<b>W</b>	Washroom
<b>M</b>	Master Bedroom
<b>E</b>	Ensuite
<b>Pw</b>	Powder Room
<b>K</b>	Kitchen
<b>P</b>	Walk-In-Pantry

Figure 4-18. Justified Plan Graph (JPG) with reduced connectivity constraints

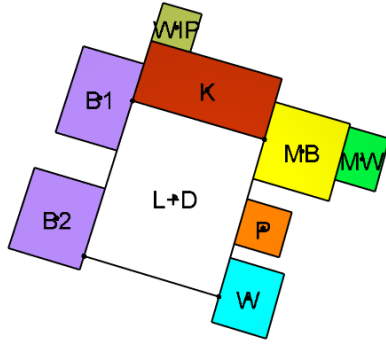
The generative design process generated 15 design outcomes with the population size and generation of 120 and 20, respectively. The scores for each objective across all design outcomes are shown in Table 4-7. Design outcomes were compared using normalized scores, as detailed in Section 4.2.3. Figure 4-19 and Table 4-8 illustrate the geometric layouts and corresponding scores of the best and worst-performing design outcomes.

Table 4-7. Scores for each objective across all design outcomes

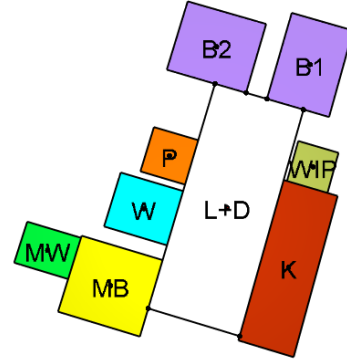
No.	Compactness 1	Compactness 2	TC1	TC6	TC8
1	0	-6	0	8000	5000
2	0	-6	0	7000	6000
3	0	-6	0	7000	6000
4	0	0	0	6000	6000
5	0	-2	0	3570	7000
6	0	-6	-10000	4000	6815
7	0	-6	0	7000	6000
8	0	-6	0	7000	6000
9	0	-2	0	3570	7000
10	0	0	0	6000	6000
11	0	-6	0	7000	6000
12	0	-6	0	9000	-15
13	0	-6	0	7000	6000
14	0	-6	0	8000	5000
15	0	-6	0	8000	5000
Min	0	-6	-10000	3570	-15
Max	0	0	0	9000	7000

Table 4-8. Best and worst scored design outcomes

No.	Compactness 1	Compactness 2	TC1	TC6	TC8	Overall
4	-	1	1	0.448	0.857	0.861
6	-	0	0	0.079	0.974	0.411



(a) No. 4



(b) No. 6

Figure 4-19. Best and worst scored design outcomes

Even though the parameter settings of population size of 120 and generation number of 20 has yielded the most optimal outcome in the previous case study, applying the same NSGA-II settings does not guarantee the best results in new scenarios, as the fundamental characteristics of the design problem have changed. Therefore, it is challenging to definitively assess how the relaxed constraints have impacted overall scores and whether their effects are positive or negative. This uncertainty is partly due to the randomness inherent in the genetic algorithm process. However, one clear impact was that many design outcomes negatively scored in compactness 2, which counts the outer perimeter curves, with frequent scores of -6 — a score not observed in the prior case study with stricter connectivity constraints. This suggests that the relaxed constraints increased the overall sparsity of the design, resulting in fewer shared edges between rooms. This effect is evident through both the scored data and visual layouts. Among the five objectives, only compactness was significantly affected by these reduced connectivity constraints, while other objectives, such as optimizing for a south-facing living room or open space, were unaffected.

## 5 Conclusion

### 5.1 Summary

Architectural design challenges are complex, without straightforward solutions, making traditional methods slow and inflexible. The adoption of generative design represents a significant shift in architecture, using advanced computational tools to dynamically explore multiple design options, improving both efficiency and effectiveness. In this respect, this research aimed to develop an end-to-end layout generation process using generative design with 3 stages: pre-generative design, generative design, and post generative design stage.

In the pre-generative design stage of the process, a detailed analysis of client requirements was conducted using the Quality Function Deployment (QFD) methodology and the tool House of Quality (HoQ). This stage focused on translating qualitative client needs into definitive, actionable design objectives. This comprehensive preparation was crucial for establishing fitness functions that would later guide the optimization algorithms in the generative design stage.

For the generative design stage, two distinct algorithms were developed using the visual programming tool Dynamo to initiate the layout generation process: the bottom-up method and top-down method. The top-down method was for generating layouts with predefined boundaries, while the bottom-up method was tailored for scenarios where connectivity between spaces was a critical factor. The layout generated was fed into an evaluation stage where designs were assessed based on fitness functions derived from the initial QFD analysis. This was followed by the iterative generative design phase, utilizing the NSGA-II genetic algorithm to evolve the initial design populations towards the most suitable solutions by optimizing for multiple objectives simultaneously.

The final stage of this study's methodology involved refining the optimized layouts and converting them into Building Information Modelling (BIM) elements. This process facilitated the transformation of generative design outputs, depicted as geometries, into BIM models that are ready for further design development. These BIM models not only support the subsequent design and construction phases but also serve as visual representations for client discussions, enhancing communication and providing a clearer understanding of the proposed designs.

The methodology was applied in the case study of designing a 3bed 2.5bath house. The case study focused on the detailed generation of layouts considering various client-specific requirements such as entrance locations, daylight intake, and spatial relationships. Each generating method approached the layout generation differently, with the top-down method organizing spaces within predefined boundaries and the bottom-up method assembling spaces based on connectivity needs. The generative design process was run multiple times with varying population size and generation numbers to find the effect of them in genetic algorithm.

This research has demonstrated the effectiveness of integrating generative design with QFD to enhance the early stages of architectural design, ensuring designs meet client expectations right from the start. QFD facilitates a deep understanding of client needs, translating these into clear, actionable project objectives. This alignment reduces design revisions and increases efficiency. Generative design harnesses advanced algorithms to explore numerous design possibilities faster than human, in this case study specifically, produced more than 300 outcomes in 2 days over multiple runs. Together, these methodologies significantly refine the conceptual design phase, fostering innovation while closely adhering to client specifications.

## 5.2 Contributions

- This study established a holistic framework that integrates Quality Function Deployment (QFD) and its tool House of Quality (HoQ) with generative design to enhance automation in user-centric architectural layout design.
- Two layout generating algorithms were developed, bottom-up method and top-down method, which are adaptable to different initial settings. The bottom-up method is useful when connectivity between spaces is a priority, making it ideal for complex, non-uniform spaces. On the other hand, the top-down method efficiently manages space allocation within a given boundary, suitable for more uniform layouts such as high-rise buildings, where repeated layouts are common.
- The script was developed to be versatile and functional for various design problems, rather than being limited to a specific problem. Its adaptability allows designers to tackle a wide range of layout challenges.
- The implementation of the NSGA-II genetic algorithm significantly enhanced the architectural design process. By incorporating this multi objectives optimization technique, the research demonstrated the capability to balance and optimize multiple user requirements discovered from the HoQ.

## 5.3 Limitations and future work

- While genetic algorithms can automate and optimize design processes efficiently, they typically focus on predefined, quantifiable objectives and may overlook aspects of human experience or creativity. This necessitated manual modifications to achieve more suitable outcomes. In future work, machine learning techniques can be integrated to incorporate

user feedback, allowing the system to learn from design preferences and produce more contextually appropriate choices.

- The methodology developed in this research is only applicable with rectangular spaces. Future research can expand the algorithm to accommodate more various spatial configurations, enhancing its applicability in diverse architectural scenarios.
- The parameter setting for genetic algorithms, population size and generation, were selected arbitrarily for this study. As these parameters critically influence genetic algorithms' performance, future studies can consider employing a systematic method to establish these settings.

## References

- Autodesk. (n.d.). *Dynamo Dictionary*. <https://Dictionary.Dynamobim.Com/>.
- Bahrehmand, A., Batard, T., Marques, R., Evans, A., & Blat, J. (2017). Optimizing layout using spatial quality metrics and user preferences. *Graphical Models*, 93, 25–38. <https://doi.org/10.1016/j.gmod.2017.08.003>
- Bisht, S., Shekhawat, K., Upasani, N., Jain, R. N., Tiwaskar, R. J., & Hebbar, C. (2022). Transforming an Adjacency Graph into Dimensioned Floorplan Layouts. *Computer Graphics Forum*, 41(6), 5–22. <https://doi.org/10.1111/cgf.14451>
- Chan, L.-K., & Wu, M.-L. (2002). Quality function deployment: A literature review. *European Journal of Operational Research*, 143(3), 463–497. [https://doi.org/10.1016/S0377-2217\(02\)00178-9](https://doi.org/10.1016/S0377-2217(02)00178-9)
- Chatzikonstantinou, I. (2014). *A 3-Dimensional Architectural Layout Generation Procedure for Optimization Applications : DC-RVD*. 287–296. <https://doi.org/10.52842/conf.ecaade.2014.1.287>
- Coello, C. A. C., & Lamont, G. B. (2004). *Applications of Multi-objective Evolutionary Algorithms* (Vol. 1). World Scientific.
- Coello Coello, C. A. (1999). A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, 1(3), 269–308. <https://doi.org/10.1007/BF03325101>



- Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2), 182–197. <https://doi.org/10.1109/4235.996017>
- Delgado-Hernandez, D. J., Bampton, K. E., & Aspinwall, E. (2007). Quality function deployment in construction. *Construction Management and Economics*, 25(6), 597–609. <https://doi.org/10.1080/01446190601139917>
- Diaz-Gomez, P. A., & Hougen, D. F. (2007). Initial Population for Genetic Algorithms: A Metric Approach. *Proceedings of the 2007 International Conference on Genetic and Evolutionary Methods, GEM 2007*, 43–49.
- Dikmen, I., Talat Birgonul, M., & Kiziltas, S. (2005). Strategic use of quality function deployment (QFD) in the construction industry. *Building and Environment*, 40(2), 245–255. <https://doi.org/10.1016/j.buildenv.2004.07.001>
- Eiben, A. E., & Smith, J. E. (2015). *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg. <https://doi.org/10.1007/978-3-662-44874-8>
- Elhegazy, H., Ebid, A., Mahdi, I., Haggag, S., & Abdul-Rashied, I. (2021). Implementing QFD in decision making for selecting the optimal structural system for buildings. *Construction Innovation*, 21(2), 345–360. <https://doi.org/10.1108/CI-12-2019-0149>
- Eltaweel, A., & SU, Y. (2017). Parametric design and daylighting: A literature review. *Renewable and Sustainable Energy Reviews*, 73, 1086–1103. <https://doi.org/10.1016/j.rser.2017.02.011>
- Fargnoli, M., Lombardi, M., Haber, N., & Guadagno, F. (2020). Hazard function deployment: a QFD-based tool for the assessment of working tasks – a practical study in the construction

- industry. *International Journal of Occupational Safety and Ergonomics*, 26(2), 348–369.  
<https://doi.org/10.1080/10803548.2018.1483100>
- Gunduz, M., & Al-Naimi, N. H. (2022). Construction projects delay mitigation using integrated balanced scorecard and quality function deployment. *Engineering, Construction and Architectural Management*, 29(5), 2073–2105. <https://doi.org/10.1108/ECAM-12-2020-1082>
- Guo, Z., & Li, B. (2017). Evolutionary approach for spatial architecture layout design enhanced by an agent-based topology finding system. *Frontiers of Architectural Research*, 6(1), 53–62. <https://doi.org/10.1016/j.foar.2016.11.003>
- Hauser, J. R., & Clausing, D. (1988). The house of quality. *Harvard Business Publishing*, 63–73.
- Holland, J. (1975). *Adaptation in Natural and Artificial Systems*. <https://doi.org/10.1137/1018105>
- Hua, H. (2016). Irregular architectural layout synthesis with graphical inputs. *Automation in Construction*, 72, 388–396. <https://doi.org/10.1016/j.autcon.2016.09.009>
- Hufton+Crow. (n.d.). *Heydar Aliyev Center / Zaha Hadid Architects*.  
<https://www.archdaily.com/448774/Heydar-Aliyev-Center-Zaha-Hadid-Architects>.
- Ilozor, B. D., & Kelly, D. J. (2012). Building Information Modeling and Integrated Project Delivery in the Commercial Construction Industry: A Conceptual Study. *Journal of Engineering, Project, and Production Management*, 2(1), 23–36.  
<https://doi.org/10.32738/JEPPM.201201.0004>
- Iwan Baan. (n.d.). *Heydar Aliyev Center / Zaha Hadid Architects*.  
<https://www.archdaily.com/448774/Heydar-Aliyev-Center-Zaha-Hadid-Architects>.

- John, R., Smith, A., Chotipanich, S., & Pitt, M. (2014). Awareness and effectiveness of quality function deployment (QFD) in design and build projects in Nigeria. *Journal of Facilities Management*, 12(1), 72–88. <https://doi.org/10.1108/JFM-07-2013-0039>
- Ko, J., Ennemoser, B., Yoo, W., Yan, W., & Clayton, M. J. (2023). Architectural spatial layout planning using artificial intelligence. *Automation in Construction*, 154, 105019. <https://doi.org/10.1016/j.autcon.2023.105019>
- Koenig, R., & Knecht, K. (2014). Comparing two evolutionary algorithm based methods for layout generation: Dense packing versus subdivision. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, 28(3), 285–299. <https://doi.org/10.1017/S0890060414000237>
- Koning, H., & Eizenberg, J. (1981). The Language of the Prairie: Frank Lloyd Wright's Prairie Houses. *Environment and Planning B: Planning and Design*, 8(3), 295–323. <https://doi.org/10.1068/b080295>
- Krish, S. (2011). A practical generative design method. *Computer-Aided Design*, 43(1), 88–100. <https://doi.org/10.1016/j.cad.2010.09.009>
- Lee, J. H., Ostwald, M. J., & Gu, N. (2015). A syntactical and grammatical approach to architectural configuration, analysis and generation. *Architectural Science Review*, 58(3), 189–204. <https://doi.org/10.1080/00038628.2015.1015948>
- Marson, F., & Musse, S. R. (2010). Automatic Real-Time Generation of Floor Plans Based on Squarified Treemaps Algorithm. *International Journal of Computer Games Technology*, 2010, 1–10. <https://doi.org/10.1155/2010/624817>

- McKnight, M. (2017). Generative Design: What it is? How is it being used? Why it's a game changer. *KnE Engineering*, 2(2), 176. <https://doi.org/10.18502/keg.v2i2.612>
- Merrell, P., Schkufza, E., & Koltun, V. (2010). Computer-generated residential building layouts. *ACM SIGGRAPH Asia 2010 Papers on - SIGGRAPH ASIA '10*, 1. <https://doi.org/10.1145/1866158.1866203>
- Monedero, J. (2000). Parametric design: a review and some experiences. *Automation in Construction*, 9(4), 369–377. [https://doi.org/10.1016/S0926-5805\(99\)00020-5](https://doi.org/10.1016/S0926-5805(99)00020-5)
- Murata, T., & Ishibuchi, H. (1995). MOGA: multi-objective genetic algorithms. *Proceedings of 1995 IEEE International Conference on Evolutionary Computation*, 289. <https://doi.org/10.1109/ICEC.1995.489161>
- Nagy, D., Lau, D., Locke, J., Stoddart, J., Villaggi, L., Wang, R., Zhao, D., & Benjamin, D. (2017). Project Discover: An Application of Generative Design for Architectural Space Planning. *Proceedings of the 2017 Symposium on Simulation for Architecture and Urban Design (SimAUD 2017)*. <https://doi.org/10.22360/SimAUD.2017.SimAUD.007>
- Nagy, D., & Villaggi, L. (2018, January 10). *Generative Design for Architectural Space Planning*. Autodesk University. <https://medium.com/autodesk-university/generative-design-for-architectural-space-planning-9f82cf5dc0>
- Nagy, D., Villaggi, L., Zhao, D., & Benjamin, D. (2017). *Beyond Heuristics: A Novel Design Space Model for Generative Space Planning in Architecture*. 436–445. <https://doi.org/10.52842/conf.acadia.2017.436>

- Oxman, R. (2017). Thinking difference: Theories and models of parametric design thinking. *Design Studies*, 52, 4–39. <https://doi.org/10.1016/j.destud.2017.06.001>
- Reas, C., & Fry, B. (2007). *Processing: a programming handbook for visual designers and artists* (Vol. 6812). Mit Press.
- Rittel, H. W. J., & Webber, M. M. (1973). Dilemmas in a general theory of planning. *Policy Sciences*, 4(2), 155–169. <https://doi.org/10.1007/BF01405730>
- Saha, N., Haymaker, J., & Shelden, D. (2020). *Space Allocation Techniques (SAT)*. 248–257. <https://doi.org/10.52842/conf.acadia.2020.1.248>
- Schaffer, J. D. (1985). Multiple objective optimization with vector evaluated genetic algorithms. *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*.
- Srinivas, M., & Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4), 656–667. <https://doi.org/10.1109/21.286385>
- Srinivas, N., & Deb, K. (1994). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation*, 2(3), 221–248. <https://doi.org/10.1162/evco.1994.2.3.221>
- Tsoy, Y. R. (2003). The influence of population size and search time limit on genetic algorithm. *7th Korea-Russia International Symposium on Science and Technology, Proceedings KORUS 2003. (IEEE Cat. No.03EX737)*, 3, 181–187 vol.3.

- Wang, X.-Y., Yang, Y., & Zhang, K. (2018). Customization and generation of floor plans based on graph transformations. *Automation in Construction*, 94, 405–416. <https://doi.org/10.1016/j.autcon.2018.07.017>
- Weber, R. E., Mueller, C., & Reinhart, C. (2022). Automated floorplan generation in architectural design: A review of methods and applications. *Automation in Construction*, 140, 104385. <https://doi.org/10.1016/j.autcon.2022.104385>
- Woodbury, R. (2010). *Elements of Parametric Design*. Routledge.
- Yang, X.-S. (2014). Multi-Objective Optimization. In *Nature-Inspired Optimization Algorithms* (pp. 197–211). Elsevier. <https://doi.org/10.1016/B978-0-12-416743-8.00014-2>
- Zhang, G.-L., Xiao-Xia Liu, & Zhang, T. (2009). The impact of population size on the performance of GA. *2009 International Conference on Machine Learning and Cybernetics*, 1866–1870. <https://doi.org/10.1109/ICMLC.2009.5212113>
- Zitzler, E., & Thiele, L. (1999a). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257–271. <https://doi.org/10.1109/4235.797969>
- Zitzler, E., & Thiele, L. (1999b). Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4), 257–271. <https://doi.org/10.1109/4235.797969>